МИНОБРНАУКИ РОССИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Санкт-Петербургский государственный технологический институт
(технический университет)»

УТВЕРЖДАЮ
Проректор по учебной
и методической работе
_____ Б.В.Пекаревский
« 20 » мая 2019 г.

Рабочая программа дисциплины

РАЗРАБОТКА ПРОГРАММНЫХ СИСТЕМ

Направление подготовки

09.03.01 Информатика и вычислительная техника

Направленность программы бакалавриата

Автоматизированные системы обработки информации и управления

Квалификация

Бакалавр

Форма обучения

Заочная

Факультет **информационных технологий и управления**

Кафедра **систем автоматизированного проектирования и управления**

Санкт-Петербург

2019

Б1.О.13

ЛИСТ СОГЛАСОВАНИЯ

Должность разработчика	Подпись	Ученое звание, фамилия, инициалы
доцент		доцент В.Н. Уланов
доцент		И.Г. Корниенко
старший преподаватель		А.К. Федин
аспирант		Э.Э. Мусаев

Рабочая программа дисциплины «Разработка программных систем» обсуждена на заседании кафедры систем автоматизированного проектирования и управления
протокол от «18» апреля 2019 года №9
Заведующий кафедрой профессор Т.Б. Чистякова

Одобрено учебно-методической комиссией факультета информационных технологий и управления
протокол от «15» мая 2019 года №9

Председатель доцент В.В. Куркина

СОГЛАСОВАНО

Руководитель направления подготовки «Информатика и вычислительная техника»		профессор Т.Б. Чистякова
Директор библиотеки		Т.Н. Старостенко
Начальник методического отдела учебно-методического управления		Т.И. Богданова
Начальник учебно-методического управления		С.Н. Денисенко

СОДЕРЖАНИЕ

Приложения: 1. Фонд оценочных средств для проведения промежуточной аттестации.

1. Перечень планируемых результатов обучения по дисциплине, соотнесенных с планируемыми результатами освоения образовательной программы.

В результате освоения образовательной программы бакалавриата обучающийся должен овладеть следующими результатами обучения по дисциплине:

Код и наименование компетенции	Код и наименование индикатора достижения компетенции	Планируемые результаты обучения (дескрипторы)
ОПК-4 Способен участвовать в разработке стандартов, норм и правил, а также технической документации, связанной с профессиональной деятельностью	ОПК-4.3 Определение основных стандартов оформления технической документации на различных стадиях жизненного цикла программной системы	Приводит примеры основных технологий создания и внедрения информационных систем, стандартов управления жизненным циклом программной системы (ЗН-1)
		Сопоставляет основные технологии создания и внедрения информационных систем, стандартов управления жизненным циклом программной системы (У-1)
		Решает задачу выбора эффективной технологии для создания и внедрения информационной системы, стандартов управления жизненным циклом программной системы (Н-1)
ПК-1 Способен выполнять работы и управлять работами по созданию (модификации) и сопровождению ИС, автоматизирующих задачи организационного управления и бизнес-процессы	ПК-1.15 Выявление и анализ требований к программной системе	Приводит примеры требований к программной системе (ЗН-2)
		Анализирует и объясняет требования к программной системе в соответствии с техническим заданием (У-2)
		Решает задачи создания программной системы в соответствии с выявлением и анализом требований к программной системе с использованием компьютерных средств проектирования (Н-2)
	ПК-1.16 Разработка архитектуры программной системы	Перечисляет принципы разработки архитектуры программной системы (ЗН-3)
		Определяет закономерности при разработке архитектуры программной системы (У-3)
		Демонстрирует навыки разработки архитектуры программной системы (Н-3)
ПК-2 Способен осуществлять концептуальное, функциональное и логическое проектирование систем среднего и крупного масштаба и сложности	ПК-2.1 Разработка концепции программной системы	Перечисляет принципы разработки концепции программной системы (ЗН-4)
		Определяет закономерности при разработке концепции программной системы (У-4)
		Демонстрирует навыки разработки концепции системы (Н-4)

Код и наименование компетенции	Код и наименование индикатора достижения компетенции	Планируемые результаты обучения (дескрипторы)
	ПК-2.2 Разработка технического задания на программную систему	Знает состав, содержание, требования и стандарты необходимые при разработке технического задания на программную систему (ЗН-5)
		Разрабатывает техническое задание на программную систему (У-5)
		Имеет навыки разработки технического задания на программную систему (Н-5)

2. Место дисциплины в структуре образовательной программы.

Дисциплина относится к дисциплинам обязательной части (Б1.О.13) и изучается на 2 курсе в 3-ем и на 33 курсе в 5-ом и 6-ом семестрах.

В методическом плане дисциплина опирается на элементы компетенций, сформированные при изучении дисциплин «Информатика» и «Программирование». Полученные в процессе изучения дисциплины «Разработка программных систем» знания, умения и навыки могут быть использованы при изучении дисциплин «Системы тестирования программного обеспечения», «Основы разработки автоматизированных информационных систем», «Математическая логика и теория алгоритмов» при прохождении производственной практики, а также при выполнении выпускной квалификационной работы.

3. Объем дисциплины.

Вид учебной работы	Всего, академических часов		
	Заочная форма обучения		
	4 семестр	5 семестр	6 семестр
Общая трудоемкость дисциплины (зачетных единиц/ академических часов)	1/36	3/108	3/108
Контактная работа с преподавателем:	**4**	**12**	**8**
занятия лекционного типа	4	4	-
занятия семинарского типа, в т.ч.	-	8	8
семинары, практические занятия	-	8	8
лабораторные работы	-	-	-
курсовое проектирование (КР или КП)	-	-	-
КСР	-	-	-
другие виды контактной работы	-	-	-
Самостоятельная работа	**32**	**92**	**91**
Форма текущего контроля (Кр, реферат, РГР, эссе)	**-**	**2 КР**	**3 КР**
Форма промежуточной аттестации (КР, КП, зачет, экзамен)	-	Зачёт (4)	Экзамен (9)

4. Содержание дисциплины.

4.1. Разделы дисциплины и виды занятий.

№ п/п	Наименование раздела дисциплины	Занятия лекционного типа, акад. часы	Занятия семинарского типа, академ. часы		Самостоятельная работа, акад. часы	Формируемые компетенции	Формируемые индикаторы
			Семинары и/или практические занятия	Лабораторные работы			
4 семестр:							
1.	Введение	0,5	-	-	4	ОПК-4	ОПК-4.3
2.	Алгоритмы и структуры данных	1,5	-	-	4	ПК-1	ПК-1.16
3.	Процесс проектирования	0,5	-	-	6	ПК-1	ПК-1.15 ПК-1.16
4.	Жизненный цикл программ	0,5	-	-	10	ОПК-4 ПК-2	ОПК-4.3 ПК-2.2
5.	Методология программирования	0,5	-	-	4	ПК-1	ПК-1.16
6.	Интерфейсы	0,5	-	-	4	ПК-1 ПК-2	ПК-1.16 ПК-2.2
5 семестр:							
2.	Алгоритмы и структуры данных	-	4	-	20	ПК-1	ПК-1.16
3.	Процесс проектирования	-	2	-	10	ПК-1	ПК-1.15 ПК-1.16
5.	Методология программирования	-	2	-	6	ПК-1	ПК-1.16
6.	Интерфейсы	-	-	-	24	ПК-1 ПК-2	ПК-1.16 ПК-2.2
7.	Веб-программирование	1	-	-	-	ПК-2	ПК-2.1
8.	Многопроцессное и многопоточное программирование	0,75	-	-	-	ПК-2	ПК-2.1
9.	Сетевое программирование	0,75	-	-	-	ПК-2	ПК-2.1
10.	Тестирование и отладка	1	-	-	20	ПК-2	ПК-2.1
11.	Документирование и стандартизация	0,5	-	-	12	ПК-2	ПК-2.2
6 семестр:							
7.	Веб-программирование	-	2	-	20	ПК-2	ПК-2.1
8.	Многопроцессное и многопоточное программирование	-	2	-	20	ПК-2	ПК-2.1
9.	Сетевое программирование	-	2	-	20	ПК-2	ПК-2.1
10.	Тестирование и отладка	-	2	-	20	ПК-2	ПК-2.1
11.	Документирование и стандартизация	-	-	-	11	ПК-2	ПК-2.2

4.2. Занятия лекционного типа.

№ раздела дис-	Наименование темы и краткое содержание занятия	Объем, акад. ча-	Инновационная форма
1	Введение Основные понятия. Программно-технический комплекс. Технологии программирования. Понятие обеспечения системы. Виды обеспечения. Понятие программы, подпрограммы, сопрограммы. Свойства программы как объекта системы. Понятие составляющих программного обеспечения: программный компонент, комплекс программ, пакет программ, программный модуль, библиотека программ, программная система, сборка программ, программный продукт, программная услуга. Понятия: предметная область, конфигурация, окружение, ресурс, среда, задача, функция, функциональность, спецификация, уровень, и т.д. Классификация программ по назначению и по выполнению. тестирования: терминология тестирования, различия тестирования и отладки, фазы и технология тестирования. Критерии выбора тестов: структурные, функциональные, стохастические, мутационный, оценки покрытия проекта. Разновидности тестирования: модульное, интеграционное, системное, регрессионное, автоматизация тестирования, издержки тестирования. Тестирования инсталляции. Особенности процесса и технологии индустриального тестирования.	0,5	
2	Алгоритмы и структуры данных Алгоритмы сортировки и порядковая статистика (Пирамидальная сортировка, быстрая сортировка, сортировка за линейное время, медиана и порядковые статистики). Структуры данных: простые структуры (стек, списки, очередь, корневые деревья, хеш-таблицы, бинарные и красно-черные деревья), сложные структуры данных (B-деревья). Алгоритмы для работы с графами.	1,5	

№ раздела дис-	Наименование темы и краткое содержание занятия	Объем, акад. ча-	Инновационная форма
3	Процесс проектирования Понятие программы как изделия и основные задачи проектирования. Стадии и этапы проектирования. Перечень и содержание работ стадий и этапов. Организация процесса проектирования программного обеспечения. Методы проектирования: сверху-вниз, снизу-вверх. Схемы проектирования: каскадная, откатная, спиральная. Распределение работ между участниками проектов. Взаимодействие участников в процессе проектирования. Постановка задачи на проектирование программного обеспечения. Понятие требования к программному обеспечению. Свойства требований. Сбор, формулировка, анализ, и документирование требований к программному обеспечению. Исследование и анализ предметной области. Техническое задание (ТЗ) на разработку программного обеспечения. Состав и содержание технического задания.	0,5	
4	Жизненный цикл программ Понятие жизненного цикла. Модель жизненного цикла программы. Процессы жизненного цикла: основные (заказ, поставка, разработка, эксплуатация, сопровождение), вспомогательные (документирование, управление конфигурацией, обеспечение качества, верификация, аттестация, совместный анализ, аудит, решение проблем), организационные (управление проектом, создание инфраструктуры, совершенствование процессов, обучение). Состав и содержание работ каждого процесса.	0,5	
5	Методология программирования Понятие и виды программирования. Внутренняя структура и сегментация программ. Методологии программирования: процедурная, структурная, функциональная, логическая, объектно-ориентированная, визуальная, обобщенная. Понятие языка программирования. Классификация языков программирования. Уровни языка. Базовые составляющие языка программирования. Тенденции развития языков программирования.	0,5	
6	Интерфейсы Понятие интерфейса. Классификация интерфейсов. Формы представления: текстовые и графические. Способы организации: командные, диалоговые, оконные, языковые. Сценарии взаимодействия: жёсткие, свободные, иерархические, прямого манипулирования. Виды диалогов: директивный, фразовый, табличный. Этапы разработки интерфейса. Эргономические требования к организации интерфейсов.	0,5	

№ раздела дис-	Наименование темы и краткое содержание занятия	Объем, акад. ча-	Инновационная форма
7	Веб-программирование Особенности языка PHP и его отличия от языка C. Интеграция языков PHP и HTML. Основные типы данных. Операции. Работа со строками. Прикладные библиотеки: растровая графика и взаимодействие с СУБД mySQL. Примеры программ. Особенности языка Node.js и его отличия от языка php и C++. Интеграция языков javascript и HTML. Разработка простого Web-сервера. Особенности языка программирования Java и его отличия от языка C++. Пакеты, классы, объекты. Простые типы данных, строки, массивы. Операторы. Исключения и их обработка. События. Разработка распределенных приложений средствами CORBA. Примеры программ.	1	
8	Многопроцессное и многопоточное программирование Понятие процесса, его свойства и атрибуты. Цикл жизни процесса. Порождение и завершение процессов. Сигналы и реакция на них. Сигнальный механизм для синхронизации работы процессов. Обмен данных через программные каналы. Примеры программ. Обзор методов межпроцессного взаимодействия. Сообщения, семафоры и разделяемая память: создание, управление доступом, организация обмена, удаление. Отображение файлов в оперативную память. Примеры программ. Многопроцессорные ЭВМ. Структура и методика использования потоков выполнения. Создание потоков и их завершение. Объекты синхронизации потоков управления: взаимоисключающие блокировки, условные переменные, семафоры. Примеры программ.	0,75	
9	Сетевое программирование Архитектура стека протоколов TCP/IP. Организация сетевого взаимодействия посредством механизма “гнезд” (sockets). Потоковые и дейтаграммные гнезда. Создание гнезд, привязка адреса, запросы на прием/передачу данных, завершение соединений. Примеры программ. Механизм передачи сообщений согласно прикладному протоколу MPI (Message-Passing Interface). Взаимодействие “один к одному” и коллективное взаимодействие. Синхронные и асинхронные операции. Представление в сообщениях разнотипных данных. Группирование процессов. Виртуальные топологии процессов. Средства выполнения и мониторинга MPI-программ. Примеры программ.	0,75	

№ раздела дис-	Наименование темы и краткое содержание занятия	Объем, акад. ча-	Инновационная форма
10	Тестирование и отладка Понятие ошибки. Основные виды ошибок: синтаксические, алгоритмические, структурные, концептуальные. Понятия тестирования. Основные понятия: случай, контрольные данные, тест-план, протокол, покрытие. Концепции чёрного, серого, белого ящика при тестировании. Методы тестирования: индукции, дедукции, ручной, обратного прослеживания. Составление тестовых планов. Поиск ошибок в программном коде. Отладка.	1	
11	Документирование и стандартизация Документирование программного обеспечения. ЕСПД и её содержание. Основы составления руководства пользователя и руководства администратора. Разработка справочных систем в формате HTML/CHM. Оформление и комментирование программного кода. Подготовка презентаций на программное обеспечение.	0,5	

4.3. Занятия семинарского типа.

4.3.1. Семинары, практические занятия.

№ раздела дисциплины	Наименование темы и краткое содержание занятия	Объем, акад. часы: всего	Инновационная форма
2	Алгоритмы и структуры данных Разбор программной реализации алгоритмов сортировки (сортировка выбором, сортировка пузырьком, сортировка вставками, сортировка слиянием, сортировка Шелла, быстрая сортировка). Разбор программной реализации структур данных (стеки, очереди, связанные списки, корневые дерьевья, бинарные деревья, B-деревья).	4	Слайд-презентация, групповая дискуссия
3	Процесс проектирования Разбор примеров постановки задач на проектирование программного обеспечения. Анализ технического задания на разработку программного обеспечения, его состав и содержание.	2	Слайд-презентация
5	Методология программирования Ознакомление с методологиями программирования (процедурная, структурная, функциональная, логическая, объектно-ориентированная, визуальная, обобщенная).	2	Слайд-презентация
7	Веб-программирование Разбор примеров программ на PHP. Разбор примера простого web-сервера на Node.js. Разбор примеров программ на Java.	2	Слайд-презентация, групповая дискуссия
8	Многопроцессное и многопоточное программирование Разбор примеров многопроцессных и многопоточных программ.	2	Слайд-презентация, групповая дискуссия
9	Сетевое программирование Разбор примеров программ с сетевым взаимодействие посредством sockets. Разбор примеров MPI-программ.	2	Слайд-презентация, групповая дискуссия
10	Тестирование и отладка Ознакомление с методами тестирования. Разбор примеров поиска ошибок в программном коде и отладке.	2	Слайд-презентация

4.3.2. Лабораторные работы.

Не предусмотрены.

4.4. Самостоятельная работа обучающихся.

