ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ
ЮЖНО-УРАЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
КАФЕДРА «ПРИКЛАДНАЯ ИНФОРМАТИКА И МАТЕМАТИКА»

УЧЕБНО-МЕТОДИЧЕСКИЙ КОМПЛЕКС

Программирование и основы алгоритмизации

СПЕЦИАЛЬНОСТЬ

210100 -

«Управление и информатика в технических системах»

Миасс
2006

Внутренняя опись документов УМК

1. Требования к обязательному минимуму содержания образовательной программы
2. Учебный план дисциплины
3. График учебного процесса дисциплины
4. Рабочая программа дисциплины
5. Сведения об обеспеченности учебной литературой
6. Виды контрольных мероприятий и форм аттестации дисциплины
7. Экземпляры – образцы учебно-методической литературы
8. Образцы выполняемых работ

1 Требования к обязательному минимуму содержания образовательной программы

1 Содержание дисциплины

Обязательный минимум в соответствии с ГОС, приведен в таблице 1а

ОПД.Ф.07	Программирование и основы алгоритмизации: основные виды, этапы проектирования и жизненный цикл программных продуктов; синтаксис и семантика алгоритмического языка программирования; структурное и модульное программирование; типизация и структуризация программных данных; статические и динамические данные; сложные структуры данных (списки, деревья, сети); потоки ввода-вывода; файлы; проектирование программных алгоритмов (основные принципы и подходы); классы алгоритмов; методы частных целей, подъемы ветвей и границ, эвристика; рекурсия и итерация; сортировка и поиск; методы и средства объектно-ориентированного программирования; стандарты на разработку прикладных программных средств; документирование, сопровождение и эксплуатация программных средств.	130

2 Учебный план дисциплины

Дисциплина: Программирование и основы алгоритмизации

Кафедра: СУиММ

Распределение по семестрам

Экз.	Зач.	К.п.	К.р.	РГР
	2			
3				

Объем работы студентов

Всего	Из них				
	Всего ауд.	Лекции	Практики	Лаб. раб.	Сам. раб.
130	51	17		34	14
	51	17		34	14

Распределение по курсам и семестрам

1 курс		II курс		III курс		IV курс		V курс	
1	2	3	4	5	6	7	8	9	10
17	17	17	17	17	17	17	17	17	17
	3								
		3							

3 График учебного процесса

Структура дисциплины по видам занятий	Всего часов	II семестр					III семестр				
		сен	ок	ноя	дек	янв	фев	мар	апр	май	ию
1 Лекции	34	8	9				6	8	3		
2 Практические занятия											
3 Лабораторные занятия	68	8	10	8	8		6	10	8	6	2
4 консультации	8					4					4
5 Контрольная работа											
6 СРС	28			7	7				7	7	
7 Зачет											4
8 Экзамен						4					
Итого	138	16	19	15	15	8	12	18	18	15	10

Министерство образования Российской Федерации
Южно-Уральский государственный университет

Кафедра Автоматизация и управление

СОГЛАСОВАНО
Зав. выпускающей кафедрой
Автоматика

С.С. Голощапов

УТВЕРЖДАЮ
Декан электротехнического факультета

А.И. Телегин

РАБОЧАЯ ПРОГРАММА

Дисциплины Программирование и основы алгоритмизации
для специальности 210100 Управление и информатика в технических системах,
направление подготовки 651900 Автоматизация и управление
факультет Электротехнический
кафедра-разработчик Прикладная информатика и математика

Рабочая программа составлена в соответствии с Государственным образовательным стандартом высшего профессионального образования и примерной программой дисциплины по направлению подготовки
651900 Автоматизация и управление специальности 210100 Управление и информатика в технических системах

Рабочая программа рассмотрена и одобрена на заседании кафедры Автоматика протокол

Зав. кафедрой разработчика

В.И. Киселев

Ученый секретарь кафедры

Разработчик программы

В.Н. Переходюк

1 Введение
Рабочая программа составлена с учетом СТП 4.0-2003 Стандарт предприятия. Инструкция по процедуре управления качеством. Требования к качеству разработки рабочих программ учебных дисциплин.

3.1 Требования к уровню освоения содержания дисциплины.
В результате изучения дисциплины студенты должны:
- знать основные понятия программирования,
- знать базовый язык программирования,
- знать наиболее употребляемые алгоритмы,
- уметь решать задачи на персональных компьютерах.

3.2 Требования к уровню подготовки для освоения дисциплины.
Студенты должны быть знакомы с принципами работы вычислительной машины (ВМ), десятичной, двоичной, восьмеричной и шестнадцатеричной системами счисления, с основными понятиями информатики.

4 Цели и задачи преподавания и изучения дисциплины
Цель дисциплины состоит в поэтапном формировании у студентов следующих знаний и умений:
4.1 знание основных понятий программирования;
4.2 знание базового языка программирования;
4.3 знание наиболее употребляемых алгоритмов;
4.4 умение решать задачи на вычислительных машинах.

5 Объем дисциплины и виды учебной работы

Таблица 1 – Состав и объем дисциплины

Вид учебной работы	Всего часов	Распределение по семестрам в часах			
		Семестр			
		I	II	III	др.
Общая трудоемкость дисциплины	130	-	63	67	-
Аудиторные занятия	105	-	51	54	-
Лекции (Л)	35	-	17	18	-
Практические занятия (ПЗ)	70	-	34	36	-
Семинары (С) Лабораторные работы (ЛР)	-	-	-	-	-
Самостоятельная работа студентов (СРС)	25	-	12	13	-
Курсовой проект	-	-	-	-	-
Реферат и (или) другие виды самостоятельной работы	-	-	-	-	-
Вид итогового контроля (зачет, экзамен)		-	зачет	экз.	-
(указывается объем работы в соответствии с ГОС и учебным планом)	130 / 130	-	63	67	-

Таблица 2 – Разделы дисциплины, виды и объем занятий

Номер раздела, темы	Наименование разделов, тем дисциплины	Объем в часах по видам			
		всего	Л	ПЗ	СРС
1	Основные понятия алгоритмизации и программирования.	16	10	-	6
2	Базовый язык программирования С++	47	7	34	6
3	Динамические структуры данных. Динамическая память	34	10	18	6
4	Прикладное программирование	33	8	18	7
	Итого	130	35	70	25

4.2 Краткое содержание разделов.

Раздел 1. Рассматриваются общие вопросы программирования. Состав, структура программ. Общие принципы описания языков программирования. Уровни языков программирования.

Раздел 2. Рассматривается базовый язык С++. Типы данных, операторы и их использование. Выражения и операции.

Раздел 3. Изучение сложных структур базового языка С++. Указатели. Динамическая память. Динамические структуры данных.

Раздел 4. рассматриваются типовые задачи программирования. Поиск. Сортировка. Разные реализации.

Таблица 3- Содержание лекций и объем в часах

Раздел	Краткое содержание лекций	Объем в часах
1	1 Цели и задачи изучения языков программирования. Какой язык считать хорошим	2
	2 Основные виды, этапы проектирования и жизненный цикл программных продуктов. Фазы жизненного цикла: требования, проектирование, разработка, отладка, сопровождение, особенности больших и малых проектов программ.	2
	3 Алгоритмы и способы их представления. Базовые операции: присвоение, ввод/вывод; базовые операции: следование, развилка, цикл.	2
	4 Классификация языков программирования и их краткая характеристика. Машинные языки. Ассемблерные языки. Макроассемблерные языки. Машинно-независимые языки	2

	5 Язык программирования высокого уровня. ЯП и его описание. Алфавит, синтаксис, семантика. Определение. Структурное и модульное программирование. Дополнительные формы записи алгоритмов. Типизация и структуризация программных данных.	2
	Итого по разделу 1	10
2	6 Введение в С++. Структура и конструкция программы на Си/С++. Пример программы на С++. Комментарии, идентификаторы, служебные слова, константы.	1
	7 Типы данных и их атрибуты. Имена. Целые типы данных, логические, с плавающей точкой, тип void. Простые и производные типы данных. Данные. Класс хранения: область действия и время жизни. Внешние и внешние статические данные. Автоматические, регистровые и внутренние статические данные. Инициализация данных	1
	8 Производные типы данных. Массивы. Структуры	1
	9 Потоки ввода/вывода. Файлы.	1
	10 Операторы управления. Пустой оператор. Операторы-выражения. Операторы break и continue. Блок операторов. Оператор return. Оператор goto и метки операторов.	1
	11 Операторы выражений. Операторы ветвления if, switch. Оператор for, while, do-while	1
	12 Функции	1
	Итого по разделу 2	7
3	13 Производные типы данных. Классификация типов. Массивы. Описание. Размещение в памяти. Операции. Начальная инициализация. Массивы переменной длины. Алгоритмы работы с одномерными массивами. Многомерные массивы.	2
	14 Указатели. Поля битов и побитовые операции	2
	15 Динамическая память. Выделение динамической памяти простых типов данных, массивов, матриц	2
	16 Динамические структуры данных. Линейные списки. Стеки.	1
	17 Динамические структуры данных. Очереди. Бинарные деревья.	1
	18 Базовые понятия объектно-ориентированного программирования. Отличие от традиционного модульного подхода. Инкапсуляция. Наследование. Полиморфизм	1
	19 Краткое описание стандартной библиотеки языка.	1
	Итого по разделу 3	10
4	20 Технология создания программ. Кодирование и документирование программы. Проектирование и	2

	тестирование программы. Этапы создание программ	
	21 Среда разработки программ. Отладка программного проекта. Устранение логических (алгоритмических ошибок)	2
	22 Поиск. Сортировки. Сортировка простым обменом	2
	23 Быстрая сортировка Ч. Хора	2
	Итого по разделу 4	8
	Итого по разделам 1,2,3,4	35

Контроль усвоения конкретных лексических единиц студентам предлагаются вопросы из Приложения 1.

5 Лабораторные работы не предусматриваются.

6 Практические занятия

6.1 В таблице 4 представлен: объем, содержание, характер и цель практических занятий.

Таблица 4 – Состав и объем практических занятий

Раздел		Краткое содержание занятий	Характер занятий и цель	Объем в часах
2	1	Освоение программ линейной структуры. Ввод из файла. Вывод в файл.	Выполнение базового и индивидуального заданий. Приобретение навыков и опыта работы в С++. Вычисление простых и сложных выражений	4
	2	Освоение программ с ветвящейся структурой. Оператор if.		4
	3	Освоение программ с ветвящейся структурой. Оператор switch		4
	4	Освоение программ с циклической структурой. Оператор for.	Выполнение базового и индивидуального заданий. Приобретение навыков и опыта работы в С++ с циклами	4
	5	Освоение программ с циклической структурой. Оператор while.		3
	6	Освоение программ с циклической структурой. Оператор do_while.		3
	7	Освоение программ с использованием строчных переменных.	Приобретение опыта для работы со строками.	4
	8	Функции. Программы, включающие составление и	Приобретение опыта для работы с	4

		вызов функций.	функциями	
	9	Структуры. Считывание, выборка и обработка структур.	Приобретение опыта для работы со структурами	4
			Итого по разделу 2	34
4	10	Обработка массива. Поиск максимума (минимума) в массиве.	Приобретение опыта работы с алгоритмами поиска и сортировки с динамической памятью. Выполнение базовых и индивидуальных заданий	6
	11	Сортировка массива простым выбором с использованием указателей.		6
	12	Обработка матриц с использованием алгоритмов поиска и сортировки динамической памяти		6
	13	Быстрая сортировка массива методом Ч. Хора	Получения знаний и опыта для работы со сложными алгоритмами и динамическими структурами	6
	14	Быстрая сортировка массива методом Ч. Хоора с использованием динамических структур (стека)		6
	15	ООП. Среда разработки С++ Builder. Составление программы с использованием оконного интерфейса.	Приобретение знаний и опыта для работы в ООП.	6
			Итого по разделу 4	36
			Итого по виду практические занятия	70

Примеры выполнения базовых заданий и варианты индивидуальных заданий по каждому занятию приведены в Приложении 2.

6.2 Контроль за практическими занятиями осуществляется преподавателем, путем консультаций и наблюдением в процессе выполнения базового и индивидуального заданий, а также собеседования со студентом при сдаче и учете выполнения заданий.

7 Семинарские занятия не предусмотрены.

8 Самостоятельная работа студента.

Самостоятельная работа студента заключается в дополнительном изучении разделов и тем в соответствии с приведенной таблицей.

