МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ КЫРГЫЗСКОЙ РЕСПУБЛИКИ

ОШСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ МАТЕМАТИКИ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Кафедра «Автоматизированных систем и цифровых технологий»

«Согласовано»
председателем методсовета
факультета МИТ
к.п.н., д.: Д. Зулпукарова

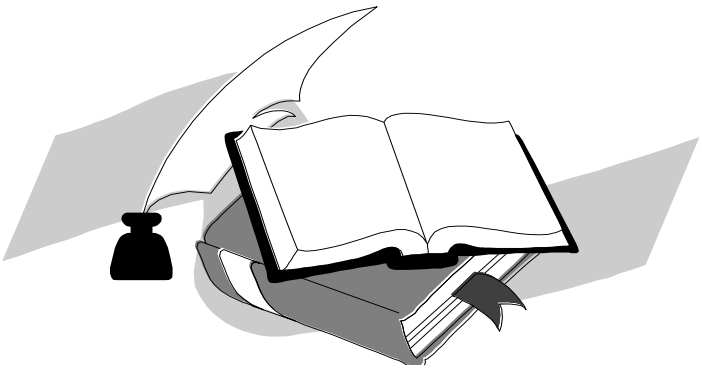
«___» _____2020 г.

«Утверждено»
на заседании кафедры АСЦТ
прот. №1 от «28» августа 2020 г.
Зав. каф. АСЦТ: Молдояров У.

УЧЕБНО-МЕТОДИЧЕСКИЙ КОМПЛЕКС

по дисциплине: **«Технология разработки программного обеспечения»**
для магистрантов очного отделения, обучающихся по специальности:

ИВТ



УМК составлен на основе Государственного образовательного стандарта по направлению 710100 “ИВТ”, академическая степень - магистр “ИВТ” для магистрантов, обучающихся в очном отделении

Составитель, доцент каф. АСЦТ: _____ Беделова Н.С.

2020-2021 учебный год

Сведения о преподавателя

Беделова Нургуль Салибаевна – доцент каф. АСЦТ,

ФМИТ ОшГУ, общий стаж работы – 22 года,

образование – высшее, закончила

физико-математический факультет ОшГУ, 2000 г.

Рабочий телефон: 03222-2-11-85,

Рабочее место: 723500. Главный корпус ОшГУ,

ул. Ленина 331, каб. 224.

Мобильный телефон: 0772-43-73-31

E-mail: nbedelova@oshsu.kg.

I. Аннотация к дисциплине

Целью преподавания дисциплины «Технология разработки программного обеспечения» является изучение методов проектирования и производства программного продукта, принципов построения, структуры и приемы работы с инструментальными средствами, поддерживающими создание программного обеспечения; методов организации работы в коллективах разработчиков программного обеспечения, формирование навыков проектирования, реализации, оценки качества и анализа эффективности программного обеспечения.

Содержание

Внутренняя и внешняя рецензии на рабочую программу

РЕЦЕНЗИЯ

на рабочую программу по дисциплине «Технология разработки программного обеспечения» для магистрантов очного отделения, обучающихся по направлению 710100 «Информатика и вычислительная техника»

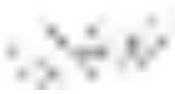
Рабочая программа дисциплины «ТРПО» разработана в соответствии с требованиями ГОС ВПО КР, на основе бюллетеня №19 ОшГУ и ООП специальности по направлению 710100 «Информатика и вычислительная техника».

Рецензируемая программа предназначена для методического обеспечения учебной работы магистрантов очной формы обучения. Содержание представленной на рецензию рабочей учебной программы включает в себя разделы: цели и задачи освоения дисциплины; место дисциплины в структуре ООП; требования к результатам освоения дисциплины; объем дисциплины и виды учебной работы; содержание дисциплины; библиотечно-информационные ресурсы; оценочные средства; материально-техническое обеспечение.

Содержание курса представлено шести разделами, которые в полной мере отражают необходимый объем изучаемого материала. По каждому разделу составлен перечень вопросов, рассмотрение которых позволит сформировать знания, умения и навыки, отвечающие требованиям ГОС ВПО КР.

Информация о видах и объеме учебной работы содержит лабораторных работ и тематику лекционных занятий, призванных помочь магистранту получить теоретические знания и практические навыки создания, внедрения, функционирования, применения технология разработки программного обеспечения, обеспечивающих поддержку работы специалиста.

Программа соответствует всем требованиям бюллетеня №19 ОшГУ и ООП специальности по направлению 710100 «Информатика и вычислительная техника».

Рецензент, к.т.н., доцент по спец.
«Информатика и управление»,
проректор по учебной работе ОшТУ:  Матисаков Ж.К.

РЕЦЕНЗИЯ

на рабочую программу по дисциплине «ТРПО» для магистрантов очного отделения, обучающихся по направлению 710100 «Информатика и вычислительная техника»

Технология разработки программного обеспечения является важнейшим средством при решении многих сложных задач. Именно в связи с этим фактором обучение созданию ТРПО так необходимо для подготовки инженеров-программистов.

Рабочая программа дисциплины «ТРПО» разработана в соответствии с требованиями ГОС ВПО КР, на основе бюллетеня №19 ОшГУ и ООП специальности по направлению 710100 «Информатика и вычислительная техника».

Рабочая учебная программа содержит:

- ✓ цели и задачи освоения дисциплины;
- ✓ результаты обучения и компетенции формируемые в процессе изучения дисциплины;
- ✓ место дисциплины в структуре ООП;
- ✓ карта компетенций дисциплины в разрезе тем;
- ✓ технологическая карта дисциплины;
- ✓ карта накопления баллов по дисциплине;
- ✓ тематический план распределения часов по видам занятий;
- ✓ программа дисциплины;
- ✓ требования к результатам освоения дисциплины;
- ✓ объем дисциплины и виды учебной работы;
- ✓ содержание дисциплины;
- ✓ библиотечно-информационные ресурсы;
- ✓ оценочные средства;
- ✓ материально-техническое обеспечение;
- ✓ учебно-методическое обеспечение дисциплины;
- ✓ политика выставления баллов.

На основании вышеизложенного считаю, что рецензируемая программа полностью соответствует требованиям, предъявляемым к рабочей программе по дисциплине «ТРПО» и может быть рекомендована для обучения магистрантов по направлению 710100 «Информатика и вычислительная техника».

Рецензент, к.ф.-м.н., доцент и зав. каф. ИСП:  Аркабаев Н.К.

Выписка из матрицы компетенций ООП специальности по направлению 710100 «Информатика и вычислительная техника»

710100 – «Информатика жана эсептөө техникасы» багыты боюнча магистранттарды даярдоодогу билим берүү программасынын максаттары (М):

М1. Магистранттарда компетенттик мамиленин негизинде социалдык-инсандык сапаттарды калыптандырууну улантуу, жалпы маданияттуулук деңгээлин жана кругозорун жогорулатуу; бүтүрүүчүлөрдүн мамлекеттик жана расмий тилдерде ишмердүүлүгүн жүргүзө алуусун өркүндөтүү, англис тилиндеги маалыматтарды колдоно алуу жөндөмдүүлүктөрүн арттыруу; интеллектуалдык потенциалын, билимин жана билгичтиктерин Кыргызстандын ар тараптан өнүгүшү үчүн жүзөгө ашырууга даяр экендигин тарбиялоо, өлкө үчүн болгон патриоттук сезимдерин жогорулатуу.

М2. Билим берүү тутумунун, иш берүүчүлөрдүн жана регионалдык жана аймактык эмгек рыногунун талаптарына жооп берген тутумдук иш-аракеттердин негизинде магистрлерди теоретикалык жана практикалык жактан даярдоо;

М3: Ишканаларды жана уюмдарды башкарууда колдонулган автоматташтырылган маалымат тутумдарын долбоорлоо, түзүү, башкаруу, эксплуатациялоо жана техникалык тейлөө жаатындагы кесиптик маселелерди чечүүгө бүтүрүүчүлөрдүн жаңы муунун окутуу;

М4: Бүтүрүүчүлөрдү өзгөчө жеке сапаттарга ээ болуусу, компьютерлердин жана тармактардын жабдыктарынын жана программалык камсыздоосунун тез өзгөрүү шартында, жогорку технологиялуу тармактарда эмгек рыногундагы атаандаштык чөйрөсүндө иштөөгө даярдыгы, компьютерлерди жана тармактарды колдонгон уюмдун каржылык туруктуулугуна жана стратегиялык натыйжалуулугуна жетишүү үчүн кесиптик көйгөйлөрдү чечүү мүмкүнчүлүгү, анын жашоо циклинин ар кандай баскычтарында пайдалануу;

М5: Келечектеги бүтүрүүчүлөрдүн заманбап менталитетин, Кыргызстанда атаандаштыкка жөндөмдүү базар экономикасын, инновациялык бизнес жана коомду маалыматташтыруу процесстерин эске алуу менен калыптандыруу.

М6: Тажрыйбалардын илимий-техникалык маалыматтарын жыйноо, анализдөө жана тажрыйбаларды жүргүзүү менен аларды колдоно алуу, стратегиялык пландаштыруу, инвестицияларды жана гранттарды тартуу алкагында информациялык-коммуникациялык технологияларды пайдаланып чет элдик өнөктөштөр менен кызматташып иштөөгө жана тажрыйба бөлүшүүгө жетишүү көндүмдөрүн калыптандыруу.

710100 – «Информатика жана эсептөө техникасы» магистранттарды даярдоо багытынын НББПсында төмөнкүдөй күтүлүүчү натыйжалар (КН) пландаштырылат:

№	**Күтүлүүчү натыйжалар**	**Компетенциялар**
КН1	**Долбоордук-конструктордук ишмердүүлүк:** техникалык каражаттардын мүмкүнчүлүктөрүн, үлгүлөрүн жана колдонуучулардын талаптарын анализдөөнүн негизинде, кесиптик ишмердүүлүктүн объекттеринин өзүнчө компоненттерине коюлуучу талаптарды жана өзгөчөлүктөрдү иштеп чыгуу; аппараттык комплекстердин компоненттерин долборлоо; аппараттык комплекстерди эффективдүү ишке ашыруу үчүн программалоо каражаттарын жана эсептөө техника каражаттарын колдонууга жөндөмдүү	КК-1, КК-2, КК-3, КК-4, КК-5, КК-8
КН2	**Технологиялык ишмердүүлүк:** эсептөөчү системаларды, автоматташтырылган системалардын компоненттерин,	ЖИК-2, КК-11, КК-12

	программаларды, программалык комплекстерди алдыңкы технологияларды пайдалануу менен керектүү сапатта жана мөөнөттө түзүү; аппараттык-программалык комплекстерди тестирлөө жана оңдоо; программаларды жана аларды текшерүүнүн ыкмаларын иштеп чыгуу, кесиптик ишмердүүлүктүн объекттерин текшерүүнү өткөрүү; кесиптик ишмердүүлүктүн объектерин сыноочу программаларды жана сыноо усулдарын талдап иштеп чыгуу, сыноолорду жүргүзүүгө жөндөмдүү	
КН3	**Өндүрүштүк ишмердүүлүк:** жаңы өндүрүмдү чыгарууда өндүрүштүк технологиялык процесстерди табууга жана өздөштүрүүгө; кесиптик ишмердүүлүктүн объекттерин иштеп чыгуунун технологияларын өздөштүрүү, өндүрүштөгү ар түрдүү татаал кырдаалдардан жана конфликтик ситуациялардан чыгымды азайтуу менен мыйзам чегинде чыга алууга; программалык өндүрүмдөрдүн эсептөөчү жана автоматташтырылган системдердин ишин көзөмөлдөп коштоого; кесипкөй ишмердиктин объекттерин эксплуатациялык мүнөздөмөлөрүн өлчөөнүн усулдарын жана каражаттарын тандоого; эсептөөчү жана автоматташтырылган системдердин системдик, инструменталдык жана колдонмо программалык жабдууларын орнотууга, калыптоого(настройка) жана тейлөөгө үйрөтүүгө жөндөмдүү	ЖОЖК-1
КН4	**Уюштуруу-башкаруу ишмердүүлүгү:** кесиптик ишмердүүлүктүн объекттерин иштеп чыгуу жараянынын этаптарын талап кылынган сапатта жана мөөнөттө уюштуруу; кесиптик ишмердүүлүктүн объекттерин иштеп чыгуу жараянын баалоо, көзөмөлдөө жана башкаруу; кесиптик ишмердүүлүктүн объекттерин иштеп чыгуу жараянын уюштурууда технологияны, аспаптык каражаттарды жана ЭТ каражаттарды тандоого жөндөмдүү	КК-11, КК-12
КН5	**Түздөө-оңдоо ишмердүүлүгү:** өздүк менчик программалар жана программалык комплекстерди талдап иштеп чыгуу, оңдоп түзөөдөн (отладка) өткөрүү, модернизациялоо; программаларды жана программалык комплекстерди оңдоп түзөө, тестирлөө үчүн усулдарды жана каражаттарды тандай, баалай билүү; эсептөө системаларынын жана автоматташтырылган системалардын аспаптык жана колдонмо программалык каражаттарды адаптациялоо жана тейлөөгө жөндөмдүү	КК-9, КК-10
КН6	**Илимий-изилдөө ишмердүүлүгү:** илимий-изилдөөдөгү, долбоордук-конструктордук ишмердүүлүктөгү, башкаруу системаларындагы компьютердик технологиялардын жана чечим кабыл алууну колдоочу системалардын математикалык үлгүлөрүн, ыкмаларын тандоо; кабыл алынган долбоорлоо чечимдерин шартто жана анын тууралыгын аныктоо үчүн эксперименттерди өткөрүү; аткарылган иштердин жыйынтыктарынын негизинде илимий-техникалык отчетторду, презентацияларды даярдоо, изилдөө жыйынтыктарын статья, доклад түрүндө илимий-	ЖИК-5, КК-6, КК-7,

	техникалык конференцияларда чыгарууга жөндөмдүү	
КН7	**Инновациялык ишмердүүлүгү:** өз ишмердүүлүк чөйрөсүндө үзгүлтүксүз пайда болуп жаткан жаңы инновацияларды өздөштүрүү, колдоно билүү жана аларга атаандаш идеяларды түзөө алуу; жаңы инновациялардын эффективдүүлүгүн баалай билүү, традициялык жана инновациялык ойлордун айырмачылыктарын изилдөө, түшүнө билүү, жаңы инновацияларды пайдалануу менен заманбап программалардын жана программалык комплекстердин үстүнөн иштөөгө, катышууга жөндөмдүү	ЖИК-4
КН8	**Сервистик-тейлөө ишмердүүлүгү:** жөнөкөй жана пайдаланууучулардын бардык катмарларын эске алуу менен жасалган интерфейске ээ болгон программалык өндүрүмдөрдү түзүүгө жана пайдалана билүү, программалардын жана программалык комплекстердин үзгүлтүксүз иштеп туруусун жана алардын бүтүндүгүн камсыздап туруу, программалардын жана программалык комплектерди сервистик зарылчылыкка жараша адаптациялоого жана оңдоп түзөөгө жөндөмдүү	ЖОЖК-2
КН9	**Илимий-педагогикалык ишмердүүлүгү:** түрдүү деңгээлиндеги билим берүү мекемелеринде тиешелүү багыттагы дисциплиналар боюнча педагогикалык иштерди жүргүзүүгө, лекциялык материалдарды, лабораториялык жана изилдөө комплекстерин иштеп чыгууга, окутуу процессинде методикалык колдоо көрсөтүүгө жөндөмдүү	ЖИК-1, СИЖМК-4
КН10	**Мамлекеттик жана расмий тилдерде иштөө жөндөмдүүлүгү:** мамлекеттик жана расмий тилдерде өзүнүн оозеки жана жазуу сүйлөмдөрүн логикалык туура, аргументтүү жана ачык-айкын түзүүгө жана иш жүргүзө билүүгө, аткарылган илимий-техникалык жана өндүрүштүк маселелердин отчетторун түзө алат, аларды презентация, статья, доклад формасында баяндай алууга жөндөмдүү	ЖИК-4, АК-2, КК-6, КК-7
КН11	**Дүйнөлүк маалыматтарды колдонуу ишмердүүлүгү:** адис өзүнүн ишмердүүлүк чөйрөсү боюнча англис тилинде эркин баарлашууга; башка дүйнөлүк тилдерде жарыяланган маалыматтарды алууга жана аларды өз ишмердүүлүк чөйрөсүндө колдоно билүүгө жөндөмдүү	АК-3, ЖИК-2
КН12	**Мамлекеттин өнүгүшүнө салым кошуу ишмердүүлүгү:** жигердүү атуулдук позицияда болууда, атуулдук демократиялык коомдун баалуулуктарынын негизинде диалог жүргүзө билүүгө, экономикалык эффективдүүлүгү жогору болгон өндүрүмдөрдү (программаларды, программалык комплекстерди ж.б.) иштеп чыгууга жөндөмдүү. Региондорду өнүктүрүүрүүсүнө жана санариптештирүүсүнө салым кошуугажөндөмдүү	СИЖМК-3, КК-5, КК-9, КК-10, КК-11
КН13	**Электрондук маалыматтар дүйнөсүндө иштөө жөндөмдүүлүгү:** ар кандай электрондук маалыматтарды түзө билүүгө, окуй алууга, колдоно билүүгө, интернет тармагы боюнча демократиялык баалуулуктардын негизинде диалог жүргүзүүгө жөндөмдүү; команданын курамында интранет, интернет тармактарындагы электрондук маалыматтарды	АК-1, СИЖМК-3, КК-1, ЖИК-3, КК-5, КК-9, КК-10, КК-11

	иштете алууга жөндөмдүү. региондорду өнүктүрүүсүнө жана санариптештирүүсүнө салым кошууга жөндөмдүү	
КН14	**Веб чөйрөсүндөгү ишмердүүлүк:** интернет тармагында клиент-сервер технологияларын колдонуу менен веб тиркемелерди түзүүгө, берилгендер базаларын түзүүгө жана эң алдынкы технологиядагы берилгендер базасын башкаруу системалар менен иштей билүүгө, интернет тармагы боюнча бизнес долбоорлорду түзө жана ишке ашыра билүүгө, аракеттеги бизнес долбоорлорго катыша билүүгө жөндөмдүү	АК-5, АК-6, КК-4
КН15	**Өздүк өнүгүү ишмердүүлүгү:** өздүк жетишкендиктерин баалай билүү жана кемчилдиктерине сын көз менен кароо, жетишкендиктерди андан ары өнүктүрүү жана кемчиликтерди кыска убакытта жоюу; жамааттык ишмердүүлүктү өнүктүрүү, коомдо кабыл алынган моралдык жана укуктук негизде социалдык өз-ара аракеттенишүүгө жөндөмдүү, элге сый, башка маданиятка толеранттуулук жана шериктештик мамилелерди колдоого даярдыкты көрсөтөт, ишмердик баарлашууну жүзөгө ашырууга жөндөмдүү: эл алдына чыгып сүйлөө, сүйлөшүүлөр, кеңешме өткөрүү, ишмердүүлүктө жазуу түрүндө баарлашууга жөндөмдүү	АК-4, СИЖМК-1, СИЖМК-2, СИЖМК-5, ЖИК-6

II. РАБОЧАЯ ПРОГРАММА

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ КЫРГЫЗСКОЙ РЕСПУБЛИКИ

ОШСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ МАТЕМАТИКИ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Кафедра «Автоматизированных систем и цифровых технологий»

«Утверждено»
на заседании кафедры АСЦТ
прот. № 1 от «28» августа 2020 г.

Зав. каф. АСЦТ, доц.: Молдояров У.

РАБОЧАЯ ПРОГРАММА

Дисциплина: **“Технология разработки программного обеспечения”**

Профиль подготовки**:**	ИВТ
Курс:	1
Группа:	ИВТ(м)-1-20

для магистрантов очного отделения, обучающихся по специальности:

ИВТ

Сетка часов по учебному плану

Наименование дисциплин	Количество часов					СРС	Отчетность
	Всего	Аудит. занятия					
		Ауд. зан.	Лекция	Лаб.	Модуль, ТК, РК		экзамен
ТРПО	90	36	20	16	4	54	экзамен
I, II семестр	90	36	20	16	4	54	1 с. - конс.

Рабочая программа составлена на основе Государственного образовательного стандарта по направлению 710100 “ИВТ”, академическая степень - магистр «ИВТ», для магистрантов обучающихся в очном отделении.

Составитель, доцент каф. АСЦТ: Беделова Н.С.

2020-2021 учебный год

2. 1. Цели освоения дисциплины

Целями преподавания предмета являются: предоставление обучаемым знаний и умений в области проектирования, тестирования, отладки, внедрения и сопровождения программного обеспечения (ПО) вычислительной техники с использованием современных CALS-технологий и CASE-средств. Поставленные цели полностью соответствуют целям ООП.

2.2. Результаты обучения и компетенции формируемые в процессе изучения дисциплины «ТРПО».

По дисциплине «Технология разработки программного обеспечения» магистранты должны:

знать:

- современные тенденции развития информатики и вычислительной техники, компьютерных технологий;
- основы создания информационных систем и использование новых информационных технологий обработки информации;
- жизненный цикл программного обеспечения;
- объектно-ориентированное программирование;
- теории и методы классификации;
- элементы теории сложности.

уметь:

- применять математические методы, физические законы и вычислительную технику для решения практических задач;
- программировать на одном из алгоритмических языков;
- применять алгоритмы поиска информации при разработке ПО;

владеть:

- основами алгоритмизации.

Код результатов	Результат обучения (выпускник должен быть готов)
P2	Применять глубокие специальные знания в области информатики и вычислительной техники для решения междисциплинарных инженерных задач.
P5	Эффективно работать индивидуально и в качестве члена и руководителя группы, в том числе междисциплинарной и международной, при решении инновационных инженерных задач.

В результате освоения модуля студент должен **знать**:

– жизненный цикл программ, методы оценки качества программных продуктов, технологии разработки программных комплексов, CASE-средства;

– методы и алгоритмы объектно-ориентированного программирования;

– методики, языки и стандарты информационной поддержки изделий (CALS-технологий) на различных этапах их жизненного цикла;

уметь:

– использовать типовые программные продукты, ориентированные на решение научных, проектных и технологических задач;

– эффективно работать в качестве члена команды по разработке программного обеспечения;

владеть:

– методиками сбора, переработки и представления научно-технических материалов по результатам исследований к опубликованию в печати, а также в виде обзоров, рефератов, отчетов, докладов и лекций;

– способностью брать на себя ответственность за результаты работы по разработке программных средств.

В процессе освоения дисциплины у магистрантов развиваются следующие **компетенции**:

1.Универсальные (общекультурные):

– способность использовать на практике умения и навыки в организации исследовательских и проектных работ, в управлении коллективом (ОК- 4);

– способность самостоятельно приобретать с помощью информационных технологий и использовать в практической деятельности новые знания и умения, в том числе в новых областях знаний, непосредственно не связанных со сферой деятельности (ОК-6);

2. Профессиональные:

– готовность применять перспективные методы исследования и решения профессиональных задач на основе знания мировых тенденций развития вычислительной техники и информационных технологий (ПК-1);

– способность выбирать методы и разрабатывать алгоритмы решения задач управления и проектирования объектов автоматизации (ПК-5);

– применять современные технологии разработки программных комплексов с использованием CASE-средств, контролировать качество разрабатываемых программных продуктов (ПК-6);

– организовывать работу и руководить коллективами разработчиков программных средств информационных и автоматизированных систем.

2.3. Место дисциплины в структуре ООП.

Дисциплина «Технология разработки программного обеспечения» относится к базовой составляющей профессионального цикла дисциплин учебного плана.

2.4. Карта компетенций дисциплины.

	ОК-4	**ОК-6**	**ПК-5**	**ПК-6**	**Общее количество компетенций**
1 часть. Инструментарий технологии разработки программирования	+		+		2
2 часть. Модели и методологии разработки ПО		+		+	2

2.5. Технологическая карта дисциплины.

Всего	Ауд. часы	СРС	1-модуль (30 б.)				2-модуль (30 б.)				Итоговый контроль (40 б.)				**Всего**
			Ауд. ч.		СРС		Ауд. ч.		СРС		Лекция	Лаборатория	СРС	**Итоговый контроль (ИК)**	
			Лекция	Лаб_я			Лекция	Лаб_я							
90	36	54	10	8	28		10	8	26						
Баллы			16	10	4		16	10	4		20	16	54	**40 б**	
Итоги модулей			К1=16+10+4=30 б.				К2=16+10+4= 30 б.				Э=20+12+8=40 б.				**100**

2.6. Карта накопливаемости баллов по дисциплине.

Карта накапливаемости баллов – это информация, предоставляющая студентам сведения о количестве баллов, получаемых им по всем видам работ, проверок и контроля по каждой теме изучаемой дисциплины.

На первой занятии, кроме общей информации, проводится ознакомление студентов с видами работ, контроли и картой накапливаемости баллов. В ходе занятий студентам регулярно проводятся напоминания о накапливаемости баллов.

В процессе изучения дисциплины магистранты должны выполнить 15 лабораторых работ оговоренных рабочей программой. На лабораторных работах магистранты осваивают работу с программным обеспечением путем выполнения плановых и индивидуальных заданий. В итоговой работе каждого рубежа в виде СРС дается выполнение проекта.

Оценивание уровня знания студентов в модулях проводится следующим образом:

В первом модуле проводятся два текущих контроля (ТК) и один рубежный контроль (РК). Каждый вид контроля оценивается по 30-балльной системе.

На 4 неделе занятий организуется ТК1, на 8 неделе – ТК2, а РК организуется также на 8 неделе.

ТК1 оценивается по результатам освоения студентами лекционного материала, пройденных до 4 недели учебного процесса, выполненных лабораторных занятий и самостоятельных работ. Оценивание производится по средне-арифметической системе:

ТК2 оценивается по результатам освоения студентами лекционного материала, пройденных с 4 недели по 8 неделю учебного процесса, выполненных лабораторных занятий и самостоятельных работ. Оценивание производится по средне-арифметической системе:

Результаты оценивания ТК1 и ТК2 фиксируются в групповом журнале и ведомости.

На 8 неделе по материалам первого модуля проводится первый рубежный контроль (РК1). В РК1 включаются все пройденные и освоенные лекционные материалы, выполненные лабораторные и самостоятельные работы первого модуля. Оценивание РК1 производится по средне-арифметической системе:

Оценивание первого модуля (М1) производится путем вычисления средне-арифметического текущих контролей и первого рубежного контроля:

$$M1 = \frac{ТК1 + ТК2 + РК1}{3}$$

Оценивание второго модуля производится по аналогичной схеме.

Итоговый контроль включает все пройденные лекционные материалы, выполненные лабораторные работы и самостоятельные работы, включенные в текущий семестр. Оценивание производится путем нахождения средне-арифметического:

$$ИК = \frac{Лек + Лаб + СРС}{3}$$

Экзаменнационный балл берется от средне-арифметического всех модулей, итогового контроля и дополнительного балла:

$$Экз = М1 + М2 + ИК + Д$$

Д – дополнительный балл. Дополнительный балл (поощрительный) указан в проекте «Система оценивания знаний».

2.7. Программа дисциплины.

1 тема: Введение ТРПО. Технологии.			
Компетенции	**ПК-**3: способен приобретать новые знания с большой степенью самостоятельности с использованием современных образовательных и информационных технологий		
РОд	**РО-2:** применять теоретические знания дисциплины (основные понятия технологии);		
Цель темы	Введение в ТРПО		
РОт	Лекция	1 ч.	**Ознакомится** с технологиями
	Лаб.	1 ч.	**Умеет** создать ПО
	СРС	3 с.	**Узнает** разные технологии
2 тема: Этапы развития			

Компетенции	**ПК-3:** способен приобретать новые знания с большой степенью самостоятельности с использованием современных образовательных и информационных технологий		
РОд	**РО-5:** применять теоретические знания дисциплины		
Цель темы:	Узнать этапы развития ТРПО		
РОт	Лекция	1 ч.	**Знает,** этапы развития ТРПО
	Лаб.	1 ч.	Создает программу
3 тема: Методы проектирования (ЛР №1).			
Компетенции	**ПК-3:** способен моделировать исследуемые объекты, применять средства и методики автоматизации принятия решений;		
Цель темы	Узнают методы проектирования		
Результаты обучения темы (РОт)	Лекция	1 ч.	**Знает** методы
	Лаб.	1 ч.	Может проектировать
	СРС	3 ч.	Узнает методы проектирования
4 тема: Этапы и элементы процесса разработки (ЛР №2).			
Компетенции	ОК-4: способен анализировать и критически переосмысливать накопленный опыт, изменять при необходимости профиль своей профессиональной деятельности, вносить собственный оригинальный вклад в развитие данной дисциплины, включая исследовательский контекст		
Цель темы	Процесс разработки ТРПО		
Результаты обучения темы (РОт)	Лекция	1 ч.	Знает элементы процесса разработки
	Лаб.	1 ч.	Создает этапы процесса разработки
	СРС	3 ч.	Получает знание о этапах и элементах процесса разработки
5 тема: Инструментарий технологии разработки программирования (ЛР №3).			
Компетенции	**ИК-4:** способен к дальнейшему самостоятельному обучению;		
Цель темы	Ознакомить с интрументариями ТРПО		
Результаты обучения темы (РОт)	Лекция	1 ч.	Знает инструментарий ТРПО
	Лаб.	1 ч.	Создает ПО
	СРС	3 ч.	Отличает инструментарии ТРПО
6 тема: Жизненный цикл программного обеспечения (ЛР №4).			
Компетенции	**ПК-1:** способен к разработке программного обеспечения проведения научных исследований и технических разработок, подготовке отдельных заданий для исполнителей;		
Цель темы	Ознакомить с жизненным циклом ПО		
Результаты обучения темы (РОт)	Лекция	1 ч.	Знает жизненный цикл ПО
	Лаб.	1 ч.	Создает ПО
	СРС	3 ч.	Отличает ПО
7 тема: Процесс реализации программных средств (ЛР №5).			
Компетенции	ИК-3: способен моделировать исследуемые объекты, применять средства и методики автоматизации принятия решений;		
Цель темы	Ознакомить с процессом реализации программных средств		
Результаты обучения темы (РОт)	Лекция	1 ч.	Знает программных средств
	Лаб.	1 ч.	Создает программу
	СРС	3 ч.	Процесс реализации программных средств
8 тема: Процессы проектирования (ЛР №6).			
Компетенции	ПК-3: способен готовить технических заданий на разработку проектных решений;		

Цель темы	Ознакомиться с процессом проектирования		
Результаты обучения темы (РОт)	Лекция	1 ч.	Узнает процесс проектирования
	Лаб.	1 ч.	Создает программу
	СРС	3 ч.	Процессы проектирования
9 тема: Модели и методологии разработки ПО (ЛР №7).			
Компетенции	ИК-4: способен к дальнейшему самостоятельному обучению;		
Результаты обучения темы (РОт)	Лекция	1 ч.	Знает модели и методологии разработки ПО
	Лаб.	1 ч.	Умеет создать программу
	СРС	3 ч.	Разработка ПО
10 тема: Каскадная модель (ЛР №8).			
Компетенции	**ПК-3:** способен готовить технических заданий на разработку проектных решений;		
Результаты обучения темы (РОт)	Лекция	1 ч.	Знает модели и методологии разработки ПО
	Лаб.	1 ч.	Умеет создать программу
	СРС	3 ч.	Каскадная модель
11 тема: V-образная модель (ЛР №9).			
Компетенции	**ИК-4:** способен к дальнейшему самостоятельному обучению;		
Результаты обучения темы (РОт)	Лекция	2 ч.	Знает V-образную модель
	Лаб.	2 ч.	Умеет создавать программу
	СРС	8 ч.	Владеет новым знанием
12 тема: Методологии разработки ПО (ЛР №10).			
Компетенции	**ПК-3:** способен моделировать исследуемые объекты, применять средства и методики автоматизации принятия решений;		
Результаты обучения темы (РОт)	Лекция	2 ч.	Знает методологии разработки ПО
	Лаб.	2 ч.	Умеет создавать программу
	СРС	6 ч.	Владеет новым знанием
13 тема: Качество ПО (ЛР №11).			
Компетенции	ПК-6: способен к постановке и модернизации отдельных лабораторных работ и практикумов по дисциплинам направления;		
Результаты обучения темы (РОт)	Лекция	2 ч.	Знает качество ПО
	Лаб.	2 ч.	Умеет создать программу
	СРС	8 ч.	Владеет новым знанием
14 тема: Методы выявления требований к ПО. Уровни требований. Анализ требований к ПО (ЛР №12).			
Компетенции	ПК-8: способен готовить технических заданий на разработку проектных решений		
Результаты обучения темы (РОт)	Лекция	2 ч.	Знает методы выявления требований к ПО
	Лаб.	2 ч.	Умеет создать программу
	СРС	8 ч.	Владеет новым знанием
15 тема: Обзор методологий проектирования программных продуктов (ЛР №13).			
Компетенции	**ПК-6:** способен к постановке и модернизации отдельных лабораторных работ и практикумов по дисциплинам направления		
Результаты обучения темы (РОт)	Лекция	2 ч.	Знает методологии проектирования программных продуктов
	Лаб.	2 ч.	Умеет создать программу
	СРС	8 ч.	Владеет новым знанием
16 тема: Технологии быстрой разработки ПО (ЛР №15).			
Компетенции	**ИК-4:** способен к дальнейшему самостоятельному обучению;		
Результаты обучения темы	Лекция	2 ч.	Знает технологию быстрой разработки ПО
	Лаб.	2 ч.	Умеет создать программу

(РОт)	СРС	8 ч.	Владеет новым знанием
17 тема: Объектно-ориентированное проектирование программной системы (ЛР №16)			
Компетенции	ПК-6: способен к постановке и модернизации отдельных лабораторных работ и практикумов по дисциплинам направления;		
Результаты обучения темы (РОт)	Лекция	2 ч.	Знает ООП ПС
	Лаб.	2 ч.	Умеет создать программу
	СРС	6 ч.	Владеет новым знанием
18 тема: Средства информационной поддержки программных проектов и изделий (CALS) технологий.			
Компетенции	**ПК-3:** способен моделировать исследуемые объекты, применять средства и методики автоматизации принятия решений;		
Результаты обучения темы (РОт)	Лекция	2 ч.	Знает CALS технологии
	Лаб.	2 ч.	Умеет создать программу
	СРС	6 ч.	Владеет новым знанием
19 тема: Тестирование и отладка программных систем.			
Компетенции	**ИК-4:** способен к дальнейшему самостоятельному обучению;		
Результаты обучения темы (РОт)	Лекция	2 ч.	Знает тестирование и отладку программных систем
	Лаб.	2 ч.	Умеет создать программу
	СРС	6 ч.	Владеет новым знанием
20 тема: Оценка качества ПО.			
Компетенции	**ПК-8:** способен готовить технических заданий на разработку проектных решений;		
Результаты обучения темы (РОт)	Лекция	2 ч.	Знает оценку качества ПО
	Лаб.	2 ч.	Умеет создать программу
	СРС	6 ч.	Владеет новым знанием

2.8. Тематический план распределения часов по видам занятий

Наименование разделов дисциплины (модулей) и тем	Аудиторные занятия					СРС	Используе-мые образователь-ные техн.	Формы Контроля
	Лекции	Лаб. занят.	Модуль	Консультация	Экзамен			
I модуль	**10**	**8**					1. МК 2. ДМ 3. ЭВМ	Т, КР, отчет,_лаб
1. Технологии	1							
2. Этапы развития	1							
3. Методы проектирования	1	1						
4. Этапы и элементы процесса разработки	1	1						
5. Инструментарий технологии разработки программирования	1	1						
6. Жизненный цикл программного обеспечения	1	1						

7. Процесс реализации программных средств	1	1						Т, КР, отчет, лаб
8. Процессы проектирования	1	1						
9. Модели и методологии разработки ПО	1	1						
10. Каскадная модель	1	1						
II модуль	**10**	**8**						
11. V-образная модель	1	1						Т, КР, отчет_лаб
12. Методологии разработки ПО	1	1						
13. Качество ПО	1	1					1. МК	Т, КР, СР, отчет_лаб
14. Методы выявления требований к ПО. Уровни требований. Анализ требований к ПО	1	1					2. ДМ	
15. Обзор методологий проектирования программных продуктов	1	1					3. ЭВМ	Т, КР, отчет_лаб
16. Технологии быстрой разработки ПО	1	1						
17. Объектно-ориентированное проектирование программной системы	1	1						Т, КР, отчет_лаб
18. Средства информационной поддержки программных проектов и изделий (CALS) технологий	1						1. МК	
19. Тестирование и отладка программных систем	1	1					2. ДМ	Т, КР, отчет_лаб
20. Оценка качества ПО	1						3. ЭВМ	
Итого за семестр:	**20**	**16**	**4**	**1**	**2**	**54**		

2.9. Календарно-тематический план по видам занятий

2.9.1. Лекция

№	Наименование разделов, модулей, темы и учебных вопросов	К-во часов
	I модуль	
1.	**Лекция № 1. Введение в технологию разработки программного обеспечения** 1. Введение 2. ТРПО **Литература:** Основная: [1,2,4]. Дополнительная: [1,3]. *Контрольные вопросы:* 1. Что такое технология разработки ПО? 2. Что явилось предпосылкой становления дисциплины «Технология	2

	разработки ПО»? Что явилось причиной стремительного развития ПО? 3. Чем отличаются программа и программное обеспечение? *Формы проверки знаний:* 1. Опрос. 2. Тест	
2.	**Лекция № 2. Этапы развития** **1.** British Computer Society **2.** Средства для создания информационных систем **Литература:** Основная: [2,4]. Дополнительная: [1,3]. *Контрольные вопросы:* 1. На сколько делится средства CASE-технологий? 2. Что такое компилятор? *Формы проверки знаний:* 1. Опрос. 2. Тест	
3.	**Лекция №3. Методы проектирования.** **1.** Программная система 2. Методы проектирования **Литература:** Основная: [2,4]. Дополнительная: [1,3]. *Контрольные вопросы:* 1. Что такое технология разработки ПО? 2. Что явилось предпосылкой становления дисциплины «Технология разработки ПО»? Что явилось причиной стремительного развития ПО? 3. Чем отличаются программа и программное обеспечение? *Формы проверки знаний:* 1. Опрос. 2. Тест	
4.	**Лекция №4. Этапы и элементы процесса разработки** 1. Этапы процесса разработки 2. Элементы процесса разработки **Литература:** Основная: [2,4]. Дополнительная: [1,3]. *Контрольные вопросы:* 1. Может ли большая программная система быть отлажена до конца и почему? 2. При каких условиях созданный программный комплекс может быть назван программным продуктом? 3. Что такое системное программное обеспечение? 4. Что такое инструментарий технологии программирования? *Формы проверки знаний:* 1. Опрос. 2. Тест	
5.	**Лекция №5. Инструментарий технологии разработки программирования** 1. Инструментарий технологии программирования 2. Технологии разработки программирования **Литература:** Основная: [2,4]. Дополнительная: [1,3]. *Контрольные вопросы:* 1. Что такое системное программное обеспечение? 2. Что такое инструментарий технологии программирования? *Формы проверки знаний:* 1. Опрос. 2. Тест	
6.	**Лекция № 6. Жизненный цикл программного обеспечения**	

	1. Понятие жизненного цикла ПО. 2. Процесс квалификационного тестирования программного средства. **Литература:** Основная: [1,2,4]. Дополнительная: [1,3]. *Контрольные вопросы:* 1. Что понимается под процессом жизненного цикла? 2. Основная цель процесса анализа требований к программным средствам. Что является результатом успешного осуществления процесса? 3. Процесс реализации. Какие виды деятельности и задачи входят в состав процесса реализации? *Формы проверки знаний:* 1. Опрос. 2. Тест	
7.	**Лекция № 7. Процесс реализации программных средств** 1. Процессы реализации программных средств 2. Эталонная модель **Литература:** Основная: [1, 4]. Дополнительная: [2,3]. *Контрольные вопросы:* 1. Основная цель процесса анализа требований к программным средствам. Что является результатом успешного осуществления процесса? 3. Процесс реализации. Какие виды деятельности и задачи входят в состав процесса реализации? 4. Процесс проектирования архитектуры программных средств. Что является результатом успешной реализации процесса. Что понимается под базовой линией? *Формы проверки знаний:* 1. Опрос. 2. Тест	
8.	**Лекция № 8. Процессы проектирования.** 1. Системы программирования **Литература:** Основная: [2,4]. Дополнительная: [1,3]. Контрольные вопросы: 1. Требования к процессам проектирования. *Формы проверки знаний:* 1. Опрос. 2. Тест	
9.	**Лекция № 9. Модели и методологии разработки ПО.** 1. Модели разработки ПО 2. Методологии разработки ПО **Литература:** Основная: [1,2]. Дополнительная: [3, 4]. *Контрольные вопросы:* 1. Основная цель процесса анализа требований к программным средствам. Что является результатом успешного осуществления процесса? 2. Процесс реализации. Какие виды деятельности и задачи входят в состав процесса реализации? 3. Процесс проектирования архитектуры программных средств. Что является результатом успешной реализации процесса. Что понимается под базовой линией? *Формы проверки знаний:* 1. Опрос. 2. Тест	
10.	**Лекция № 10. Каскадная модель.**	

	1. Модель ЖЦ ПО 2. Стандартная водопадная модель **Литература:** Основная: [1,2]. Дополнительная: [3, 4]. *Контрольные вопросы:* 1. Что является результатом успешного осуществления процесса? 2. Какие виды деятельности и задачи входят в состав процесса реализации? 3. Что понимается под базовой линией? *Формы проверки знаний:* 1. Опрос. 2. Тест	
11.	**Лекция № 11. V-образная модель** 1. V-образная модель, как разновидность каскадной модели 2. Стандартная водопадная модель **Литература:** Основная: [1,2]. Дополнительная: [3, 4]. *Контрольные вопросы:* 1. Основная цель процесса анализа требований к программным средствам. Что является результатом успешного осуществления процесса? 2. Процесс реализации. Какие виды деятельности и задачи входят в состав процесса реализации? *Формы проверки знаний:* 1. Опрос. 2. Тест	
12.	**Лекция № 12. Методологии разработки ПО.** 1. Методологии разработки ПО 2. Rational Unified Process 3. Жизненный цикл проекта **Литература:** Основная: [2,3]. Дополнительная: [1, 4]. *Контрольные вопросы:* 1. Что понимается под моделью ЖЦ ПО? Назовите существующие модели ЖЦ ПО. 2. Чем модель ЖЦ ПО отличается от методологии разработки ПО? Назовите существующие гибкие методологии разработки ПО. 3. Назовите основные особенности и стадии «Каскадной модели». 4. Назовите основные особенности и стадии «Эволюционной модели». 5. Методология Scrum. Что такое Спринт в рамках методологии Scrum? Какие группы ролей определены в данной методологии? *Формы проверки знаний:* 1. Опрос. 2. Тест	
13.	**Лекция № 13. Качество ПО.** 1. Характеристика программы 2. Концепция и сущность управления качеством ПС. **Литература:** Основная: [1,2,4]. Дополнительная: [1,3]. *Контрольные вопросы:* 1. Назовите основные особенности и стадии «Каскадной модели». 2. Назовите основные особенности и стадии «Эволюционной модели». 3. Методология Scrum. Что такое Спринт в рамках методологии Scrum? Какие группы ролей определены в данной методологии? *Формы проверки знаний:* 1. Опрос. 2. Тест.	
14.	**Лекция № 14.** Методы выявления требований к ПО.	

