Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Кубанский государственный университет»
Физико-технический факультет

УТВЕРЖДАЮ:
Проректор по учебной работе,
качеству образования – первый
проректор

_____ Хагуров Т.А.

«___» _____ 2020 г.

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ (МОДУЛЯ)

Б1.В.17 КОЛЛЕКТИВНАЯ РАЗРАБОТКА ПРОГРАММНЫХ ПРОДУКТОВ

Направление подготовки 09.03.02 Информационные системы и технологии

Направленность (профиль) Программное обеспечение информационных систем в цифровой экономике

Форма обучения очная

Квалификация выпускника бакалавр

Краснодар 2020

Рабочая программа дисциплины Коллективная разработка программных продуктов составлена в соответствии с федеральным государственным образовательным стандартом высшего образования (ФГОС ВО) по направлению подготовки 09.03.02 Информационные системы и технологии (Цифровые вычислительные комплексы и сети)

Программу составил(и):
В.А. Исаев, профессор кафедры теоретической физики и компьютерных технологий,
к. физ.- мат. наук, доцент

Рабочая программа дисциплины Коллективная разработка программных продуктов на заседании кафедры теоретической физики и компьютерных технологий
№ 10 от 16 апреля 2020 г.
Заведующий кафедрой (разработчика) В.А. Исаев

Рабочая программа обсуждена на заседании кафедры теоретической физики и компьютерных технологий
№ 10 от 16 апреля 2020 г.
Заведующий кафедрой (выпускающей) В.А. Исаев

Утверждена на заседании учебно-методической комиссии физико-технического факультета
№ 9 от 20 апреля 2020 г.
Председатель УМК факультета Богатов Н.М.

Рецензенты:

Н.М. Богатов, зав. кафедрой физики и информационных систем КубГУ, д. м.-ф. наук

Л.Р. Григорьян, ген. директор ООО НПФм «Мезон», к. м.-ф. наук

1 Цели и задачи изучения дисциплины (модуля)

Цель освоения дисциплины – ознакомить студентов с современными методами коллективного выполнения проектов по разработке программного обеспечения

1.2 Задачи дисциплины:

1. научить будущих выпускником-бакалавров практическому навыку коллективного выполнения проекта по разработке программного обеспечения, в соответствии с технологическим процессом, принятым в индустрии.

2. изучить проектную документацию, средства контроля версий, планирование потоков работ, управление задачами и управления дефектами.

1.3 Место дисциплины (модуля) в структуре образовательной программы

Дисциплина «Коллективная разработка программных продуктов» относится к обязательной части Блока 1 "Дисциплины (модули)" учебного плана.

Дисциплины, необходимые для изучения дисциплины «Корпоративные информационные системы»: «Информатика», «Компьютерная геометрия и графика», «Информационные технологии», «Проектирование информационных систем».

1.4 Перечень планируемых результатов обучения по дисциплине (модулю), соотнесенных с планируемыми результатами освоения образовательной программы

Изучение данной учебной дисциплины направлено на формирование у обучающихся *универсальных, общепрофессиональных* компетенций (*УК/ОПК*)

№ п.п.	Код и наименование компетенции	Индикаторы достижения компетенции		
		знает	**умеет**	**владеет**
3.	ПК-6 Способность к проектированию информационных ресурсов	основные этапы, методологию и средства проектирования информационных систем	самостоятельно составлять проект информационной системы; проводить предпроектное обследование; проводить выбор исходных данных для проектирования информационной системы	методами и средствами проектирования, модернизации и модификации информационных систем

2. Структура и содержание дисциплины

2.1 Распределение трудоёмкости дисциплины по видам работ

Общая трудоёмкость дисциплины составляет 3 зач. ед. (108 часов), их распределение по видам работ представлено в таблице

(для студентов ОФО)

Вид учебной работы	Всего часов	Семестры (часы)			
		7			

Контактная работа, в том числе:		**46,2**	**46,2**			
Аудиторные занятия (всего):						
Занятия лекционного типа		14	14			
Лабораторные занятия		26	26			
Иная контактная работа:						
Контроль самостоятельной работы (КСР)		6	6			
Промежуточная аттестация (ИКР)		0,2	0,2			
Самостоятельная работа, в том числе:		**61,8**	**61,8**			
Проработка учебного (теоретического) материала		30,8	30,8			
Подготовка к текущему контролю		31	31			
Контроль:		-	-			
Подготовка к экзамену			-			
Общая трудоемкость	**час.**	**108**	**108**			
	в том числе контактная работа	**46,2**	**46,2**			
	зач. ед	**3**	**3**			

2.2 Структура дисциплины

Распределение видов учебной работы и их трудоемкости по разделам дисциплины.

Разделы (темы) дисциплины, изучаемые в 7 семестре *(очная форма)*

№	Наименование разделов (тем)	Количество часов				
		Всего	Аудиторная работа			Внеаудиторная работа
			Л	ПЗ	ЛР	СРС
1	2	3	4	5	6	7
1.	Стандарты и технологии управления жизненным циклом ИТ-проектов	19	5	-	2	12
2.	Календарное планирование ИТ-проектов	19	5	-	2	12
3.	Управление ресурсами ИТ-проектов	19	5	-	2	12
4.	Управление рисками ИТ-проектов	21	5	-	4	12
5.	Управление версиями и документооборотом ИТ-проектов	23,8	6	-	4	13,8
	ИТОГО по разделам дисциплины	101,8	26		14	61,8
	Контроль самостоятельной работы (КСР)	6	6			
	Промежуточная аттестация (ИКР)	0,2	0,2			
	Общая трудоемкость по дисциплине	108	108			

Примечание: Л – лекции, ПЗ – практические занятия / семинары, ЛР – лабораторные занятия, СРС – самостоятельная работа студента

2.3 Содержание разделов (тем) дисциплины

2.3.1 Занятия лекционного типа

№	Наименование раздела (темы)	Содержание раздела (темы)	Форма текущего контроля
1	2	3	4
1.	Введение	Введение в предмет • Проблемы коллективной разработки • Инструменты коллективной разработки программ	Проработка учебного (теоретического) материала, ЛР

		• Главный вопрос конфигурационного управления	
2.	Системы контроля версий	Классические системы контроля версий (СКВ). Базовые принципы работы с системами контроля версий. • История появления, файловые СКВ, CVS • Основные принципы работы (команды checkout, commit, update) • Дальнейшее развитие СКВ. Subversion	Подготовка к текущему контролю, ЛР
3.	Основные инструменты коллективной разработки	Багтрекеры • Необходимость багтрекеров • История появления багтрекеров, JIRA • Жизненный цикл бага • Современные багтрекеры • Вариации жизненного цикла задачи	Проработка учебного (теоретического) материала, ЛР
4.	Примеры использования инструментов коллективной разработки	Техническая инфраструктура open source проектов. • Особенности инфраструктуры open source проектов • Взаимодействие между участниками проекта (IRC, листы рассылки) • Хранение кода (sourceforgfe, github, google code) • Распространение знаний • Нумерация версий.	Подготовка к текущему контролю, ЛР
5.	Вспомогательные инструменты коллективной	Инструменты статического анализа кода • Возможности и ограничения статических анализаторов кода • Статический анализ кода на C на примере linq • Статический анализ кода на Java на примере PMD • Метрики программного обеспечения	Проработка учебного (теоретического) материала, ЛР

2.3.2 Занятия семинарского типа

Не предусмотрены.

2.3.3 Лабораторные занятия

№	Наименование лабораторных работ	Форма текущего контроля
1	3	4
1.	Лабораторная работа 1. Техническое задание на проектирование программы	Отчет по лабораторной работе

2.	Лабораторная работа 2. Стадия разработки программного обеспечения «Эскизный проект»	Отчет по лабораторной работе
3.	Лабораторная работа 3. . Стадия разработки программного обеспечения «Технический проект»	Отчет по лабораторной работе
4.	Лабораторная работа 4. Использование объектно-ориентированного программирования (ООП) для создания качественного программного обеспечения.	Отчет по лабораторной работе
5.	Лабораторная работа 5. Использование визуальных компонент для создания качественных программ	Отчет по лабораторной работе
6.	Лабораторная работа 6. Средства отладки программ в объектно-ориентированном программировании	Отчет по лабораторной работе

2.3.4 Примерная тематика курсовых работ (проектов)
Не предусмотрены.

2.4 Перечень учебно-методического обеспечения для самостоятельной работы обучающихся по дисциплине (модулю)

№	Вид СРС	Перечень учебно-методического обеспечения дисциплины по выполнению самостоятельной работы
1	2	3
1	Проработка учебного (теоретического) материала	Методические указания по организации аудиторной и самостоятельной работ, утвержденные кафедрой теоретической физики и компьютерных технологий, протокол № 9 от «14» марта 2017г
2	Подготовка к текущему контролю	Методические рекомендации для подготовки к практическим, семинарским и лабораторным занятиям, утвержденные кафедрой теоретической физики и компьютерных технологий, протокол № 9 от «14» марта 2017г.

Учебно-методические материалы для самостоятельной работы обучающихся из числа инвалидов и лиц с ограниченными возможностями здоровья (ОВЗ) предоставляются в формах, адаптированных к ограничениям их здоровья и восприятия информации:

Для лиц с нарушениями зрения:
– в печатной форме увеличенным шрифтом,
– в форме электронного документа.
Для лиц с нарушениями слуха:
– в печатной форме,
– в форме электронного документа.
Для лиц с нарушениями опорно-двигательного аппарата:
– в печатной форме,
– в форме электронного документа.

Данный перечень может быть конкретизирован в зависимости от контингента обучающихся.

3. Образовательные технологии

При изучении дисциплины проводятся следующие виды учебных занятий и работ: лекции, лабораторные работы, опрос, тестирование, консультации с преподавателем, самостоятельная работа студентов (изучение теоретического материала, подготовка к практическими занятиям, подготовка к тестированию и зачету).

Для проведения части лекционных занятий используются мультимедийные средства воспроизведения активного содержимого (занятия в интерактивной форме), позволяющего студенту воспринимать особенности изучаемой дисциплины, играющие решающую роль в понимании и восприятии, а также в формировании профессиональных компетенций. По ряду тем дисциплины лекций проходит в классическом стиле.

При проведении лабораторных работ может использоваться доска, для расчетов и анализа данных могут применяться дополнительные справочные материалы. Предварительно изучая рекомендованную литературу студенты готовятся к практическому занятию. На практических занятиях учебная группа делится на подгруппы по 5-7 человека. В ходе проверки промежуточных результатов, поиска и исправления ошибок, осуществляется интерактивное взаимодействие всех участников занятия.

В соответствии с требованиями ФГОС ВО по направлению подготовки 09.03.02 Информационные системы и технологии реализация компетентностного подхода должна предусматривать широкое использование в учебном процессе активных и интерактивных форм проведения занятий в сочетании с внеаудиторной работой с целью формирования и развития профессиональных навыков студентов.