Таблица 3- рекомендуемое содержание тем СРС объем в часах

Раздел	Краткое содержание лекций	Объем в часах

2	1 Структуры. Поиск в массиве структур. Сортировка.	4
	9 Потоки ввода/вывода в С++. Бинарные файлы.	4
	12 Функции. Передача массивов в функцию.	4
	Итого по разделу 2	12
4	Графика. Холст. Карандаш. Кисть. Графические примитивы. Иллюстрации. Битовые образы. Мультипликации.	7
	Базы данных. создание базы данных в С++ Builder.	6
	Итого по разделу 4	13

5 Сведения об обеспеченности учебной литературой

5 Учебно-методическое обеспечение

5.1 Рекомендуемая литература, имеющаяся в библиотеке

а) основная литература

1 Павловская Т.А. С/С++ Программирование на языке высокого уровня: Учебник. - Санкт-Петербург: Изд-во ПИТЕР, 2006-460 с.

2 Павловская Т.А., Щупак Ю.А. С/С++ Программирование на языке высокого уровня. Структурное программирование. Практикум: Учебное пособие. - Санкт-Петербург: Изд-во ПИТЕР, 2002-237 с.

3 Керниган Б.У., Пайк Р. Практика программирования: Пер с англ.- Москва: Изд. Дом "Вильямс", 2004-287 с.

4 Конова Е.А., Поллак Г.А., Ткачев А.М. Основы программирования на языке С: Учебное пособие.- Челябинск: Изд-во ЮУрГУ, 2005.-251 с.

б) дополнительная литература

1 Керниган Б.У., Ритчи Д., Фьюер А. Язык программирования С: Задачи по языку С: Пер с англ. - Москва: Изд -во. Финансы и статистика, 1985-287 с.

2 Подбельский В.В., Фомин С.С. Программирование на языке С: - Москва: Финансы и статистика, 2000-600 с.

3 Давыдов В.Г., Программирование и основы алгоритмизации: Учебное пособие. – Москва: Высшая школа.2003-448 с.

4 Культин Н.Б. С/С++ в задачах и примерах: Санкт-Петербург, Изд-во "БХВ-Петербург", 2004 – 288 с.

5 Культин Н.Б. Самоучитель С/С++ Builder: Санкт-Петербург, Изд-во "БХВ-Петербург", 2004 – 320 с.

9.2 Средства и материально-техническое обеспечение дисциплины

Компьютерный класс, оснащенный ПК не ниже 256 МГц с ОС Windows 95 и выше или Windows NT. Версия языка высокого уровня Borland С++5.0 и выше.

Приложение 1.

6 Виды контрольных мероприятий и форм аттестации дисциплины. . Вопросы для контроля основных понятий

1 Объявление переменных

Объявите переменные,

1.1 необходимые для вычисления площади прямоугольника.

1.2 необходимые для пересчета веса из фунтов в килограммы.

1.3 Определите исходные данные и объявите переменные, необходимые для вычисления дохода по вкладу.

1.4 Объявите переменные, необходимые для вычисления площади круга.

1.5 Объявите переменные, необходимые для вычисления площади кольца.

1.6 Объявите переменные, необходимые для вычисления объема и площади поверхности цилиндра.

1.7Объявите переменные, необходимые для вычисления стоимости покупки, состоящей из нескольких тетрадей, карандашей и линейки.

1.8 Объявите переменные, необходимые для вычисления стоимости покупки, состоящей из нескольких тетрадей и такого же количества обложек.

2 Инструкция присваивания

2.1Запишите программу, которая присваивает переменной x значение -1,5.

2.2 Запишите программу, которая присваивает переменной summa нулевое значение.

2.3 Запишите программу, которая увеличивает на единицу значение переменной n.

2.4 Запишите программу, которая уменьшает на два значение переменной counter.

2.5 Запишите программу вычисления среднего арифметического переменных xl и x2.

2.6 Запишите в виде инструкции присваивания формулу вычисления значения функции y = -2,7x3+0,23x2-1,4.

2.7 Запишите программу, которая увеличивает значение переменной x на величину, находящуюся в переменной dx.

2.8 Запишите в виде инструкции присваивания формулу пересчета веса из фунтов в килограммы (один фунт — это 405,9 грамма).

2.9 Запишите в виде инструкции присваивания формулу пересчета расстояния из километров в версты (одна верста — это 1066,8 м).

2.10 Запишите в виде инструкции присваивания формулу вычисления площади прямоугольника.

2.11 Запишите в виде инструкции присваивания формулу вычисления площади треугольника: s = 0.5*a*h, где a — длина основания; h — высота треугольника.

3 Вывод

3.1Написать программу, которая выводит на экран ваши имя и фамилию.
3.2Написать программу, которая выводит на экран путь к файлу stdio.h.
3.3Написать программу, которая выводит на экран четверостишие:
Унылая пора! Очей очарованье! Приятна мне твоя прощальная краса — Люблю я пышное природы увяданье, В багрец и золото одетые леса. А. С. Пушкин
3.4Написать программу вывода значений переменных a, b и c (типа float) с пятью цифрами целой части и тремя — дробной, в виде:
a = значение b = значение c = значение
3.5Написать программу вывода значений переменных h и l (типа float), которые содержат значения высоты и длины прямоугольника. Перед значением переменной должен быть пояснительный текст (высота= ширина"), а после — единица измерения (см).

4 Ввод

4.6 Написать программу, обеспечивающую ввод с клавиатуры значения переменных radius типа float.
4.7Написать инструкции, которые обеспечивают ввод значений дробных (тип float) переменных и и г. Предполагается, что пользователь после набора каждого числа будет нажимать клавишу <Enter>.
4.8Написать программу, которая обеспечивает ввод значений переменных и и г. Предполагается, что пользователь будет набирать числа в одной строке.
4.9Объявите необходимые переменные и напишите фрагмент программы вычисления объема цилиндра, обеспечивающий ввод исходных данных.
4.10 Объявите необходимые переменные и напишите инструкции ввода исходных данных для программы вычисления стоимости покупки, состоящей из нескольких тетрадей и карандашей. Предполагается, что пользователь будет вводить данные о каждой составляющей покупки в отдельной строке: сначала цену, затем количество.

5 Опреатор if

5.1Написать программу, которая вычисляет частное двух чисел. Программа должна проверять правильность введенных пользователем данных и, если они неверные (делитель равен нулю), выдавать сообщение об ошибке. Ниже приведен рекомендуемый вид экрана во время выполнения программы (данные, введенные пользователем, выделены полужирным шрифтом).
вычисление частного.
введите в одной строке делимое и делитель, затем нажмите <Enter>
Вы ошиблись. Делитель не должен быть равен нулю.
5.2Написать программу, которая проверяет, является ли год високосным. Ниже приведен рекомендуемый вид экрана во время программы. Данные, введенные пользователем, выделены: полужирным шрифтом.,
введите год, например 2000, и нажмите <Enter>
2001год - не високосный

для завершения нажмите <Enter>

6 Оператор switch

6.3Напишите программу, которая запрашивает у пользователя номер дня недели, затем выводит название дня недели или сообщение об ошибке, если введены неверные данные.
6.4Написать программу, которая вычисляет стоимость междугородного телефонного разговора (цена одной минуты определяется расстоянием до города, в котором находится абонент). Исходными данными для программы являются код города и длительность разговора. Ниже приведены коды некоторых городов и рекомендуемый вид экрана во время работы программы (данные, введенные пользователем, выделены полужирным шрифтом).

7 Цикл for

7.1Написать программу, которая выводит на экран ваши имя и фамилию 10 раз.

7.2Написать программу, которая выводит таблицу квадратов первых десяти целых положительных чисел. Ниже приведен рекомендуемый вид экрана во время работы программы.
7.3Написать программу, которая выводит таблицу квадратов первых пяти целых положительных нечетных чисел. Ниже приведен рекомендуемый вид экрана во время работы программы.

8 цикл do while

8.4Написать программу, вычисляющую сумму и среднее арифметическое последовательности положительных чисел, которые вводятся с клавиатуры. Ниже приведен рекомендуемый вид экрана во время выполнения программы (данные, введенные пользователем, выделены полужирным шрифтом). ->
8.5Вычисление среднего арифметического последовательности положительных чисел.
8.6Вводите после стрелки числа. Для завершения ввода введите ноль.
8.7Написать программу, которая определяет максимальное число из введенной с клавиатуры последовательности положительных чисел (длина последовательности неограниченна). Ниже приведен рекомендуемый вид экрана во время выполнения программы (данные, введенные пользователем, выделены полужирным шрифтом).

9 Цикл while

9.8Напишите программу, которая выводит на экран таблицу значений функции y = 2x2-5x-8 в диапазоне от -4 до 4. Шаг изменения аргумента 0,5.
9.9Напишите программу, которая вычисляет число "Пи" с заданной пользователем точностью. Для вычисления значения числа "Пи" воспользуйтесь тем, что значение частичной суммы ряда 1-1/3+1/5-1/7+1/9-

..• при суммировании достаточно большого количества членов приближается к значению п4. Рекомендуемый вид экрана во время выполнения программы приведен ниже (данные, введенные пользователем, выделены полужирным шрифтом).

Задайте точность вычисления Пи -> 0.001
Значение числа ПИ с точностью 0.001000 равно 3.143589
Просуммировано 502 члена ряда!

10 Массивы

10.1 Написать программу, которая вводит с клавиатуры одномерный массив из 5 целых чисел, после чего выводит количество ненулевых элементов. Перед вводом каждого элемента должна выводиться подсказка с номером элемента.

Ввод массива целых чисел.
После ввода каждого числа нажмите <Enter>
a[1] -> 12
a[2] -> 0
a[3] -> 3
a[4] » -1
a[5] -> 0
В массиве 3 ненулевых элемента.

10.2 Написать программу, которая выводит минимальный элемент введенного с клавиатуры массива целых чисел. Ниже приведен рекомендуемый вид экрана во время работы программы (данные, введенные пользователем, выделены полужирным шрифтом).

Поиск минимального элемента массива.
Введите в одной строке элементы массива (5 целых чисел)
и нажмите <Enter>
-> 23 0 45 -5 12
Минимальный элемент массива: -5

11 Символы и строки

11.1 Написать программу, которая запрашивает имя пользователя и здоровается с ним. Рекомендуемый вид экрана во время выполнения программы приведен ниже (данные, введенные пользователем, выделены полужирным шрифтом).

11.2 Как Вас зовут?
11.3 Введите свои имя и фамилию, затем нажмите <Enter>
11.4 -> Вася Иванов
11.5 Здравствуйте, Вася Иванов! '

11.6 Написать программу, которая запрашивает у пользователя имя и отчество, затем здоровается с ним. Для ввода используйте функцию getchO.

11.7 Напишите программу, которая вычисляет длину введенной с клавиатуры строки.

11.8 Напишите программу, которая выводит на экран сообщение в "телеграфном" стиле: буквы сообщения должны появлять ся по одной, с некоторой задержкой.

11.9 Напишите программу, которая выводит код введенного пользователем символа. Программа должна завершать работу в результате ввода, например, точки. Рекомендуемый вид экрана во время выполнения программы приведен ниже (данные, введенные пользователем, выделены полужирным шрифтом).

Введите символ и нажмите <Enter>.
Для завершения введите точку.
1
Символ: 1 Код: 49
2

12 Функции

12.1 Написать функцию, которая вычисляет объем цилиндра. Параметрами функции должны быть радиус и высота цилиндра.

12.2 Написать функцию, которая возвращает максимальное из двух целых чисел, полученных в качестве аргумента.

12.3 Написать функцию, которая сравнивает два целых числа и возвращает результат сравнения в виде одного из знаков: >, < или=.

12.4 Написать функцию, которая вычисляет сопротивление цепи, состоящей из двух резисторов. Параметрами функции являются величины сопротивлений и тип соединения (последовательное или параллельное). Функция должна проверять корректность параметров: если неверно указан тип соединения, то функция должна возвращать -1.

12.5 Написать функцию, которая вычисляет значение аь. Числа а и Ь могут быть любыми дробными положительными числами.

12.6 Написать функцию Procent, которая возвращает процент от полученного в качестве аргумента числа.

12.7 Написать функцию "Факториал" и программу, использующую эту функцию для вывода таблицы факториалов.

12.8 Написать функцию Dohod, которая вычисляет доход по вкладу. Исходными данными для функции являются: величина вклад; процентная ставка (годовых) и срок вклада (количество дней).

12.9 Написать функцию glasn, которая возвращает 1, если символ, полученный функцией в качестве аргумента, является гласной буквой русского алфавита, и ноль — в противном случае.

Приложение 2 –

7 Экземпляры- образцы учебно-методической литературы

Индивидуальные варианты практических заданий и программы на языке С++ базовых вариантов.

Задание 10. Обработка массивов. Поиск минимума (максимума) в массиве.

Выполненное задание должно содержать следующие разделы:

постановка задачи, алгоритм на псевдокоде, текст программы, пример исходных данных и результатов.

Задание выполняется в трех модификациях в соответствии с представленными ниже пунктами содержания 1.1, 1.2, 1.3.

Содержание задания.

1.1 В массиве mas[n] целых чисел найти значение элемента массива и его номер для условий поиска и и заданного значения x (далее в таблице варианты индивидуальных заданий).