	1. Цель анализа требований. Уровни требований 2. Анализ требований к ПО. 3. Особенности интерпретации требований IEEE Standard Glossary of Software Engineering Terminology **Литература:** Основная: [2, 3]. Дополнительная: [1, 4]. *Контрольные вопросы:* 1. Что является результатом успешного осуществления процесса? 2. Какие виды деятельности и задачи входят в состав процесса реализации? 3. Что понимается под базовой линией? *Формы проверки знаний:* 1. Опрос. 2. Тест	
15.	**Лекция № 15. Обзор методологий проектирования программных продуктов.** 1. Анализ требований 2. Проектирования программных продуктов. **Литература:** Основная: [1,4]. Дополнительная: [2,3]. *Контрольные вопросы:* 1. Назовите основные особенности и стадии «Каскадной модели». 2. Назовите основные особенности и стадии «Эволюционной модели». 3. Какие группы ролей определены в данной методологии? *Формы проверки знаний:* 1. Опрос. 2. Тест.	
16.	**Лекция № 16. Технологии быстрой разработки ПО.** 1. Спецификации требований к ПО. 2. Проверка требований. **Литература:** Основная: [1, 4]. Дополнительная: [1, 3]. *Контрольные вопросы:* 1. Что понимается под процессом жизненного цикла? 2. Основная цель процесса анализа требований к программным средствам. Что является результатом успешного осуществления процесса? 3. Процесс реализации. Какие виды деятельности и задачи входят в состав процесса реализации? *Формы проверки знаний:* 1. Опрос. 2. Тест	
17.	**Лекция № 17. Объектно-ориентированное проектирование программной системы.** 1. Модели качества 2. Стандартная водопадная модель **Литература:** Основная: [1, 2]. Дополнительная: [3, 4]. *Контрольные вопросы:* 1. Что является результатом успешного осуществления процесса? 2. Процесс реализации. Какие виды деятельности и задачи входят в состав процесса реализации? *Формы проверки знаний:* 1. Опрос. 2. Тест	
18.	**Лекция № 18. Средства информационной поддержки программных проектов и изделий (CALS) технологий**	

	1. Предпрограммная подготовка задачи 2. **Литература:** Основная: [2, 3]. Дополнительная: [1, 4]. *Контрольные вопросы:* 1. Какие виды деятельности и задачи входят в состав процесса реализации? 2. Что является результатом успешного осуществления процесса? *Формы проверки знаний:* 1. Опрос. 2. Тест	
19.	**Лекция № 19. Тестирование и отладка программных систем.** 1. Тестирование программных средств. 2. Отладка программных средств. **Литература:** Основная: [1,2,4]. Дополнительная: [1,3]. *Контрольные вопросы:* 1.Как осуществляется тестирование? 2. Основная цель процесса анализа требований к программным средствам. 3. Процесс реализации. *Формы проверки знаний:* 1. Опрос. 2. Тест	
20.	**Лекция № 20. Оценка качества ПО.** 1. Оценка качества ПО. 2. Концепция и сущность управления качеством ПС **Литература:** Основная: [2, 4]. Дополнительная: [1, 3]. *Контрольные вопросы:* 1. Назовите существующие модели ЖЦ ПО. 2. Назовите существующие гибкие методологии разработки ПО. 3. Методология Scrum. Что такое Спринт в рамках методологии Scrum? 4. Какие группы ролей определены в данной методологии? *Формы проверки знаний:* 1. Опрос. 2. Тест	

2.9.2. Содержание лабораторных занятий

№	Темы и содержание	Количество часов
1.	Создание таблиц с помощью Конструктора, включая установления связи между таблицами	1
2.	*Создание форм и отчетов с помощью Конструктора и Мастера*	1
3.	Создание простых SQL запросов на редактирование и выборку данных одной таблицы	1
4.	Создание SQL запросов на основе нескольких таблиц (параметрических, перекрестных)	1
5.	Создание макросов	1
6.	Создание простого приложения по работе с БД	1
7.	Организация деятельности администрации гостиницы	1
8.	Организация работы службы автоинспекции	1
9.	Деятельность налоговой службы	1

10.	Организация работы службы социальной помощи	1
11.	Деятельность абонентской службы АТС	1
12.	Организация работы рекламного агентства	1
13.	Деятельность службы трудоустройства	1
14.	Организация работы службы общественного питания	1
15.	Организация работы службы скорой помощи	1
16.	Деятельность фирмы бартерного обмена	1
Всего:		**16**

2.9.3. Задания на самостоятельной работы студентов

1. Самостоятельную работу магистрантов (СРС) можно разделить на текущую и творческую (исследовательскую).

Текущая СРС:

- работа с лекционным материалом, поиск и обзор литературы и электронных источников информации по индивидуально заданной проблеме курса,
- выполнение домашних заданий, домашних контрольных работ,
- опережающая самостоятельная работа,
- перевод текстов с иностранных языков,
- изучение тем, вынесенных на самостоятельную проработку,
- подготовка к лабораторным работам;
- подготовка к контрольной работе, к экзамену.

Творческая проблемно-ориентированная самостоятельная работа (ТСР):

- поиск, анализ, структурирование и презентация информации;
- исследовательская работа и участие в научных студенческих конференциях, семинарах и олимпиадах;
- анализ научных публикаций по заранее определенной преподавателем теме.

2. Содержание самостоятельной работы магистрантов

1. Перечень научных проблем и направлений научных исследований
2. Изучение процесса развития технологии проектирования программных систем.
3. Изучение технологий быстрой разработки программных систем.
4. Проектирование приложений с использованием технологии быстрой разработки ПО.
5. Изучение средств командной разработки ПО.

3. Темы индивидуальных заданий (примеры)

1. Организация деятельности администрации гостиницы.
2. Организация работы службы автоинспекции.
3. Деятельность налоговой службы.
4. Организация работы службы социальной помощи.
5. Деятельность абонентской службы АТС.
6. Организация работы рекламного агентства.
7. Деятельность службы трудоустройства.
8. Организация работы службы общественного питания.
9. Организация работы службы скорой помощи.
10. Деятельность фирмы бартерного обмена.

2.10. Задании итогового контроля

Темы курсовых проектов (примеры)

1. Гостиница

Ведение справочников: Номера, Услуги, Клиенты Функции: Ведение справочников, поселение и выселение клиентов, бронирование мест, учёт оказанных услуг Выходные документы: Счёт за проживание и услуги, Список проживавших на момент времени, Список номеров, Прейскурант услуг.
2. Успеваемость студентов Ведение справочников: Институты, кафедры, магистранты, предметы Функции: ведение справочников, учёт успеваемости студентов Выходные документы: Ведомость успеваемости по группе студентов, Приложение к диплому, Аналитические отчёты.
3. Больница Ведение справочников: Пациенты, Болезни, Палаты, Врачи, История болезни Функции: Ведение справочников, приём пациента, ведение истории болезни, выписка. Выходные документы: Список пациентов, Список врачей, Карточка больного.
4. Аптека Ведение справочников: Группы лекарств, Лекарства, Производители, Поставщики Функции: ведение справочников, учёт прихода и продаж лекарств Выходные документы: Отчёт по наличию лекарств на складе по группам, Отчёт по продажам по группам, Счёт-фактура.
5. Кафе Ведение справочников: Продукты, Блюда, Заказы Функции: Ведение справочников, хранение рецептов, расчёт себестоимости блюда, приём заказов Выходные документы: Меню, Счёт заказа, Отчёт по продуктам на складе, Заказы за период.

Тестовый контроль проводится в виде опроса по теоретическим вопросам дисциплины при защите лабораторной работы.

При изучении дисциплины магистранты должны выполнить 8 лабораторных работ по темам, оговоренным рабочей программой. На лабораторных работах магистранты осваивают работу с путем выполнения индивидуальных заданий.

Своевременной называется исполнение лабораторной работы в течение двух недель с момента предоставления задания по плану занятий. По результатам выполнения каждой лабораторной работы магистранту выставляется оценка.

2.11. Учебно-методическое обеспечение дисциплины

2.11. 1. Основная

1. Буч Г., Рамбо Д., Декобсон А. Язык UML. Руководство пользователя: Пер. с англ. — М.: ДМК, 2018. – 432 с.
2. С. Орлов. Технологии разработки программного обеспечения. Учебное пособие. — СПб.: Изд-во «Питер», 2016. – 480 с.
3. Мирошниченко Е.А. Технология программирования: Учебное пособие. – Томск: Изд. ТПУ, 2020. – 42 с.

2.11.2. Дополнительная

1. Фокс Дж.. Программное обеспечение и его разработка. – М.: Мир, 2017. – 360 с.
2. Коннолли Т. Базы данных: проектирование, реализация и сопровождение. Теория и практика: [пер. с англ.] / Т. Коннолли, К. Бегг.

13.3. Электронные источники

1. Кузнецов С.Д. Основы современных баз данных. http://www.citforum.ru

2. Когаловский М.Р. Абстракции и модели в системах баз данных [Электронный ресурс]. //СУБД. 2018 №4,5. С.7. Режим доступа: http://www.osp.ru/dbms/2018/04_05/07.htm.
3. Чен П. П.-Ш. Модель «сущность-связь» - шаг к единому представлению данных [Электронный ресурс]. //СУБД. 2017. № 3, С.137 - 158. Режим доступа: http://www.osp.ru/dbms/1995/03/271.htm.
4. Учебно-методические материалы, находящие в локальной сети кафедры ОСУ.

2.12. Критерии оценки знаний студентов на экзамене.

В I, II семестре установлен экзамен, результаты по дисциплине общий балл (О*балл*) ставится следующим образом:

Рейтинг (балл)	Оценка по баллной системе	Оценка GPA по цифровой эквивалентности	Оценка по традиционной системе
87 – 100	A	4,0	Отлично
80 – 86	B	3,33	Хорошо
74 – 79	C	3,0	
68 – 73	Д	2,33	Удовлетворительно
61 – 67	E	2,0	
31 -60	FX	0	Не удовлетворительно

В соответствии с действующими нормативными актами и рекомендациями Министерства образования и науки КР устанавливаются следующие критерии выставления оценок на экзаменах:

- оценка ***"отлично"*** выставляется студенту, который обнаружил на экзамене всестороннее, систематическое и глубокое знание учебно-программного материала, умение свободно выполнять задания, предусмотренные программой, который усвоил основную литературу и ознакомился с дополнительной литературой, рекомендованной программой. Как правило, оценка "отлично" выставляется студентам, усвоившим взаимосвязь основных понятий дисциплины и их значений для приобретаемой профессии, проявившим творческие способности в понимании, изложении и использовании учебно-программного материала;

- оценка ***"хорошо"*** выставляется студенту, который на экзамене обнаружил полное знание учебно-программного материала, успешно выполнил предусмотренные в программе задания, усвоил основную литературу, рекомендованную в программе. Как правило, оценка "хорошо" выставляется студентам, показавшим систематический характер знаний по дисциплине и способным к их самостоятельному выполнению и обновлению в ходе дальнейшей учебной работы и профессиональной деятельности;

- оценка ***"удовлетворительно"*** выставляется студенту, обнаружившему знание основного учебного материала в объеме, необходимом для дальнейшей учебы и предстоящей работы по профессии, справляющемуся с выполнением заданий, предусмотренных программой, который ознакомился с основной литературой, рекомендованной программой. Как правило, оценка "удовлетворительно" выставляется студентам, допустившим погрешности в ответе на экзамене и при выполнении экзаменационных заданий, но обладающим необходимыми знаниями для их устранения под руководством преподавателя;

- оценка ***"неудовлетворительно"*** выставляется студенту, обнаружившему пробелы в знаниях основного учебно-программного материала, допустившему принципиальные ошибки в выполнении предусмотренных программой заданий, не ознакомившемуся с основной литературой, предусмотренной программой, и не овладевшему базовыми знаниями, предусмотренными по данной дисциплине и определенными соответствующей программой дисциплины.

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ КЫРГЫЗСКОЙ РЕСПУБЛИКИ

ОШСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ МАТЕМАТИКИ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Кафедра «Автоматизированных систем и цифровых технологий»

«Утверждено»
на заседании кафедры АСЦТ
прот. № 1 от «28» августа 2020 г.
Зав. каф. АСЦТ, доцент: Молдояров У.

УЧЕБНАЯ ПРОГРАММА МАГИСТРАНТА (СИЛЛАБУС)

Дисциплина: **“Технология разработки программного обеспечения”**

Профиль подготовки**:** ИВТ
Курс: 1
Группа: ИВТ(м)-1-20

Для магистрантов очного отделения, обучающихся по специальности:

ИВТ

Расчет часов по учебному плану

Наименование дисциплин	Количество часов					СРС	Отчетность
	Всего	Аудиторные занятия					
		Ауд. зан.	Лекция	Лаб-я	Модуль		I
1 курс, I с., 3 кредита	90	36	20	16	6	54	экзамен
	90	36	20	16	6	54	1 ч.-конс.

Силлабус составлен на основе Государственного образовательного стандарта специальности “ИВТ” для магистрантов, обучающихся в очном отделении

Составитель, доцент каф. АСЦТ: _____ Беделова Н.С.

2020-2021 учебный год

Сведения о преподавателей

Лектор-преподаватель:

Беделова Нургуль Салибаевна – доцент каф. АСЦТ,

ФМИТ ОшГУ, общий стаж работы – 22 года,

образование – высшее, закончила

физико-математический факультет ОшГУ, 2000 г.

Рабочий телефон: 03222-2-11-85,
Рабочее место: 723500. Главный корпус ОшГУ,
ул. Ленина 331, каб. 205.
Мобильный телефон: 0772-43-73-31
E-mail: kireshe78@mail.ru

Сетка часов

№	Сем	Всего	Ауд	Лекции	Лаб	Конс	Экз	Модуль	СРС
1	I	90	36	20	16	1	2	4	54
Всего:		90	36	20	16	1	2	4	54

Дисциплина «Технология разработки программного обеспечения» относится к базовой составляющей профессионального цикла дисциплин учебного плана.

3.1. Цель и задачи дисциплины

Целями преподавания предмета являются: предоставление обучаемым знаний и умений в области проектирования, тестирования, отладки, внедрения и сопровождения программного обеспечения (ПО) вычислительной техники с использованием современных CALS-технологий и CASE-средств. Поставленные цели полностью соответствуют целям ООП.

Задачи дисциплины

По дисциплине «Технология разработки программного обеспечения» магистранты должны:

знать:

- современные тенденции развития информатики и вычислительной техники, компьютерных технологий;
- основы создания информационных систем и использование новых информационных технологий обработки информации;
- жизненный цикл программного обеспечения;
- объектно-ориентированное программирование;
- теории и методы классификации;
- элементы теории сложности.

уметь:

- применять математические методы, физические законы и вычислительную технику для решения практических задач;
- программировать на одном из алгоритмических языков;
- применять алгоритмы поиска информации при разработке ПО;

владеть:

- основами алгоритмизации.

3.2. Результаты освоения дисциплины

В результате освоения модуля студент должен **знать**:

- жизненный цикл программ, методы оценки качества программных продуктов, технологии разработки программных комплексов, CASE-средства;
- методы и алгоритмы объектно-ориентированного программирования;
- методики, языки и стандарты информационной поддержки изделий (CALS-технологий) на различных этапах их жизненного цикла;

уметь:

- использовать типовые программные продукты, ориентированные на решение научных, проектных и технологических задач;
- эффективно работать в качестве члена команды по разработке программного обеспечения;

владеть:

- методиками сбора, переработки и представления научно-технических материалов по результатам исследований к опубликованию в печати, а также в виде обзоров, рефератов, отчетов, докладов и лекций;
- способностью брать на себя ответственность за результаты работы по разработке программных средств.

В процессе освоения дисциплины у студентов развиваются следующие **компетенции**:

1.Универсальные (общекультурные):

- способность использовать на практике умения и навыки в организации исследовательских и проектных работ, в управлении коллективом (ОК- 4);
- способность самостоятельно приобретать с помощью информационных технологий и использовать в практической деятельности новые знания и умения, в том числе в новых областях знаний, непосредственно не связанных со сферой деятельности (ОК-6);

2. Профессиональные:

- готовность применять перспективные методы исследования и решения профессиональных задач на основе знания мировых тенденций развития вычислительной техники и информационных технологий (ПК-1);
- способность выбирать методы и разрабатывать алгоритмы решения задач управления и проектирования объектов автоматизации (ПК-5);
- применять современные технологии разработки программных комплексов с использованием CASE-средств, контролировать качество разрабатываемых программных продуктов (ПК-6);
- организовывать работу и руководить коллективами разработчиков программных средств информационных и автоматизированных систем.
-

3.3. Политика курса

Этот курс, читаемый для магистрантов очного отделения «ИВТ» факультета Математики и информационных технологий, предполагает рассмотрение на практических занятиях задач и примеров, непосредственно связанных с будущей специальностью магистрантов.

Учебный процесс осуществляется с применением модульно–рейтинговой системы оценивания успеваемости магистрантов с помощью информационной системы AVN.

Магистрантам предъявляется, следующие системы требований и правил поведения на занятиях:

а). Обязательное посещение занятий;

б). Активность во время занятий;

в). Подготовка к занятиям, к выполнению домашнего задания и СРС.

Недопустимо:

- Опоздание и уход с занятий;
- Пользование сотовыми телефонами во время занятий;
- Обман и плагиат;
- Несвоевременная сдача заданий.

3.4. Пререквизиты курса

«Вычислительные системы», «Современные проблемы информатики и вычислительной техники».

3.5. Постреквизиты курса «Технология программирования».

3.6. Образовательные технологии

В лекционной части курса рассматриваются знания, умения и навыки технологии разработки программного обеспечения. Изучение всех тем сопровождается иллюстрирующими примерами.

Лабораторные работы в компьютерных классах служат для индивидуальной работы магистрантов над учебными задачами и итоговым проектом с целью выработки и закрепления практических навыков в области прикладной математики, IT-технологии, информатики и вычислительной техники.

Применяемые информационные технологии: *лекции в форме презентаций, обучающие и тестирующие программы, электронные учебники.*

Формой рубежного (промежуточного) контроля знаний и умений магистрантов по курсу *являются модули.*

Формой итогового контроля знаний и умений магистрантов по курсу *является экзамен.*

Текущий и рубежный контроль. Магистранты, после выполнения соответствующих (первому или второму модулю) лабораторных работ допускаются к текущему контролю. Текущий контроль осуществляется на платформе ИС AVN ОшГУ в виде компьютерного тестирования. Каждый из двух рубежных контролей (модулей) оценивается по 30 балльной шкале.

Итоговый контроль. Итоговый контроль реализуется в форме защиты собственно созданных программ (в виде компьютерного тестирования) и оценивается по 40 балльной шкале.

Правила оценивания рубежного и итогового контроля.

Рубежный контроль (М1, М2)		Итоговый контроль (ИК)	
Количество правильных ответов	Вставляемые баллы	Количество правильных ответов	Вставляемые баллы

K, K=1,2,…,30	K, K=1,2,…,30	N, N=1,2,…,40	N, N=1,2,…,40

3.7. Технологическая карта

Всего	Ауд. часы	СРС	1-модуль (30 б.)			2-модуль (30 б.)			Итоговый контроль (40 б.)				**Всего**
			Ауд. ч.		СРС	Ауд. ч.		СРС	Лекция	Лаборатория	СРС	**Итоговый контроль (ИК)**	
			Лекция	Лаб_я		Лекция	Лаб_я						
90	36	54	10	8	28	10	8	26					
Баллы			16	10	4	16	10	4	20	16	54	**40 б**	
Итоги модулей			К1=16+10+4=30 б.			К2=16+10+4= 30 б.			Э=20+12+8=40 б.				**100**

3.8. Содержание лекционных и лабораторных занятий

Наименование разделов дисциплины (модулей) и тем	Аудиторные занятия					СРС	Используемые образователь-ные техн.	Формы Контроля
	Лекции	Лаб. занят.	Модуль	Консультация	Экзамен			
I модуль	**10**	**8**					1. МК	Т, КР, отчет,_лаб
1. Технологии	1							
2. Этапы развития	1						2. ДМ	
3. Методы проектирования	1	1						
4. Этапы и элементы процесса разработки	1	1					3. ЭВМ	
5. Инструментарий технологии разработки программирования	1	1						
6. Жизненный цикл программного обеспечения	1	1						Т, КР, отчет, лаб
7. Процесс реализации программных средств	1	1						
8. Процессы проектирования	1	1						
9. Модели и методологии разработки ПО	1	1						
10. Каскадная модель	1	1						
II модуль	**10**	**8**						
11. V-образная модель	1	1						Т, КР, отчет_лаб
12. Методологии разработки ПО	1	1						
13. Качество ПО	1	1					1. МК	Т, КР, СР,

14. Методы выявления требований к ПО. Уровни требований. Анализ требований к ПО	1	1					2. ДМ 3. ЭВМ	отчет_лаб
15. Обзор методологий проектирования программных продуктов	1	1						Т, КР, отчет_лаб
16. Технологии быстрой разработки ПО	1	1						
17. Объектно-ориентированное проектирование программной системы	1	1						Т, КР, отчет_лаб
18. Средства информационной поддержки программных проектов и изделий (CALS) технологий	1						1. МК	
19. Тестирование и отладка программных систем	1	1					2. ДМ	Т, КР, отчет_лаб
20. Оценка качества ПО	1						3. ЭВМ	
Итого за семестр:	**20**	**16**	**4**	**1**	**2**	**54**		

Тематический план лабораторных занятий

№	Темы и содержание	Количество часов
1.	Создание таблиц с помощью Конструктора, включая установления связи между таблицами	1
2.	*Создание форм и отчетов с помощью Конструктора и Мастера*	1
3.	Создание простых SQL запросов на редактирование и выборку данных одной таблицы	1
4.	Создание SQL запросов на основе нескольких таблиц (параметрических, перекрестных)	1
5.	Создание макросов	1
6.	Создание простого приложения по работе с БД	1
7.	Организация деятельности администрации гостиницы	1
8.	Организация работы службы автоинспекции	1
9.	Деятельность налоговой службы	1
10.	Организация работы службы социальной помощи	1
11.	Деятельность абонентской службы АТС	1
12.	Организация работы рекламного агентства	1
13.	Деятельность службы трудоустройства	1
14.	Организация работы службы общественного питания	1
15.	Организация работы службы скорой помощи	1
16.	Деятельность фирмы бартерного обмена	1
Всего:		**16**

Темы курсовых проектов (примеры)

1. Гостиница Ведение справочников: Номера, Услуги, Клиенты Функции: Ведение справочников, поселение и выселение клиентов, бронирование мест, учёт оказанных услуг Выходные документы: Счёт за проживание и услуги, Список проживавших на момент времени, Список номеров, Прейскурант услуг.
2. Успеваемость студентов Ведение справочников: Институты, кафедры, магистранты, предметы Функции: ведение справочников, учёт успеваемости студентов Выходные документы: Ведомость успеваемости по группе студентов, Приложение к диплому, Аналитические отчёты.
3. Больница Ведение справочников: Пациенты, Болезни, Палаты, Врачи, История болезни Функции: Ведение справочников, приём пациента, ведение истории болезни, выписка. Выходные документы: Список пациентов, Список врачей, Карточка больного.
4. Аптека Ведение справочников: Группы лекарств, Лекарства, Производители, Поставщики Функции: ведение справочников, учёт прихода и продаж лекарств Выходные документы: Отчёт по наличию лекарств на складе по группам, Отчёт по продажам по группам, Счёт-фактура.
5. Кафе Ведение справочников: Продукты, Блюда, Заказы Функции: Ведение справочников, хранение рецептов, расчёт себестоимости блюда, приём заказов Выходные документы: Меню, Счёт заказа, Отчёт по продуктам на складе, Заказы за период.

Тестовый контроль проводится в виде опроса по теоретическим вопросам дисциплины при защите лабораторной работы.

При изучении дисциплины магистранты должны выполнить 8 лабораторных работ по темам, оговоренным рабочей программой. На лабораторных работах магистранты осваивают работу с путем выполнения индивидуальных заданий.

Своевременной называется исполнение лабораторной работы в течение двух недель с момента предоставления задания по плану занятий. По результатам выполнения каждой лабораторной работы магистранту выставляется оценка.

3.9. Задания для СРС

3.9.1. Самостоятельную работу магистрантов (СРС) можно разделить на текущую и творческую (исследовательскую).

Текущая СРС:

- работа с лекционным материалом, поиск и обзор литературы и электронных источников информации по индивидуально заданной проблеме курса,
- выполнение домашних заданий, домашних контрольных работ,
- опережающая самостоятельная работа,
- перевод текстов с иностранных языков,
- изучение тем, вынесенных на самостоятельную проработку,
- подготовка к лабораторным работам;
- подготовка к контрольной работе, к экзамену.

Творческая проблемно-ориентированная самостоятельная работа (ТСР):

- поиск, анализ, структурирование и презентация информации;
- исследовательская работа и участие в научных студенческих конференциях, семинарах и олимпиадах;
- анализ научных публикаций по заранее определенной преподавателем теме.

3.9.2. Содержание самостоятельной работы магистрантов

1. Перечень научных проблем и направлений научных исследований
2. Изучение процесса развития технологии проектирования программных систем.
3. Изучение технологий быстрой разработки программных систем.
4. Проектирование приложений с использованием технологии быстрой разработки ПО.
5. Изучение средств командной разработки ПО.

3.9.3. Темы индивидуальных заданий (примеры)

11. Организация деятельности администрации гостиницы.
12. Организация работы службы автоинспекции.
13. Деятельность налоговой службы.
14. Организация работы службы социальной помощи.
15. Деятельность абонентской службы АТС.
16. Организация работы рекламного агентства.
17. Деятельность службы трудоустройства.
18. Организация работы службы общественного питания.
19. Организация работы службы скорой помощи.
20. Деятельность фирмы бартерного обмена.

3.10. Учебно-методическое обеспечение дисциплины

3. 10.1. Основная

1. Буч Г., Рамбо Д., Декобсон А. Язык UML. Руководство пользователя: Пер. с англ. — М.: ДМК, 2018. – 432 с.
2. С. Орлов. Технологии разработки программного обеспечения. Учебное пособие. — СПб.: Изд-во «Питер», 2016. – 480 с.
3. Мирошниченко Е.А. Технология программирования: Учебное пособие. – Томск: Изд. ТПУ, 2020. – 42 с.

3.10.2. Дополнительная

1. Фокс Дж.. Программное обеспечение и его разработка. – М.: Мир, 2017. – 360 с.
2. Коннолли Т. Базы данных: проектирование, реализация и сопровождение. Теория и практика: [пер. с англ.] / Т. Коннолли, К. Бегг.

3.10.3. Электронные источники

1. Кузнецов С.Д. Основы современных баз данных. http://www.citforum.ru
2. Когаловский М.Р. Абстракции и модели в системах баз данных [Электронный ресурс]. //СУБД. 2018 №4, 5. С.7. Режим доступа: http://www.osp.ru/dbms/2018/04_05/07.htm.

3. Чен П. П.-Ш. Модель «сущность-связь» - шаг к единому представлению данных [Электронный ресурс]. //СУБД. 2017. № 3, С.137 - 158. Режим доступа: http://www.osp.ru/dbms/1995/03/271.htm.
4. Учебно-методические материалы, находящие в локальной сети кафедры ОСУ.

3.11. Примерный перечень вопросов для подготовки к экзамену

1. Что такое промышленный программный продукт. Дать определения пакета прикладных программ, программной системы.

2. Жизненный цикл программного обеспечения. Дать краткую характеристику каждого этапа.
3. Почему программные системы сложны. Привести пять признаков сложной системы.
4. Техническое задание. Перечислить и охарактеризовать разделы, входящие в техническое задание.
5. Унифицированный процесс разработки программного обеспечения. Жизненный цикл унифицированного процесса.
6. Работа с кадрами. Перечислить роли разработчиков и дать характеристику каждой из них.
7. Дать определения проекта, процесса, продукта с точки зрения унифицированного процесса разработки программного обеспечения.
8. Что такое артефакт. В чем преимущества организованного процесса разработки программного обеспечения.
9. Использование языка UML при проектировании сложных программных систем. Какие диаграммы используются в UML для создания моделей программной системы.
10. Диаграмма вариантов использования, ее назначение. Рассказать о варианте использования и действующем лице. Правила построения диаграммы вариантов использования.
11. Понятие класса и объекта. Что может быть объектом. Что такое атрибут и операция.
12. Пять критериев проверки правильности построения класса.
13. Что такое классификация с точки зрения объектно-ориентированного проектирования программных систем. Теории классификации.
14. Методы классификации.
15. Микропроцесс проектирования. Перечислить этапы и основные виды деятельности выполняемые на каждом из них.
16. Микропроцесс проектирования – первый этап.
17. Микропроцесс проектирования – второй этап.
18. Микропроцесс проектирования – третий этап.
19. Микропроцесс проектирования – четвертый этап.
20. Диаграммы взаимодействия. Основное назначение.
21. Диаграмма классов. Ее назначение. Что она включает. Рассказать об основных видах связей между классами.
22. Дать определение тестированию и отладке. Особенности и объекты тестирования. Автономное и комплексное тестирование.
23. Дать определение тестированию и отладке. Направления тестирования. Стратегия тестирования. Контрольный лист тестирования модуля.
24. Дать определение тестированию и отладке. Локализация ошибок. Классификация ошибок. Безопасное программирование.
25. Оценки ошибок.

26. Документирование. Состав и содержание документов прилагаемых к программной системе.
27. Внедрение программного комплекса. Планирование испытаний.
28. Внедрение программного комплекса. Подготовка тестовых данных. Анализ результатов испытаний.
29. Что такое качество с точки зрения квалиметрии. Дать определение свойству и показателю качества ПО. Основные задачи решаемые при оценке качества.
30. Оценка качества программного обеспечения. Методы оценки свойств программного обеспечения.

3.12. Критерии оценки знаний студентов

Выставление оценок на экзаменах осуществляется на основе принципов объективности, справедливости, всестороннего анализа качества знаний магистрантов, и других положений, способствующих повышению надежности оценки знаний обучающихся и устранению субъективных факторов.

Оценка знаний (академической успеваемости) магистранту осуществляется по 100 балльной системе (шкале) следующим образом:

Рейтинг (баллы)	**Оценка по буквенной системе**	**Цифровой эквивалент оценки по GPA**	**Оценка по традиционной системе**
87 – 100	A	4,0	Отлично
80 – 86	B	3,33	Хорошо
74 – 79	C	3,0	
68 - 73	Д	2,33	Удовлетворительно
61 – 67	E	2,0	
31 - 60	FX	0	Неудовлетворительно
0 - 30	F	0	

В соответствии с действующими нормативными актами и рекомендациями Министерства образования и науки КР устанавливаются следующие критерии выставления оценок на экзаменах:

- оценка ***"отлично"*** выставляется студенту, который обнаружил на экзамене всестороннее, систематическое и глубокое знание учебно-программного материала, умение свободно выполнять задания, предусмотренные программой, который усвоил основную литературу и ознакомился с дополнительной литературой, рекомендованной программой. Как правило, оценка "отлично" выставляется студентам, усвоившим взаимосвязь основных понятий дисциплины и их значений для приобретаемой профессии, проявившим творческие способности в понимании, изложении и использовании учебно-программного материала;

- оценка ***"хорошо"*** выставляется студенту, который на экзамене обнаружил полное знание учебно-программного материала, успешно выполнил предусмотренные в программе задания, усвоил основную литературу, рекомендованную в программе. Как правило, оценка "хорошо" выставляется студентам, показавшим систематический характер знаний по дисциплине и способным к их самостоятельному выполнению и обновлению в ходе дальнейшей учебной работы и профессиональной деятельности;

- оценка ***"удовлетворительно"*** выставляется студенту, обнаружившему знание основного учебного материала в объеме, необходимом для дальнейшей учебы и предстоящей работы по профессии, справляющемуся с выполнением заданий, предусмотренных программой, который ознакомился с основной литературой, рекомендованной программой. Как правило, оценка "удовлетворительно" выставляется студентам, допустившим погрешности в ответе на экзамене и при выполнении экзаменационных заданий, но обладающим необходимыми знаниями для их устранения под руководством преподавателя;

- оценка ***"неудовлетворительно"*** выставляется студенту, обнаружившему пробелы в знаниях основного учебно-программного материала, допустившему принципиальные ошибки в выполнении предусмотренных программой заданий, не ознакомившемуся с основной литературой, предусмотренной программой, и не овладевшему базовыми знаниями, предусмотренными по данной дисциплине и определенными соответствующей программой дисциплины.

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ

КЫРГЫЗСКОЙ РЕСПУБЛИКИ

ОШСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ МАТЕМАТИКИ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

КАФЕДРА АСЦТ

«Утверждена»
на заседании кафедры АСЦТ
от 28 августа 2020 года, протокол №1
Зав. каф. АСЦТ, доцент: Молдояров У.

ФОНД ОЦЕНОЧНЫХ СРЕДСТВ ПО УЧЕБНОЙ ДИСЦИПЛИНЕ

Технология разработки программного обеспечения

НАПРАВЛЕНИЕ: 710100 “ИВТ”

КВАЛИФИКАЦИЯ (СТЕПЕНЬ) ВЫПУСКА - магистр

ПАСПОРТ

фонда оценочных средств по дисциплине «Технология разработки программного обеспечения»

1. *Ожидаемые результаты и компетенции*

Процесс изучения дисциплины «ТРПО» направлен на формирования следующих **универсальных** компетенций:

общенаучными (ОК):

✓ способен глубоко понимать и критически оценивать новейшие теории, методы и способы, использовать междисциплинарный подход и интегрировать достижения различных наук для приобретения новых знаний (ОК-1);

- **инструментальными (ИК):**

способен к дальнейшему самостоятельному обучению (ИК-4);

- **социально-личностными и общекультурными (СЛК):**

- способен руководить коллективом, в том числе междисциплинарными проектами, влиять на формирование целей команды, воздействовать на ее социально-психологический климат в нужном для достижении целей направлении, корректно оценивать качество результатов деятельности (СЛК-4);

профессиональными компетенциями (ПК):

✓ способен анализировать и систематизировать научно-техническую информацию по теме исследования, выбирать методики и средства решения задач (ПК-2);

✓ способен к постановке и модернизации отдельных лабораторных работ и практикумов по дисциплинам направления (ПК-6);

✓ способен готовить технических заданий на разработку проектных решений (ПК-8);

✓ способен проектировать и применять инструментальные средства реализации программно-аппаратных проектов (ПК-13).

Перечень оценочных средств

№	Вид деятельности	Определения	Примечание
1.	**Парная работа**	***Парная работа*** – взаимодействие участников учебного процесса в парах сменного состава («обучая — учусь»)	Используется в различных формах деятельности
2.	**Групповая работа**	***Групповая работа*** – это взаимодействие в малых группах, где обсуждение происходит с каждым и решения принимаются сообща	Используется в различных формах деятельности
3.	**Работа с контрольными карточками**	***Контрольные карточки*** могут содержать вопросы, оформленные в виде тестовых заданий, вопросов соответствий, контрольных заданий и др.	
4.	**Исследователь-ская работа**	Исследовательская работа – продукт самостоятельной работы студента, представляющий собой решение определенной учебно-практической научной темы	
5.	**Элективный (выборочный) тест**	Тест – система стандартизированных заданий, позволяющая автоматизировать процедуру измерения уровня знаний и умений обучающегося.	
6.	**Выполнение практических и лабораторных заданий**	Практическое (лабораторное) задание – задание задаваемое преподавателем студенту для самостоятельного выполнения, в целях усвоения пройденного материала	Выполняется в тетрадях
7.	**Конспектирование**	***Конспект*** (лат. conspectus — обозрение, обзор, очерк) — краткое изложение содержания нового материала	в тетради
8.	**Выполнение реферата**	***Реферат*** – продукт самостоятельной работы студента по определённой теме, в котором собрана информация из одного или нескольких источников, представляющий собой краткое изложение в письменном виде.	На листах формата А4, объемом 2-15 листа
9.	**Авторизация системы**	В информационных технологиях посредством ***авторизации*** устанавливаются права доступа к информационным ресурсам и системам	

		обработки данных.	
10.	**Установка программного обеспечения;**	***Установка программного обеспечения, инсталляция*** — процесс установки программного обеспечения (Pascal ABC) на компьютер конечного пользователя	
11.	**Презентация проектной работы**	***Презентация***, на основе программного обеспечения, для демонстрации конечного продукта, сопровождается устным пояснением	Представление прак-го рез-та. на основе пройденных материалов
12.	**Графические (визуальные) организаторы**	Генерирование, визуализация, структурирование идей в процессе изучения и организации информации, разрешения проблем, принятия решений	Диаграммы Венна, Т-диаграммы, графические органайзеры сравнения и сопоставления и т.д.
13.	**Подготовка портфолио**	Целевая подборка работ студента, раскрывающая его индивидуальные образовательные достижения в одной или нескольких учебных дисциплинах.	используется весь семестр
14.	**Дополни предложение**	Дополняя предложение, учащиеся активизируют предварительные знания	
15.	**Составление кластера**	***Кластер*** представляется, как подмножество результатов поиска, связанных единством темы	понятие используется в итоговых занятиях

Критерии оценивания

№	Вид деятельности	Критерии оценивания	Баллы
1.	**Дополни предложение**	1. Дополняя предложение, учащиеся активизируют предварительные знания 2. Повторение освоенных понятий и терминов, освоение новых 3. Воспроизведение 4. Продумайте ключевые слова по теме. Это должно быть слово, которое может стать подлежащим в предложении и значение которого известно учащимся, то есть это не должно быть абстрактным понятием.	• *Полный объем работы: 5 б;* • *Неполный: 3 б;* • *Минимальный: 1 б*

2.	**Графические организаторы для активизации: мишень практика**	1. Учащиеся заполняют мишень идеями и именами людьми, которые относятся к теме урока 2. Запоминание и понимание 3. Письмо (короткие записи) и говорение (неформальное) 4. Раздайте каждому учащемуся рисунок мишени	в практических уроках используется
3.	**Составление кластера**	1. Приобретение новых знаний с большой степенью самостоятельности; 2. Восприятие, обобщение и анализ итоговой информации; 3. Выделение и объединение однородных элементов, связанных единством темы; 4. Грамотное составление наглядной схемы; 5. Использование различных методов построений схем 6. Обоснование принимаемых решений	• *Полный объем работы: 5 б;* • *Неполный: 3 б;* • *Минимальный: 1 б*

4.1. Критерии оценки на экзамене

Выставление оценок на экзаменах осуществляется на основе принципов объективности, справедливости, всестороннего анализа качества знаний студентов, и других положений, способствующих повышению надежности оценки знаний обучающихся и устранению субъективных факторов.

В соответствии с действующими нормативными актами и рекомендациями Министерства образования и науки КР устанавливаются следующие критерии выставления оценок на экзаменах по гуманитарным, естественным, техническим и другим дисциплинам:

- оценка **"отлично"** выставляется студенту, который обнаружил на экзамене всестороннее, систематическое и глубокое знание учебно-программного материала, умение свободно выполнять задания, предусмотренные программой, который усвоил основную литературу и ознакомился с дополнительной литературой, рекомендованной программой. Как правило, оценка "отлично" выставляется студентам, усвоившим взаимосвязь основных понятий дисциплины и их значений для приобретаемой профессии,

проявившим творческие способности в понимании, изложении и использовании учебно-программного материала;

- оценка **"хорошо"** выставляется студенту, который на экзамене обнаружил полное знание учебно-программного материала, успешно выполнил предусмотренные в программе задания, усвоил основную литературу, рекомендованную в программе. Как правило, оценка "хорошо" выставляется студентам, показавшим систематический характер знаний по дисциплине и способным к их самостоятельному выполнению и обновлению в ходе дальнейшей учебной работы и профессиональной деятельности;

- оценка **"удовлетворительно"** выставляется студенту, обнаружившему знание основного учебного материала в объеме, необходимом для дальнейшей учебы и предстоящей работы по профессии, справляющемуся с выполнением заданий, предусмотренных программой, который ознакомился с основной литературой, рекомендованной программой. Как правило, оценка "удовлетворительно" выставляется студентам, допустившим погрешности в ответе на экзамене и при выполнении экзаменационных заданий, но обладающим необходимыми знаниями для их устранения под руководством преподавателя;

- оценка **"неудовлетворительно"** выставляется студенту, обнаружившему пробелы в знаниях основного учебно-программного материала, допустившему принципиальные ошибки в выполнении предусмотренных программой заданий, не ознакомившемуся с основной литературой, предусмотренной программой, и не овладевшему базовыми знаниями, предусмотренными по данной дисциплине и определенными соответствующей программой курса (перечень основных знаний и умений, которыми должны овладеть магистранты, является обязательным элементом рабочей программы курса).

«ТЕХНОЛОГИЯ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ»

Существует множество различных процессов для создания ПО. Тем не менее, технологий, рассматривающих полный жизненный цикл проекта разработки ПО, сочетающих в себе научный подход, серьезную базу исследований и имеющих историю реального использования и адаптации, относительно немного. За несколько десятилетий эволюции аппаратное обеспечение значительно усовершенствовалось. Вычислительные мощности, которые еще десять-пятнадцать лет назад могли себе позволить лишь немногие научные учреждения и обслуживание которых требовало целого штата специалистов, сегодня доступны практически каждому инженеру. Однако эти мощности требуют соответствующего программного обеспечения. И именно в этой области, несмотря на то что аппаратные ресурсы стали значительно более доступны, наблюдаются значительные проблемы. [25] Так, по данным американских исследователей, в 80-е только 14% проектов по созданию ПО завершались успешно. Но и сегодня - после нескольких десятилетий эволюции языков программирования, инструментальных средств разработки, при практически неограниченном (по сравнению с 70-ми и 80-ми) машинном времени - процент успешно завершенных проектов составляет всего 26%. В СССР достижения в области производства ПО были значительно лучшими. Тому способствовали следующие

[redacted]

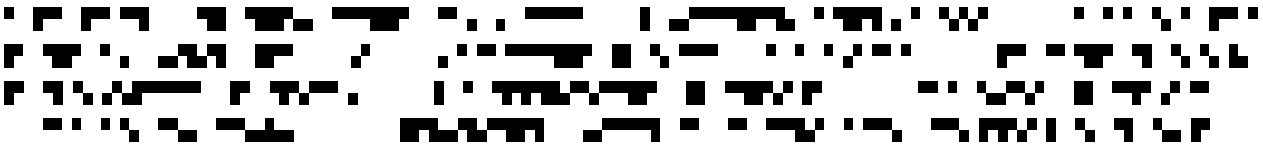
не на удовлетворение требований заказчика, а на удовлетворение изначально

[redacted]

проекты в основном ориентировались на ВПК, бюджеты были фактически не ограниченными (по сегодняшним меркам). Но по ряду причин советская школа разработки ПО прекратила свое развитие и многие достижения были утрачены. В рыночных условиях 7 (быстро меняющиеся требования, ограниченные бюджеты, ориентация на результат, острая конкуренция за высококвалифицированный персонал) использование старых наработок советской школы оказалось ограничено очень узкими областями. 1.1. Технологии Термин «технология» – он подчеркивает аналогию между созданием программного продукта и промышленным производством. Он отражает современную тенденцию к вводу дисциплины, организации и инструментирования в такой творческий процесс, как программирование. Слово фиксирует ту точку зрения, что программирование, несмотря на интеллектуальность и творческий характер этой деятельности, нуждается в организации и регламентировании, наборе соглашений и правил, не говоря уже об инструментальном обеспечении. Сейчас это кажется тривиальным, но в 60-е годы такую точку приходилось отстаивать. Да и сейчас порой возникают трения на почве регламентирования деятельности разработчиков. Сам русский термин «технология программирования» был введен русским академиком Андреем Петровичем Ершовым. Он трактовал термин «программирование» в обобщенном виде и подразумевал все виды деятельности, выполняемые в ходе создания программных систем. На западе для определения этой деятельности использовался термин «engineering». Сейчас обобщённый термин, применимый к созданию программных средств, обозначают как «разработка» или «конструирование». Справедлива формула: разработка = анализ + проектирование + программирование (кодирование) + тестирование + отладка Иногда сюда также включают “сопровождение”. Чтобы подчеркнуть промышленно-производственный аспект, говорят о “технологии разработки” или “технологии конструирования”. 1.2. Этапы развития Лишь в начале 90-х Британское сообщество вычислительной техники (British Computer Society) начало присваивать разработчикам программ звание инженера. В США только в 1998 году стало возможным хоть где-то зарегистрироваться в качестве профессионального инженера программного обеспечения. Но по-прежнему, даже в начале нынешнего века, общепризнанным остается тот факт, что

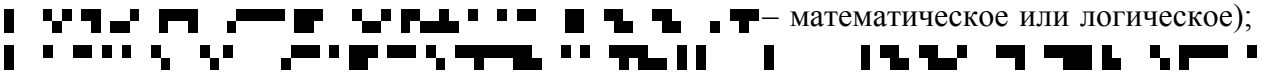
разработке программного обеспечения не достает достаточно развитой научной базы. По 8 некоторым оценкам, 75% организаций, занимающихся разработкой программ, делают это на примитивном уровне. С момента зарождения технология разработки программ испытала несколько подъемов в своем развитии. Один из них связан с публикацией письма Эдстера Дийкстры (Edsger Dijkstra) в Ассоциацию вычислительной техники, озаглавленного так: «О вреде использования операторов GOTO» (GOTO statement considered harmful, 1968). В те времена программы писались с активным использованием операторов безусловного перехода. Обращая внимание на недостатки таких программ, Дийкстра предложил свою концепцию структурного программирования, позволяющую избежать использования таких операторов. Концепция Дийкстры основывалась на том наблюдении Бема и Якоби (Flow Diagrams, Turning Machines and languages with only two formation rules, 1966), что для записи любой программы в принципе достаточно только трех конструкций управления – последовательного выполнения, ветвления и цикла. То есть теоретически необходимость в использовании операторов перехода отсутствует [24, 25]. Следующий шаг в развитии структурного программирования связан с введением аппарата функций, позволяющих разбивать структурную программу на обозримые по своим размерам части. При таком подходе программа пишется в терминах вызова функций верхнего уровня, которые реализуются при помощи функций более низкого уровня. Нисходящее программирование еще так же называли модульным программированием. Структурным программам недоставало одного важного свойства – в их структуре непосредственно не отображалась сущность предметной области. Из-за этого было трудно их модифицировать в условиях изменяющихся требований. Позднее возникла парадигма объектной ориентированности. Она основана на использовании объектов, объединяющих в себе данные и функциональность. На ОО парадигме основаны многие современные языки и системы программирования. 1.3. Методы проектирования Говорят, что Генри Форд совершил революцию в производстве автомобилей, когда заметил, что узлы автомобиля можно стандартизировать, так что при сборке автомобилей данной модели можно будет использовать любой экземпляр требуемого узла. Столь же важным в настоящее время признается возможность при разработке одних приложений заимствовать идеи, архитектуру, проект и исходный код других приложений. Если приложения проектируются 9 таким образом, что различные их части могут быть использованы многократно, то в конечном итоге это приводит к уменьшению стоимости разработки приложений. Однако, чтобы это было возможным, приложения должны быть модульными. Модульность приложения, собственно, и означает, что оно состоит из легко идентифицируемых и заменяемых частей. По этой причине при правильном проектировании программного продукта особое внимание должно уделяться модульности, особенно на стадии разработки архитектуры. К формальным методам проектирования относятся те методы, которые основаны на математике. Формальные методы помогают решить задачи обеспечения надежности программ. Они могут быть применены как при анализе требований для обеспечения точности формулировки требований, так и в процессе реализации для обеспечения соответствия кода программы сформулированным требованиям. Как правило формальные методы используют математику в ее логическом аспекте. В вычислительном же аспекте математика задействована в связи с использованием метрик, которые мы будем рассматривать далее. Сегодня существует огромное количество различных процессов для создания ПО. Тем не менее, именно технологий, рассматривающих полный жизненный цикл проекта разработки ПО, сочетающих в себе научный подход, серьезную базу исследований и имеющих историю реального использования и адаптации, относительно немного. Из методологий и технологий, получивших определенное признание на данный момент, можно назвать следующие: Datarun, CMM, Microsoft Solution Framework (MSF), Oracle Method, Rational Unified Process (RUP), SADT (IDEFx). Особое место в этом списке

занимает технология компании Rational Software. В ее методологии применен наиболее современный процессноориентированный подход: так как разработка ПО является производством, то, как и на всяком производстве, при выявлении проблем в продукции (симптомов) необходимо корректировать процесс (устранять причины). Особенностью этой технологии является то, что в ее создании участвуют ведущие методисты в области разработки ПО, такие как Г. Буч (ООАП), Дж. Рамбо (ОМТ), А. Джекобсон (Objectory), внесшие весомый вклад в теорию и практику разработки современного ПО. Кроме того, следует заметить, что эта технология развивалась и проходила проверку с участием военного ведомства США [2, 24]. 1.4. Этапы и элементы процесса разработки 10 В 80-е и 90-е в области разработки ПО преобладали две тенденции. Одна – это быстрый рост приложений, в том числе создаваемых для Web. Другая – расцвет инструментальных средств и парадигм (подходов к проектированию). Несмотря на появление новых

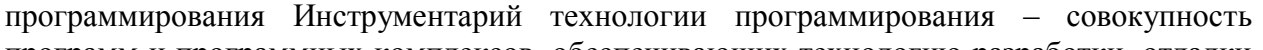


Сопровождение продукта. Разработчики меняют последовательность проработки этих направлений. В реальности разработка ПО обычно определяется требуемым набором функций или сроком сдачи проекта. В результате, только хорошо организованные группы инженеров, владеющих методами разработки ПО, способны правильно построить работу. В противном случае разработчиков обычно ожидает хаос. Система разработки ПО включает в себя 4 “П” (Персонал, процесс, проект, продукт)[15, 24]. Персонал – те, кем это делается. Команда разработчиков наилучшим образом работает, если каждый участник знает, что он должен делать, и имеет определенные обязанности. Другая сторона аспекта персонала – это заинтересованные в проекте лица: заказчиками, пользователи и инвесторы. В любом производстве результаты определяются используемой технологией. В силу специфичности производства ПО (практически нулевая стоимость тиражирования, очень быстрое моральное старение и т.д.) технология его создания сильно зависит от качества команды разработчиков, поэтому должна включать в себя организационный и управленческий аспекты. Процесс – способ, которым это делается. Выделяют: водопадный процесс, итеративный процесс, XP. Индивидуальный процесс разработки (Personal Software Process), командный процесс разработки (Team Software Process). Модель зрелости возможностей (Capability Maturity Model) для оценки возможностей команды разработчиков. Проект – совокупность действий, необходимая для создания артефакта. Проект включает контакт с заказчиком, написание 11 документации, проектирование, написание кода и тестирование продукта. Продукт – это не только программное обеспечение, но и все составляющие его артефакты. Под артефактами понимается объектные модули, исходный код, документация, результаты тестов и измерений продуктивности. Качество приложения должны удовлетворять заранее определенному уровню качества. Для достижения требуемого уровня качества пр

ориентированный на команды разработчиков. Он применяется на всех этапах разработки);



– математическое или логическое);



программирования Инструментарий технологии программирования – совокупность программ и программных комплексов, обеспечивающих технологию разработки, отладки и внедрения создаваемых программных продуктов. Рис.1.1. Группы программных продуктов 1.5.1. Средства для создания приложений 12 Средства для создания приложений – локальные средства, обеспечивающие выполнение отдельных видов работ

инструментальная среда пользователя. Язык программирования – формализованный язык

машинные языки – языки программирования, воспринимаемые аппаратной частью комп[illegible]-ориентированные языки – языки программирования, которые отражают структуру конкретного типа компьютера [illegible] – не зависящие от архитектуры компьютера языки программирования для отражения структуры алгоритма (Паскаль, бейсик, Фортран [illegible]–ориентированные языки – языки программирования, где имеется [illegible] проблемно–ориентированные языки – предназначены для решения задач определенного класса (Lisp); Другой классификацией языков является их деление на языки, ориентированные па реализацию основ структурного программирования, основанного на модульной структуре программного продукта и типовых управляющих структурах алгоритмов обработки данных различных программных модулей, и объектно-ориентированные языки, поддерживающие понятие объектов, их свойств и методов [illegible] интегрированную среду разрабо[illegible] [illegible] ения продуктами программного комплекса. 13 Компилятор транслирует всю программу без ее выполнения. Трансляторы (интерпретаторы) выполняют пооперационную обработку и выполнение программы. Отладчики – специальные программы, предназначенные для трассировки и анализа выполнения других программ. Трассировка – это обеспечение выполнения в пооператорном варианте. Инструментальная среда пользователя – это специальные [illegible] процедур, объе[illegible] [illegible] Интегрированные среды разработки программ объединяют набор средств для их комплексного применения на технологических этапах создания программы. 1.5.2. Средства для создания информационных систем (Case– технология) CASE-технология (CASE – Computer-Aided System Engineering) – программный комплекс, автоматизирующий весь технологический процесс анализа, проектирования, разработки и сопровождения сложных программных систем. Средства CASE-[illegible] встроенные в систему реализации – все решения по проектированию и реализации привязки к выбранной [illegible]– все решения по проектированию ориентированы на унификацию (определение) начальных этапов жизненного цикла программы и средств их документирования, обеспечивают большую гибкость в выборе средств реализации. Основное достоинство CASE-технологии – это поддержка коллективной работы над проектом за счет возможности работы в локальной сети разработчиков, экспорта (импорта) любых фрагментов проекта, организованного управления проектами. В некоторых CASE системах поддерживается создание каркаса программ и создание полного продукта.