В преподавании курса используются современные образовательные технологии:

1. Дискуссия;
2. Анализ ситуаций профессиональной деятельности;
3. Метод проектов;
4. Метод малых групп;
5. Интерактивная лекция (лекция – дискуссия, лекция с разбором конкретных ситуаций, лекция с мультимедийной системой и активным вовлечением студентов в учебный процесс). Удельный вес занятий, проводимых в интерактивных формах, определяется главной целью ООП, особенностью контингента обучающихся и содержанием конкретных дисциплин, и в целом в учебном процессе должен составлять не менее 10 процентов от общего объема аудиторных занятий.

Для лиц с ограниченными возможностями здоровья предусмотрена организация консультаций с использованием электронной почты.

7. Оценочные и методические материалы

4.1 Оценочные средства для текущего контроля успеваемости и промежуточной аттестации

Оценочные средства предназначены для контроля и оценки образовательных достижений обучающихся, освоивших программу учебной дисциплины «Коллективная разработка программных продуктов».

Оценочные средства включает контрольные материалы для проведения **текущего контроля** в форме выполнения лабораторных работ и **промежуточной аттестации** в форме защиты лабораторных работ.

Оценочные средства для инвалидов и лиц с ограниченными возможностями здоровья выбираются с учетом их индивидуальных психофизических особенностей.

– при необходимости инвалидам и лицам с ограниченными возможностями здоровья предоставляется дополнительное время для подготовки ответа на экзамене;

– при проведении процедуры оценивания результатов обучения инвалидов и лиц с ограниченными возможностями здоровья предусматривается использование технических средств, необходимых им в связи с их индивидуальными особенностями;

– при необходимости для обучающихся с ограниченными возможностями здоровья и инвалидов процедура оценивания результатов обучения по дисциплине может проводиться в несколько этапов.

Процедура оценивания результатов обучения инвалидов и лиц с ограниченными возможностями здоровья по дисциплине (модулю) предусматривает предоставление информации в формах, адаптированных к ограничениям их здоровья и восприятия информации:

Для лиц с нарушениями зрения:

– в печатной форме увеличенным шрифтом,

– в форме электронного документа.

Для лиц с нарушениями слуха:

– в печатной форме,

– в форме электронного документа.

Для лиц с нарушениями опорно-двигательного аппарата:

– в печатной форме,

– в форме электронного документа.

Данный перечень может быть конкретизирован в зависимости от контингента обучающихся.

Структура оценочных средств для текущей и промежуточной аттестации

№ п/п	Контролируемые разделы (темы) дисциплины*	Код контролируемой компетенции (или ее части)	Наименование оценочного средства	
			Текущий контроль	Промежуточная аттестация
1	Введение	ПК-6 (знать)	ЛР	Защита ЛР
2	Системы контроля версий	ПК-6 (уметь)	ЛР	Защита ЛР
3	Основные инструменты коллектив ной разработки	ПК-6 (владеть)	ЛР	Защита ЛР
4	Примеры использования инструментов коллектив ной разработки	ПК-6 (знать)	ЛР	Защита ЛР
5	Вспомогательные инструменты коллектив ной	ПК-6 (уметь)	ЛР	Защита ЛР

Показатели, критерии и шкала оценки сформированных компетенций

Код и наименование компетенции	Соответствие уровней освоения компетенции планируемым результатам обучения и критериям их оценивания		
	Пороговый	Базовый	Продвинутый
	Оценка		
	Удовлетворительно/ зачтено	Хорошо/ зачтено	Отлично/ зачтено
ПК-6	**Знает не в полном объеме** основные этапы,	**Знает в достаточном объеме** основные этапы,	**Знает в полном объеме** основные

	методологию и средства проектирования информационных систем **Умеет в требуемых пределах** самостоятельно составлять проект информационной системы; проводить предпроектное обследование; проводить выбор исходных данных для проектирования информационной системы **Владеет некоторыми** методами и средствами проектирования, модернизации и модификации информационных систем	методологию и средства проектирования информационных систем **Умеет квалифицированно** самостоятельно составлять проект информационной системы; проводить предпроектное обследование; проводить выбор исходных данных для проектирования информационной системы **Владеет свободно и большинством** методами и средствами проектирования, модернизации и модификации информационных систем	этапы, методологию и средства проектирования информационных систем **Умеет на высоком научном уровне** самостоятельно составлять проект информационной системы; проводить предпроектное обследование; проводить выбор исходных данных для проектирования информационной системы **Владеет свободно и всеми** методами и средствами проектирования, модернизации и модификации информационных систем

Типовые контрольные задания или иные материалы, необходимые для оценки знаний, умений, навыков и (или) опыта деятельности, характеризующих этапы формирования компетенций в процессе освоения образовательной программы

Лабораторная работа 1

Техническое задание на проектирование программы

Цель работы: ознакомиться с правилами написания технического задания.

ГОСТ 2.610-2006. Настоящий стандарт устанавливает порядок построения и оформления технического задания на разработку программы или программного изделия для вычислительных машин, комплексов и систем независимо от их назначения и области применения.

Общие сведения

1. Требования к оформлению

1.1. Техническое задание оформляют в соответствии с ГОСТ 2.610-2006 на листах формата А4 и А3, как правило, без заполнения полей листа. Номера листов (страниц) проставляют в верхней части листа над текстом.

1.2. Лист утверждения и титульный лист оформляют в соответствии с ГОСТ 2.610-2006. Информационную часть (аннотацию и содержание), лист регистрации изменений допускается в документ не включать.

1.3. Для внесения изменений и дополнений в техническое задание на последующих стадиях разработки программы или программного изделия к нему выпускают дополнение. Согласование и утверждение дополнения к техническому заданию проводят в том же порядке, который установлен для технического задания.

1.4. Техническое задание должно содержать следующие разделы:

- название программы и область применения;
- основание для разработки;
- назначение разработки;
- технические требования к программе или программному изделию;
- технико–экономические показатели;

- стадии и этапы разработки:
- порядок контроля и приемки;
- приложения.

В зависимости от особенностей программы или программного изделия допускается уточнять содержание разделов, вводить новые разделы или объединять отдельные из них.

2. Содержание разделов

2.1. В разделе «Наименование и область применения» указывают наименование, краткую характеристику области применения программы или программного изделия и объекта, в котором используют программу или программное изделие.

2.2. В разделе «Основание для разработки» должны быть указаны:

- документ (документы), на основании которых ведется разработка;
- организация, утвердившая этот документ, и дата его утверждения;
- наименование и (или) условное обозначение темы разработки.

2.3. В разделе «Назначение разработки» должно быть указано функциональное и эксплуатационное назначение программы или программного изделия.

2.4. Раздел «Технические требования к программе или программному изделию» должен содержать следующие подразделы:

- требования к функциональным характеристикам;
- требования к надежности;
- условия эксплуатации;
- требования к составу и параметрам технических средств;
- требования к информационной и программной совместимости;
- требования к маркировке и упаковке;
- требования к транспортированию и хранению;
- специальные требования.

2.4.1. В подразделе «Требования к функциональным характеристикам» должны быть указаны требования к составу выполняемых функций, организации входных и выходных данных, временным характеристикам и т.п.

2.4.2. В подразделе «Требования к надежности» должны быть указаны требования к обеспечению надежного функционирования (обеспечение устойчивого функционирования, контроль входной и выходной информации, время восстановления после отказа и т.п.).

2.4.3. В подразделе «Условия эксплуатации» должны быть указаны условия эксплуатации (температура окружающего воздуха, относительная влажность для выбранных типов носителей данных), при которых должны обеспечиваться заданные характеристики, а также вид обслуживания, необходимое количество и квалификация персонала.

2.4.4. В подразделе «Требования к составу и параметрам технических средств» указывают необходимый состав технических средств с указанием их технических характеристик.

2.4.5. В подразделе «Требования к информационной и программной совместимости» должны быть указаны требования к информационным структурам на входе и выходе и методам решения, исходным кодам, языкам программирования. При необходимости должна обеспечиваться защита информации и программ.

2.4.6. В подразделе «Требования к маркировке и упаковке» в общем случае указывают требования к маркировке программного изделия, варианты и способы упаковки.

2.4.7. В подразделе «Требования к транспортированию и хранению» должны быть указаны для программного изделия условия транспортирования, места хранения, условия хранения, условия складирования, сроки хранения в различных условиях.

2.5. В разделе «Технико–экономические показатели» должны быть указаны: ориентировочная экономическая эффективность предполагаемая годовая потребность, экономические преимущества разработки по сравнению с лучшими отечественными и зарубежными образцами или аналогами.

2.6. В разделе «Стадии и этапы разработки* устанавливают необходимые стадии разработки, этапы и содержание работ (перечень программных документов, которые должны быть разработаны, согласованы и утверждены), а также, как правило, сроки разработки и определяют исполнителей.

2.7. В разделе «Порядок контроля и приемки» должны быть указаны виды испытаний и общие требования к приемке работы.

2.8. В приложениях к техническому заданию при необходимости приводят:

- перечень научно-исследовательских и других работ, обосновывающих разработку;
- схемы алгоритмов, таблицы, описания, обоснования, расчеты и другие документы, которые могут быть использованы при разработке;
- другие источники разработки.

Пример выполнения задания

1. Введение

Работа выполняется в рамках проекта «Автоматизированная система оперативно-диспетчерского управления электро-, теплоснабжением корпусов института».

2. Основание для разработки

2.1. Основанием для данной работы служит договор № ___ от _____ 20___ г.

2.2. Наименование работы:

2.3. «Модуль автоматизированной системы оперативно-диспетчерското управления теплоснабжением корпусов института».

2.4. Исполнители: ОАО «Лаборатория создания программного обеспечения.

2.5. Соисполнители: нет.

3. Назначение разработки

Создание модуля для контроля и оперативной корректировки состояния основных параметров теплообеспечения корпусов Московского института.

4.Технические требования

4.1. Требования к функциональным характеристикам

4.1.2. Состав выполняемых функций. Разрабатываемое ПО должно обеспечивать:

– сбор и анализ информации о расходовании тепла, горячей и холодной воды по данным теплосчетчиков SA-94 на всех тепловых выходах;

– сбор и анализ информации с устройств управления системами воздушного отопления и кондиционирования типа РТ1 и РТ2 (разработки кафедры СММЭ и ТЦ);

– предварительный анализ информации на предмет нахождения параметров в допустимых пределах и сигнализирование при выходе параметров за пределы допуска;

– выдачу рекомендаций по дальнейшей работе;

– отображение текущего состояния по набору параметров – циклически постоянно (режим работы круглосуточный), при сохранении периодичности контроля прочих параметров;

– визуализацию информации по расходу теплоносителя;

– текущую, аналогично показаниям счетчиков;

– с накоплением за прошедшие сутки, неделю, месяц – в виде почасового графика для информации за сутки и неделю;

– суточный расход – для информации за месяц.