№ раздела дисциплины	Перечень вопросов для самостоятельного изучения	Объем, акад. часы	Форма контроля
1	Введение Терминология, используемая в разработке программных систем. Классификация программ по назначению и по выполнению.	4	Защита контрольной работы №1-5
2	Алгоритмы и структуры данных Хеширование и хеш-таблицы. Бинарные деревья. Красно-черные деревья. Динамические порядковые статистики. Деревья отрезков. B-деревья.	24	Защита контрольной работы №1-2.
3	Процесс проектирования Методы проектирования программных систем. Распределение работ между участниками проектов. Требования, предъявляемые к программному обеспечению. Требования в SWEBOOK. Требования в RUP. Схема Лефингвелла. IEEE 830. ГОСТ 34 серии.	16	Защита контрольной работы №1-5
4	Жизненный цикл программ ГОСТ 34.601-90. ISO/IEC 12207:2008. Модель жизненного цикла программного обеспечения. Процессы жизненного цикла программного обеспечения. Стадии жизненного цикла программного обеспечения.	10	Защита контрольной работы №1-5
5	Методология программирования Методология императивного программирования. Методология ООП. Методология функционального программирования. Методология логическое программирование. Методология программирования в ограничениях. Классификация языков программирования.	10	Защита контрольной работы №1-5
6	Интерфейсы Виды интерфейсов пользователя. Эргономические требования к организации интерфейсов. Текстовые редакторы. Типовая структура интерфейса. Графические процессоры. Электронные таблицы.	28	Защита контрольной работы №1-5
7	Веб-программирование Синтаксис языка PHP. Переменные. Управляющие структуры. Массивы. Функции пользователя. Объектная модель. Особенности языка Node.js. Свойства языка Java. Отличия Java от C. Классы и объекты в Java.	20	Защита контрольной работы №3

№ раздела дисциплины	Перечень вопросов для самостоятельного изучения	Объем, акад. часы	Форма контроля
8	Многопроцессное и многопоточное программирование Программирование для систем с разделяемой памятью. Программирование для систем с передачей сообщений. Многопоточность на C++. Семафоры. Приоритеты. Очереди, FIFO, LIFO и многопоточность.	20	Защита лабораторной работы №4
9	Сетевое программирование Сетевые серверы. Сетевые клиенты. Архитектура стека протоколов TCP/IP. Организация сетевого взаимодействия посредством механизма “гнезд” (sockets). Протокол MPI.	20	Защита контрольной работы №5
10	Тестирование и отладка Стандартный глоссарий терминов, используемых в тестировании программного обеспечения. Классификация видов тестирования. Функциональные виды тестирования. Нефункциональные виды тестирования. Связанные с изменениями виды тестирования. Уровни тестирования программного обеспечения.	40	Защита контрольной работы №1-5
11	Документирование и стандартизация Документация, используемая в разработке программного обеспечения. Документ руководство пользователя и руководства администратора. Средства разработка справочных систем. Оформление и комментирование программного кода.	23	Защита контрольных работ №1-5

4.4.1 Темы контрольных работ

В плане предусмотрено выполнение студентами 6 контрольных работ. Контрольные работы соответствуют базовым темам курса «Разработка программных систем».

Выполнение 1-ой контрольной работы включает ответы на тестовые вопросы, развёрнутые ответы, содержащие аналитический обзор и анализ разделов, выносимых на самостоятельную работу по разделам 1 и 2.

Цель 1-ой контрольной работы включает ознакомление со средой программирования, приобретение навыков разработки алгоритмов ветвления, декомпозиции задач и составления тест-планов.

Пример выполнения

Контрольная работа 1

Задание:

Даны фигуры, которые разбивают плоскость на области: парабола с вершиной в точке (4,8) и пересекающая ось Ox в точках (0,0) и (8,0). В точке (4,8) находится центр окружности с радиусом r=4. Прямая проходит параллельно оси Oy пересекает ось Ox в точке (4,0). Составить алгоритм, который определяет, в какую из имеющихся на плоскости областей попадает точка с заданными координатами (x,y) и выводом соответствующего сообщения.

Графическая модель:

Построим фигуры и пронумеруем области. Графики с указанием номеров областей представлены на рисунке 1.1:

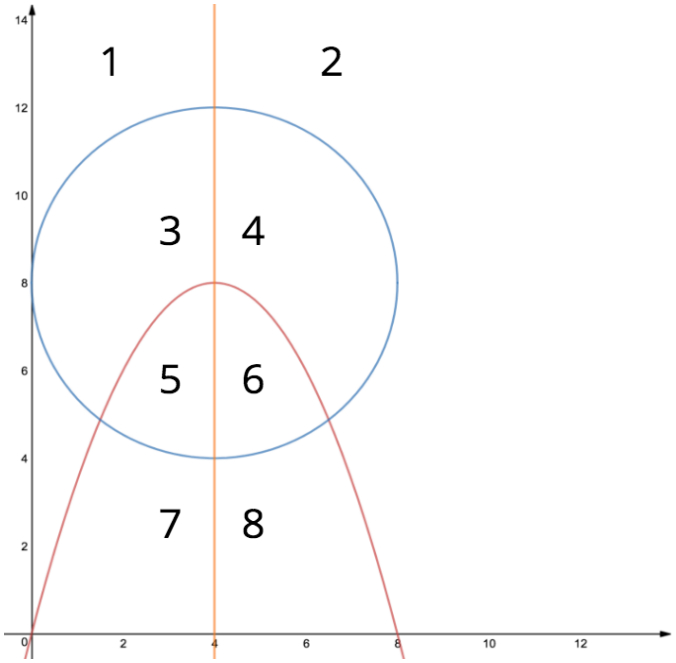

Рисунок 1.1 – Графики с указанием номеров областей

Математическая модель:

Получим математические уравнения кривых и линий, образующих фигуры на плоскости:

Таблица 1.1 – математические уравнения кривых и линий, образующих фигуры на плоскости

1	парабола	$Y = -0.5 * X^2 + 4 * X$
2	окружность	$(X - 4)^2 + (Y - 8)^2 = 16$
3	прямая	$X = 4$

Составим условия проверки попадания точки с координатами (Xz, Yz) в каждую из фигур, изображенных на рисунке 1.1:

Таблица 1.2 – условия проверки попадания точки с координатами (Xz, Yz)

1	внутри окружности	$(X_z - 4)^2 + (Y_z - 8)^2 \leq 16$
2	снаружи окружности	$(X_z - 4)^2 + (Y_z - 8)^2 > 16$
3	внутри параболы	$Y_z + 0.5 * X_z^2 - 4 * X_z \leq 0$
4	снаружи параболы	$Y_z + 0.5 * X_z^2 - 4 * X_z > 0$
5	слева от прямой	$X_z \leq 4$
6	справа от прямой	$X_z > 4$

Составим условия проверки попадания точки в каждую из областей, изображенных на рис. 1.1:

Таблица 1.3 – условия проверки попадания точки в каждую из областей

№	Окружность	Парабола	Слева от прямой	Справа от прямой	Доп. Условия
1	Нет	Нет	Да	Нет	
2	Нет	Нет	Нет	Да	
3	Да	Нет	Да	Нет	
4	Да	Нет	Нет	Да	
5	Да	Да	Да	Нет	
6	Да	Да	Нет	Да	
7	Нет	Да	Да	Нет	
8	Нет	Да	Нет	Да	

Тест-план:

Для тестирования программы необходимо выбрать по точке в каждой из областей, используя следующие данные:

Таблица 1.4 – координаты точек, в каждой из областей

Тест	X_z	Y_z	Результат
1	2.0	13.0	Область №1
2	6.0	12.5	Область №2
3	2.0	8.0	Область №3
4	6.0	8.7	Область №4
5	3.0	6.0	Область №5
6	5.0	5.3	Область №6
7	1.9	1.9	Область №7
8	7.0	1.5	Область №8
9	ab	1.0	Incorrect X-value
10	1.0	ab	Incorrect Y-value
11	-3.5	-1.0	Область №1
12	12.0	8.0	Область №2
13	2.5	-4.0	Область №7
14	5.0	-1.0	Область №8

Декомпозиция задач:

Исходя из условий задания, в основной задаче Main можно выделить самостоятельную задачу CheckArea – проверка попадания точки в одну из областей на плоскости. Эту задачу можно разбить на три независимые подзадачи:

IsCurcle – проверка попадания точки внутрь окружности

IsParabola – проверка попадания точки внутрь параболы

IsLeftSide – проверка попадания точки с левой стороны от прямой

Дерево декомпозиции задач:

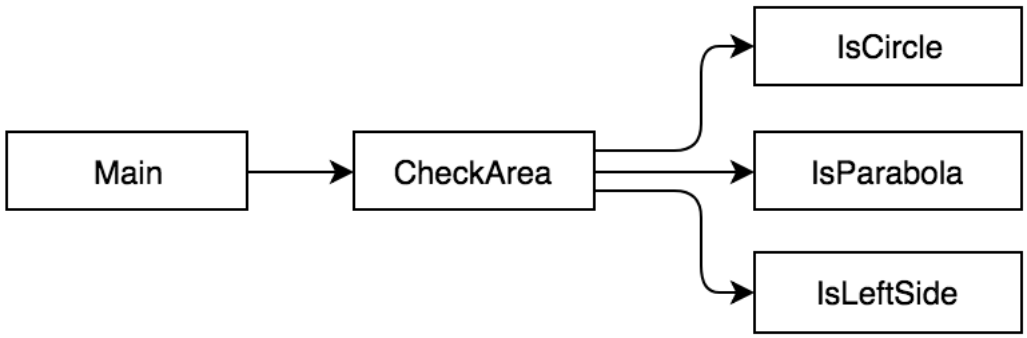

Рисунок 1.2 – Дерево декомпозиции задач

Блок-схемы и описание алгоритмов:

Алгоритм **IsCircle** получает координаты точки, как исходные данные, проверяет условия 1 и 2, и возвращает **True**, если точка лежит внутри окружности, или **False**, если точка лежит снаружи окружности.

Алгоритм **IsParabola** получает координаты точки, как исходные данные, проверяет условия 3 и 4, и возвращает **True**, если точка лежит внутри параболы, или **False**, если точка лежит снаружи параболы.

Алгоритм **IsLeftSide** получает координаты точки, как исходные данные, проверяет условие 5, и возвращает **True**, если точка лежит слева от прямой или на прямой, или False, если точка лежит справа от прямой.

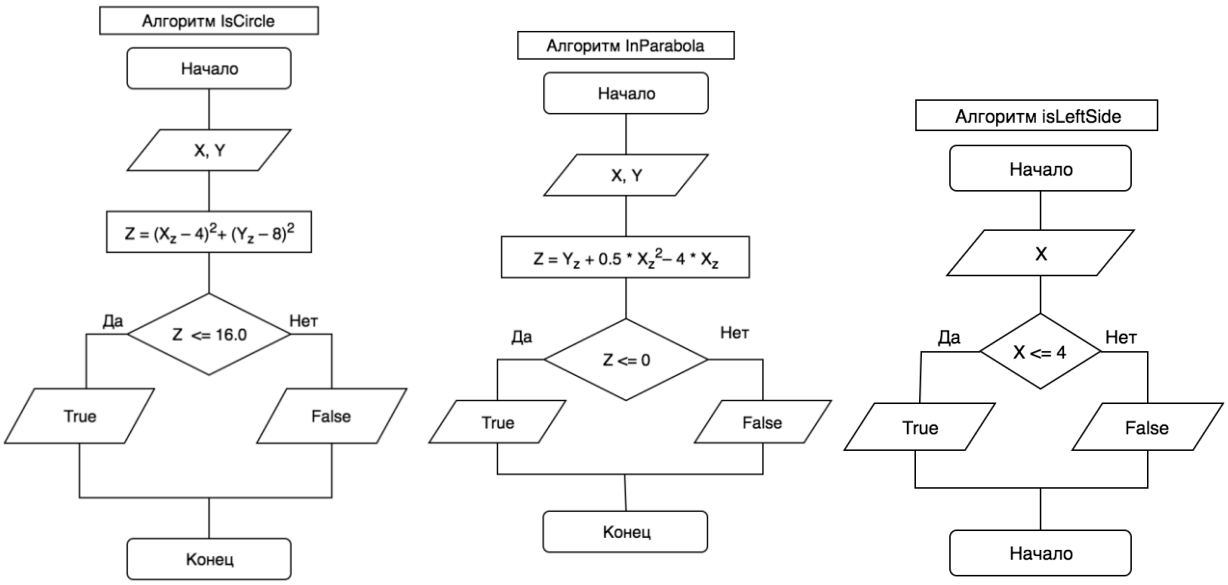

Рисунок 1.3 – Алгоритмы IsCircle, IsParabola, IsLeftSide

Алгоритм CheckArea

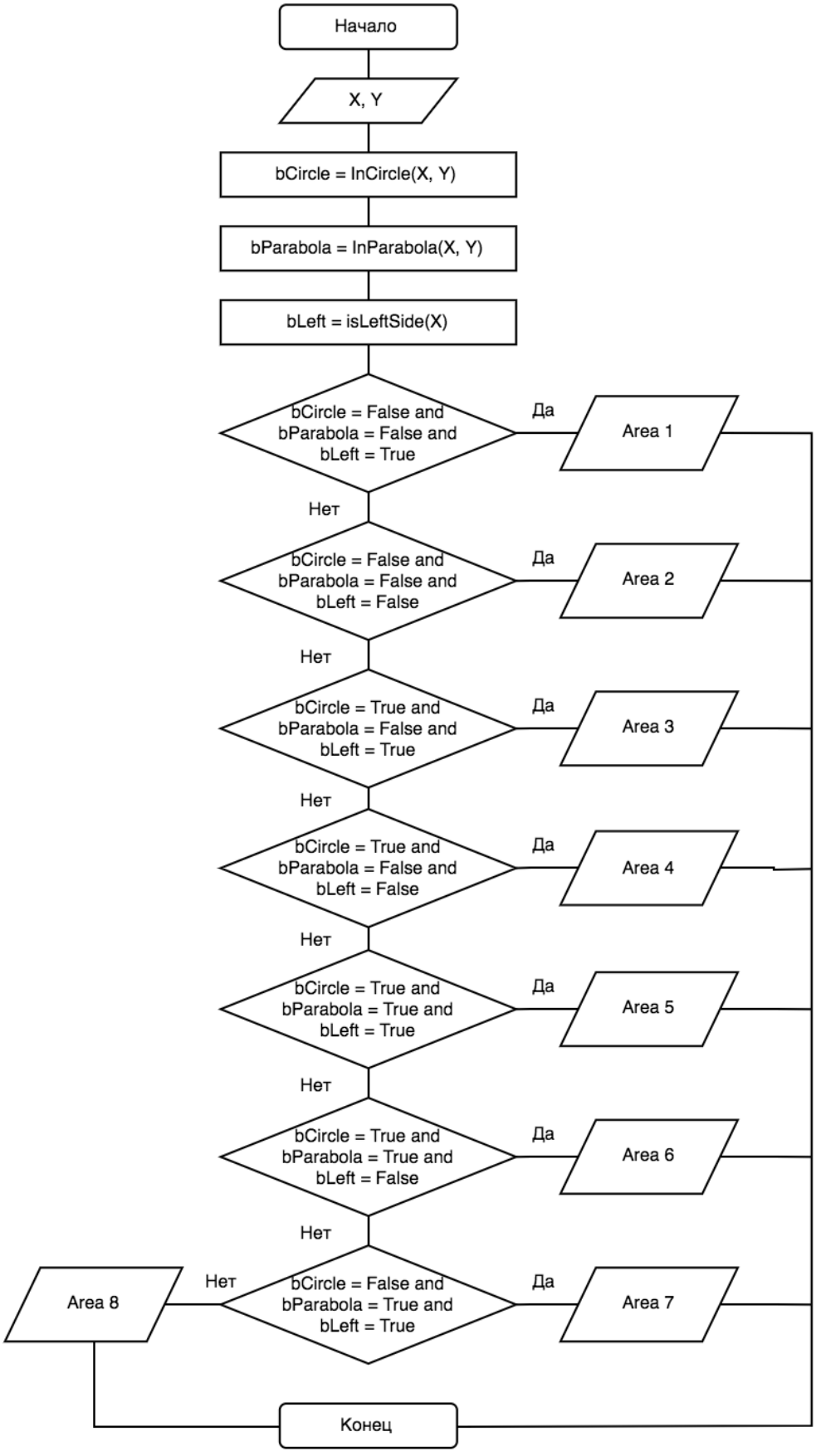

Рисунок 1.4 – Алгоритм CheckArea

Алгоритм **CheckArea** получает координаты точки, как исходные данные, и использует алгоритмы **IsCircle, IsParabola, IsLeftSide** для проверки попадания точки в каждую из фигур по отдельности и, затем, осуществляет блок последовательных проверок в соответствии с таблицей условий по областям для проверки попадания точки в каждую из областей. Область 8 определяется как исключение, если ни одно из условий не выполнилось.

Листинг программы:

```
#include <iostream>
#include <locale.h>
using namespace std;
/*
 Функция проверяет, находится ли точка с координатами (x,y)
 внутри параболы (true) или снаружи параболы (false)
*/
bool isParabola(double x, double y) {
    return (y + 0.5 * (x * x) - 4 * x) <= 0;
}

/*
 Функция проверяет, находится ли точка с координатами (x,y)
 внутри окружности (true) или снаружи окружности (false)
*/
bool isCircle(double x, double y) {
    return (x - 4) * (x - 4) + (y - 8) * (y - 8) <= 16;
}

/*
 Функция проверяет, находится ли точка (x,y) слева от прямой   (true) или справа
(false),  при этом, если точка лежит на прямой  (x=4), считается, что она слева
*/
bool isLeftSide(double x) {
    return x <= 4;
}

/*
 Функция проверяет, какой области принадлежит точка
 с координатами (x,y) и возвращает номер области от 1 до 8
*/
int checkArea(double x, double y) {
    bool bCircle = isCircle(x, y);
    bool bPrabola = isParabola(x, y);
    bool bLeftSide = isLeftSide(x);

    if (!bCircle && !bPrabola && bLeftSide) {
        return 1;
    }

    if (!bCircle && !bPrabola && !bLeftSide) {
        return 2;
```

```
    }

    if (bCircle && !bPrabola && bLeftSide) {
        return 3;
    }

    if (bCircle && !bPrabola && !bLeftSide) {
        return 4;
    }

    if (bCircle && bPrabola && bLeftSide) {
        return 5;
    }

    if (bCircle && bPrabola && !bLeftSide) {
        return 6;
    }

    if (!bCircle && bPrabola && bLeftSide) {
        return 7;
    }

    // !bCircle && bPrabola && !bLeftSide
    return 8;
}

/*
 Главная функция. Реализует интерфейс с пользователем.
*/
int main(void) {
    char ch;
    double x,y;
    do {
        cout << "\nEntry X: ";
        cin >> x;

        if(cin.fail()) {
            cin.clear();
            cout << "Incorrect X-value";
        } else {
            cout << "\nEntry Y: ";
            cin >> y;

            if(cin.fail()) {
                cin.clear();
                cout << "Incorrect Y-value";
            } else {
                cout << "\nPoint is placed in area: " << checkArea(x,y);
            }
        }
        cin.sync();
        cout << "\nDo you want to entry a new point [Y/N]?";
```

```
            cin.get(ch);
        } while((ch == 'Y') || (ch == 'y'));
    }
```

Целью контрольной работы №2 является изучение среды программирования, отладочных средств, циклических конструкций языка.