Вопросы и задания для самоконтроля

1. Что такое технология разработки ПО?
2. Что явилось предпосылкой становления дисциплины «Технология разработки ПО»? Что явилось причиной стремительного развития ПО?
3. Чем отличаются программа и программное обеспечение?
4. Достаточно ли при работе над проектом большой программной системы быть компетентным в области вычислительной техники и программировании. Почему?
5. Может ли большая программная система быть отлажена до конца и почему?

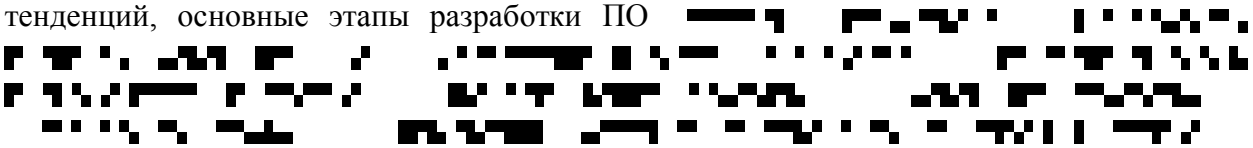
6. При каких условиях созданный программный комплекс может быть назван программным продуктом?
7. Что такое системное программное обеспечение?
8. Что такое инструментарий технологии программирования.

Этапы развития

Лишь в начале 90-х Британское сообщество вычислительной техники (British Computer Society) начало присваивать разработчикам программ звание инженера. В США только в 1998 году стало возможным хоть где-то зарегистрироваться в качестве профессионального инженера программного обеспечения. Но по-прежнему, даже в начале нынешнего века, общепризнанным остается тот факт, что разработке программного обеспечения не достает достаточно развитой научной базы. По 8 некоторым оценкам, 75% организаций, занимающихся разработкой программ, делают это на примитивном уровне. С момента зарождения технология разработки программ испытала несколько подъемов в своем развитии. Один из них связан с публикацией письма Эдстера Дийкстры (Edsger Dijkstra) в Ассоциацию вычислительной техники, озаглавленного так: «О вреде использования операторов GOTO» (GOTO statement considered harmful, 1968). В те времена программы писались с активным использованием операторов безусловного перехода. Обращая внимание на недостатки таких программ, Дийкстра предложил свою концепцию структурного программирования, позволяющую избежать использования таких операторов. Концепция Дийкстры основывалась на том наблюдении Бема и Якоби (Flow Diagrams, Turning Machines and languages with only two formation rules, 1966), что для записи любой программы в принципе достаточно только трех конструкций управления – последовательного выполнения, ветвления и цикла. То есть теоретически необходимость в использовании операторов перехода отсутствует. Следующий шаг в развитии структурного программирования связан с введением аппарата функций, позволяющих разбивать структурную программу на обозримые по своим размерам части. При таком подходе программа пишется в терминах вызова функций верхнего уровня, которые реализуются при помощи функций более низкого уровня. Нисходящее программирование еще так же называли модульным программированием. Структурным программам недоставало одного важного свойства – в их структуре непосредственно не отображалась сущность предметной области. Из-за этого было трудно их модифицировать в условиях изменяющихся требований. Позднее возникла парадигма объектной ориентированности. Она основана на использовании объектов, объединяющих в себе данные и функциональность. На ОО парадигме основаны многие современные языки и системы программирования.

Методы проектирования Генри Форд совершил революцию в производстве автомобилей, когда заметил, что узлы автомобиля можно стандартизировать, так что при сборке автомобилей данной модели можно будет использовать любой экземпляр требуемого узла. Столь же важным в настоящее время признается возможность при разработке одних приложений заимствовать идеи, архитектуру, проект и исходный код других приложений. Если приложения проектируются таким образом, что различные их части могут быть использованы многократно, то в конечном итоге это приводит к уменьшению стоимости разработки приложений. Однако, чтобы это было возможным, приложения должны быть модульными. Модульность приложения, собственно, и означает, что оно состоит из легко идентифицируемых и заменяемых частей. По этой причине при правильном проектировании программного продукта особое внимание должно уделяться модульности, особенно на стадии разработки архитектуры. К формальным методам проектирования относятся те методы, которые основаны на математике. Формальные методы помогают решить задачи обеспечения надежности программ. Они могут быть применены как при анализе требований для обеспечения точности формулировки

требований, так и в процессе реализации для обеспечения соответствия кода программы сформулированным требованиям. Как правило формальные методы используют математику в ее логическом аспекте. В вычислительном же аспекте математика задействована в связи с использованием метрик, которые мы будем рассматривать далее. Сегодня существует огромное количество различных процессов для создания ПО. Тем не менее, именно технологий, рассматривающих полный жизненный цикл проекта разработки ПО, сочетающих в себе научный подход, серьезную базу исследований и имеющих историю реального использования и адаптации, относительно немного. Из методологий и технологий, получивших определенное признание на данный момент, можно назвать следующие: Datarun, CMM, Microsoft Solution Framework (MSF), Oracle Method, Rational Unified Process (RUP), SADT (IDEFx). Особое место в этом списке занимает технология компании Rational Software. В ее методологии применен наиболее современный процессноориентированный подход: так как разработка ПО является производством, то, как и на всяком производстве, при выявлении проблем в продукции (симптомов) необходимо корректировать процесс (устранять причины). Особенностью этой технологии является то, что в ее создании участвуют ведущие методисты в области разработки ПО, такие как Г. Буч (ООАП), Дж. Рамбо (OMT), А. Джекобсон (Objectory), внесшие весомый вклад в теорию и практику разработки современного ПО. Кроме того, следует заметить, что эта технология развивалась и проходила проверку с участием военного ведомства США. 1.4. Этапы и элементы процесса разработки 10 В 80-е и 90-е в области разработки ПО преобладали две тенденции. Одна – это быстрый рост приложений, в том числе создаваемых для Web. Другая – расцвет инструментальных средств и парадигм (подходов к проектированию). Несмотря на появление новых тенденций, основные этапы разработки ПО



Сопровождение продукта. Разработчики меняют последовательность проработки этих направлений. В реальности разработка ПО обычно определяется требуемым набором функций или сроком сдачи проекта. В результате, только хорошо организованные группы инженеров, владеющих методами разработки ПО, способны правильно построить работу. В противном случае разработчиков обычно ожидает хаос. Система разработки ПО включает в себя «Персонал, процесс, проект, продукт». Персонал – те, кем это делается. Команда разработчиков наилучшим образом работает, если каждый участник знает, что он должен делать, и имеет определенные обязанности. Другая сторона аспекта персонала – это заинтересованные в проекте лица: заказчиками, пользователи и инвесторы. В любом производстве результаты определяются используемой технологией. В силу специфичности производства ПО (практически нулевая стоимость тиражирования, очень быстрое моральное старение и т.д.) технология его создания сильно зависит от качества команды разработчиков, поэтому должна включать в себя организационный и управленческий аспекты. Процесс – способ, которым это делается. Выделяют: водопадный процесс, итеративный процесс, XP. Индивидуальный процесс разработки (Personal Software Process), командный процесс разработки (Team Software Process). Модель зрелости возможностей (Capability Maturity Model) для оценки возможностей команды разработчиков. Проект – совокупность действий, необходимая для создания артефакта. Проект включает контакт с заказчиком, написание 11 документации, проектирование, написание кода и тестирование продукта. Продукт – это не только программное обеспечение, но и все составляющие его артефакты. Под артефактами понимается объектные модули, исходный код, документация, результаты тестов и измерений продуктивности. Качество – приложения должны удовлетворять заранее определенному уровню качества. Для достижения требуемого уровня качества

ориентированный на команды разработчиков. Он применяется на всех этапах разработки); – математическое или логическое); программирования Инструментарий технологии программирования – совокупность программ и программных комплексов, обеспечивающих технологию разработки, отладки и внедрения создаваемых программных продуктов. Группы программных продуктов. Средства для создания приложений. Средства для создания приложений – локальные средства, обеспечивающие выполнение отдельных видов работ по созданию программ, пользователя. Язык программирования – формализованный язык для описания алгоритма решения задачи на компьютере. – языки программирования, воспринимаемые аппаратной частью компьютера (машинные коды); –ориентированные языки – языки программирования, которые отражают алгоритмические языки – не зависящие от архитектуры компьютера языки программирования для отражения –ориентированные языки – языки программирования, где имеется возможность описания программы как –ориентированные языки – предназначены для решения задач определенного класса (Lisp); Другой классификацией языков является их деление на языки, ориентированные па реализацию основ структурного программирования, основанного на модульной структуре программного продукта и типовых управляющих структурах алгоритмов обработки данных различных программных модулей, и объектно-ориентированные языки, поддерживающие понятие объектов, их свойств и методов обработки блиотеками, текстовыми и продуктами программного комплекса.

Компилятор транслирует всю программу без ее выполнения. Трансляторы (интерпретаторы) выполняют пооперационную обработку и выполнение программы. Отладчики – специальные программы, предназначенные для трассировки и анализа выполнения других программ. Трассировка – это обеспечение выполнения в пооператорном варианте. Инструментальная среда пользователя – это специальные средст

Интегрированные среды разработки программ объединяют набор средств для их комплексного применения на технологических этапах создания программы.

Средства для создания информационных систем (Case– технология) CASE-технология (CASE – Computer-Aided System Engineering) – программный комплекс, автоматизирующий весь технологический процесс анализа, проектирования, разработки и сопровождения сложных программных систем. Средства CASE- встроенные в систему реализации – все решения по проектированию и реализации – все решения по проектированию ориентированы на унификацию (определение) начальных этапов жизненного цикла программы и средств их документирования, обеспечивают большую гибкость в выборе средств реализации. Основное достоинство CASE-технологии – это поддержка коллективной работы над проектом за счет возможности работы в локальной

сети разработчиков, экспорта (импорта) любых фрагментов проекта, организованного управления проектами. В некоторых CASEсистемах поддерживается создание каркаса программ и создание полного продукта.

Этапы и элементы процесса разработки

В 80-е и 90-е в области разработки ПО преобладали две тенденции. Одна – это быстрый рост приложений, в том числе создаваемых для Web. Другая – расцвет инструментальных средств и парадигм (подходов к проектированию). Несмотря на появление новых проце

Сопровождение продукта. Разработчики меняют последовательность проработки этих направлений. В реальности разработка ПО обычно определяется требуемым набором функций или сроком сдачи проекта. В результате, только хорошо организованные группы инженеров, владеющих методами разработки ПО, способны правильно построить работу. В противном случае разработчиков обычно ожидает хаос. Система разработки ПО включает в себя 4 “П” (Персонал, процесс, проект, продукт)[15, 24]. Персонал – те, кем это делается. Команда разработчиков наилучшим образом работает, если каждый участник знает, что он должен делать, и имеет определенные обязанности. Другая сторона аспекта персонала – это заинтересованные в проекте лица: заказчиками, пользователи и инвесторы. В любом производстве результаты определяются используемой технологией. В силу специфичности производства ПО (практически нулевая стоимость тиражирования, очень быстрое моральное старение и т.д.) технология его создания сильно зависит от качества команды разработчиков, поэтому должна включать в себя организационный и управленческий аспекты. Процесс – способ, которым это делается. Выделяют: водопадный процесс, итеративный процесс, XP. Индивидуальный процесс разработки (Personal Software Process), командный процесс разработки (Team Software Process). Модель зрелости возможностей (Capability Maturity Model) для оценки возможностей команды разработчиков. Проект – совокупность действий, необходимая для создания артефакта. Проект включает контакт с заказчиком, написание 11 документации, проектирование, написание кода и тестирование продукта. Продукт – это не только программное обеспечение, но и все составляющие его артефакты. Под артефактами понимается объектные модули, исходный код, документация, результаты тестов и измерений продуктивности. Качество – приложения должны удовлетворять заранее определенному уровню качества. Для достижения требуемого уровня качества ориентированный на команды разработчиков. Он применяется на всех этапах разработки); – математическое или 2 технологии программирования Инструментарий технологии программирования – совокупность программ и программных комплексов, обеспечивающих технологию разработки, отладки и внедрения создаваемых программных продуктов. Рис.1.1. Группы программных продуктов 1.5.1. Средства для создания приложений 12 Средства для создания приложений – локальные средства, обеспечивающие выполнение отдельных инструментальная среда пользователя. Язык программирования – формализованный язык для описания алгоритма решения задачи на компьютере. Он машинные языки – языки программирования, воспринимаемые аппаратной частью -ориентированные языки – языки программирования, которые отражают структуру конкретного типа компьютера

[redacted]горитмические языки – не зависящие от архитектуры компьютера языки программирования для отражения структуры алгоритма (Паскаль, бейсик, Фортран [redacted]–ориентированные языки – языки программирования, где имеется возможность описания программы [redacted] проблемно–ориентированные языки – предназначены для решения задач определенного класса (Lisp); Другой классификацией языков является их деление на языки, ориентированные па реализацию основ структурного программирования, основанного на модульной структуре программного продукта и типовых управляющих структурах алгоритмов обработки данных различных программных модулей, и объектно-ориентированные языки, поддерживающие понятие объектов, их свойств и методов обработки [[redacted] средства (утилиты) (для работ[redacted] комплекса. 13 Компилятор транслирует всю программу без ее выполнения. Трансляторы (интерпретаторы) выполняют пооперационную обработку и выполнение программы. Отладчики – специальные программы, предназначенные для трассировки и анализа выполнения других программ. Трассировка – это обеспечение выполнения в пооператорном варианте. Инструментальная среда пользователя – это специальные [redacted] Интегрированные среды разработки программ объединяют набор средств для их комплексного применения на технологических этапах создания программы. 1.5.2. Средства для создания информационных систем (Case– технология) CASE-технология (CASE – Computer-Aided System Engineering) – программный комплекс, автоматизирующий весь технологический процесс анализа, проектирования, разработки и сопровождения сложных программных систем. Средства CASE-технологий делятся [redacted] встроенные в систему реализации – все решения по проектированию и реализации [redacted]– все решения по проектированию ориентированы на унификацию (определение) начальных этапов жизненного цикла программы и средств их документирования, обеспечивают большую гибкость в выборе средств реализации. Основное достоинство CASE-технологии – это поддержка коллективной работы над проектом за счет возможности работы в локальной сети разработчиков, экспорта (импорта) любых фрагментов проекта, организованного управления проектами. В некоторых CASEсистемах поддерживается создание каркаса программ и создание полного продукта.

Вопросы и задания для самоконтроля

1. Что такое технология разработки ПО?
2. Что явилось предпосылкой становления дисциплины «Технология разработки ПО»? Что явилось причиной стремительного развития ПО?
3. Чем отличаются программа и программное обеспечение?
4. Достаточно ли при работе над проектом большой программной системы быть компетентным в области вычислительной техники и программировании. Почему?
5. Может ли большая программная система быть отлажена до конца и почему? 6. При каких условиях созданный программный комплекс может быть назван программным продуктом?
7. Что такое системное программное обеспечение?
8. Что такое инструментарий технологии программирования?

Инструментарий технологии программирования

Инструментарий технологии программирования – совокупность программ и программных комплексов, обеспечивающих технологию разработки, отладки и внедрения создаваемых программных продуктов. Рис.1.1. Группы программных продуктов 1.5.1. Средства для создания приложений 12 Средства для создания приложений – локальные средства, обеспечивающие выполнение отдельных видов работ по созданию программ, делятся на: [illegible] программирования – формализованный язык для описания алгоритма решения задачи на [illegible] – языки программирования, [illegible] ориентированные языки – языки программирования, которые отражают структуру [illegible] – не зависящие от архитектуры компьютера языки программирования для отражения структуры алгоритма [illegible] –ориентированные языки – языки программирования, где имеется возможность описания программы как совокупности [illegible] –ориентированные языки – предназначены для решения задач определенного класса (Lisp); Другой классификацией языков является их деление на языки, ориентированные па реализацию основ структурного программирования, основанного на модульной структуре программного продукта и типовых управляющих структурах алгоритмов обработки данных различных программных модулей, и объектно-ориентированные языки, поддерживающие понятие объектов, их свойств и методов обработки [4, 24]. Системы программирования включают: [illegible] продуктами программного комплекса. 13 Компилятор транслирует всю программу без ее выполнения. Трансляторы (интерпретаторы) выполняют пооперационную обработку и выполнение программы. Отладчики – специальные программы, предназначенные для трассировки и анализа выполнения других программ. Трассировка – это обеспечение выполнения в пооператорном варианте. Инструментальная среда пользователя – это специальные средства, встроенные в пакеты прикладных программ, такие, как [illegible] др. Интегрированные среды разработки программ объединяют набор средств для их комплексного применения на технологических этапах создания программы. 1.5.2. Средства для создания информационных систем (Case– технология) CASE-технология (CASE – Computer-Aided System Engineering) – программный комплекс, автоматизирующий весь технологический процесс анализа, проектирования, разработки и сопровождения сложных программных систем. Средства CASE- [illegible] встроенные в систему реализации – все решения по проектированию и реализации [illegible] – все решения по проектированию ориентированы на унификацию (определение) начальных этапов жизненного цикла программы и средств их документирования, обеспечивают большую гибкость в выборе средств реализации. Основное достоинство CASE-технологии – это поддержка коллективной работы над проектом за счет возможности работы в локальной сети разработчиков, экспорта (импорта) любых фрагментов проекта, организованного управления проектами. В некоторых CASEсистемах поддерживается создание каркаса программ и создание полного продукта. 14

Вопросы и задания для самоконтроля

1. Что такое технология разработки ПО?
2. Что явилось предпосылкой становления дисциплины «Технология разработки ПО»? Что явилось причиной стремительного развития ПО?
3. Чем отличаются программа и программное обеспечение?
4. Достаточно ли при работе над проектом большой программной системы быть компетентным в области вычислительной техники и программировании. Почему?
5. Может ли большая программная система быть отлажена до конца и почему? 6. При каких условиях созданный программный комплекс может быть назван программным продуктом?
7. Что такое системное программное обеспечение?
8. Что такое инструментарий технологии программирования?

ЖИЗНЕННЫЙ ЦИКЛ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Жизненный цикл ПО определяется как период времени, который начинается с момента принятия решения о необходимости ПО и заканчивается в момент его полного изъятия из эксплуатации. Основным нормативным документом, регламентирующим состав ЖЦ ПО, является международный стандарт ISO/IEC 12207: 2008 «System and software engineering – Software life cycle processes». Данный стандарт, используя устоявшуюся терминологию, устанавливает общую структуру процессов жизненного цикла программных средств, на которую можно ориентироваться в программной индустрии. Стандарт определяет процессы, виды деятельности и задачи, которые используются при приобретении программного продукта или услуги, а также при поставке, разработке, применении по назначению, сопровождении и прекращении применения программных продуктов. (его российский аналог ГОСТ Р ИСО/МЭК 12207-2010 «Информационная технология. Системная и программная инженерия. Процессы жизненного цикла программных средств»). Каждый процесс (процесс – совокупность взаимосвязанных действий, преобразующий некоторые входные данные в выходные) разделен на набор действий, каждое действие – на набор задач. В соответствии с ГОСТ Р ИСО/МЭК 12207-2010 различные виды деятельности, которые могут выполняться в течение жизненного цикла программных систем в семь групп процессов: 1) процессы соглашения – 2; 2) процессы организационного обеспечения проекта – 5; 3) процессы проекта – 7; 4) технические процессы – 11; 5) процессы реализации программных средств – 7; 6) процессы поддержки программных средств – 8; 7) процессы повторного применения программных средств – 3. Каждый из процессов жизненного цикла в пределах этих групп описывается в терминах цели и желаемых выходов, список действий и задач, которые необходимо выполнять для достижения этих результатов. В рамках курса «Технология разработки программного обеспечения» детально будет рассмотрена группа процессов, описывающих реализацию программных средств [5]. 2.1. Процессы реализации программных средств 16 Процессы реализации программных средств используются для создания конкретного элемента системы (составной части), выполненного в виде программного средства. Эти процессы преобразуют заданные характеристики поведения, интерфейсы и ограничения на реализацию в действия, результатом которых становится системный элемент, удовлетворяющий требованиям, вытекающим из системных требований. [ГОСТ Р ИСО/МЭК 12207-2010]. Примечание. В курсе «Технология разработки программного обеспечения» программное средство рассматривается ни как составная часть системы, а как независимое автономное программное обеспечение, состоящее из программных модулей [17]. 2.1.1. Процесс реализации При реализации проекта необходимо осуществлять следующие виды деятельности в соответствии с принятыми в 2 организации

политиками и процедурами в отношении процесса реализации программных средств: 1) Если не оговорено в контракте, разработчик должен определить или выбрать модель жизненного цикла, соответствующую области применения, размерам и сложности проекта. Модель жизненного цикла должна содержать стадии, цели и выходы каждой стадии. Виды деятельности и задачи процесса реализации программных средств должны быть выбраны и отражены в модели жизненного цикла. Подробно существующие модели и методологии будут рассмотрены во второй теме текущего документа. Эти виды деятельности и задачи могут пересекаться или взаимодействовать друг с другом, могут выполняться итеративно или рекурсивно. В идеальном случае рассматриваемые виды деятельности и задачи выполняются и решаются с использованием определенной [illegible] передавать результаты в процесс менеджмента конфигурации программных средств и [illegible] проблемы и снимать несоответствия, найденные в программных продуктах и задачах в соответствии с процессом решения проблем в программных [illegible] соединять элементы конфигурации в сроки, определенные приобретающей стороной и поставщиком. 3) Исполнитель должен выбирать, адаптировать и применять те стандарты, методы, инструментарий и языки программирования (если не оговорено в контракте), которые документально оформлены, являются подходящими и установлены организацией для выполнения деятельности в рамках процесса реализации программных средств и поддерживающих процессов. 4) Исполнитель должен разрабатывать планы проведения действий процесса реализации программных средств. Планы должны включать в себя конкретные стандарты, методы, инструментарий, действия и обязанности, связанные с разработкой и квалификацией всех требований, включая безопасность и защиту. При необходимости могут разрабатываться отдельные планы. Эти планы должны документироваться и выполняться. 5) При разработке или сопровождении программных продуктов могут применяться не поставляемые элементы. Однако должно гарантироваться, что функционирование и сопровождение поставляемых программных продуктов после поставки приобретающей стороне не зависит от таких элементов; другими словами, эти элементы следует также рассматривать как поставляемые. Результатом процесса является создание программной составной части, удовлетворяющей как требованиям к архитектурным решениям, что подтверждается посредством верификации, так и требованиям правообладателей, что подтверждается посредством валидации. В результате 3 успешного осуществления процесса реализации программных средств: 1) определяется стратегия реализации; 2) определяются ограничения по технологии реализации проекта; 3) изготавливается программная составная часть; 4) программная составная часть упаковывается и хранится в соответствии с соглашением о ее поставке. Процесс реализации программных средств включает в себя несколько специальных процессов более низкого уровня: 1) процесс анализа требований к программным средствам; 2) процесс проектирования архитектуры программных средств; 3) процесс детального проектирования программных средств; 4) процесс конструирования программных средств; 5) процесс комплексирования программных средств; 6) процесс квалификационного тестирования программных средств. 2.1.2. Процесс анализа требований к программным средствам Цель процесса анализа требований к программным средствам заключается в установлении и документировании требований к программному обеспечению. В результате успешного выполнения процесса определяется перечень требований к функциональным модулям программного обеспечения и их интерфейсам, определяются приоритеты реализации требований, требования к ПО оцениваются по стоимости, графикам работ и техническим воздействиям. Подробно о способах выявления

и видах требований будет описано в третьей теме текущего документа [6, 17]. 2.1.3. Процессы проектирования (детального проектирования) архитектуры программных средств Цель процесса заключается в обеспечении проекта для программных средств, которые реализуются и могут быть проверены относительно требований сформулированных в ходе процесса анализа требований. В рамках процесса исполнитель осуществляет преобразование выявленных требований в архитектуру, которая описывает верхний уровень структуры программного средства и идентифицирует программные компоненты. Исполнитель должен разработать проект, описывающий внешние и внутренние интерфейсы, структуру и метод доступа к базе данных (БД), так же исполнитель оформляет предварительные версии пользовательской документации и требования к предварительному тестированию. В результате успешной реализации процесса разрабатывается проект архитектуры программных средств, определяются внутренние и внешние интерфейсы, устанавливается соответствие между требованиями и программным проектом. Подробно о методах проектирования программных средств рассказывается в четвертой теме. 2.1.4. Процесс конструирования программных средств Целью процесса является создание исполняемых программных блоков (модулей), которые созданы на основе архитектурного проекта. При реализации процесса исполнитель разрабатывает документацию на каждый программный модуль и базу данных, процедуры и данные для 19 тестирования модулей и базы данных. В данном процессе также происходит тестирование модулей исполнителем, гарантируя, что они удовлетворяют требованиям. В ходе тестирования ведется журнал тестирования, фиксирующий информацию о соответствующих работах (когда проводится, какой тест, кем проводится и т.п.). Неожиданные или некорректные результаты тестов могут записываться в специальной подсистеме ведения отчетности по сбоям. Исполнитель должен оценивать программный код и результаты испытаний, учитывая следующие критерии: 1) прослеживаемость к требованиям и проекту программных элементов; 2) внешнюю согласованность с требованиями и архитектурным проектом для программных модулей; 3) тестовое покрытие модулей; 4) соответствие методов кодирования и используемых стандартов; 5) осуществимость функционирования и сопровождения. В результате успешного осуществления процесса определяется критерий верификации для всех модулей относительно требований, разработка программных модулей, тестирование [17]. 2.1.5. Процесс комплексирования программных средств В ходе процесса комплексирования программных средств осуществляется объединение функциональных программных модулей, создание интегрированных программных элементов, согласованных с проектом программного средства, которые демонстрируют, что функциональные и нефункциональные требования к программному средству удовлетворяются. Для каждого модуля программного средства исполнитель должен разработать план комплексирования для объединения программных модулей. План должен включать в себя требования к тестированию, данные для тестирования, обязанности и графики работ. Так же исполнителю необходимо объединить программные модули в соответствии с планом комплексирования и разработать комплекс тестов. Результаты комплексирования и тестирования должны быть оформлены документально. Любое изменение в пользовательском интерфейсе и функциональности сопровождается обновлением пользовательской документации по мере необходимости [17]. 2.1.6. Процесс квалификационного тестирования программного средства. Цель процесса квалификационного тестирования программного средства заключается в подтверждении того, что комплектованный программный продукт удовлетворяет установленным требованиям. В рамках процесса исполнитель должен провести квалификационное тестирование (согласно требованиям). Исполнителю необходимо провести оценку проекта, кода, тестов и их результаты, а также пользовательской документации, учитывая следующие критерии: 1) тестовое покрытие требования к программному средству; 2)

соответствие с ожидаемыми результатами; 3) осуществимость функционирования и сопровождения. После успешного тестирования программный продукт готов к передаче заказчику. После чего в действие вступают процессы поддержки программного средства [17]. Заключение Эталонная модель, описанная в стандарте, не представляет конкретного подхода к осуществлению процесса, как и не определяет модель жизненного цикла системы (программного средства), методологию или технологию. В зависимости от принятой методологии разработки ПО или модели ЖЦ в конкретной компании используется различный набор процессов. В следующей теме рассмотрены основные модели и методологии разработки программного обеспечения.

Вопросы и задания для самоконтроля

1. Понятие жизненного цикла ПО. Что понимается под процессом жизненного цикла? Назовите основные группы процессов согласно ГОСТ Р ИСО
2. Основная цель процесса анализа требований к программным средствам. Что является результатом успешного осуществления процесса?
3. Процесс реализации. Какие виды деятельности и задачи входят в состав процесса реализации?
4. Процесс проектирования архитектуры программных средств. Что является результатом успешной реализации процесса. Что понимается под базовой линией?

Процессы реализации программных средств

Процессы реализации программных средств используются для создания конкретного элемента системы (составной части), выполненного в виде программного средства. Эти процессы преобразуют заданные характеристики поведения, интерфейсы и ограничения на реализацию в действия, результатом которых становится системный элемент, удовлетворяющий требованиям, вытекающим из системных требований. [ГОСТ Р ИСО]. Примечание. В курсе «Технология разработки программного обеспечения» программное средство рассматривается ни как составная часть системы, а как независимое автономное программное обеспечение, состоящее из программных модулей. 2.1.1. Процесс реализации При реализации проекта необходимо осуществлять следующие виды деятельности в соответствии с принятыми в организации политиками и процедурами в отношении процесса реализации программных средств: 1) Если не оговорено в контракте, разработчик должен определить или выбрать модель жизненного цикла, соответствующую области применения, размерам и сложности проекта. Модель жизненного цикла должна содержать стадии, цели и выходы каждой стадии. Виды деятельности и задачи процесса реализации программных средств должны быть выбраны и отражены в модели жизненного цикла. Подробно существующие модели и методологии будут рассмотрены во второй теме текущего документа. Эти виды деятельности и задачи могут пересекаться или взаимодействовать друг с другом, могут выполняться итеративно или рекурсивно. В идеальном случае рассматриваемые виды деятельности и задачи выполняются и решаются с использованием определенной организационной модели

соответствии с процессом мен

результаты в процесс менеджмента конфигурации программных средств и выполнять

снимать несоответствия, найденные в программных продуктах и задачах в соответствии с процессом решения проблем в программных средствах; выполнять поддержку процессов

конфигурации в сроки, определенные приобретающей стороной и поставщиком. 3) Исполнитель должен выбирать, адаптировать и применять те стандарты, методы, инструментарий и языки программирования (если не оговорено в контракте), которые

документально оформлены, являются подходящими и установлены организацией для выполнения деятельности в рамках процесса реализации программных средств и поддерживающих процессов. 4) Исполнитель должен разрабатывать планы проведения действий процесса реализации программных средств. Планы должны включать в себя конкретные стандарты, методы, инструментарий, действия и обязанности, связанные с разработкой и квалификацией всех требований, включая безопасность и защиту. При необходимости могут разрабатываться отдельные планы. Эти планы должны документироваться и выполняться. 5) При разработке или сопровождении программных продуктов могут применяться не поставляемые элементы. Однако должно гарантироваться, что функционирование и сопровождение поставляемых программных продуктов после поставки приобретающей стороне не зависит от таких элементов; другими словами, эти элементы следует также рассматривать как поставляемые. Результатом процесса является создание программной составной части, удовлетворяющей как требованиям к архитектурным решениям, что подтверждается посредством верификации, так и требованиям правообладателей, что подтверждается посредством валидации. В результате успешного осуществления процесса реализации программных средств: 1) определяется стратегия реализации; 2) определяются ограничения по технологии реализации проекта; 3) изготавливается программная составная часть; 4) программная составная часть упаковывается и хранится в соответствии с соглашением о ее поставке. Процесс реализации программных средств включает в себя несколько специальных процессов более низкого уровня: 1) процесс анализа требований к программным средствам; 2) процесс проектирования архитектуры программных средств; 3) процесс детального проектирования программных средств; 4) процесс конструирования программных средств; 18 5) процесс комплексирования программных средств; 6) процесс квалификационного тестирования программных средств. 2.1.2. Процесс анализа требований к программным средствам Цель процесса анализа требований к программным средствам заключается в установлении и документировании требований к программному обеспечению. В результате успешного выполнения процесса определяется перечень требований к функциональным модулям программного обеспечения и их интерфейсам, определяются приоритеты реализации требований, требования к ПО оцениваются по стоимости, графикам работ и техническим воздействиям. Подробно о способах выявления и видах требований будет описано в третьей теме текущего документа [6, 17]. 2.1.3. Процессы проектирования (детального проектирования) архитектуры программных средств Цель процесса заключается в обеспечении проекта для программных средств, которые реализуются и могут быть проверены относительно требований сформулированных в ходе процесса анализа требований. В рамках процесса исполнитель осуществляет преобразование выявленных требований в архитектуру, которая описывает верхний уровень структуры программного средства и идентифицирует программные компоненты. Исполнитель должен разработать проект, описывающий внешние и внутренние интерфейсы, структуру и метод доступа к базе данных (БД), так же исполнитель оформляет предварительные версии пользовательской документации и требования к предварительному тестированию. В результате успешной реализации процесса разрабатывается проект архитектуры программных средств, определяются внутренние и внешние интерфейсы, устанавливается соответствие между требованиями и программным проектом. Подробно о методах проектирования программных средств рассказывается в четвертой теме. 2.1.4. Процесс конструирования программных средств Целью процесса является создание исполняемых программных блоков (модулей), которые созданы на основе архитектурного проекта. При реализации процесса исполнитель разрабатывает документацию на каждый программный модуль и базу данных, процедуры и данные для 19 тестирования модулей и базы данных. В данном процессе также происходит тестирование модулей исполнителем, гарантируя, что они удовлетворяют

требованиям. В ходе тестирования ведется журнал тестирования, фиксирующий информацию о соответствующих работах (когда проводится, какой тест, кем проводится и т.п.). Неожиданные или некорректные результаты тестов могут записываться в специальной подсистеме ведения отчетности по сбоям. Исполнитель должен оценивать программный код и результаты испытаний, учитывая следующие критерии: 1) прослеживаемость к требованиям и проекту программных элементов; 2) внешнюю согласованность с требованиями и архитектурным проектом для программных модулей; 3) тестовое покрытие модулей; 4) соответствие методов кодирования и используемых стандартов; 5) осуществимость функционирования и сопровождения. В результате успешного осуществления процесса определяется критерий верификации для всех модулей относительно требований, разработка программных модулей, тестирование [17]. 2.1.5. Процесс комплексирования программных средств В ходе процесса комплексирования программных средств осуществляется объединение функциональных программных модулей, создание интегрированных программных элементов, согласованных с проектом программного средства, которые демонстрируют, что функциональные и нефункциональные требования к программному средству удовлетворяются. Для каждого модуля программного средства исполнитель должен разработать план комплексирования для объединения программных модулей. План должен включать в себя требования к тестированию, данные для тестирования, обязанности и графики работ. Так же исполнителю необходимо объединить программные модули в соответствии с планом комплексирования и разработать комплекс тестов. Результаты комплексирования и тестирования должны быть оформлены документально. Любое изменение в пользовательском интерфейсе и функциональности сопровождается обновлением пользовательской документации по мере необходимости [17]. 2.1.6. Процесс квалификационного тестирования программного средства 20 Цель процесса квалификационного тестирования программного средства заключается в подтверждении того, что комплектованный программный продукт удовлетворяет установленным требованиям. В рамках процесса исполнитель должен провести квалификационное тестирование (согласно требованиям). Исполнителю необходимо провести оценку проекта, кода, тестов и их результаты, а также пользовательской документации, учитывая следующие критерии: 1) тестовое покрытие требования к программному средству; 2) соответствие с ожидаемыми результатами; 3) осуществимость функционирования и сопровождения. После успешного тестирования программный продукт готов к передаче заказчику. После чего в действие вступают процессы поддержки программного средства [17]. Заключение Эталонная модель, описанная в стандарте, не представляет конкретного подхода к осуществлению процесса, как и не определяет модель жизненного цикла системы (программного средства), методологию или технологию. В зависимости от принятой методологии разработки ПО или модели ЖЦ в конкретной компании используется различный набор процессов. В следующей теме рассмотрены основные модели и методологии разработки программного обеспечения.

Вопросы и задания для самоконтроля

1. Понятие жизненного цикла ПО. Что понимается под процессом жизненного цикла? Назовите основные группы процессов согласно ГОСТ Р ИСО.
2. Основная цель процесса анализа требований к программным средствам. Что является результатом успешного осуществления процесса?
3. Процесс реализации. Какие виды деятельности и задачи входят в состав процесса реализации?
4. Процесс проектирования архитектуры программных средств. Что является результатом успешной реализации процесса. Что понимается под базовой линией?

СИСТЕМНОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

Основные понятия, факты

Назначение и классификация СПО. Требования к СПО. Базовое и сервисное СПО. Операционные системы. Системы управления файлами. Системные утилиты. Системы программирования.

Навыки и умения

Разработка системного программного обеспечения в соответствии с требованиями к СПО на языках Assembler, C++.

Установка и использование системных утилит.

Классификация программного обеспечения

Традиционно все программное обеспечение подразделяют на два класса:

1) системное программное обеспечение (СПО) и

2) прикладное (пользовательское)программное обеспечение (ППО)

Выделим еще один класс (скорее группу) программ - **специальное программное обеспечение информационных и управляющих систем.**

Прикладные программы предназначены для решения функциональных задач, они выполняют обработку информации различных предметных областей.

Это самый многочисленный класс программных продуктов.

К **специальному программному обеспечению информационных и управляющих систем** относятся

· программы (системы) управления базами данных;

· программы управления языком интерфейса информационных систем;

· программы сбора и предварительной обработки информации (в информационно-измерительных системах, например, бортовые системы).

ПО этого класса часто оказывается скрытым в составе драйверов оборудования или поставляется в виде библиотек функционального расширения языков программирования.

Поэтому часто такие ПО относят к системному программному обеспечению.

Мы будем считать ПО такого типа отдельным классом и рассматривать не будем.

Системное программное обеспечение (System Software) - совокупность программ и программных комплексов для обеспечения работы компьютера и сетей ЭВМ.

СПО управляет ресурсами компьютерной системы и позволяет пользователям программировать в более выразительных языках, чем машинных язык компьютера. Состав СПО мало зависит от характера решаемых задач пользователя.

Назначение системного программного обеспечения

Системное программное обеспечение предназначено для:

· создания операционной среды функционирования других программ (другими словами, для организации выполнения программ);

· автоматизации разработки (создания) новых программ;

· обеспечения надежной и эффективной работы самого компьютера и вычислительной сети;

· проведения диагностики и профилактики аппаратуры компьютера и вычислительных сетей;

· выполнения вспомогательных технологических процессов (копирование, архивирование, восстановление файлов программ и баз данных и т.д.).

Данный класс программных продуктов тесно связан с типом компьютера и является его неотъемлемой частью.

Программные продукты данного класса в основном ориентированы на квалифицированных пользователей - профессионалов в компьютерной области: системного программиста, администратора сети, прикладного программиста, оператора.

Однако знание базовой технологии работы с этим классом программных продуктов требуется и конечным пользователям персонального компьютера, которые

самостоятельно не только работают со своими программами, но и выполняют обслуживание компьютера, программ и данных.
Программные продукты данного класса носят общий характер применения, независимо от специфики предметной области.
К системным программным продуктам предъявляются высокие требования по надежности и технологичности работы, удобству и эффективности использования.

Классификация системного программного обеспечения

В СПО традиционно включают
· системные управляющие и
· системные обрабатывающие программы.

Управляющие системные программы организуют корректное функционирование всех устройств системы.
Основные системные функции управляющих программ - 3
Интерпретатор последовательно переводит на машинный язык и выполнят операторы исходного модуля
(У интерпретаторов два основных недостатка. Первый - низкая скорость работы интерпретируемых программ.)
Преимущество интерпретатора перед компилятором состоит в том, что программа пользователя имеет одно представление - в виде текста. При компиляции одна и та же программа имеет несколько представлений - в виде текста и в виде выполняемого файла.
Компоновщик, или редактор связей - системная обрабатывающая программа, редактирующая и объединяющая объектные (ранее оттраслированные) модули в единые загрузочные, готовые к выполнению программные модули. Загрузочный модуль может быть помещен ОС в основную память и выполнен.
Отладчик позволяет управлять процессом исполнения программы, является инструментом для поиска и исправления ошибок в программе. Базовый набор функций отладчика включает:
· пошаговое выполнение программы (режим трассировки) с отображением результатов,
· остановка в заранее определенных точках,
· возможность остановки в некотором месте программы при выполнении некоторого условия;
· изображение и изменение значений переменных.
Загрузчик - системная обрабатывающая программа, объединяющая основные функции редактора связей и программы выборки в одном пункте задания. Загрузчик помещает находящиеся в его входном наборе данных объектные и загрузочные модули в оперативную память, объединяет их в единую программу, корректирует перемещаемые адресные константы с учетом фактического адреса загрузки и передает управление в точку входа созданной программы.
Средства сетевого доступа обеспечивают обработку, передачу и хранение данных в сети.
Заметим, что чаще говорят о сетевых операционных системах, которые предоставляют пользователям различные виды сетевых служб (управление файлами, электронная почта, процессы управления сетью и др.)
Ключом к использованию этих ресурсов является сервер, специальная программа на компьютере, подключенному к сети, которая принимает запросы (или команды) и посылает ответы автоматически.
Программы, предназначенные для подачи запросов серверу, называются программами-клиентами. Сервер предназначен для их обслуживания. Клиент посылает запросы пользователя на сервер, используя стандартизированный формат, называемый протоколом. Ответ сервера содержит информацию, представленную в виде файла, содержащего данные того или иного формата.