Для устройств управления приточной вентиляцией текущая информация должна содержать номер приточной системы и все параметры, выдаваемые на собственный индикатор.

По отдельному запросу осуществляются внутренние настройки.

В конце отчетного периода система должна архивировать данные.

4.1.2. Организация входных и выходных данных

Исходные данные в систему поступают в виде значений с датчиков, установленных в помещениях института. Эти значения отображаются на компьютере диспетчера. После анализа поступившей информации оператор диспетчерского пункта устанавливает необходимые параметры для устройств, регулирующих отопление и вентиляцию в помещениях. Возможна также автоматическая установка некоторых параметров для устройств регулирования.

Основной режим использования системы – ежедневная работа.

4.2. Требования к надежности

Для обеспечения надежности необходимо проверять корректность получаемых данных с датчиков.

4.3. Условия эксплуатации и требования к составу и параметрам технических средств

Для работы системы должен быть выделен ответственный оператор. Требования к составу и параметрам технических средств уточняются на этапе эскизного проектирования системы.

4.4. Требования к информационной и программной совместимости

Программа должна работать на платформах Windows.

4.5. Требования к транспортировке и хранению

Программа поставляется на лазерном носителе информации. Программная документация поставляется в электронном и печатном виде.

4.6. Специальные требования. Программное обеспечение должно иметь дружественный интерфейс, рассчитанный на пользователя (в плане компьютерной грамотности) средней квалификации. Ввиду объемности проекта задачи предполагается решать поэтапно, при этом модули ПО, созданные в разное время, должны предполагать возможность наращивания системы и быть совместимы друг с другом, поэтому документация на принятое эксплуатационное ПО должна содержать полную информацию, необходимую для работы программистов. Язык программирования выбирает исполнитель, он должен обеспечивать возможность интеграции программного обеспечения с некоторыми видами периферийного оборудования (например, счетчик SA-94 и т.п.).

5. Требования к программной документации

Основными документами, регламентирующими разработку будущих программ, должны быть документы Единой системы программной документации (ЕСПД); руководство пользователя, руководство администратора, описание применения.

6. Технико-экономические показатели

Эффективность системы определяется удобством использования системы для контроля и управления основными параметрами теплообеспечения помещений Московского института, а также экономической выгодой, полученной от внедрения аппаратно–программного комплекса.

7. Порядок контроля и приемки

После передачи исполнителем отдельного функционального модуля программы заказчику последний имеет право тестировать модуль в течение семи дней. После тестирования заказчик должен принять работу по данному этапу или в письменном виде изложить причину отказа от принятия. В случае обоснованного отказа исполнитель обязуется доработать модуль.

Стадия разработки программного обеспечения «Эскизный проект»

Цель работы: научиться создавать формальные модели и на их основе определять спецификации разрабатываемого программного обеспечения.

Подготовка к лабораторной работе

Ознакомиться с лекционным материалом по теме «Этапы разработки программного обеспечения. Анализ требований и определение спецификаций программного обеспечения» учебной дисциплины «Технология разработки программного обеспечения».

Теоретическая часть

Разработка спецификаций программного обеспечения начинается с анализа требований к нему. В результате анализа получают спецификации разрабатываемого программного обеспечения, строят общую модель его взаимодействия с пользователем или другими программами и конкретизируют его основные функции.

При структурном подходе к программированию на этапе анализа и определения спецификаций разрабатывают три типа моделей: модели функций, модели данных и модели потоков данных. Поскольку разные модели описывают проектируемое программное обеспечение с разных сторон, рекомендуется использовать сразу несколько моделей, разрабатываемых в виде диаграмм, и пояснять их текстовыми описаниями, словарями и т. п.

Структурный анализ предполагает использование следующих видов моделей:

- диаграмм потоков данных (DFD – Data Flow Diagrams), описывающих взаимодействие источников и потребителей информации через процессы, которые должны быть реализованы в системе;
- диаграмм «сущность – связь» (ERD – Entity-Relationship Diagrams), описывающих базы данных разрабатываемой системы;
- диаграмм переходов состояний (STD – State Transition Diagrams), характеризующих поведение системы во времени;
- функциональных диаграмм (методика SADT);
- спецификаций процессов;
- словаря терминов.

Спецификации процессов обычно представляют в виде краткого текстового описания, схем алгоритмов, псевдокодов, Flow-форм или диаграмм Насси – Шнейдермана.

Словарь терминов представляет собой краткое описание основных понятий, используемых

при составлении спецификаций. Он должен включать определение основных понятий предметной области, описание структур элементов данных, их типов и форматов, а также всех сокращений и условных обозначений.

С помощью диаграмм переходов состояний можно моделировать последующее функционирование системы на основе ее предыдущего и текущего функционирования. Моделируемая система в любой заданный момент времени находится точно в одном из конечного множества состояний. С течением времени она может изменить свое состояние, при этом переходы между состояниями должны быть точно определены.

Функциональные диаграммы отражают взаимосвязи функций разрабатываемого программного обеспечения.

Они создаются на ранних этапах проектирования систем, для того чтобы помочь проектировщику выявить основные функции и составные части проектируемой системы и, по возможности, обнаружить и устранить существенные ошибки. Для создания функциональных диаграмм предлагается использовать методологию SADT. Диаграммы потоков данных.

Для описания потоков информации в системе применяются диаграммы потоков данных (DFD – Data flow diagrams). DFD позволяет описать требуемое поведение системы в виде совокупности процессов, взаимодействующих посредством связывающих их потоков данных. DFD показывает, как каждый из процессов преобразует свои входные потоки данных в выходные потоки данных и как процессы взаимодействуют между собой.

Диаграмма «сущность – связь» – инструмент разработки моделей данных, обеспечивающий стандартный способ определения данных и отношений между ними.

Она включает сущности и взаимосвязи, отражающие основные бизнес-правила предметной области. Такая диаграмма не слишком детализирована, в нее включаются основные сущности и связи между ними, которые удовлетворяют требованиям, предъявляемым к ИС.

Порядок выполнения работы

1. На основе технического задания из лабораторной работы №1 выполнить анализ функциональных и эксплуатационных требований к программному продукту.
2. Определить основные технические решения (выбор языка программирования, структура программного продукта, состав функций ПП, режимы функционирования) и занести результаты в документ, называемый «Эскизным проектом».
3. Определить диаграммы потоков данных для решаемой задачи.
4. Определить диаграммы «сущность – связь», если программный продукт содержит базу данных.
5. Определить функциональные диаграммы.
6. Определить диаграммы переходов состояний.
7. Определить спецификации процессов.
8. Добавить словарь терминов.
9. Оформить результаты, используя MS Office в виде эскизного проекта.
10. Сдать и защитить работу.

Защита отчета по лабораторной работе

Отчет по лабораторной работе должен содержать:

1. Постановку задачи.
2. Документ «Эскизный проект», включающий:

- выбор метода решения и языка программирования;
- спецификации процессов;
- все полученные диаграммы;
- словарь терминов.

Защита отчета по лабораторной работе заключается в предъявлении преподавателю полученных результатов (на экране монитора), демонстрации полученных навыков и ответах на вопросы преподавателя.

Допустим, на предыдущих стадиях разработки было составлено и утверждено техническое задание на создание информационной системы СУБД «Пенсионный Фонд», разработанное на основании ГОСТ 34.602 – 89 на написание ТЗ на автоматизированные системы управления от 01.01.1990 г.

Пояснительная записка к эскизному проекту

Данный документ является эскизным проектом на создание Системы Управления Базой Данных для Библиотечного Фонда Российской Федерации (СУБД «Библиотека»).

Перечень организаций, участвующих в разработке системы, сроки и стадии разработки, а также ее цели и назначение указаны в техническом задании на создание информационной системы.

Лабораторная работа 3

Стадия разработки программного обеспечения «Технический проект»

Цель работы: изучить вопросы проектирования программного обеспечения.

Подготовка к лабораторной работе

Ознакомиться с лекционным материалом по теме «Этапы разработки программного обеспечения. Проектирование программного обеспечения» учебной дисциплины «Технология разработки программного обеспечения».

Теоретическая часть

Технический проект – образ намеченного к созданию объекта, представленный в виде его описания, схем, чертежей, расчетов, обоснований, числовых показателей.

Цель технического проекта – определение основных методов, используемых при создании информационной системы, и окончательное определение ее сметной стоимости.

Техническое проектирование подсистем осуществляется в соответствии с утвержденным техническим заданием.

Технический проект программной системы подробно описывает:

- выполняемые функции и варианты их использования;
- соответствующие им документы;
- структуры обрабатываемых баз данных;
- взаимосвязи данных;
- алгоритмы их обработки.

Технический проект должен включать данные об объемах и интенсивности потоков обрабатываемой информации, количестве пользователей программной системы, характеристиках оборудования и программного обеспечения, взаимодействующего с проектируемым программным продуктом.

При разработке технического проекта оформляются:

- ведомость технического проекта. Общая информация по проекту;
- пояснительная записка к техническому проекту. Вводная информация, позволяющая ее потребителю быстро освоить данные по конкретному проекту;
- описание систем классификации и кодирования;
- перечень входных данных (документов). Перечень информации, которая используется как входящий поток и служит источником накопления;
- перечень выходных данных (документов). Перечень информации, которая используется для анализа накопленных данных;
- описание используемого программного обеспечения.
- перечень программного обеспечения и СУБД, которые планируется использовать для создания информационной системы;
- описание используемых технических средств. Перечень аппаратных средств, на которых планируется работа проектируемого программного продукта;
- проектная оценка надежности системы. Экспертная оценка надежности с выявлением наиболее благополучных участков программной системы и ее узких мест;
- ведомость оборудования и материалов. Перечень оборудования и материалов, которые потребуются в ходе реализации проекта.

Структурная схема

Структурной называют схему, отражающую состав и взаимодействие по управлению частями разрабатываемого программного обеспечения.

Структурная схема определяется архитектурой разрабатываемого ПО:

- структуры обрабатываемых баз данных;
- взаимосвязи данных;
- алгоритмы их обработки.

Функциональная схема

Функциональная схема – это схема взаимодействия компонентов программного обеспечения с описанием информационных потоков, состава данных в потоках и указанием используемых файлов и устройств.

Порядок выполнения работы

1. На основе технического задания из лабораторной работы №1 и спецификаций из лабораторной работы №2 разработать уточненные алгоритмы программ, составляющих заданный программный модуль.
2. На основе уточненных и доработанных алгоритмов разработать структурную схему программного продукта.
3. Разработать функциональную схему программного продукта.
4. Оформить результаты, используя MS Office в виде технического проекта.
5. Сдать и защитить работу.