Пример выполнения

Контрольная работа 2

Задание:

Разработать алгоритмы, распознающие и распечатывающие при вводе те числа, сумма цифр которых является треугольным числом и разность между соседними цифрами равна *k*. Например:

357 → 3+5+7=15 (*k*=2); 678 → 6+7+8=21 (*k*=1), ...

Математическая модель:

Треугольные числа получаются по формуле:

$$T_n = \frac{n(n+1)}{2}$$

Последовательность чисел для n = 0, 1, 2 … начинается так:

0, 1, 3, 6, 10, 15, 21, 28, 36, 45, …

Тестовые планы:

Тестирование программы предлагается разбить на три части:

1. Тестирование печати чисел в заданном диапазоне

Вызвать функцию печати чисел, обладающих свойством, для заданного диапазона и использовать следующие данные:

Таблица 2.1 – Данные для вызова функции печати чисел

Тест	От	До	К	Результат
1	10	1000	2	24, 42, 46, 64, 357, 420, 424, 579, 753, 975
2	10	1000	4	15, 37, 51, 73, 159, 262, 951, 5959, 9595
3	10	1000	6	28, 60, 82, 393, 717, 939
4	10	1000	8	19, 91
5	100	500	3	141, 258, 303, 474
6	100	500	4	159, 262
7	100	500	6	393
8	10000	100000	3	25858, 30363, 30369, 36303, 36363, 69696, 85258, 85852, 96303
9	10000	100000	6	82828
10	10000	100000	8	19191
11	-90000	-40000	3	-85852, -85258, -69696
12	100000	60000	7	70707
13	10	ff		Incorrect upper bound value
14	ff			Incorrect lower bound value
15	10	100	A	Incorrect digit difference value
16	10	100	9	Incorrect digit difference value
17	10	100	-1	Incorrect digit difference value

2. Тестирование последовательности ввода чисел

В качестве тестовых последовательностей использовать следующие данные (жирным выделены числа, обладающие свойством):

Таблица 2.2 – Тестовые последовательности

Тест	K	Последовательность чисел											N	M
1	2	525	**357**	67	83	887	173	**42**	**424**	473	59	0	3	7
2	4	732	28	92	691	**159**	238	44	213	22	**37**	0	2	8
3	3	45	73	83	37	873	746	689	453	272	62	0	0	10
4	1	**24**	**42**	**46**	**64**	**357**	**420**	**424**	**579**	**753**	**975**	0	10	0
5	1	83	473	dg	**753**	59	0	123	jj	**64**	zz	0	2	4
6	8											0		

Тесты 1 и 2 – последовательности, содержащие числа, обладающие и не обладающие свойством. Тест 3 – последовательность, не содержащая чисел, обладающих свойством. Тест 4 – последовательность, содержащая только числа, обладающие свойством. Тест 5 – последовательность, содержащая ошибочные данные. Тест 6 – пустая последовательность (сразу два нуля)

3. Тестирование печати ряда треугольных чисел

Вызвать функцию печати треугольных чисел. Результат:

1, 3, 6, 10, 15, 21, 28, 36, 45, 55, 66, 78, 91, 105, 120, 136, 153, 171, 190, 210, 231, 253, 276, 300, 325, 351, 378, 406, 435, 465, 496, 528, 561, 595, 630, 666, 703, 741, 780, 820, 861, 903, 946, 990, 1035, 1081, 1128, 1176, 1225 …

Декомпозиция задач:

Исходя из условий задания, главную задачу Main можно разбить на две основных задачи:

CheckNumber – проверка свойств чисел,

Entry&Print – ввод данных и печать результатов.

Задачу **CheckNumber** можно разбить на три независимые подзадачи:

1 – проверка числа на принадлежность к ряду треугольных чисел,

2 – проверка разности между цифрами,

3 – вычисление суммы цифр.

Задачу **Entry&Print** можно разбить на три независимые подзадачи:

1 – печать ряда треугольных чисел,

2 – печать в заданном диапазоне чисел, удовлетворяющих свойствам,

3 – определение и подсчёт чисел, удовлетворяющих свойствам, при вводе, а условием прекращения ввода является ввод двух нулей подряд.

Для **CheckNumber** предлагается использовать алгоритмы:

IsTriangleNumber – для проверки, принадлежит ли число к ряду треугольных чисел,

IsDifference– для проверки, равна ли разность между цифрами числа заданному значению,

SumDigits –для вычисления суммы цифр числа.

CheckNumber – для проверки, является сумма цифр числа треугольным числом и разность между соседними цифрами равна k.

Для решения **Entry&Print** предлагается использовать алгоритмы: **PrintTriangleNumbers** – для печати ряда треугольных чисел,

PrintNumbers – для печати в заданном диапазоне тех чисел, которые обладают заданными свойствами,

InputNumbers – для ввода чисел с консоли, проверки их на заданные свойства, подсчёта чисел, пока не будут введены два нуля подряд.

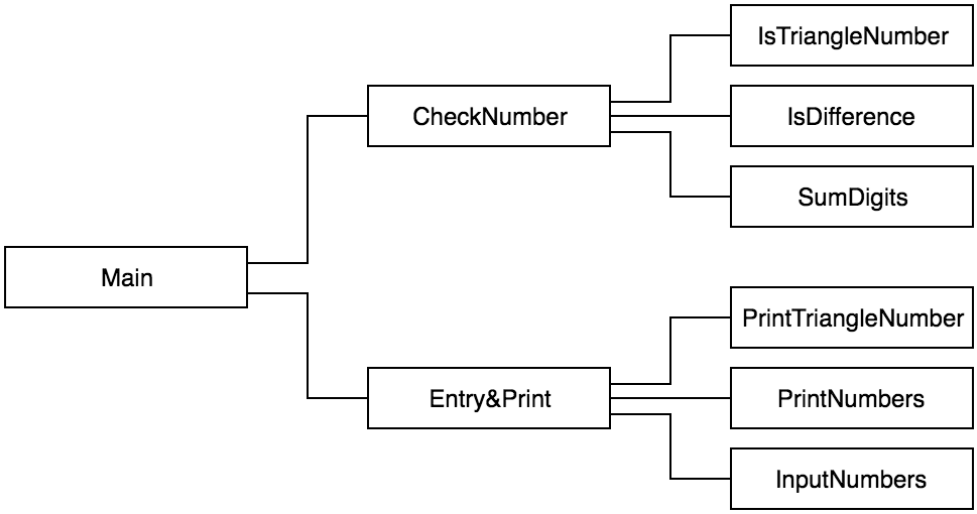

Рисунок 2.1 – Дерево декомпозиции задач

Блок-схемы и описание алгоритмов:

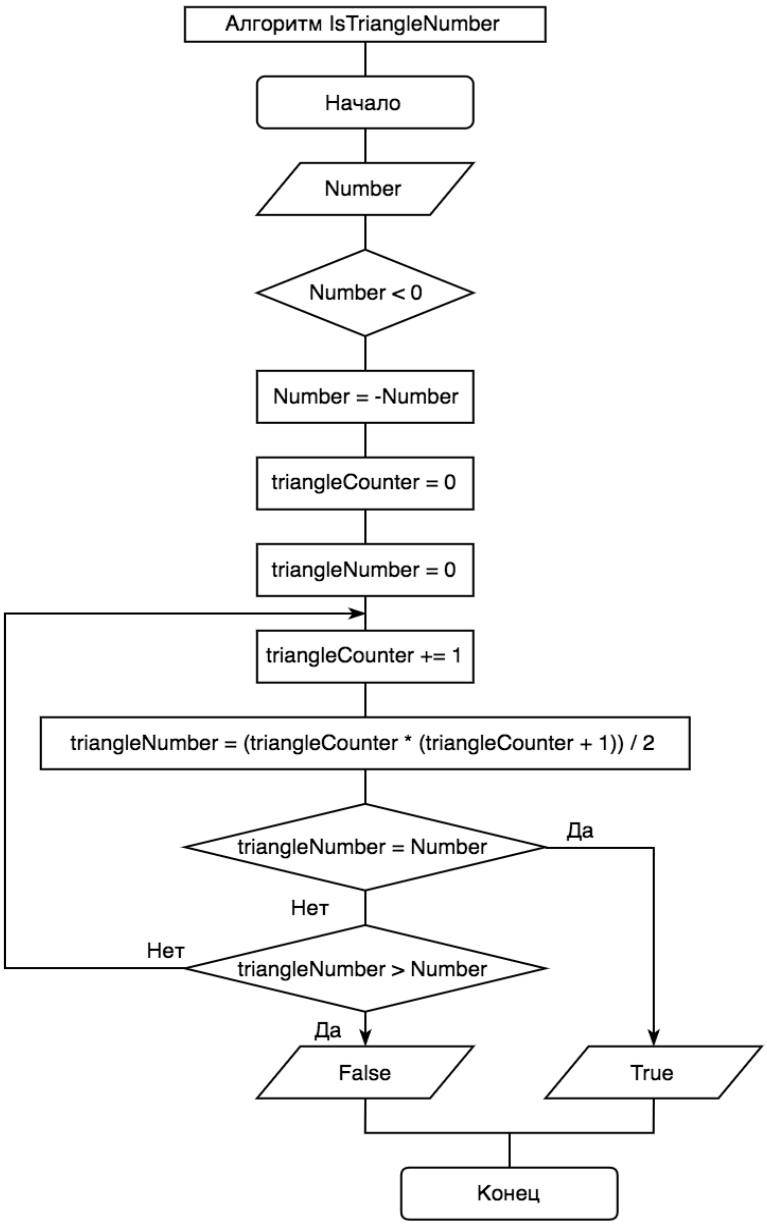

Рисунок 2.3 – Алгоритм IsTriangleNumber

Алгоритм **IsTriangleNumber** получает число для проверки, как исходные данные. Затем он последовательно в цикле вычисляет следующее число ряда треугольных чисел и сравнивает его с числом, переданным как исходные данные. Если текущее треугольное число равно заданному числу, то цикл прекращается и алгоритм возвращает True. Если в процессе вычислений, треугольное число стало больше заданного, то вычисления можно прекратить, и алгоритм возвращает False.

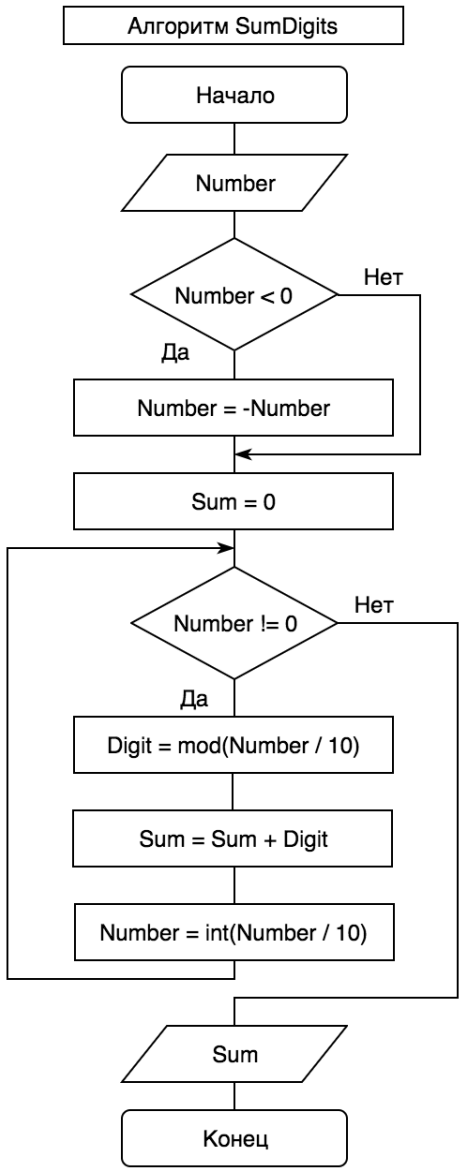

Рисунок 2.3.1 – Алгоритм SumDigits

Алгоритм **SumDigits** получает число для проверки, как исходные данные. Затем, он последовательно в цикле разбивает число на цифры и суммирует их. Чтобы отделить очередную цифру, алгоритм использует операцию вычисления остатка от деления на цело. Чтобы получить число без отделённой цифры, алгоритм использует обычную операцию деления и отбрасывает дробную часть. Цикл выполняется до тех пор, пока в числе не останется цифр. Алгоритм возвращает сумму цифр, как результат.

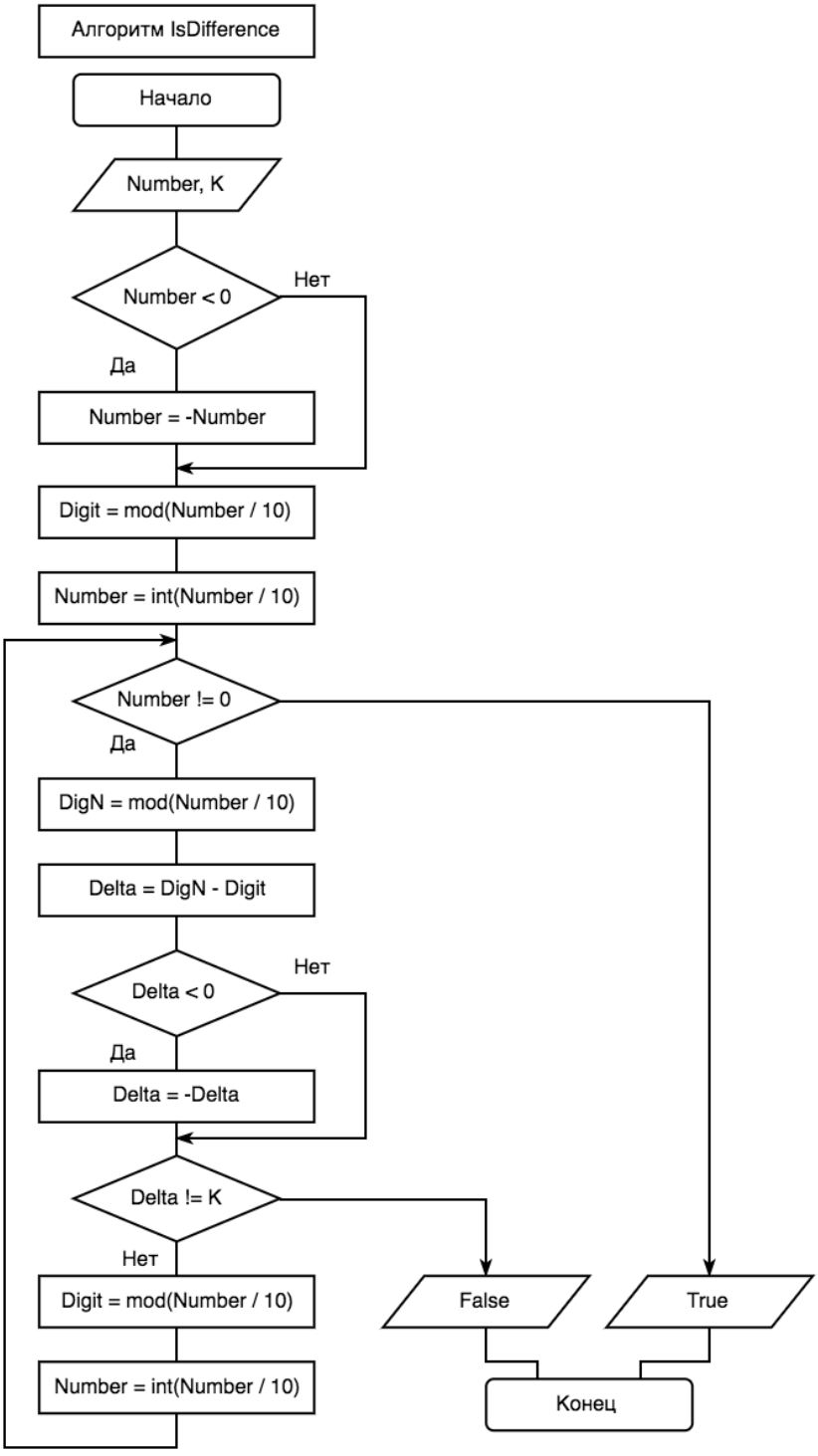

Рисунок 2.5 – Алгоритм IsDifference

Алгоритм **IsDifference** получает число для проверки и разность между цифрами, как исходные данные. Затем, он последовательно в цикле разбивает число на цифры, вычисляет разность между соседними цифрами и сравнивает её с заданной разностью. Чтобы отделить очередную цифру, алгоритм использует операцию вычисления остатка от деления на цело. Чтобы получить число без отделённой цифры, алгоритм использует обычную операцию деления и отбрасывает дробную часть. Цикл выполняется до тех пор, пока в

числе не останется цифр. Если на очередном шаге вычислений разность оказывается больше или меньше заданной, то вычисления можно прекратить, и алгоритм возвращает False, как результат. Если в числе больше не осталось цифр, то цикл прекращается, и алгоритм возвращает True, как результат.