Постоянно ведется разработка все новых программ-клиентов, предлагающих более удобные способы взаимодействия с сервером.
Пример. Приложения Netscape Navigator , Internet Explorer - программы- клиенты.
Таким образом, в системном ПО мы выделили **пять групп системных программ**:
· операционные системы;
· интерфейсные оболочки для взаимодействия пользователя с ОС (операционная оболочка) и программные среды;
· системы управления файлами;
· системы программирования;
· утилиты;
· средства сетевого доступа.
Обратим внимание на то, что в ходе развития компьютерных систем наиболее используемые прикладные программы могут быть перенесены на уровень системных, что позволяет использовать их в различных приложениях. Например, средства управления диалоговым взаимодействием с пользователем в системных оболочках (типа Windows).
С другой стороны, наиболее распространенные и критические по времени системные функции были частично или полностью реализованы аппаратно. Например, средства управления многопрограммным защищенным режимом и средства управления мультимедиа-устройствами в процессорах фирмы Intel.
Требования к системному программному обеспечению
Системные программы должны удовлетворять следующим требованиям:
· прозрачность работы;
· гарантированная надежность выполнения в соответствии со спецификациями (спецификациями называются функциональные требования);
· максимальная скорость выполнения;
· минимальные затраты на хранение машинных кодов;
· поддержка стандартных средств связи с прикладными программами.
Эффективность системных программ зависит от времени их создания и надежности исполняемого кода.
Требование эффективности системных программ вызывает необходимость использования специальных языков
· машинно-ориентированных типа языка Assembler и
· высокого уровня типа С или С++.

Процессы проектирования (детального проектирования) архитектуры программных средств

Цель процесса заключается в обеспечении проекта для программных средств, которые реализуются и могут быть проверены относительно требований сформулированных в ходе процесса анализа требований. В рамках процесса исполнитель осуществляет преобразование выявленных требований в архитектуру, которая описывает верхний уровень структуры программного средства и идентифицирует программные компоненты. Исполнитель должен разработать проект, описывающий внешние и внутренние интерфейсы, структуру и метод доступа к базе данных (БД), так же исполнитель оформляет предварительные версии пользовательской документации и требования к предварительному тестированию. В результате успешной реализации процесса разрабатывается проект архитектуры программных средств, определяются внутренние и внешние интерфейсы, устанавливается соответствие между требованиями и программным проектом. Подробно о методах проектирования программных средств рассказывается в четвертой теме. 2.1.4. Процесс конструирования программных средств Целью процесса является создание исполняемых программных блоков (модулей), которые созданы на

основе архитектурного проекта. При реализации процесса исполнитель разрабатывает документацию на каждый программный модуль и базу данных, процедуры и данные для 19 тестирования модулей и базы данных. В данном процессе также происходит тестирование модулей исполнителем, гарантируя, что они удовлетворяют требованиям. В ходе тестирования ведется журнал тестирования, фиксирующий информацию о соответствующих работах (когда проводится, какой тест, кем проводится и т.п.). Неожиданные или некорректные результаты тестов могут записываться в специальной подсистеме ведения отчетности по сбоям. Исполнитель должен оценивать программный код и результаты испытаний, учитывая следующие критерии: 1) прослеживаемость к требованиям и проекту программных элементов; 2) внешнюю согласованность с требованиями и архитектурным проектом для программных модулей; 3) тестовое покрытие модулей; 4) соответствие методов кодирования и используемых стандартов; 5) осуществимость функционирования и сопровождения. В результате успешного осуществления процесса определяется критерий верификации для всех модулей относительно требований, разработка программных модулей, тестирование [17]. 2.1.5. Процесс комплексирования программных средств В ходе процесса комплексирования программных средств осуществляется объединение функциональных программных модулей, создание интегрированных программных элементов, согласованных с проектом программного средства, которые демонстрируют, что функциональные и нефункциональные требования к программному средству удовлетворяются. Для каждого модуля программного средства исполнитель должен разработать план комплексирования для объединения программных модулей. План должен включать в себя требования к тестированию, данные для тестирования, обязанности и графики работ. Так же исполнителю необходимо объединить программные модули в соответствии с планом комплексирования и разработать комплекс тестов. Результаты комплексирования и тестирования должны быть оформлены документально. Любое изменение в пользовательском интерфейсе и функциональности сопровождается обновлением пользовательской документации по мере необходимости [17]. 2.1.6. Процесс квалификационного тестирования программного средства 20 Цель процесса квалификационного тестирования программного средства заключается в подтверждении того, что комплектованный программный продукт удовлетворяет установленным требованиям. В рамках процесса исполнитель должен провести квалификационное тестирование (согласно требованиям). Исполнителю необходимо провести оценку проекта, кода, тестов и их результаты, а также пользовательской документации, учитывая следующие критерии: 1) тестовое покрытие требования к программному средству; 2) соответствие с ожидаемыми результатами; 3) осуществимость функционирования и сопровождения. После успешного тестирования программный продукт готов к передаче заказчику. После чего в действие вступают процессы поддержки программного средства [17]. Заключение Эталонная модель, описанная в стандарте, не представляет конкретного подхода к осуществлению процесса, как и не определяет модель жизненного цикла системы (программного средства), методологию или технологию. В зависимости от принятой методологии разработки ПО или модели ЖЦ в конкретной компании используется различный набор процессов. В следующей теме рассмотрены основные модели и методологии разработки программного обеспечения.

Вопросы и задания для самоконтроля

1. Понятие жизненного цикла ПО. Что понимается под процессом жизненного цикла? Назовите основные группы процессов согласно ГОСТ Р ИСО.
2. Основная цель процесса анализа требований к программным средствам. Что является результатом успешного осуществления процесса?
3. Процесс реализации. Какие виды деятельности и задачи входят в состав процесса реализации?

4. Процесс проектирования архитектуры программных средств. Что является результатом успешной реализации процесса. Что понимается под базовой линией?

МОДЕЛИ И МЕТОДОЛОГИИ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Процесс жизни любой системы или программного продукта может быть описан посредством модели жизненного цикла, состоящей из стадий. Модели могут использоваться для представления всего жизненного цикла от замысла до прекращения применения или для представления части жизненного цикла, соответствующей текущему проекту. Модель жизненного цикла представляется в виде последовательности стадий, которые могут перекрываться и (или) повторяться циклически в соответствии с областью применения, размером, сложностью, потребностью в изменениях и возможностях. Каждая стадия описывается формулировкой цели и выходов. Процессы и действия жизненного цикла отбираются и исполняются на этих стадиях для полного удовлетворения цели и результатам каждой стадии. Различные организации могут использовать различные стадии в пределах жизненного цикла. Однако каждая стадия реализуется организацией, ответственной за эту стадию, с надлежащим рассмотрением информации, имеющейся в планах жизненного цикла и решениях, принятых на предшествующих стадиях. Аналогичным образом организация, ответственная за текущую стадию, ведет записи принятых решений и записи допущений, относящихся к последующим стадиям данного жизненного цикла [2, 25]. 3.1. Модели жизненного цикла программного обеспечения Под моделью жизненного цикла ПО понимается структура, определяющая последовательность выполнения и взаимосвязи процессов, действий и задач на протяжении ЖЦ. Модель ЖЦ зависит от спецификации, масштаба и сложности проекта и спецификации условий, в которых система создается и функционирует. Модель ЖЦ ПО включает в себя: стадии, результаты выполнения работ на каждой стадии, ключевые события – точки завершения работ и принятия решений. Модель ЖЦ любого конкретного ПО определяет характер процесса его создания, который представляет собой совокупность упорядоченных во времени, взаимосвязанных и объединенных в стадии работ, выполнение которых необходимо и достаточно для создания ПО, соответствующего заданным требованиям. Под стадией понимается часть процесса создания ПО, ограниченная определенными временными рамками и 22 заканчивающаяся выпуском конкретного продукта (моделей, программных компонентов, документации), определяемого заданными для данной стадии требованиями. На каждой стадии могут выполнятся несколько процессов, определённых в стандарте ГОСТ Р ИСО/МЭК 12207-2010, и наоборот один и тот процесс может выполняться на различных стадиях. Соотношение между стадиями и процессами также определяется используемой моделью ЖЦ ПО. Далее рассмотрим модели и их классификации [2, 25]. 3.1.1. Каскадная модель Первой моделью, получившей широкую известность и действительно структурирующей процесс разработки, является каскадная (водопадная) модель. Каждая стадия каскадной модели заканчивается получением некоторых результатов, которые служат в качестве исходных данных для следующей стадии. Требования к разрабатываемому ПО, определенные на стадии формирования требований, строго документируются в виде технического задания и фиксируются на все время разработки проекта. Рис. 3.1 Стандартная водопадная модель Преимущества [redacted] стадии формируется законченный набор проектной документации, отвечающий критериям [redacted] позволяют планировать сроки завершения всех работ и соответствующие затраты. Каскадная модель может использоваться при создании ПО, для которого в самом начал разработки можно достаточно точно и полно сформулировать все требования. В то же время этот подход обладает рядом недостатков, вызванных прежде всего тем, чьл реальный процесс создания ПО никогда полностью не укладывался в такую жесткую

схему. Спустя непродолжительное время после появления на свет каскадная модель была доработана Уинстом Ройсом с учетом взаимозависимости этапов и необходимости возврата на предыдущие ступени, что может быть вызвано, например, неполнотой требований или ошибками в формировании задания. Процесс создания ПО носит как правило, итерационный характер: результаты очередной стадии часто вызывают изменения в проектных решениях, выработанных на более ранних стадиях. Таким образом, постоянно возникает потребность в возврате к предыдущим стадиям и уточнении или пересмотре ранее принятых решений. В результате реальный процесс создания ПО принимает вид, изображенный на рисунке .1. Рис. 3.1. Модифицированная водопадная модель Наиболее распространенным результатом каскадного похода к разработке ПО является поздняя неудача. Кажется, что проекты выполняются нормально, но только до тех пор, пока работы не вступят в 24 завершающий этап, и тогда выясняется, что потребители недовольны созданным продуктом [25]. 3.1.2. V-образная модель, как разновидность каскадной модели Основной принцип V-образной модели заключается в том, что детализация проекта возрастает при движении слева направо, одновременно с течением времени, и ни то, ни другое не может повернуть вспять. Итерации в проекте производятся по горизонтали, между левой и правой сторонами буквы. V-модель – вариация каскадной модели, в которой задачи разработки идут сверху вниз по левой стороне буквы V, а задачи тестирования – вверх по правой стороне буквы V. Внутри V проводятся горизонтальные линии, показывающие, как результаты каждой из стадий разработки влияют на развитие системы тестирования на каждой из стадий тестирования. Модель базируется на том, что приемо-сдаточные испытания основываются, прежде всего, на требованиях, системное тестирование – на требованиях и архитектуре, комплексное тестирование – на требованиях, архитектуре и интерфейсах, а компонентное тестирование – на требованиях, архитектуре, интерфейсах и алгоритмах. Подготовка процесса разработки Анализ требований к системе Проектирование системы Проектирование ПС Программирование и тестирование ПС Эксплуатация и сопровождение Ввод в действие и обеспечение приемки Сборка и квалификационные испытания системы Сборка и квалификационные испытания ПС Рис. 3.2 – V-образная модель Особенностью данной модели является разбиение стадий на три логических этапа: проектирование (детализация требований), реализация, тестирование. 25 V-модель дает организациям и проектным группам руководство по выполнению и завершению проектов последовательным и воспроизводимым образом. Применение принципов V-модели гарантирует выявление и фиксацию требований пользователей. Утвержденные требования могут быть переведены в функции готового приложения, (и) приложение отражает требования пользователей [2, 25]. 3.1.3. Итеративный инкрементный подход к разработке (эволюционная модель) 3.1.3.1 Итеративная модель Итеративная модель предполагает разбиение жизненного цикла проекта на последовательность итераций, каждая из которых напоминает «мини-проект», включая все фазы жизненного цикла в применении к созданию меньших фрагментов функциональности, по сравнению с проектом, в целом. Цель каждой итерации – получение работающей версии программной системы, включающей функциональность, определенную интегрированным содержанием всех предыдущих и текущей итерации. Результата финальной итерации содержит всю требуемую функциональность продукта. Таким образом, с завершением каждой итерации, продукт развивается инкрементально. Шансы успешного создания сложной системы будут максимальными, если она реализуется в серии небольших шагов и если каждый шаг заключает в себе четко определенный результат, а также возможность возврата к результатам предыдущей успешной итерации, в случае неудачи. Перед тем, как пустить в дело все ресурсы, предназначенные для создания ПО, разработчик имеет возможность получать обратную связь из реального мира (заказчиков, пользователей) и исправлять возможные ошибки в проекте. Итеративная модель подразумевает возможность не только

сборки работающей (с точки зрения результатов тестирования) версии системы - прототипа, но и её развертывания в реальных операционных условиях с анализом откликов пользователей для определения содержания и планирования следующей итераци[illegible] соответствии с бюджетом, полностью защищающую от перерасхода времени или средств (в частности, за счет сокращения второстепенной функциональности). 26 3.1.3.2 Инкрементная модель Идея, лежащая в основе инкрементной модели, состоит в том, что программную систему следует разрабатывать по принципу приращений, так, чтобы разработчик мог использовать данные, полученные при разработке более ранних версий ПО. Новые данные получаются как в ходе разработки ПО, так и в ходе его использования, где это возможно. Ключевые этапы этого процесса – простая реализация подмножества требований к программе и совершенствование модели в серии последовательных релизов до тех пор, пока не будет реализовано ПО во всей полноте. В ходе каждой итерации организация модели изменяется, и к ней добавляются новые функциональные возможности. Для организации инкрементной разработки обычно выбирается характерный временной интервал, например, неделя. Затем в течение этого интервала происходит обновление проекта: добавляется новая документация как текстовая, так и графическая, расширяется набор тестов, добавляются новые программные коды и т. д. Теоретически шаги разработки могут выполняться и параллельно, но такой процесс очень сложно скоординировать. Инкрементная разработка проходит лучше всего, если следующая итерация начинается после того, как обновление всех артефактов в предыдущей итерации закончено, и существенно хуже, если время, требуемое на обновление артефактов, значительно превышает выбранный интервал [27]. В результате каждой итерации получается работающее, но не полнофункциональное ПО, которое еще не является программным продуктом и не подлежит распространению. В результате каждой итерации создается версия некоторой части ПО. Необходимо заметить, но как правило на каждой итерации определяются и реализуются новые требования, некоторые итерации могут быть целиком посвящены усовершенствованию существующей программы, например, с целью повышения ее производительности. Вывод С точки зрения структуры жизненного цикла эволюционную модель называют итеративной. С точки зрения развития продукта – инкрементальной. Опыт показывает, что невозможно рассматривать каждый из этих взглядов изолировано. Чаще всего такую смешанную эволюционную модель называют просто итеративной (говоря о процессе) и/или инкрементальной (говоря о наращивании функциональности продукта). Значимость эволюционной модели на основе организации итераций особо проявляется в снижении 27 неопределенности с завершением каждой итерации. В свою очередь, снижение неопределенности позволяет уменьшить риски. Рисунок 3.3 иллюстрирует идею эволюционной модели, предполагая, что итеративному разбиению может быть подвержен не только жизненный цикл в целом, включающий перекрывающиеся стадии – формирование требований, проектирование, конструирование и т.п., но и каждая стадия может, в свою очередь, разбиваться на уточняющие итерации, связанные, например, с детализацией структуры декомпозиции проекта – например, архитектуры модулей системы [25]. Рисунок 3.3 – Снижение неопределенности и инкрементальное расширение функциональности при итеративной организация жизненного цикла.

Каскадная модель

Первой моделью, получившей широкую известность и действительно структурирующей процесс разработки, является каскадная (водопадная) модель. Каждая стадия каскадной модели заканчивается получением некоторых результатов, которые служат в качестве исходных данных для следующей стадии. Требования к разрабатываемому ПО,

определенные на стадии формирования требований, строго документируются в виде технического задания и фиксируются на все время разработки проекта.

Стандартная водопадная модель. Преимущества применения каскадной модели [illegible] выполняемые в логичной последователь стадии работ позволяют планировать сроки завершения всех работ и соответствующие затраты. Каскадная модель может использоваться при создании ПО, для которого в самом начал разработки можно достаточно точно и полно сформулировать все требования. В то же время этот подход обладает рядом недостатков, вызванных прежде всего тем, чьл реальный процесс создания ПО никогда полностью не укладывался в такую жесткую схему. Спустя непродолжительное время после появления на свет каскадная модель была доработана Уинстом Ройсом с учетом взаимозависимости этапов и необходимости возврата на предыдущие ступени, что может быть вызвано, например, неполнотой требований или ошибками в формировании задания. Процесс создания ПО носит как правило, итерационный характер: результаты очередной стадии часто вызывают изменения в проектных решениях, выработанных на более ранних стадиях. Таким образом, постоянно возникает потребность в возврате к предыдущим стадиям и уточнении или пересмотре ранее принятых решений. В результате реальный процесс создания ПО принимает вид, изображенный на рисунке .1. Рис. 3.1. Модифицированная водопадная модель Наиболее распространенным результатом каскадного похода к разработке ПО является поздняя неудача. Кажется, что проекты выполняются нормально, но только до тех пор, пока работы не вступят в 24 завершающий этап, и тогда выясняется, что потребители недовольны созданным продуктом.

V-образная модель, как разновидность каскадной модели

Основной принцип V-образной модели заключается в том, что детализация проекта возрастает при движении слева направо, одновременно с течением времени, и ни то, ни другое не может повернуть вспять. Итерации в проекте производятся по горизонтали, между левой и правой сторонами буквы. V-модель – вариация каскадной модели, в которой задачи разработки 2

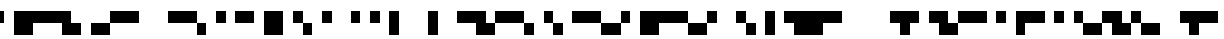
идут сверху вниз по левой стороне буквы V, а задачи тестирования – вверх по правой стороне буквы V. Внутри V проводятся горизонтальные линии, показывающие, как результаты каждой из стадий разработки влияют на развитие системы тестирования на каждой из стадий тестирования. Модель базируется на том, что приемо-сдаточные испытания основываются, прежде всего, на требованиях, системное тестирование – на требованиях и архитектуре, комплексное тестирование – на требованиях, архитектуре и интерфейсах, а компонентное тестирование – на требованиях, архитектуре, интерфейсах и алгоритмах. Подготовка процесса разработки Анализ требований к системе Проектирование системы Проектирование ПС Программирование и тестирование ПС Эксплуатация и сопровождение Ввод в действие и обеспечение приемки Сборка и квалификационные испытания системы Сборка и квалификационные испытания ПС. Особенностью данной модели является разбиение стадий на три логических этапа: проектирование (детализация требований), реализация, тестирование. V-модель дает организациям и проектным группам руководство по выполнению и завершению проектов последовательным и воспроизводимым образом. Применение принципов V-модели гарантирует выявление и фиксацию требований пользователей. Утвержденные требования могут быть переведены в функции готового приложения, (и) приложение отражает требования пользователей.

Методологии разработки ПО

Методологии представляют собой ядро теории управления разработкой ПО. К существующей классификации в зависимости от используемой в ней модели жизненного цикла (каскадные и эволюционные) добавилась более общая классификация на прогнозируемые и адаптивные методологии. Прогнозируемые (предикативные) методологии фокусируются на детальном планировании будущего. Известны запланированные задачи и ресурсы на весь срок проекта. Команда с трудом реагирует на возможные изменения. План оптимизирован исходя из состава работ и существующих требований. Изменение требований может привести к существенному изменению плана, а также дизайна проекта. Адаптивные (гибкие) методологии нацелены на преодоление ожидаемой неполноты требований и их постоянного изменения. Когда меняются требования, команда разработчиков тоже меняется. Команда, участвующая в адаптивной разработке, с трудом может предсказать будущее проекта. Существует точный план лишь на ближайшее время. Более удаленные во времени планы существуют лишь как декларации о целях проекта, ожидаемых затратах и результатах. Среди адаптивных методологий: (Scrum, Crystal, Extreme Programming, Adaptive Software Development, DSDM, Feature Driven Development, Lean software development). Рассмотрим самые основные и популярные методологии.

RUP (Rational Unified Process) Один из самых известных процессов, использующих итеративную модель разработки – RUP. Он был создан во второй половине 1990-х годов в компании Rational Software. Термином RUP обозначает как методологию, так и продукт компании IBM (ранее Rational) для управления процессом разработки. Методология RUP описывает абстрактный общий процесс, на основе которого организация или 31 проектная команда должна создать специализированный процесс, ориентированный на ее

требований в RUP используются прецеденты использования (use cases). Полный набор прецедентов использования системы вместе с логическими отношениями между ними называется моделью прецедентов использования. Каждый прецедент использования – это описание сценариев взаимодействия пользователя с системой, полностью выполняющего конкретную пользовательскую задачу. Согласно RUP все функциональные требования



проект RUP состоит из последовательности итераций с рекомендованной продолжительностью от 2 до 6 недель. Перед началом очередной итерации определяется набор прецедентов использования, которые будут реализованы к её завершению. Можно сказать, что RUP – ориентированная на архитектуру методология. Считается, что реализация и тестирование архитектуры системы должны начинаться на самых ранних стадиях проекта. RUP использует понятие исполняемой архитектуры (executable architecture) – основы приложения, позволяющей реализовать архитектурно значимые прецеденты использования. Основы исполняемой архитектуры должны быть реализованы как можно раньше. Это позволяет оценить адекватность принятых архитектурных решений и внести необходимые коррективы еще в начале проекта. Таким образом, для первых нескольких итераций необходимо выбирать прецеденты, которые требуют реализации большей части архитектурных компонентов. RUP поощряет использование визуальных средств для анализа и проектирования. Как правило, используется нотация и, соответственно, средства моделирования UML (такие как Rational Rose). Модель предметной области документируется в виде диаграммы классов, модель прецедентов использования – при помощи диаграммы прецедентов, взаимодействие компонентов системы между собой описывается диаграммой последовательности и т.д.

Жизненный цикл проекта

Жизненный цикл проекта RUP состоит из четырех фаз. Последовательность этих фаз фиксирована, но число итераций, необходимых для завершения каждой фазы, определяется индивидуально для каждого конкретного проекта. Фазы RUP нельзя

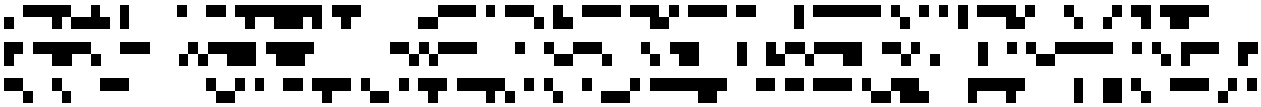
отождествлять с фазами водопадной модели – их назначение и содержание принципиально различны. Начало (Inception) Стадия «начало» обычно состоит из одной [illegible] прецедентов использования и подробно описа[illegible] риски проекта. Если после завершения первой итерации заинтересованные лица приходят к выводу о целесообразности выполнения проекта, проект переходит в следующую стадию. В противном случае проект может быть отменен или проведена еще одна итерация стадия «начало». Проектирование (Elaboration) В результате выполнения этой стадии на основе требований и рисков проекта создается основа архитектуры системы. Проектирование может занимать до двух-трех итераций или быть полностью пропущенным (если в проекте используется архитектура существующей системы без [illegible] прецедентов использования[illegible] рисков и уточнение бюджета и графика проекта. В отличие от каскадной модели, основным результатом этой стадии является не множество документов со спецификациями, а действующая система с 20-30% реализованных прецедентов использования. Построение (Construction) В этой стадии (длящейся от двух до четырех итераций) происходит разработка окончательного продукта. Вовремя ее выполнения создается основная часть исходного кода системы и выпускаются промежуточные демонстрационные прототипы. Внедрение (Transition) Целями стадии «внедрения» являются проведение бетатестирования и тренингов пользователей, исправление обнаруженных дефектов, развертывание системы на рабочей площадке, при необходимости – миграция данных. Кроме того, на этой стадии выполняются задачи, необходимые для проведения маркетинга и продаж. Стадия «внедрения» занимает от одной до трех итераций. После ее завершения проводится анализ результатов выполнения всего проекта: что можно изменить для улучшения эффективности в будущих проектах. Рабочий процесс В терминах RUP участники проектной команды создают так называемые артефакты (work products), выполняя задачи (tasks) в рамках определенных ролей (roles). Артефактами являются спецификации, модели, исходный код и т.п. Задачи разделяются по девяти процессным областям, называемым дисциплинами (discipline). В RUP определены шесть инженерных и три вспомогательные дисциплины. В них вхо[illegible] Бизнес-моделирование (Business Modeling) – исследование и описание существующих бизнес-[illegible] требованиями (Requirements Management) – определение границ проекта, разработка функционал[illegible] проектирование (Analysis and Design) – проектирование архитектуры системы на основе [illegible] (Implementation) – разработка, юнит-[illegible] Тестирование (Test) – поиск и отслеживание дефектов в системе, проверка корректности [illegible] – создание дистрибутива, установка системы, обучен[illegible] изменениями (Configuration and Change Management) – управление версиями исходного [illegible] Управление проектом (Project Management) – создание проектной команды, планирование [illegible] – создание инфраструктуры для выполнения проекта, включая организацию и настройку процесса разработки. Рис. 3.5 Распределение усилий при выполнении проекта Заключение В ходе жизненного цикла проекта распределение усилий проектной команды между

дисциплинами постоянно меняется. Например, как правило, в начале проекта большая часть усилий затрачивается на анализ и дизайн, а ближе к завершению – на реализацию и тестирование системы. Однако в общем случае задачи из всех девяти дисциплин выполняются параллельно. Для полноценного внедрения RUP организация должна затратить значительные средства на обучение сотрудников. При этом попытка обойтись своими силами скорее всего будет обречена на неудачу – необходимо искать специалиста по процессам (process engineer) с соответствующим опытом или привлекать консультантов. 3.2.2. Microsoft Solutions Framework (MSF) Данная методология описывает подход и организацию работы при создании программных продуктов. Подробно про методологию MSF вы 35 можете прочитать в переводе Microsoft Solutions Frameworks for Agile Software Development, которая входит в поставку Microsoft Team Foundation Server.
Scrum Scrum предоставляет эмпирический подход к разработке ПО. Этот процесс быстр, адаптивен, умеет подстраиваться и отличен от каскадной модели. Scrum основан на повторяющихся циклах, это делает его более гибким и предсказуемым. Для начала определим роли, которые участвуют в процессе: Scrum мастер (Scrum Master), Владелец продукта (Product Owner), Команда (Team). Scrum Мастер - самая важная роль в методологии. Scrum Мастер отвечает за успех Scrum в проекте. Как правило, эту роль в проекте играет менеджер проекта или лидер команды (Team Leader). Важно подчеркнуть, что Scrum Мастер не раздает задачи членам команды. В Scrum команда является самоорганизующейся и самоуправляемой. Основные обязанности Scrum Мастера таковы:

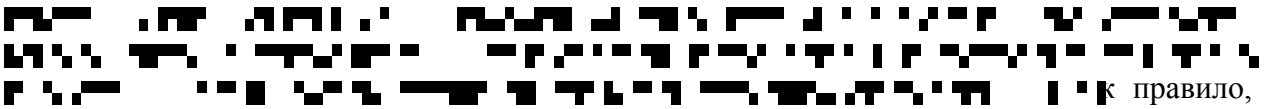
соблюдение практик и процесса в команде Scrum Мастер отслеживает прогресс команды при помощи Sprint Backlog, отмечая статус всех задач в спринте. Scrum Мастер может также помогать заказчику создавать список задач для команды Product Owner - это человек, отвечающий за разработку продукта. Как правило представитель заказчика для заказной разработки. Владелец продукта - это единая точка принятия окончательных решений для команды в проекте, именно поэтому это всегда один человек, а не группа или комитет. Команда (Team) - в методологии Scrum команда является самоорганизующейся и самоуправляемой. Команда берет на себя обязательства по выполнению объема работ на спринт перед Владельцем продукта. Работа команды оценивается как работа единой группы. В Scrum вклад отдельных членов проектной команды не оценивается, так как это разваливает самоорганизацию команды. Размер команды ограничивается размером группы людей, способных эффективно взаимодействовать лицом к лицу. Типичные 36 размер команды - 7 плюс минус 2. Команда в Scrum кроссфункциональна. В нее входят люди с различными навыками - разработчики, аналитики, тестировщики. Нет заранее определенных и поделенных ролей в команде, ограничивающих область действий членов команды. Команда состоит из инженеров, которые вносят свой вклад в общий успех проекта в соответствии со своими способностями и проектной необходимостью. Scrum В основе лежат короткие ежедневные встречи – Scrum и циклические 30-дневные встречи, называемые спринтом. Результатом спринта является готовый продукт, который можно передавать заказчику (по крайней мере, система должна быть готова к показу заказчику). Короткие спринты обеспечивают быструю обратную связь проектной команды с заказчиком. Заказчик получает возможность гибко управлять системой, оценивая результат спринта и предлагая улучшения к созданной функциональности. Такие улучшения попадают в список имеющихся на данный момент бизнес-требований и технических требований к системе (Product Backlog), приоритезируются наравне с прочими требованиями и могут быть запланированы на следующий (или на один из следующих) спринтов. Каждый спринт представляет собой маленький «водопад». В течение спринта делаются все работы по сбору требований, дизайну, кодированию и

тестированию продукта. Цель спринта должна быть фиксированным. Это позволяет команде давать обязательства на тот объем работ, который должен быть сделан в 37 спринте. Это означает, что Sprint Backlog не может быть изменен никем, кроме команды. 3.2.4. Экстремальное программирование (eXtreme Programming) Методология XP, разработанная Кентом Беком (Kent Beck), Уордом Каннингемом (Ward Cunningham) и Роном Джеффрисом (Ron Jeffries), является сегодня одной из самых популярных гибких методологий. Она описывается как набор практик: игра в планирование, короткие релизы, метафоры, простой дизайн, переработки кода (refactoring), разработка «тестами вперед», парное программирование, коллективное владение кодом, 40-часовая рабочая неделя, постоянное присутствие заказчика и стандарты кода. Интерес к XP рос снизу вверх – от разработчиков и тестировщиков, замученных тягостным процессом, документацией, метриками и прочим формализмом. Они не отрицали дисциплину, но не желали бессмысленно соблюдать формальные требования и искали новые быстрые и гибкие подходы к разработке высококачественных программ. При использовании XP тщательное предварительное проектирование ПО заменяется, с одной стороны, постоянным присутствием в команде заказчика, готового ответить на любой вопрос и оценить любой прототип, а с другой – регулярными переработками кода (так называемый рефакторинг). Основой проектной документации считается тщательно прокомментированный код. Очень большое внимание в методологии уделяется тестированию. Как правило, для каждого нового метода сначала пишется тест, а потом уже разрабатывается собственно код метода до тех пор, пока тест не начнет выполняться успешно. Эти тесты сохраняются в наборах, которые автоматически выполняются после любого изменения кода. Хотя парное программирование и 40-часовая рабочая неделя и являются, возможно, наиболее известными чертами XP, но все же носят вспомогательный характер и способствуют высокой производительности разработчиков и сокращению количества ошибок при разработке.

Легковесная гибкая методология, созданная Алистером Коуберном, которая предназначена для небольших команд в 6-8 человек для разработки некритичных бизнес-приложений. Как и все гибкие методологии, Crystal Clear больше опирается на людей, чем на процессы и артефакты. Crystal Clear использует семь методов/практик, три из которых



XP по производительности, зато максимально проста в использовании. Она требует минимальных усилий для внедрения, поскольку ориентирована на человеческие привычки. Считается, что эта методология описывает тот естественный порядок разработки ПО, который устанавливается в достаточно квалифицированных коллективах, если в них не занимаются целенаправленным внедрением другой методологии. Основные



к правило, используются средства контроля версий кода. В графическом виде практики Crystal Clear можно изобразить таким образом. Графическое представление Crystal Clear Заключение В данной главе были рассмотрены только несколько самых основных методологий, на самом деле их намного больше, например ICONIX, Канбан, Feature-driven development и другие. Каждая методология применяется в зависимости от типа программного продукта, а также определяет процесс проектирования ПО.

Вопросы и задания для самоконтроля

1. Что понимается под моделью ЖЦ ПО? Назовите существующие модели ЖЦ ПО.
2. Чем модель ЖЦ ПО отличается от методологии разработки ПО? Назовите существующие гибкие методологии разработки ПО.

3. Назовите основные особенности и стадии «Каскадной модели».
4. Назовите основные особенности и стадии «Эволюционной модели».
5. Методология Scrum. Что такое Спринт в рамках методологии Scrum? Какие группы ролей определены в данной методологии?

КАЧЕСТВО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Процессы разработки, приобретения и внедрения сложных систем, к которым относятся в частности программные комплексы, должны находиться под жестким управленческим контролем. В настоящее время практически во всех организациях обеспечивается контроль важнейших характеристик, связанных с производством и использованием программных продуктов, таких как время, финансовые средства, ресурсы и т.п. Однако в большинстве случаев вне пределов сферы контроля оказывается наиболее важная характеристика программных продуктов, ради которой, собственно и осуществляются затраты времени, финансовых средств и ресурсов – это качество продукта, поскольку «невозможно контролировать то, что нельзя измерить» («You cannot control what you cannot measure»). Отсутствие возможности установки полного контроля вызывает рост количества необоснованных решений, увеличивает финансовые и проектные риски, связанные с разработкой и внедрением систем. Однако в настоящее время уже существуют организации, в которых накоплен достаточно большой опыт использования метрик в управлении качеством разрабатываемых и внедряемых программных продуктов. Использование апробированных подходов в управлении качеством разработки и внедрения крупных программных систем значительно повышает предсказуемость проектов, снижает финансовые и ресурсные издержки. Сейчас существует несколько определений качества, которые в целом совместимы друг с другом. К числу наиболее распространенных относятся [7]: Определение 1 (ISO): Качество – это полнота свойств и характеристик продукта, процесса или услуги, которые обеспечивают способность удовлетворять заявленным или подразумеваемым потребностям. Определение 2 (IEEE): Качество программного обеспечения – это степень, в которой оно обладает требуемой комбинацией свойств. Анализ всех составляющих качества должен проводиться с учетом сфер ответственности заинтересованных сторон, как внутренних участников исполняемого процесса (in-process stakeholder), так и пользователей процесса (end-of-process stakeholders). Очевидно, что управление качеством требует контроля всех измерений и оценок качества в жизненном цикле ПС.

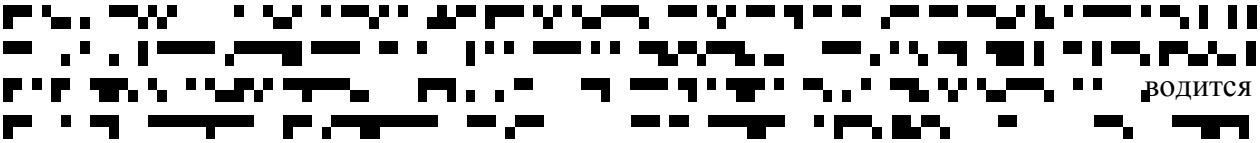
Измерение и оценка характеристик качества ПО. Программное обеспечение, в зависимости от особенностей разработки и применения, может представлять программу, программный комплекс, программное средство или программный продукт (изделие). Введем и будем в дальнейшем использовать следующие определения. Качество программного обеспечения – это совокупность свойств, характеризующих способность программного обеспечения удовлетворять потребностям пользователя в соответствии с предназначением. Анализ качества (продукта, процесса) Сферы ответственности заинтересованных сторон Управление качеством (будет успешным, если под контролем находятся все измерения и оценка качества) измерения и оценки внутрение факторы внешние факторы Рис. 4.1 Анализ Измерения и оценки качества в контуре управления качеством Управление качеством – это система организационных, экономических, технологических и правовых мероприятий, осуществляемых для удовлетворения требований к качеству программного обеспечения в течение жизненного цикла. Свойства программы – это особенности, объективно присущие программе, которые проявляются в ее жизненном цикле (разработке, применении, сопровождении). Характеристика программы – это понятие, отражающее проявление отдельного измеримого фактора присущего программе свойства. Иначе говоря, характеристика – это проявляемый и

измеримый атрибут свойства. Измерение (оценка) одной или нескольких характеристик программы дает представление о том, насколько программе присуще то или иное свойство. Каждому свойству соответствует одна или несколько характеристик программного обеспечения. Для решения задачи количественной оценки характеристик программного обеспечения необходимо наличие системы измерений и методов оценки. Система измерений характеристик программного обеспечения – это совокупность измеряемых характеристик, единиц измерения, измерительных шкал и связей, установленных между ними. Если между измеряемыми характеристиками установлены иерархические связи, систему измерений называют иерархической, в противном случае – одноранговой. Измерительная шкала устанавливает границы (диапазон) и точность измерений характеристик свойств в установленных единицах. Результаты измерений в избранной измерительной шкале позволяют обнаружить сходство и различие в свойствах программного обеспечения с целью последующей оценки и классификации. Применительно к ПО используют главным образом следующие известные виды измерительных шкал: номинальные (категорийные), порядковые, интервальные. Номинальная (категорийная) шкала фиксирует наличие или отсутствие некоторой характеристики свойства без учета градаций и позволяет классифицировать программы по этому принципу. Порядковая шкала фиксирует отношение порядка и позволяет ранжировать программы относительно некоторого опорного значения характеристик свойств. Интервальная шкала фиксирует не только отношение порядка, но и величину, отличающую одно значение характеристики от другого (интервал между значениями). Методы оценки характеристик программного обеспечения делят на следующие шесть групп: измерительные, регистрационные, органолептические, расчетные, экспертные, социологические, традиционные. Измерительные методы основаны на получении информации о характеристиках программного обеспечения с использованием специальных инструментальных средств (технических или программных средств, обеспечивающих проведение измерений и их автоматизацию). Регистрационные методы основаны на получении информации о характеристиках программного обеспечения во время испытаний или функционирования путем регистрации и подсчета определенных событий (например, моментов и количества ошибок, времени начала и окончания расчетов и т.д.), регистрируемых извне программы с помощью средств измерений общего назначения. Органолептические методы основаны на получении информации о характеристиках программного обеспечения путем их восприятия органами чувств – в первую очередь зрения, слуха и осязания. Расчетные методы основаны на получении информации о характеристиках программного обеспечения за счет использования теоретических или эмпирических зависимостей. Экспертные методы используют опыт экспертов-специалистов, компетентных в оценке характеристик программного обеспечения. Социологические методы используют обработку специальных анкет-опросников, содержащих качественные оценки характеристик программного обеспечения социальными группами, имеющими отношение к применению программного обеспечения. Традиционные методы объединяют группу сформировавшихся и традиционно используемых в организациях, на предприятиях и иных учреждениях методов количественной оценки характеристик программного обеспечения. Проявляемые свойства программного обеспечения условно можно разделить на две группы (рис.): функциональные (внешние), конструктивные (внутренние). Для разработчиков и пользователей программы представляют интерес определенные функциональные и конструктивные свойства, (например, надежность, эффективность, модульность, структурность). Как правило, пользователя (заказчика) интересуют те функциональные свойства, которые характеризуют полезность программного обеспечения. Именно эти внешние свойства, отражающие точку зрения пользователя, обуславливают качество программного обеспечения, то есть являются его факторами. Заметим, что для

разработчиков представляют интерес не только внешние, но и внутренние, или конструктивные свойства, от которых зависит выполнение требований к программному обеспечению и восприятие его пользователем. Характеристики качества отражают свойства, определяющие качество программного обеспечения. В силу сложной природы количественной оценки характеристик качества программного обеспечения для их оценки используют иерархические системы измерений. Иерархию характеристик качества образуют факторы, критерии, метрики и оценочные элементы (рис.3). Факторы и критерии, составляющие два верхних уровня иерархии измерений, отражают функциональные характеристики программного обеспечения, а нижние (метрики и оценочные элементы) – конструктивные характеристики, от которых зависит качество программного обеспечения. Измеримость характеристик качества обеспечивается составом характеристик самого нижнего уровня – оценочных элементов. Фактором качества будем называть свойство, в той или иной степени обуславливающее качество программного обеспечения. При оценке качества учитывают несколько факторов. Для получения численной оценки фактора качества используют один или несколько критериев качества. Критерий качества – это понятие, признак или численный показатель, характеризующий оцениваемый фактор качества. Критерий качества может быть представлен имеющим физический смысл вычислимым выражением, составленным из характеристик качества, значением которого является показатель качества. Для вычисления значения критерия используют одну или несколько метрик. Метрика – мера количественной оценки качества ПО по заданному критерию, система или способ измерений качества программного обеспечения. Метрика содержит один или несколько оценочных элементов. Оценочный элемент – измеримая характеристика программного обеспечения, имеющая численное значение в избранной измерительной шкале. Показатель качества – численное значение критерия качества, определяющее степень, в которой программе присуще определенное критерием свойство. В соответствии с ГОСТ 15467-79 под показателем качества следует понимать количественную характеристику одного или нескольких свойств программной продукции, составляющих ее качество применительно к определенным условиям ее создания и эксплуатации. Комплексный показатель качества – показатель качества, значение которого получают в результате композиции значений других, в том числе комплексных показателей. Таким образом, качество ПС многомерное понятие. Базовое значение показателя качества – это реально достижимое значение показателя, отражающее современный уровень развития программного обеспечения. Совокупность операций, включающих выбор номенклатуры (состава) показателей качества, определения значений этих показателей и сравнения их с базовыми значениями, называют оценкой качества программного обеспечения. Процесс определения соответствия программного обеспечения действующему стандарту качества называют сертификацией. Процесс определения соответствия программного обеспечения предназначению называют верификацией. Процесс подтверждения функциональной пригодности программного обеспечения называют аттестацией. Очевидно, что сертификация, верификация и аттестация программного обеспечения не исключают, а предполагают проведение количественных оценок характеристик программ. Учитывая, что по определению программное обеспечение состоит не только из программ, но и документации к ним, одной из задач оценки качества ПО является измерение и оценка характеристик программных и эксплуатационных документов.

Концепция и сущность управления качеством ПС

Основное содержание концепции управления качеством сводится к следующим



водится

информационный и целенаправленный процесс воздействия на программы и документацию, а также на коллективы разработчиков ПС в целях обеспечения требуемого качества при изменяющихся внешних и внутренних условиях путем принятия управленческих решений. Следуя этой концепции, можно констатировать, что сущность управления качеством в процессе разработки ПС состоит из трех основных видов [illegible]ние множества организационных [illegible] качества. Выбор из этого множества соответствующего подмножества процедур и [illegible]ства. Определение и проведение мероприятий, гарантирующих выполнение нормативных процедур и стандартов качества всеми членами команды разработчиков ПО. 46 Свойства программного обеспечения Функциональные (внешние) Конструктивные (внутренние) Характеристики программного обеспечения Система измерений Методы оценки Характеристики качества программного обеспечения Показатели качества

Группы свойств и характеристики программного обеспечения Управление качеством предполагает возможность независимого контроля за процессом разработки ПС. Контрольные проектные элементы, получаемые в процессе разработки ПС, являются основой контроля качества. Они тщательно проверяются на соответствие стандартам и целям проекта (рис. 4.3Ошибка! Источник ссылки не найден.). Так как работы, выполняемые по обеспечению и контролю качества, в определенной степени независимы, это предполагает возможность объективного взгляда на процесс разработки ПС, благодаря чему руководство компании может своевременно получить информацию о проблемах или трудностях, которые возникают в работе над проектом. По мнению известного специалиста в области программной инженерии Иана Соммервилла, процесс управления качеством необходимо отделять от процесса управления проектом с тем, чтобы не ставить вопрос о компромиссе между качеством создаваемого ПО и бюджетом или графиком выполнения проекта. Над контролем качества должна работать независимая команда, которая отчитывается непосредственно руководству заказчика, минуя звено управляющего проектом (менеджера проекта). Вместе с тем признается факт, что команда контроля качества должна быть связана с группой (группой разработки) и несет ответственность за качество на уровне всей организации разработчика. Полезность программного обеспечения Исходная полезность Удобство применения Надежность Эффективность Корректность Понятность Простота Мобильность Факторы качества, отражающие полезность ПО ФАКТОРЫ КАЧЕСТВА КРИТЕРИИ КАЧЕСТВА МЕТРИКИ ОЦЕНОЧНЫЕ ЭЛЕМЕНТЫ. Модель измерений характеристик качества. Процесс разработки ПО Процесс управления качеством План обеспечения качества Отчеты по контролю качества Фазы стадии. Управление качеством и разработка ПС. Роль стандартизации и сертификации в управлении качеством ПС Одним из краеугольных камней современного управления качеством является стандартизация. По определению Международной организации по стандартизации (ISO) стандартизация представляет собой «процесс установления и применения правил с целью упорядочения в данной области на пользу и при участии всех заинтересованных сторон, в частности, для достижения всеобщей максимальной экономии с соблюдением функциональных условий и требований безопасности». В толковом словаре по информатике В.И. Першикова и В.М. Савинкова понятие стандартизация определяется как принятие соглашения по спецификации, производству и использованию аппаратных и программных средств вычислительной техники; установление и применение стандартов, норм, правил и т.п. [illegible]рядочивание объектов (продукции, [illegible] установление правил применения этих нормативных документов. На международном

[redacted]

содействует взаимообмену научно-[redacted] международной торговле [redacted]-технический прогресс участников международных организаций. Необходимость стандартизации разработки ПС на международном уровне, согласно стандарту ИСО/МЭК 12207 (введение), определена следующим образом: Программное обеспечение является неотъемлемой частью информационных технологий и традиционных систем, таких, как транспортные, военные, медицинские и финансовые. Имеется множество разнообразных стандартов, процедур, методов, инструментальных средств и типов операционной среды для разработки и управления программным обеспечением. Это разнообразие создает трудности при проектировании и управлении программным обеспечением, особенно при объединении программных продуктов и сервисных программ. Стратегия разработки программного обеспечения требует перехода от этого множества альтернатив к общему порядку, который позволит специалистам, практикующимся в программном обеспечении, «говорить на одном языке» при разработке и управлении программным обеспечением. Этот международный стандарт обеспечивает такой общий порядок". Таким образом, в процессе стандартизации вырабатываются нормы, правила, требования, характеристики, касающиеся объекта стандартизации, которые оформляются в виде нормативного документа. Поскольку сертификация устанавливает соответствие действующему стандарту, без наличия стандартов невозможна и сертификация. Виды нормативных документов, рекомендуемые международными организациями по стандартизации (ИСО/МЭК), а также принятые в государственной системе стандартизации. (стандарты, технические условия, своды правил, регламенты, положения). Стандарты предварительные международные, региональные, национальные, территориальные, отраслевые, ведомственные, внутрифирменные, предприятий Нормативные документы Регламенты (обязательные правовые нормы) Положения Документы технических условий. К типам данных этих языков отнесены указатели на данные различных типов или адреса данных и программных объектов.