Лабораторная работа 4

Использование объектно-ориентированного программирования (ООП) для создания качественного программного обеспечения

Цель работы: познакомиться с принципами объектно-ориентированного программирования в среде Delphi. Написать и отладить программу линейного алгоритма, в которой присутствовали бы некоторые критерии и примитивы качественного программного обеспечения.

Общие сведения

Класс – абстрактный тип данных, включающий свойства объекта (поля) и методы. Класс позволяет упростить процесс программирования, так как человеку проще представлять любой объект из реальности, обладающий некоторыми характеристиками (свойствами) и действиями, которые может совершать объект или которые можно совершать над ним.

Класс – это тип данных. Объект класса – переменная типа «класс». Из определения класса следует первое свойство ООП – инкапсуляция. Инкапсуляция данных означает, что они являются не глобальными – доступными всей программе, а локальными – доступными только малой ее части. Инкапсуляция автоматически подразумевает защиту данных. Для этого в структуре class используется спецификатор раздела private, содержащий данные и методы, доступные только для самого класса. Если данные и методы содержатся в разделе public, они доступны извне класса. Раздел protected содержит данные и методы, доступные из класса и любого его *производного класса.*

Интегрированная среда разработки Delphi

Среда Delphi визуально реализуется в виде нескольких одновременно раскрытых на экране монитора окон. Количество, расположение, размер и вид окон может меняться в зависимости от текущих нужд, что значительно повышает производительность работы. При запуске Delphi можно увидеть на экране картинку (рис. 4.1).

Главное окно всегда присутствует на экране и предназначено для управления процессом создания программы. Основное меню содержит все необходимые средства для управления проектом. Пиктограммы облегчают доступ к наиболее часто применяемым командам основного меню. Через меню компонентов осуществляется доступ к набору стандартных сервисных программ среды Delphi, которые описывают некоторый визуальный элемент (компонент), помещенный программистом в окно формы. Каждый компонент имеет определенный набор свойств (параметров), которые программист может задавать, например цвет, заголовок окна, надпись на кнопке, размер и тип шрифта и др.

Окно инспектора объектов (вызывается с помощью клавиши F11) предназначено для изменения свойств выбранных компонентов и состоит из двух страниц. Страница Properties (Свойства) предназначена для изменения необходимых свойств компонента, страница Events (События) – для определения реакции компонента на то или иное событие (например, нажатие определенной клавиши или щелчок мышью по кнопке).

Окно формы представляет собой проект Windows–окна программы. В это окно в процессе написания программы помещаются необходимые компоненты. Причем при выполнении

программы помещенные компоненты будут иметь тот же вид, что и на этапе проектирования.

Окно текста программы предназначено для просмотра, написания и корректирования текста программы. В системе Delphi используется язык программирования Object Pascal. При первоначальной загрузке в окне текста программы находится текст, содержащий минимальный набор операторов для нормального функционирования пустой формы в качестве Windows-окна. При помещении некоторого компонента в окно формы текст программы автоматически дополняется описанием необходимых для его работы библиотек стандартных программ (раздел uses) и типов переменных (раздел type).

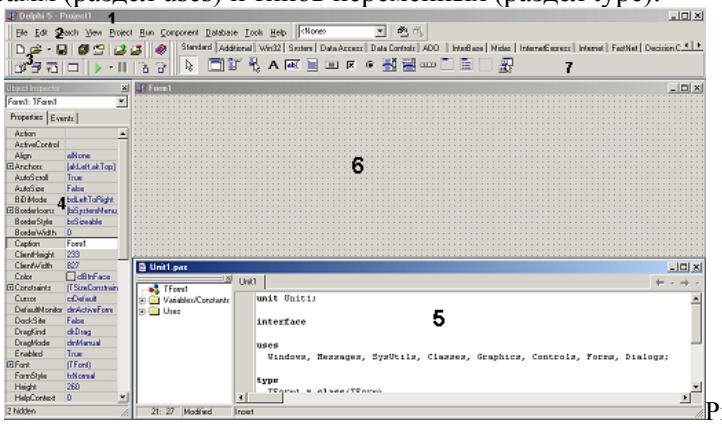

Рис. 4.1. Внешний вид окна Delphi.
1 – главное окно; 2 – основное меню; 3 – пиктограммы основного меню; 4 – окно инспектора объектов; 5 – окно текста программы; 6 – окно пустой формы, 7 – меню компонентов

Программа в среде Delphi составляется как описание алгоритмов, которые необходимо выполнить, если возникает определенное событие, связанное с формой (например, щелчок мыши на кнопке – событие OnClick, создание формы – OnCreate). Для каждого обрабатываемого в форме события с помощью страницы Events инспектора объектов в тексте программы организуется процедура (procedure), в которой записывается на языке Object Pascal требуемый алгоритм.

Переключение между окном формы и окном текста программы осуществляется с помощью клавиши F12.

Меню и команды Delphi

Для того чтобы выдать команду в среде Delphi, можно воспользоваться тремя основными способами:

- с помощью меню;
- с помощью полоски SpeedBar (инструментальной линейки);
- с помощью SpeedMenu (одного из локальных меню, которое активизируется при нажатии правой кнопки мыши).

Меню File

Команды выпадающего меню File можно использовать для работы как с проектами, так и файлами исходного кода.

К командам, работающим с проектами, относятся New, New Application, Open, Reopen, Save Project As, Save All, Close All, Add to Project и Remove from Project. С файлами исходного кода работают команды New, New Form, New Data Module, Open, Reopen, Save As, Save, Close и Print. Основной командой является File/New, которую можно использовать для вызова экспертов, для начала работы с новым приложением, наследования формы из уже существующей и т.д. Для того чтобы открыть проект или файл исходного кода, с которыми вы работали последний раз, используйте команду File/Reopen.

Меню Edit

Стандартные возможности меню Edit применимы как к тексту, так и к компонентам формы. Можно копировать и вставлять тот или иной текст в редакторе, копировать и вставлять компоненты в одной форме или из одной формы в другую. Также можно копировать и вставлять компоненты в другое групповое окно той же формы, например в панель или блок группы; копировать компоненты из формы в редактор, и наоборот. Delphi помещает компоненты в буфер обмена, преобразуя их в текстовое описание. Можно соответствующим образом отредактировать этот текст, а затем вставить его обратно в форму в виде нового компонента. Можно выбрать несколько компонентов и скопировать их как в другую форму, так и в текстовый редактор. Это может пригодиться, когда вам придется работать с рядом схожих компонентов. Вы сможете

скопировать один компонент в редактор, размножить его нужное число раз, а затем вставить назад в форму целую группу.

Меню Search

Если вы выберете команду Incremental Search, то вместо того чтобы показать диалоговое окно, где вводится образец для поиска, Delphi переходит в редактор. Когда вы введете первую букву, редактор перейдет к первому слову, которое начинается с этой буквы. Продолжайте набор букв и, курсор будет последовательно переходить к словам, в начале которых будут стоять введенные символы. Эта команда очень эффективна и чрезвычайно быстра. Команда Browse Symbol вызывает Object Browser – инструмент, который можно использовать для просмотра многих деталей при исследовании откомпилированной программы.

Меню View

Большинство команд меню View применяются для отображения какого–либо окна среды Delphi, например Project Manager, Breakpoints List или Components List. Эти окна не связаны друг с другом. Команда Toggle Form/Unit используется для перехода от формы, над которой вы работаете, к ее исходному коду, и обратно. Команда New edit window создает дубликат окна редактирования и его содержимого. В Delphi это единственный способ просмотреть два файла рядом друг с другом, поскольку редактор для показа нескольких загруженных файлов использует ярлыки. После дублирования окна редактирования могут содержать разные файлы. Последние две команды меню View можно использовать для удаления с экрана полоски SpeedBar и палитры Components, хотя при этом среда Delphi становится менее удобнее для пользователя. Команда Build All заставляет Delphi откомпилировать каждый исходный файл проекта, даже если после последней трансляции он не был изменен. Для проверки написанного кода без создания программы можно использовать команду Syntax Check. Команда Information дает некоторые подробности о последней выполненной трансляции. Команда Options применяется для установки опций проекта: опций компилятора и редактора связей, опций объекта приложения и т.д.

Меню Run

Меню Run можно назвать Debug (отладка), так как большинство команд в нем относится к отладке, включая саму команду Run. Программа, запускаемая внутри среды Delphi, выполняется в ее интегрированном отладчике (если не отключена соответствующая опция). Для быстрого запуска приложения используется клавиша F9. Остальные команды применяются в процессе отладки для пошагового выполнения программы, установки точек прерывания, просмотра значений переменных и объектов и т.п.

Меню Component

Команды меню Component можно использовать для написания компонентов, добавления их в библиотеку, а также для конфигурирования библиотеки или палитры компонентов.

Меню Tools

Меню Tools содержит список нескольких внешних программ и инструментальных средств. Команда Tools позволяет сконфигурировать это выпадающее меню и добавить в него новые внешние средства. Меню Tools также включает команду для настройки репозитория и команду Options, которая конфигурирует всю среду разработки Delphi.

Работа с формами

Проектирование форм – ядро визуальной разработки в среде Delphi. Каждый помещаемый в форму компонент или любое задаваемое свойство сохраняется в файле, описывающем форму (DFM–файл), а также оказывает некоторое влияние на исходный текст, связанный с формой (PAS–файл). Можно начать новый пустой проект, создав пустую форму или начать с существующей формы (используя различные доступные шаблоны), или добавить в проект новые формы. Проект (приложение) может иметь любое число форм.

При работе с формой можно обрабатывать ее свойства, свойства одного из ее компонентов или нескольких компонентов одновременно. Для того чтобы выбрать форму или компонент, можно просто щелкнуть по нему мышью или воспользоваться Object Selector (комбинированный список в Object Inspector), где всегда отображены имя и тип выбранного элемента. Для выбора нескольких компонентов можно или нажать клавишу Shift и щелкать по компонентам левой кнопкой мыши, или отбуксировать в форме рамку выбора.

SpeedMenu формы содержит ряд полезных команд. Для изменения относительного расположения компонентов одного вида можно использовать команды Bring To Front и Send To Back. Командой Revert To Inherited можно воспользоваться, чтобы в унаследованной форме установить те значения свойств выбранного компонента, которые были у них в родительской

форме. При выборе сразу нескольких компонентов вы можете выровнять их или изменить их размеры.

С помощью SpeedMenu можно также открыть два диалоговых окна, в которых устанавливается порядок обхода визуальных управляющих элементов и порядок создания невизуальных управляющих элементов. Команда Add To Repository добавляет текущую форму в список форм, доступных для использования в других проектах.

Для установки положения компонента кроме применения мыши существуют еще два способа:

- установка значений для свойств Top и Left;
- использование клавиш курсора при нажатой клавише Ctrl.