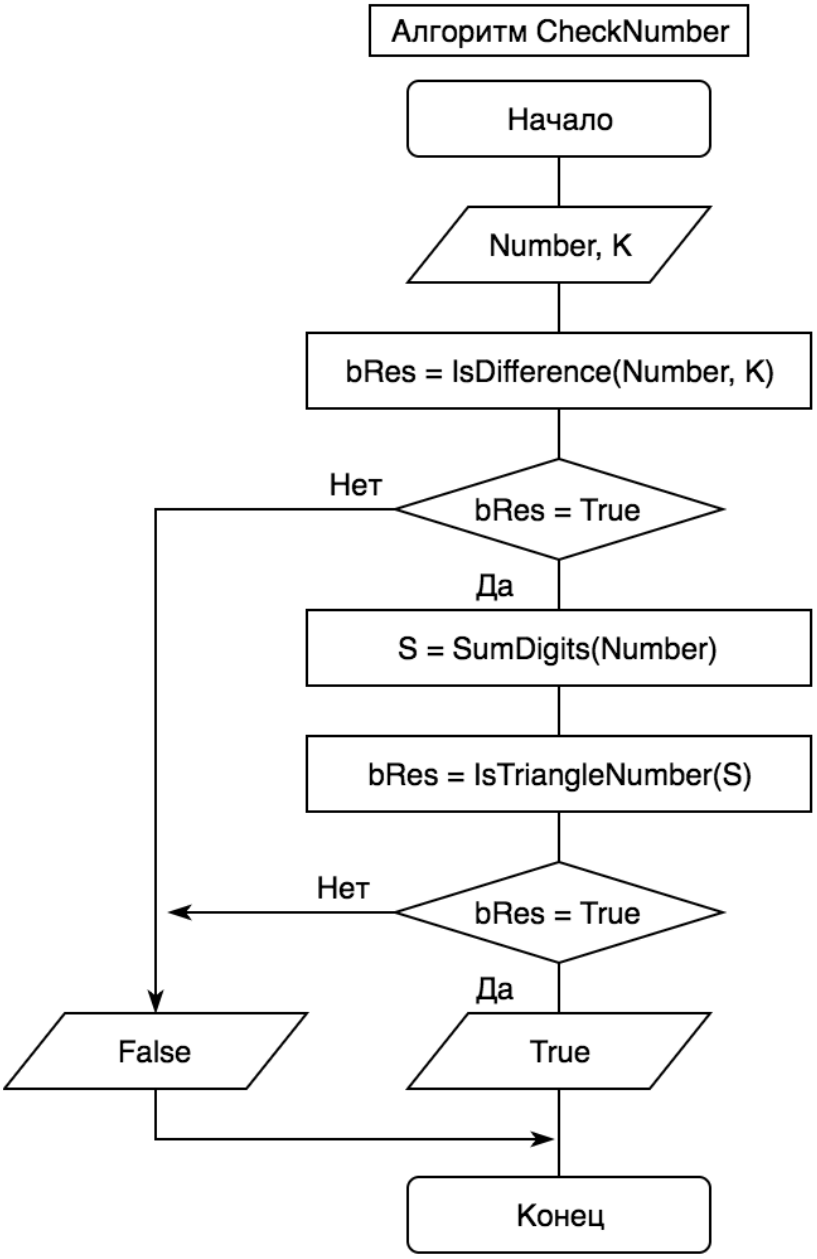

Рисунок 2.4 – Алгоритм CheckNumber

Алгоритм **CheckNumber** получает число для проверки и разность между цифрами, как исходные данные. Затем, он последовательно использует алгоритмы **IsTriangleNumber, IsDifference и SumDigits**, описанные выше, для проверки, принадлежит ли число к ряду треугольных чисел при заданной разности между соседними цифра-

ми. Алгоритм возвращает **True**, если результаты всех проверок оказались **True**, или **False**, если хотя бы одна проверка не выполнилась.

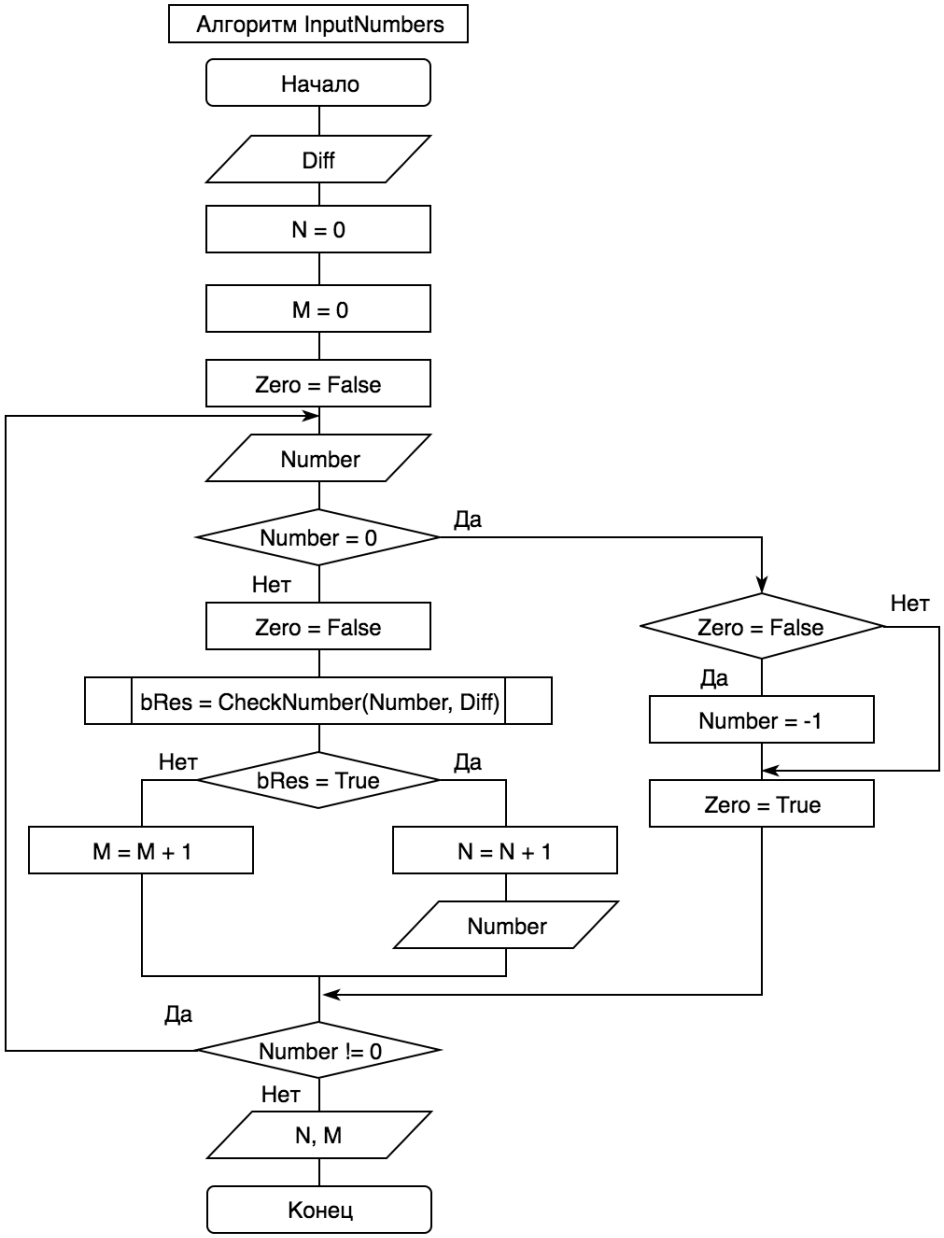

Рисунок 2.6 – Алгоритм InputNumbers

Алгоритм InputNumbers запрашивает только разность между цифрами, количество вводимых чисел ему не известно. Алгоритм организует цикл по вводу чисел с проверкой каждого нового числа до пор, пока пользователь не введет два нуля подряд. Для проверки введенного числа он использует алгоритм CheckNumber. Если введенное число удовлетворяет свойствам, то алгоритм увеличивает счётчик проверяемых чисел (N) на единицу и печатает информацию о числе, если нет, то увеличивает счётчик прочих чисел (M). Чтобы проверить ввод двух нулей подряд, алгоритм использует переменную Zero, которая полу-

чает значение True, всякий раз, когда был введен ноль, и значение False, всякий раз, когда было введено ненулевое значение. Если был введен первый ноль, т.е. Zero еще False, то алгоритм присваивает введенному числу ненулевое значение, чтобы не прекращать цикл.

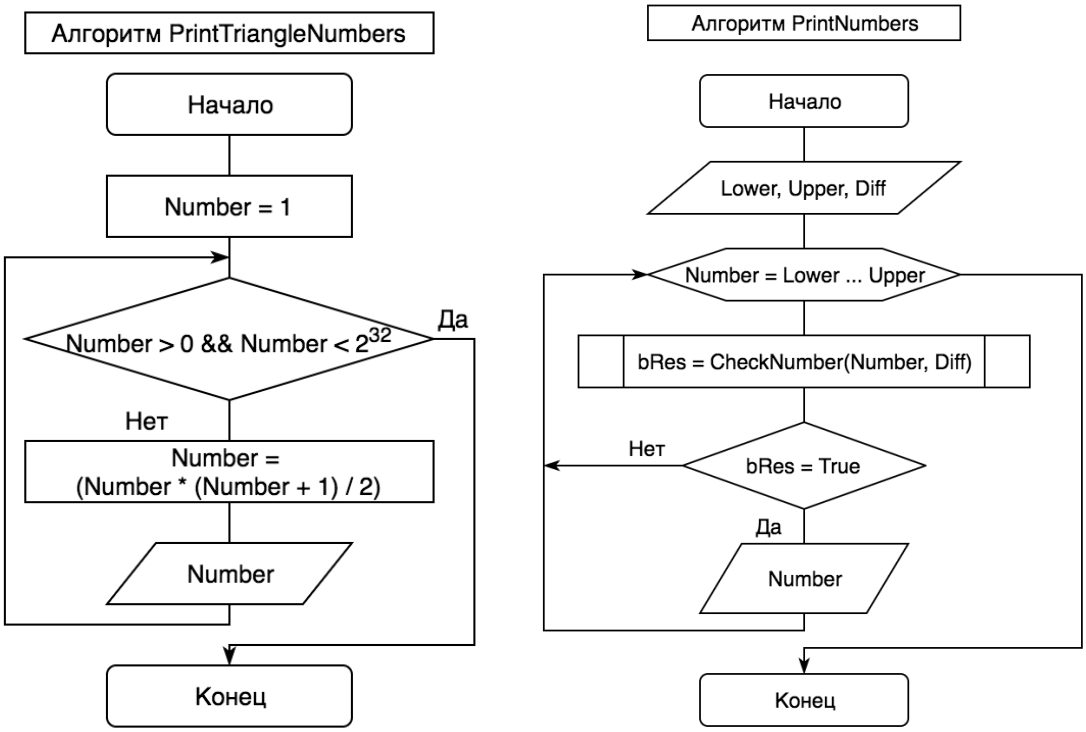

Рисунок 2.7 – Алгоритмы PrintTriangleNumbers и PrintNumbers

Алгоритм **PrintTriangleNumbers** печатает все числа принадлежащие к ряду треугольных чисел, используя цикл от 1 до предельного значения используемого типа данных (long). Цикл выполняется, пока текущее вычисленное значение меньше предела и является положительным.

Алгоритм **PrintNumbers** запрашивает нижнее и верхнее значение диапазона чисел и разность между цифрами. Алгоритм организует цикл от нижнего значения до верхнего значения и использует в цикле алгоритмы **IsTriangleNumber, IsDifference** и **SumDigits**, описанные выше, для проверки, принадлежит ли сумма цифр очередного числа к ряду треугольных чисел при заданной разности между соседними цифрами.

Листинг программы:

```
/*****************************************************************
   Содержание модуля TASK.H
*****************************************************************/
#ifndef task_h
#define task_h
```

```
void InputNumbers();
void PrintNumbers();
void PrintTraingleNumbers();
long SumDigits(long number);
bool IsTriangleNumber(long number);
bool IsDifference(long number, __int8_t k);
bool CheckNumber(long number, __int8_t k);

#endif /* task_h */

/*****************************************************************************
   Содержание модуля SOLVE.CPP
*****************************************************************************/

#include <stdio.h>

/*
 Функция проверят, принадлежит ли число к ряду треугольных чисел
*/
bool IsTriangleNumber(long number) {
    if (number < 0) {
        number = -number;
    }

    int triangleCounter = 0;
    int triangleNumber = 0;
    while (true) {
        triangleCounter++;
        triangleNumber = (triangleCounter * (triangleCounter + 1)) / 2;

        if (triangleNumber == number) {
            return true;
        } else if (triangleNumber > number) {
            return false;
        }
    }
}

/* Функция проверяет разность между цифрами числами Возвращает true – если разность между цифрами равна K
 или false – если хотя бы для одной пары цифр разность неравна K
*/
bool IsDifference(long number, __int8_t k) {
    __int8_t digit, delta;

    if(number < 0)
        number = -number;

    digit = (__int8_t)(number % 10);
    number /= 10;
```

```
    while(number != 0) {
        delta = (__int8_t)(number % 10 - digit);

        if(delta < 0) delta = -delta;
            if(delta != k)
                return false;

        digit = (__int8_t)(number % 10);
        number /= 10;
    }
    return true;
}

/* Функция вычисляет сумму цифр числа */
long SumDigits(long number)
{
    long SumDigits = 0;
    if(number < 0) number = -number;
    while(number != 0) {
        SumDigits += number % 10;
        number /= 10;
    }
    return SumDigits;
}

/*
 Функция проверяет, обладает ли число заданными свойствами
*/
bool CheckNumber(long number, __int8_t k)
{
    return (IsDifference(number, k) && IsTriangleNumber(SumDigits(number)));
}

/*****************************************************************************
    Содержание модуля INTERFACE.CPP
*****************************************************************************/

#include <iostream>
#include <limits.h>
#include "task.h"
using namespace std;

/*
 Функция печатает числа принадлежащие к ряду треугольных чисел
 */
void PrintTraingleNumbers() {
    long number = 0;
    cout << "\nSeries of triangle numbers:\n";
    for (number = 1; number < LONG_MAX && number > 0; number++) {
        cout << (number * (number + 1) / 2) << "\n";
    }
}
```

```
/*
 Функция печатает в заданном диапазоне те числа, сумма цифр которых принадлежит к
ряду треугольных чисел и разность между соседними цифрами числа равна k
 */
void PrintNumbers() {
    int diff;           // Difference between digits
    long lower, upper;  // Lower and Upper of the bound
    long number;        // Current number

    cout << "Entry the lower bound of numbers: ";
    cin >> lower;

    if(cin.fail()) {
        cin.clear();
        cout << "Incorrect lower bound value";
        return;
    }
    cout << "Entry the upper bound of numbers: ";
    cin >> upper;

    if(cin.fail()) {
        cin.clear();
        cout << "Incorrect upper bound value"; return;
    }
    cout << "Entry the digit difference in range [1...8]: ";
    cin >> diff;

    if(cin.fail()) {
        cin.clear();
        diff = -1;
    }

    if(diff < 1 || diff > 8) {
        cout << "Incorrect digit difference value";
        return;
    }

    if(lower > upper) {
        number = lower;
        lower = upper;
        upper = number;
    }

    cout << "List of numbers:";
    for(number=lower ; number <= upper ; number++)
        if(CheckNumber(number, diff))
            cout << "\n" << number;
}
/*
    Функция вводит числа до тех пор, пока не будут введены
    Два нуля подряд, и проверяет каждое число, чтобы сумма
```

```
    цифр была числом принадлежащим к ряду треугольных чисел и разность между сосед-
ними цифрами числа равна k, и подсчитывает их.
*/
void InputNumbers() {
    bool zero = false; // Flag to be finished
    int n = 0, m = 0;  // Counters of numbers
    int diff;          // Difference between digits
    long number;       // Entered number

    cout << "Entry the digit difference in range [1...8]: ";
    cin >> diff;

    if(cin.fail()) {
        cin.clear();
        diff = -1;
    }

    if(diff < 1 || diff > 8) {
        cout << "Incorrect digit difference value";
        return;
    }

    do {
        cin.sync();
        cin >> number;
        if(cin.fail()) {
            cin.clear();
            cout << "Incorrect number value";
        }
        else if(number == 0) {
            if(zero == false)
                number = -1;

            zero = true;
        } else {
            zero = false;
            if(CheckNumber(number, diff)) {
                n++;
                cout << "(" << number << ")\t is (" << n << ")" << "\tsum=" << SumDigits(number)
<< "\n";
            } else m++;
        }
    }
    while(number != 0);
    cout << endl << "Numbers having properties: " << n << endl << "Other numbers: " << m <<
endl;
}
/* Главная функция. Реализует интерфейс с пользователем. */
int main(void)
{
    char ch = '0';
    do {
```

```
        cout << "\nPress [N] to count the numbers";
        cout << "\nPress [P] to print the numbers in the bound";
        cout << "\nPress [D] to print the triangle numbers";
        cout << "\nPress [Q] to quit the program\n?>";

        cin.get(ch);

        if((ch == 'D') || (ch == 'd')) {
           PrintTraingleNumbers();
        } else if ((ch == 'P') || (ch == 'p')) {
           PrintNumbers();
        } else if ((ch == 'N') || (ch == 'n')) {
           InputNumbers();
        }
        cin.sync();
    } while((ch != 'Q') && (ch != 'q'));

    return 0;
}
```

Задание 3-ой контрольной работы включает разработку и отладку динамических WWW-страниц средствами языка PHP. Разрабатываемый сайт представляет краткую информацию об организации, в которой работает студент. Информация на сайте должна легко обновляться, не требуя от менеджера специальных знаний. Изменение или расширение функциональности сайта должно происходить быстро и с небольшими затратами. Сайт должен занимать все пространство экрана по ширине. Высота сайта зависит от объема отображаемой информации. Код страницы должен быть лаконичным, но без ущерба к оформлению и функционалу сайта. Архитектура сайта должна быть проста и интуитивно удобна.

Задание 4ой контрольной работы включает тестовые вопросы по:
- разработке и отладке многопроцессной программы с использованием программных каналов и сигналов и с использованием разделяемой памяти и сообщений;
- разработке и отладке многопоточной программы;
- разработка и отладка сетевой программы;
- разработке и отладке сетевого приложения.

Дополнительно выполняется задание по разработке и отладке простой параллельной программы средствами MPI.

Целью 5-ой контрольной работы является приобретения навыков декомпозиции задач, модульного проектирования программ.