Работа с большинством пакетов для разработки системного программного обеспечения предполагает знание и использование ассемблера для создания модулей и ассемблерных вставок. Управление вычислительными процессами и вычислительными комплексами и работа с внутренними данными ОС.

Как правило, они находятся в основной памяти. Это резидентные программы, составляющие ядро ОС. Управляющие программы, которые загружаются в память непосредственно перед выполнением, называю транзитными (transitive).

В настоящее время системные управляющие программы поставляются фирмами-разработчиками и фирмами-дистрибьюторами в виде инсталляционных пакетов операционных систем и драйверов специальных устройств.

Обрабатывающие системные программы выполняются как специальные прикладные задачи, или приложения.

Эти программы поставляются чаще в виде дистрибутивных пакетов, включающих ПО

Замечание. В пакеты системных программ помимо основных программ, допускающих реконфигурацию, входят ***специальные настроечные программы***, называемые программами инсталляции.

Другая классификация

Часто Системное ПО компьютера подразделяют на БАЗОВОЕ и СЕРВИСНОЕ программное обеспечение.

БАЗОВОЕ программное обеспечение (base software) - минимальный набор программных средств, обеспечивающих работу компьютера.

К базовому программному обеспечению компьютера относятся

· операционные системы и драйверы в составе ОС;
· интерфейсные оболочки для взаимодействия пользователя с ОС (операционные оболочки) и программные среды;
· системы управления файлами.
Операционная система - совокупность программных средств, обеспечивающая управление аппаратной частью компьютера и прикладными программами, а также их взаимодействием между собой и пользователем.
Операционная система предназначена для управления выполнением пользовательских программ, планирования и управления вычислительными ресурсами ЭВМ.
Операционная система, с одной стороны, выступает как интерфейс между аппаратурой компьютера и пользователем с его задачами, с другой стороны, предназначена для эффективного использования ресурсов вычислительной системы и организации надежных вычислений.
Системы управления файлами предназначены для организации более удобного доступа к данным, организованным как файлы. Вместо низкоуровневого доступа к данным с указанием конкретных физических адресов система управления файлами позволяет использовать логический доступ с указанием имени файла.
Любая система управления файлами не существует сама по себе - она разработана для работы в конкретной ОС и с конкретной файловой системой. То есть можно было бы систему управления файлами отнести к ОС.
Но в связи с тем, что
1) ряд ОС позволяет работать с несколькими файловыми системами (либо с одной из нескольких, либо сразу с несколькими одновременно); а дополнительную файловую систему можно установить (т.е. они самостоятельны)
2) простейшие ОС могут работать и без файловых систем;
системы управления файлами выделяются в отдельную группу системных программ.
Заметим, что часто в специальной литературе системы управления файлами относят все-таки к операционным системам.
СЕРВИСНОЕ программное обеспечение - программы и программные комплексы, которые расширяют возможности базового программного обеспечения и организуют более удобную среду работы пользователя.
Это набор сервисных, дополнительно устанавливаемых программ, которые можно классифицировать по функциональному признаку следующим образом:
· драйверы специфических и специальных устройств (те, которые не поставляются в составе ОС).
· программы диагностики работоспособности компьютера;
· антивирусные программы, обеспечивающие защиту компьютера, обнаружение и восстановление зараженных файлов;
· программы обслуживания дисков, обеспечивающие проверку качества поверхности магнитного диска, контроль сохранности файловой системы на логическом и физической уровнях, сжатие дисков, создание страховых копий дисков, резервирование данных на внешних носителях и др.;
· программы архивирования данных, которые обеспечивают процесс сжатия информации в файлах с целью уменьшения объема памяти для ее хранения;
· программы обслуживания сети.
Эти программы часто называются **системными утилитами**. (Заметим, что к антивирусным средствам этот термин обычно не применяется)
Утилиты - программы, служащие для выполнения вспомогательных операций обработки данных или обслуживания компьютеров (диагностики, тестирования аппаратных и программных средств, оптимизации использования дискового пространства, восстановления разрушенной на магнитном диске информации и т.п.). Наибольшее

распространение сегодня имеют комплекты утилит: Norton Utilities - фирма Symantec; Checkit PRO Deliuxe 2.0 - фирма Touch Stone; PC Tools for Windows 2.0; программа резервного копирования HP Colorado Backup for Windows 95.

Системы программирования

Отдельно рассмотрим такую группу системного ПО как **системы программирования**.

Это набор специализированных программных продуктов, которые являются инструментальны средствами разработчика. Программные продукты данного класса поддерживают все этапы процесса программирования, отладки и тестирования создаваемых программ.

Система программирования включает следующие программные компоненты:

· редактор текста;
· транслятор с соответствующего языка;
· компоновщик (редактор связей);
· отладчик;
· библиотеки подпрограмм.

Заметим, что любая система программирования может работать только в соответствующей ОС, под которую она и создана, однако при этом она может позволять разрабатывать программное обеспечение и под другие ОС.

Например, одна из популярных систем программирования на языке C/C++ от фирмы Watcom для OS/2 позволяет получать программы и для самой OS/2, и для DOS, и для Windows.

Редактор текста - это программа для ввода и модификации текста.

Трансляторы предназначены для преобразования программ, написанных на языках программирования, в программы на машинном языке. Программа, подготовленная на каком-либо языке программирования, называется исходным модулем. В качестве входной информации трансляторы применяют исходные модули и формируют в результате своей работы объектные модули, являющиеся входной информацией для редактора связей. Объектный модуль содержит текст программы на машинном языке и дополнительную информацию, обеспечивающую настройку модуля по месту его загрузки и объединение этого модуля с другими независимо оттранслированными модулями в единую программу.

Трансляторы делятся на два класса: компиляторы и интерпретаторы. Компиляторы переводят весь исходный модуль на машинный язык.

МЕТОДЫ ВЫЯВЛЕНИЯ ТРЕБОВАНИЙ К ПО. УРОВНИ ТРЕБОВАНИЙ. АНАЛИЗ ТРЕБОВАНИЙ К ПО

Цель анализа требований в проектах – получить максимум информации о заказчике и специфике его задач, уточнить рамки проекта, оценить возможные риски, а также сформировать проектную группу, на которую будет возложена значительная часть предстоящих работ. На этом этапе происходит идентификация принципиальных требований методологического и технологического характера, формулируются цели и задачи проекта, а также определяются критические факторы успеха, которые впоследствии будут использоваться для оценки результатов внедрения. Анализ требований выполняется на основе совещаний и собеседований с руководителями и специалистами заказчика, а продолжительность этого этапа, в зависимости от сложности задач и масштаба внедрения, может составлять от нескольких дней до нескольких недель. Определение и описание требований (методологических и технических) – шаги, которые во многом определяют успех всего проекта, поскольку именно они влияют на все остальные этапы. Практика показывает, что недостаточная проработка требований зачастую проявляется лишь тогда, когда проект почти завершен, а значительная часть ресурсов, выделенных на его реализацию, уже затрачена. К сожалению, устранение

проблем на этапе разработки обходится гораздо дороже, чем тщательная проработка на стадии анализа.

1. **Особенности интерпретации требований IEEE Standard Glossary of Software Engineering Terminology** (1990) определяет требования как:

1. Условия или возможности, необходимые пользователю для решения проблем или достижения целей;

2. Условия или возможности, которыми должна обладать система или системные компоненты, чтобы выполнить контракт или удовлетворять стандартам, спецификациям или другим формальным документам;

3. Документированное представление условий или возможностей для пунктов 1 и 2. Это определение охватывает требования как пользователей (внешнее поведение системы), так и разработчиков (некоторые скрытые 54 параметры). Термин пользователи следует распространить на всех заинтересованных лиц, так как не все, кто заинтересован в проекте – пользователи. Требования – это спецификация того, что должно быть реализовано. В них описано поведение системы, свойства системы или ее атрибуты. Они могут быть ограничены процессом разработки системы. Уровни требований. Три уровня требований к

2. Типы требований.

Каждая система имеет свои функциональные и нефункциональные требования. Бизнес-требования (business requirements) содержат высокоуровневые цели организации или заказчиков системы. Как правило, их высказывают те, кто финансируют проект, покупатели системы, менеджер реальных пользователей, отдел маркетинга. Бизнестребования формулируют в документе об образе и границах проекта – уставе проекта (project charter), или документом рыночных требований (market requirements document). Определение границ проекта представляет собой первый этап управление общими проблемами расползания границ. Требования пользователей (user requirements) описывают цели и задачи, которые пользователям позволит решить система. К отличным способам представления этого вида требований относятся варианты использования, сценарии и таблицы «событие – отклик». Таким образом, в этом документе указано, что клиенты смогут делать с помощью системы. Функциональные требования (functional requirements) определяют функциональность ПО, которую разработчики должны построить, чтобы пользователи смогли выполнить свои задачи в рамках бизнес-требований. Иногда именуемые требованиями поведения (behavioral requirements), они содержат положения с традиционным «должен» или «должна»: «Система должна по электронной почте отправлять пользователю подтверждение о заказе». Термином системные требования (system requirements) обозначают высокоуровневые требования к продукту, которые содержат многие подсистемы, то есть система (IEEE, 1998c). Говоря о системе, мы подразумеваем программное обеспечение или подсистемы ПО и оборудования. Люди – часть системы, поэтому определенные функции системы могут распространяться и на людей. Бизнес-правила (business rules) включают корпоративные политики, правительственные постановления, промышленные стандарты и вычислительные алгоритмы. Бизнес-правила не являются требованиями к ПО, потому что они находятся снаружи границ любой системы ПО. Однако они часто налагают ограничения, определяя, кто может выполнять конкретные варианты использования, или диктовать, какими функциями должна обладать система, подчиняющаяся соответствующим правилам. Иногда бизнес-правила становятся источником атрибутов качества, которые реализуются в функциональности. Следовательно, вы можете отследить происхождение конкретных функциональных требований вплоть до соответствующих им бизнес-правил. Нефункциональные требования содержат цели и атрибуты качества. Атрибуты качества (quality attributes) представляют собой дополнительное описание функций продукта, выраженное через описание его характеристик, важных для

пользователей или разработчиков. К таким характеристикам относятся легкость и простота использования, легкость перемещения, целостность, эффективность и устойчивость к сбоям. Другие нефункциональные требования описывают внешние взаимодействия между системой и внешним миром, а также ограничения дизайна и реализации. Ограничения (constraints) касаются выбора возможности разработки внешнего вида и структуры продукта. Спецификация требований не содержит деталей дизайна или реализации (кроме известных ограничений), данных о планировании проекта или сведений о тестировании.

3. Приемы формулирования требований

Обучение аналитиков требований. Всем членам команды, которые будут исполнять функции аналитиков, необходимо научиться приемам формулирования требований – это может занять несколько дней. Квалифицированный аналитик требований терпелив и методичен, обладает навыками межличностного общения и коммуникативными навыками, сведущ в предметной области и знает множество способов формулирования требований к ПО. 56 Ознакомление пользователей и менеджеров с требованиями. Пользователи, которые будут принимать участие в разработке ПО, должны пройти непродолжительный тренинг (один-два дня), чтоб научиться формулировать требования. Он полезен и для менеджеров по разработке и по работе с клиентами. Обучение поможет понять особое значение выделения требований, суть процесса их разработки, а также опасность пренебрежения ими. Посетив мои семинары по требованиям, некоторые пользователи замечали, что стали теплее относиться к разработчикам ПО. Ознакомление разработчиков с концепциями предметной области. Чтобы помочь разработчикам в общих чертах понять предметную область, проведите семинар, на котором познакомьте их с бизнесом клиента, терминологией и назначением создаваемого продукта. Это уменьшит вероятность путаницы, непонимания и доработок. Можно также на время проекта назначить каждому разработчику «личного пользователя», который будет разъяснять профессиональные термины и бизнес-концепции. Лучше, если это будет настоящий фанат продукта. Создание бизнес-словаря. Словарь со специализированными терминами из предметной области снизит вероятность непонимания, включите в него синонимы, термины, имеющие несколько значений, и термины, имеющие в предметной области и повседневной жизни разные значения.

4. Выявление требований

Определение процесса формулирования требований. Документация этапов выявления, анализа, определения и проверки требований. Наличие инструкций по выполнению ключевых операций поможет аналитикам качественно и согласованно выполнить их работу. Кроме того, будет проще поставить задачи по созданию требований и графики, а также продумать необходимые ресурсы. Определение образа и границы проекта. Документ об образе и границах проекта содержит бизнес-требования к продукту. Описание образа проекта позволит всем заинтересованным лицам в общих чертах понять назначение продукта. Границы проекта определяют, что следует реализовать в этой версии, а что – в следующих. Образ и границы проекта – хорошая база для оценки предлагаемых требований, Образ продукта должен оставаться от версии к версии относительно стабильным, но для каждого выпуска необходимо составлять отдельный документ о границах. Определение классов пользователей и их характеристик. Чтобы не упустить из виду потребности отдельных пользователей, необходимо их объединить в группы. Например, по частоте работе с ПО, используемым функциям, уровню привилегий и навыкам работы. Производится описание их обязанности, местоположение и личные характеристики, способные повлиять на архитектуру продукта. Выбор сторонника продукта (product champion) в каждом классе пользователей. Это человек, который сможет точно передавать настроения и нужды клиентов. Он представляет потребности определенного класса пользователей и принимает решения от их лица. В случае

разработки внутрикорпоративных информационных систем, когда все пользователи – ваши коллеги, такого человека выбрать проще. При коммерческой разработке расспросите клиентов или используйте площадки бета-тестирования. Выбранные вами люди должны принимать постоянное участие в проекте и обладать полномочиями для принятия решений, касающихся пользовательских требований. Создание фокус-групп типичных пользователей. Определите группы типичных пользователей предыдущих версий вашего продукта или похожих. Выясните у них подробности о функциональности и качественных характеристиках разрабатываемого продукта. Фокусгруппы особенно значимы при разработке коммерческих продуктов, когда приходится иметь дело с большой и разнородной клиентской базой. В отличие от сторонников продукта, у фокус-групп обычно нет полномочий на принятие решений. Работа с пользователями для выяснения назначения продукта. Выясните у пользователей, какие задачи им требуется выполнять средствами ПО. Обсудите, как должен клиент взаимодействовать с системой для выполнения каждой такой задачи. Воспользуйтесь стандартным шаблоном для документирования всех задач и для каждой сформулируйте функциональные требования. Похожий способ, часто применяемый в правительственных проектах, — создать документ с концепциями операций (ConOps), где указаны характеристики новой системы сточки зрения пользователя (IEEE, 1998a). Определение системных событий и реакции на них. Определите возможные внешние события и ожидаемую реакцию системы на них. Это могут быть сигналы и данные, получаемые от внешнего оборудования, а также временные события, вызывающие ответную реакцию, например ежевечерняя передача данных, генерируемых системой, внешнему объекту. В бизнес-приложениях бизнес-события напрямую связаны с задачами. Проведение совместных семинаров. Совместные семинары по выявлению требований, где тесно сотрудничают аналитики и клиенты – отличный способ выявить нужды пользователей и составить наброски документов с требованиями (Gottesdiener, 2002). Конкретные примеры таких семинаров – Joint Requirements Planning (JRP – совместное планирование требований) (Martin, 1991) и Joint Application Development (JAD – совместная разработка приложений) (Wood и Silver, 1995). Наблюдение за пользователями на рабочих местах. Наблюдая за работой пользователей, выявляют контекст потенциального применения нового продукта (Beyer и Holtzblatt, 1998). Простые диаграммы рабочих потоков, а также диаграммы потоков данных позволяют выяснить, где, как и какие данные задействовал пользователь. Документируя ход бизнес-процесса, удается определить требования к системе предназначенной для поддержки этого процесса. Иногда даже выясняется, что для выполнения деловых задач клиентам вовсе и не требуется новое ПО (McGraw и Harbison, 1997). Изучение отчетов о проблемах работающих систем с целью по иска новых идей. Поступающие от клиентов отчеты о проблемах и предложения о расширении функциональности — отличный источник: идей о возможностях, которые можно реализовать в следующей версии или новом продукте. За подобной информацией стоит обратиться и к персоналу службы поддержки. Повторное использование требований в разных проектах. Если необходимая клиенту функциональность аналогична уже реализованной в другом продукте, подумайте, готовы ли клиенты гибко пересмотреть свои требования для использования существующих компонентов Требования, соответствующие бизнес-правилам компании, можно применить в нескольких проектах. Это требования к безопасности, определяющие порядок доступа к приложениям, и требования, соответствующие постановлениям правительства, например Закон о гражданах США с ограниченными возможностями (Americans with Disabililies Act).

Анализ требований

Создание контекстной диаграммы. Контекстная диаграмма – простая модель анализа, отображающая место новой системы в соответствующей среде. Она определяет границы и

интерфейсы между разрабатываемой системой и сущностями, внешними для этой системы, например пользователями, устройствами и прочими информационными системами. Создание пользовательского интерфейса и технических прототипов. Если разработчики или пользователи не совсем уверены насчет требований, создайте прототип – частичную, возможную или предварительную версию продукта, которая сделает концепции и возможности более осязаемыми. Оценка прототипа поможет всем заинтересованным лицам достичь взаимопонимания по решаемой проблеме. Анализ осуществимости требований. Проанализируйте, насколько реально реализовать каждое требование при разумных затратах и с приемлемой производительностью в предполагаемой среде. Рассмотрите риски, связанные с реализацией каждого требования, включая конфликты с другими требованиями, зависимость от внешних факторов и препятствия технического характера. Определение приоритетов требований. Воспользуйтесь аналитическим подходом и определите относительные приоритеты реализации функций продукта, решаемых задач или отдельных требований. На основании приоритетов установите, в какой версии будет реализована та или иная функция или набор требований. Подтверждая изменения, распределите все их по конкретным версиям и включите в план выпуска этих версий затраты, необходимые на внесение изменений. В ходе работы над проектом периодически корректируйте приоритеты в соответствии с потребностями клиента, условиями рынка и бизнесцелями. Моделирование требований. В отличие от подробной информации, представленной в спецификации требований к ПО или пользовательского интерфейса прототипа, графическая модель анализа отображает требования на высоком уровне абстракции. Модели позволяют выявить некорректные, несогласованные, отсутствующие и избыточные требования. К таким моделям относятся диаграммы потоков данных, диаграммы «сущность – связь», диаграммы перехода состояний, называемые также автоматами (statecharts), карты диалогов, диаграммы классов, диаграммы последовательностей, диаграммы взаимодействий, таблицы решений и деревья решений. Создание словаря терминов. В нем соберите определения всех элементов и структур данных, связанных с системой, что позволяет всем участникам проекта использовать согласованные определения данных. На стадии работы над требованиями словарь должен содержать определения элементов данных, относящихся к предметной области, чтобы клиентам и разработчикам было проще общаться. Распределение требований по подсистемам. Требования к сложному продукту, включающему несколько подсистем, следует соразмерно распределять между программными, аппаратными и операторскими подсистемами и компонентами (Nelsen, 1990). Как правило, это осуществляет системный инженер или разработчик. Применение технологий развертывания функций качества. Технология развертывания функций качества (Quality Function Deployment, QFD) – точная методика, соотносящая возможности и атрибуты продукта с их значимостью для клиента (Zultner, 1993; Pardee, 1996). Она позволяет аналитически выявить функции, которые максимально удовлетворят потребности клиента. Технология развертывания функций качества рассчитана на три класса требований: ожидаемые, о которых клиент может не упомянуть, но будет расстроен, если их не окажется в продукте, обычные требования и отдельные, специальные требования, которые обеспечивают удобство работы клиентам, но отсутствие которых не влечет санкций со стороны клиента [12]. 5.6. Спецификации требований Использование шаблона спецификации требований к ПО. Создайте стандартный шаблон для документирования требований к ПО в вашей организации. Шаблон предоставляет согласованную структуру, позволяющую фиксировать описания нужной функциональности, а также прочую информацию, касающуюся требований. Вместо того чтобы изобретать новый шаблон, модифицируйте один из существующих в соответствии со спецификой проекта. Многие компании начинают с использования шаблона спецификации требований к ПО, описанного в стандарте IEEE 830-1998 (IEEE,

1998b). Если ваша компания занимается разными проектами, например, проектирует новое крупное приложение и параллельно дорабатывает версии старых программ, создайте соответствующие шаблоны для всех типов проектов. Шаблоны и процессы должны быть масштабируемыми. Определение источников требований. Чтобы гарантировать, что все заинтересованные лица понимают, почему-то или иное требование зафиксировано в спецификации требований к ПО, и упростить последующее прояснение требований, выявите источники всех требований. Это может быть вариант использования или другая информация от пользователей, системное требование высокого уровня, бизнес-правило или иной внешний фактор. Указав всех лиц, заинтересованных в каждом требовании, вы будете знать, к кому обратиться при поступлении запроса на изменение. Источники 61 требований устанавливают на основе связей или определяют для этой цели атрибут требования. Присвоение уникальных идентификаторов всем требованиям. Выработайте соглашение о присвоении уникальных идентификаторов требованиям, зафиксированным в спецификации требований к ПО. Соглашение должно быть устойчивым к дополнению, удалению элементов и изменениям, вносимым в требования. Присвоение идентификаторов позволяет отслеживать требования и фиксировать вносимые изменения. Указание атрибутов качества. Выявляя качественные характеристики, удовлетворяющие потребности клиента, не ограничивайтесь только обсуждением функциональности. Выясните ожидаемые производительность, эффективность, надежность, удобство в использовании др. Информация от клиентов об относительной важности тех или иных качественных характеристиках позволит разработчику принять правильные решения, касающиеся архитектуры приложения. Документирование бизнес-правил. К бизнес-правилам относятся корпоративные политики, правительственные распоряжения и алгоритмы вычислений. Ведите список бизнес-правил отдельно от спецификации требований к ПО, поскольку правила обычно существуют вне рамок конкретного проекта. Для выполнения некоторых приходится создавать реализующие их функциональные требования, и поэтом необходимо определить связь между этими требованиями и соответствующими правилами.

Спецификации требований

Использование шаблона спецификации требований к ПО. Создайте стандартный шаблон для документирования требований к ПО в вашей организации. Шаблон предоставляет согласованную структуру, позволяющую фиксировать описания нужной функциональности, а также прочую информацию, касающуюся требований. Вместо того чтобы изобретать новый шаблон, модифицируйте один из существующих в соответствии со спецификой проекта. Многие компании начинают с использования шаблона спецификации требований к ПО, описанного в стандарте IEEE 830-1998 (IEEE, 1998b). Если ваша компания занимается разными проектами, например, проектирует новое крупное приложение и параллельно дорабатывает версии старых программ, создайте соответствующие шаблоны для всех типов проектов. Шаблоны и процессы должны быть масштабируемыми. Определение источников требований. Чтобы гарантировать, что все заинтересованные лица понимают, почему-то или иное требование зафиксировано в спецификации требований к ПО, и упростить последующее прояснение требований, выявите источники всех требований. Это может быть вариант использования или другая информация от пользователей, системное требование высокого уровня, бизнес-правило или иной внешний фактор. Указав всех лиц, заинтересованных в каждом требовании, вы будете знать, к кому обратиться при поступлении запроса на изменение. Источники 61 требований устанавливают на основе связей или определяют для этой цели атрибут требования. Присвоение уникальных идентификаторов всем требованиям. Выработайте соглашение о присвоении уникальных идентификаторов требованиям, зафиксированным в спецификации требований к ПО. Соглашение должно быть устойчивым к дополнению,

удалению элементов и изменениям, вносимым в требования. Присвоение идентификаторов позволяет отслеживать требования и фиксировать вносимые изменения. Указание атрибутов качества. Выявляя качественные характеристики, удовлетворяющие потребности клиента, не ограничивайтесь только обсуждением функциональности. Выясните ожидаемые производительность, эффективность, надежность, удобство в использовании др. Информация от клиентов об относительной важности тех или иных качественных характеристиках позволит разработчику принять правильные решения, касающиеся архитектуры приложения. Документирование бизнес-правил. К бизнес-правилам относятся корпоративные политики, правительственные распоряжения и алгоритмы вычислений. Ведите список бизнес-правил отдельно от спецификации требований к ПО, поскольку правила обычно существуют вне рамок конкретного проекта. Для выполнения некоторых приходится создавать реализующие их функциональные требования, и поэтом необходимо определить связь между этими требованиями и соответствующими правилами.

Проверка требований

Изучение документов с требованиями. Официальная проверка документирования требований – один из наиболее ценных способов проверки качества ПО. Соберите небольшую команду, члены которой представляют различные направления (например, аналитик, клиент, разработчик и специалист по тестированию}, и тщательно изучите спецификацию требований к ПО, модель анализа и соответствующую информацию на предмет недостатков. Также полезно провести в ходе формулирования требований их неофициальный предварительный просмотр. И хотя реализовать это на практике непросто, данный прием – один из самых ценных, так что начинайте внедрять проверку требований в вашей организации прямо сейчас. Тестирование требований. На основе пользовательских требований создайте сценарии функционального тестирования и задокументируйте ожидаемое поведение продукта в конкретных условиях. Совместно с клиентами изучите сценарии тестирования и убедитесь, что они отражают нужное поведение системы. Проследите связь сценариев тестирования с функциональными требованиями и удостоверьтесь, что ни одно требование не пропущено и что для всех требований есть соответствующие сценарии тестирования. Запустите сценарии, чтобы удостовериться в правильности моделей анализа и прототипов. Определение критериев приемлемости. Предложите пользователям описать, как они собираются определять соответствие продукта их потребностям и его пригодность к работе. Тесты на приемлемость следует основывать на сценариях использования (Hsia, Kung и Sell, 1997). 5.8. Управление требованиями Определение процесса управления изменениями. Определите процесс представления, анализа и утверждения или отклонения изменений. Применяйте его для управления всеми предлагаемыми изменениями. В контексте процесса управления изменениями полезно использовать коммерческие средства отслеживания недостатков. Создание совета по управлению изменениями. Из представителей заинтересованных в проекте лиц организуйте совет по управлению изменениями, который будет получать информацию о предполагаемых изменениях требований, оценивать ее, решать, какие изменения принять, а какие отклонить, и определять, в какой версии продукта будет внедрена та или иная функция. Анализ влияния изменений требований. Анализ влияния изменений помогает совету по управлению изменениями принимать обоснованные решения. Оцените, как каждое предлагаемое изменение требований повлияет на проект. На основе матрицы связей выявите другие требования, элементы архитектуры, исходный код и сценарии тестирования, которые, возможно, придется изменить. Определите, что необходимо для реализации изменений, и оцените затраты на реализацию. Создание базовой версии и управление версиями требований. Базовая версия содержит требования, утвержденные для реализации в конкретной версии продукта. После определения базовых требований

изменения можно вносить только в соответствии с процессом управления изменениями. Присвойте всем версиям спецификации требований уникальные идентификаторы, чтобы избежать путаницы между черновыми вариантами и базовыми версиями, а также между предыдущей и текущей версиями требований. Более надежное решение – управлять версиями документов с требованиями при помощи соответствующих средств управления конфигурацией. Ведение журнала изменений требований. Фиксируйте даты изменения спецификаций требований, сами коррективы, их причины, а также лиц, вносивших изменения. Автоматизировать эти задачи позволяет утилита управления версиями или коммерческая утилита управления требованиями. Контроль за состоянием всех требований. Создайте БД, включающую по одной записи для каждого дискретного функционального требования. Занесите в БД ключевые атрибуты каждого требования, включая его состояние (например, «предложено», «одобрено», «реализовано» или «проверено»), чтобы в любой момент вы могли узнать количество требований в каждом состоянии. Оценка изменяемости требований. Еженедельно фиксируйте количество требований, внесенных в базовую версию, а также число предложенных и одобренных изменений (добавлений, модификаций и удалений). Если требования формируются не самим клиентом, а от его лица, может оказаться, что проблема понята плохо, границы проекта определены нечетко, бизнес стремительно меняется, при сборе информации многие требования были упущены или внутрикорпоративные политики меняются в худшую сторону. Использование средств управления требованиями. Коммерческие утилиты управления требованиями позволяют хранить различные типы требований в БД. Для каждого требования можно определить атрибуты, отслеживать его состояние, а также выявить связи между требованиями и другими рабочими продуктами. Данный прием поможет вам автоматизировать прочие задачи по управлению требованиями, описанные ниже. Создание матрицы связей требований. Создайте таблицу, сопоставляющую все функциональные требования с элементами архитектуры и кода, которые реализуют данное требование, и с тестами, проверяющими его. Матрица связей требований позволяет также сопоставить функциональные требования с требованиями более высоких уровней, на основе которых они созданы, и с другими родственными требованиями. Заполняйте эту таблицу входе, а не в конце работы над проектом.

Управление проектом

Выбор цикла разработки ПО. Вашей компании следует определить несколько жизненных циклов разработки для проектов различного типа и различных степеней неопределенности требований (McCormel, 1996). Каждый менеджер проекта должен выбрать и использовать цикл, оптимальным образом подходящий для его проекта. Включите цикл операции по созданию требований. Если на ранних этапах работы над проектом требования или границы проекта определены нечетко, разрабатывайте продукт постепенно (небольшими этапами), начиная с наиболее понятных требований и устойчивых элементов архитектуры. По возможности реализуйте наборы функций, чтобы периодически выпускать промежуточные версии продукта и как можно раньше предоставлять клиенту работоспособные образцы приложение (Gilb, 1988; Cockburn, 2002). Планы реализации проекта должны быть основаны на требованиях. Разрабатывайте планы и графики работы над проектом постепенно, по мере прояснения границ и подробных требований. Начните с оценки затрат, необходимых на реализацию функциональных требований, определенных на основе первоначальных образа и границ продукт. Графики и оценка затрат, построенные на основе нечетких требований, окажутся крайне неточными, однако по мере детализации требований их следует уточнить. Пересмотр обязательств по проекту при изменении требований. Добавляя в проект новые требования, оцените, удается ли соблюдать обязательства, касающиеся графика и требований к качеству, при доступном объеме ресурсов. Если нет, обсудите реалии

проекта с менеджерами и согласуйте новые, достижимые обязательства (Humphrey, 1997; Fisher, Ury, и Patton, 1991; Wiegers, 2002). Если переговоры не увенчаются успехом, сообщите менеджерам и клиентам о их результатах, чтобы нарушение планов в реализации проекта не стало для них неожиданностью. Документирование и управление рисками, связанными с требованиями. Одна из составляющих управления рисками проекта – выявление и документирование рисков, связанных с требованиями. Уменьшайте или предотвращайте их посредством мозговых штурмов, реализуйте корректирующие действия и отслеживайте их эффективность. Контроль объема работ по созданию требований. Фиксируйте усилия, прилагаемые вашей командой на разработку требований и управление проектом. Эти данные позволят оценить соответствие планам и эффективнее спланировать необходимые ресурсы для будущих проектов. Также отслеживайте, как ваши действия по регламентации требований влияют на проект в целом. Это позволит оценить отдачу от этой работы. Извлечение уроков из полученного опыта. Для этого в организации следует провести ретроспективу проектов, называемую также изучением законченных проектов в области проблем и способов создания требований, накопленным в ходе работы над предыдущими проектами, помогает менеджерам и аналитикам требований более эффективно работать в будущем.

Вопросы и задания для самоконтроля

1. Назовите основные цели, преследуемые при анализе требований в проектах.
2. Перечислите типы требований.
3. Назовите методы выявления требований.
4. Перечислите задачи, которые решаются на стадии анализа требований.
5. Аналитик требований. Перечислите основные задачи аналитика.

Классификация прикладного программного обеспечения

К этим относятся программные комплексы, обеспечивающие выполнение различных прикладных задач, т. е. выполнение фактических задач пользователей. Множество таких программ и комплексов огромно и исчерпывающей классификации не поддается. Среди таких комплексов можно выделить несколько часто используемых видов:

а) Офисные пакеты. Комплексы программ, решающих основные задачи делопроизводства: подготовку документов, выполнение подсчетов, презентации, ведение переписки и организацию работы и др.

б) Системы управления базами данных (СУБД), справочные системы и оболочки автоматизированных информационных систем. Эти программы позволяют организовать ввод, хранение и работу с большими объемами специализированных данных. СУБД часто являются общими компонентами, обеспечивающими работу большого количества специализированных комплексов.

в) Программы обработки графической информации. Крупный класс программ, целью применения которых является формирование какого-либо изображения. Среди них можно упомянуть программы обработки фотоизображений, издательские комплексы, системы подготовки реалистичных трехмерных изображений и многие другие.

Итак, рассмотрим Производителей Прикладного ПО. В данное время есть огромнейшее количество различных программ организовывающих работу в этой сфере. Рассмотрим несколько пакетов предлогаемых нам разными фирмами.

Самая известная организация и я думаю самая популярная – это Microsoft. Думаю любой, даже самый не уверенный пользователь ПК, знает об этой марке. Офисный пакет Microsoft office включает в себя целый ряд программ различных категорий, которые дают обширные возможности рядовому пользователю. В их число входят :

- Microsoft Office Excel
- Microsoft Office Outlook
- Microsoft Office PowerPoint

• Microsoft Office Word
• Microsoft Office Access
• Microsoft Office InfoPath
• Microsoft Office Communicator
• Microsoft Office Visio
• Microsoft Office Publisher
• Microsoft Office OneNote
• Microsoft Office Groove
• Microsoft Office SharePoint Designer
• Microsoft Office Picture Manager
• Microsoft Office Document Image Writer
• Microsoft Office Diagnostics

Наиболее популярные утилиты из этого списка:

• Microsoft Office Word
• Microsoft Office Excel
• Microsoft Office Outlook
• Microsoft Office PowerPoint
• Microsoft Office Access
• Microsoft Office Visio
• Microsoft Office Picture Manager

Microsoft Office Word - это текстовый процессор, предназначенный для создания, просмотра и редактирования текстовых документов, с локальным применением простейших форм таблично-матричных алгоритмов. Microsoft Word является наиболее популярным из используемых в данный момент текстовых процессоров, что сделало его бинарный формат документа стандартом де-факто, и многие конкурирующие программы имеют поддержку совместимости с данным форматом. Расширение «.doc» на платформе IBM PC стало синонимом двоичного формата Word 97—2003. Фильтры экспорта и импорта в данный формат присутствуют в большинстве текстовых процессоров.Формат документа разных версий Word меняется, различия бывают довольно тонкими. Форматирование, нормально выглядящее в последней версии, может не отображаться в старых версиях программы, однако есть ограниченная возможность сохранения документа с потерей части форматирования для открытия в старых версиях продукта. Последняя версия MS Word 2007 "использует по умолчанию" формат основанный на XML - Microsoft Office Open XML. Ранее, большая часть информации, нужной для работы с данным форматом, добывалась посредством обратного инжиниринга, поскольку основная её часть отсутствовала в открытом доступе или была доступна лишь ограниченному числу партнеров и контролирующих организаций. Как и прочие приложения из Microsoft Office, Word может расширять свои возможности посредством использования встроенного макроязыка (сначала использовался WordBasic, с версии Word 97 применяется VBA — Visual Basic для приложений).

Microsoft Office Excel —программа для работы с электронными таблицами. Она предоставляет возможности экономико-статистических расчетов, графические инструменты и язык макропрограммирования VBA (Visual Basic для приложений).

Microsoft Office Outlook - персональный информационный менеджер с функциями почтового клиента и Groupware.Помимо функций почтового клиента для работы с электронной почтой, Microsoft Outlook является полноценным органайзером, предоставляющим функции календаря, планировщика задач, записной книжки и менеджера контактов. Кроме того, Outlook позволяет отслеживать работу с документами пакета Microsoft Office для автоматического составления дневника работы. Outlook может использоваться как качестве

отдельного приложения, так и выступать в роли клиента для почтового сервера Microsoft Exchange Server, что предоставляет дополнительные функции для совместной работы пользователей одной организации: общие почтовые ящики, папки задач, календари, конференции, планирование и резервирование времени общих встреч, согласование документов. Microsoft Outlook и Microsoft Exchange Server являются платформой для организации документооборота, так как они обеспечены системой разработки пользовательских плагинов и скриптов, с помощью которых возможно программирование дополнительных функций документооборота (и не только документооборота), не предусмотренных в стандартной поставке.

Microsoft Office PowerPoint - это программа для создания и проведения презентаций. PowerPoint является частью Microsoft Office. Это позволило PowerPoint стать наиболее распространенной во всем мире программой для создания презентаций. Файлы презентаций Power Point часто пересылаются пользователями программы на другие компьютеры, что означает необходимую совместимость с ними программ конкурентов. Однако, поскольку PowerPoint имеет возможность подключения элементов других приложений через OLE, некоторые презентации становятся сильно привязанными к платформе Windows, что делает невозможным открытие данных файлов, например, в версии для Mac OS. Это привело к переходу на открытые стандарты, такие как PDF и OASIS OpenDocument.

Microsoft Office Access - реляционная СУБД корпорации Microsoft. Имеет широкий спектр функций, включая связанные запросы, сортировку по разным полям, связь с внешними таблицами и базами данных. Благодаря встроенному языку VBA, в самом Access можно писать приложения, работающие с базами данных.

Основные компоненты MS Access:

-запросов (язык SQL в MS Access не соответствует стандарту ANSI);

Они могут вызывать скрипты на языке VBA, поэтому MS Access позволяет разрабатывать приложения и БД практически «с нуля» или написать оболочку для внешней БД.

MS Access является файл-серверной СУБД и потому применима лишь к маленьким приложениям. Отсутствует ряд механизмов необходимых в многопользовательских БД, таких, например, как транзакции. Опыт показывает, что даже для проектов на 5-20 пользователей предпочтительно использовать клиент-серверные решения.

Microsoft Office Visio - редактор диаграмм и блок-схем для Windows. Использует векторную графику для создания диаграмм.

Выпускается в двух редакциях: Standard и Professional.

Microsoft Office Picture Manager – Графический процессор пакета Майкрософт. Предоставляет довольно гибкие возможности управления, редактирования и совместного использования рисунков. Пользователи могут просматривать все рисунки вне зависимости от того, где они сохраняются. Средство поиска рисунков помогает найти рисунки, выполняя поиск автоматически. Picture Manager также помогает автоматически исправлять рисунки. Если необходима специфическая корректировка изображения, можно выбирать отдельные инструменты редактирования. Кроме того, Picture Manager предлагает мощные средства совместного использования рисунков, позволяющие отправлять рисунки в сообщениях электронной почты или создавать библиотеки рисунков в интрасети организации.

На ряду с Майкрософт на рынке уже довольно таки давно появилась компания Adobe. Так же очень известная марка Производителей Прикладного ПО. Возможности пакета

программ Adobe в некоторых сферах напимер графических даже превосходят вездесущий Майкрософт. Итак рассмотрим программы входящие в стандартный пакет Abode:

• Adobe Acrobat Reader
• Adobe Photoshop
• Adobe Photoshop Lightroom
• Adobe Bridge
• Adobe Flash
• Adobe Device Central
• Adobe Extension Manager
• Adobe Help Viewer
• Adobe Stock Photos
• Adobe InDesign
• Adobe Encore DVD

Как видите Adobe имеет обширный список программ, но основная масса из них это графические процессоры и некие аналоги стандартному проводнику Windows. Однако из этого списка я хотела бы выделить 2 программы о которых знают очень большое количество народа.

• Adobe Acrobat Reader
• Adobe Photoshop

Adobe Acrobat — пакет программ, выпускаемый компанией Adobe Systems для создания и просмотра электронных публикаций в формате PDF. Он был создан в 1984 году. Существует несколько версий пакета, отличающихся возможностями: Adobe Acrobat Standard, Adobe Acrobat Professional, Adobe Acrobat Professional Extended (бывший Adobe Acrobat 3D) и Adobe Acrobat Elements. Для просмотра и печати публикаций (без возможности редактирования) доступен бесплатный Adobe Reader.

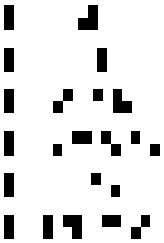
Adobe Photoshop-(Фотошóп) — растровый графический редактор, разработанный и распространяемый фирмой Adobe Systems. Этот продукт является лидером рынка в области коммерческих средств редактирования растровых изображений, и наиболее известным продуктом фирмы Adobe. Часто эту программу называют просто **Photoshop** (**Фотошоп**).

Несмотря на то, что изначально программа была разработана для редактирования изображений для печати на бумаге (прежде всего, для полиграфии), в данное время она широко используется в веб-дизайне. В более ранней версии была включена специальная программа для этих целей — Adobe ImageReady, которая была исключена из версии CS3 за счёт интеграции её функций в сам Photoshop, а также включения в линейку программных продуктов Adobe Fireworks, перешедшего в собственность Adobe после приобретения компании Macromedia.

Photoshop тесно связан с другими программами для обработки медиафайлов, анимации и другого творчества. Совместно с такими программами, как Adobe ImageReady (программа упразднена в версии CS3), Adobe Illustrator, Adobe Premiere, Adobe After Effects и Adobe Encore DVD, он может использоваться для создания профессиональных DVD, обеспечивает средства нелинейного монтажа и создания таких спецэффектов, как фоны, текстуры и т. д. для телевидения, кинематографа и всемирной паутины. Основной формат Photoshop, PSD, может быть экспортирован и импортирован во весь ряд этих программных продуктов. Photoshop CS поддерживает создание меню для DVD. Совместно с Adobe Encore DVD, Photoshop позволяет создавать меню или кнопки DVD. Photoshop CS3 в версии Extended поддерживает также работу с трёхмерными слоями.

Из-за высокой популярности Photoshop, поддержка его формата файлов, PSD, была реализована в его основных конкурентах, таких, как Macromedia Fireworks, Corel PHOTO-PAINT, Pixel image editor, WinImages, GIMP, Jasc Paintshop Pro и т. д.

Photoshop поддерживает следующие цветовые модели:



Photoshop v.10.0, датируемый апрелем 2007 года, имеет название «Photoshop CS3». Аббревиатура «CS3» означает, что продукт интегрирован в третью версию пакета программ «Adobe Creative Suite». В предыдущих продуктах — Photoshop CS и CS 2, с целью отличия от прежних версий и укрепления принадлежности к новой линейке продуктов, был изменён символ программы: вместо изображения глаза, которое присутствовало в версиях с 3-й по 7-ю, в стилевом решении использовалось изображение перьев. В Photoshop CS3 в иконке приложения и экране-заставке используются буквы из названия продукта «Ps» на синем градиентном фоне. Список нововведений включает в себя новый интерфейс, увеличенную скорость работы, новый Adobe Bridge, новые фильтры и инструменты, а также приложение Device Central, позволяющее осуществлять предварительный просмотр работы в шаблонах популярных устройств, например мобильных телефонов.

Последние версии включают в себя *Adobe Camera RAW* — плагин, разработанный Томасом Кноллом, который позволяет читать ряд RAW-форматов различных цифровых камер и импортировать их напрямую в Photoshop.

Теперь я вам хочу рассказать об ещё одном пакете программ широко используемую современными пользователями. Этот пакет менее популярен чем предыдущие, однако он не менее эффективен. Это пакет OpenOffice.org.

OpenOffice.org - — это свободный пакет офисных приложений, разработанный с целью предоставить альтернативу Microsoft Office как на уровне форматов, так и на уровне интерфейса пользователя. Одним из первых стал поддерживать новый открытый формат OpenDocument (ISO/IEC 26300). OOo основан на коде StarOffice, который был приобретён, а затем выпущен с открытым исходным кодом фирмой Sun Microsystems. Ранее он распространялся по схеме двойного лицензирования: по лицензиям LGPL и SISSL. Но 3 сентября 2005 года компания Sun Microsystems объявила об отказе от SISSL для всех своих открытых проектов, и пакет с тех пор имеет только лицензию LGPL. Существует версия пакета OOo для операционных систем семейства Microsoft Windows с возможностью использования без установки, что позволяет запускать пакет, например, с флеш-накопителя.