Метод Ctrl+клавиша курсора особенно удобен при тонкой подстройке положения элемента. Точно также, нажимая клавиши курсора при нажатой клавише Shift, можно подстроить размер компонента.

Палитра компонентов

Для того чтобы добавить в текущую форму новый компонент, можно щелкнуть на одной из страниц палитры Components, а затем щелкнуть в форме, чтобы поместить новый элемент. Причем в форме можно или буксировать мышь с нажатой левой кнопкой, чтобы установить сразу и размер, и положение компонента, или просто щелкнуть один раз, позволяя Delphi установить размер по умолчанию.

Каждая страница палитры содержит компоненты, которые обозначены пиктограммами и именами, появляющимися в виде подсказки. Эти имена являются официальными названиями компонентов. В действительности это названия классов, описывающих компоненты без первой буквы T (например, если класс называется Tbutton, имя будет Button). Если необходимо поместить в форму несколько компонентов одного и того же вида, то при выборе компонента щелчком в палитре удерживайте нажатой клавишу Shift. Затем при каждом щелчке в форме Delphi будет вставлять новый компонент выбранного вида. Чтобы остановить эту операцию, просто щелкните по стандартному селектору (пиктограмма стрелки) слева от палитры Components.

Структура программ Delphi

Программа в Delphi состоит из файла проекта (файл с расширением .dph), одного или нескольких файлов исходного текста (с расширением .pas), файлов с описанием окон формы (с расширением .dfm).

В файле проекта находится информация о модулях, составляющих данный проект. Файл проекта автоматически создается и редактируется средой Delphi и не предназначен для редактирования.

Файл исходного текста – программный модуль (Unit), предназначенный для размещения текстов программ. В этом файле программист размещает текст программы, написанный на языке PASCAL.

В разделе объявлений описываются типы, переменные, заголовки процедур и функции, которые могут быть использованы другими модулями через операторов подключения библиотек (Uses). В разделе реализации располагаются тела процедур и функций, описанных в разделе объявлений, а также типы переменных, процедуры и функции, которые будут функционировать только в пределах данного модуля. Раздел инициализации используется редко. Модуль имеет следующую структуру:

```
unit Unit1;
interface
// Раздел объявлений
implementation
// Раздел реализации
begin
// Раздел инициализации
end.
```

При компиляции программы Delphi создает файл с расширением .dcu, содержащий результат перевода в машинные коды содержимого файлов с расширением *.pas* и *.dfm.* Компоновщик преобразует файлы с расширением .dcu в единый загружаемый файл с расширением .exe. В файлах, имеющих расширение *.~df, .~dp, .~pa,* хранятся резервные копии файлов с образом формы, проекта и исходного текста соответственно.

Пример написания программы линейного алгоритма

Задание: составить программу вычисления для заданных значений *x, y, z* арифметического выражения:

$$u = tg^2(x+y) - e^{y-z} \cdot \cos x^2 + \sin z^2$$

Настройка формы

Пустая форма в правом верхнем углу имеет кнопки управления, которые предназначены для свертывания формы в пиктограмму, разворачивания формы на весь экран и возвращения к исходному размеру и для закрытия формы. С помощью мыши, «захватывая» одну из кромок формы или выделенную строку заголовка, отрегулируйте нужные размеры формы и ее положение на экране (рис. 4.2).

Новая форма имеет одинаковые имя (Name) и заголовок (Caption) – FORM1. Имя формы менять не рекомендуется, так как оно содержится в тексте программы.

Изменение заголовка формы

Для изменения заголовка вызовите окно инспектора объектов (F11) и щелкните кнопкой мыши на форме. В форме инспектора объектов найдите и щелкните мышью на Properties – Caption . В выделенном окне наберите «Лраб 1».

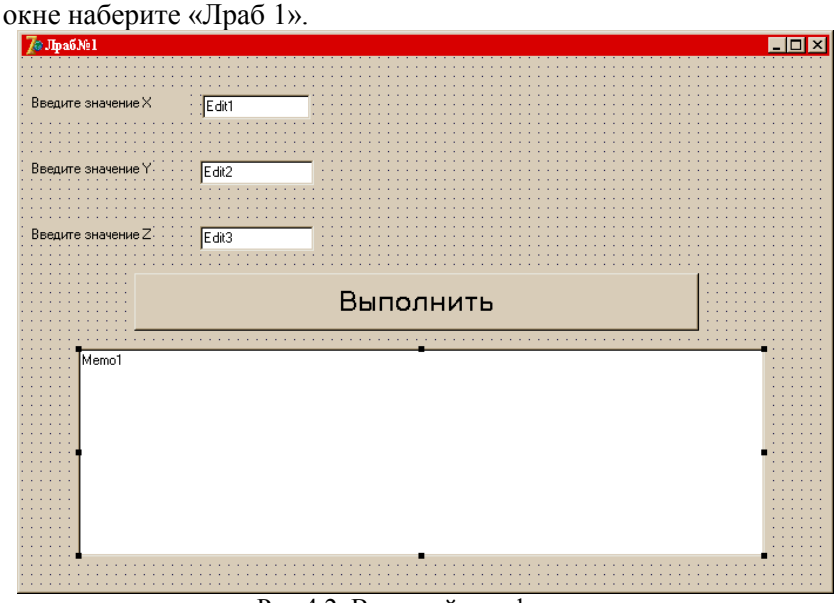

Рис.4.2. Внешний вид формы

Размещение строки ввода (TEdit)

Если необходимо ввести из формы в программу или вывести на форму информацию, которая вмещается в одну строку, используют окно однострочного редактора текста, представляемого компонентом TEdit. В данной программе с помощью однострочного редактора будут вводиться переменные *x, y, z* типа extended или integer.

Выберите в меню компонентов Standard пиктограмму , щелкните мышью в том месте формы, где вы хотите ее поставить. Вставьте три компонента TEdit в форму. Захватывая их мышью, отрегулируйте размеры окон и их положение. Обратите внимание на то, что в тексте программы появились три новых однотипных переменных Edit1, Edit2, Edit3. В каждой из этих переменных с расширением .Text будет содержаться строка символов (тип String) и отображаться в соответствующем окне Edit.

Так как численные значения переменных *x, y, z* имеют действительный тип для преобразования строковой записи числа, находящегося в переменной Edit.Text, в действительное, используется стандартная функция X:=StrToFloat(Edit1.Text).

Если исходные данные имеют целочисленный тип, например integer, то используется стандартная функция X:=StrToInt(Edit1.Text).

При этом в записи числа не должно быть пробелов, а действительное число пишется с десятичной запятой.

С помощью инспектора объектов установите шрифт и размер символов, отображаемых в строке Edit (свойство Font).

Размещение надписей (TLabel)

На форме имеются четыре пояснительные надписи. Для нанесения таких надписей на форму используется компонент TLabel. Выберите в меню компонентов Standard пиктограмму ,

щелкните на ней мышью. После этого в нужном месте формы щелкните мышью, появится надпись Label1. Проделайте это для четырех надписей. Для каждой надписи, щелкнув на ней мышью, отрегулируйте размер и, изменив свойство Caption инспектора объектов, введите строку, например "Введите значение X:", а также выберите размер символов (свойства Font).

Обратите внимание на то, что в тексте программы автоматически появились четыре новых переменных типа .TLabel. В них хранятся пояснительные строки, которые можно изменять в процессе работы программы.

Размещение многострочного окна вывода (TMemo)

Для вывода результатов работы программы обычно используется текстовое окно, которое представлено компонентом (TMemo). Выберите в меню компонентов пиктограмму  и поместите компонент TMemo на форму. С помощью мыши отрегулируйте его размеры и местоположение. После установки с помощью инспектора свойства ScrollBars – SSBoth в окне появятся вертикальная и горизонтальная полосы прокрутки.

В тексте программы появится переменная Memo1 типа TMemo. Информация, которая отображается построчно в окно типа TMemo, находится в массиве строк Memo1.Lines. Каждая строка имеет тип String.

Для чистки окна используется метод Memo1.Clear. Для того чтобы добавить новую строку в окно, используется метод Memo1.Lines.Add (переменная типа String).

Если нужно вывести число, находящееся в переменной действительного или целого типа, то его надо предварительно преобразовать к типу String и добавить в массив Memo1.Lines. Например, если переменная u:=100 целого типа, то метод Memo1.Line.Add сделает это и в окне появится строка «Значение u=100». Если переменная u:=-256,38666 действительного типа, то при использовании метода Meno1.Lines.Add('Значение u=' + FloatToStrF(u.ffFixed,8,2)) будет выведена строка «Значение u= –256.39». При этом под все число отводится восемь позиций, из которых две позиции занимает его дробная часть.

Если число строк в массиве Memo1 превышает размер окна, то для просмотра всех строк используется вертикальная полоса прокрутки. Если длина строки Memo1 превосходит количество символов в строке окна, то в окне отображается только начало строки. Для просмотра всей строки используется горизонтальная полоса прокрутки.

Написание программы обработки события нажатия кнопки(ButtonClick)

Поместите на форму кнопку, которая описывается компонентом TButton, для чего выберем в меню компонентов Standard пиктограмму  . С помощью инспектора объектов измените заголовок (Caption) – Button1 на слово «Выполнить» или другое по вашему желанию. Отрегулируйте положение и размер кнопки.

После этого два раза щелкните мышью на кнопке, появится текст программы, дополненной заголовком процедуры обработчика события – нажатия кнопки (Procedure TForm1.ButtonClick(Sender : TObject);).

Наберите текст этой процедуры, приведенный в примере.

Запуск и работа с программой

Запустить программу можно, нажав Run в главном меню Run, или клавишу F9, или пиктограмму  . При этом происходит трансляция и, если нет ошибок, компоновка программы и создание единого загружаемого файла с расширением .exe. На экране появляется активная форма программы.

Работа с программой происходит следующим образом. Нажмите (щелкните мышью) кнопку «Выполнить». В окне Memo1 появится результат. Измените исходные значения *x, y, z* в окнах Edit и снова нажмите кнопку «Выполнить» – появятся новые результаты. Завершить работу программы можно нажав в главном меню Run или кнопку {} на форме.