Пример выполнения

Контрольная работа 5

Задание:

Непрерывная последовательность чисел имеет либо минимальную, либо максимальную (*должно выбираться опционально*) сумму разностей между соседними элементами.

Математическая модель:

Для решения задачи предлагается использоваться метод перебора всех возможных непрерывных последовательностей элементов массива длины 1 < M < Size – 1 с вычислением суммы разностей между соседними элементами, которое сравнивается с критериальным значением суммы.

Пусть имеется массив:

0	1	2	3	4	5	6	7	8	9	10
8.0	4.0	15.6	-7.3	89.5	1.0	-6.2	3.0	24.1	50.0	2.2

Для перебора последовательностей необходимо завести два индекса Left и Right. Индекс Left обозначает начало очередной последовательности и сдвигается все время влево к концу массива. Индекс Right обозначает конец последовательности и сдвигается от индекса (Left + 1) влево не более длины

последовательности M. Длины последовательностей перебираются, начиная

от 2 до (Size – Left). Для каждой последовательности вычисляется критерий как $\sum_{j=left}^{j=right} |A_j - A_{j+1}|$

План работы:

Таблица 1 – План работы

№	Вид работы	Время, ч
1	Изучение предметной области, анализ интернет-публикаций, составление технического задания	8
2	Определение функциональности, декомпозиция задач и проектирование архитектуры модулей	8
3	Разработка и написание спецификаций (SRS, AMS, FFS)	6
4	Разработка алгоритмов решения задач, составление блок-схем	20
5	Анализ ситуаций, подбор входных данных, разработка тестовых планов	16
6	Кодирование алгоритмов, написание C++ кода	24
7	Тестирование программы в соответствии с тестовыми планами, отладка модулей, исправление ошибок	10
8	Документирование: API reference, комментарии к коду, оформление отчёта	10

Техническое задание:

Разработать программу поиска в массиве вещественных чисел непрерывной последовательности, удовлетворяющей критерию: максимум или минимум суммы разности элементов.

Требования к функциональности:

- консольный ввод-вывод массива чисел,
- файловый ввод-вывод массива чисел,
- обработка массива чисел для поиска последовательности,
- диагностика ошибок.

Требование к интерфейсу:

- диалоговое консольное меню ко всем функциям,
- наличие подсказок при при вводе данных и выборе команд,
- запрос подтверждений от пользователя в неоднозначных ситуациях,
- вывод понятной информации об ошибках.

Требования к тестированию:

- все функции программы должны иметь тестовые планы,
- программа должна хранить готовые тестовые массивы,
- отработать тестовые случаи при работе с файлами.

Требования к форматам хранения данных:

- разработать формат хранения массива в файле,
- составить спецификацию хранения данных.

Требования к архитектуре:

- программа должна строиться из четырёх модулей, каждый из которых отвечает за опередеенную функциональность программы,
- составить спецификацию на содержание и назначение модулей.

Требования к комплектности и поставке:

- программа должна поставляться в виде zip-архива,
- в поставку должны входить тестовые файлы,
- в поставку должно входить руководство пользователя.

Требования к техническим и программным средствам:

- PC Pentium, не менее 128M ОЗУ,
- ОС Microsoft Windows XP SP1/SP2/SP3/

Декомпозиция задач:

Исходя из условий задания, главную задачу Main можно разбить на четыре основные задачи:

Console I/O – Организация консольного ввода-вывода массива,

File I/O – Организация файлового ввода-вывода массива,

Solve – Обработка массива в соответствии с заданием,

Interface – Организация интерфейса с пользователем.

Задачу Console I/O можно разбить на три независимые подзадачи:

ShowArray – вывод массива на консоль,

InputArray – вывод массива с консоли,

Error – вывод информации об ошибках.

Задача InputArray дополнительно требует решения задачи диагностики ошибок при вводе размера и элементов массива (InputChecking)

Задачу File I/O можно разбить на две независимые подзадачи:

SaveArray – сохранение массива в файле,

LoadArray – загрузка массива из файла.

Задача LoadArray дополнительно требует решения задачи диагностики ошибок при чтении размера и элементов массива из файла (FileChecking).

Обе задачи дополнительно требуют решения задачи проверки корректности имени файла (CheckName), задачи проверки существования файла (CheckExist) и доступности файла для чтения-записи (CheckAccess).

Задачу Solve можно разбить на две независимые подзадачи:

Search – поиск непрерывной последовательности элементов массива, которая имеет минимальную или максимальную сумму элементов,

TestArray – создание тестового массива.

Задача Search дополнительно требует решения задачи вычисления суммы последовательности элементов массива (Criteria).

Задача Interface можно разбить на подзадачу разработки консольного меню (MainMenu) и подзадачу обращений к пользователю по ситуации. В рамках данной работы выделяется одна ситуация – необходимость перезаписи файла при сохранении массива (IsRewrite).

Дерево декомпозиции задач следующее:

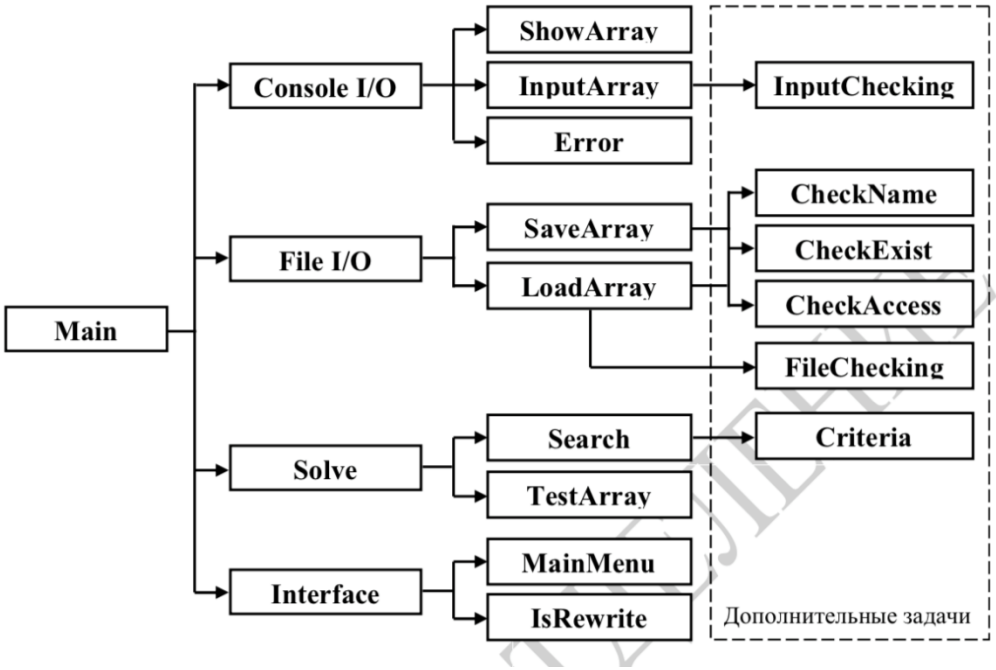

Рисунок 1 - Дерево декомпозиции задач

Таблица 2 - Спецификация требований к задачам (System Requirement Specification)

Требования		Задачи	
1	Консольный ввод-вывод	1	Вывод массива чисел на экран
		2	Ввод чисел с консоли и сохранение их в массиве
		3	Диагностика ошибок при вводе размера массива и чисел
		4	Вывод информации об ошибках пользователя
2	Файловый ввод-вывод	1	Сохранение массива в файле
		2	Загрузка массива из файла
		3	Диагностика ошибок работы файлом при чтении- записи данных
		4	Проверка корректности имени файла
		5	Обработка ситуаций существования файла
3	Обработка массива	1	Вычисление критерия
		2	Поиск последовательности, удовлетворяющей критерию
		3	Загрузка тестового массива
4	Интерфейс пользователя	1	Разработка интерактивного консольного меню для выбора всех команд

Таблица 3 - Спецификация архитектуры модулей (Architectural Module Specification)

Модуль	Тип	Функция	Назначение	SRS
INOUT	CPP	ShowArray	Вывод массива на консоль. Массив передается через аргументы.	1.1
		InputArray	Ввод размера массива, резервирование массива, ввод чисел и сохранение их в массиве с проверкой ошибок при вводе. Новый массив передается через аргументы.	1.2 1.3
		Error	Вывод информации об ошибке на консоль.	1.4
FILE	CPP	SaveArray	Сохранение массива в файле с диагностикой возникающих ошибок. Массив передается через аргументы.	2.1 2.3
		LoadArray	Загрузка массива из файла с диагностикой возникающих ошибок. Корректно загруженный массив передается через аргументы.	2.2 2.3
		CheckFileName	Проверка корректности имени файла.	2.4
SOLVE	CPP	Search	Поиск последовательности чисел, удовлетворяющей критерию.	3.2
		Criteria	Рассчет суммы чисел от одного индекса до другого.	3.1
		TestArray	Копирование тестового массива на место текущего массива	3.3
LAB3	CPP	MainMenu	Организация диалога с пользователем в виде консольного меню.	4.1
		IsRewrite	Запрос подтверждения на перезапись файла.	2.5
		Main	Главная функция	
TASK	H	enum errCode	Коды ошибок	
		Прототипы всех функций		

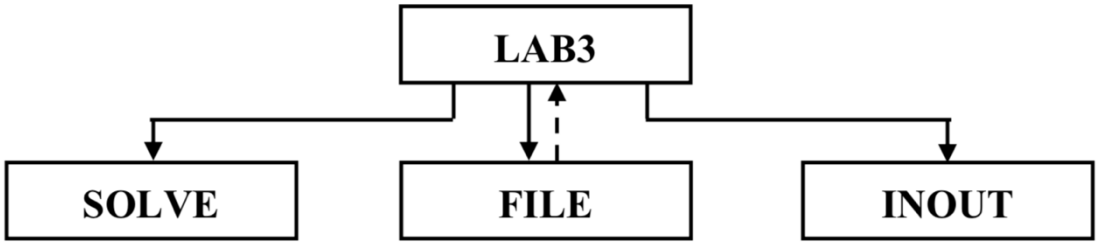

Рисунок 2– Диаграмма взаимосвязи модулей

Спецификация формата файла (File Format Specification)

✓Назначение:

Разработанный формат файла предназначен для хранения массива вещественных чисел, обрабатываемых программой LAB3.

✓Основные требования:

Программа использует текстовые файлы формата ASCII. Файл содержит последовательность чисел, разделенных символами пробела, табуляции или переноса строки. При записи чисел допускаются цифры 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 и знаки +, –, точка. Точка разделяет целую и дробную части числа. В файле не допускаются иные символы, кроме указанных.

✓Внутренняя структура:

Первое число в файле хранит количество чисел в массиве. Количество последующих чисел должно быть не меньше указанного количества. Каждое число отделяется хотя бы одним пробельным символом. Первое число присутствует обязательно и не может иметь точку или знак минус.

N Element1 Element2 ... Element*i* ... ElementN

где N – количество чисел – размер массива

Element*i* – числа – элементы массива

Пример файла:

4

1.25 3.45 –0.56 10

Таблица 4 – Диагностика ошибок формата файла

Код ошибки	Описание ошибки
errArraySize	Если первое число – размер массива – указано не правильно. Например, если использован алфавитный символ, число отрицательно, или выходит за допустимый предел.
errArrayElement	Если число – элемент массива – указано не правильно. Например, если использован алфавитный символ.
errAbsentElements	Если количество чисел в последовательности меньше, чем размер массива, указанный первым числом.
errCanNotSaveValue	Если значение не может быть сохранено в файле из-за системных

	ошибок работы с файлом.
errFileName	Если имя файла записано некорректно.
errFileExists	Если файл существует с указанным именем, а должен отсутствовать.
errFileNotFound	Если файл не существует с указанным именем, а должен присутствовать.
errCanNotOpenFile	Если файл не может быть открыт.
errFileIsReadOnly	Если файл предназначен только для чтения.
errFileIsWriteOnly	Если файл предназначен только для записи.

Блок-схемы и описание алгоритмов:

Алгоритм CalcCriteria рассчитывает сумму последовательности и получает ссылку на массив, в котором находится последовательность, индекс начала и индекс конца последовательности как исходные данные. Алгоритм обходит все элементы последовательности и суммирует их в переменную sum. Алгоритм обнуляет сумму перед вычислениями. Алгоритм возвращает вычисленное значение суммы разности между соседними элементами.

Алгоритм Search осуществляет поиск массиве чисел такой непрерывной последовательности элементов, которая имеет максимальную или минимальную сумму. Алгоритм получает ссылку на массив, размер массива и опцию поиска (True – максимум, False – минимум) как исходные данные. Алгоритм организует перебор последовательностей длины от 2 до (Size – 1). Для этого он использует счётчик I, который проходит элементы от первого до предпоследнего. Для каждого значения счётчика I алгоритм перебирает для него возможные длины последовательностей с помощью счётчика M. Для I-ого положения можно перебрать длины от 2 до (Size – I), так как последовательности больших длин уже будут перебраны на предыдущих шагах. Для каждой последовательности длины M, начинающейся от элемента I, алгоритм вычисляет конечный элемент как (I + M – 1) и использует алгоритм вычисления критерия CalcCriteria, описанный выше, чтобы рассчитать сумму.

Полученное значение суммы алгоритм Search сравнивает со значением критерия. В зависимости от значения опции поиска, чтобы запомнить границы последовательности, полученное значение суммы либо должно быть больше значения критерия при поиске максимума, либо должно быть меньше значения критерия при поиске максимума. Если условие критерия выполняется, то алгоритм запоминает начало последовательности как I, конец последовательности как (I + M – 1), и значение суммы как новое значение критерия.

Перед началом вычислений алгоритм Search инициализирует критериальное значение суммы Criteria либо минимально возможным значением для используемого типа данных (double) при поиске максимума суммы, либо максимально возможным значением для используемого типа данных (double) при поиске минимума суммы в зависимости от опции поиска. Так как алгоритм использует предельное значение типа данных в качестве стартового значения критерия, то при первом же вычислении критерия в циклах алгоритма, он получит значение соответственно либо большее предельно минимального, либо меньшее предельно максимального, и первое сравнение полученного значения со стартовым значением критерия приведёт к корректировке значения критерия. Это позволяет избежать контроля номеров итераций для установки значения критерия.

Алгоритм возвращает начало последовательности и конец последовательности. Поскольку алгоритм возвращает два значения, то при практической реализации рекомендуется использовать передачу аргументов по ссылке.

Алгоритм ShowArray получает ссылку на массив и размер массива как исходные данные. Алгоритм последовательно выводит все элементы массива на консоль.

Алгоритм InputArray получает ссылку на массив, который должен быть перезаписан, и ссылку на переменную, хранящую размер массива как исходные данные. Алгоритм запрашивает размер массива, резервирует память под массив, и последовательно вводит элементы массива с консоли. алгоритм проверяет вводимые данные. Так, размер массива не может быть отрицательным числом и должен быть в диапазоне от 1 до 1000 элементов. Чтобы не испортить текущий массив, находящийся в главной функции, алгоритм InputArray сначала запоминает ссылку и размер нового массива во временных переменных. Если в процессе ввода произойдут ошибки, то алгоритм освободит зарезервированную память, не изменяя массив в главной программе. Если ввод нового массива будет успешным, то алгоритм освободит память текущего массива, используя ссылку на него, и сделает новый массив текущим.

Алгоритм Search

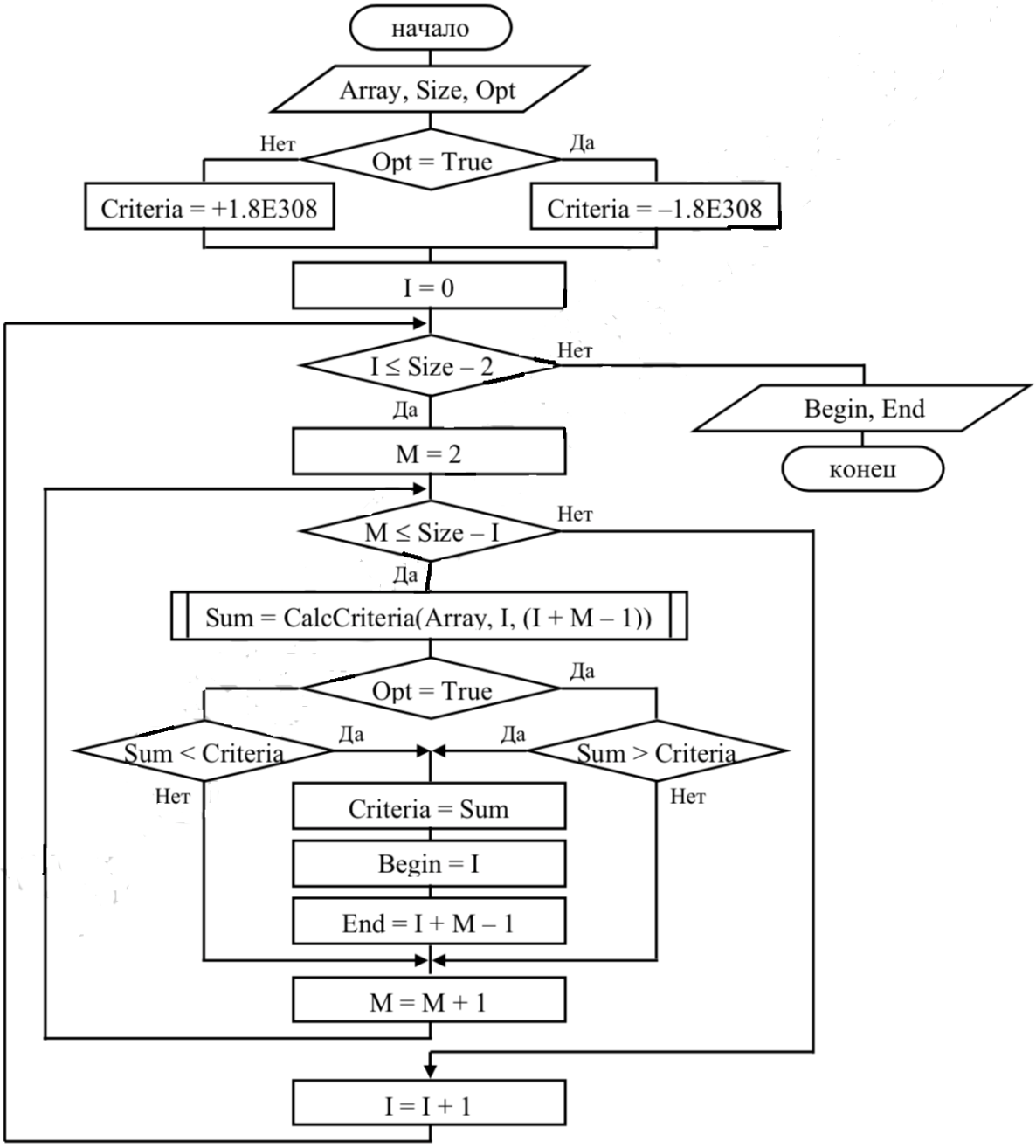

Рисунок 3 – Алгоритм Search

Алгоритм **SaveArray** получает ссылку на массив, размер массива, имя файла для сохранения, и признак перезаписи как исходные данные. Сначала алгоритм осуществляет проверку ситуаций работы с файлами. Если имя файла задано неправильно, то алгоритм возвращает ошибку **errFileName**. Если файл существует, но имеет атрибут **Read-Only**, то алгоритм возвращает ошибку **errFileIsReadOnly**. Если файл существует, то алгоритм ис-

пользует признак перезаписи для выбора действий. Если признак перезаписи **False**, т.е. перезапись запрещена, то алгоритм возвращает ошибку **errFileExists**. Если признак перезаписи **True**, т.е. перезапись разрешена, о алгоритм запрашивает подтверждение на выполнение перезаписи, используя функцию **IsRewrite**, определенную в интерфейсе. Если файл существует, но не может быть открыт для записи, например, файл используется другим приложением, то алгоритм возвращает ошибку **errCanNotOpenFile**. Если ошибок работы с файлом не произошло, то алгоритм последовательно записывает все элементы массива в файл.