1.2 Удалить найденный элемент с изменением длины массива.

1.3 Использовать указатели для обращения к элементам массива.

вариант	Условие поиска	
1	Min ($\|mas_1\|$,…, $\|mas_n\|$)	Для $mas_i > x$
2	Max ($\|mas_1\|$, …, $\|mas_n\|$)	Для $mas_i < x$
3	Min (mas_1 , …, mas_n)	Для $\|mas_i\| > x$
4	Max (mas_1, …, mas_n)	Для $\|mas_i\| < x$
5	Min (mas_1 , …, mas_n)	Для $mas_i^2 > x$
6	Max (mas_1, …, mas_n)	Для $mas_i^2 < x$
7	Min (mas_1^2, …, mas_n^2)	Для $mas_i > x$
8	Max (mas_1^2, …, mas_n^2)	Для $mas_i < x$
9	Min ($\|mas_1\|$,…, $\|mas_n\|$)	Для $mas_i^3 > x$
10	Max ($\|mas_1\|$, …, $\|mas_n\|$)	Для $mas_i^3 < x$
11	Min ($\|mas_1\|$,…, $\|mas_n\|$)	Для $mas_i >= x$
12	Max ($\|mas_1\|$, …, $\|mas_n\|$)	Для $mas_i <= x$
13	Min (mas_1 , …, mas_n)	Для $\|mas_i\| >= x$
14	Max (mas_1, …, mas_n)	Для $\|mas_i\| <= x$
15	Min (mas_1 , …, mas_n)	Для $mas_i^2 >= x$
16	Max (mas_1, …, mas_n)	Для $mas_i^2 <= x$
17	Min (mas_1^2, …, mas_n^2)	Для $mas_i >= x$
18	Max (mas_1^2, …, mas_n^2)	Для $mas_i <= x$
19	Min ($\|mas_1\|$,…, $\|mas_n\|$)	Для $mas_i^3 >= x$
20	Max ($\|mas_1\|$, …, $\|mas_n\|$)	Для $mas_i^3 <= x$

mas

44 55 12 42 94 18 06 67 19 39 20 11 19 51 29 03

Программа на языке С++ базового варианта

//Задание 1.1 Вариант 1


```
//В одномерном массиве найти минимальный по абсолютной величине элемент
и его номер
//Исходные данные: n,mas[n]
//Результат: i,mas[i]
//Текст программы
//----------------------------
#pragma hdrstop        // учет особенности компилятора
#include <stdio.h>
#include <conio.h>
#include <math.h>
#include <time.h>
#pragma argsused       // учет особенности компилятора
void main()
{
time_t t;
time(&t);
printf("                    %s", ctime(&t));
int im,x,n,mas[]={44,55,12,42,94,18,06,67,19,39,20,11,19,51,29};
x=1e6;
printf("input n\n");
scanf("%d",&n);
printf(" value n=%2i\n",n);
if(n>15)
printf("Error - n > size mas[]\n");
for(int i=0;i<n;i++)
if(abs(mas[i])<x)
{ x=mas[i];
im=i;  }
printf("index i=%d element mas[i]=%d\n", im,mas[im]);
getch();
}
//---------------------------------------
//Задание  1.2, Вариант 1
//В одномерном массиве найти минимальный по абсолютной
//величине элемент и его номер
//удалить найденный элемент с изменением длины массива
//Исходные данные: n,mas[n]
// Результат: i,mas[i]
//Текст фрагмента программы – удаление найденного элемента
printf("index i=%d element mas[i]=%d\n", im,mas[im]);
for(int i=im; i<n;i++)
mas[i]=mas[i+1];
n--1;
for(int i=1;i<n;i++)
printf("%3i",mas[i]);
```

```
getch();
}
//Задание 1.3, Вариант 1
//применение указателей при работе с массивами
//В одномерном массиве найти минимальный по абсолютной
//величине элемент и его номер
//удалить найденный элемент с изменением длины массива
//Исходные данные: n,mas[n]
//Результат: i,mas[i],mas[n]
//Текст программы
//----------------------------
#pragma hdrstop        // учет особенности компилятора
#include <stdio.h>
#include <conio.h>
#include <math.h>
#include <time.h>
//---------------------------------------------------------------------------
#pragma argsused       // учет особенности компилятора
void main()
{
time_t t;
time(&t);
printf("                %s", ctime(&t));
int im,x,n,mas[]={44,55,12,42,94,18,06,67,19,39,20,11,-19,51,29};
x=1e6;
for(int i=0;i<15;i++)
printf("%3i",*(mas+i));
printf("\n input n\n");
scanf("%d",&n);
printf(" value n=%2i\n",n);
if(n>15)
printf("Error - n > size mas[]\n");
for(int i=0;i<n;i++)
if(abs(*(mas+i))<x)
{
x=*(mas+i);
im=i;
}
printf("index i=%d min element mas[i]=%d\n", im,*(mas+im));
for(int i=im; i<n;i++)
*(mas+i)=*(mas+(i+1));
n--;
for(int i=0;i<n;i++)
printf("%3i",*(mas+i));
getch();
```



```
time(NULL);
}
```

Задание 2 – Простая сортировка выбором с использованием указателей

Выполненное задание должно содержать следующие разделы:

постановка задачи, алгоритм на псевдокоде, текст программы, пример исходных данных и результатов.

Варианты заданий

вариант	Условие сортировки
1	По убыванию $\|mas_i\|$
2	По возрастанию $\|mas_i\|$
3	По убыванию mas_i
4	По возрастанию mas_i
5	По убыванию mas_i <0
6	По возрастанию mas_i<0
7	По убыванию mas_i >0
8	По возрастанию mas_i >0
9	По убыванию $\|mas_i\|$ > x
10	По возрастанию $\|mas_i\|$> x
11	По убыванию mas_i > x
12	По возрастанию mas_i> x
13	По убыванию $\|mas_i\|$< x
14	По возрастанию $\|mas_i\|$< x
15	По убыванию mas_i < x
16	По возрастанию mas_i < x
13	По убыванию $\|mas_i\|$<= x
14	По возрастанию $\|mas_i\|$<= x
15	По убыванию mas_i < =x
16	По возрастанию mas_i <= x

Программа на языке С++ базового варианта

```
//Задание  2, select mas [] Вариант 1
//сортировкв массива простым выбором
//применение указателей при работе с массивами
//В одномерном массиве провести сортирову
//по убыванию абсолютных значений элементов
//Исходные данные: n, mas[n]
//Резуьтат: mas[n]
//Текст программы
//----------------------------
#pragma hdrstop        // учет особенности компилятора
#include <stdio.h>
```

```
#include <conio.h>
#include <math.h>
#include <time.h>
//---------------------------------------------------------------------------

#pragma argsused        // учет особенности компилятора
void main()
{
FILE *f_out=fopen("z2_select.rez","w");
time_t t;
time(&t);
printf("                    %s", ctime(&t));
int im,x,n,mas[]={44,55,12,42,94,18,06,67,19,39,20,11,19,51,29};
printf("\n input n\n");
scanf("%d",&n);
printf(" value n=%2i\n",n);
if(n>15)
printf("Error - n > size mas[]\n") ;
else
for(int i=0;i<n;i++)
printf("%3i",i);
printf("\n");
for(int i=0;i<n;i++)
printf("%3i",*(mas+i));//---------------
for(int k=n-1;k>=1;k--)
{
im=0;x=1e6;
for(int i=0;i<=k;i++)
if(abs(mas[i])<x)
{
im=i;
x=*(mas+im);
}
*(mas+im)=*(mas+k);
*(mas+k)=x;
//}
printf("\n k= %d min element mas[i]=%d\n",k,x);
for(int j=0;j<n;j++)
printf("%3i",*(mas+j));
}
printf("\n");
for(int i=0;i<n;i++)
fprintf(f_out,"%3i",*(mas+i));
getch();
fclose(f_out);
```



```
}
//------
```

Практическое задание 12

Двумерные массивы (матрицы). Выполнить программу по упорядочиванию элементов матрицы в соответствии с вариантом задания.

Рекомендуется выполнять каждое задание в двух вариантах: используя локальные и динамические массивы. Размерности локальных массивов задавать именованными константами, значения элементов массива — в списке инициализации. Ввод данных в динамический массив выполнять из файла.

Вариант 1 Дана целочисленная прямоугольная матрица. Определить;

1) количество строк, не содержащих ни одного нулевого элемента;

2) максимальное из чисел, встречающихся в заданной матрице более одного

Вариант 2 Дана целочисленная прямоугольная матрица. Определить количество столбцов, не содержащих ни одного нулевого элемента.

Характеристикой строки целочисленной матрицы назовем сумму ее положительных четных элементов. Переставляя строки заданной матрицы, расположить их в соответствии с ростом характеристик.

Вариант 3 Дана целочисленная прямоугольная матрица. Определить;

1) количество столбцов, содержащих хотя бы один нулевой элемент;

2) номер строки, в которой находится самая длинная серия одинаковых элементов.

Вариант 4 Дана целочисленная квадратная матрица. Определить;

1) произведение элементов в тех строках, которые не содержат отрицательных элементов;

2) максимум среди сумм элементов диагоналей, параллельных главной диагонали матрицы.

Вариант 5 Дана целочисленная квадратная матрица. Определить;

1) сумму элементов в тех столбцах, которые не содержат отрицательных элементов;

2) минимум среди сумм модулей элементов диагоналей, параллельных побочной диагонали матрицы.

Вариант 6 Дана целочисленная прямоугольная матрица. Определить;

1) сумму элементов в тех строках, которые содержат хотя бы один отрицательный элемент;

2) номера строк и столбцов всех седловых точек матрицы.

Примечание. Матрица A имеет седловую точку A_{ij} если A_{ij} является минимальным элементом в i-й строке и максимальным в j-м столбце.

Вариант 7 Для заданной матрицы размером 8 на 8 найти такие k, что k-я строка матрицы совпадает с k-м столбцом.

Найти сумму элементов в тех строках, которые содержат хотя бы один отрицательный элемент.

Вариант 8 Характеристикой столбца целочисленной матрицы назовем сумму модулей его отрицательных нечетных элементов. Переставляя столбцы заданной матрицы, расположить их в соответствии с ростом характеристик.

Найти сумму элементов в тех столбцах, которые содержат хотя бы один отрицательный элемент.

Вариант 9 Соседями элемента A_{ij} в матрице назовем элементы A_{kl} с $i - 1 \le k < i+1$, $j-1 \le l \le j+l$; $(k, l) \ne (i, j)$. Операция сглаживания матрицы дает новую матрицу того же размера, каждый элемент которой получается как среднее арифметическое имеющихся соседей соответствующего элемента исходной матрицы. Построить результат сглаживания заданной вещественной матрицы размером 10 на 10. В сглаженной матрице найти сумму модулей элементов, расположенных ниже главной диагонали.

Вариант 10

Элемент матрицы называется локальным минимумом, если он строго меньше всех имеющихся у него соседей. Подсчитать количество локальных минимумов заданной матрицы размером 10 на 10. Найти сумму модулей элементов, расположенных выше главной диагонали.

Вариант 11 Коэффициенты системы линейных уравнений заданы в виде прямоугольной матрицы. С помощью допустимых преобразований привести систему к треугольному виду. Найти количество строк, среднее арифметическое элементов которых меньше заданной величины.

Вариант 12 Уплотнить заданную матрицу, удаляя из нее строки и столбцы, заполненные нулями. Найти номер первой из строк, содержащих хотя бы один положительный элемент.

Вариант 13 Осуществить циклический сдвиг элементов прямоугольной матрицы на п элементов вправо или вниз (в зависимости от введенного режима), п может быть больше количества элементов в строке или столбце.

Вариант 14 Осуществить циклический сдвиг элементов квадратной матрицы размерности M x N вправо на к элементов таким образом; Элементы 1-й строки сдвигаются в последний столбец сверху вниз, из него — в последнюю строку справа налево, из нее - в первый столбец снизу вверх, из него — в первую строку; для остальных элементов — аналогично.

Вариант 15 Дана целочисленная прямоугольная матрица. Определить номер первого из столбцов, содержащих хотя бы один нулевой элемент.

Характеристикой строки целочисленной матрицы назовем сумму ее отрицательных четных элементов. Переставляя строки заданной матрицы, расположить их в соответствии с убыванием характеристик.

Вариант 16 Упорядочить строки целочисленной прямоугольной матрицы по возрастанию количества одинаковых элементов в каждой строке.

Найти номер первого из столбцов, не содержащих ни одного отрицательного элемента.

Вариант 17 Путем перестановки элементов квадратной вещественной матрицы добиться того, чтобы ее максимальный элемент находился в левом верхнем углу, следующий по величине — в позиции (2,2), следующий по

величине — в позиции (3,3) и т. д., заполнив таким образом всю главную диагональ.

Найти номер первой из строк, не содержащих ни одного положительного элемента.