Офисный пакет OpenOffice.org согласно решениям Правительства РФ передан в 2008 году во все школы России для обучения информатике и компьютерной грамотности в составе базовых пакетов программ лицензионного и открытого программного обеспечения.

Офисный пакет OpenOffice.org может свободно устанавливаться и использоваться в школах, офисах, вузах, домашних компьютерах, государственных, бюджетных и коммерческих организациях и учреждениях России и стран СНГ согласно GNU General Public License.

В основной пакет **OpenOffice.org** входят:

• OOo Writer
• OOo Calc
• OOo Impress
• OOo Base
• OOo Draw
• OOo Math

OOo Writer - текстовый процессор и визуальный (WYSIWYG) редактор HTML.

Writer похож на Microsoft Word и функциональность этих редакторов примерно равна. Writer также имеет некоторые возможности, отсутствующие в Word, например:

[redacted]
Office 2007 в виде плагина, который не входит в стандартную поставку и который необходимо устанавливать отдельно);

[redacted]ёты и другие формулы в таблицах;

[redacted]
таблиц от изменений;

[redacted]
Writer позволяет сохранять документы в различных форматах, включая Microsoft Word, RTF, XHTML и OASIS Open Document Format, который является форматом, используемым по умолчанию начиная с версии OpenOffice.org 2.0, а также в формате предыдущих версий Writer (включая и версию SO Writer 5.2). Writer также позволяет выполнять экспорт в wikimedia.

При этом он позволяет импортировать документы Corel WordPerfect (WDP), 602 Document(.602), WPS Word (WPS), файлов некоторых форматов мобильных текстовых процессоров на платформах PocketPC и Palm и некоторых других.

Список поддерживаемых форматов и качество экспорта/импорта постоянно улучшается **OOo Calc -** табличный процессор. С его помощью можно анализировать вводимые данные, заниматься расчётами, прогнозировать, сводить данные с разных листов и таблиц, строить диаграммы и графики.

Пошаговый ввод формул в ячейки электронных таблиц с помощью Мастера облегчает формирование сложных и вложенных формул, демонстрирует описания каждого параметра и конечный результат на любом этапе ввода.

Условное форматирование и стили ячеек позволяют упорядочить готовые данные, а сводные таблицы и графики показывают итоги работы.

Более двух десятков форматов импорта и экспорта файлов, включая функции импорта текста позволяют оперировать практически любыми данными. Также с помощью специального инструмента можно импортировать данные из других источников, например, баз данных, а можно создать обновляемый диапазон, чтобы импортируемые данные всегда были актуальны.

Поддерживаются связи между разными электронными таблицами и совместное редактирование данных (начиная с версии OpenOffice.org 3.0).

Доступны разнообразные настройки для печати готовых листов на принтере: масштаб, поля, колонтитулы. А встроенная проверка орфографии, как в текстовом редакторе, позволит улучшить качество готового отчёта.

OOo Impress - программа подготовки презентаций. Способна создавать PDF файлы из презентаций, а кроме того, экспортировать их в формат Adobe Flash (SWF), что даёт возможность просматривать их на любом компьютере с инсталлированным Flash-проигрывателем. Может показывать, редактировать и сохранять файлы в нескольких форматах, включая формат .ppt, который используется в Microsoft PowerPoint.

В программе явно недостаточно готовых презентаций, однако многочисленные шаблоны, сделанные пользователями, доступны в Интернете.[2]

Пользователи OpenOffice.org Impress могут установить Open Clip Art Library, которая содержит большую галерею изображений для использования в презентациях и рисунках. Дистрибутивы GNU/Linux Debian, Gentoo и Ubuntu содержат пакет *openclipart* готовый для скачивания и инсталляции с их онлайновых репозитариев программного обеспечения.

OOo Base - механизм подключения к внешним СУБД, плюс встроенная СУБД HSQLDB. Начиная с версии, 3.0 — поддерживает формат базы данных Microsoft Access. **OOo Draw -** векторный графический редактор, по функциональности сравнимый с CorelDRAW. Пакет включает полнофункциональные «коннекторы» между фигурами, которые могут

использовать разнообразные стили линий и позволяют рисовать чертежи, например блок-схемы.

Пользователи OOo также могут инсталлировать Open Clip Art Library, которая содержит огромную галерею флагов, логотипов, иконок, и баннеров для использования в презентациях и рисунках.

OOo Math - редактор формул, входит в состав OpenOffice.org. Созданные формулы можно встраивать в другие документы OpenOffice.org, например в текстовые документы программы Writer. OOo Math поддерживает несколько шрифтов и может экспортировать формулы в формат PDF, а также поддерживает формат MathML.

Другие проекты

Существуют сторонние проекты, позволяющие использовать дополнительные функции в OpenOffice.org и упростить работу с ним. Они включают документацию, интернационализацию, локализацию и API.

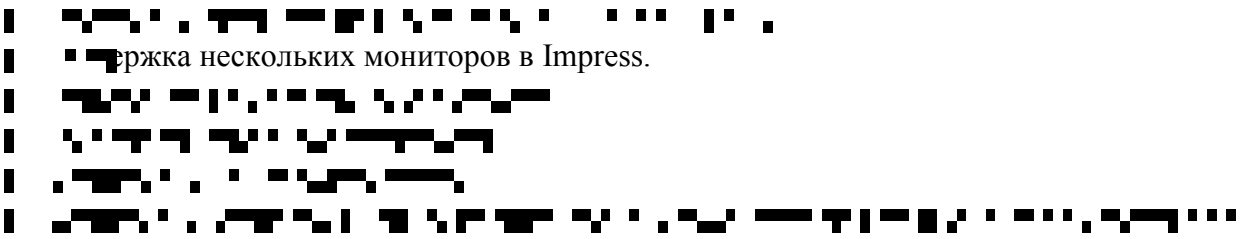
OpenGroupware.org — это набор программ расширения для обмена документами OpenOffice.org, календарями, записными книжками, работы с электронной почтой, мгновенными сообщениями и предоставления общего доступа к различным приложениям коллективной работы.

OOExtras представляет собой попытку систематизации создания и обмена шаблонами документов и другими полезными дополнениями.[10]

Через CPAN доступен набор расширений Perl, позволяющий обрабатывать документы OpenOffice.org внешними программами.[11] Эти библиотеки не используют API OpenOffice.org. Они взаимодействуют с файлами OpenOffice.org напрямую, используя языковые механизмы Perl компрессии/декомпрессии файлов, XML доступ и модули кодирования UTF-8. «Portable OpenOffice.org» — версия OpenOffice.org, предназначенная для работы с USB диска.[12]

OOo4Kids — предназначен для облегчения использования пакета в образовательных целях[13].

OxygenOffice Professional — расширенная версия OpenOffice.org, обеспечивающая: Возможность запуска скриптов на Visual Basic for Applications (VBA) в Calc.



ержка нескольких мониторов в Impress.

начинающих пользователей.

лее 90 свободных шрифтов.

Начиная с версии 2.0.4, OpenOffice.org поддерживает расширения подобно Mozilla Firefox. Компания Ulteo бесплатно предлагает доступ онлайн ко всем приложениям OpenOffice.org Online с любого компьютера без необходимости устанавливать ПО.

Ну в заключении хочу сказать, что это далеко не полны перечень программ используемых на современных ПК. Это одни из самых. Хотя сказать что один лучше другого просто нельзя. Так как каждому пользователю нужна разная функциональность и разные возможности. Так что ключ к оптимизации ПО лежит в компоновке разных программ.

Ка́чество програ́ммного обеспечения — способность программного продукта при заданных условиях удовлетворять установленным или предполагаемым потребностям (ISO).

Другие определения из стандартов:

[illegible] программ, который относится к их способности удовлетворять установленным или предполагаемым потребностям (ГОСТ ИСО);

[illegible] ожиданиям заказчика или пользователя.

Ранние подходы к определения

Том Демарко предлагал при оценке качества программного обеспечения учитывать, что «качество программного продукта является показателем того, насколько он меняет мир к лучшему.

Джеральд Вайнберг в своей работе *Quality Software Management: Volume 1, Systems Thinking* давал определение качества как «значимого для какого-либо человека, подчеркивая тем самым, что понятие качества является по своей природе субъективным — разные люди будут оценивать качество одного и того же программного обеспечения по-разному. Одной из сильных сторон этого определения являются вопросы, на которые должны ответить команды разработчиков программного обеспечения, такие как «Кто те люди, которые будут оценивать наше программное обеспечение?» и «Что будет ценным для них?». Стандарт ISO/IEC определяет *модель качества продукта*, которая включает восемь характеристик верхнего уровня:

[illegible]

[illegible]ность;

[illegible]

В этом стандарте модель качества продукта (англ. *software product quality model*) рассматривается отдельно от субъективного *качества в использовании*, которое может сильно отличаться для различных стейкхолдеров[9]. Модель качества в использовании (англ. *quality in use model*) включает следующие характеристики верхнего уровня:

[illegible]

Роберт Гласс в известной книге «Факты и заблуждения профессионального программирования» утверждает, что большинство профессиональных разработчиков согласны с выделением семи показателей качества как основных[10]:

[illegible]

[illegible]бство использования (юзабилити);

[illegible]

Среди относительно новых моделей качества программного обеспечения можно упомянуть SQUALE и Quamoco[11], которые были применены в промышленных условиях, но пока не получили широкого распространения.

Том Демарко в 1999 году предлагал при оценке качества программного обеспечения учитывать, что «качество программного продукта является показателем того, насколько он меняет мир к лучшему».

Джеральд Вайнберг в своей работе 1992 года *Quality Software Management: Volume 1, Systems Thinking* давал определение качества как «значимого для какого-либо человека»[6][7], подчеркивая тем самым, что понятие качества является по своей природе субъективным — разные люди будут оценивать качество одного и того же программного обеспечения по-разному. Одной из сильных сторон этого определения являются вопросы, на которые должны ответить команды разработчиков программного обеспечения, такие как «Кто те люди, которые будут оценивать наше программное обеспечение?» и «Что будет ценным для них?».

Стандарт ISO/IEC 25010:2021 (ГОСТ Р ИСО/МЭК 25010-2015)[8] определяет *модель качества продукта*, которая включает восемь характеристик верхнего уровня:

тельности;

В этом стандарте модель качества продукта (англ. *software product quality model*) рассматривается отдельно от субъективного *качества в использовании*, которое может сильно отличаться для различных стейкхолдеров[9]. Модель качества в использовании (англ. *quality in use model*) включает следующие характеристики верхнего уровня[8]:

одительность;

Роберт Гласс в известной книге «Факты и заблуждения профессионального программирования» утверждает, что большинство профессиональных разработчиков согласны с выделением семи показателей качества как основных[10]:

Среди относительно новых моделей качества программного обеспечения можно упомянуть SQUALE и Quamoco[11], которые были применены в промышленных условиях, но пока не получили широкого распространения.

Модели качества

Стандарт ISO/IEC 25010:2021 (ГОСТ Р ИСО/МЭК 25010) определяет *модель качества продукта*, которая включает восемь характеристик верхнего уровня:

вания (юзабилити);

В этом стандарте модель качества продукта (англ. *software product quality model*) рассматривается отдельно от субъективного *качества в использовании*, которое может сильно отличаться для различных стейкхолдеров[9]. Модель качества в использовании (англ. *quality in use model*) включает следующие характеристики верхнего уровня:

ска;

Роберт Гласс в известной книге «Факты и заблуждения профессионального программирования» утверждает, что большинство профессиональных разработчиков согласны с выделением семи показателей качества как основных:

Среди относительно новых моделей качества программного обеспечения можно упомянуть SQUALE и Quamoco[11], которые были применены в промышленных условиях, но пока не получили широкого распространения.

Структурное программирование

На всех этапах подготовки к алгоритмизации задачи широко используется структурное представление алгоритма.

Структурное программирование воплощает принципы системного подхода в процессе создания и эксплуатации программного обеспечения ЭВМ. В основу структурного программирования положены следующие достаточно простые положения:

1. алгоритм и программа должны составляться поэтапно (по шагам).
2. сложная задача должна разбиваться на достаточно простые части, каждая из которых имеет один вход и один выход.

3. логика алгоритма и программы должна опираться на минимальное число достаточно простых базовых управляющих структур.

Структурное программирование иногда называют еще "программированием без GO TO". Рекомендуется избегать употребления оператора перехода всюду, где это возможно, но чтобы это не приводило к слишком громоздким структурированным программам.
К полезным случаям использования оператора перехода можно отнести выход из цикла или процедуры по особому условию, "досрочно" прекращающего работу данного цикла или данной процедуры, т.е. завершающего работу некоторой структурной единицы (обобщенного оператора) и тем самым лишь локально нарушающего структурированность программы.
Фундаментом структурного программирования является *теорема о структурировании*. Эта теорема устанавливает, что, как бы сложна ни была задача, схема соответствующей программы всегда может быть представлена с использованием ограниченного числа элементарных управляющих структур. Базовыми элементарными структурами являются структуры: следование, ветвление и повторение (цикл), любой алгоритм может быть реализован в виде композиции этих трех конструкций.
Первая (*а*) структура - тип последовательность (или просто последовательность), вторая (*б*) – структура выбора (ветвление), третья (*в*) – структура цикла с предусловием. При словесной записи алгоритма указанные структуры имеют соответственно следующий смысл:
«выполнить ; выполнить »,
если , то выполнить , иначе выполнить »,
«до тех пор, пока , выполнять »,
где - условие; - действия.
Применительно к языку Паскаль, в котором наиболее полно нашли свое отражение идеи структурного программирования, целесообразно при проектировании алгоритмов дополнительно использовать еще четыре элементарные структуры: сокращенную запись разветвления (рис. 16,); структуру варианта (рис. 16,); структуру повторения или цикла с параметром (рис. 16,); структуру цикла с постусловием (рис. 6,). Каждая из этих структур имеет один вход и один выход.
Ветвящимся (разветвляющимся) называется вычислительный процесс, в котором происходит выбор одного из возможных вариантов вычислений в зависимости от проверки заданных условий.
В зависимости от типа и числа проверяемых условий различают:
- ветвление с простым условием (условие - выражение отношения);
- ветвление с составным условием (условие - логическое выражение);
- сложное ветвление (несколько условий).
Вариант вычислений, определяемый в результате проверки условия, называется *ветвью*.
Циклическим называется процесс многократного повторения некоторого участка вычислений при изменении хотя бы одной из входящих в него величин. Повторяющийся участок вычисления называется *циклом*. Операции, осуществляемые в цикле, составляют *тело цикла.*
Величина, изменяющая своё значение от цикла к циклу, называется *параметром цикла.* Зависимость, связывающая текущее и предыдущее значения параметра цикла, определяет *закон изменения параметра цикла.* Зависимость, предписывающая повторение цикла, либо выход из него, называется *условием повторения цикла.*
Полный однократный проход цикла от начала до конца называется *итерацией.* Все циклические процессы по признаку определения количества повторений (М) разделяются на два класса.

Арифметическим называется циклический процесс, число повторений в котором может быть определено заранее, т.е. не зависит от результатов счёта в теле цикла. *Итерационным* является циклический процесс, число повторений в котором зависит от результатов вычислений в теле цикла и не может быть определено заранее.

Независимо от того, к какому классу относится вычислительный процесс, каждый из них содержит обязательные элементы:

[redacted] его значения функций, формирования нового значения параметра цикла, а также вспомогательные операции);

[redacted] прекращение).

По своему содержанию эти элементы зависят от класса и особенностей цикла, в котором используются.

В соответствии с видом задания (изменения) параметра цикла арифметические циклы подразделяются на:

[redacted]

Выполнение арифметических циклов, т.е. многократное вычисление значений функции при изменяющихся значениях аргумента, называется *табуляцией функции*.

Распространены **две методики (стратегии) разработки программ**, относящиеся к структурному программированию:

– программирование «сверху вниз»;

– программирование «снизу вверх».

Программирование «сверху вниз», или нисходящее программирование – это методика разработки программ, при которой разработка начинается с определения целей решения проблемы, после чего идет последовательная детализация, заканчивающаяся детальной программой.

Такой подход удобен тем, что позволяет человеку постоянно мыслить на предметном уровне, не опускаясь до конкретных операторов и переменных. Кроме того, появляется возможность некоторые подпрограммы не реализовывать сразу, а временно откладывать, пока не будут закончены другие части.

Программирование «снизу вверх», или восходящее программирование – это методика разработки программ, начинающаяся с разработки подпрограмм (процедур, функций), в то время когда проработка общей схемы не закончилась.

Такая методика является менее предпочтительной по сравнению с нисходящим программированием, так как часто приводит к нежелательным результатам, переделкам и увеличению времени разработки.

Подпрограммы бывают двух видов – процедуры и функции. Процедура просто выполняет группу операторов, а функция вдобавок вычисляет некоторое значение и передает его обратно в главную программу. Это значение имеет определенный тип. Данные передаются подпрограмме в виде параметров или аргументов, которые обычно описываются в ее заголовке так же, как переменные. Подпрограммы вызываются, как правило, путем простой записи их названия с нужными параметрами.

Подпрограммы могут быть вложенными – допускается вызов подпрограммы не только из главной программ, но и из любых других программ.

В некоторых языках программирования допускается вызов подпрограммы из себя самой. Такой прием называется рекурсией и опасен тем, что может привести к зацикливанию – бесконечному самовызову. **Достоинства структурного программирования:**

– повышается надежность программ (благодаря хорошему структурированию при проектировании, программа легко поддается тестированию и не создает проблем при отладке);

– повышается эффективность программ (структурирование программы позволяет легко находить и корректировать ошибки, а отдельные подпрограммы можно переделывать (модифицировать) независимо от других);
– уменьшается время и стоимость программной разработки;
– улучшается читабельность программ.

Предпрограммная подготовка задачи

На ЭВМ могут решаться задачи различного характера, например: научно-инженерные; разработки системного программного обеспечения; обучения; управления производственными процессами и т. д. В процессе подготовки и решения на ЭВМ научно-инженерных задач можно выделить следующие этапы:
1. постановка задачи;
2. формирование математической модели задачи;
3. выбор и обоснование метода решения;
4. алгоритмизация вычислительного процесса;
5. программирование;
6. отладка и тестирование программы;
7. решение задачи на ЭВМ и анализ результатов;
8. сопровождение программы.

В задачах другого класса некоторые этапы могут отсутствовать, например, в задачах разработки системного программного обеспечения отсутствует математическое описание. Перечисленные этапы связаны друг с другом. Для уменьшения числа подобных изменений необходимо на каждом этапе по возможности учитывать требования, предъявляемые последующими этапами. Постановка задачи - этап словесной формулировки, определяющий цель решения, исходные данные, основные закономерности, условия и ограничения применения этих закономерностей. Анализируются характер и сущность всех величин, используемых в задаче, и определяются условия, при которых она решается.

Постановка задачи должна отвечать следующим требованиям:
- четкая формулировка цели с указанием вида и характеристик конечных результатов;
- представление значений и размерностей исходных данных;
- определение всех возможных вариантов решения, условий выбора каждого;
- обозначения границы применимости и действия в случае выхода за них.

Формирование математической модели задачи - этап перевода словесной постановки задачи в совокупность математических зависимостей, описывающих исходные данные и вычисления промежуточных и конечных результатов.

Математическая модель формируется с определенной точностью, допущениями и ограничениями. При этом в зависимости от специфики решаемой задачи могут быть использованы различные разделы математики и других дисциплин.

Математическая модель должна удовлетворять по крайней мере двум требованиям: реалистичности и реализуемости. Под реалистичностью понимается правильное отражение моделью наиболее существенных черт исследуемого явления. Реализуемость достигается разумной абстракцией, отвлечением от второстепенных деталей, чтобы свести задачу к проблеме с известным решением. Условием реализуемости является возможность практического выполнения необходимых вычислений за отведенное время при доступных затратах требуемых ресурсов.

Полученная математическая модель должна отвечать следующим требованиям:
- вначале составляется модель исходных данных, затем - расчетные зависимости;
- в модели исходных данных не изменяются размерности данных и не используются никакие математические операции;
- обозначение всех входящих в зависимости величин именами, определяющими их суть;

- указание размерностей всех используемых величин для контроля и дальнейшей модернизации решения;

Выбор и обоснование метода решения - этап разработки или выбора из уже имеющихся метода решения, в том числе выбор стандартных структур вычислительных процессов (линейной, ветвящейся, циклической). Критерии выбора определяются математической моделью решения (предыдущий этап), требованиями к универсальности метода и точности результата, ограничениями технического и программного обеспечении. При обосновании выбора метода необходимо учитывать различные факторы и условия, точность вычислений, время решения задачи на ЭВМ, требуемый объем памяти и другие. Здесь следует указать альтернативные методы и аргументы сделанного выбора.

Алгоритмизация вычислительного процесса - этап разработки совокупности предписаний, однозначно определяющих последовательность преобразования исходных данных в конечные результаты. На данном этапе составляется алгоритм решения задачи согласно действиям, задаваемым выбранным методом решения. Процесс обработки данных разбивается на отдельные относительно самостоятельные блоки, и устанавливается последовательность выполнения блоков. Разрабатывается блок-схема алгоритма.

Программирование. При составлении программы алгоритм решения задачи переводится на конкретный язык программирования. Для программирования обычно используются языки высокого уровня, поэтому составленная программа требует перевода ее на машинный язык ЭВМ. После такого перевода выполняется уже соответствующая машинная программа.

1. Программа должна быть универсальной, то есть не зависящей от конкретного набора данных. Например, если количество обрабатываемых данных может меняться, то следует предусмотреть хранение максимально возможного их количества. Универсальная программа должна уметь обрабатывать ошибки, которые могут возникнуть в процессе обработки информации.
2. Вместо констант1 лучше использовать переменные2. Если в программе используются константы, то при их изменении нужно изменять в исходной программе каждый оператор3, содержащий прежнюю константу. Эта процедура отнимает много времени и часто вызывает ошибки. В программе следует предусмотреть контроль вводимых данных (в частности, программа не должна выполняться, если данные выходят за пределы допустимого диапазона).
3. Некоторые простые приемы позволяют повысить эффективность программы (то есть уменьшить количество выполняемых операций и время работы программы). К таким приемам относится:

- использование операции умножения вместо возведения в степень;
- если некоторое арифметическое выражение встречается в вычислениях несколько раз, то его следует вычислить заранее и хранить в памяти ЭВМ, а по мере необходимости использовать;
- при организации циклов в качестве границ индексов4 использовать переменные, а не выражения, которые вычислялись бы при каждом прохождении цикла;
- особое внимание обратить на организацию циклов, убрав из них все повторяющиеся с одинаковыми данными вычисления и выполняя их до входа в цикл.

4. Программа должна содержать комментарии, позволяющие легко проследить за логической взаимосвязью и функциями отдельных ее частей.

При написании программы следует структурировать ее текст так, чтобы она хорошо читалась. В частности, в программе должно быть хорошо видно, где начинается и где заканчивается цикл.

Отладка программы – процесс выявления и исправления синтаксических и логических ошибок в программе. Суть отладки заключается в том, что выбирается некоторый набор исходных данных, называемый тестовым набором (тестом), и задача с этим набором

решается дважды: один раз – исполнением программы, второй раз – каким-либо иным способом, исходя из условия задачи, так сказать, «вручную». При совпадении результатов алгоритм считается верным. В качестве тестового набора можно выбрать любые данные, которые позволяют:
- обеспечить проверку выполнения всех операций алгоритма;
- свести количество вычислений к минимуму.

В ходе синтаксического контроля программы транслятором выявляются конструкции и сочетания символов, недопустимые с точки зрения правил их построения или написания, принятых в данном языке. Сообщения об ошибках ЭВМ выдает программисту, при этом вид и форма выдачи подобных сообщений зависят от вида языка и версии используемого транслятора.

После устранения синтаксических ошибок проверяется логика работы программы в процессе ее выполнения с конкретными исходными данными. Для этого используются специальные методы, например, в программе выбираются контрольные точки, для которых вручную рассчитываются промежуточные результаты. Эти результаты сверяются со значениями, получаемыми ЭВМ в данных точках при выполнении отлаживаемой программы. Кроме того, для поиска ошибок могут быть использованы отладчики, выполняющие специальные действия на этапе отладки, например, удаление, замена или вставка отдельных операторов или целых фрагментов программы, вывод или изменение значений заданных переменных.

Тестирование - это испытание, проверка правильности работы программы в целом, либо её составных частей.

Отладка и тестирование (англ. test - испытание) - это два четко различимых и непохожих друг на друга этапа:
- при отладке происходит локализация и устранение синтаксических ошибок и явных ошибок кодирования;
- в процессе же тестирования проверяется работоспособность программы, не содержащей явных ошибок.

Тестирование устанавливает факт наличия ошибок, а отладка выясняет ее причину. В современных программных системах (Turbo Basic, Turbo Pascal, Turbo C и др.) отладка осуществляется часто с использованием специальных программных средств, называемых отладчиками. Эти средства позволяют исследовать внутреннее поведение программы.

Программа-отладчик обычно обеспечивает следующие возможности:
- пошаговое исполнение программы с остановкой после каждой команды (оператора);
- просмотр текущего значения любой переменной или нахождение значения любого выражения, в том числе, с использованием стандартных функций; при необходимости можно установить новое значение переменной;
- установку в программе "контрольных точек", т.е. точек, в которых программа временно прекращает свое выполнение, так что можно оценить промежуточные результаты, и др.

При отладке программ важно помнить следующее:
- в начале процесса отладки надо использовать простые тестовые данные;
- возникающие затруднения следует четко разделять и устранять строго поочередно; - не нужно считать причиной ошибок машину, так как современные машины и трансляторы обладают чрезвычайно высокой надежностью.

Как бы ни была тщательно отлажена программа, решающим этапом, устанавливающим ее пригодность для работы, является контроль программы по результатам ее выполнения на системе тестов.

Программу условно можно считать правильной, если её запуск для выбранной системы тестовых исходных данных во всех случаях дает правильные результаты.

Для реализации метода тестов должны быть изготовлены или заранее известны эталонные результаты. Вычислять эталонные результаты нужно обязательно до, а не после

получения машинных результатов. В противном случае имеется опасность невольной подгонки вычисляемых значений под желаемые, полученные ранее на машине.
Тестовые данные должны обеспечить проверку всех возможных условий возникновения ошибок:
- должна быть испытана каждая ветвь алгоритма;
- очередной тестовый прогон должен контролировать то, что еще не было проверено на предыдущих прогонах;
- первый тест должен быть максимально прост, чтобы проверить, работает ли программа вообще;
- арифметические операции в тестах должны предельно упрощаться для уменьшения объема вычислений;
- количества элементов последовательностей, точность для итерационных вычислений, количество проходов цикла в тестовых примерах должны задаваться из соображений сокращения объема вычислений;
- минимизация вычислений не должна снижать надежности контроля;
- тестирование должно быть целенаправленным и систематизированным, так как случайный выбор исходных данных привел бы к трудностям в определении ручным способом ожидаемых результатов; кроме того, при случайном выборе тестовых данных могут оказаться непроверенными многие ситуации;
- усложнение тестовых данных должно происходить постепенно.
Процесс тестирования можно разделить на три этапа.
1. Проверка в нормальных условиях. Предполагает тестирование на основе данных, которые характерны для реальных условий функционирования программы.
2. Проверка в экстремальных условиях. Тестовые данные включают граничные значения области изменения входных переменных, которые должны восприниматься программой как правильные данные. Типичными примерами таких значений являются очень маленькие или очень большие числа и отсутствие данных. Еще один тип экстремальных условий - это граничные объемы данных, когда массивы состоят из слишком малого или слишком большого числа элементов.
3. Проверка в исключительных ситуациях. Проводится с использованием данных, значения которых лежат за пределами допустимой области изменений. Известно, что все программы разрабатываются в расчете на обработку какого-то ограниченного набора данных. Поэтому важно получить ответ на следующие вопросы:
Что произойдет, если программе, не рассчитанной на обработку отрицательных и нулевых значений переменных, в результате какой-либо ошибки придется иметь дело как раз с такими данными?
Как будет вести себя программа, работающая с массивами, если количество их элементов превысит величину, указанную в объявлении массива?
Что произойдет, если числа будут слишком малыми или слишком большими?
Наихудшая ситуация складывается тогда, когда программа воспринимает неверные данные как правильные и выдает неверный, но правдоподобный результат.
Программа должна сама отвергать любые данные, которые она не в состоянии обрабатывать правильно.
Решение задачи на ЭВМ и анализ результатов. После отладки программы ее можно использовать для решения прикладной задачи. При этом обычно выполняется многократное решение задачи на ЭВМ для различных наборов исходных данных. Получаемые результаты интерпретируются и анализируются специалистом или пользователем, поставившим задачу.
Сопровождение программы:
Разработанная программа длительного использования устанавливается на ЭВМ, как правило, в виде готовой к выполнению машинной программы. К программе прилагается

документация, включая инструкцию для пользователя. Так как при установке программы на диск для ее последующего использования помимо файлов с исполняемым кодом устанавливаются различные вспомогательные программы (утилиты, справочники, настройщики и т. д.), а также необходимые для работы программ разного рода файлы с текстовой, графической, звуковой и другой информацией.

А также:

- доработка программы для решения конкретных задач;
- составление документации к решенной задаче, математической модели, алгоритму, программе по их использованию.

1. Конста́нта в программировании — способ адресования данных, изменение которых рассматриваемой программой не предполагается или запрещается.

2. Перемéнная в императивном программировании — поименованная, либо адресуемая иным способом область памяти, адрес которой можно использовать для осуществления доступа к данным. Данные, находящиеся в переменной (то есть по данному адресу памяти), называются значением этой переменной.

В других парадигмах программирования, например, в функциональной и логической, понятие переменной оказывается несколько иным. В таких языках переменная определяется как имя, с которым может быть связано значение, или даже как место (location) для хранения значения.

3 Инструкция или оператор (англ. statement) — наименьшая автономная часть языка программирования; команда. Программа обычно представляет собой последовательность инструкций.

Многие языки (например, Си) различают инструкцию и определение. Различие в том, что инструкция исполняет код, а определение создаёт идентификатор (то есть можно рассматривать определение как инструкцию присваивания). Ниже приведены основные общие инструкции императивных языков программирования. Индекс в языках программирования — элемент перечислимого множества, который указывает на конкретный элемент массива, обычно является неотрицательным скалярным целым числом.

Есть три способа, как элементы массива могут быть проиндексированы целыми неотрицательными числами:

0 («индекс с началом с нуля») первый элемент массива имеет индекс 0;

1 («индекс с началом с единицы»)

первый элемент массива имеет индекс 1;

n («индекс началом с n»)

базисный индекс массива может быть свободно выбран. Обычно языки программирования, позволяющие «индекс началом с n», разрешают также в качестве индекса массива выбирать отрицательные значения, а также и другие скалярные типы данных, как перечисления или символы.

Массив может иметь несколько измерений, таким образом обычная практика обращаться к массиву, используя несколько индексов. Например к двумерному массиву с тремя строками и четырьмя столбцами можно было бы обратиться к элементу в 2-ом ряду и 4-ой столбце с помощью выражения: (в языке в котором приоритет у строки) и [3,1] (в языке в котором приоритет у столбца) в случае индексом с началом с нуля. Таким образом два индекса используются для двумерных массивов, три — для трехмерного массива, и n — для n-мерного массива.

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ КЫРГЫЗСКОЙ РЕСПУБЛИКИ

ОШСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ МАТЕМАТИКИ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Кафедра «Информационных технологий и автоматизированных систем»

«Утверждено»
на заседании кафедры ИТАС
прот. № 1 от «28» сентября 2020 г.
Зав. каф. ИТАС, доцент: Молдояров У.

ЛАБОРАТОРНАЯ РАБОТА

Дисциплина: **“Технология разработки программного обеспечения”**

Профиль подготовки**:** ИВТ

Курс: 1

Группа: ИВТ(м)-1-20

Составитель, доцент каф. ИТАС: _____ Беделова Н.С.

2020-2021 учебный год

Лабораторная работа №1.

Предварительное проектирование программного обеспечения

Цель работы:

- Проведение предварительного проектирования конкретной программы.

- Составить перечень требований и функциональных характеристик разрабатываемой программы.

- Разработка документа «Постановки задачи».

Порядок выполнения работы и отчетность.

Во время выполнения лабораторной работы необходимо определить потребность в программном изделии, его назначение и основные функциональные характеристики; составить перечень требований к нему.

Работа должна быть оформлена в виде документа «Постановка задачи».

Теоретические сведения.

Определение полного комплекса требований к программному изделию является первоначальной задачей его разработки. Некачественное определение требований приводит к созданию программного изделия, которое будет правильно решать неверно сформулированную задачу, а программный продукт не будет соответствовать истинным потребностям заказчика.

Поэтому при определении требований к программному изделию требуется соблюдать максимально возможную аккуратность и точность, чтобы затем эти требования можно было транслировать в разрабатываемый проект с минимальным числом ошибок. Требования задаются на естественном языке и должны быть очень точно сформулированы.

Требования оформляются в виде документа, в котором письменно излагается то, что будет, и что не будет сделано при выпуске программного изделия. В учебном заведении такой документ называется " Постановка задачи ".

Постановка задачи пишется на естественном языке в терминах понятных и пользователю и разработчику программного обеспечения и может содержать следующие разделы:

1. **Заголовок к программе.**

2. **Условие задачи.**

Формулируется условие задачи, краткое описание разрабатываемой программы, ее назначение и необходимые уточнения.

3. Начало/окончание работы.

Указывается месяц и год начала/окончания разработки программы.

4. Основание для разработки программы.

Основанием для разработки программы может быть заказ пользователя, задание администрации учебного заведения, контракт учебного заведения с другой организацией и пр.

5. Краткая характеристика объекта разработки.

Описывается объект разработки: как решается поставленная задача в настоящее время без разрабатываемой программы и какая часть ручной работы будет заменена программой.

6. Пользователь.

Указываются пользователи программы.

7. Цель и назначение разработки.

8. Основные требования.

Описываются требования пользователя к разрабатываемой программе.

Здесь же с точки зрения пользователя следует подробно перечислить функции программы.

9. Входная информация.

Перечисляются все входные данные программы с точки зрения их содержания и назначения - отчеты, файлы, записи, поля данных, таблицы... Их возможные носители и средства отображения информации и т.д.

10. Выходная информация.

Описываются выходные данные так же, как в пункте 9.

11. Требования к аппаратному и программному обеспечению.

Описывается конфигурация аппаратуры и программного обеспечения, в которых разрабатываемая программа может работать, другие программные продукты, от которых она зависит.

12. Внешние ограничения.

13. Эффективность.

Цели производительности, такие, как временные и объемные характеристики, пропускная способность, использование ресурсов и пр.

14. Безопасность данных от несанкционированного доступа.

15. Эргономические характеристики.

Эргономическими характеристиками изделия являются такие свойства, которые обеспечивают надежность, комфорт и продуктивность работы пользователей и операторов. Эргономика (греч.) - труд + закон - отрасль знания, изучающая трудовые процессы с целью создания наилучших условий труда.

16. Мобильность.

Описываются требования и цели обеспечения переноса программного продукта из одних рабочих условий в другие.

17. Окупаемость капиталовложений.

Определяется прибыль, которую даст создание программного продукта в понятиях, соответствующих целевому назначению организации.

18. Другие соглашения сторон.

19. Терминология.

Четко определяется вся терминология, которая может оказаться специфической для данной разработки.

- Контрольные вопросы

1. Предмет "Технология программирования", основные понятия.

2. Предварительное проектирование (Системный анализ).

3. Постановка целей проекта и продукта.

4. Определение требований.

5. Документ "Постановка задачи". ("Соглашение о требованиях")

Лабораторная работа №2.

Разработка программного обеспечения

Цель работы:

- Определение этапов разработки конкретной программы.
- Разработка календарного плана создания конкретной программы.

Порядок выполнения работы и отчетность.

Во время выполнения лабораторной работы необходимо подробно проанализировать этапы разработки конкретной программы (ее жизненный цикл), начиная от возникновения потребности в ней до полного прекращения ее использования вследствие ее морального старения или потери необходимости решения соответствующих задач.

Работа должна быть оформлена в виде календарного плана разработки программы по форме:

№	**Наименование этапа разработки программы**	**Срок исполнения**		Примечания
		Начало	**Окончание**	

Теоретические сведения.

Обобщенная модель жизненного цикла программы может выглядеть так:

1. Системный анализ (предварительное проектирование ПИ)

а) исследование

б) осуществимость

- эксплуатационная
- экономическая
- коммерческая

2. Проектирование программы

а) конструирование программы

- функциональная декомпозиция задачи
- разработка архитектуры системы
- внешнее проектирование программы
- разработка архитектуры программы
- проектирование базы данных

б) программирование

- внутреннее проектирование форм и модулей
- определение свойств объектов и кодирование
- отладка форм и модулей
- компоновка форм и модулей в программу

г) отладка программы в целом

3. Оценка (испытания) программы

4. Использование программного изделия

- **Контрольные вопросы**
-
- Разработка программного обеспечения.
- 1. Технология разработки программного обеспечения.
- 2. Требования, предъявляемые к «идеальной» технологии разработки программного обеспечения..
- 3. Объектно - ориентированные технологии разработки ПО.
- 4. Программное обеспечение как изделие.
- 5. Проектирование ПО. Обзор этапов проектирования ПО (жизненный цикл)

Лабораторная работа №3, 4.

Построение функциональной схемы системы ПО

Цель работы:

- проведение функциональной декомпозиции решаемой задачи;

- построение функциональной схемы;

Порядок выполнения работы и отчетность.

Во время выполнения лабораторной работы необходимо провести функциональную декомпозицию решаемой задачи, построить соответствующую схему.

Работа должна быть оформлена в виде спецификации, содержащей функциональную схему решаемой задачи.

Теоретические сведения.

Проектирование программного обеспечения часто начинается с функциональной декомпозиции решаемой задачи.

Функциональная декомпозиция задачи представляет собой иерархическое разбиение сложной задачи на ряд проще решаемых небольших задач, которые, в свою очередь, разделяются на подзадачи до тех пор, пока каждая необходимая деталь в ней не будет определена достаточно ясно.

Концепция иерархической декомпозиции настолько естественна, что мы не всегда в состоянии осознать, как часто нам приходится использовать ее на практике. Она вытекает из человеческой потребности иметь дело с поддающимся управлению вполне определенным числом дискретных источников информации и производить «отсечение» информации до тех пор, пока число дискретных источников не станет приблизительно равно семи.

Строгая иерархическая декомпозиция подчиняется правилам:

1. На каждом уровне иерархии задача должна иметь законченный вид на данном уровне детализации;

2. На любом уровне иерархии каждое разбиение полностью охватывает отдельную задачу (функцию), соответствующую данному уровню детализации.

Контрольные вопросы

Методы проектирования программного обеспечения.

2. Восходящее и нисходящее проектирование программного обеспечения.

3. Объектно – ориентированное проектирование программного обеспечения.

4. Функциональная декомпозиция системы программного обеспечения.

5. Надежность программного обеспечения.

Лабораторная работа №5, 6.

Внешнее проектирование программного обеспечения

Цель работы:

- проведение внешнего проектирования конкретной программы;

- разработка взаимодействия разрабатываемой программы с пользователем: сценарий, экранные формы, набор подсказок, и пр.

Порядок выполнения работы и отчетность.

Во время выполнения лабораторной работы необходимо описать ожидаемое поведение разрабатываемой программы с точки зрения внешнего по отношению к нему наблюдателя (обычно - пользователя), то есть осуществить "конструирование" внешних взаимодействий будущей программы продукта с пользователем без конкретизации его внутреннего устройства.

Работа должна быть оформлена в виде внешней спецификации.

Теоретические сведения.

Внешнее проектирование мало, чем связано (если связано вообще) с программированием; более непосредственно оно касается понимания обстановки, проблем и нужд пользователя, психологии общения человека с машиной. Эта сторона внешнего проектирования становится все более значительной по мере того, как применение ЭВМ все больше начинает затрагивать пользователей, незнакомых с программированием.

Результаты внешнего проектирования программы отражаются во внешней спецификации, в которой может быть представлено описание следующих внешних аспектов программы:

- организация диалога программы с пользователем;

- состав меню, подменю ...;

- описание действий функциональных клавиш;

- все экранные формы или протокольные экранные сообщения;

- сообщения, выдаваемые пользователю во время проведения сеанса работы программы и выдаваемые пользователем на них ответы;

- сообщения об ошибках;
- подсказки пользователю, организация "помощи";
- структура и организация баз данных;
- описание и подготовка входных данных;
- выходные печатные формы;
- другие внешние сопряжения программы.

Внешняя спецификация должна быть написана на понятном пользователю и разработчику языке для уменьшения вероятности возможных недоразумений. Причем, проверку корректности и полноты спецификации необходимо проводить еще до начала программирования.

Основные правила организации диалога программы с пользователем.

1. Согласовывайте способ взаимодействия программы с пользователем, с его подготовкой и уровнем, с ограничениями, в условиях которых он работает.

2. Выходные данные должны выдаваться программой в требуемой форме и обязательно с комментариями. Нельзя, например, выдавать их в виде числа, а тем более - в виде набора чисел.

3. Обеспечьте концептуальную целостность для разных типов вводимых / выводимых сообщений. Например, все сообщения выдачи на экран, отчеты должны иметь одинаковые форматы, стиль и сокращения.

4. Старайтесь, чтобы пользователь вводил данные с клавиатуры как можно меньше. Будет лучше, если ему будет дана возможность выбора вводимых данных в виде меню, что исключит ошибки ввода пользователем. Сообщения, вводимые пользователем, должны быть как можно короче, но не настолько, чтобы исчезла их осмысленность.

5. Обеспечьте средства "помощи" - специальный набор функций (подсказки) по оказанию пользователю помощи, если тот запутался или забудет какое-либо правило взаимодействия.

6. Старайтесь, чтобы программа не рассердила пользователя. Избегайте оскорбительных сообщений. Общайтесь с пользователем на его языке, а не на тарабарском жаргоне программистов.

7. Помните о дизайне экрана. С эстетично оформленным экраном приятней работать. Экранная форма может быть разнообразной.

8. Старайтесь на каждое входное сообщение выдавать какое-либо уведомление. Программа должна принимать любые вводимые данные. Если данные не являются тем, что программа считает допустимым, то она должна информировать об этом пользователя.

9. Спроектируйте программу так, чтобы пользователь в любой момент работы с ней мог закончить эту работу или перейти в предыдущее состояние. Предполагается, что в первом случае программа успешно завершит свою работу (закроет открытые файлы, очистит переменные памяти и т.д.)

10. Ошибки пользователя должны обнаруживаться немедленно.

11. Не стремитесь исправлять входное сообщение пользователя.

Например, в медицинской информационной системе пользователь случайно нажимает на лишнюю клавишу, вследствие чего входное сообщение принимает вид "Рэтиловый спирт" вместо сообщения "Этиловый спирт". Система исправляет это сообщение на "Метиловый спирт". Известно, что этиловый спирт опьяняет, а метиловый спирт убивает.

12. Любые действия пользователя, как правильные, так и неправильные, должны контролироваться программой. В качестве отрицательного примера можно привести программу, которая может вдруг аварийно, преждевременно закончить свою работу.

- Контрольные вопросы

Внешнее проектирование ПО.

1. Методическая, технологическая, инструментальная и организационная поддержка процесса проектирования ПО.

2. Конструирование программного обеспечения.

3. Внешнее проектирование ПО.

4. Правила организации диалога ПО с пользователем.

5. Создание интерфейса приложения.

Лабораторная работа №7, 8.

Разработка архитектуры программного обеспечения

Цель работы:

- разработать архитектуру программного изделия.

Порядок выполнения работы и отчетность.

Во время выполнения лабораторной работы необходимо разработать архитектуру разрабатываемой программы: спроектировать структуру всех его компонент, его объектно - модульно - иерархическое построение.

Работа должна быть оформлена в виде спецификации, содержащей архитектуру разрабатываемой программы.

Теоретические сведения.

Типовая архитектура программы может иметь вид:

1. Уровень -центральный диспетчер

2. Уровень - местные диспетчеры

3. Уровень - функциональные программы

4. Уровень – функциональные программы

5. Уровень - стандартные программы, библиотеки программ

Общие правила структурного построения программных модулей.

1. Каждый модуль характеризуется функциональной законченностью, автономностью и независимостью в оформлении от модулей, которые его используют и которые он вызывает. Высокую степень независимости модулей можно достичь с помощью двух методов оптимизации:

- усилением внутренних связей в каждом модуле, т.е. реализовать отдельные функции отдельными модулями (высокая прочность модуля).

- ослаблением взаимосвязи между модулями, применяя формальный механизм передачи параметров (слабое сцепление модулей).

2. Применяются стандартные правила организации связей по управлению и информации с другими модулями (смотри далее).

3. Комплексы программ разрабатываются в виде совокупности небольших по количеству (до 100) программных модулей, связанных иерархическим образом, что дает возможность полностью и относительно просто уяснить функцию и правила работы отдельных частей и комплекса программ в целом.

4. Как правило, модуль содержит от 10 до 1000 выполняемых операторов языка высокого уровня. Размеры модуля влияют на степень независимости программы, легкость ее чтения и тестирования.

5. Модуль прочный. Прочность модуля измеряется его внутренними связями. Модуль - это замкнутая программа, которая выполняет одну или несколько функций, обладает некоторой логикой.

6. Модуль предсказуемый, т.е. модуль, работа которого не зависит от пред-ыстории его использования. Модули не должны сохранять никаких "воспоминаний" о предыдущем вызове.

7. Определена структура принятия решений. Желательно, чтобы те модули, на которые прямо влияет принятое решение, были подчиненными (вызываемыми) по отношению к принимающему решение модулю.

8. Объем данных, на которые модуль может ссылаться, должен быть сведен к минимуму.