Текст программы:

```
unit tema1;
interface
uses
Windows, Messages. SysUtils, Classes, Graphics, Controls, Forms, Dialogs, StdCtrls;
type
TForm1 = class(TForm)
Label1:TLabel;
Edit1:TEdit;
Label2: TLabel;
Edit2:TEdit:
Label3: TLabel;
```

```
Edit3: TEdit;
Label4: TLabel;
Memo1:TMemo;
Buttonl : TButfon;
procedure FormCreate(Sender: TObject);
procedure Button1Click(Sender: TObject);
 private
{ Private declarations }
 public
{ Public declarations }
 end;
var
Form1: TForm1;
Implementation
 {$R*.DFM}

procedure TForm1.Button1Click(Sender: TObject);
var
x, y, z, a, b, c, u : extended;
begin
x:=StrToFloat(Edit1.Text);                    // Считывается значение X
Memo1.Lines.Add('X = '+Edit1.Text);          // Вывод X в окно Memo1
y:=StrToFloat(Edit2.Text);                    // Считывается значение Y
Memo1.Lines.Add('Y = '+Edit2.Text);          // Вывод Y в окно Memo1
z:=StrToFloat(Edit3.Text);                    // Ситывается значение Z
Memo1.Lines.Add('Z = '+Edit3.Text);          // Вывод Z в окно Memo1
                          // Вычисляем арифметическое выражение
a:=Sqr(Sin(x+y)/Cos(x+y));
b:=Exp(y–z);
c:=Sqrt(Cos(Sqr(x))+Sin(Sqr(z)));
u:=a–b*c;
                          // Выводим результат в окно Memo1
Memo1.Lines.Add('Результат  U = '+FloatToStrF(u,ffFixed,8,3));
end;
end.
```

Лабораторная работа 5

Использование визуальных компонент для создания качественных программ

Цель работы: научиться пользоваться простейшими компонентами организации переключений (TCheckBox, TRadioGroup).

Общие сведения

Операторы if и case языка Паскаль

Для программирования разветвляющихся алгоритмов в языке Pascal используются специальные переменные типа boolean, которые могут принимать только два значения – true и false (да, нет), а также операторы if и case. Оператор if проверяет результат логического выражения, или значение переменной типа boolean, и организует разветвление вычислений.

Например, если bl: boolean, x,y,u:integer, то фрагмент программы с оператором if может быть таким:

```
bl:=x>y;
if   bl    then u:=x–y
 else u:=x–y;
```

Оператор выбора case организует разветвления в зависимости от значения некоторой переменной перечисляемого типа.

Например, если In: integer, то после выполнения

```
case in of
    0: u:=x+y;
    1: u:=x–y;
    2: u:=x*y;
    else u=0;
    end;
```

В соответствии со значением in вычисляется u. Если in = 0, то u = x + y, если in = 1, то u = x – y, если in = 2, то u = x * y и, наконец, u = 0 при любых значениях in, отличных от 0, 1 или 2.

Кнопки-переключатели в Delphi

При создании программ в Delphi для организации разветвлений часто используются компоненты в виде кнопок-переключателей. Состояние такой кнопки (включено – выключено) визуально отражается на форме.

Компонент TCheckBox организует кнопку независимого переключателя, с помощью которой

пользователь может указать свое решение типа да/нет. В программе состояние кнопки связано со значением булевской переменной, которая проверяется с помощью оператора if.

Компонент TRadioGroup организует группу кнопок-зависимых переключателей. При нажатии одной из кнопок группы все остальные кнопки отключаются. В программу передается номер включенной кнопки (0, 1, 2, …), который анализируется с помощью оператора case.

Пример программы разветвляющегося алгоритма

Задание: ввести три числа – *x, y, z*. Вычислить по усмотрению *u = sin(x)* или *u = cos(x)*, или *u = tg(x)*. Найти по желанию максимальное из трех чисел: max(*x, y, z*), или min (|*x*|, |*y*|, |*z*|). Создать форму и написать соответствующую программу.

Создайте форму, такую же как в первом задании, скорректировав текст надписей и положение окон TEdit.

Работа с компонентом TCheckBox

Выберите в меню компонентов Standard пиктограмму {} и поместите ее в нужное место формы. С помощью инспектора объектов измените заголовок (Caption) на «maabs». В тексте программы появится переменная CheckBox типа TCheckBox. Теперь в зависимости от того, нажата кнопка или нет, булевская переменная CheckBox.Checked будет принимать значения True или False.

Работа с компонентом TRadioGroup

Выберите в меню компонентов Standard пиктограмму  и поместите ее в нужное место формы. На форме появится окаймленный линией чистый прямоугольник с заголовком RadioGroup1. Замените заголовок (Caption) на *U*(*x*). Для того чтобы разместить на компоненте кнопки, необходимо свойство Columns установить равным единице (кнопки размещаются в одном столбце). Дважды щелкните по правой части свойства Items мышью, появится строчный редактор списка заголовков кнопок. Наберите три строки с именами: в первой строке sin(*x*), во второй – cos(*x*), в третьей – tg(*x*), нажмите ОК. После этого на форме внутри окаймления появzтся три кнопки-переключателя с введенными надписями.

Обратите внимание на то, что в тексте программы появилась переменная RadioGroup типа TRadioGroup. Теперь при нажатии одной из кнопок группы в переменной целого типа RadioGroup1.ItemIndex будет находиться номер нажатой клавиши (отсчитывается от нуля), что используется в тексте приведенной программы.

Создание обработчиков событий FormCreate и Button1Click

Процедуры – обработчики событий FormCreate и Button1Click – создаются аналогично тому, как и в первой теме.

Запустите программу и убедитесь в том, что все ветви алгоритма выполняются правильно. Форма приведена на рис. 5.1.

Рис. 5.1. Форма программы разветляющегося алгоритма

Текст программы приведен ниже.

```
unit tema2;
interface

uses
 Windows, Messages, SysUtils, Classes, Graphics, Controls,  Forms, Dialogs, StdCtrls, ExtCtrls;
 type
TForm1 = class(TForm)
CheckBox1: TCheckBox;
RadioGroup1: TRadioGroup;
Memo1: TMemo;
Button1:TButton;
```

```
Edit1:TEdit;
Label1: TLabel;
Label2: TLabel;
Edit2:TEdit;
Label3: TLabel:
Edi3: TEdlt;

        procedure FormCreate(Sender: TObject);
        procedure Button1Click(Sender: TObject);
          private
        {Private declarations}
        public
{public declarations}
var
Form1: TForm1;

implementation

{$R.DFM}
procedure TForm1.FormCreate(Sender: TObject);
begin
Edit1.Text:='0,1';
Edit3.Ttext:='0,356';
Memo1.Clear;
Memo1.Lines.Add('Рез–ты  ст.гр. 9383 Валента А.А.');
end;
procedure TForm1.Button1Click(Sender: TObject);
var x, y, z, u, ma : extended;
begin
      // Ввод исходных данных и их вывод в окно Memo1
x:=StrToFloat(Edit1.Text);
      Memo1.Lines.Add('x='+Edit1.Text);
y:=StrToFloat(Edlt2.Text);
      Memo1.Lines.Add('y='+Edit2.Text);
z:=StrToFloat(Edtt3.Text);
      Memo1.Lines.Add('z='+Edit3.Text);
            // Проверка номера нажатой кнопки и выбор соответствующей ей функции
 case RadioGroup.ltemlndex of
          0: u:=cos(x);
          1: u:=sin(x);
          2: u:=sin(x)/cos(x);
  end:
            // Проверка состояния кнопки CheckBox1
     if  CheckBox1.Checked      then
                begin
                      u:=abs(u);
                      y:=abs(y);
                      z:=abs(z)
                 end;
                          //    Нахождение максимального из трех чисел
        if   u>y     then  ma:=u   else ma: =y;
        if   z>ma   then  ma:=z;
              if  CheckBox1.Checked   then
                    Memo1.Lines.Add(' maxabs='+FloatToStrF(ma,ffFixed,8,2))
                    else
                    Memo1.Lines.Add('max='+FtoatToStrF(ma,ffGeneral,8,2));
end;
end.
```

Индивидуальные задания

Ниже приведены 15 вариантов задач. По указанию преподавателя выберите свое индивидуальное задание. Уточните условие задания, количество, наименование, типы исходных данных. В соответствии с этим установите количество окон Edit, тексты заголовков на форме, размеры шрифтов, а также типы переменных и функции преобразования при вводе и выводе результатов. С помощью инспектора объектов измените цвет формы, шрифт выводимых символов.

1. Известно, что из четырех чисел *a1,a2,a3* и *a4* одно отлично от трех других, равных между собой. Присвоить номер этого числа переменной *n*.

2. По номеру n ($n > 0$) некоторого года определить c – номер его столетия (учесть, что, к примеру, началом XX столетия был 1901, а не 1900 год!).

3. Значения переменных *a, b* и *c* поменять местами так, чтобы оказалось $a \le b \le c$.

4. Дано целое k от 1 до 180. Определить, какая цифра находится в *k*–й позиции последовательности 10111213...9899 , в которой выписаны подряд все двузначные числа.

5. Дано натуральное *k*. Определить *k*-ю цифру в последовательности 110100100010000100000..., в которой выписаны подряд степени 10.

6. В старояпонском календаре был принят 60–летний цикл, состоявший из пяти 12-летних подциклов. Подциклы обозначались названиями цвета: green(зеленый), red (красный), yellow (желтый), white(белый) и black (черный). Внутри каждого подцикла годы носили названия животных: крысы, коровы, тигра, зайца, дракона, змеи, лошади, овцы, обезьяны, курицы, собаки и свиньи (1984 год – *год зеленой крысы* – был началом очередного цикла). Разработать программу, которая вводит номер некоторого года нашей эры и выводит его название по старояпонскому календарю.

7. Если сумма трех попарно различных действительных чисел *x, y, z* меньше единицы, то наименьшее из этих трех чисел заменить полусуммой двух в противном случае заменить меньшее из *x* и *y* полусуммой двул оставшихся значений,

8. Для целого числа *k* от 1 до 99 вывести фразу «мне *k* лет», учитывая при этом, что при некоторых значениях *k* слово «лет» надо заменить на слово «год» или «года».

9. Для натурального числа *k* вывести фразу «мы выпили *k* бутылок пива», согласовывая окончание слова «бутылка» с числом *k*.

10. *Type курс=(С,В,Ю,З); {север, восток, юг, запад}*

Приказ=(вперед, вправо, назад, влево);

Var К1, К2: курс; ПР: приказ;

Корабль сначала шел по курсу К1, а затем его курс был изменен согласно прикаказу ПР1. Определить К2 – новый курс корабля.

11. Type месяц = (январь, февраль, март, апрель, май, июнь, июль, август, сентябрь, октябрь, ноябрь, декабрь);

день=1...31;

Var d1,d2: день;

m1,m2: месяц;

t: boolean;

Переменной *t* присвоить значение 1 если дата *d*1, *m*1 предшествует (в рамках года) дате *d*2, *m2*, и значение 0 в других случаях.

12. Type нота=(до, ре, ми, фа, соль, ля, си);

интервал=(секунда, терция, кварта, квинта, секста, септима);

var n1, n2: нота;

i: интервал;

Определить *i*-й интервал, образованный нотами *n*1 и *n*2 *(n*1<>*n*2*):* секунда – это интервал из двух соседних (по кругу) нот (например, ре и ми, си и до), терция – интервал через ноту (например, фа и ля, си и ре) и т.д.