Вариант 18 Дана целочисленная прямоугольная матрица. Определить;

1) количество строк, содержащих хотя бы один нулевой элемент;

2) номер столбца, в котором находится самая длинная серия одинаковых элементов.

Вариант 19 Дана целочисленная квадратная матрица. Определить;

1) сумму элементов в тех строках, которые не содержат отрицательных элементов;

2) минимум среди сумм элементов диагоналей, параллельных главной диагонали матрицы.

Вариант 20 Дана целочисленная прямоугольная матрица. Определить;

1) количество отрицательных элементов в тех строках, которые содержат хотя бы один нулевой элемент;

2) номера строк и столбцов всех седловых точек матрицы.

Примечание. Матрица А имеет седловую точку A_{ij} если A_{ij} является минимальным элементом в i-й строке и максимальным в j-м столбце.

Программа на языке С++ базового варианта

```
//Задание 3 программа сортировки строк прямоугольной
 //целочисленной матрицы  по возрастанию сумм их элементов
 //
#pragma hdrstop

//---------------------------------------------------------------------------

#pragma argsused
#include <stdio.h>    //
#include <conio.h>
#include <stdlib.h>
#include <time.h>


void main()
{
time_t t;
time(&t);
FILE *f_in, *f_out;
unsigned rows, cols, i,j;
f_in=fopen("sel_mat.dat","r");
f_out=fopen("sel_mat.rez","w");
fprintf(f_out,"            %s", ctime(&t));
```

```
fscanf(f_in,"%i %i",&rows, &cols);          //read rows,cols
fprintf(f_out,"rows %i cols %i\n",rows,cols); //write rows,cols
unsigned **pm=new unsigned * [rows];      //din memory rows
for (i=0; i<rows; i++)                    //din memory cols
pm[i] = new unsigned [cols];
for (i=0; i<rows; i++)                    //read pmx
for (j=0; j<cols; j++)
fscanf(f_in,"%i",&pm[i][j]);
fprintf(f_out,"m[][]\n");
for(i=0; i<rows; i++)                     //write pmx
{
for(j=0; j<cols; j++)
fprintf(f_out,"%4d", pm[i][j]); //------
fprintf(f_out,"\n");          //------
}
long *sum = new long [rows];              //din memory sum [rows]
for(i=0; i<rows; i++){                    //calc sum
sum[i]=0;
for(j=0; j<cols; j++) sum[i]+=pm[i][j];
}
for(i=0; i<rows; i++){                    //control/write sum
fprintf(f_out," result sum=%4d\n", sum[i]);
for(j=0; j<cols; j++)
fprintf(f_out,"%4d", pm[i][j]); //------
fprintf(f_out,"\n");
}
long temp_s;
int nmin, temp_pm;
for(i=0; i<rows-1; i++){       //select matr
nmin=i;
for(j=i+1; j<rows; j++)
 if(sum[j] < sum[nmin]) nmin=j;
 temp_s=sum[i];
 sum[i]=sum[nmin];
 sum[nmin]=temp_s;
for(j=0; j<cols; j++){
temp_pm=pm[i][j];
pm[i][j]=pm[nmin][j];
pm[nmin][j]=temp_pm;
}
}
fprintf(f_out,"out result\n ");
for(i=0; i<rows; i++)                     //write pmx
{
for(j=0; j<cols; j++)
```



```
fprintf(f_out,"%4d", pm[i][j]); //------
fprintf(f_out,"\n");          //------
}
fclose(f_in);
fclose(f_out);
time(NULL);
for (i=0; i<rows; i++)
delete [ ] pm[i];
delete [ ] pm;
delete sum;
//getch( );
}
//-------------
Содержание файла "sel_mat.dat"
2 8
44 55 12 42 94 18 06 67
54 12 42 94 18 06 67 44
Содержание файла "sel_mat.rez"
          Fri Dec 29 17:56:47 2006
rows 2 cols 8
m[][]
 44  55  12  42  94  18   6  67
 54  12  42  94  18   6  67  44
 result sum= 338
 44  55  12  42  94  18   6  67
 result sum= 337
 54  12  42  94  18   6  67  44
out result
 54  12  42  94  18   6  67  44
 44  55  12  42  94  18   6  67
```

Задание 4 – Сложная сортировка Хоара с использованием динамической памяти

Написать, отладить и распечатать в среде C++ текст программы.
По заданию оформить отчет, состоящий из следующих пунктов:

1 Постановка задачи.
2 Алгоритм на псевдокоде.
3 Текст программы.
4 Пример исходных данных и результатов.

Варианты индивидуальных заданий по теме "Сортировка"

Номер варианта	Дана матрица *A[N,[M]*
1	Упорядочить каждую строку матрицы по убыванию
2	Найти четыре минимальных по абсолютной величине элемента в *D* -ом столбце *(D <= M)*
3	Упорядочить каждый столбец матрицы по убыванию абсолютных величин
4	Найти три максимальных элемента в предпоследней строке матрицы
5	Упорядочить каждую строку матрицы по возрастанию абсолютных величин
6	Найти четыре максимальных по абсолютной величине элемента в *S*-ом столбце *(S <= M)*
7	Найти три минимальных элемента в *D*-ой строке (*D <= N*)
8	Упорядочить каждую строку матрицы по убыванию абсолютных величин
9	Найти два максимальных по абсолютной величине элемента в предпоследней строке матрицы
10	Упорядочить каждый столбец матрицы по возрастанию
11	Найти три минимальных элемента в предпоследнем столбце матрицы
12	Найти три максимальных элемента в каждом столбце матрицы
13	Упорядочить каждый столбец матрицы по возрастанию абсолютных величин
14	Найти три минимальных элемента в последней строке матрицы
15	Упорядочить каждую строку матрицы по возрастанию
16	Найти три минимальных по абсолютной величине элемента в каждой строке матрицы
17	Упорядочить каждый столбец матрицы по убыванию
18	Найти четыре максимальных по абсолютной величине элемента в предпоследнем столбце матрицы
19	Найти два минимальных элемента в каждой строке матрицы
20	Найти три максимальных элемента в последнем столбце матрицы

Программа на языке С++ базового варианта

```
/*Задание  4, q_select  сортировка массива mas []
-методом быстрой сортировки
-применение указателей при работе с массивами
В одномерном массиве провести сортировку
по возрастанию значений элементов
Исходные данные: n, mas[n]
Резуьтат: mas[n]
Текст программы  */
#include <stdio.h>
#include <conio.h>
#include <math.h>
#include <time.h>
#include <stdlib.h>
int main()
{
```



```
FILE *f_out=fopen("z2_select.rez","w");
time_t t;
time(&t);
printf("                    %s", ctime(&t));
int im,x,n,i,j,mas[]={44,55,12,42,94,18,06,67,19,39,20,11,-19,51,29};
printf("\n input n\n");
scanf("%d",&n);
printf(" value n=%2i\n",n);
fprintf(f_out," value n=%2i\n",n);
if(n>15)
printf("Error - n > size mas[]\n") ;
else
for(i=0;i<n;i++)
fprintf(f_out,"%3d",i);
fprintf(f_out,"\n");
for(i=0;i<n;i++)
fprintf(f_out,"%3d",mas[i]);
fprintf(f_out,"\n");
                    //---------------
float  middle, temp;
int *stackl = new int [n], *stackr = new int [n],sp = 0;
int  left, right;

                        // Сортировка
sp = 1; stackl[1] = 0; stackr[1] = n-1;        //1
while (sp > 0)                                 //2
{                                    // Выборка из стека последнего запроса
left = stackl[sp];                             //3
right = stackr[sp];                            //4
sp--;                                          //5
while(left < right)                            //6
{                                  // Разделение (arr[left] .. arr[right]}
i = left; j = right;                           //7
middle = mas[(left + right) / 2];              //8
while (i < j)                                  //9
{while (mas[i] < middle) i++;                  //10
while (middle < mas[j]) j--;                   //11
if (i <= j)
{temp = mas[i]; mas[i] = mas[j]; mas[j] = temp;
i++; j--;
}
}
 if (1 < right)                                //12
{                                  // Запись в стек запроса из правой части
sp++ ;
```

```
stackl[sp] = i;
stackr[sp] = right;
}
right = j;                    // Теперь left и right ограничивают левую часть
                                 //13
}
}
                          // Вывод результата
fprintf(f_out,"result working program\n");
for(i=0;i<n;i++)
fprintf(f_out,"%3i",*(mas+i));
getch();
fclose(f_out);
return 0;
}
//---------------------------------------------------------------------------
```

Ссодержание файла q_select.rez

```
 value n=12
  0  1  2  3  4  5  6  7  8  9 10 11
 44 55 12 42 94 18  6 67 19 39 20 11
result working program
  6 11 12 18 19 20 39 42 44 55 67 94
```

Задание 5 – Сложная сортировка Ч.Хоара с использованием динамической памяти и динамических данных (стека)

Написать, отладить и распечатать в среде C++ текст программы.

По заданию оформить отчет, состоящий из следующих пунктов:

5 Постановка задачи.
6 Алгоритм на псевдокоде.
7 Текст программы.
8 Пример исходных данных и результатов.

Варианты индивидуальных заданий по теме "Сортировка"

Номер варианта	Дана матрица *A[N][M]*
1	Упорядочить каждую строку матрицы по убыванию
2	Найти четыре минимальных по абсолютной величине элемента в *D* -ом столбце *(D <= M)*
3	Упорядочить каждый столбец матрицы по убыванию абсолютных величин
4	Найти три максимальных элемента в предпоследней строке матрицы
5	Упорядочить каждую строку матрицы по возрастанию абсолютных величин
6	Найти четыре максимальных по абсолютной величине элемента в *S*-ом столбце *(S <= M)*
7	Найти три минимальных элемента в *D*-ой строке *(D <= N)*
8	Упорядочить каждую строку матрицы по убыванию абсолютных величин
9	Найти два максимальных по абсолютной величине элемента в предпоследней строке матрицы
10	Упорядочить каждый столбец матрицы по возрастанию
11	Найти три минимальных элемента в предпоследнем столбце матрицы
12	Найти три максимальных элемента в каждом столбце матрицы
13	Упорядочить каждый столбец матрицы по возрастанию абсолютных величин
14	Найти три минимальных элемента в последней строке матрицы
15	Упорядочить каждую строку матрицы по возрастанию
16	Найти три минимальных по абсолютной величине элемента в каждой строке матрицы
17	Упорядочить каждый столбец матрицы по убыванию
18	Найти четыре максимальных по абсолютной величине элемента в предпоследнем столбце матрицы
19	Найти два минимальных элемента в каждой строке матрицы
20	Найти три максимальных элемента в последнем столбце матрицы

Программа на языке C++ базового варианта

/* Zadanie 5- Быстрая сортировка массива вещественных чисел

методом быстрой сортировки Ч.Хоара

с использованием динамической памяти и динамических структур данных.

исходные данные

- длина массива
- массив вещественных чисел

результат работы программы

-массив, отсортированный по возрастанию значений*/

```
#include <stdio.h>
#include <conio.h>
struct Node {
int left, right;
Node* p; };

Node* push(Node* top, const int l, const int r);
Node* pop(Node* top, int &l, int &r);
int main( )
{
int i, n;
float middle, temp;
FILE *f_in,*f_out;
f_in=fopen("z5_dat.txt","r");
```

```
f_out=fopen("z5.rez","w");
printf("begin wohrking program\n");
 if (!f_in) { printf(" no founded  z5_dat.txt\n") ;
 getch();
  return 0;}
  else
  fscanf(f_in, "%i", &n);
  fprintf(f_out,"n=%2d\n",n);

float * mas = new float[n];
for (i = 0; i < n; i++) fscanf(f_in,"%f",&mas[i]);
for (i = 0; i < n; i++) fprintf(f_out,"%3.0f", mas[i]);
fprintf(f_out,"\n");
getch();                                  // Сортировка
int j, left, right;
Node* top = 0;
top = push(top, 0, n - 1);
while (top)
{ top = pop(top, left, right);
while (left < right) {          // Разделение {,arr[left] .. arr[right] }
i = left; j = right;
middle = mas[(left + right) / 2];
while (i < j)
{ while (mas[i] < middle) i++;
while (middle < mas[j]) j--;
if (i <= j)
{ temp = mas[i];
      mas[i] = mas[j];
      mas[j] = temp;
      i++; j--; }
 }
if (i < right)
{                    // Запись в Стек запроса из правой части
top = push(top, i, right);
}
right = j;
                // Теперь left и right ограничивают левую часть
}
}
                              // Вывод результата
fprintf(f_out,"result wohrking program\n");
for (i = 0; i < n; i++) fprintf(f_out,"%3.0f", mas[i]);
fprintf(f_out,"\n");
delete mas;
printf("\n damping spasebar");
```



```
getch();
return 0;
}
                        // function - Занесение в стек
Node* push(Node * top, const int l, const int r)
{
Node* pv = new Node;
pv->left = l;
pv->right =r;
pv->p = top;
 return pv;
}
                        //function - Выборка из стека
Node* pop(Node* top, int &l, int &r)
 {
  Node * pv = top->p;
 l = top->left;
 r = top->right;
 delete top;
 return pv;
 }
```

Содержание файла z5_dat.txt	Содержание файла z5.rez
10	n=10
44 55 12 6 67 94 64 28 32 31	44 55 12 6 67 94 64 28 32 31
	result wohrking program
	6 12 28 31 32 44 55 64 67 94

8 Образцы выполняемых работ

Задание 6 – Технология визуального проектирования и событийного программирования

Разработка приложения для вычисления силы тока по формуле I=U/R.