Алгоритм **LoadArray** получает ссылку на массив, который должен быть перезаписан, и ссылку на переменную, хранящую размер массива, и имя файла для загрузки как исходные данные. Алгоритм работает также как **InputArray**, но дополнительно осуществляет проверку ситуаций работы с файлами. Если имя файла задано неправильно, то алгоритм возвращает ошибку **errFileName**. Если файл не существует или путь к файлу неправильный, то алгоритм возвращает ошибку **errFileNotFound**. Если файл существует, но не может быть открыт для чтения, то алгоритм возвращает ошибку **errCanNotOpenFile**. При чтении данных алгоритм проверяет корректность размера массива, соответствие размера массива и количества элементов в файле, правильность задания чисел и соответствие спецификации формата.

Алгоритм **TestArray** использует статические массивы для хранения тестовых массивов. Алгоритм освобождает память из под текущего массива, резервирует память под тестовый массив, копирует элементы из статического массива, и делает тестовый массив текущим. Чтобы определить размер статического массива, он использует операцию sizeof.

Алгоритм **CheckFileName** получает имя файла как исходные данные и проверяет его на наличие недопустимых символов <, >, |, *, ?, “, символ двоеточия может быть только на вторым символом в строке, имя не должно содержать двух и более обратных подряд, и т.д.

Тестовые планы:

Тестирование программы предлагается разбить на четыре части:

1.Тестирование поиска последовательности чисел в массиве

В качестве тестового массива использовать следующие данные:

2.1	0.1	-4.5	3.1	8.5	9.3	-9.8	0.2	-1.0	8.2	-5.4

Результат поиска последовательности с максимальной суммой разности:

Left Index = 4

Right Index = 6

Результат поиска последовательности с минимальной суммой разности:

Left Index = 7

Right Index = 9

2.Тестирование ввода массива с консоли

Для тестирования ввода размера массива необходимо использовать следующие данные:

Таблица 5 – Данные для тестирования ввода размера массива

№	Размер	Ожидаемый результат
1	A1	Вывод ошибки Incorrect array size
2	0	Вывод ошибки Incorrect array size
3	-10	Вывод ошибки Incorrect array size
4	пробелы	Повтор ввода
5	5	Правильный ввод. Резервирование трех элементов.

Для тестирования ввода элемента массива необходимо использовать следующие данные:

Таблица 6 - Данные для тестирования ввода элемента массива

№	Элемент	Ожидаемый результат
1	A1	Вывод ошибки Incorrect value
2	Пробелы	Повтор ввода
3	3.2	Ввод числа 3.2
4	-4.89	Ввод числа -4.89
5	1.2.3	Ввод числа 1.2
6	00091	Ввод числа 91
7	1.20000	Ввод числа 1.2

Для тестирования функций ввода, отображения и создания тестового массива выполнить следующие действия:

1) Вызвать функцию ввода массива и ввести пять элементов:

00004	-.1	9.4	3.1.5	1.20000

2) Вызвать функцию отображения массива. Результат на экране:

4-0.19.43.11.2

3) Вызвать функцию ввода массива и ввести три элемента:

14	35	-1.4

4) Вызвать функцию отображения массива. Результат на экране:

1435-1.4

5) Вызвать функцию ввода массива и ввести некорректные данные.

6) Вызвать функцию отображения массива. Результат на экране:

1435-1.4

7) Вызвать функцию создания тестового массива.

6) Вызвать функцию отображения массива. Результат на экране:

2.10.11.2-4.53.18.59.3-9.80.2-1.08.2-5.4

3.Тестирование загрузки массива из файла

Создать тестовые файлов со следующим содержанием:

Файл	Содержание	Ожидаемый результат
1.txt	U3 1.1 -2.3 4.0	Вывод ошибки Incorrect array size
2.txt	0	Вывод ошибки Incorrect array size
3.txt	-10 1.1 4.0	Вывод ошибки Incorrect array size
4.txt	3 1.A 1/0 e4	Вывод ошибки Incorrect element value
5.txt	3 1.1 -2.3	Вывод ошибки There are absent elements
6.txt	3 2.1 -1.4 6.0	Правильный ввод Array is loaded successfully

Для тестирования загрузки вместе с сохранением выполнить следующие действия:

1) Вызвать функцию создания тестового массива.

2) Вызвать функцию сохранения массива в файле. Ввести имя 0.txt. Вывод сообщения: Array is saved successfully

3) Открыть файл 0.txt в программе «Блокнот». Результат в файле:

12 2.1 0.1 1.2 –4.5 3.1 8.5 9.3 –9.8 0.2 –1.0 8.2 –5.4

4) Вызвать функцию загрузки массива из файла. Ввести имя 6.txt.

5) Вызвать функцию отображения массива. Результат на экране:

2.1 –1.4 6.0

6) Вызвать функцию загрузки массива из файла. Ввести имя 0.txt.

7) Вызвать функцию отображения массива. Результат на экране:

2.1 0.1 1.2 –4.5 3.1 8.5 9.3 –9.8 0.2 –1.0 8.2 –5.4

4. Тестирование работы с файлами

Для тестирования функции проверки имени файла использовать следующие данные:

№	Имя файла	Ожидаемый результат
1	1?.txt	Вывод ошибки: Incorrect filename
2	0*1.txt	Вывод ошибки: Incorrect filename
3	0.t:x	Вывод ошибки: Incorrect filename
4	C:\\1.txt\\	Вывод ошибки: Incorrect filename
5	11\2.txt\\	Вывод ошибки: Incorrect filename
6	C:\0.txt	Корректное сохранение или загрузка файла

Для тестирования ситуаций при сохранении данных в файл выполнить следующие действия:

1) Скопировать файл 6.txt в файл 7.txt и в файл 8.txt.

2) В проводнике установить атрибут Read-Only для файла 7.txt.

3) Вызвать функцию создания тестового массива.

4) Вызвать функцию сохранения массива в файле. Ввести имя 7.txt. Вывод ошибки: File is accessed for read-only

5) Вызвать функцию сохранения массива в файле. Ввести имя 8.txt. Вывод сообщения: File already exists. Do you want to overwrite it [Y/N]?

6) Ввести [Y]. Вывод сообщения: Array is saved successfully

7) Открыть файл 8.txt в программе «Блокнот». Результат в файле:

12 2.1 0.1 1.2 –4.5 3.1 8.5 9.3 –9.8 0.2 –1.0 8.2 –5.4

8) Вызвать функцию сохранения массива в файле. Ввести имя 6.txt. Вывод сообщения: File already exists. Do you want to overwrite it [Y/N]?

9) Ввести [N].

10) Открыть файл 6.txt в программе «Блокнот». Результат в файле:

3 2.1 –1.4 6.0

11) Открыть файл 6.txt в программе Word.

12) Вызвать функцию сохранения массива в файле. Ввести имя 6.txt. Вывод сообщения: File can not be opened

Для тестирования ситуации при загрузке данных из файла выполнить следующие действия:

1) Вызвать функцию загрузки массива из файла. Ввести имя 100.txt. Вывод сообщения: File not found

API reference основных функции:

Прототип:

errCode LoadArray(double* &pArray, int &Size, char *Name) Описание:

Функция загружает массив вещественных чисел из файла.

Аргументы:

pArray – ссылка к указателю на массив вещественных чисел,

Size – ссылка к переменной, которая хранит размер массива,

Filename – указатель на строку имени файла с путем.

Возвращаемое значение: Код:

errOK- загрузка успешна, ошибок нет

errFileName – некорректное имя файла

errFileNotFound – файл не найден или неверный путь

errCanNotOpenFile – невозможно открыть

errArraySize – файл хранит некорректный размер массива

errNotMemory – недостаточно памяти для резервирования массива

errArrayElement – файл хранит некорректные элементы массива

errAbsentElements – количество элементов массива меньше его размера

Прототип:

errCode **SaveArray**(double* pArray, int Size, char *Name, bool rw)

Описание:

Функция записывает массив вещественных чисел в файл.

Аргументы:

pArray – указатель на массив вещественных чисел,

Size– размер массива

Name– указатель на строку имени файла с путем,

Rw - режим перезаписи в случае существования файла:

true - запросить подтверждение на перезапись

false - отменить сохранение с выдачей ошибки.

Возвращаемое значение: Код:

errOK– сохранение было успешно, ошибок нет

errFileName– некорректное имя файла

errFileExists– файл существует и перезапись отменена,

errFileIsReadOnly – файл существует и доступен только для чтения,

errCanNotOpenFile – невозможно открыть или создать файл.

Прототип:

void **ShowArray**(double * pArray, int Size)

Описание:Функция выводит массив вещественных чисел на консоль. **Аргументы:**

pArray – указатель на массив вещественных чисел,

Size – размер массива.

Возвращаемое значение: нет.

Прототип:errCode **InputArray**(double* &pArray, int &Size)

Описание: Функция вводит массив вещественных чисел с консоли. **Аргументы:**

pArray – ссылка к указателю на массив вещественных чисел,

Size – ссылка к переменной, которая хранит размер массива.

Возвращаемое значение: Код:

errOK – ввод был успешен, ошибок нет,

errArraySize – был введен некорректный размер массива

errNotMemory – недостаточно памяти для резервирования массива

errArrayElement – элементы были введены некорректно

Прототип:

Double **CalcCriteria**(double*pArray,intLeft,int Right)

Описание: Функция вычисляет сумму последовательности элементов массива.

Аргументы:

pArray – указатель на массив вещественных чисел

Left– индекс начала последовательности,

Right – индекс конца последовательности.

Возвращаемое значение: Сумма элементов последовательности.

Прототип:

Void **Search**(double*Arr,intSize,bool Opt,int&Begin,int&End)

Описание: Функция ищет в массиве вещественных чисел такую непрерывную последовательность элементов, которая имеет либо максимальную, либо минимальную сумму элементов.

Аргументы:

Arr– указатель на массив вещественных чисел,

Size– размер массива,

Opt–опция поиска:

true–максимум суммы,

false–минимум суммы.

Begin – ссылка на переменную, в которую помещается индекс начала последовательности,

End– ссылка на переменную, в которую помещается индекс конца последовательности.

Возвращаемое значение:

Функция возвращает индексы начала и конца последовательности через свои аргументы.

Прототип:

bool **CheckFileName**(char*name)

Описание: Функция проверяет корректность имени файла.

Аргументы:

Filename– указатель на строку имени файла с путем.

Возвращаемое значение:

true –если имя файла корректно,

false–если имя файла некорректно.

Прототип:

errCode **TestArray**(double*&pArray,int&Size)

Описание: Функция копирует тестовый массив.

Аргументы:

pArray –ссылкакуказателюнамассиввещественныхчисел,

Size–ссылка к переменной, которая хранит размер массива.

Возвращаемое значение: всегда errOK.