9. Внутренняя процедура (или подпрограмма) - это замкнутая программа, физически расположенная в вызывающем ее модуле. Их следует избегать, т.к. их трудно изолировать для автономного тестирования и они не могут быть вызваны из модулей, отличных от тех, которые их физически содержат. Когда возникает потребность во внутренней процедуре, проектировщик должен рассмотреть возможность оформления ее в виде отдельного модуля.

10. В параметры процедуры следует включать только те переменные, через которые идет обмен информацией с другими программными единицами. Другие переменные - это внутреннее дело процедуры. Процедуры, которые выдают в

качестве результата только одно значение, оформляются как функции. Функция удобнее в использовании, так как ее результат непосредственно можно использовать в арифметическом и/или в логическом выражениях.

Правила связи программных модулей по управлению.

1. Передача управления вызываемому модулю всегда осуществляется через его начало, т.е. через первый оператор.

2. Выход из вызываемого модуля всегда происходит через его естественное окончание, т.е. после нормального его завершения.

3. По окончании исполнения вызываемого модуля управление передается в вызывающий модуль на оператор, следующий непосредственно за оператором вызова.

4. Модули низших уровней или одного уровня иерархии могут вызываться для исполнения только модулями высших уровней, т.е. модули низших уровней не могут вызывать модули высших уровней, а модули одного уровня - вызывать друг друга.

5. Если все же необходимо исполнить модуль с некоторой внутренней точки, то вызов все равно осуществляется стандартным образом (через его первый оператор), а точка начала задается в виде параметра. При этом в начале вызываемого модуля должен стоять переключатель, который обеспечивает передачу управления программой к его внутренним точкам по параметру, указанному при обращении к модулю.

6. В каждом модуле должна быть предусмотрена возможность подключения контрольных и отладочных средств; операторы, реализующие эти средства, обычно сосредотачиваются в конце модуля.

Правила связи программных модулей по информации.

1. Информация зон глобальных переменных доступна для использования любым модулям, входящим в комплекс программ или в группу программ в соответствии с областью действия зоны глобальных переменных, т.е. глобальные переменные, могут быть доступны не для всего комплекса программ, а лишь для указанной в описании группы модулей.

2. Локальные переменные доступны лишь в пределах того модуля, в котором они определены или объявлены.

3. Для взаимодействия вызываемых и вызывающих модулей создаются зоны обменных переменных, информация из которых доступна лишь модулям, непосредственно связанным по управлению.

4. После окончания работы вызываемого модуля считается, что соответствующие регистры не содержат информации, являющейся результатом его работы. Запрещается их использовать в вызывающем модуле.

5. Информация, находящаяся в регистрах вызывающего модуля, при вызове должна быть сохранена на период выполнения вызываемого модуля и восстановлена при возврате управления в вызывающий модуль. Сохранение регистров может осуществлять как вызывающий, так и вызываемый модуль.

- **Контрольные вопросы**

Конструирование архитектуры ПО.

2. Типовая архитектура программного обеспечения.

3. Общие правила структурного построения ПО.

4. Правила связи программных модулей по управлению.

5. Правила связи программных модулей по информации.

Лабораторная работа №9, 10.

Описание алгоритма

Цель работы:

- разработка алгоритма решения задачи;

- запись алгоритма в блок схемной форме;

Порядок выполнения работы и отчетность.

Во время проведения лабораторной работы необходимо разработать алгоритм решения конкретной задачи и представить его в блок-схемной форме

Работа должна быть оформлена в виде спецификации, содержащей описание алгоритма конкретной программы.

Теоретические сведения.

Представленный в любой форме алгоритм пишется на естественном (разговорном) языке, используя термины отрасли, в которой решается задача и не должен содержать программистские термины; тем более алгоритм не может содержать операторы языка программирования. Текст алгоритма должен быть читаемым без дополнительных пояснений автора.

В алгоритме не показывают описания, т.к. описания - это особенности языка программирования, а не алгоритма. Алгоритм не может быть "привязан" к конкретному языку программирования. В алгоритме описываются конкретные действия - "что делается", а не "как это делается".

В блок-схеме обычно пишут "скупые" тексты. Поэтому ее дополняют пояснениями. В пояснении к схеме описывают функциональные действия всего алгоритма в целом, действия отдельных частей схемы и в случае крайней необходимости - действие отдельного блока. Описание каждого блока схемы бессмысленно. Пояснения должны содержать некоторую дополнительную информацию, а не перефразировку алгоритма.

Основные требования к блок-схеме.

- схема выполняется с соблюдением условий ГОСТа

- запись в блоках должна быть словесной или математической, а не в виде операторов языка программирования

- управление по схеме должно в основном идти вниз (вправо), возвращаясь назад только в циклах.

- альтернативно выполняемые ветви должны размещаться параллельно.

- переменные должны быть определены в каком-либо блоке.

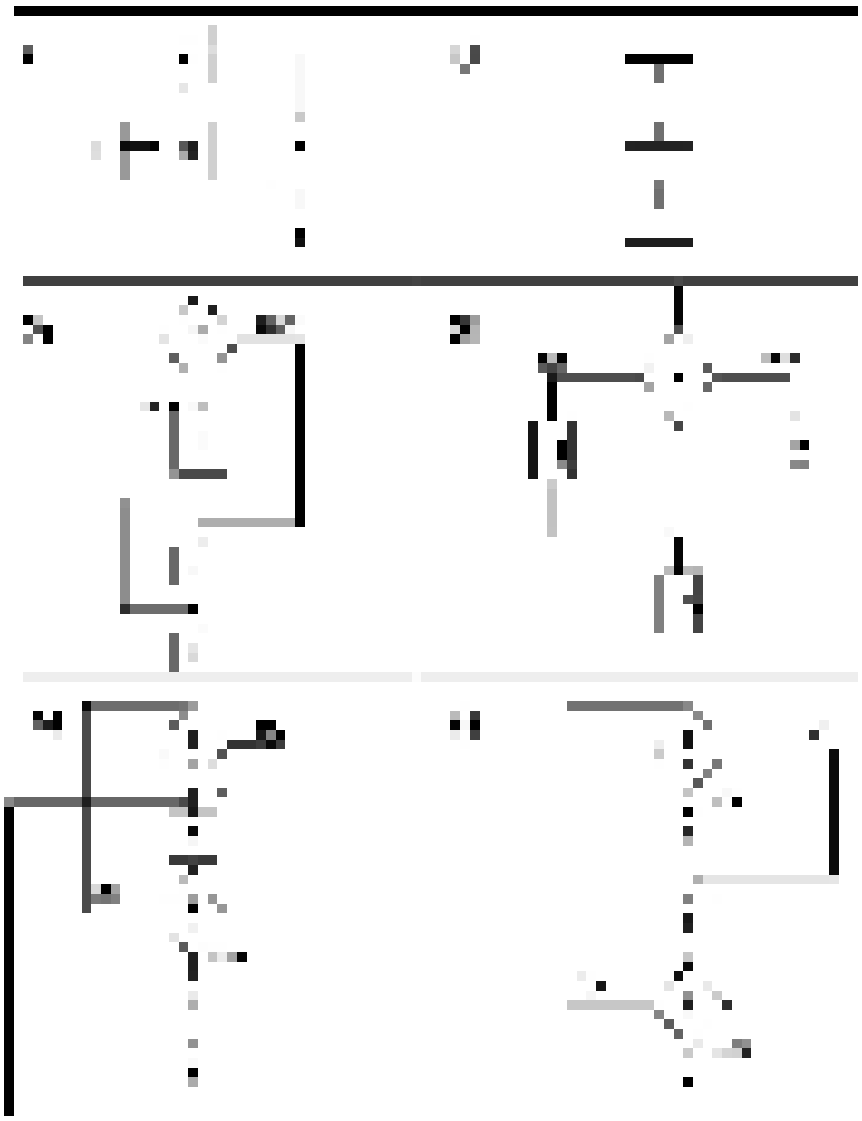
- входные и выходные блоки процедур должны содержать, соответственно, входные и выходные (формальные) параметры.

- блоки можно объединять в более крупные пунктирными линиями, которые нужно комментировать - описать их назначения.

- на каждую подпрограмму (модуль) составляется отдельная схема.

Примеры неправильно и правильно составленных блок-схем.

Неправильно Правильно



- Варианты задач

Варианты задач

1 Из заданного массива чисел, вывести сначала отрицательные значения, потом ноль, а затем положительные в возрастающем порядке.

2. Сгенерировать следующие комбинации: 1111, 1112, 1113, 1121, 1122, 1123, 1131, 1132, 1133, 1211, ...,3333.

3. Найти корни уравнения $y=x^2-2$ на заданном отрезке методом половинного деления. Нарисовать график функций.

4. Найти наибольший общий делитель заданных N чисел.

5. Вычислить определитель N -го порядка (N<=5), пользуясь формулой разложения определителя по i-й строке и зная формулу вычисления определителя 2-го порядка.

6. На местности имеется N населенных пунктов, пронумерованных от 1 до N (N<=10). Некоторые из пунктов соединены между собой дорогами. Информация о дорогах задается в виде последовательности пар чисел i, j (i<j), указывающих, что i-йи j-й пункты соединены дорогой, признак конца этой последовательности - пара нулей. Определить, можно ли попасть по этим дорогам из первого пункта в п -ый.

7. Реализовать алгоритм так называемой «быстрой сортировки» Хоара: имеются два указателя i и j, причем вначале i=l, а j=N (номеру последнего элемента).

Сравним a[i] и a[j] , и если обмен не требуется, то уменьшим j на 1 и повторим этот процесс.

После первого обмена увеличим i на 1 и будем продолжать сравнения, увеличивая i , пока не произойдет еще один обмен.

Тогда снова уменьшим j и т.д., то есть будем «сжигать свечку с обоих концов», пока не станет i=j. В результате получим, что слева от a[i] оказались только меньшие элементы, а справа - только большие (тем самым элемент a[i] окажется на своем окончательном месте).

Продолжить вышеописанный метод для левой и правой частей массива до тех пор, пока в подмассиве не останется только один элемент.

8. Найти наименьшее общее кратное заданных N чисел.

9. В заданной строке проверить чередование букв и цифр. Вывести строку, состоящую из единиц и нулей (1 - буква, 0 - цифра).

10. Сформировать последовательность строк:

A

BB

CCC

DDDD

. . . . . .

ZZZZZZZ . . . . . 26 раз

11. Сформировать все перестановки без повторений некоторого массива значений. Например если задан массив, содержащий символы ABC, то необходимо сформировать следующие комбинации: ABC, ACB, BAC, BCA, CAB, CBA.

12. Вычислить функцию Аккермана (m и n – натуральные числа):

$$A(n,m) = \begin{cases} m+1 & \text{, если } n=0 \\ A(n-1,1) & \text{, если } n<>0 \text{ и } m=0 \\ A(n-1,A(n,m-1)) & \text{, если } n<>0 \text{ и } m>0 \end{cases}$$

13. Имеются сведения о росте N учеников одного класса (данные вводятся в алфавитном порядке фамилий учеников). Определить средний рост учеников, чьи фамилии расположены в журнале между фамилиями учеников с минимальным и максимальным ростами (рост этих учеников тоже учитывать), считая, что самый высокий и самый низкий ученики в классе единственные.

14. Кот Матроскин и Шарик загадывали четные и нечетные числа в произвольном порядке, пока не кончилось место на печке, где они записывали эти числа. Определить, каких чисел было загадано больше: четных или нечетных, если последним было записано число 0 (при подсчете это число не учитывать). Сколько четных чисел предшествовало первому нечетному?

15. В коробке перемешались кубики. На всех гранях каждого кубика нарисованы одинаковые буквы или цифры. Нужно разложить их в разные коробки: кубики с цифрами поместить в одну, а с буквами - в другую. Вывести на экран содержимое всех трех коробок (что вначале лежало в первой коробке, затем распечатать содержимое полученных двух коробок).

16. Наглядно продемонстрировать графики функций: y=a sin(bx+c)+d, меняя шаг аргумента и границы отрезка. Провести исследование функции на четность, убывание и возрастания. Найти корни и экстремальные точки.

17. Сформировать всевозможные варианты расстановки m (m>3) ферзей на шахматной доске m x m клеток, при которых ферзи не бьют друг друга.

18. В арифметическом выражении 1*2*3*4*5 вместо звездочек расставить арифметические операции + , - , * , / так, чтобы получилось число 543.

19. Фунтик и Хрюша решили сравнить свои книжки «по толщине». Они составили список, указывая вместе с названиями число страниц в каждой книге. Попутно выяснилось, что все книжки имели различное количество страниц, некоторые книги были как в библиотеке Фунтика, так и в библиотеке у Хрюши. Помогите Фунтику найти самую «толстую» его книжку среди тех, которых нет у Хрюши. (Книжки сравнивать по числу страниц.).

20. У двух спортсменов были гири для тренировок, на каждой был указан ее вес. Каждый раз после очередного занятия они составляли их по возрастанию весов гирь. Потом решили объединить свои гири, сохраняя привычное расположение гирь. Требуется помощь в их расстановке (известен вес каждой гири в исходных наборах). Напечатать порядок гирь в исходных наборах и в полученном.

21. Чтобы закрепить понятие «чет» и «нечет», мальчик записывал на бумаге различные целые числа, а затем четные раскрашивал красным карандашом, нечетные - синим. После раскрашивания он разрезал полоску бумаги на отдельные числа и решил составить узор: выстроить их в линию в таком порядке: красный квадратик, синий квадратик, красный, синий и т.д. Оставшиеся числа положил в конец полученного узора. Составить программу отображения полученной последовательности чисел на экране в соответствующем цвете.

22. Даны действительные числа a(1)...a(50). Они определяют интервалы на числовой оси [a(1), a(2)], . . . [a(49), a(50)]. Определить, имеют ли все интервалы общие точки? Если да, то указать какую-нибудь из этих точек.

23. Участники игры «О! Счастливчик» рассаживаются на пронумерованных стульях вокруг большого круглого стола. Посредством константы счета начинается их отсчет по часовой стрелке. Игрок, на которого попадает константа счета, обязан освободить место. Отсчет игроков продолжается до тех пор, пока не останутся два человека. При известном числе игроков N и константы счета C определить номера стульев, которые нужно занять игрокам, чтобы попасть в число двух «счастливчиков».

24. Необходимо среди введенных N целых чисел распознать степени двойки (например, 1, 2, 4, 8, 16 и т.д.).

25. На пронумерованных дискетах записано различное количество файлов, и только две дискеты имеют одинаковое число файлов. Найти их номера.

26. Каждый ученик 8 класса нашел на археологических раскопках по одной глиняной табличке с зашифрованными на них числами (на каждой табличке по два числа.). Когда удалось расшифровать эти числа, то оказалось, что на некоторых табличках сумма двух чисел является простым числом, а на остальных - составным. Определить, сколько табличек было с составными числами.

27. Имеется массив N натуральных чисел. Определить, сколько в нем чисел:

а) различных;

б) различных четных;

в) различных простых.

28. Таблица круга футбольного чемпионата задана квадратной матрицей размера N x N. Выигрыш - 2 очка, ничья - 1 очко, проигрыш - 0 очков.

Напечатать номер команды, занявшей первое место, номера команд, которые прошли чемпионат без поражений, подсчитать количество очков у каждой команды, определить номер команды -аутсайдера (при равном количестве очков у двух команда преимущество определяется из результата игры между этими командами).

29. Три мушкетера получили сведения, что 12 украденных алмазных подвесок хранятся в десяти различных замках Англии, причем, в каждом замке их по 12 штук (часть из них подлинные, а недостающие заменены фальшивыми). Известно, что у подлинных подвесок по 13 алмазов, у поддельных же различное количество камней, но не 13. Помоги мушкетерам найти все 12 подлинных алмазных подвесок королевы.

30. В зрительном зале N рядов по M мест в каждом. Информация о проданных билетах хранится в двух матрицах: в первой матрице в соответствующей ячейке указана цена билета за данное места, а во второй - продан билет на это место или нет (билет продан - «1», не продан - «0»). Подсчитать общую выручку за проданные билеты.

31. На мячиках были написаны различные буквы и цифры. Необходимо отобрать те мячики, которые встречаются по одному разу. Полученный набор напечатать.

32. Из элементов заданного массива создать новый массив, отобрав в него те числа из исходного массива, которые являются степенью заданного числа R. Числа в новом массиве не должны повторяться.

33. Напечатать таблицу умножения в системах счисления:

а) десятичной;

б)восьмеричной;

в) шестнадцатеричной.

34. В квадратной целочисленной матрице порядка N найти одинаковые элементы, которые встречаются одновременно как выше, так и ниже главной диагонали (саму главную диагональ не рассматривать).

34. Пусть заданы N окружностей на плоскости координатами их центра X,Y ирадиусом R. Имеются ли среди данных окружностей две попарно пересекающиеся? Если да, то напечатать их параметры (номера, координаты и радиусы).

35. Строка символов представляет собой запись арифметического выражения. Преобразовать подряд стоящие цифры в натуральные числа и произвести необходимые вычисления:

а) выполнить действия по порядку;

б) действия +,-,*,/ выполняются соответственно приоритету этих операций;

в) в записи может присутствовать одна пара или даже несколько пар круглых скобок, что меняет порядок выполнения операций.

36. Дано N строк русского текста. Составить программу зашифровки, а затем ее расшифровки, пользуясь таким простым правилом : каждая буква заменяется на следующую по алфавиту (при этом буква «я» заменяется на букву «а»).

Усложненный вариант: буква заменяется на другую, которая следует за ней через N позиций (замена также циклическая).

37. На метеорологической станции в компьютер введены сведения о среднесуточной температуре за март месяц. Составить программу, которая бы определяла:

а) количество дней, когда температура была ниже 0°C ;

б) сумму положительных температур;

в) среднюю температуру месяца;

г) день, когда температура ближе всего подходило к среднемесячной.

38. Рассчитать таблицу значений и построить график функции одного аргумента y=f(x) на заданном отрезке по введенной формуле, меняя шаг аргумента и границы отрезка.

39. В литерном файле lit.txt найти символ, повторяющийся максимальное число раз. Напечатать этот символ и количество его повторений.

40. Вычислить заданный интеграл методами Трапеций и Симпсона. В обоих случаях вывести также количество разбиений отрезка интегрирования для достижения одной и той же точности вычисления интеграла.

41. Наглядно продемонстрировать графики функций: y=ax2+bx+c, меняя шаг аргумента и границы отрезка. Провести исследование функции на четность, убывание и возрастания. Найти корни и экстремальные точки.

42. Маленький Мук пробовал печатать в текстовом редакторе. Определить, сколько различных символов он напечатал на экране, а также литеру, напечатанную максимальное число раз. Вывести на экран полученный результат.

43. Пусть заданы N окружностей на плоскости координатами их центра X,Y ирадиусом R. Найти номера всех уединенных окружностей, то есть таких, которые не имеют общих точек ни с одной из остальных окружностей, не лежат целиком внутри и не заключают внутри себя какой - либо из остальных окружностей.

44. Дана строка текста из букв и цифр. Не пользуясь стандартной процедурой языка Паскаль для перевода цифр в число, организовать свою процедуру перевода подряд стоящих цифр в целые числа и найти их среднее арифметическое. Напечатать полученные числа и результат.

45. В заколдованном замке в парадном зале пол выложен мозаичными мраморными плитами, имеющими форму неправильных восьмиугольников, всего в зале было 57 таких плит. Если наступить на плитку с наименьшей площадью, то двери зала откроются. Каждая плита задана координатами своих вершин. Найти номер этой потайной плиты.

46. В трех таблицах записаны оценки за семестр в трех восьмых классах A,B и C. Количество учеников в классах Ma, Mb, Mc (строк) различно, количество предметов - N (столбцов) одинаково. Напечатать ту таблицу, в которой наибольшее количество строк со всеми пятерками (то есть больше отличников) и, если таких таблиц несколько, напечатать их все.

48. «Военкомат». По имеющемуся списку призывников в файле prizyv.txt (строка содержит фамилию призывника, затем через запятую его заболевания) и списку заболеваний в файле bolezny.txt, по причине которых могут освободить от срочной службы в армии, вывести список новобранцев.

49. В стене имеется прямоугольное отверстие размерами X и Y. Пролезет ли в данную дыру кирпич, с размерами A, B и C, если проталкивать его таким образом, чтобы стороны кирпича были параллельны сторонам отверстия?

50. Дама сдавала в багаж коробку и пустой чемодан с соответствующими размерами X, Y, Z и A, B, C. Удастся ли сэкономить место в багажном отделении, поместив данную коробку в чемодан, если стороны коробки и чемодана будут параллельны?

51. Одна дама решила купить новый мягкий уголок (диван, 2 кресла, журнальный столик). Известна стоимость каждого изделия в отдельности. Хватит ли у нее денег на покупку всего набора, если нет, то что сможет она купить из предложенного (вывести все варианты набора).

52. Белоснежка на Рождество решила купить гномам новые рубашки. Она определила размеры каждого: R1, R2, R3, R4, R5, R6, R7. В магазине ей предложили рубашки трех размеров: I - 25 - 35 см, II — 36 — 45 см и III - 46 -55 см. Сколько Белоснежка купит рубашек I -го, II -го и III - го размеров?

53. Координаты любого поля шахматной доски могут быть заданы парой натуральных чисел, не превосходящих восьми. Дано четыре числа (K, L и M, N), задающие координаты двух клеток (при счете слева направо и снизу вверх). Определить:

а) являются ли эти два поля шахматной доски полями одного цвета?

б) если на первом поле расположен ферзь, то угрожает ли он фигуре противника, расположенной на втором поле?

в) если на первом поле расположен конь, то угрожает ли он фигуре противника, расположенной на втором поле?

54. По данному трехзначному числу N найти и напечатать все новые трехзначные числа, построенные из тех же цифр, а также выделить из них те числа, которые оказались больше исходного.

55. Вычислить корни заданной функции на заданном отрезке методом хорд. Нарисовать график функций.

56. За весь учебный год учительница поставила в восьмом классе К двоек, L троек, M четверок и N пятерок. Какой получился средний балл за каждый триместр, если половина двоек и троек были выставлены в первом триместре, остальная половина - поровну во втором и третьем? Четверки и пятерки распределились так: в III - м триместре - половина, ав1-м и во П-м триместрах - поровну.

57. В массиве содержатся сведения о росте “n” учеников одного класса (данные вводятся в алфавитном порядке). Определить средний рост учеников, чьи фамилии расположены между фамилиями учеников с минимальным и максимальным ростом (их рост также учитывается).

58. Сформировать матрицу m x n элементов, заполнив ее целыми числами от -9 до 9 случайным образом. Отсортировать элементы каждой строки с первого до предпоследнего по возрастанию. Последний элемент каждой строки заменить суммой положительных элементов этой строки.

59. Заполнить матрицу n x n случайными элементами и выполнить следующие действия:

1. Найти сумму элементов матрицы по его периметру.

2. Найти сумму положительных элементов.

3. Найти среднее арифметическое элементов случайностей строки.

60. Имеются m различных предметов, их вес и стоимость. Определить, какие предметы можно положить в рюкзак емкостью до 50 кг так, чтобы общая стоимость предметов была минимальной.

Контрольные вопросы

Описание алгоритма.

1. Программирование. Проектирование программного модуля.
2. Планирование программирования.
3. Алгоритм. Формы описания алгоритмов.
4. Основные требования к блок-схеме.
5. Проектирование логики программного модуля

Лабораторная работа №11, 12.

Пошаговая разработка программы

Цель работы:

- Освоить метод пошаговой разработки программ.

Порядок выполнения работы и отчетность.

Во время выполнения лабораторной работы необходимо осуществить пошаговую разработку конкретной программы.

Работа должна быть оформлена в виде спецификации, содержащей описание пошаговой разработки конкретной программы.

Теоретические сведения.

Пошаговая разработка (пошаговая детализация) программы представляет собой простой процесс, предполагающий первоначальное выражение логики модуля в терминах гипотетического (условного) языка очень высокого уровня с последующей детализацией каждого предложения в терминах языка более низкого уровня, до тех пор, пока, наконец, не будет, достигнут уровень используемого языка программирования.

Достоинство пошаговой детализации состоит в том, что этот метод позволяет проектировщику упорядочить свои рассуждения - на каждом шаге детализации решается элементарная задача.

Пример метода пошаговой детализации.

ЗАДАЧА. Дана матрица размером 10*10 элементов. Для каждого столбца среди элементов, лежащих выше первого нулевого, и значения которых лежат в интервале [c,d], найти наименьший и наибольший элементы и их номера в строке. Если нулевого элемента в столбце нет, то обрабатывается весь столбец.

План решения задачи .

1. Входные / выходные переменные.

2. Основной алгоритм – перебор столбцов исходной матрицы (цикл по столбцам)

3. Перебор элементов столбца матрицы (внутренний цикл по строкам)

4. Обработка элементов матрицы

5. Поиск наибольшего и наименьшего элементов в столбце.

6. Обработка начальных и конечных операторов циклов

7. Оптимизация и шлифовка программы

Входные / выходные переменные.

A(10,10) - исходная матрица

C и D - границы интервала

max(10) - массивы, содержащие наибольшие и наименьшие значения

min(10) каждого столбца исходной матрицы.

Imax(10) - массивы, содержащие номера, строк в которых встречаются

Imin(10) найденные наибольшие и наименьшие значения в столбце.

а) **Первый шаг.**

Детализация ввода-вывода.

```
PROGRAMM PRIMER;

VAR

A: ARRAY[1..10,1..10] OF REAL;        (*  Исходнаяматрица       *)

C,D: REAL;                            (*  Границы интервала     *)
```

```
I,J: INTEGER;                                (*  Номера строк и столбцов *)
MAX,MIN: ARRAY[1..10] OF REAL;   (*  Значения наибольших     *)
                                             (*  и наименьших элементов *)
IMAX,IMIN: ARRAY[1..10] OF INTEGER; (*  и их номера строк *)
```

Другие переменные

```
BEGIN
WRITELN ('Введите элементы  матрицы');
FOR I:=1 TO 10 DO BEGIN
      FOR  J:=1 TO 10  DO  READ (A[I,J]);       WRITELN;
END;
WRITE('Введите  границы  интервала');          READ (C,D);
```

Основной алгоритм

```
FOR  I:=1 TO 10  DO WRITELN ('MIN =', MIN[I],  ', егономер', IMIN[I],
 '; MAX=', MAX[I], ', егономер', IMAX[I]);
END.
```

б) **Второй шаг.**

Детализация основного алгоритма.

Необходимо выполнить одно и тоже для каждого столбца.

Начальные операторы

```
FOR J:=1 TO 10  DO
BEGIN
```

Обработка столбца

```
END;
```

Конечные операторы

в) **Третий шаг.**

Обработка столбца исходной матрицы.

В этом столбце необходимо обработать элементы, лежащие выше первого нулевого элемента матрицы A.

Начальные
операторы

FOR I:=1 TO 10 DO WHILE A[I,J]#0 DO

BEGIN

Обработка элементов матрицы

END;

Конечные операторы

г) **Четвертый шаг.**

Определение принадлежности элемента матрицы A заданному отрезку CD.

IF (C<A[I,J]) AND (A[I,J]<D) THEN

BEGIN

Поиск наибольшего элемента
в столбце

Поиск наименьшего элемента
в столбце

END;

д) **Пятый шаг.**

Вычисление наибольшего и наименьшего элемента в столбце матрицы A и их номера в строке.

IF A[I,J]>=MAXT THEN BEGIN MAXT:=A[I,J]; IMAXT:=I; END;

IF A[I,J]<=MINT THEN BEGIN MINT := A[I,J]; IMINT := I; END;

Появились новые переменные: MAXT и MINT, IMAXT и IMINT, которые необходимо включить в описания переменных с присвоением типа REAL и INTEGER соответственно.

е) **Шестой шаг.**

Детализация начальных и конечных операторов циклов.

Движемся по программе изнутри циклов наружу и смотрим, что необходимо для работы циклов.

Чтобы сработал внутренний цикл, надо чтобы на первом шаге MAXT и MINT имели какие-то значения и что делать в случае отсутствия в столбце элементов из интервала (c,d)?

Примем, что индексы в этом случае равны нулю, а значения наибольших и наименьших элементов, попадающие в выходные массивы, несущественны.

Итак, вначале MAXT=C, MINT=D, IMAXT=0, IMINT=0.

Результаты этого цикла надо заслать в J-е элементы результирующих массивов, поэтому конечные операторы будут такими:

MAX[J]:=MAXT, MIN[J]:=MINT

IMAX[J]:=IMAXT , IMIN[J]:=IMINT

Для цикла по столбцам никаких начальных и конечных операторов не требуется.

ж) **Последний шаг.**

Шлифование и оптимизация программы.

Получив программу, можно заняться ее улучшением, чтобы она стала короче или выполнялась быстрее.

Например, если использовать MAX(J), MIN(J), IMAX(J), IMIN(J) вместо MAXT, MINT, IMAXT, IMINT, то строка конечных операторов не потребуется, но программа будет дольше выполняться (хотя ее текст сократится), т.к. участится выполнение операции "Обращение к элементу массива" (довольно длинная, с точки зрения ЭВМ, операция).

Можно выполнить другие оптимизационные действия.

Результирующая программа

```
PROGRAMM PRIMER;

VAR

A: ARRAY[1..10, 1..10] OF REAL;          (* Исходная матрица   *)

I, J: INTEGER                    (* Их номера строк и столбцов *)

C,D: REAL;                              (* Границы интервала *)
```

```
MAX,MIN: ARRAY[1..10] OF REAL;    (* Значения наибольших и *)
                                  (*  наименьших элементов *)
IMAX,IMIN: ARRAY[1..10] OF INTEGER;   (*  и их номера строк *)
MAXT,MINT: REAL;       (* Временные переменные, наибольший и *)
                       (* наименьший значения элементов *)
IMAXT,IMINT: INTEGER;                  (* и их номера в столбце *)
BEGIN
WRITELN ('Введите  элементы  матрицы:');
FOR I:=1 TO 10  DO
BEGIN
FOR J:=1 TO 10  DO  READ (A[I,J]);            WRITELN;
END;
WRITE('Введите границы интервала');              READLN (C,D);
FOR J:=1 TO 10  DO              (* обработка столбцов матрицы A *)
BEGIN
     IMAXT:=0;       IMINT:=0;        MINT:=C;        MINT:=D;
(*  обработка элементов столбца матрицы A  *)
FOR  I:=1   TO   10   DO     WHILE   A[I,J]#0     DO
BEGIN
IF (C<=A[I,J]) AND (A[I,J]<=D) THEN   (* элементматрицы *)
BEGIN                          (* принадлежит отрезку CD? *)
IF  A[I,J]>=MAXT     (* элемент матрицы наибольший? *)
THEN  BEGIN  MAXT:=A[I,J];  IMAXT:=I; END;
IF   [I,J]<=MINT       (* элемент матрицы наименьший? *)
THEN  BEGIN  MINT:=A[I,J];  IMINT:=I;   END;
```

```
END;
END;
MAX[J]:= MAXT;              MIN[J]:= MINT;
IMAX[J]:=IMAXT;             IMIN[J]:=IMINT;
END;
FOR I:=1 TO 10 DO WRITELN ('MIN=',MIN[I],',егономер', IMIN[I],
  '; MAX=',MAX[I],',егономер',IMAX[I]);
END.
```

- Контрольные вопросы

Пошаговая разработка программы.

1. Структурное программирование.
2. Метод пошаговой разработки программ.
3. Оптимизация программ. Основные приемы оптимизации программ.
4. Корректность (правильность) программы.
5. Эффективность программы. Эффективность или удобочитаемость?

Лабораторная работа №13, 14.

Запись текстов программ на алгоритмическом языке высокого уровня

Цель работы:

- Запись текстов программ на алгоритмическом языке высокого уровня, используя правила хорошего стиля программирования.

Порядок выполнения работы и отчетность.

Во время проведения лабораторной работы необходимо написать программу на конкретном алгоритмическом языке программирования для решения конкретной задачи, используя приемы и методы программирования, которые используют, опытные программисты, чтобы получить правильные, надежные, эффективные, удобные для применения и легко читаемые программы.

В отчете должны быть представлены листинги текстов программ.

Теоретические сведения.

Программа должна быть составлена таким образом, чтобы ее могли прочитать в первую очередь люди, а не машины.

Программа - это документ для последующего использования, учебный материал по кодированию алгоритмов и средство для дальнейшей разработки более совершенных программ.

Как правило, к разработке программы приступают со скромными целями. В дальнейшем ее возможности расширяют. Трудно читаемые программы модифицировать сложно. Особенно, если это приходится делать не автору программы.

Программа должна передавать логику и структуру алгоритма настолько, насколько это возможно. Ее кодируют просто и рационально. Следует избегать всевозможных программистских трюков, т.к. чем их больше, тем труднее будет разобраться в логике программы самому автору, а кто-либо другой это сделать не сможет. На ранних этапах разработки сложной программы лучше без колебания переписать заново ее громоздкие блоки, если это ведет к ее упрощению.

Общая организация программы и ее запись

Также как разделение большого произведения на главы и параграфы облегчает чтение, так и разбиение большой программы на параграфы, разделы (подпрограммы и модули), путем выделения логических единиц улучшает восприятие программы; помогает избежать однообразия и хорошо организовать материал. Название раздела отражает его цель.

Структура программы хорошо реализуется с некоторым смещением. Для ЭВМ форма записи программы безразлична: смещение строк предназначено для удобства чтения программы человеком. Смещение строк не должно быть большим: двух-трех позиций вполне достаточно, чтобы выделить структуру программы.

Основные правила записи различных структурных конструкций

на примере алгоритмического языка Pascal:

1. Короткий составной оператор:

```
BEGIN  A1; A2; ...; An  END;
```

2. Длинный составной оператор:

```
BEGIN

 A1; A2; ...; An

END;
```

3. Короткое ветвление и обход:

а) IF e THEN A1

ELSE A2;

б) IFeTHEN A3;

4. Более длинное ветвление:

IF e THEN A1

ELSE A2;

5. Очень длинное ветвление:

IFeTHENBEGIN

A1; A2; ...; An

END

ELSE BEGIN

A1; A2; ...; An

END;

6. Циклыкороткие:

а) WHILE e DO a;

б) REPEAT a UNTIL e;

в) FOR k:=b1 TO b2 DO a;

г) FOR k:=b2 DOWNTO b1 DO a;

7. Циклыдлинные:

а) WHILE e DO BEGIN

A1; A2; ...; An

END;

б) REPEAT

A1; A2; ...; An

UNTIL e;

в) FOR k:=b1 TO b2 DO

BEGIN

A1; A2; ...; An

END;

8. Циклы с общим началом и/или концом:

FOR k:=b1 TO b2 DO WHILE e DO

BEGIN

A1; A2; ...; An

END;

9. Выбор: CASE e OF

N1,N2,...,Nk:: A1;

.....……

Ne,.....,Nv: An1;

END;

10. Длинные операторы. Если оператор не помещается в строку, то его продолжение в следующей строке следует записывать со смещением, причем небезразлично, в каком месте сделан перенос. Например:

WRITELN ('первый корень уравнения равен'

,X1,'второй корень уравнения равен',X2);

читается хуже, чем

WRITELN ('первый корень уравнения равен',X1,

'второй корень уравнения равен',X2);

11. Объединение операторов в строку. Если язык программирования позволяет размещать несколько операторов в одной строке, то в строку следует группировать логически связанные операторы, и так, чтобы его можно было кратко откомментировать.

Вместо

I:=0;

X:=RO*Cos(FI);

Y:=RO*Sin(FI);

можно написать

```
I:=0;

X:=RO*Cos(FI);    Y:=RO*Sin(FI);    (* координатывектора *)
```

Или вместо

```
WRITE('Введите  коэффициенты:');

READLN(A,B,C);
```

можно написать

```
WRITE('Введите коэффициенты:');        READLN (A,B,C);
```

12. Увеличив интервалы между некоторыми операторами (добавив пустые строки), можно дополнительно улучшить восприятие программы.

13. Все метки оператора должны быть вынесены в начало строки и должны выступать из общего текста программы влево и в порядке возрастания, чтобы их можно было легко найти.

14. К началу строк выносятся заголовки блоков, начала крупных циклов и очень крупных составных операторов. Операторы END должны размещаться либо в той же строке, что и соответствующее ему DO или BEGIN либо под парным ему DO или BEGIN.

15. Использование достаточного количества пробелов, отступов, пустых строк в тексте программы является мощным средством, позволяющим сделать программу более выразительной.

Контрольные вопросы

Запись текстов программ на алгоритмическом языке высокого уровня.

1. Стиль программирования.

2. Типовая структура программного модуля.

3. Общая организация программы и ее запись.

4. Читабельность программы. Правила записи структурных конструкций. Комментарии. Блок комментариев. Идентификация имен переменных, функций, процедур, файлов.

5. Малый программистский стандарт. Правила сокращений имен.

Лабораторная работа №15.

Тестирование и отладка разработанной программы

Цель работы:

- Осуществить тестирование и отладку разработанной ранее конкретной программы на алгоритмическом языке высокого уровня.

Порядок выполнения работы и отчетность.

Во время выполнения лабораторной работы необходимо составить набор тестов к разработанной ранее программе и провести ее отладку.

Составленный набор тестов необходимо представить в отчете.

Теоретические сведения.

Тестирование - это процесс выполнения программы с целью определения места некорректного ее функционирования. Оно включает преднамеренное конструирование трудных наборов входных данных, создающих наибольшие возможности для отказа программного изделия. Тестирование является основным методом обнаружения ошибок в программе. Результаты тестирования являются исходными данными для отладки.

Отладка программы - это этап ее разработки, на котором устраняются недостатки только что созданной программы.

Если программа правильно ведет себя для солидного набора тестов, нет оснований, утверждать, что в ней нет ошибок. Просто неизвестно, когда она не работает и можно говорить лишь о некотором уровне ее правильности.

Тесты, не способствующие обнаружению ошибок и только подтверждающие корректность функционирования программы, являются не эффективными, т.к. приводят к бесполезным затратам ресурсов и времени.

Тест - это просчитанный вручную или другим способом пример, промежуточные и конечные результаты которого используются для контроля правильности (живучести) программного изделия. Тест состоит из исходных данных и тех значений, которые должны выдать отладочные печати при работе по этому тесту. Эти значения должны быть записаны в точности в том виде, в котором их должна выдать ЭВМ. Эти значения желательно получить любым путем, но не тем, который реализован в программе, т.к. в последнем случае можно не заметить ошибки в алгоритмизации.

Комплект тестов должен быть таким:

- чтобы проверить все варианты внешнего эффекта программы и варианты ее внутренней работы алгоритма;

- чтобы все ветви алгоритма были пройдены, по крайней мере, по одному разу.

- чтобы проконтролировать предельные и вырожденные случаи.

Тестовые данные должны подбираться таким образом, чтобы программист был в состоянии вычислить правильный результат ещё до начала тестирования.

Процесс тестирования программы можно разделить на три этапа:

1. Проверка в нормальных условиях.

2. Проверка в экстремальных условиях.

3. Проверка в исключительных ситуациях.

Каждый из этих трех этапов проверки должен гарантировать получение верных результатов при правильных входных данных и выдачу сообщений об ошибках при неправильных входных данных.

Проверка в нормальных условиях

Случаи, когда программа должна работать со всеми возможными исходными данными, чрезвычайно редки. Обычно имеют место конкретные ограничения на область изменения данных, в которой программа должна сохранять свою работоспособность. Программа должна выдавать правильные результаты для характерных совокупностей исходных данных.

Проверка в экстремальных условиях

Тестовые данные этого этапа включают граничные значения области изменения входных переменных, которые должны восприниматься программой как правильные данные. Типичные примеры очень большие числа, очень малые числа или отсутствие информации.

Проверка в исключительных ситуациях.

Используются исходные данные, значения которых лежат за пределами допустимой области их изменения. Наихудшая ситуация когда программа воспринимает неверные данные как правильные и выдаёт неверный, но правдоподобный результат. Программа должна отвергать любые данные, которые она не в состоянии обрабатывать верно.

Пример тестов.

Пусть требуется вычислить длину диагонали параллелепипеда по формуле  Необходимо сформировать тестовые данные для нормальных, экстремальных и исключительных условий.

	Стороны Параллепипеда			Примечание	
1	1	1		Хороший нормальный тест d = 1,7320508	
1	2	3		Тест в нормальных условиях d = 3,7416577	
0	0	0		Результат должен быть равен нулю	
0	1	2		Не параллепипед. Что произойдет?	
1	0	3		Неверные данные	
2	1	0		Неверные данные	
1	-6	3		Неверные данные	
A	B	C		Неверные данные	

- Контрольные вопросы

Тестирование и отладка программы.

1. Ошибки - их причины появления и последствия. Классы ошибок

2. Первичные и вторичные ошибки. Синтаксические и семантические ошибки. Защитное программирование. Программирование без ошибок.

3. Тестирование ПО. Понятие теста. Комплект тестов.Пример теста.

4. Методы тестирования. Этапы тестирования. Аксиомы тестирования.

5. Отладка ПО. Этапы, методы и инструменты отладки программ.

Лабораторная работа №16.

Составление документа «Руководство пользователю»

Цель работы:

- Составить документ «Руководство пользователю» к разработанной ранее программы.

Порядок выполнения работы и отчетность.

Во время выполнения лабораторной работы необходимо составить документ «Руководство пользователю» к разработанной ранее программе.

Работа должна быть оформлена в виде документа «Руководство пользователю».

Теоретические сведения.

Формальные требования к документации программного обеспечения описаны в ЕСПД (Единая система программной документации), неформально: состав документации к программному обеспечению состоит из описания его внешнего эффекта и описания его внутреннего устройства.

Первая часть документации, так называемая «Инструкция пользователю» или «Руководство пользователю» предназначена для того, кто собирается использовать программное обеспечение (для пользователя), не вникая в подробности его внутреннего устройства.

Вторая часть - «Руководство программисту» необходима при модификации программного обеспечения или при необходимости исправить в нем ошибку.

В целом, документация к программному обеспечению может содержать ниже перечисленные сведения:

1. Наименование ПО и описание задачи, которую оно решает.

2. Область применимости ПО.

3. Режим работы ПО, сообщения, выдаваемые по ходу его работы, ответы пользователя на них (если это необходимо).

4. Исходные данные, необходимые для работы ПО; а также выдаваемые им результаты;.

5. Правила подготовки исходных данных на внешних носителях (если они применяются) и вид выдаваемой информации.

6. Описание структуры данных. Для любой переменной описывается ее назначение, атрибуты (тип, размер массива и т.д.), структура информации в ней, если она не очевидна. Описание переменных должно начинаться с тех, которые служат исходными данными и результатами.

7. Описания форм, объектов. Опись свойств форм и объектов.

8. Тексты программ, процедур (в виде распечатки ЭВМ) с комментариями.

9. Тесты.

10. Инструкция (руководство) пользователю.

Инструкция по использованию программы (или просто «Инструкция пользователю», или «Руководство для пользователя») - это выдержка из полной документации, предназначенная для эксплуатации программы. Она представляет собой независимый документ для пользователя программы, в котором описывается: что делает программа и как им пользоваться.

«Инструкция пользователю» должна содержать всю необходимую для пользователя информацию и должна быть ему понятна без дополнительных материалов (без обращения к другим спецификациям). Следовательно, необходимая для этой инструкции информация переписывается полностью из соответствующих спецификаций.

Первая часть инструкции является описательной и должна содержать:

- наименование программы;

- краткое описание программы;

- перечень выполняемых программой функций;

- краткую характеристику метода (или методов) решения поставленной задачи, его достоинство и недостатки;

- полную библиографическую ссылку на полное описание метода;

- описание входных и выходных данных.

- описание структуры базы данных (если она имеется), всех ее таблиц в словесной (вербальном) форме.

Вторая часть документа должна описывать порядок работы с программой. Она должна содержать описание всех режимов работы программы, а также содержание всех печатей и диагностических сообщений, которые выдаются по ходу выполнения программы.

Следует помнить, что пользователь по своей квалификации не является программистом и поэтому его работа с программой описывается на понятном ему языке и достаточно подробно, а именно:

- как запустить программу;

- как продолжить работу с программой (описывается подробный интерактивный режим работы пользователя с программой);

- подготовка и ввод исходных данных в программу;

- как реагировать на запросы программы;

- как вести работу в исключительных ситуациях;

- как реагировать на ошибки;

- как восстановить работу программы в случае аварийного его завершения;

- как получить требуемый результат;

- как правильно закончить работу с программой (запланированный программой выход);

- другие сведения, необходимые пользователю программы.

- Контрольные вопросы

Составление документа «Руководство пользователю».

1. Использование программного обеспечения.

2. Сопровождение программного обеспечения.

3. Разработка программного обеспечения «Под ключ».

4. Документация к программному обеспечению.

5. Инструкция по использованию программного обеспечения.

Дополнительные вопросы

1. Надежность программного обеспечения.

2. Программные изделия с малой и большой длительностью эксплуатации.

3. Системный анализ (предварительное проектирование).

4. Конструирование программного обеспечения.

5. Программирование программного обеспечения.

6. Оценка (испытание) программного обеспечения.

7. Использование (эксплуатация и сопровождение) ПО.

8. Разбиение программных проектов на группы с точки зрения разработки требований к ним.

9. Документ «Соглашение о требованиях».

10. Проблемы проектирования больших программных средств.

11. Система автоматизированного проектирования ПО (САПР ПО).

12. Объектно – ориентированное проектирование ПО.

13. Моделирование системы. Модель системы. Объектная модель. Объект, класс, свойства объектов, операции, методы.Обобщение и наследование.

14. Основные принципы проектирования программного обеспечения.

15. Конструирование объектной модели.

16. Документ «Внешняя спецификация».

17. Система автоматизации программирования (САП).

18. Документ «Внутренняя спецификация».