Предварительные сведения.

Запуск C++ Builder 6 , далее File/New/Application

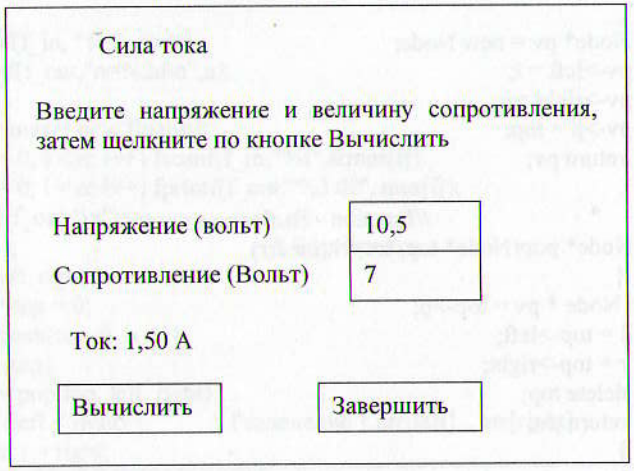
Вид экрана после запуска С:

- главное окно – **C++ Builder 6**, содержит меню команд, панели инструментов и палитру компонентов,
- окно стартовой формы – **Form1**, содержит заготовку главного окна разрабатываемой программы (приложения),
- окно редактора свойств объектов – **Object Inspektor**, предназначено для редактирования свойств объектов; *объекты* – диалоговые окна и элементы управления (поля ввода/вывода, командные кнопки, переключатели и т.д.); *свойства объекта* – определяют вид, положение и поведение объекта,
- окно просмотра (выбора) списка объектов – **Object TreeView**,

- окно редактора кода – **Unit1.cpp,** в котором набирается текст (код) программы.

Первоначально окно кода Unit1.cpp почти полностью закрыто окном стартовой формы - Form1.

Окно формы после разработки программы:



1 Изменение заголовка окна Form1 на Сила тока – последовательность действий:

в окне **Object Inspektor** щелкнуть левой кнопкой мыши в строке **Caption** и вести в выделенное значение свойства текст *Сила тока* .

2 Аналогично проводится изменение других свойств окна в соответствии с таблицей:

Свойство	**Значение**	**Комментарий**
Caption	Сила тока	
Height	200	
Width	330	
BorderStyle	bsSingle	Тонкая граница не позволяет изменить размер окна во время работы программы путем захвата и перемещения границы
BorderIcons.biMinimize	False	В заголовке нет кнопки Свернуть
BorderIcons.biMaximize	False	В заголовке нет кнопки Развернуть
Font.Name		
Font.Size	10	

3 Для ввода исходных данных ввести поля редактирования при помощи компонентов пользовательского интерфейса.

Последовательность действий.

3.1 В палитре компонентов на вкладке **Standard** щелкнуть на значке компонента *Edit* (**ab**) – компонент *Edit* появится на форме.

3.2 Повторить действия для введения второго компонента.

В результате в окне формы появятся два компонента *Edit1* и *Edit2*. Компоненты автоматически отображаются в окнах **Object Inspektor** и **Object TreeView.** Размер и положение компонентов на форме можно изменить при помощи мыши; свойства компонентов автоматически отображаются в окне **Object Inspektor.**

Значение свойств компонентов устанавливаются в соответствии с таблицей.

Свойство	Компонент	
	Edit1	Edit2
Text		
Top	48	72
Left	144	144
Width	65	65

4 Ввод текста в окно программы.

4.1 В палитре компонентов на вкладке **Standard** щелкнуть на значке компонента *Label* (**A**) – компонент *Label1* появится на форме.

В форму разрабатываемого приложения необходимо ввести четыре фрагмента текста (компоненты *Label)*:

- *Label1* – для ввода информационного сообщения,
- *Label2* , *Label3* – для ввода информации о назначении полей ввода,
- *Label4* – для вывода результата расчета.

После ввода компонентов осуществляют их настройку в соответствии с таблицей.

Если поле *Label* должно содержать несколько строк текста, перед тем как ввести в поле текст (изменить значение поля *Caption*), нужно присвоить свойству *AutoSize* значение false, а свойству *WordWrap – true*. Затем установить требуемый размер поля (при помощи мыши или вводом значений свойств *Height* и *Width*) и только после этого ввести значение свойства *Caption*.

Свойство	Компонент			
	Label1	*Label2*	*Label3*	*Label 4*
AutoSize	*false*	*true*	*True*	*false*
WordWrap	*true*	*false*	*False*	*true*
Caption	Введите напряжение и величину сопротивления, затем щелкните по кнопке Вычислить	Напряжение (вольт)	Сопротивление (Ом)	
Top	8	56	80	112
Left	8	8	8	8
Height	33	16	16	16

Width	300	120	120	200

5 Ввод в окно программы командных кнопок.
5.1 В палитре компонентов на вкладке **Standard** щелкнуть на значке компонента *Button* (**OK**) – компонент *Button1* появится на форме.
В форму разрабатываемого приложения необходимо ввести два фрагмента текста (компоненты *Button)*:
- *Button1* – для ввода команды **Вычислить**,
- *Button2* – для ввода команды **Завершить**.
5.2 Установить свойства командных кнопок *Button1 и Button2* в соответствии с таблицей.

Свойство	Компонент	
	Button1	*Button2*
Caption	Вычислить	Завершить
Top	144	144
Left	16	104
Height	25	25
Width	75	75

6 При щелчке по командным кнопкам происходит так называемое *событие OnClick*, которое затем должно быть программно обработано. Это производится *функцией обработки события*. Ввод значений в поле ввода относится к стандартным событиям *OnKeyPress*, для которых функцию обработки берет на себя компонент. Функцию обработки события при вводе команды **Вычислить** произведем в следующем порядке.
6.1 Выбрать компонент Button1 в окне **Objekt Inspector** или двойным щелчком на изображении компонента в окне формы.
В окне **Objekt Inspector** выбрать вкладку **Events** (События);
В левой колонке вкладки **Events** выбрать событие *OnClick*, а в соответствующем правом поле сделать двойной щелчок для для выбора функции, которая будет обрабатывать это событие.
В результате этого
- в окне **Objekt Inspector** рядом с именем события появится сгенерированное имя функции обработки события *TForm1::Button1Click* (первая часть имени идентифицирует форму (*Form1*), содержащую компонент, вторая часть – сам компонент *Button1* и событие *Click*),
- откроется окно редактора кода обработки события, с шаблоном функции *TForm1::Button1Click*.
6.2 В окне генератора кода между фигурными скобками набирается прогамма обработки события.

```
void_fastcall TForm1::Button1Click (Tobjekt *Sender)
{
float u; //напряжение
float r; //сопротивление
```



```
float i; //ток
    //выборка данных в виде строки символов из полей ввода
u=StrToFloat(Edit1->Text); //из поля редактирования Edit1
r=StrToFloat(Edit2->Text); // и Edit2 путем обращения к свойству Text
                    //функция StrToFloat преобразует стоку в число
    //вычисление тока
i=u/r;
// вывод результата в поле метки путем присвоения значения свойству Caption
Label4->Caption="Ток i "+ FloatToStrF(i,ffGeneral,7,2) + "A";
                    //функция FloatToStrF преобразует число в строку, т.к.
                    //Caption имеет строковый тип
}
```

7 Аналогично выполняется процедура обработки события **Завершить**.
Программа должна завершить работу – закрыть окно программы при помощи метода *Close*.

```
void_fastcall TForm1::Button2Click(Tobjekt *Sender)
{
Form1->Close();
};
```

8 Переключение между окном кода и окном формы реализуется при помощи команды **View/Toggle Form (Unit)**, клавиши **F12** или соответствующих кнопок на панели инструментов.
9 Редактор кода поддерживает функцию *контексно-зависимую подсказку* при наборе текста в виде краткой справочной информации.
10 В окне редактора кода находится также *навигатор классов (ClassExplorer)* в виде списка объектов проекта и его элементов. Выбрав элемент списка можно быстро перейти к нужному фрагменту кода.
11 В процессе набора кода удобно использовать шаблоны кода (Code Templates), в общем виде конструкции программы. Шаблоны предлагает редактор, если нажать комбинацию клавиш <Ctrl> <J>. Из списка можно выбрать необходимый фрагмент и вставить в текст командой <Enter>.
12 В процессе набора кода можно получить справку о конструкции языка, типе данных, классе или функции. Для этого в окне редактора необходимо набрать слово, о котором надо получить справку и нажать F1. Или из меню выбрать команду **C++ Builder Help**, затем на вкладке **Предметный указатель** ввести ключевое слово.
13 *Проект* – это набор файлов, которые компилятор использует при создании исполняемого фала: описания (bpr-файл), главного модуля (cpp-файл), ресурсов (res-файл), описания формы (dmf-файл), заголовочный (h-файл), описания функций формы (cpp-файл).
Сохранения проекта реализуется командой **File/Save Project As**, при этом в начале предлагается сохранить модуль-содержимое редактора кода **Save Unit1 As**, после выбора папки и имени /OK будут созданы три файла cpp, h и dmf, затем предлагается сохранить имя проекта **Save Project As**. Т.к. компилятор

генерирует cpp-файл главного модуля проекта, то чтобы его отличать от cpp-файла описаний кода имена файла модуля и проекта должны быть разными,

14 **Компиляция** реализуется командой **Project/Compile**, результат которой отражается в окне **Compiling**: при отсутствии ошибок дается сообщение **Done: Compiling Unit** или **Done: There are errors.**

Для перехода к фрагменту кода с ошибкой – щелкнуть по строке с сообщением об ошибке в нижней части экрана левой кнопкой мыши и из контекстного меню выбрать команду **Edit Source.**

15 Ошибки на этапе компиляции носят синтаксический характер и устраняются в процессе отладки и повторной компиляции. В таблице приведены типичные ошибки и соответствующие сообщения компилятора.

Сообщения	Ошибки
Undefined symbol (неизвестный символ)	Используется необъявленная переменная. Имя переменной, функции или параметра записано неверно.
Statement missing ; (отсутствует точка с запятой)	После инструкции не поставлена точка с запятой.
Unterminated string or character consnant (незаконченная строковая или символьная константа)	В конце строковой константы, например, текста нет двойных кавычек.
) expected (ожидается закрывающая скобка)	При записи арифметического выражения, содержащего ошибки, нарушен баланс открывающих и закрывающих скобок.
if statement missing ((в инструкции if нет открывающей скобки)	В конструкции if условие не заключено в скобки
Compound statement missing }	Нарушен баланс открывающих и закрывающих фигурных скобок.
Extra parametr in call to (лишний параметр при вызове функции)	Неверно записана инструкция вызова функции, указан лишний параметр.

15 Компоновка файлов проекта выполняется командой **Project/Make**, а также **Project/Build** после принудительной перекомпиляции.

16 Запуск программы можно реализовать командой **Run**.

17 При возникновении ошибок при выполнении программы на экран выводится окно **Debugger Exception Notification** с сообщением об ошибке и ее типе. Выполнение может быть прервано командой **Run/Program Reset** или продолжено при команде **Run/Step Over**. Обработка исключения может быть добавлена в программу.

18 После отладки программы может быть задано ее название при помощи команды **Project/Options/Aplication/Title**-Сила тока.

9 Тесты. Используются как контрольные работы и для формирования вопросов на экзамене.

Основные разделы курса

1. Ввод
2. Вывод
3. Простейшие ветвления
4. Циклы
5. Структуры
6. Функции
7. Области действия переменных
8. Массивы и указатели
9. Работа с динамической памятью и операции с линейным списком
10. Препроцессор, перечисления, функции с умалчиваемыми, значениями аргументов, перезагрузка функций, шаблоны функций, перезагрузка операций.

1 **Ввод в языках C/C++**. Варианты тестов

Далее приведены фрагменты программного кода, содержащего определения некоторых переменных. В комментариях к определениям переменных указано, какие значения переменных нужно ввести.

Написать фрагмент программы, обеспечивающий:

- открытие файла (потока C) “Input” для работы с файлом операционной системы “Test2.in”;
- ввод из этого файла (потока C) значений переменных, указанных в комментариях к программному фрагменту, соответствующего варианта;
- закрытие файла.