13. Type единица = (дециметр, километр, метр, миллиметр, сантиметр);

длина=real;

var x: длина;

P: единица;

Значение переменной *x,* означающее некоторую длину в единицах *p,* заменить на величину этой же длины в метрах.

14. Type сезон = (зима, весна, лето, осенъ);

Var m: месяц; {определение «месяц» см. в 26}

S: сезон;

Определить *S* – сезон, на который приходится месяц *m*.

15. Var k: 1...9;

Вывести значение переменной *k* римскими цифрами.

Лабораторная работа 6

Средства отладки программ в объектно-ориентированном программировании

Цель работы: изучить простейшие средства отладки программ в среде Delphi.

Общие сведения

Операторы организации циклов repeat, whyle, for языка Pascal

Под циклом понимается многократное выполнение одних и тех же операторов при различных значениях промежуточных данных. Число повторений может быть задано в явной или неявной форме.

Для организации повторений в языке Pascal предусмотрены три различных оператора цикла.

Оператор

repeat
<операторы>
until<условие>;

Организует повторение операторов, помещенных между ключевыми словами repeat и until, до тех пор, пока не выполнится <условие> = true, после чего управление передается следующему за циклом оператору.

Оператор

While <условие> do begin
<операторы>
end;

Организует повторение операторов, помещенных между begin и end, до тех пор, пока не выполнится <условие> = false. Следует отметить, что если <условие> = false при первом входе, то <операторы> не выполнятся ни разу, в отличие от repeat, в котором хотя бы один раз они выполнятся.

Оператор

for i:=i1 to i2 do begin
<операторы>
end;

Организует повторение операторов при нарастающем изменении переменной цикла *i* от начального значения *i*1 до конечного *i*2 с шагом «единица». Если *i*2 > *i*1, то <операторы> не выполнятся ни разу. Модификация оператора for i:=i2 downto i1 do begin <операторы> end организует повторения при убывающем изменении *i* на единицу.

Средства отладки программ в Delphi

Практически в каждой вновь написанной программе после запуска обнаруживаются ошибки.

Ошибки первого уровня (ошибки компиляции) связаны с неправильной записью операторов (орфографические, синтаксические). При обнаружении ошибки компилятор Delphi останавливается напротив первого оператора, в котором обнаружена ошибка. В нижней части экрана появляется текстовое окно, содержащее сведения обо всех ошибках, найденных в проекте. Каждая строка этого окна содержит имя файла, в котором найдена ошибка, номер строки с ошибкой и характер ошибки. Для быстрого перехода к интересующей ошибке необходимо дважды щелкнуть на строке с ее описанием. Для получения более полной информации о характере ошибки необходимо обратится к HELP нажатием клавиши F1. Следует обратить внимание на то, что одна ошибка может повлечь за собой другие, которые исчезнут при ее исправлении. Поэтому следует исправлять ошибки последовательно, сверху вниз и после исправления каждой ошибки компилировать программу снова.

Ошибки второго уровня (ошибки выполнения) связаны с ошибками выбранного алгоритма решения или с неправильной программной реализацией алгоритма. Эти ошибки проявляются в том, что результат расчета оказывается неверным либо происходит переполнение, деление на ноль и др. Поэтому перед использованием отлаженной программы ее надо протестировать, т.е. сделать просчеты при таких комбинациях исходных данных, для которых заранее известен результат. Если тестовые расчеты указывают на ошибку, то для ее поиска следует использовать встроенные средства отладки среды DELPHI.

В простейшем случае для локализации места ошибки рекомендуется поступать следующим образом. В окне редактирования текста установить курсор в строке перед подозрительным участком и нажать клавишу F4 (выполнение до курсора). Выполнение программы будет остановлено на строке, содержащей курсор. Теперь можно увидеть, чему равно значение интересующих переменных. Для этого можно поместить на нужную переменную курсор (на экране будет высвечено ее значение) либо нажать Ctrl-F7 и в появившимся диалоговом окне указать интересующую переменную (с помощью данного окна можно также изменить значение переменной во время выполнения программы). Нажимая клавишу F7 (пошаговое выполнение), можно построчно выполнять программу, контролируя изменение тех или иных переменных и правильность вычислений. Если курсор находится внутри цикла, то после нажатия F4 расчет останавливается после одного выполнения тела цикла. Для продолжения расчетов следует нажать <Run> меню Run.

Пример написания программы циклического алгоритма

Задание: написать и отладить программу, которая выводит таблицу значений функции $S(x) = \sum_{k=0}^{N} (-1)^k \frac{x^k}{k!}$ для x изменяющихся в интервале от $X1$ до $X2$ с шагом h.

Окно диалога представлено на рис. 6.1.

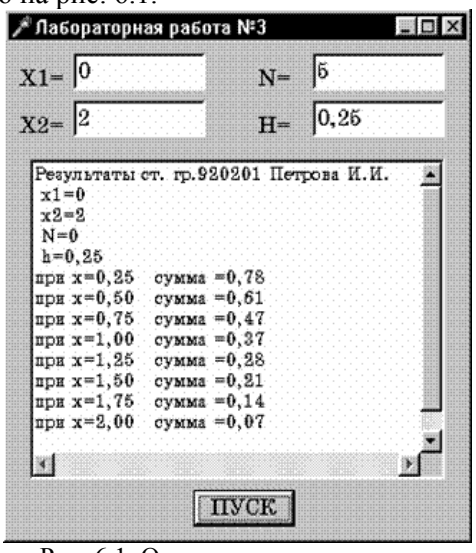

Рис. 6.1. Окно диалога программы

Текст программы приведен ниже.

```
unit tema3;
interface
uses
Windows,  Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs, StdCtrls, ExtCtrls;
type
        TForm1 = class(TForm)
         Memo1: TMemo;
         Button1: TButton;
         Label1: TLabel;
         Label2: TLabel;
         Label3: TLabel;
         Label4: TLabel;
         Edit1: TEdit;
         Edit2: TEdit;
          Edit3: TEdit;
          Edit4: TEdit;
procedure FormCreate(Sender: TObject);
procedure Button1Click(Sender: TObject);
private
  { Private declarations }
public
  { Public declarations}
end;
var Form1: TForm1;
implementation
{$R*.DFM}
procedure TForm1.FormCreate(Sender: TObject);
begin
  Edit1.Text :='0';
  Edit2.Text :='2';
  Edit3.Text :='5';
  Edit4.Text :='0.25';
Memo1.Clear;
Memo.Lines.Add('Результаты ст. гр. 9383 Валета А.В.');
end;
procedure TForm1.Button1Cllck(Sender: TObject);
var x1, x2, x, h, a, s : extended;
      N, k, c : integer;
begin
x1:=StrToFloat(Edit1.Text);
   Memo1.Lines.Add('x1='+Edit1.Text);
x2:=StrToFloat(Edit2.Text);
    Memo1.Lines.Add('x2='+Edit2.Text);
N:=StrTolnt(Edit3.Text);
    Memo1.lines.Add('N='+Edit3.Text);
h:=StrToFloat(Edit4.Text);
    Memo1.Lines.Add(' h='+Edit4.Text);
      c:=–1;
      x:=x1;
```

```
repeat
     a:=1;
     S:=1;
       for k:=1 to N do
           begin
   a:=c*a*x/k;
    s:=s+a;
end;
Memo1.lines.Add('при x=' +FloatToStrF(x,ffFixed,6,2)+ 'сумма =' + FloatToStrF(s,tfFtxed,6,2));
   x:=x+h;
until x>x2;
        end;
        end.
```

После отладки программы составьте тест ($N = 2$, $X1 = 0$, $X2 = 1$, $h = 0{,}3$), установите курсор на первый оператор (N:=), нажмите клавишу F4. После этого нажимая клавишу F7, выполните пошаговую программу и проследите, как меняются все переменные в процессе выполнения.

Зачетно-экзаменационные материалы для промежуточной аттестации (зачёт)

Студенты обязаны сдать зачет в соответствии с расписанием и учебным планом. Зачет является формой контроля усвоения студентом учебной программы по дисциплине или ее части, выполнения лабораторных работах.

Результат сдачи зачета по прослушанному курсу должны оцениваться как итог деятельности студента в семестре, а именно - по посещаемости лекций, результатам работы на лабораторных работах, выполнения самостоятельной работы. При этом допускается на очной форме обучения пропуск не более 20% занятий, с обязательной отработкой пропущенных занятий. Студенты, у которых количество пропусков, превышает установленную норму, не выполнившие все виды работ и неудовлетворительно работавшие в течение семестра, проходят собеседование с преподавателем, который опрашивает студента на предмет выявления знания основных положений дисциплины.

Перечень компетенций (части компетенции), проверяемых оценочным средством:

ПК-6

4.2 Методические материалы, определяющие процедуры оценивания знаний, умений, навыков и (или) опыта деятельности, характеризующих этапы формирования компетенций

Методические рекомендации определяющие процедуры оценивания на экзамене

Критерии оценки:

- оценка «зачтено»: студент владеет теоретическими знаниями по данному разделу, знает основную теорию дисциплины, допускает незначительные ошибки; студент умеет правильно объяснять материал, иллюстрируя его примерами.

- оценка «не зачтено»: материал не усвоен или усвоен частично, студент затрудняется привести примеры, довольно ограниченный объем знаний программного материала Оценочные средства для инвалидов и лиц с ограниченными возможностями здоровья выбираются с учетом их индивидуальных психофизических особенностей.

– при необходимости инвалидам и лицам с ограниченными возможностями здоровья предоставляется дополнительное время для подготовки ответа на экзамене;

– при проведении процедуры оценивания результатов обучения инвалидов и лиц с ограниченными возможностями здоровья предусматривается использование технических средств, необходимых им в связи с их индивидуальными особенностями;

– при необходимости для обучающихся с ограниченными возможностями здоровья и инвалидов процедура оценивания результатов обучения по дисциплине может проводиться в несколько этапов.

Процедура оценивания результатов обучения инвалидов и лиц с ограниченными возможностями здоровья по дисциплине (модулю) предусматривает предоставление информации в формах, адаптированных к ограничениям их здоровья и восприятия информации:

Для лиц с нарушениями зрения:

– в печатной форме увеличенным шрифтом,

– в форме электронного документа.

Для лиц с нарушениями слуха:

– в печатной форме,

– в форме электронного документа.

Для лиц с нарушениями опорно-двигательного аппарата:

– в печатной форме,

– в форме электронного документа.

Данный перечень может быть конкретизирован в зависимости от контингента обучающихся.

5. Перечень основной и дополнительной учебной литературы, необходимой для освоения дисциплины (модуля)

5.1 Основная литература:

1 Гибкая методология разработки программного обеспечения: курс / Национальный Открытый Университет "ИНТУИТ". – Москва : Интернет-Университет Информационных Технологий (ИНТУИТ), 2010. – 134 с. : ил. – Режим доступа: по подписке. – URL: http://biblioclub.ru/index.php?page=book&id=233769.