19. Оптимизация эффективности эксплуатации ПО.

20 Объектно-ориентированное программирование.

21. Понятия объектно-ориентированного программирования: объект, класс, событие, метод, инкапсуляция, наследование, полиморфизм, форма.

22. Распределение операций по классам.

23. Объектно-ориентированные языки программирования.

24. Визуальное программирование. Интегрированная среда разработки приложения. Мастер разработки приложения. Элементы управления приложением.

25. «Альфа» и «Бета» тестирования программного обеспечения.

26. Система автоматизации отладки программного обеспечения.

27. Оценка (испытания) программного обеспечения.

28. Предварительные и совместные испытания ПО.

29. Свойства качественного программного обеспечения.

30. Проблема оценки качества программных продуктов.

ТЕСТЫ

ВАРИАНТ 1

1. Какие программы можно отнести к системному программному обеспечению:
   а) прикладные программы;
   б) операционные системы;
   в) игровые программы.
2. Можно ли отнести операционную систему к программному обеспечению:
   а) да;
   б) нет.
3. Самый большой этап в жизненном цикле программы:
   а) изучение предметной области;
   б) программирование;
   в) тестирование;
   г) эксплуатация;
   д) корректировка ошибок.
4. Какой этап выполняется раньше:
   а) отладка;
   б) тестирование.
5. Способы оценки качества:
   а) наличие документации;
   б) сравнение с аналогами;
   в) оптимизация программы;
   г) структурирование алгоритма.
6. Существует ли связь между эффективностью и оптимизацией программы:
   а) да;
   б) нет.
7. Можно ли внутри цикла поместить еще один цикл:
   а) да;
   б) нет.
8. Можно ли ставить знак подчеркивания в начале имени:
   а) да, без ограничений;
   б) да, но не рекомендуется;
   в) нет.
9. Как называется способ составления имен переменных, когда в начале имени сообщается тип переменной:
   а) прямым указанием;
   б) венгерской нотацией;
   в) структурным программированием;

г) поляризацией.

10. Можно ли писать комментарии в отдельной строке:
    а) да;
    б) нет.
11. Наличие комментариев позволяет:
    а) быстрее писать программы;
    б) быстрее выполнять программы.
    в) быстрее найти ошибки в программе;
12. Возможно ли комбинирование языков программирования в рамках одной задачи:
    а) нет.
    б) да;
13. Для решения инженерных задач характерно применение:
    а) САПР (систем автоматизированного проектирования);
    б) СУБД (систем управления базами данных);
    в) ОС (операционных систем).
14. Причины синтаксических ошибок:
    а) ошибки в исходных данных;
    б) ошибки, допущенные на более ранних этапах;
    в) плохое знание языка программирования;
    г) неправильное применение процедуры тестирования.
15. Защитное программирование это:
    а) встраивание в программу отладочных средств;
    б) создание задач защищенных от копирования;
    в) разделение доступа в программе;
    г) использование паролей;
16. Отладка – это:
    а) определение списка параметров;
    б) правило вызова процедур (функций);
    в) процедура поиска ошибок, когда известно, что ошибка есть;
    г) составление блок-схемы алгоритма.
17. Когда программист может проследить последовательность выполнения команд программы:
    а) при тестировании;
    б) при трассировке;
    в) при компиляции;
    г) при выполнении программы;
    д) при компоновке.

18. На каком этапе создания программы могут появиться синтаксические ошибки:
    а) анализ требований;
    б) проектирование;
    в) программирование;
    г) тестирование.
19. Позволяет ли автоматизация программирования всегда создавать эффективные программы:
    а) да.
    б) нет;

20. Позволяет ли автоматизация программирования всегда создавать надежные программы:

    а) нет;
    б) да.

21. Что легко поддается автоматизации:

    а) работа с файлами;

    б) сложные логические задачи;

    в) интерфейс;

    г) алгоритмизация.

22. Что такое оптимизация программ:

    а) создание удобного интерфейса пользователя;

    б) улучшение работы существующей программы;

    в) разработка модульной конструкции программы;

    г) применение методов объектно-ориентированного программирования.

23. Сущность оптимизации циклов:

    а) ;трассировка циклов;

    б) сокращение тела цикла;

    в) представление циклов в виде блок-схем;

    г) сокращение количества повторений выполнения тела цикла

24. В чем сущность модульного программирования:

а) в разбиении программы на отдельные равные части;
б) в разбиении программы на отдельные функционально независимые части;
в) в разбиение программы на процедуры и функции;
г) снижает количество ошибок.

25. Недостаток модульного программирования:

а) увеличивает трудоемкость программирования;

б) снижает быстродействие программы;

в) не позволяет выполнять оптимизацию программы.

г) усложняет процедуру комплексного тестирования;

26. При структурном программировании задача выполняется:

а) поэтапным разбиением на более легкие задачи;
б) без участия программиста;
в) объединением отдельных модулей программы.

27. Достоинство структурного программирования:

а) можно приступить к автономному тестированию на раннем этапе разработки;
б) нет необходимости выполнять тестирование;
в) можно приступить к комплексному тестированию на раннем этапе разработки;
г) можно пренебречь отладкой.

28. Может ли дочерний элемент иметь двух родителей:

а) да;

б) нет;

в) только для визуальных элементов;

г) если их свойства совпадают.

29. Есть ли различие между объектом и экземпляром:

а) да;

б) нет;

в) если у них общий предок.

30. Могут ли два экземпляра одного объекта реагировать на событие по-разному:

а) да;

б) нет.

31. Какие этапы проектирования можно объединять:

а) эскизный и рабочий;
б) технический и эскизный.
в) технический и рабочий;

32. Процесс преобразования постановки задачи в план алгоритмического или вычислительного решения это:

а) анализ требований;
б) программирование;
в) проектирование;
г) тестирование.

33. Модульное программирование применимо при:

а) проектировании сверху вниз;
б) проектирование снизу-вверх;

34. Проектирование сверху вниз это:

а) последовательное разбиение общих задач на более мелкие;
б) составление из отдельных модулей большой программы.

35. Проектирование снизу-вверх это:

а) составление из отдельных модулей большой программы;
б) последовательное разбиение общих задач на более мелкие.

36. Зависит ли трудоемкость разработки от вида информации:

а) да;
б) нет.

37. Кому принадлежит право собственности на ПО:

а) продавцу;
б) разработчику;
в) покупателю.

38. Кому принадлежит авторское право на ПО:

а) покупателю.

б) продавцу;

в) разработчику;

39. Если вы приобрели программы законным путем, имеете ли вы право продать ее:

а) да;

б) нет.

40. Если вы приобрели программу законным путем, являетесь ли вы

собственником программы:

а) нет;

б) да.

ВАРИАНТ 2

1. Специфические особенности ПО как продукта:
   а) продажа по ценам ниже себестоимости (лицензирование);
   б) низкие материальные затраты при создании программ;
   в) возможность создание программ небольшие коллективом или даже одним человеком;
   г) разнообразие решаемых задач с помощью программных средств.

2. Можно ли отнести операционную систему к прикладному программному обеспечению:
   а) да;
   б) нет.

3. Какой этап выполняется раньше:

а) отладка;

б) тестирование.

4. В стадии разработки программы не входит:

а) постановка задачи;

б) составление спецификаций;

в) эскизный проект;

г) автоматизация программирования;

д) тестирование.

5. Наиболее важный критерий качества:

а) надежность;

б) быстродействие;

в) удобство в эксплуатации;

г) удобный интерфейс;

д) эффективность.

6. Способы оценки надежности:

а) сравнение с аналогами;
б) тестирование;
в) трассировка;
г) оптимизация.

7. Можно ли внутри условного оператора поместить еще одно условие:

а) да;
б) нет.

8. Какие символы не допускаются в именах переменных:

а) цифры;
б) подчеркивание
в) пробелы;

9. Транслируются ли комментарии:

а) да;
б) нет.

10. Что определяет выбор языка программирования:

а) область приложения;
б) знание языка;
в) наличие дополнительных библиотек.

11. Наличие комментариев позволяет:

а) применять сложные структуры;
б) легче разобраться в программе;
в) увеличить быстродействие.

12. Допустимо ли комбинирование языков программирования в рамках одной задачи :

а) нет.
б) да;

13. Для решения экономических задач характерно применение:

а) СУБД (систем управления базами данных);
б) языков высокого уровня;
в) языков низкого уровня;
г) применение сложных математических расчетов.

14. Когда можно обнаружить синтаксические ошибки:

а) при отладке;

б) при тестировании;

в) на этапе проектирования;

г) при компиляции;

д) при эксплуатации.

15. Вид ошибки с неправильным написанием служебных слов (операторов):

а) синтаксическая;
б) семантическая;
в) логическая;
г) символьная.

16. Когда программист может проследить последовательность выполнения команд программы:

а) при тестировании;
б) при компиляции;
в) при выполнении программы;
г) при трассировке;
д) при компоновке.

17. Когда приступают к тестированию программы:

а) после постановки задачи;
б) на этапе программирования;
в) на этапе проектирования;
г) когда программа уже закончена;
д) после составления спецификаций,

18. Тестирование бывает:

а) инструментальное;
б) автономное;
в) визуальное;
г) алгоритмическое.

19. Назначение отладки:

а) поиск возможных ошибок;
б) составление спецификаций;
в) разработка алгоритма.
г) поиск причин существующих ошибок;

20. Отладка программ это:

а) локализация и исправление ошибок;
б) алгоритмизация программирования;
в) компиляция и компоновка.

21. В чем сущность автоматизации программирования:

а) получение готовой программы без выполнения компоновки;
б) создание программы без написания ее текста;
в) в отсутствии компиляции.

22. Выполняется ли процедура компиляции при автоматизации программирования:

а) да;

б) нет.

23. Относится ли визуальное программирование к средствам автоматизации:

а) нет.

б) да;

24. Нахождение наилучшего варианта из множества возможных:

а) тестирование;

б) оптимизация;

в) автоматизация;

г) отладка;

д) сопровождение.

25. Результат оптимизации программы:

а) надежность;

б) машино-независимость;

в) эффективность;

г) мобильность.

26. Критерии оптимизации программы:

а) быстродействие или размер программы;

б) быстродействие и размер программы;

в) надежность или эффективность;

г) надежность и эффективность.

27. Достоинство модульного программирования:

а) не требует комплексного тестирования;

б) возможность приступить к тестированию до завершения написания всей программы;

в) уменьшает размер программы;

г) повышает надежность программы.

28. Разрешается ли использование циклов при структурном программированиир:

а) да;
б) нет.

29. Достоинство структурного программирования:

а) повышает быстродействие программы;
б) облегчает работу над большими и сложными проектами;
в) снижает затраты на программирование.

1. Какое утверждение верно:

а) предки наследуют свойства родителей;

б) родители наследуют свойства потомков;

в) потомки не могут иметь общих предков;

г) потомки наследуют свойства родителей.

31. Изменение свойств, приводит к изменению поведения экземпляра:

а) нет;

б) только для визуальных;

в) только НЕ для визуальных;

г) да.

32. Какой методикой проектирования пользуются при структурном программировании:

а) сверху вниз;
б) снизу-вверх.

33. Какой этап проектирования может быть исключен:

а) эскизный проект;
б) технический проект;
в) рабочий проект.

34. Какие этапы проектирования можно объединять:

а) эскизный и рабочий;
б) технический и рабочий;
в) технический и эскизный.

35. Модульное программирование применимо при:

а) проектировании сверху вниз;
б) проектирование снизу-вверх;

36. Модульное программирование применимо при:

а) проектировании сверху вниз;
б) проектирование снизу-вверх;
в) и в том, и другом случае;
г) ни в коем случае.

37. Какой методикой проектирования пользуются при структурном программировании:

а) сверху вниз;
б) снизу-вверх.

38. Зависит ли трудоемкость разработки от вида информации:

а) да;
б) нет.

39. Если вы приобрели программы законным путем, имеете ли вы право вносить в нее изменения:

а) да
б) нет;

40. Если вы приобрели программы законным путем, имеете ли вы право продать ее:

а) да;
б) нет.

ВАРИАНТ 3

1. Какие программы можно отнести к системному ПО:

а) драйверы;
б) текстовые редакторы;
в) электронные таблицы;
г) графические редакторы.

2. Специфические особенности ПО как продукта:

а) универсальность;
б) низкие затраты при дублировании;
в) простота эксплуатации;
г) наличие поддержки (сопровождения) со стороны разработчика.

3. Какой этап выполняется раньше:

а) отладка;

б) оптимизация;

в) программирование;

г) тестирование.

4. Специфические особенности ПО как продукта:

а) низкие затраты при дублировании;
б) универсальность;
в) простота эксплуатации;
г) наличие поддержки (сопровождения) со стороны разработчика.

5. Повышает ли качество программ оптимизация:

а) да;
б) нет.

6. Существует ли связь между надежностью и быстродействием:

а) нет:

б) да.

7. Можно ли одно большое (длинное) выражение разбить на несколько выраженийр:

а) да;
б) нет.

8. Найдите НЕ правильное условие для создания имен:

а) длинное имя можно сократить;
б) из имени лучше выбрасывать гласные;
в) имена могут содержать пробелы;
г) можно использовать большие буквы.

9. Наличие комментариев позволяет:

а) улучшить читабельность программы;
б) улучшить эксплуатацию программы;
в) повысить надежность программы.

10. Что определяет выбор языка программирования:

а) знание языка;
б) область приложения;
в) наличие дополнительных библиотек.

11. Наличие комментариев позволяет:

а) улучшить читабельность программы;
б) улучшить эксплуатацию программы;
в) повысить надежность программы.

12. Для каких задач характерно использование большого количества исходных данных, выполнение операций поиска, группировки:

а) для системных задач;
б) для экономических задач;
в) для инженерных задач.

13. Можно ли использовать комбинацию языков программирования в рамках одного проекта:

а) да;
б) нет.

14. Ошибки компоновки заключаются в том, что:

а) неправильно использовано зарезервированное слово;

б) составлено неверное выражение;

в) указано внешнее имя, но не объявлено;

г) указан неверный тип переменной.

15. Вид ошибки с неправильным использованием служебных слов (операторов):

а) синтаксическая;
б) семантическая;
в) логическая;
г) символьная.

16. Программа для просмотра значений переменных при выполнении программы:

а) компилятор;
б) интерпретатор;
в) отладчик;
г) трассировка;
д) тестирование.

17. Тестирование бывает:

а) инструментальное;
б) комплексное;
в) визуальное;
г) алгоритмическое.

18. При комплексном тестировании проверяются:

а) правильность работы отдельных частей программы;
б) согласованность работы отдельных частей программы;
в) быстродействие программы;
г) эффективность программы.

19. Существует ли различие между отладкой и тестированием:

а) да;
б) нет.

20. Что выполняется раньше, отладка или тестирование:

а) отладка;
б) тестирование.

21. В чем сущность автоматизации программирования:

а) получение готовой программы без выполнения компоновки;

б) создание программы без написания ее текста;
в) в отсутствии компиляции.

22. Влияет ли автоматизация программирования на эффективность программы:

а) нет;
б) да

23. Позволяет ли автоматизация программирования всегда создавать надежные программы:

а) нет;
б) да.

24. Критерии оптимизации:

а) размер программы и ее эффективность;

б) время выполнения или размер требуемой памяти;

в) независимость модулей;

г) качество программы, ее надежность.

25. Нахождение наилучшего варианта из множества возможных:

а) тестирование;

б) автоматизация;

в) отладка;

г) оптимизация;

д) сопровождение.

26. Рекомендуемые размеры модулей:

а) большие;
б) равные;
в) небольшие;
г) фиксированной длины.

27. В чем заключается независимость модуля:

а) в написании, отладке и тестировании независимо от остальных модулей;
б) в разработке и написании независимо от других модулей;
в) в независимости от работы основной программы.

28. Допустимо ли использование оператора GO TO при структурном программировании:

а) нет;
б) да.

29. Возможно, ли преобразовать неструктурированную программу к структурному виду:

а) да;
б) нет.

30. Недостаток структурного программирования:

а) снижает эффективность;
б) уменьшает количество ошибок;
в) увеличивает размер программы;
г) не требует отладки.

31. Три "кита" объектно-ориентированного метода программирования:

а) предки, родители, потомки;

б) полиморфизм, инкапсуляция, наследование;

в) свойства, события, методы;

г) визуальные, не визуальные компоненты и запросы.

32. Можно ли свойствам присваивать значения:

а) да (всегда);

б) не всегда;

в) нет.

33. Модульное программирование применимо при:

а) проектировании сверху вниз;
б) проектирование снизу-вверх;

34. Процесс преобразования постановки задачи в план алгоритмического или вычислительного решения это:

а) проектирование;
б) анализ требований;
в) программирование;

г) тестирование.

35. Процесс преобразования постановки задачи в план алгоритмического или вычислительного решения это:

а) анализ требований;
б) программирование;
в) проектирование;
г) тестирование.

36. Этап разработки программы, на котором дается характеристика области применения программы:

а) эскизный проект;
б) технический проект;
в) внедрение;
г) рабочий проект.
д) техническое задание;

37. Составление спецификаций это:

а) эскизный проект;
б) поиск алгоритма;
в) формализация задачи;
г) отладка.

38. В чем заключается иерархический подход в решении задачи:

а) в выделении основных и второстепенных элементов;
б) в последовательном разбиении задачи на более мелкие составные части;
в) в возможности параллельного выполнения отдельных частей задачи.

39. Какой метод проектирования соответствует иерархическому подходу в решении задачи:

а) нисходящее (сверху вниз);
б) восходящее (снизу-вверх).

40. Кому принадлежит авторское право на ПО:

а) разработчику;
б) продавцу;
в) покупателю.

ВАРИАНТ 4

1. Какие программы можно отнести к системному ПО:
   а) программа расчета заработной платы;

б) электронные таблицы;

в) СУБД (системы управления базами данных).

2. Какие программы можно отнести к системному ПО:

а) утилиты;

б) экономические программы;

в) статистические программы;

г) мультимедийные программы.

3. Что выполняется раньше:

а) компиляция;

б) отладка;

в) компоновка;

г) тестирование.

4. Этап, занимающий наибольшее время, в жизненном цикле программы:

а) проектирование;

б) тестирование;

в) программирование;

г) сопровождение;

д) формулировка требований.

5. В каких единицах можно измерить надежность:

а) км/час;

б) отказов/час;

в) Кбайт/сек;

г) операций/сек.

6. Что относится к этапу программирования:

а) написание кода программы;

б) разработка интерфейса;

в) работоспособность;

г) анализ требований.

7. . Если имеется стандартная функция, нужно ли писать собственную:

а) нет;

б) да.

8. . Доступ, при котором записи файла читаются в физической последовательности, называется:

1) прямым;

2) простым;

3) последовательным;

4) основным.

9. Можно ли ставить знак подчеркивания в начале имени:

а) да, но не рекомендуется;

б) да, без ограничений;

в) нет.

10. Как называется способ составления имен переменных, когда в начале имени сообщается тип переменной:

а) прямым указанием;

б) венгерской нотацией;

в) структурным программированием;

г) поляризацией.

11. . Что определяет выбор языка программирования:

а) область приложения;

б) знание языка;

в) наличие дополнительных библиотек.

12. Для каких задач характерен большой объем вычислений, использование сложного математического аппарата:

а) для системных задач;

б) для инженерных задач;

в) для экономических задач.

13. . На каком этапе производится выбор языка программирования:

а) проектирование;

б) программирование;

в) отладка;

г) тестирование.

14. Могут ли проявиться ошибки при изменении условий эксплуатации:

а) да;

б) нет.

15. Ошибки при написании программы бывают:

а) орфографические;

б) лексические;

в) синтаксические;

г) фонетические;

д) морфологические.

16. Отладка – это:

а) определение списка параметров;

б) правило вызова процедур (функций);

в) процедура поиска ошибок, когда известно, что ошибка есть;

г) составление блок-схемы алгоритма.

17. При комплексном тестировании проверяются:

а) правильность работы отдельных частей программы;

б) согласованность работы отдельных частей программы;

в) быстродействие программы;
г) эффективность программы.

18. Чему нужно уделять больше времени, чтобы получить хорошую программу:

а) программированию;
б) отладке;
в) тестированию;
г) проектированию.

19. Назначение тестирования:

а) обнаружение ошибок;
б) повышение эффективности программы;
в) улучшение эксплуатационных характеристик;
г) повышение надежности программы;
д) приведение программы к структурированному виду.

20. Инструментальные средства отладки (НЕ правильный ответ):

а) трассировка.
б) отладчики;
в) компиляторы;

21. Возможны ли ошибки при автоматизации программирования:

а) да;

б) нет.

22. Один из методов автоматизации программирования:

а) структурное программирование;
б) модульное программирование;
в) визуальное программирование;
г) объектно-ориентированное программирование.

23. Нахождение наилучшего варианта из множества возможных:

а) тестирование;

б) автоматизация;

в) отладка;

г) сопровождение.

д) оптимизация;

24. Критерии оптимизации:

а) эффективность использования ресурсов;

б) структурирование алгоритма;

в) структурирование программы.

25. В чем заключается оптимизация условных выражений:

а) в использовании простых логических выражений;

б) в изменении порядка следования элементов выражения;

в) в использовании сложных логических выражений;

г) в использовании операций AND, OR и NOT.

26. В чем сущность модульного программирования:

а) в разбиении программы на отдельные равные части;

б) в разбиении программы на отдельные функционально независимые части;

в) в разбиение программы на процедуры и функции;

27. Можно ли сочетать модульное и структурное программирование:

а) да;

б) нет.

28. Можно ли сочетать структурное программирование с модульным:

а) можно;

б) нельзя;

в) только в особых случаях.

29. При структурном программировании задача выполняется:

а) без участия программиста;

б) поэтапным разбиением на более легкие задачи;

в) объединением отдельных модулей программы.

30. Повышает ли читабельность программ структурное кодирование:

а) да;

б) нет.

31. Полиморфизм это:

а) передача свойств по наследству;

б) изменение поведения потомков на разные события;

в) изменение поведения потомков, имеющих общих предков;

г) изменение поведения экземпляров, имеющих общих предков;

32. Можно ли переопределять методы:

а) да;

б) нет.

33. В каких единицах измеряются затраты на проектирование:

а) в человеко-днях;
б) в долларах;
в) в тенге;
г) в килобайтах.

34. Можно ли переопределять свойства:

а) да;

б) нет.

35. Составление спецификаций это:

а) эскизный проект;
б) формализация задачи;
в) поиск алгоритма;
г) отладка.

36. Этап разработки программы, на котором дается характеристика области применения программы:

а) эскизный проект;
б) технический проект;
в) внедрение;
г) рабочий проект.
д) техническое задание;

37. Этап разработки программы, на котором дается характеристика области применения программы:

а) технический проект;
б) техническое задание;
в) эскизный проект;
г) внедрение;
д) рабочий проект.

38. Укажите правильную последовательность создания программы:

а) анализ требований, проектирование, программирование, тестирование, отладка;
б) анализ требований, программирование, проектирование, тестирование;
в) анализ требований, проектирование, программирование, модификация, трассировка;
г) формулирование задачи, анализ требований, проектирование, программирование;
д) формулирование задачи, анализ требований, программирование, проектирование, отладка.

39. В каких единицах измеряются затраты на проектирование:

а) в долларах;
б) в человеко-днях;
в) в тенге;
г) в килобайтах.

40. Зависит ли трудоемкость разработки от сложности алгоритма:

а) да;
б) нет.

ВАРИАНТ 5

1. Какие программы нельзя отнести к системному ПО:
   а) компиляторы языков программирования;
   б) операционные системы;
   в) игровые программы;
   г) системы управления базами данных.
2. Этап, занимающий наибольшее время, в жизненном цикле программы:
   а) тестирование;
   б) программирование;

в) формулировка требований.

г) сопровождение;

д) проектирование;

3. Что выполняется раньше:
   а) программирование;
   б) отладка;

   в) тестирование.

   г) проектирование;

4. Самый большой этап в жизненном цикле программы:
   а) эксплуатация;
   б) изучение предметной области;

   в) программирование;

   г) тестирование;

   д) корректировка ошибок.

5. В каких единицах можно измерить быстродействие:
   а) отказов/час;
   б) км/час;
   в) Кбайт/сек;
   г) операций/сек.
6. Последовательность этапов программирования:
   а) компоновка, отладка, компилирование;
   б) отладка, компилирование, компоновка;

   в) компилирование, отладка, компоновка.

   г) компилирование, компоновка, отладка;

7. Инструментальные средства программирования:
   а) СУБД (системы управления базами данных);
   б) BIOS (базовая система ввода-вывода);
   в) ОС (операционные системы).
   г) компиляторы, интерпретаторы;
8. Что выполняется раньше:
   а) разработка алгоритма;

б) выбор языка программирования;

в) написание исходного кода;

г) компиляция.

9. Если имеется стандартная функция, нужно ли писать собственную:
   а) нет;
   б) да.
10. Наличие комментариев позволяет:
   а) быстрее найти ошибки в программе;
   б) быстрее писать программы;
   в) быстрее выполнять программы.
11. На каком этапе производится выбор языка программирования:
   а) программирование;
   б) отладка;
   в) тестирование.
   г) проектирование;
12. Для каких задач характерен большой объем вычислений, использование сложного математического аппарата:
   а) для инженерных задач;
   б) для системных задач;
   в) для экономических задач.
13. Могут ли проявиться ошибки при изменении в предметной области:
   а) да;
   б) нет.
14. Процедура поиска ошибки, когда известно, что она есть это:
   а) тестирование;
   б) компоновка;
   в) отладка;
   г) транзакция;
   д) трансляция.
15. Ошибки при написании программы бывают:
   а) синтаксические;
   б) орфографические;
   в) лексические;
   г) фонетические;
   д) морфологические.
16. Процесс исполнения программы с целью обнаружения ошибок:
   а) кодирование;
   б) тестирование;
   в) сопровождение;

г) проектирование.

17. Автономное тестирование это:

а) составление блок-схем;

б) пошаговая проверка выполнения программы

в) тестирование отдельных частей программы;

г) инструментальное средство отладки;

18. Инструментальные средства отладки (НЕ правильный ответ):

а) компиляторы;

б) отладчики;

в) трассировка.

19. Отладка программ это:

а) алгоритмизация программирования;

б) локализация и исправление ошибок;

в) компиляция и компоновка.

20. Недостаток автоматизации программирования;

а) низкое быстродействие;

б) большой размер программы;

в) сложность программы.

21. Возможны ли ошибки при автоматизации программирования:

а) да;

б) нет.

22. Возможна ли оптимизация программ без участия программиста:

а) да;

б) нет.

23. Нахождение наилучшего варианта из множества возможных:

а) тестирование;

б) автоматизация;

в) отладка;

г) сопровождение.

д) оптимизация;

24. В чем заключается независимость модуля:

а) в разработке и написании независимо от других модулей;

б) в независимости от работы основной программы.

в) в написании, отладке и тестировании независимо от остальных модулей;

25. При модульном программировании желательно, чтобы модуль имел:
    а) большой размер;
    б) небольшой размер;
    в) фиксированный размер;
    г) любой размер.
26. Любую ли программу можно привести к структурированному виду:
    а) любую;
    б) не все;
    в) нельзя.
27. Разрешается ли использование оператора GO TO при структурном программировании:
    а) да;
    б) иногда.
    в) нет;
28. Разрешается ли использование циклов при объектно-ориентированном программировании:
    а) да;
    б) нет.
29. Наследование это:

    а) передача свойств экземплярам;

    б) передача свойств предкам;

    в) передача свойств потомкам;

    г) передача событий потомкам.

30. Предусматривает ли объектно-ориентированное программирование использование стандартных процедур и функций:
    а) да;
    б) нет.
31. Какой методикой проектирования пользуются при структурном программировании:

    а) сверху вниз;
    б) снизу-вверх.
32. Составление спецификаций это:

    а) эскизный проект;
    б) формализация задачи;
    в) поиск алгоритма;

г) отладка.

33. Могут ли два различных объекта реагировать на событие по-разному:

а) да;

б) нет.

34. Несуществующий метод проектирования:

а) алгоритмическое;
б) нисходящее;
в) восходящее.

35. Укажите правильную последовательность создания программы:

а) анализ требований, проектирование, программирование, тестирование, отладка;
б) анализ требований, программирование, проектирование, тестирование;
в) анализ требований, проектирование, программирование, модификация, трассировка;
г) формулирование задачи, анализ требований, программирование, проектирование, отладка.
д) формулирование задачи, анализ требований, проектирование, программирование;

36. Уточнение структуры входных и выходных данных, разработка алгоритмов, определение элементов интерфейса входят в:

а) рабочий проект;
б) эскизный проект.
в) технический проект;

37. Несуществующий метод проектирования:

а) алгоритмическое;
б) нисходящее;
в) восходящее.

38. Зависит ли трудоемкость разработки от сложности алгоритма:

а) да;
б) нет.

39. Какой метод проектирования соответствует иерархическому подходу в решении задачи:

а) нисходящее (сверху вниз);

б) восходящее (снизу-вверх).

40. Если вы приобрели программы законным путем, имеете ли вы право продать ее:

а) да;
б) нет.

ВАРИАНТ 6

1. Какие программы можно отнести к прикладному программному обеспечению:

а) электронные таблицы;
б) таблицы решений;
в) СУБД (системы управления базами данных).

2. В стадии разработки программы не входит:

а) составление спецификаций;
б) эскизный проект;
в) тестирование.

г) автоматизация программирования;
д) постановка задачи;

3. Что выполняется раньше:

а) программирование;

б) проектирование;

в) отладка;

г) тестирование.

4. В стадии разработки программы не входит:

а) постановка задачи;
б) составление спецификаций;
в) эскизный проект;
г) тестирование.

д) автоматизация программирования;

5. На языке программирования составляется:

а) исполняемый код;
б) объектный код;
в) алгоритм.
г) исходный код;

6. Правила, которым должна следовать программа это:

а) алгоритм;
б) структура;
в) спецификация;
г) состав информации.

7. Можно ли переменным присваивать произвольные идентификаторы:

а) да;
б) нет.

8. Найдите НЕ правильное условие для создания имен:

а) длинное имя можно сократить;
б) из имени лучше выбрасывать гласные;
в) можно использовать большие буквы.
г) имена могут содержать пробелы;

9. Доступ, при котором записи файла обрабатываются в произвольной последовательности, называется:

а) последовательным;
б) простым;
в) основным.
г) прямым;

10. Что определяет выбор языка программирования:

а) знание языка;
б) наличие дополнительных библиотек.
в) область приложения;

11. Транслируются ли комментарии:

а) да;
б) нет.

12. Можно ли использовать комбинацию языков программирования в рамках одного проекта:

а) да;
б) нет.

13. На каком этапе производится выбор языка программирования:

а) проектирование;
б) программирование;
в) отладка;
г) тестирование.

14. Возможно ли программирование с защитой от ошибок:

а) да;

б) нет.

15. Программа для просмотра значений переменных при выполнении программы:

а) компилятор;
б) интерпретатор;
в) трассировка;
г) тестирование.
д) отладчик;

16. Вид ошибки с неправильным использованием служебных слов (операторов):

а) синтаксическая;
б) логическая;
в) символьная.
г) семантическая;

17. Трассировка это:

а) проверка пошагового выполнения программы;
б) тестирование исходного кода;
в) отладка модуля;
г) составление блок-схемы алгоритма.

18. Локализация ошибки:

а) определение причин ошибки;
б) определение места возникновения ошибки;
в) обнаружение причин ошибки;
г) исправление ошибки.

19. Локализация ошибки:

а) определение причин ошибки;
б) обнаружение причин ошибки;

в) определение места возникновения ошибки;
г) исправление ошибки.

20. Назначение тестирования:

а) обнаружение ошибок;
б) повышение эффективности программы;
в) улучшение эксплуатационных характеристик;
г) приведение программы к структурированному виду.
д) повышение надежности программы;

21. Выполняется ли процедура компиляции при автоматизации программирования:

а) да;

б) нет.

22. Что легко поддается автоматизации:

а) интерфейс;

б) работа с файлами;

в) сложные логические задачи;

г) алгоритмизация.

23. Модульное программирование это:

а) использование стандартных процедур и функций
б) разбиение программы на отдельные части;
в) структурирование;

24. Можно ли использовать оператор GO TO в модульном программах:

а) можно;
б) нельзя.

25. Разрешается ли использование циклов при структурном программированиир:

а) да;
б) нет.

26. Разрешается ли использование оператора IF при объектно-ориентированном программировании:

а) нет
б) да

27. Что такое объект, в объектно-ориентированное программировании:

а) событие;
б) обработка событий;
в) тип данных;
г) структура данных;
д) использование стандартных процедур.

28. Могут ли два экземпляра одного объекта реагировать на событие по-разному:

а) да;

б) нет.

29. Укажите правильную последовательность создания программы:

а) анализ требований, проектирование, программирование, тестирование, отладка;
б) анализ требований, программирование, проектирование, тестирование;
в) формулирование задачи, анализ требований, проектирование, программирование;
г) анализ требований, проектирование, программирование, модификация, трассировка;
д) формулирование задачи, анализ требований, программирование, проектирование, отладка.

30. Уточнение структуры входных и выходных данных, разработка алгоритмов, определение элементов интерфейса входят в:

а) технический проект;
б) рабочий проект;
в) эскизный проект.

31. Метод проектирования:

а) алгоритмическое;
б) логическое;
в) нисходящее;
г) использование языков программирования;
д) составление блок-схем.

32. Нисходящее проектирование это:

а) составление блок-схем;
б) разделение программы на отдельные участи (блоки);
в) последовательное уточнение (детализация);

г) трассировка.

33. В каких единицах измеряются затраты на проектирование:

а) в человеко-днях;
б) в долларах;
в) в тенге;
г) в килобайтах.

34. Зависит ли трудоемкость разработки от языка или системы программирования:

а) да;
б) нет.

35. Зависит ли трудоемкость разработки от сложности алгоритма:

а) да;
б) нет.

36. Зависит ли трудоемкость разработки от вида информации:

а) да;
б) нет.

37. Зависит ли трудоемкость разработки от количества обрабатываемой информации:

а) да;
б) нет.

38. Кому принадлежит право собственности на ПО:

а) продавцу;
б) разработчику;
в) покупателю.

39. Если вы приобрели программы законным путем, имеете ли вы право продать ее:

а) да;
б) нет.

40. Кому принадлежит право собственности на ПО:

а) разработчику;
б) продавцу;
в) покупателю.

ВАРИАНТ 7

1. Какие программы можно отнести к прикладному ПО:
   а) программа расчета заработной платы;
   б) диспетчер программ;

в) программа «Проводник» (Explorer).

2. Этап, занимающий наибольшее время, при разработке программы:
   а) тестирование;
   б) сопровождение;
   в) проектирование;
   г) программирование;
   д) формулировка требований.
3. Первый этап в жизненном цикле программы:
   а) анализ требований;
   б) проектирование;
   в) формулирование требований;
   г) автономное тестирование;
   д) комплексное тестирование.
4. Самый важный критерий качества программы:
   а) надежность;
   б) эффективность;
   в) работоспособность;
   г) быстродействие;
   д) простота эксплуатации.
5. В каких единицах можно измерить надежность:
   а) км/час;
   б) Кбайт/сек;
   в) отказов/час;
   г) операций/сек.
6. Способы оценки надежности:
   а) тестирование;
   б) сравнение с аналогами;
   в) трассировка;
   г) оптимизация.
7. Какие символы не допускаются в именах переменных:
   а) цифры;
   б) пробелы;
   в) подчеркивание
8. Транслируются ли комментарии:
   а) да;
   б) нет.
9. Наличие комментариев позволяет:
   а) быстрее найти ошибки в программе;

б) быстрее писать программы;

в) быстрее выполнять программы.

10. Какие символы не допускаются в именах переменных:

а) цифры

б) подчеркивание

в) пробелы

11. Можно ли ставить знак подчеркивания в начале имени:

а) да, без ограничений;

б) да, но не рекомендуется;

в) нет.

12. Можно ли писать комментарии в отдельной строке:

а) да;

б) нет.

13. Для решения экономических задач характерно применение:

а) языков низкого уровня;

б) применение сложных математических расчетов.

в) СУБД (систем управления базами данных);

г) языков высокого уровня;

14. Есть ли недостатки программирования с защитой от ошибок:

а) да;

б) нет.

15. Отладка – это:

а) определение списка параметров;

б) процедура поиска ошибок, когда известно, что ошибка есть;

в) правило вызова процедур (функций);

г) составление блок-схемы алгоритма.

16. Вид ошибки с неправильным написанием служебных слов (операторов):

а) семантическая;

б) логическая;

в) символьная.

г) синтаксическая;

17. На каком этапе создания программы могут появиться синтаксические ошибки:

а) проектирование;

б) анализ требований;

в) тестирование.

г) программирование;

18. Когда приступают к тестированию программы:

а) когда программа уже закончена;
б) после постановки задачи;
в) на этапе программирования;
г) на этапе проектирования;
д) после составления спецификаций,

19. Процесс исполнения программы с целью обнаружения ошибок:

а) кодирование;
б) сопровождение;
в) тестирование;
г) проектирование.

20. Трассировка это:

а) тестирование исходного кода;
б) отладка модуля;
в) проверка пошагового выполнения программы;
г) составление блок-схемы алгоритма.

21. Автоматизация программирования позволяет:

а) повысить надежность программы;
б) сократить время разработки программы;
в) повысить быстродействие программы.

22. Позволяет ли автоматизация программирования всегда создавать эффективные программы:

а) да.
б) нет;

23. Что такое оптимизация программ:

а) улучшение работы существующей программы;

б) создание удобного интерфейса пользователя;

в) разработка модульной конструкции программы;

г) применение методов объектно-ориентированного программирования.

24. Выполняется ли процедура компиляции при автоматизации программирования:

а) да;

б) нет.

25. Сущность оптимизации циклов:

а) сокращение тела цикла;

б) представление циклов в виде блок-схем;

в) сокращение количества повторений выполнения тела цикла;

г) трассировка циклов;

д) поиск ошибок в циклах.

26. В чем сущность модульного программирования:

а) в разбиении программы на отдельные функционально независимые части;
б) в разбиении программы на отдельные равные части;
в) в разбиение программы на процедуры и функции;

27. Можно ли использовать оператор GO TO в структурированных программах:

а) можно;
б) нельзя;
в) только в особых случаях.

28. Разрешается ли использование оператора IF при структурном программировании:

а) да;
б) нет.

29. Предусматривает ли объектно-ориентированное программирование использование стандартных процедур и функций:

а) да;
б) нет.

30. Предусматривает ли объектно-ориентированное программирование использование стандартных процедур и функций:

а) да;
б) нет.

31. Какой методикой проектирования пользуются при структурном программировании:

а) сверху вниз;
б) снизу-вверх.

31. Составление спецификаций это:

а) эскизный проект;
б) формализация задачи;
в) поиск алгоритма;
г) отладка.

32. Этап разработки программы, на котором дается характеристика области применения программы:

а) техническое задание;
б) эскизный проект;
в) технический проект;
г) внедрение;
д) рабочий проект.

33. Признаки нисходящего программирования:

а) наличие оптимизации;
б) наличие тестирования;
в) последовательная детализация;
г) автоматизация программирования.

34. Какой этап проектирования может быть исключен:

а) эскизный проект;
б) технический проект;
в) рабочий проект.

35. Какой методикой проектирования пользуются при структурном программировании:

а) сверху вниз;
б) снизу-вверх.

36. В чем заключается иерархический подход в решении задачи:

а) в последовательном разбиении задачи на более мелкие составные части;
б) в выделении основных и второстепенных элементов;
в) в возможности параллельного выполнения отдельных частей задачи.

37. Зависит ли трудоемкость разработки от языка или системы программирования:

а) да;
б) нет.

38. Зависит ли трудоемкость разработки от количества обрабатываемой информации:

а) да;

б) нет.

39. Зависит ли трудоемкость разработки от вида информации:

а) да;

б) нет.

40. Если вы приобрели программу законным путем, являетесь ли вы собственником программы:

а) нет;

б) да.

ВАРИАНТ 8

1. Какие программы нельзя отнести к прикладному ПО:
   а) компиляторы и (или) интерпретаторы;
   б) текстовые и (или) графические редакторы;
   в) электронные таблицы.
2. Один из необязательных этапов жизненного цикла программы:
   а) оптимизация;
   б) проектирование;
   в) тестирование;
   г) программирование;
   д) анализ требований.
3. Самый важный критерий качества программы:
   а) надежность;
   б) эффективность;
   в) работоспособность;
   г) быстродействие;
   д) простота эксплуатации.
4. В стадии разработки программы не входит:
   а) автоматизация программирования;
   б) постановка задачи;
   в) составление спецификаций;
   г) эскизный проект;
   д) тестирование.
5. Способы оценки качества:
   а) оптимизация программы;
   б) наличие документации;
   в) сравнение с аналогами;
   г) структурирование алгоритма.
6. Повышает ли качество программ оптимизация:

а) да;

б) нет.

7. Можно ли использовать имена, которые уже были использованы в другой программе (модуле):

а) да;

б) нет.

8. Как называется способ составления имен переменных, когда в начале имени сообщается тип переменной:

а) прямым указанием;

б) венгерской нотацией;

в) структурным программированием;

г) поляризацией.

9. Можно ли писать комментарии в отдельной строке:

а) да;

б) нет.

10. Найдите НЕ правильное условие для создания имен:

а) имена могут содержать пробелы;

б) длинное имя можно сократить;

в) из имени лучше выбрасывать гласные;

г) можно использовать большие буквы.

11. Какие символы не допускаются в именах переменных:

а) пробелы;

б) цифры;

в) подчеркивание

12. Наличие комментариев позволяет:

а) улучшить эксплуатацию программы;

б) улучшить читабельность программы;

в) повысить надежность программы.

г) ОС (операционных систем).

13. Для решения экономических задач характерно применение:

а) языков высокого уровня;

б) СУБД (систем управления базами данных);

в) языков низкого уровня;

г) применение сложных математических расчетов.

14. Есть ли недостатки программирования с защитой от ошибок:

а) да;

б) нет.

15. Когда программист может проследить последовательность выполнения команд программы:

а) при трассировке;

б) при тестировании;

в) при компиляции;

г) при выполнении программы;

д) при компоновке.

16. Защитное программирование это:

а) встраивание в программу отладочных средств;

б) создание задач защищенных от копирования;

в) разделение доступа в программе;

г) использование паролей;

д) оформление авторских прав на программу.

17. Программа для просмотра значений переменных при выполнении программы:

а) отладчик;

б) компилятор;

в) интерпретатор;

г) трассировка;

д) тестирование.

18. Отладка – это:

а) определение списка параметров;

б) процедура поиска ошибок, когда известно, что ошибка есть;

в) правило вызова процедур (функций);

г) составление блок-схемы алгоритма.

19. Тестирование бывает:

а) комплексное;

б) инструментальное;

в) визуальное;

г) алгоритмическое.

20. Существует ли различие между отладкой и тестированием:

а) да;

б) нет.

21. Один из методов автоматизации программирования:

а) структурное программирование;

б) модульное программирование;

в) визуальное программирование;

г) объектно-ориентированное программирование.

22. Влияет ли автоматизация программирования на эффективность программы:

а) нет;

б) да

23. В чем заключается оптимизация условных выражений:

а) в использовании простых логических выражений;

б) в использовании сложных логических выражений;

в) в изменении порядка следования элементов выражения;

г) в использовании операций AND, OR и NOT.

23. Оптимизация циклов заключается в:

а) уменьшении количества повторений тела цикла;

б) просмотре задачи с другой стороны;

в) упрощение задачи за счет включения логических операций.

24. При модульном программировании желательно, чтобы модуль имел:

а) большой размер;

б) небольшой размер;

в) фиксированный размер;

г) любой размер.

25. Модульное программирование это:

а) разбиение программы на отдельные части;

б) структурирование;

в) использование стандартных процедур и функций.

26. Разрешается ли использование циклов при структурном программированиир:

а) да;

б) нет.

27. Программирование без GO TO применяется. при:

а) модульном программировании;

б) объектно-ориентированном программировании;

в) структурном программировании;

г) все ответы верные.

28. . Можно ли сочетать объектно-ориентированное и структурное программирование

а) можно;

б) нельзя.

29. Инкапсуляция это:

а) определение новых типов данных;

б) определение новых структур данных;

в) объединение переменных, процедур и функций в одно целое;

г) разделение переменных, процедур и функций;

д) применение стандартных процедур и функций.

30. Проектирование сверху вниз это:

а) последовательное разбиение общих задач на более мелкие;
б) составление из отдельных модулей большой программы.

31. Проектирование снизу-вверх это:

а) составление из отдельных модулей большой программы;
б) последовательное разбиение общих задач на более мелкие.

32. Модульное программирование применимо при:

а) проектировании сверху вниз;
б) проектирование снизу-вверх;
в) и в том, и другом случае;
г) ни в коем случае.

33. Какой методикой проектирования пользуются при структурном программировании:

а) сверху вниз;
б) снизу-вверх.

34. Какой этап проектирования может быть исключен:

б) технический проект;
в) рабочий проект.

35. Модульное программирование применимо при:

а) проектировании сверху вниз;
б) проектирование снизу-вверх;

36. Процесс преобразования постановки задачи в план алгоритмического или вычислительного решения это:

а) проектирование;
б) анализ требований;
в) программирование;
г) тестирование.

37. В каких единицах измеряются затраты на проектирование:

а) в долларах;
б) в тенге;

в) в человеко-днях;

г) в килобайтах.

38. Зависит ли трудоемкость разработки от языка или системы программирования:

а) да;

б) нет.

39. Что охраняется законом:

а) структура базы данных;

б) содержание базы данных

40. Кому принадлежит авторское право на ПО:

а) разработчику;

б) продавцу;

в) покупателю.