Указать, как при этом будут выглядеть строки исходных данных в файлу операционной системы “Test2.in”. Предусмотреть контроль коррекности значений, возвращаемых функциями библиотеки C “fopen”, “fscan”. Подключить необходимые стандартные заголовочные файлы.

Вариант 1		**Вариант 2**	
Double d;	//4.7	long double b	//4.7
char c[3];	//”ой”	char s[3]	//”я”
unsigned long uli;	//31	long i;	//-1
short si;	//12	short j;	//12
char c;	//’r’		
int i;	//-21		
Вариант 3		**Вариант 4**	
long double b;	//4.7e2	double b;	//4.7
char s[20];	//”4”	char s[20]	//”отлично”
unsigned j;	//31	long int i;	//-21
int i;	//0x2	unsigned long j;	//0x12

Вариант 5	**Вариант 6**

float a, //1.5
b; //14.7
int i, //-21
J; //12
char c1, //'y'
C2, //'p'
C3, //'a'
C4, //'!'
S[20]; //"прочитанная-строка"
	float b; //14.0
	int j; //12
	unsigned u; //21
	char c4, //'p'
	s[20]; //"зимний-вечер

Вариант 7	**Вариант 8**

double d; //2.0
float a, //1.5
b; //12.21
int i; //-21
unsigned j; //0x12
char c1, //'1'
c2, //'2'
c3, //'3'
c4, //'4'
s[20];//"1"
	long double d; //1.5
	float a; //1.5
	long int i; //-1
	unsigned j; //13
	char c1, //'4'
	s[20]; //"9"

Вариант 9	**Вариант 10**

float a; //1.5
int i, //21
j; //-12
char c1, //'a'
c2, //'e'
s[20]; //"прочитанная-строка"
	float b, //5.0
	k; //15.12
	lng int a; //27
	char c1, //'a'
	s[20];//"строка"

Вариант 11	**Вариант 12**
Написать фрагмент программы для чтения из файла следующих значений a: 1.5 i:21 j:12 c:'.' s:"строка" float a,b; int i,j; char c1,s[20]; Файл чтения имеет следующий вид -21 1.5 . Это строка 12	Какие значения получат переменные RetCode, a, b, i, j, c1,c2,c3 float a,b; int i,j; char c1,c2,c3; int RetCode; RetCode=fscanf(stdin,"%i 3%d %c %c %f %f", &i,&j,&c1,&c2,&c3,&b,&a) Файл чтения имеет следующий вид -77 -243567 2.4E3 14.7 172
Вариант 13	**Вариант 14**
Какие значения получат переменные RetCode, a, b, i, j, c1,c2,c3 float a,b; int i,j; char c1,c2,c3; int RetCode;	Ввести из файла следующие данные: a:1.5 b:4.7 i:-21 j:12 c1:'.' s: "string" float a, b; int i,j; char c1,s1[20]; Файл чтения имеет следующий вид

RetCode=fscanf(stdin,"%d 2%ld %c 5%c %c %f %f", &i,&j,&c1,&c2,&c3,&b,&a) Файл чтения имеет следующий вид -77 -243567 2.4E3 14.7 172	String 12 1.5 -21 4.7 -.-
Вариант 15	**Вариант 16**
Ввести из файла следующие данные: a:1.5 b:4.7 i:-21 j:0x12 float a; double b; int i; unsigned j; Файл чтения имеет следующий вид 0x12 1.5 -21 4.7	Ввести из файла следующие данные: a:1.5 b:4.7 i:-21 j:0x12 float a; double b; long int i; unsigned long j; Файл чтения имеет следующий вид 0x12 -1.5 -21 4.7
Вариант 17	**Вариант 18**
Какие значения получат переменные retcode, a, b, i, j, c1,c2,c3 float a,b; int i,j; char c1,c2,c3; int retcode; retcode=fscanf(stdin,"%i 4%d %c %c %c %f %f", &i,&j,&c1,&c2,&c3,&a,&b) Файл чтения имеет следующий вид -17 -1234567 2.4E3 172 1.5	Ввести из файла следующие данные: a:1.5 b:4.7 i:1 j:12 c1:'.' s:"это хорошо" float a, b; int i, j; char c1, s[20]; Файл чтения имеет следующий вид Это хорошо 1.5 12 . 1 4.7
Вариант 19	**Вариант 20**
Ввести из файла следующие данные: u:21 b:14.7 j:12 c4:'p' s:"зима-вечер" float b; int j; unsigned u; char c4, s[20]; Файл чтения имеет следующий вид Зима-вечер p 12 21 1 4.7	Ввести из файла следующие данные: u:21 b:14.7 j:12 c4:'p' s:"зима-вечер" float b; int j; unsigned u; char c4, s[20]; Файл чтения имеет следующий вид "Ура-вечер" 'p' 12 21 1 4.7

Тесты. Контрольные работы.

3. Вывод в языках Си/С**. Варианты тестов

Ниже приведены варианты фрагмента программного кода, содержащего вывод в файл (поток Си) "stdout". Укажите, как будут выглядеть строки вывода в файл (поток Си) "stdout" после выполнения заданного фрагмента. Для удобства в приведенных вариантах фрагментов программного кода символ "^" обозначает пробельный символ.

Вариант 1

float r;

```
int i = 17;
r = 1.5f * 2.0elf;
fprintf(stdout, "*r=%5.2e^%s^*x=%-+d\"n*%-3s\n", r,"_",i,"*",);
```

Вариант 2

```
float r;
int i = 17;
r = 1.5 * 2.0;
fprintf( stdout, "*r=%5.2f^%5s^*i=%-+10d\n*%-30s\n", r, "_",i "*",);
```

Вариант 3

```
double r;
int i = 17;
r = 1.543 * 2.0;
fprintf( stdout, "*r=%5.2lf^%-4s^*i=%-+10d\n*%-30s\n",r, "_", i, "*",);
```

Вариант 4.

```
float r = 3.0;
int i = 17;
fprintf( stdout,"*r=%5.2f^%5s^*i=%-+10d\n*%-30s\n",
r, "_"i, "*", );
```

Вариант 5.

```
float r = 1.5e2;
int x = 7;
fprintf( stdout, "^%30s\n^r=%f^%5s^i=%10d\n", "*",r, "^", i);
```

Вариант 6.

```
float r = 1.5e2;
int x = 5;
fprintf( stdout, "^r=%f^%-5s^i=%+10d\n^%2.3s", r, "*", i, "строка" );
```

Ниже приведены варианты фрагментов программного кода, содержащие определения данных и их инициализацию.

Написать фрагмент программы, обеспечивающий:

- Открытие файла (потока Си) "Output" для работы с файлом операционной системы "Test3. out".
- Вывод в открытый поток "Output" строк заданного вида. Указание. При выводе максимально использовать указанные в вариантах данные и возможности форматированного вывода.
- Закрытие файла (потока Си).

Предусмотреть контроль корректности значений, возвращаемых функциями библиотеки Си fopen и fclose. Подключить необходимые стандартные заголовочные файлы.

Вариант 7.

Фрагмент Си-программы:

```
long d = 254;
double f = 1234.56;
char str[ ] = "Строка 1";
```

Вид выводимых строк (ниже знак ^ обозначает пробел):

```
[+254^^]^^[^^254]
(+1234.6^)(1.234560E+03^^)
```

Вариант 8. Фрагмент Си-программы:

```
int     d = 254;
float   f = 1234.56;
char    str[ ] = "Строка";
```

Вид выводимых строк (ниже знак ^ обозначает пробел):

```
[^^+254]^[254]
(+1234.6^) (1.234560E+03)
/^^^^^^^^^^Стр/
```

Вариант 9. Фрагмент Си-программы:

```
int     d = 254;
float   f = 1234.56;
char    str[ ] = "Строка";
```

Вид выводимых строк (ниже знак ^ обозначает пробел):

```
[^^+254]^[254]
(+1234.6^) (1.234560E+03)
/^^^^^^^^^^Стр.
```

Вариант 10. Фрагмент Си-программы:

```
int     d =254;
float   f = 1234.56;
char    *str = "Строка символов";
```

Вид выводимых строк (ниже знак ^ обозначает пробел):

```
[+254]^^[^^254]
(^^1234. 6) ^*(1.234560E+03)
```

Вариант 11. Фрагмент Си-программы:

```
int     d = 254;
tloat   f = 1234.56;
char    *str ="Строка символов";
 [+254^^^^^^^]^^[^^^^^254]
(^^^^^1234.5600)^^(1234. 5600^^^^^)
/Стр/^^/м/
```

Вариант 12. Фрагмент Си-программы:

```
int     d = 113;
float   f = 12.34;
cbmr    *str= "Строка символов";
```

[+113^^^^^^]^^[^^^^^254]
(^^^+12.3)
/Строка^^^^^/^^/ил/

Вариант 13. Фрагмент Си-программы:
int d = 254;
float f = 1234.56;
char *str = "Строка символов";
Вид выводимых строк:
(^^^^254)^^(^^254^^^^^^^^)
(+1234.6)^^(^^^^^1.23E+03)
/^^^^^Строка/^^/^^лов/

Вариант 14. Фрагмент Си-программы:
int d = 254;
float f = 1234.56;
char *str = "Строка символов";
Вид выводимых строк :
[^^^^254]^^[^^254^^^^^^^^]
(1234.6)^^ (^^^^^1.234E+03)
/Стр^^^^^/^^/мв/

Вариант 15. Фрагмент Си-программы:
iat d = 254;
float f = 1234.56;
char *str = "Строка символов";
Вид выводимых строк:
[-254^^^^^]^^[^^^^+254]
(^^^+1234.5600)
/Строка симв^^^^/^^/т/

Вариант 16. Фрагмент Си-программы:
int d = 123;
float f = 1234.56;
char *str = "Прочитанная строка";
Вид выводимых строк:
[^^^^+123]^^[1234.6^^^^^]
(123^^^^^)
/^^^^^^^^Просто/

Вариант 17. Фрагмент Си-программы:
±nt d = 123;
float f = 1234.56;
char *str = "Прочитанная строка";
Вид выводимых строк:

[^^^^^+123]^^[+1234.6^^^^^]
(^^^^^123.)
/^^^^^^^^Прок/

Вариант 18. Фрагмент Си-программы:
int d = 254;
float .f = 1234.56;
chax *str = "Строка символов";
Вид выводимых строк:
[+254]^^[^^^^^254]
(^^1234.6)^^(1.234560E+03)
^^^^^^^^ Стр/^^ /м/

Вариант 19. Фрагмент Си-программы:
int d = 123;
float f = 1234.56;
char *str = "Прочитанная строка";
Вид выводимых строк:
[^^^^^+123]^^[1234.6^^^^^]
(123^^^^^)
/^^^^^^^^Прос/

Вариант 20. Фрагмент Си-программы:
int d = 123;
float f = 1234.56,
char *str = "Прочитанная строка";
Вид выводимых строк:
[^^^^^+123]^^[+1234.6^^^^^]
(^^^^^123.)
/^^^^^^^^прок.

z := x в

стейшие ветвления. Варианты тестов

Вариа
следующему

нт 1. С помощью операторов ветвлений и присваивания записать
ограммы, вычисляющий значение переменной n по следующему

при i=1 или i=5,
при i=7 или i=12,
в остальных случаях

нт 2. С помощью операторов ветвлений и присваивания записать
ограммы, вычисляющий значение переменной n по следующему

с. 103. Ф
при a>0 и b=0,
при a<=0 и b>=0,
в остальных случаях

Вариа
следующем
if(c <

нт 3. Изобразить фрагмент схемы алгоритма, соответствующий
фрагменту программы:
3) ±£ (c. -«» 2) a++; mime b++; a +- 1;

Вари
следующем

нт 4. Изобразить фрагмент схемы алгоритма, соответствующий
фрагменту программы:
3) if (c ==2) a++; else b++; a += 1;

нт 5. С помощью операторов ветвлений и присваивания записать
ограммы, вычисляющий значение переменной n по следующему

и i=1 или 2 или 7,
ри i=10,
остальных случаях

ис. 104. Ф

нт 6. Изобразить фрагмент схемы алгоритма, соответствующий
фрагменту программы;
1) a++; else if(c == 2) a -- ; else if(c == 3) a += 1;

Вари
фрагмент
правилу:
n= n
n =
n = a

нт 7. С помощью операторов ветвлений и присваивания записать
ограммы, вычисляющий значение переменной n по следующему

ри 1=4,
при i=1 или 7 или 9
в остальных случаях.