2. Схиртладзе, А.Г. Проектирование единого информационного пространства виртуальных предприятий : учебник / А.Г. Схиртладзе, А.В. Скворцов, Д.А. Чмырь. – Изд. 2-е, стер. – Москва ; Берлин : Директ-Медиа, 2017. – 617 с. : ил., схем., табл. – Режим доступа: по подписке. – URL: http://biblioclub.ru/index.php?page=book&id=469047.

Для освоения дисциплины инвалидами и лицами с ограниченными возможностями здоровья имеются издания в электронном виде в электронно-библиотечных системах «Лань» и «Юрайт».

5.2 Дополнительная литература:

1. Чусавитина, Г. Н. Математические методы управления проектами : учебное пособие / Г. Н. Чусавитина, В. Н. Макашова, И. К. Скокова. — 2-е изд. — Москва : ФЛИНТА, 2017. — 130 с. — ISBN 978-5-9765-3794-1. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/104933.

2. Контроллинг: теория и практика [Электронный ресурс] : учебник и практикум для академического бакалавриата / С. В. Осипов [и др.] ; под общ. ред. С. В. Осипова. - М. : Юрайт, 2018. - 145 с. - https://biblio-online.ru/book/891301E1-969E-455F-A4FE-AD7209AC700F

3. Григорьев, М. В. Проектирование информационных систем [Электронный ресурс] : учебное пособие для вузов / М. В. Григорьев, И. И. Григорьева. - М. : Юрайт, 2018. - 318 с. - https://biblio-online.ru/book/394E4411-7B76-4F47-BD2D-C3B981BEC3B8.

4. Одинцов, Б. Е. Информационные системы управления эффективностью бизнеса [Электронный ресурс] : учебник и практикум для бакалавриата и магистратуры / Б. Е. Одинцов. - М. : Юрайт, 2017. - 206 с. - https://biblio-online.ru/book/A776D72A-816A-4037-A427-23F71AF28852

5.3. Периодические издания:
Не используются

6. Методические указания для обучающихся по освоению дисциплины (модуля)

Основными формами контактной работы по дисциплине «Коллективная разработка программных продуктов» для очной формы обучения являются лекции, лабораторные работы и контролируемая самостоятельная работа.

Лекции по дисциплине «Коллективная разработка программных продуктов» следует проводить в классах кафедры теоретической физики и компьютерных технологий с использованием средств мультимедиа.

Лабораторные работы по дисциплине «Коллективная разработка программных продуктов» следует проводить в компьютерных классах кафедры теоретической физики и компьютерных технологий.

Контролируемую самостоятельную работу студентов по дисциплине «Коллективная разработка программных продуктов» следует проводить в компьютерных классах кафедры теоретической физики и компьютерных технологий. Проведение занятий предусматривает постановку проблемных вопросов, анализ возможных алгоритмов действий и поиск оптимального решения.

Структура дисциплины «Коллективная разработка программных продуктов» для очной формы обучения определяет следующие виды самостоятельной работы: самостоятельная работа студента (СРС) и контроль (К).

Самостоятельная работа студента является основным видом самостоятельной работы. Она проводится в целях закрепления знаний, полученных на всех видах учебных занятий, а также расширения и углубления знаний, т.е. активного приобретения студентами новых знаний.

СРС включает проработку и повторение лекционного материала. Для этого студенту рекомендуется прочитать текст лекции, пересказать его вслух, воспроизвести самостоятельно имеющиеся в тексте структурно-логические схемы, диаграммы, математические выкладки формул, доказательства теорем и т.п. Проработку лекционного материала следует проводить сначала последовательно, по каждому учебному вопросу, а затем повторно, по всему тексту лекции.

СРС также включает изучение материала по рекомендованным учебникам и учебным пособиям. Так как существует огромное количество учебной литературы, то для этого вида подготовки необходимо предварительное указание преподавателя. Преподаватель должен выступать здесь в роли опытного «путеводителя», определяя последовательность знакомства с литературными источниками и «глубину погружения» в каждый из них.

Одним из видов СРС является подготовка к лабораторным работам. Преподаватель накануне очередного занятия обозначает для студентов круг теоретического материала, необходимого для задач на семинарских занятиях. Студенты прорабатывают его. Затем, уже в аудитории, перед выполнением заданий, преподаватель производит контрольный опрос студентов. Это позволяет определить степень готовности группы по данной теме и скорректировать ход занятия.

Преподаватель должен прогнозировать затруднения, которые могут возникнуть у студентов при самостоятельном изучении и усвоении учебного материала и предусмотреть оперативную консультацию по любому вопросу. Если возникают затруднения по одному и тому же материалу (вопросу) у многих студентов, то желательно провести групповую консультацию. Консультации должны быть краткими: групповая - 2-3 мин., индивидуальная - 1-2 мин. Глубину и качество усвоения учебного материала необходимо непрерывно отслеживать при проведении текущего контроля знаний.

Консультации, выдача лабораторных заданий и прием результатов выполнения осуществляется только во время аудиторных занятий. Задания выполняются последовательно. Правильное выполнение некоторых заданий возможно только, если студент корректно выполнил предыдущие задания.

Поэтому приступать к следующему заданию студент может, только сдав преподавателю результат выполнения предыдущего. Результаты выполнения лабораторных работ демонстрируются преподавателю. Во время приема выполненной работы преподаватель вправе:

1) Требовать у студента демонстрации выполнения практического задания, предусмотренной заданием.

2) Самостоятельно производить манипуляции с средой моделирования, не изменяя модели, составленной студентом.

3) Требовать у студента пояснений, относящихся к разработанной модели.

Задание считается выполненным и принимается преподавателем только в том случае, если реализован весь функционал, предусмотренный заданием.

Если какие-то функции, предусмотренные заданием, не работают, или работают неверно, то результат выполнения подлежит доработке. Студент должен работать над кодом программы максимально самостоятельно, использовать отладочные средства, предоставляемые изучаемой программной средой.

До конца семестра студент должен сдать результаты выполнения всех лабораторных работ, предусмотренным настоящими указаниями. В противном случае студенты к сдаче зачета не допускаются.

В освоении дисциплины инвалидами и лицами с ограниченными возможностями здоровья большое значение имеет индивидуальная учебная работа (консультации) – дополнительное разъяснение учебного материала.

В освоении дисциплины инвалидами и лицами с ограниченными возможностями здоровья большое значение имеет индивидуальная учебная работа (консультации) – дополнительное разъяснение учебного материала.

Индивидуальные консультации по предмету являются важным фактором, способствующим индивидуализации обучения и установлению воспитательного контакта между преподавателем и обучающимся инвалидом или лицом с ограниченными возможностями здоровья.

7. Перечень информационных технологий, используемых при осуществлении образовательного процесса по дисциплине (модулю)

7.1 Перечень информационно-коммуникационных технологий

-Проверка заданий и консультирование посредством электронной почты и популярных социальных сетей;

– Использование электронных презентаций при проведении лекционных занятий;

– Разбор готовых программных проектов на практических занятиях.

7.2 Перечень лицензионного и свободно распространяемого программного обеспечения

1. Операционная система MS Windows;
2. Microsoft Visual Studio.

7.3 Перечень современных профессиональных баз данных и информационных справочных систем

1. Справочно-правовая система «Консультант Плюс» (http://www.consultant.ru)
2. Электронная библиотечная система eLIBRARY.RU (http://www.elibrary.ru)/
3. БД Web of Science - главный ресурс для исследователей по поиску и анализу научной литературы, охватывающей около 18000 научных журналов со всего

мира. База данных международных индексов научного цитирования http://webofscience.com/

4. zbMATH - полная математическая база данных. Охватывает материалы с конца 19 века. zbMATH содержит около 4000000 документов из более 3000 журналов и 170000 книг по математике, статистике, информатике. https://zbmath.org/
5. БД Kaggle - это платформа для сбора и обработки данных. Является он-лайн площадкой для научного моделирования. https://www.kaggle.com/
6. База данных Научной электронный библиотеки eLIBRARY.RU https://elibrary.ru/
7. База данных Всероссийского института научной и технической информации (ВИНИТИ) РАН http://www2.viniti.ru/
8. «ЭЛЕКТРОННАЯ БИБЛИОТЕКА ДИССЕРТАЦИЙ» Российской Государственной Библиотеки (РГБ) – в настоящее время ЭБД содержит более 800 000 полных текстов диссертаций. https://dvs.rsl.ru
9. Портал открытых данных Российской Федерерации https://data.gov.ru
10. База открытых данных Министерства труда и социальной защиты РФ https://rosmintrud.ru/opendata
11. Федеральный портал единое окно доступа к информационным ресурсам - http://window.edu.ru/
12. Российский фонд фундаментальных исследований предоставляет доступ к информационным наукометрическим базам данных и полнотекстовым научным ресурсами издательств Springer Nature и Elsevier - http://www.rfbr.ru/rffi/ru
13. Федеральный портал "Информационно-коммуникационные технологии в образовании" - http://www.ict.edu.ru/

8. Материально-техническое обеспечение по дисциплине (модулю)

№	Вид работ	Материально-техническое обеспечение дисциплины (модуля) и оснащенность
1.	*Лекционные занятия*	Лекционная аудитория, оснащенная презентационной техникой (проектор, экран, компьютер/ноутбук) и соответствующим программным обеспечением (ПО) для воспроизведения файлов формата jpg и avi, достаточным количеством посадочных мест. 300, 114, 209, 201 корп. С.
2.	*Семинарские занятия*	Аудитория для проведения семинарских занятий, оснащенная магнитно-маркерной доской, комплектом учебной мебели и презентационной техникой. 142, 114, 227, 209, 201 корп. С.
3.	*Лабораторные занятия*	Лаборатория, укомплектованная специализированной мебелью и техническими средствами обучения. 207, 212, 213 корп. С.
4.	*Курсовое проектирование*	Не предусмотрено
5.	*Групповые (индивидуальные) консультации*	Аудитория для проведения групповых (индивидуальных) занятий, оснащенная доской и комплектом учебной мебели. 212, 213, 207 корп. С.
6.	*Текущий контроль, промежуточная аттестация*	Аудитория для текущего контроля и промежуточной аттестации студентов, оснащенная компьютерной техникой с возможностью подключения к сети “Интернет”, с соответствующим программным обеспечением в режиме подключения к терминальному серверу, с программой

		экранного увеличения и обеспеченный доступом в электронную информационно-образовательную среду университета. 114, 212, 230 корп. С.
7.	*Самостоятельная работа*	Кабинет для самостоятельной работы, оснащенный компьютерной техникой с возможностью подключения к сети «Интернет», программой экранного увеличения и обеспеченный доступом в электронную информационно-образовательную среду университета. 208 корп. С.