Листинг программы:

```
/*****************************************************************
 * Содержание модуля TASK.H
*****************************************************************/

/* Список кодов ошибок */
typedef enum {
   errOK = 0,
   errNotMemory = 1,
   errArraySize = 2,
   errArrayElement = 3,
   errAbsentElements = 4,
   errCanNotOpenFile = 5,
   errCanNotSaveValue = 6,
   errFileName = 7,
   errFileExists = 8,
   errFileNotFound = 9,
   errFileIsReadOnly = 10,
   errFileIsWriteOnly = 11
} errCode;
bool IsRewrite();
bool CheckFileName(char *name);
errCode Error(char *operate, errCode err);
errCode InputArray(double* &pArray, int &Size);
errCode TestArray(double* &pArray, int &Size);
errCode SaveArray(double* pArray, int Size, char *filename, bool rw);
errCode LoadArray(double* &pArray, int &Size, char *filename);
void ShowArray(double *arr, int size);
double CalcCriteria(double *arr, int left, int right);
void Search(double *arr, int size, bool opt, int& begin, int& end);

/*****************************************************************
 * Содержание модуля SOLVE.CPP
 *****************************************************************/
#include <float.h>
#include <stddef.h>
#include <cmath>
#include "task.h"

/* Функция загружает тестовый массив в качестве текущего массива */
errCode TestArray(double* &pArray, int &Size)
{
   static double test[] = {
      2.1,
      0.1,
      1.2,
      -4.5,
      3.1,
      8.5,
      9.3,
      -9.8,
```

```
        0.2,
        -1.0,
        8.2,
        -5.4
    };

    if(pArray != NULL)
        delete[] pArray;

    Size = sizeof(test) / sizeof(double);

    pArray = new double[Size];
    for(int i=0; i < Size; i++)
        pArray[i] = test[i];

    return errOK;
}

/*
 Функция вычисляет сумму разностей последовательности элементов в заданном массиве
arr от индекса left до индекса right
 */
double CalcCriteria(double *arr, int left, int right) {

    int SUM_DIFF_SIZE = right - left;
    double sum_diff[SUM_DIFF_SIZE];

    int s_i = 0;
    for(int j=left; j <= right; j++) {
        sum_diff[s_i] += abs(arr[j] - arr[j + 1]);
        s_i++;
    }

    double sum = 0;
    for(int j=0; j < SUM_DIFF_SIZE; j++)
        sum += sum_diff[j];

    return sum;
}

/* Функция ищет в массиве arr размерностью size такую непрерывную последовательность
    элементов, начинающуюся с индекса begin и заканчивающуюся индексом end, которая
    имеет максимальную при opt = true или минимальную при opt = false сумму элементов.
 */
void Search(double *arr, int size, bool opt, int& begin, int& end) {
    int i,m;
    double sum;
    double criteria = ((opt) ? -DBL_MAX : DBL_MAX);

    for(i=0 ; i <= size-2 ; i++) {
        for(m=2; m <= size-i; m++) {
            sum = CalcCriteria(arr, i, (i + m - 1));
```

```
            if((opt) ? sum > criteria : sum < criteria) {
                criteria = sum;
                begin = i;
                end = i + m - 1;
            }
        }
    }
}

/*************************************************************************
 * Содержание модуля INOUT.CPP *
 *************************************************************************/
#include <iostream>
#include "task.h"
using namespace std;

/* Функция выводит на консоль массив pArray размером Size */
void ShowArray(double *pArray, int Size)
{
    if(pArray != NULL || Size < 1) {
        cout << "Current array:" << endl;

        for(int i=0 ; i < Size ; i++)
            cout << pArray[i] << " ";
    }
    else cout << "Array is not entered";
}

/* Функция вводит массив с консоли и диагностирует ошибки
   pArray – ссылка на текущий массив, Size – ссылка на его размер
 */
errCode InputArray(double* &pArray, int &Size) {
    char ch = '\0';
    double* newArray = NULL;
    int idx, newSize = 0;
    do {
        cout << "Entry the array size: ";
        cin >> newSize;
        cin.sync();
        if(cin.fail()) {
            cin.clear();
            newSize = 0;
        }

        if(newSize < 1 || newSize > 1000) {
            cout << "Invalid array size. [A]-Abort. [Enter]-Repeat.";
            cin.get(ch);

            if((ch == 'A') || (ch == 'a'))
                return errArraySize;
        } else ch = '\0';
    } while(ch != '\0');
```

```
    newArray = new double[newSize];
    if(newArray == NULL)
        return errNotMemory;

    cout << "Entry " << newSize << " elements of the array:" << endl;
    for(idx=0; idx < newSize; idx++) {
        do {
            cout << "arr[" << idx << "] ?>";
            cin >> newArray[idx];
            cin.sync();
            if(cin.fail()) {
                cin.clear();
                cout << "Invalid value. [A]-Abort. [Enter]-Repeat.";
                cin.get(ch);

                if((ch == 'A') || (ch == 'a')) {
                    delete[] newArray;
                    return errArrayElement;
                }
            } else ch = '\0';
        } while(ch != '\0');
    }
    if(pArray != NULL)
        delete[] pArray;

    pArray = newArray;
    Size = newSize;
    return errOK;
}

/* Список описаний ошибок */
char *errDescription[] = {
    "",
    "Not enough of memory",
    "Incorrect array size",
    "Incorrect element value",
    "There are absent elements",
    "File can not be opened",
    "Value can not be saved",
    "Incorrect filename",
    "File already exists",
    "File not found",
    "File is accessed for read-only",
    "File is accessed for write-only"
};

/* Функция выводит на консоль описание ошибки */
errCode Error(char *Operate, errCode err)
{
    if(err)
        cout << endl << Operate << " status: " << errDescription[err] << endl;
```

```
    return err;
}

/*********************************************************************
 * Содержание модуля FILE.CPP *
 *********************************************************************/
#include <fstream>
#include <unistd.h>
#include "task.h"
using namespace std;

/*
 Функция проверяет корректность имени файла filename,
 и Возвращает true – если имя корректно, или false – если нет.
 */
bool CheckFileName(char *name) {
    char ch;
    int i;
    for(i=0,ch=name[0]; ch != '\0'; i++, ch=name[i]) {
        if(ch == ':' && i != 1)
            return false;

        if(ch == '\\' && (name[i+1] == '\0' || name[i+1] == '\\'))
            return false;

        if(ch == '<' || ch == '>' || ch == '*' || ch == '?' || ch == '|' || ch == '\"')
            return false;
    }
    return true;
}
/* Функция загружает массив из файла с именем filename и
 диагностирует ошибки. pArray – ссылка на текущий массив,
 который должен быть перезаписан, Size – ссылка на его размер
 */
errCode LoadArray(double* &pArray, int &Size, char *filename) {
    double* tmpArray = NULL;
    int idx,
    tmpSize = 0;

    if(CheckFileName(filename) == false)
        return errFileName;

    if(access(filename, R_OK) != 0)
        return errFileNotFound;

    ifstream in(filename);

    if(!in)
        return errCanNotOpenFile;
    in >> tmpSize;
```

```
    if(!in)
        tmpSize = 0;

    if(tmpSize < 1 || tmpSize > 1000)
        return errArraySize;

    tmpArray = new double[tmpSize];

    if(tmpArray == NULL)
        return errNotMemory;

    for(idx=0 ; idx < tmpSize ; idx++) {
        in >> tmpArray[idx];
        if(!in) {
            delete[] tmpArray;
            return ((in.eof()) ? errAbsentElements : errArrayElement);
        }
    }

    if(pArray != NULL)
        delete[] pArray;

    pArray = tmpArray;
    Size = tmpSize;
    return errOK;
}
/* Функция сохраняет массив pArray размеров Size в файле с именем
 filename в режиме перезаписи rw = true или отмены записи в случае
 его существования rw = false.
 */
errCode SaveArray(double *pArray, int Size, char *filename, bool rw) {
    if(CheckFileName(filename) == false)
        return errFileName;

    if(access(filename, R_OK) == 0) {

        if(access(filename, W_OK) != 0)
            return errFileIsReadOnly;

        if(rw == false)
            return errFileExists;
        else
            if((rw = IsRewrite()) == false)
                return errFileExists;
    }

    ofstream out(filename);

    if(!out)
        return errCanNotOpenFile;

    out << Size << endl;
```

```
    if(!out)
        return errCanNotSaveValue;

    for(int idx=0 ; idx < Size ; idx++) {
        out << pArray[idx] << endl;
        if(!out)
            return errCanNotSaveValue;
    }

    return errOK;
}

/*************************************************************************
 * Содержание модуля LAB3.CPP *
 *************************************************************************/
#include <iostream>
#include "task.h"
using namespace std;

/* Функция делает запрос на перезапись файла. */
bool IsRewrite()
{
    char ch;
    cout << endl << "File already exists. Do you want to overwrite it [Y/N]?";
    cin >> ch;
    return (ch == 'Y' || ch == 'y');
}

/* Функция реализует консольное меню и интерфейс с пользователем. */
void MainMenu(double* &wrkArray, int& wrkSize)
{
    char ch = ' ';
    char filename[66];
    bool bCriteria;
    int idxLeft, idxRight;
    do {
        cout << endl << "[I] Input a new array";
        cout << endl << "[T] Create the test array";
        cout << endl << "[S] Save the array in a file";
        cout << endl << "[L] Load the array from a file";
        cout << endl << "[D] Display the current array";
        cout << endl << "[R] Search the sequence for criteria";
        cout << endl << "[Q] Quit the program" << endl << "?>";
        cin >> ch;

        switch(ch) {
            case 'I':
            case 'i':
                InputArray(wrkArray, wrkSize);
                break;
            case 'T':
```

```
    case 't':
        TestArray(wrkArray, wrkSize);
    case 'D':
    case 'd':
        ShowArray(wrkArray, wrkSize);
        break;
    case 'S':
    case 's':
        cout << "Entry the filename:>";
        cin >> filename;
        if(Error("Saving", SaveArray(wrkArray, wrkSize, filename, true)) == errOK)
            cout << endl << "Array is saved successfully" << endl;
        break;
    case 'L':
    case 'l':
        cout << "Entry the filename to be loaded:>";
        cin >> filename;
        if(Error("Loading", LoadArray(wrkArray, wrkSize, filename)) == errOK)
            cout << "Array is loaded successfully" << endl;
        break;
    case 'R':
    case 'r':
        if(wrkArray == NULL) {
            cout << "Array is not entered";
            break;
        }
        cout << "Select criteria:" << endl << "[I]-max(sum) , [D]-min(sum) ?>";
        cin >> ch;

        if(ch == 'I' || ch == 'i') {
            bCriteria = true;

        } else {
            if(ch == 'D' || ch == 'd') {
                bCriteria = false;
            } else {
                cout << "Incorrect option: " << ch;
                ch = ' ';
                break;
            }
        }

        Search(wrkArray, wrkSize, bCriteria, idxLeft, idxRight);
        ShowArray(wrkArray, wrkSize);

        cout << endl << "Left index = " << idxLeft << endl << "Right index = " << idxRight;
        break;
    default:
        cout << "Incorrect menu item";
}

cin.sync();
```

```
    } while(ch != 'Q' && ch != 'q');
}

int main(void) // Главная функция
{
    double* myArray = NULL; // Текущий рабочий массив
    int arrSize = 0; // Размер текущего массива
    MainMenu(myArray, arrSize);
    return 0;
}
```

Количество вариантов заданий по всем контрольным работам соответствует количеству студентов в группе.

4.4.2 Тестирование

В качестве примера оценочных средств для текущего контроля успеваемости студентов по 2 разделу «Алгоритмы и структуры данных» представлены следующие варианты заданий:

1) Открытые тесты (необходимо вписать слово в предложение):

- Структура _____ - это способ хранения и организации данных, облегчающий доступ к этим данным и их модификацию.
- _____ - элементарная структура данных реализующая стратегию «последним вошел – первым вышел».
- Связанный _____ - это структура данных, в которой объекты расположены в линейном порядке, где порядок определяется указателями на каждый объект.

2) Закрытые тесты (необходимо выбрать один или несколько вариантов ответа):

- Необходимо выбрать правильный ответ. Структура данных реализующая стратегию «первый вошел – первый вышел» - это

· стек;

· очередь.

- Необходимо выбрать правильный ответ. Способ обхода бинарного дерева поиска, при котором сначала выводится корень, а затем значения правого и левого поддеревьев:

· центрированный обход дерева;

· обход в прямом порядке;

· обход в обратном порядке.

- Необходимо выбрать правильный ответ. Красно черные деревья:

· сбалансированные;

· приближенно сбалансированные (путь от корня не отличается от другого пути более чем в два раза);

· несбалансированный.

В качестве примера оценочных средств для текущего контроля успеваемости студентов по 5 разделу «Методология программирования» представлены следующие варианты заданий:

1) Открытые тесты (необходимо вписать слово в предложение):

- _____ программирование — парадигма программирования, в которой процесс вычисления трактуется как вычисление значений функций в математическом понимании последних.
- _____ программирование — это парадигма программирования, которая описывает процесс вычисления в виде инструкций, изменяющих состояние данных.
- _____ — свойство системы, позволяющее объединить данные и методы, работающие с ними, в классе.

2) Закрытые тесты (необходимо выбрать один или несколько вариантов ответа):

- Необходимо выбрать правильный ответ. Какой из языков программирования относится к логическому программированию:

· C++;

· C#;

· Prolog;

· Java.

- Необходимо выбрать правильный ответ. Какой из языков программирования относится к логическому программированию:

· C++;

· C#;

· Prolog;

· Java;

· JavaScript.

В качестве примера оценочных средств для текущего контроля успеваемости студентов по 8 разделу «Многопроцессное и многопоточное программирование» представлены следующие варианты заданий:

1) Открытые тесты (необходимо вписать слово в предложение):

- _____ — свойство платформы или приложения, состоящее в том, что процесс, порождённый в операционной системе, может состоять из нескольких потоков, выполняющихся «параллельно», то есть без предписанного порядка во времени.

- _____ — это API-интерфейс, который является отраслевым стандартом для создания параллельных приложений для компьютеров с совместным использованием памяти..

- _____ позволяет задавать критические секции кода, в которые в каждый отдельный момент может входить только один поток, гарантируя, что любые временные недействительные состояния вашего объекта будут невидимы его клиентам..

2) Закрытые тесты (необходимо выбрать один или несколько вариантов ответа):

- Необходимо выбрать один или несколько правильных ответов. Когда использовать потоки:

· приложениям требуется выполнять несколько задач одновременно;

· упрощения структуры сложных систем (использование очередей и асинхронной обработки);

· наличие задач подходящих для фоновой обработки.

5. Перечень учебно-методического обеспечения для самостоятельной работы обучающихся по дисциплине.

Методические указания для обучающихся по организации самостоятельной работы по дисциплине, включая перечень тем самостоятельной работы, формы текущего контроля по дисциплине и требования к их выполнению размещены в электронной информационно-образовательной среде СПбГТИ(ТУ) на сайте: http://media.technolog.edu.ru

6. Фонд оценочных средств для проведения промежуточной аттестации.

Своевременное выполнение обучающимся мероприятий текущего контроля позволяет превысить (достигнуть) пороговый уровень («удовлетворительно») освоения предусмотренных элементов компетенций.

Результаты дисциплины считаются достигнутыми, если для всех элементов компетенций превышен (достигнут) пороговый уровень освоения компетенции на данном этапе.

Промежуточная аттестация по дисциплине проводится в форме зачета и экзамена.

К сдаче зачета допускаются студенты, выполнившие все формы текущего контроля.

Зачет предусматривают выборочную проверку освоения предусмотренных элементов компетенций и комплектуются вопросами (заданиями) двух видов: один теоретический вопрос (для проверки знаний) и комплексная задача (для проверки умений и навыков).

При сдаче зачета, студент получает два вопроса из перечня вопросов, время подготовки студента к устному ответу - до 30 мин.

Пример варианта вопросов на зачете:

Билет №1
1.Постановка задачи на проектирование программного обеспечения. 2.Алгоритм сортировки выбором. Пример программной реализации.

Заключительная аттестация по дисциплине проводится в форме экзамена.

Экзамен предусматривают выборочную проверку освоения предусмотренных элементов компетенций и комплектуются вопросами (заданиями) двух видов: теоретический вопрос (для проверки знаний) и комплексная задача (для проверки умений и навыков).

При сдаче экзамена, студент получает три вопроса из перечня вопросов, время подготовки студента к устному ответу - до 45 мин.

Вариант № 1
1.Многопоточность на C++. 2.Концепции чёрного, серого, белого ящика при тестировании 3.Алгоритм сортировки выбором. Пример программной реализации

Фонд оценочных средств по дисциплине представлен в Приложении № 1

Результаты освоения дисциплины считаются достигнутыми, если для всех элементов компетенций достигнут пороговый уровень освоения компетенции на данном этапе – оценка «удовлетворительно».

Фонд оценочных средств по дисциплине представлен в Приложении № 1

7. Перечень учебных изданий, необходимых для освоения дисциплины.

а) печатные издания:

1. Норенков, И. П. Автоматизированные информационные системы: учеб. пособие для вузов / И. П. Норенков. – М.: Изд-во МГТУ им. Н.Э. Баумана, 2011. – 342 с.
2. Чистякова, Т. Б. Применение универсальных моделирующих программ для синтеза и анализа технологических процессов: учеб. пособие / Т. Б. Чистякова, Л. В. Гольцева, А. В. Козлов. – СПб. : СПбГТИ(ТУ), 2011. – 66 с.
3. Орлов, С. А. Теория и практика языков программирования : учебник по направлению "Информатика и вычислительная техника" / С. А. Орлов. – М. ; СПб. ; Н. Новгород : Питер, 2014. - 688 с.
4. Бройдо, В. Л. Вычислительные системы, сети и телекоммуникации : учеб. пособие для вузов / В. Л. Бройдо, О. П. Ильина. – 4-е изд. – М. ; СПб. ; Н. Новгород : Питер, 2011. – 554 с.
5. Ликнесс, Д. Приложения для Windows 8 на C# и XAML / Дж. Ликнесс. - М. ; СПб. ; Н. Новгород : Питер, 2013. - 368 с.
6. Советов, Б. Я. Представление знаний в информационных системах : учеб. для вузов / Б. Я. Советов, В. В. Цехановский, В. Д. Чертовский. – М. : Академия, 2011. – 143 с.
7. Макконнелл, С. Совершенный код. Мастер-класс / С. Макконнелл ; пер. с англ. под ред. В. Г. Вшивцева. – М. : Рус. ред., 2012. - 867 с.
8. Петкович, Д. Microsoft SQL Server 2008. Руководство для начинающих : пер. с англ. / Д. Петкович. – СПб. : БХВ-Петербург, 2012. - 730 с.
9. Дейтел, П. Как программировать на Visual C# 2012 : Включая работу в Windows 7 и Windows 8 / П. Дейтел, Х. Дейтел. - 5-е изд. - М. ; СПб. ; Н. Новгород : Питер, 2014. - 864 с.
10. Лафоре, Р. Объектно-ориентированное программирование в C++ / Р. Лафоре. - 4-е изд. - М. ; СПб. ; Н. Новгород : Питер, 2015. - 928 с.
11. Лав, Р. Linux. Системное программирование / Р. Лав. - 2-е изд. - М. ; СПб. ; Н. Новгород : Питер, 2014. - 448 с.
12. Павловская, Т. А. C#. Программирование на языке высокого уровня : учеб. пособие для вузов / Т. А. Павловская. - М. ; СПб. ; Н. Новгород : Питер, 2014. - 432 с
13. Чистякова, Т. Б. Программирование на языках высокого уровня. Базовый курс : учеб. пособие / Т. Б. Чистякова, Р. В. Антипин, И. В. Новожилова ; СПбГТИ(ТУ). Каф. систем автоматизир. проектирования и упр. – СПб. : [б. и.], 2008. – 101 с.
14. Тенишев, Д. Ш. Лингвистическое и программное обеспечение автоматизированных систем: учеб. пособие для вузов / Д. Ш. Тенишев ; под ред. Т. Б. Чистяковой. – СПб. : Центр образовательных программ «Профессия», 2010. – 403 с.
15. Головин, Ю. А. Информационные сети: учеб. для вузов / Ю. А. Головин, А. А. Суконщиков, С. А. Яковлев. – М.: Академия, 2011. – 376 с.
16. Елович, И. В. Информатика : Учебник для вузов по техническим и естественнонаучным направлениям / И. В. Елович, И. В. Кулибаба; под ред. Г. Г. Раннева. - М. : Академия, 2011. - 394 с.
17. Уильямс, Б. WordPress для профессионалов. Разработка и дизайн сайтов / Б. Уильямс, Д. Дэмстра, Х. Стэрн. - М. ; СПб. ; Н. Новгород : Питер, 2014. - 464 с. : ил. - (Для профессионалов).

б) электронные учебные издания:

18. Царев, Р. Ю. Проектирование, разработка и оценка надежности сложных программных систем : монография / Р. Ю. Царев. – Красноярск : КрасГАУ, 2017. – 232 с. (ЭБС «Лань»)
19. Барков, И. А. Объектно-ориентированное программирование : учебник / И. А. Барков. – Санкт-Петербург : Лань, 2019. – 700 с. (ЭБС «Лань»)

8. Перечень электронных образовательных ресурсов, необходимых для освоения дисциплины.

учебный план, РПД и учебно-методические материалы: http://media.technolog.edu.ru
веб-страница журнала «Информационные технологии» http://www.novtex.ru/IT
сайты информационных технологий: http://inftech.webservis.ru, http://citforum.ru
информационно-аналитический портал «Научная электронная библиотека» http://elibrary.ru
международные мультидисциплинарные аналитические реферативные базы данных научных публикаций http://webofknowledge.com, http://scopus.com
электронно-библиотечные системы:
«Электронный читальный зал – БиблиоТех» https://technolog.bibliotech.ru/;
«Лань» https://e.lanbook.com/books/.

9. Методические указания для обучающихся по освоению дисциплины.

Все виды занятий по дисциплине «Информационные системы управления качеством в автоматизированных и автоматических производствах» проводятся в соответствии с требованиями следующих СТП:

СТП СПбГТИ 040-02. КС УКДВ. Виды учебных занятий. Лекция. Общие требования;

СТО СПбГТИ 018-2014. КС УКДВ. Виды учебных занятий. Семинары и практические занятия. Общие требования к организации и проведению.

СТП СПбГТИ 048-2009. КС УКВД. Виды учебных занятий. Самостоятельная планируемая работа студентов. Общие требования к организации и проведению.