Вар
следующе
if(c

нт 8. С помощью операторов ветвлений и присваивания записать
ограммы, вычисляющий значение переменной z по следующему

z= x+5 при a>2 и b=0,
z = a+b при a<0,
z = x в остальных случаях

Вариант 9. Изобразить фрагмент схемы алгоритма, соответствующий следующему фрагменту программы:
if(c < 3) if(c == 2) a++; else b++; if(c < 2) C++; a += 1;

Вариант 10. Записать фрагмент программы, соответствующий следующему фрагменту схемы программы (рис. 101):

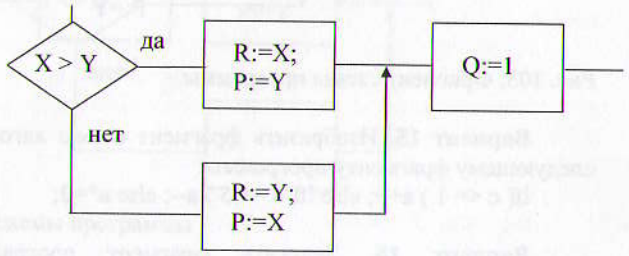


Рис. 101. Фрагмент схемы программы

Вариант 11. Записать фрагмент программы, соответствующий следующему фрагменту схемы программы (рис. 102):

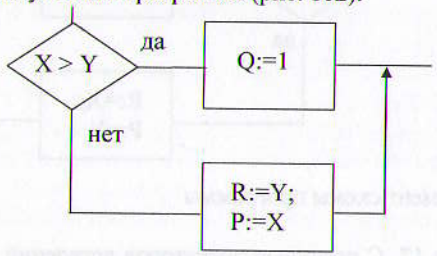


Рис. 102. Фрагмент схемы программы

Вариант 12. Изобразить фрагмент схемы алгоритма, соответствующий следующему фрагменту программы:
if (c < 3) if(c == 2) a + + ; else b+ + ; if(c < 2) C+ + ;
else a += 1; { C++; b++; }

Вариант 13. Изобразить фрагмент схемы алгоритма, соответствующий следующему фрагменту программы:
z := x+5 при i=1,3,5;
z := a+b при i=2,4,6;

4. Простейшие ветвления. Варианты тестов

Вариант 1. С помощью операторов ветвлений и присваивания записать фрагмент программы, вычисляющий значение переменной n по следующему правилу:

n = n+1 при i=1 или i=5,
n = a+b при i=7 или i=12,
n = a-b в остальных случаях

Вариант 2. С помощью операторов ветвлений и присваивания записать фрагмент программы, вычисляющий значение переменной n по следующему правилу:

n= n+1 при a>0 и b=0,
n = a+b при a<=0 и b>=0,
n = a-b в остальных случаях

Вариант 3. Изобразить фрагмент схемы алгоритма, соответствующий следующему фрагменту программы:

if (c < 3) ±£ (c. -«» 2) a++; mime b++; a +- 1;

Вариант 4. Изобразить фрагмент схемы алгоритма, соответствующий следующему фрагменту программы:

if (c < 3) if (c ==2) a++; else b++; a += 1;

Вариант 5. С помощью операторов ветвлений и присваивания записать фрагмент программы, вычисляющий значение переменной n по следующему правилу:

n=1 при i=1 или 2 или 7,
n = 2 при i=10,
n = 0 в остальных случаях

Вариант 6. Изобразить фрагмент схемы алгоритма, соответствующий следующему фрагменту программы;

if(c == 1) a++; else if(c == 2) a -- ; else if(c == 3) a += 1;

Вариант 7. С помощью операторов ветвлений и присваивания записать фрагмент программы, вычисляющий значение переменной n по следующему правилу:

n = 1 при 1=4,
n = a+b при i=1 или 7 или 9
n = a-b в остальных случаях.

Вариант 8. С помощью операторов ветвлений и присваивания записать фрагмент программы, вычисляющий значение переменной z по следующему правилу:

z= x+5 при a>2 и b=0,
z = a+b при a<0,
z = x в остальных случаях

Вариант 9. Изобразить фрагмент схемы алгоритма, соответствующий следующему фрагменту программы:

if(c < 3) if(c == 2) a++; else b++; if(c < 2) C++; a += 1;

Вариант 10. Записать фрагмент программы, соответствующий следующему фрагменту схемы программы (рис. 101):

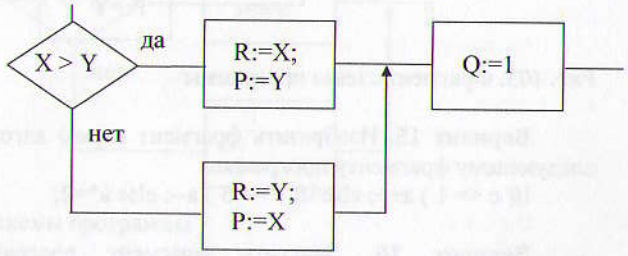


Рис. 101. Фрагмент схемы программы

Вариант 11. Записать фрагмент программы, соответствующий следующему фрагменту схемы программы (рис. 102):

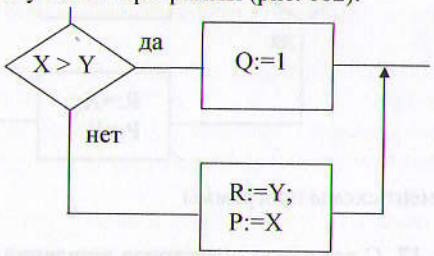


Рис. 102. Фрагмент схемы программы

Вариант 12. Изобразить фрагмент схемы алгоритма, соответствующий следующему фрагменту программы:

if (c < 3) if(c == 2) a + + ; else b+ + ; if(c < 2) C+ + ;
else a += 1; { C++; b++; }

Вариант 13. Изобразить фрагмент схемы алгоритма, соответствующий следующему фрагменту программы:

z := x+5 при i=1,3,5;
z := a+b при i=2,4,6;

z := x в остальных случаях

Вариант 14. Записать фрагмент программы, соответствующие следующему фрагменту схемы программы (рис. 103):

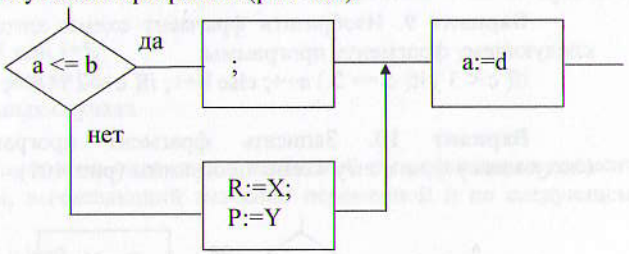


Рис. 103. Фрагмент схемы программы

Вариант 15. Изобразить фрагмент схемы алгоритма, соответствующий следующему фрагменту программы:

if(c <= 1) a++; else if(c == 5) a--; else a*=2;

Вариант 16. Записать фрагмент программы, соответствующий следующему фрагменту схемы программы (рис. 104):

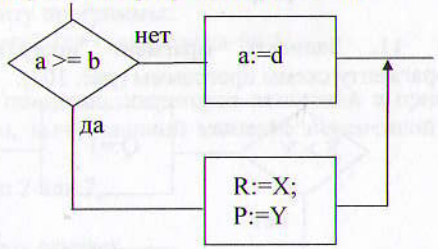


Рис. 104. Фрагмент схемы программы

Вариант 17. С помощью операторов ветвлений и присваивания записать фрагмент программы, вычисляющий величину i (i >= 0) по следующему правилу:

n= n+1 при i=0,
n = a+b при 1=2,4,6,8,10 и т.д.
n = a-b в остальных случаях

Вариант 18. Изобразить фрагмент схемы алгоритма, соответствующий следующему фрагменту программы:

if(c < 5) if(c == 1) a++; b--; a --1;

Вариант 19. С помощью операторов ветвлений и присваивания записать фрагмент программы, вычисляющий значение переменной z по следующему правилу:

z = x+5 при a>2 и b=0,
z = [a+b при a<=2,
z = x в остальных случаях

Вариант 20. Записать фрагмент программы, соответствующий следующему фрагменту схемы программы (рис. 105):

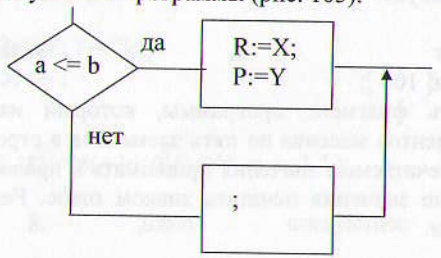


Рис. 105. Фрагмент схемы программы

5. Циклы. Варианты тестов

Вариант 1. Задан массив:
float a[34];

Написать фрагмент программы, который напечатает с новой строки значения элементов массива по пять элементов в строке и по пятнадцать позиций на элемент. Печатаемые значения прижимать к левой границе поля вывода, а положительные значения печатать со знаком плюс. Решить задачу с помощью цикла ***for***.

Вариант 2. Задан массив:
double a[104];

Написать фрагмент программы, который напечатает с нон строки значения элементов массива по пять элементов в строке и пятнадцать позиций на элемент. Печатаемые значения прижимать к правой границе поля вывода, а положительные значения печатать знаком плюс. Решить задачу с помощью Цикла *do-white*.

Вариант 3. Задан массив:
double a[43];

Написать фрагмент программы, который напечатает с HOI строки значения элементов массива по четыре элемента в строю по девятнадцать позиций на элемент. Печатаемые значения принимать к левой границе поля вывода, положительные значения печатать со знаком плюс, а в дробной части печатать три цифры. Рейн задачу с помощью цикла ***while***.

Вариант 4. Задан массив:
double a[50];

Написать фрагмент программы, который напечатает с HOI строки значения элементов массива по четыре элемента в строк! по двадцать позиций на элемент. Печатаемые значения прижимать правой границе поля вывода, а в дробной части печатать 6 цифр. Решить задачу с помощью цикла **do-while.**

Вариант 5. Задан массив:
double a[50];

Написать фрагмент программы, который напечатает в уже открытый поток "Output" с новой строки значения элементов масс по четыре элемента в строке и по двадцать позиций на элемент. Печатаемые значения прижимать к правой границе поля вывода, дробной части печатать шесть цифр. Решить задачу с помощью цикла **do..while**. Использовать только один цикл.

Подключить необходимые стандартные заголовочные файл

Вариант 6. Дано следующее определение:
int к;

При каких исходных значениях к приведенный ниже цикл будет выполняться бесконечно?

```
while ( к < 5 ) к++;
```

Возможные варианты ответов: при к <=..., или при к>= ..., или таких k не существует.

Вариант 7. Пусть определены переменные
int к, n;

Укажите, что напечатает следующий фрагмент программы (ниже знак ^ обозначает пробел):

```
printf( "\n\n^%-2s^\n", "---“);
for (к = 7; к >= 5; к-- ;
{
printf( "\n\n" );
"n = 6 - к; printf( "%i^^%+3d^%5;s^^", к, n, "-" );
}
```

Вариант 8. Дано следующее определение:
int к;

При каких исходных значениях к приведенный ниже цикл будет выполняться бесконечно?

```
while( к >= 15 ) к++;
```

Возможные варианты ответов: при к <= ..., или при к >= ..., или таких к не существует.

Вариант 9. Пусть определены переменные:
int к, n;

Укажите, что напечатает следующий фрагмент программы

```
printf( "\n\n\t^%-2s^\n","2345" );
for( к = 5; к > 5; к++ )
{
printf( "\n\n" );
n = 6 — к; printf( “%i^^%4d^%2s^^”, к, n, "-" );
}
```

Вариант 10. Дано следующее определение:
int k;

При каких исходных значениях к приведенный ниже цикл будет выполняться бесконечно?

```
do
{к++; } while ( к > 10 );
```

Возможные варианты ответов: при k<= <..., или при к >=..., или таких к не существует.

Вариант 11. Пусть определены переменные:
int к, n;

Укажите, что напечатает следующий фрагмент программы

```
printf( "\n%-3.2s\n", “*****”);
for( к = 5; к > 5; к-- )
{
n = 6 = к; printf( “%i^^%04d^%2s^^", к, n, "-" ); )
```

Вариант 12* Дано следующее определение:

```
int к;
```

При каких исходных значениях к приведенный ниже цикл будет выполняться бесконечно?

```
do
(
к++; } while ( к > -5 );
```

Возможные варианты ответов: при к <= ..., или при к >= ..., или таких к не существует.

Вариант 13. Пусть определены переменные;

```
int к, n;
```

Укажите, что напечатает следующий фрагмент программы

```
printf( "\n\n^%-5s^\n", "--" );
for( к = 5; к >= 5; к-- ) {
printf( "\n\n" );
n = 6 -к; printf("%i^^%4d^%2s^^", к, n, “-“);
```

Вариант 14. Дано следующее определение:

```
int к;
```

При каких исходных значениях к приведенный ниже цикл будет выполняться бесконечно?

```
while ( к < 12 ) к++;
```

Возможные варианты ответов: при к <= ..,, или при k>=..., или таких к не существует.