Планирование времени, необходимого на изучение данной дисциплины, лучше всего осуществлять на весь семестр, предусматривая при этом регулярное повторение пройденного материала.

Основными условиями правильной организации учебного процесса для студентов является:

плановость в организации учебной работы;
серьезное отношение к изучению материала;
постоянный самоконтроль.

На занятия студент должен приходить, имея багаж знаний и вопросов по уже изученному материалу.

10. Перечень информационных технологий, используемых при осуществлении образовательного процесса по дисциплине.

10.1. Информационные технологии.

В учебном процессе по данной дисциплине предусмотрено использование информационных технологий:

чтение лекций с использованием слайд-презентаций;
взаимодействие с обучающимися посредством ЭИОС.

10.2. Программное обеспечение.

Операционная система Microsoft Windows 10

Интегрированная среда разработки программного обеспечения: Microsoft Visual Studio 2019, Android Studio 4.2

Система управления базами данных Microsoft Access 2018

Редактор диаграмм и блок-схем Microsoft Visio 2019

Офисный пакет программ LibreOffice

10.3. Базы данных и информационные справочные системы.

Справочно-поисковая система «Консультант-Плюс»

11. Материально-техническое обеспечение освоения дисциплины в ходе реализации образовательной программы.

На кафедре систем автоматизированного проектирования и управления СПбГТИ(ТУ) имеется необходимая материально-техническая база, соответствующая действующим санитарным и противопожарным правилам и нормам:

Наименование компьютерного класса кафедры	Оборудование
Класс информационных и интеллектуальных систем	Персональные компьютеры (20 шт.): четырехядерный процессор Intel Core i7-920 (2666 МГц), ОЗУ 6 Гб; НЖМД 250 Гб; CD/DVD привод, DVD-RW; видеокарта NVIDIA GeForce GT 220 (1024 Мб); звуковая и сетевая карты, встроенные в материнскую плату.
Класс интегрированных систем проектирования и управления химико-технологическими процессами	Персональные компьютеры (15 шт.): двухядерный процессор Intel Core 2 Duo (2,33 ГГц); ОЗУ 4096 Мб; НЖМД 250 Гб; CD/DVD привод, DVD-RW; видеокарта NVIDIA GeForce 8500 GT; звуковая и сетевая карты, встроенные в материнскую плату.
Класс базовых информационных процессов и технологий	Персональные компьютеры (9 шт.): моноблок Lenovo C360 с 19,5-дюймовым дисплеем; процессор Intel Core i3-4130T (2,9ГГц); ОЗУ 4 Гб; НЖМД 1000 Гб; встроенные DVD-RW, видеокарта Intel HD Graphics 4400, звуковая и сетевая карты.
Лекционная аудитория	Учебная мебель. Мультимедийный проектор NEC NP41. Ноутбук Asus a6j на базе процессора Intel Core Duo T2000. Мультимедийная интерактивная доска ScreenMedia.

12. Особенности освоения дисциплины инвалидами и лицами с ограниченными возможностями здоровья.

Для инвалидов и лиц с ограниченными возможностями учебные процесс осуществляется в соответствии с Положением об организации учебного процесса для обучения инвалидов и лиц с ограниченными возможностями здоровья СПбГТИ(ТУ), утвержденным ректором 28.08.2014.

Приложение № 1
к рабочей программе дисциплины

Фонд оценочных средств
для проведения промежуточной аттестации по
дисциплине «РАЗРАБОТКА ПРОГРАММНЫХ СИСТЕМ»

1. Перечень компетенций и этапов их формирования.

Индекс компетенции	Содержание	Этап формирования
ОПК-4	Способен участвовать в разработке стандартов, норм и правил, а также технической документации, связанной с профессиональной деятельностью	начальный
ПК-1	Способен выполнять работы и управлять работами по созданию (модификации) и сопровождению ИС, автоматизирующих задачи организационного управления и бизнес-процессы	начальный
ПК-2	Способен осуществлять концептуальное, функциональное и логическое проектирование систем среднего и крупного масштаба и сложности	начальный

2. Показатели и критерии оценивания компетенций на различных этапах их формирования, шкала оценивания

Код и наименование индикатора достижения компетенции	Показатели сформированности (дескрипторы)	Критерий оценивания	Уровни сформированности (описание выраженности дескрипторов)		
			«удовлетворительно» (пороговый)	«хорошо» (средний)	«отлично» (высокий)
ОПК-4.3 Определение основных стандартов оформления технической документации на различных стадиях жизненного цикла программной системы	Приводит примеры основных стандартов оформления технической документации на различных стадиях жизненного цикла программной системы (ЗН-1)	Правильные ответы на вопросы №1-9	Допускает ошибки при перечислении основных технологий создания и внедрения информационных систем, стандартов управления жизненным циклом информационной системы.	Перечисляет основные технологии создания и внедрения информационных систем без ошибок, но путается в последовательности проектирования.	Идеально ориентируется в основных технологиях создания и внедрения информационных систем и последовательности их проектирования в соответствии со стандартами управления жизненным циклом.
	Сопоставляет основные стандарты оформления технической документации на различных стадиях жизненного цикла программной системы (У-1)	Правильные ответы на вопросы №1-9	Путается при сопоставлении основных технологий создания и внедрения информационных систем. Делает ошибки в использовании стандартов управления жизненным циклом информационной системы	Составляет и делает выводы по выбору технологий создания и внедрения информационных систем. Допускает неточности в использовании стандартов управления жизненным циклом информационной системы	Сопоставляет основные технологии создания и внедрения информационных систем и стандарты управления жизненным циклом информационной системы
	Решает задачу выбора эффективного стандарта оформления технической документации на различных стадиях жизненного цикла конкретной программной системы (Н-1)	Правильные ответы на вопросы №1-9	Допускает ошибки при решении задачи выбора эффективной технологии для создания и внедрения информационной системы и стандартов управления жизненным циклом.	Допускает неточности при решении задачи выбора эффективной технологии для создания и внедрения информационной системы и стандартов управления жизненным циклом.	Идеально решает задачу выбора эффективной технологии для создания и внедрения информационной системы в соответствии со стандартами управления жизненным циклом.
ПК-1.15 Выявление и анализ требований к программной системе	Приводит примеры требований к программной системе (ЗН-2)	Правильные ответы на вопросы № 10-24	С ошибками приводит примеры требований к информационной системе	Правильно приводит примеры требований к информационной системе с небольшими подсказками преподавателя	Способен самостоятельно сформировать примеры требований к информационной системе.
	Анализирует и объясняет требования к программной системе в соответствии с техническим заданием.(У-2)	Правильные ответы на вопросы № 10-24	Слабо ориентируется в требованиях к информационной системе в соответствии с техническим заданием	Допускает неточности при анализе и объяснении требований к информационной системе в соответствии с техническим заданием.	Идеально анализирует и объясняет требования к информационной системе в соответствии с техническим заданием.
	Решает задачи создания программной системы в соответствии с выявлением и анализом требований к программной системе с использованием компьютерных средств проектирования.(Н-2)	Правильные ответы на вопросы № 10-24	Слабо ориентируется в алгоритмах решения задачи создания программной системы в соответствии с выявлением и анализом требований к информационной системе с использованием компьютерных средств проектирования.	Решает задачи создания программной системы в соответствии с выявлением и анализом требований к информационной системе с использованием компьютерных средств проектирования с небольшими ошибками	Качественно и без ошибок решает задачи создания программной системы в соответствии с выявлением и анализом требований к информационной системе с использованием компьютерных средств проектирования.

Код и наименование индикатора достижения компетенции	Показатели сформированности (дескрипторы)	Критерий оценивания	Уровни сформированности (описание выраженности дескрипторов)		
			«удовлетворительно» (пороговый)	«хорошо» (средний)	«отлично» (высокий)
ПК-1.16 Разработка архитектуры программной системы	Перечисляет принципы разработки архитектуры программной системы (ЗН-3)	Правильные ответы на вопросы № 25-44 к экзамену	Путается в перечислении принципов разработки архитектуры программной системы	Перечисляет принципы разработки архитектуры программной системы с небольшими ошибками	Уверенно и без ошибок перечисляет принципы разработки архитектуры программной системы
	Определяет закономерности при разработке архитектуры программной системы (У-3)	Правильные ответы на вопросы № 25-44 к экзамену	Путается в определение закономерности при разработке архитектуры программной системы	Правильно объясняет и определяет закономерности при разработке архитектуры программной системы с помощью наводящих вопросов	Уверенно и без ошибок определяет закономерности при разработке архитектуры программной системы
	Демонстрирует навыки разработки архитектуры программной системы (Н-3)	Правильные ответы на вопросы № 25-44 к экзамену	Путается в разработке архитектуры новой программной системы	Демонстрирует с небольшими ошибками навыки разработки архитектуры программной системы	Уверенно ориентируется в решении задач разработки архитектуры программной системы.
ПК-2.1 Разработка концепции программной системы	Перечисляет принципы разработки концепции программной системы (ЗН-4)	Правильные ответы на вопросы № 45-66 к экзамену	Выбирает с ошибками принципы разработки концепции программной системы	Выбирает принципы разработки концепции программной системы, но с наводящими вопросами	Правильно сравнивает, анализирует и выбирает принципы разработки концепции программной системы
	Определяет закономерности при разработке концепции программной системы (У-4)	Правильные ответы на вопросы № 45-66 к экзамену	Путается в определение закономерности при разработке концепции программной системы	В целом правильно объясняет и определяет закономерности при разработке концепции программной системы , но допускает незначительные неточности.	Идеально анализирует, сравнивает и определяет закономерности при разработке концепции программной системы
	Демонстрирует навыки разработки концепции программной системы (Н-4)	Правильные ответы на вопросы № 45-66 к экзамену	Путается в разработке концепции новой программной системы	Демонстрирует с небольшими ошибками навыки разработки концепции новой информационной системы	Уверенно ориентируется в решении задач разработки концепции новой программной системы.
ПК-2.2 Разработка технического задания на программную систему	Знает состав, содержание, требования и стандарты необходимые при разработке технического задания на программную систему (ЗН-5)	Правильные ответы на вопросы № 67-73 к экзамену	Имеет общие представления о разработке проектной и рабочей технической документации, но допускает ошибки при перечислении требований и стандартов, необходимых при разработке технического задания на программную систему	Правильно перечисляет основные требования и стандарты, необходимые при разработке технического задания на программную систему, а также его состав и содержание.	Правильно перечисляет и объясняет все требования и стандарты, необходимые при разработке технического задания на программную систему
	Разрабатывает техническое задание на программную систему (У-5)	Правильные ответы на вопросы № 67-73 к экзамену	Допускает ошибки при разработке технического задания на программную систему	Разрабатывает техническое задание на программную систему с несущественными ошибками	Идеально разрабатывает полное техническое задание на программную систему.
	Имеет навыки разработки технического задания на программную систему (Н-5)	Правильные ответы на вопросы № 67-73 к экзамену	Имеет слабые навыки разработки проектной и рабочей технической документации на программную систему	Имеет навыки разработки проектной и рабочей технической документации на программную систему, но допускает незначительные ошибки при разработке технического задания	Демонстрирует уверенные навыки разработки технического задания на программную систему

3. Типовые контрольные задания для проведения промежуточной аттестации

а) Вопросы для оценки сформированности элементов компетенции ОПК-4:

1.Основные документы программного обеспечения.

2.Документы программного обеспечения: ГОСТ 34 серии.

3.Содержание руководства пользователя и руководства администратора.

4.Постановка задачи на проектирование программного обеспечения.

5.Понятие требования к программному обеспечению. Свойства требований.

6.Сбор, формулировка, анализ и документирование требований к программному обеспечению.

7. Понятие жизненного цикла. Модель жизненного цикла программы.

8.Процессы жизненного цикла: основные, вспомогательные, организационные. Состав и содержание работ каждого процесса.

9.Перечень и содержание работ стадий и этапов.

б) Вопросы для оценки сформированности элементов компетенции ПК-1:

10.Алгоритм сортировки выбором. Пример программной реализации.

11.Алгоритм быстрой сортировки. Пример программной реализации.

12.Алгоритм сортировки пузырьком. Пример программной реализации.

13.Алгоритм сортировки вставками. Пример программной реализации.

14.Алгоритм сортировки слиянием. Пример программной реализации.

15.Алгоритмы сортировки Шелла. Пример программной реализации.

16.Алгоритмы поиска кратчайшего пути в графе

17.Алгоритм волновой трассировки Ли.

18.Алгоритм Дейкстры.

19.Алгоритм А*.

20.Алгоритм Беллмана-Форда.

21.Алгоритм Джонсона.

22.Алгоритм Флойда-Уоршелла.

23.Алгоритм Левита.

24.Исследование и анализ предметной области.

в) Вопросы для оценки сформированности элементов компетенции ПК-2:

25.Эргономические требования к организации интерфейсов.

26.Основные понятия: Программно-технический комплекс. Технологии программирования. Понятие обеспечения системы. Виды обеспечения.

27.Основные понятия: Понятие программы, подпрограммы, сопрограммы. Свойства программы как объекта системы.

28.Понятие составляющих программного обеспечения: программный компонент, комплекс программ, пакет программ, программный модуль, библиотека программ, программная система, сборка программ, программный продукт, программная услуга.

29.Понятия: предметная область, конфигурация, окружение, ресурс, среда, задача, функция, функциональность, спецификация, уровень.

30.Классификация программ по назначению и по выполнению.

31.Понятие программы как изделия и основные задачи проектирования. Стадии и этапы проектирования.

32.Организация процесса проектирования программного обеспечения.

33.Методы проектирования: сверху-вниз, снизу-вверх. Схемы проектирования: каскадная, откатная, спиральная.

34.Распределение работ между участниками проектов. Взаимодействие участников в процессе проектирования.

35.Понятие и виды программирования. Внутренняя структура и сегментация программ.

36.Методологии программирования: процедурная, структурная, функциональная, логическая, объектно-ориентированная, визуальная, обобщенная.

37.Понятие языка программирования. Классификация языков программирования.

38.Уровни языка. Базовые составляющие языка программирования.

39.Понятие интерфейса. Классификация интерфейсов. Формы представления: текстовые и графические.

40.Способы организации: командные, диалоговые, оконные, языковые. Сценарии взаимодействия: жёсткие, свободные, иерархические, прямого манипулирования.

41.Виды диалогов: директивный, фразовый, табличный. Этапы разработки интерфейса.

42.Особенности языка PHP и его отличия от языка C. Интеграция языков PHP и HTML. Основные типы данных. Операции. Работа со строками.

43.Прикладные библиотеки: растровая графика и взаимодействие с СУБД mySQL. Примеры программ на PHP.

44.Особенности языка Node.js и его отличия от языка php и C++. Интеграция языков javascript и HTML. Разработка простого Web-сервера.

45.Особенности языка программирования Java и его отличия от языка C++. Пакеты, классы, объекты.

46.Особенности языка программирования Java. Простые типы данных, строки, массивы. Операторы. Исключения и их обработка. События.

47.Разработка распределенных приложений JAVA средствами CORBA. Примеры программ.

48.Понятие процесса, его свойства и атрибуты. Цикл жизни процесса. Порождение и завершение процессов.

49.Сигналы и реакция на них. Сигнальный механизм для синхронизации работы процессов. Обмен данных через программные каналы. Примеры программ.

50.Методы межпроцессного взаимодействия. Сообщения, семафоры и разделяемая память: создание, управление доступом, организация обмена, удаление. Отображение файлов в оперативную память. Примеры программ.

51.Многопроцессорные ЭВМ. Структура и методика использования потоков выполнения. Создание потоков и их завершение.

52.Объекты синхронизации потоков управления: взаимоисключающие блокировки, условные переменные, семафоры. Примеры программ.

53.Архитектура стека протоколов TCP/IP.

54.Организация сетевого взаимодействия посредством механизма “гнезд” (sockets). Потоковые и дейтаграммные гнезда.

55.Создание гнезд, привязка адреса, запросы на прием/передачу данных, завершение соединений. Примеры программ.

56.Механизм передачи сообщений согласно прикладному протоколу MPI (Message-Passing Interface). Взаимодействие “один к одному” и коллективное взаимодействие. Примеры программ.

57.Синхронные и асинхронные операции. Представление в сообщениях разнотипных данных. Примеры программ.

58.Группирование процессов. Виртуальные топологии процессов. Средства выполнения и мониторинга MPI-программ. Примеры программ.

59.Методы разработки справочных систем.

60.Оформление и комментирование программного кода. Примеры стилей оформления.

61.Структура презентации на программное обеспечение.

62.Структура данных стек. Пример программной реализации.

63.Структура данных очередь. Пример программной реализации.

64.Структура данных список. Пример программной реализации.

65.Структура данных дерево поиска. Пример программной реализации.

66.Сложные структуры данных. B-деревья.

67.Понятие ошибки. Основные виды ошибок: синтаксические, алгоритмические, структурные, концептуальные.

68.Понятия тестирования. Основные понятия: случай, контрольные данные, тест-план, протокол, покрытие.

69.Концепции чёрного, серого, белого ящика при тестировании.

70.Методы тестирования: индукции, дедукции, ручной, обратного прослеживания.

71.Составление тестовых планов.

72.Поиск ошибок в программном коде. Отладка.

73.Состав и содержание технического задания на разработку программного обеспечения.

5. Методические материалы для определения процедур оценивания знаний, умений и навыков, характеризующих этапы формирования компетенций.

Промежуточная аттестация по дисциплине проводится в соответствии с требованиями СТП СТО СПбГТИ(ТУ) 016-2015. КС УКДВ Порядок проведения зачетов и экзаменов.

По дисциплине промежуточная аттестация проводится в форме экзамена и зачёта.

Шкала оценивания на экзамене балльная («отлично», «хорошо», «удовлетворительно», «неудовлетворительно»), на зачёте – «зачёт», «незачет». При этом «зачёт» соотносится с пороговым уровнем сформированности компетенции.