Вариант 15. Пусть определены переменные:

```
int к, n;
```

Укажите, что напечатает следующий фрагмент программы

```
printf( "\n%3s\n", "-" );
for( к = 5; к > 5; к-- )
{
n = 6 - к; printf( "%i^^4d^%2s^^”, к, n, "-" ) ;
}
```

Вариант 16. Сколько раз будет выполнено -тело приведенного ниже цикла?

```
for(int к=4; к<17; к+=3 ) ;
```

Возможные варианты ответов: тело цикла будет выполнено ... раз или цикл будет выполняться бесконечно.

Вариант 17. Пусть определены переменные:

```
int к; n;
```

Укажите, что напечатает следующий фрагмент программы

```
printf( "\n%3s\n", "-12345" )/ for( к = 5; к >= 1; к-- )
{
n = 8 - к; printf( "%i^^%4d^%2s^^", к, n, "-12" );
}
```

Вариант 18. Сколько раз будет выполнено тело приведенное ниже цикла?

```
int c = 3;
for( int k=4; к<17; к+= 3, c+=2 );
```

Какое значение будет иметь с после выхода из цикла?

Вариант 19. Пусть определены переменные:

```
int k;
```

Укажите, что напечатает следующий фрагмент программы:

```
printf("\n^%-5s%s^\n", “*”,”+” ) ;
for( к = 1; к >= -3; к-- )
printf( "^%5d^%3s^", к, "--" );
```

Вариант 20. Пусть определены переменные:

```
int к, n;
```

Укажите, что напечатает следующий фрагмент программы

```
printf( "\n%6s\n", "-" );
for( к = 5; к >= 1; к-- )
{
n=6- к; printf( "%i^^%4d^%2s^”, к, n, “***” );
}
```

6. Структуры. Варианты тестов

В ответах на приведенные ниже варианты тестов необходимо выполнить следующее.

Закрыть открытые файлы, как только они станут не нужны.

Предусмотреть контроль корректности значения, возвращаемых функциями библиотеки Си "fopen", "fscanf[4]. Указать, как и включаемые файлы требует представленный фрагмент.

Вариант 1. В файле операционной системы "Task4.in" хранится в текстовой форме ведомость сдачи экзаменов студентами некоторой группы. Каждая строка этого файла содержит сведения об одном студенте, представленные в следующем формате:

позиции 1..,2 порядковый номер студента в группе; позиция 3 = пробельная литера; позиции 4...22 = фамилия студента длиной не более 18 символов, в произвольном месте поля; позиция 23 = пробельная литера; позиция 24.- четыре оценки по четырем предметам, разделенные не менее чем одной пробельной литерой. Количество студентов в- группе равно 16. Пример строк указанного файла:

```
01          Андреев      5 4 5 5
02 Быков .      5 5 5 5
16 Яковлев              4 4 5 4
```

Написать: 1) определение массива структур для хранения указанной ведомости; 2) фрагмент программы, который заполнит экзаменационную ведомость данными, вводимыми из файла операционной системы “f_in" (ввод данных должен осуществляться, в текстовом режиме; 3) фрагмент программы, который вычисляет среднюю экзаменационную оценку по всем предметам и студентам (т.е. среднюю оценку из 64 оценок), а затем выводит значение этого показателя в файл операционной системы "f_out".

Вариант 2. В файле операционной системы "f_in" имеется 10 строк, каждая из которых содержит длины сторон прямоугольников (значения длин задаются в формате с плавающей точкой и разделены пробелами).

Написать: 1) определение массива структур для хранения указанных длин сторон прямоугольников, их площадей и периметров; 2) фрагмент программы для чтения длин сторон прямоугольников из файла операционной системы "/.in"; 3) фрагмент программы, вычисляющий и печатающий площади и периметры прямоугольников в файл операционной системы "f_out".

Вариант 3. Имеется следующий фрагмент программы:

```
struct ExamReport// Строка экз. ведомости
{
// Фамилия студента
char      Name[ 15 ];
unsigned Mark;    // Экзаменационная оценка
}
```



```
//      MATHematics;      ведомость      по      математике
ExamReport   Math[ 16 ];
```

Написать фрагмент программы, который заполнит экзаменационную ведомость "Math" данными, вводимыми из файла операционной системы "Task4.in". Ввод данных должен осуществляться в текстовом режиме. В каждой строке файла "Task4.in" содержатся следующие поля данных: фамилия студента длиной не более : символов,' начинающаяся с позиции 1; экзаменационная оценка позиции 16). Между последней литерой фамилии и оценкой распложены пробельные литеры.

Вариант 4. В файле операционной системы "Task4.in" хранится в текстовой форме ведомость сдачи экзаменов студентами некоторой группы. Каждая строка этого файла содержит сведения об одном студенте, представленные в следующем формате: позиция 1.-. порядковый номер студента в группе; позиция 3 = пробельная литера; позиции 4... 15 - фамилия студента длиной не более 11 символов в произвольном месте поля; позиция 16 - пробельная литера; позиция 17 - три оценки по трем предметам, разделенные не менее чем одной пробельной литерой. Количество студентов в группе равно 16. Пример строк указанного файла:

```
1  Андреев      5 4 5
2  Быков        5 5 5
        16 Яковлев              4 5 4
```

Написать: 1) определение массива структур для хранения указанной ведомости, причем в связи с каждым студентом необходимо хранить только фамилию и три оценки, а порядковый номер студента должен быть представлен неявно, индексом элемента массива структур; 2) фрагмент программы, который заполнит экзаменационную ведомость данными, вводимыми из файла операционной сие! мы "Task4.in" (ввод данных должен осуществляться в текстовом р жиме); 3) фрагмент программы, который вычисляет среднюю экзаменационную оценку по всем предметам и студентам (т.е. среднюю оценку из 48 оценок), а затем выводит значение этого показателя файл операционной системы "Task4.out".

Замечание. Очевидно, каждая строка исходных данных содержит лишние сведения: порядковый номер студента в группе (в начале строки). При вводе эти номера следует игнорировать (каким-либо способом).

Вариант 5. Имеется следующий фрагмент программы:

```
struct ExamReport// Строка экз. ведомости
{
// Фамилия студента
char      Name[ 15 ];
unsigned Mark;    // Экзаменационная оценка
//      MATHematics:      экзаменационная      ведомость      по      математике
ExamReport   Math[ 16 ];
```

В каждой строке файла "Task4.in" содержатся следующие поля данных: фамилия студента длиной не более 13 символов, начинающаяся с позиции 1; экзаменационная оценка (в позиции 16). Между последней литерой фамилии и оценкой расположены пробельные литеры.

Написать фрагмент программы, который заполнит экзаменационную ведомость "Math" данными, вводимыми из файла операционной системы "Task4.in" (ввод данных должен осуществляться в текстовом режиме).

Вариант 6. В файле операционной системы "Test6.in" имеется пять строк, каждая из которых содержит длины сторон прямоугольников (значения длин разделены двумя пробелами).

Написать: 1) определение массива структур для хранения указанных длин сторон прямоугольников, их площадей и периметров; 2) фрагмент программы для чтения длин сторон прямоугольников из файла операционной системы "Test6.in"; 3) фрагмент программы, вычисляющий и печатающий площади и периметры прямоугольников в файл операционной системы "Test6.out".

Вариант 7. Имеется следующий фрагмент программы:

```
struct ExamReport// Строка экз. ведомости
{// Фамилия студента
char Name[ 15];
unsigned Mark1;// Экзаменационная оценка 1
unsigned Mark2;// Экзаменационная оценка 2
} Exam[ 16];
```

В каждой строке файла "Task4.in" содержатся следующие поля данных: фамилия студента длиной не более 13 символов, начинающаяся с позиции 1; экзаменационная оценка (в позиции 16); пробел (в позиции 17); экзаменационная оценка (в позиции 18). Между последней литерой фамилии и первой оценкой расположены пробельные литеры.

Написать фрагмент программы, который заполнит экзаменационную ведомость "Exam" данными, вводимыми из файла операционной системы "Task4.in" (ввод данных должен осуществляться в текстовом режиме). ,

Вариант 8. Имеется следующий фрагмент программы:

```
struct EXAM_REPORT // Строка экзаменационной ведомости
{
char    fam[ 21 ];// Фамилия экзаменуемого
int mark;           // Экзаменационная оценка
}       math( 16 ];
// Абсолютная успеваемость (процент студентов с положительными оценками)
float ku;
// Качественная успеваемость ( процент студентов, получивших "4" и "5" )
float   ku;
```


В каждой строке этого файла содержится фамилия студент длиной не более 19 символов, начинающаяся с позиции 1, и экзаменационная оценка (поз. 22). Между фамилией и оценкой расположены "пробелы".

Написать: 1) фрагмент программы для чтения экзаменационной ведомости из текстового файла "f.in"; 2) фрагмент программы вычисляющей и печатающей в файл "f.out" абсолютную и качественную успеваемость группы по математике.

Вариант 9. В файле операционной системы "Task4.in" хранится в текстовой форме ведомость со сведениями о продуктах. Каждая строка этого файла содержит сведения об одном виде продукта представленные в следующем формате: позиции 1...2 - порядковый номер продукта; позиция 3 - пробельная литера; позиции 4..15 - название продукта длиной не более 11 символов, в произвольном месте поля; позиция 16 - пробельная литера; позиции 17...19 - содержание белка в 100 граммах продукта (целое).

Количество продуктов в ведомости равно 16. Пример строк указанного файла:

```
1 минтай     20
2 шука       21
  16 сметана     15
```

Написать: 1) определение массива структур для хранения указанной ведомости и фрагмент программы, который заполнит ведомость данными, вводимыми из файла операционной системы "Task4.in" (ввод данных должен осуществляться в текстовом режиме); 2) фрагмент программы для нахождения и, печати (в файл "Task4.out") информацию о продукте с наибольшем содержание белка.

Вариант 10. В текстовом файле "Task4.in" содержится список книг библиотеки, имеющий следующий вид:

```
01 Иванов  Программирование 20.500
354
15         Петров          Архиваторы   7.200
```

Каждая строка списка содержит сведения об одной книге: первые две позиции - порядковый номер книги, третья позиция - "пробел", с поз. 4 начинается фамилия автора длиной не более 13 символов, поз. 18...34 - название книги (из одного слова), поз. 35 - "пробел", с поз. 36 - стоимость книги.

Написать: 1) определение массива структур для хранения указанного списка и фрагмент программы для чтения списка из файла "Task4.in"; 2) фрагмент программы, вычисляющей и печатающей среднюю стоимость книг в библиотеке в файл "Task4.out".

Вариант 11. В текстовом файле "Task4.in" содержится информация о квартире, имеющая следующий вид:

```
01 Комната 15
```

05 Кухня 5

Каждая строка содержит сведения об одной комнате: первые две позиции = порядковый номер комнаты, третья позиция = "пробел", с поз. 4 начинается название комнаты длиной не более 15 символов, с поз. 21 = метраж комнаты.

Написать: 1) определение массива структур для хранения указанных данных и фрагмент программы для чтения данных о квартире из файла "Task4.in"; 2) фрагмент программы для нахождения и печати общего метража данной квартиры в файл "Task4.out".

Вариант 12. В текстовом файле "Task4.in" содержится ведомость сдачи экзаменов студентами некоторой группы, имеющая следующий вид:

01 Андреев. 5 4 5
16 Петров 4 5 4

Каждая строка ведомости содержит сведения об одном студенте: первые две позиции = порядковый номер студента, третья позиция = "пробел", с поз. 4 начинается фамилия студента длиной не более 11 символов, поз. 16...20 = оценки по трем предметам. Каждой оценке предшествует пробел, а первой оценке может предшествовать и большее число "пробелов".

Написать: 1) определение массива структур для хранения указанной ведомости и фрагмент программы для чтения ведомости из файла "Task4.in"; 2) фрагмент программы для нахождения и печати списка должников (.студентов, имеющих хотя бы одну двойку ; в файл "Task4.out".

Вариант 13. В текстовом файле "Task4.in" содержится ведомость сдачи экзаменов студентами некоторой группы; имеющая следующий вид:

01 Андреев 5 4 5
16 Петров 4 5 4

Каждая строка ведомости содержит сведения об одном студенте: первые две позиции = порядковый номер студента, третья позиция - "пробел", с поз. 4 начинается фамилия студента длиной не более 11 символов, поз. 16...20 -оценки по трем предметам (математике, программированию и физике). Каждой оценке предшествует пробел, а первой оценке может предшествовать и большее числе "пробелов".

Написать: 1) определение массива структур для хранения указанной ведомости и фрагмент программы для чтения ведомости и: файла "Task4.in"; 2) фрагмент программы для нахождения и печати списка должников по программированию в файл "Task4.out".

Вариант 14. В текстовом файле "Taisk4.in" имеется ведомости сдачи экзаменов студентами некоторой группы:

01 Андреев 5 4 5
16 Петров 4 5 4

Каждая строка ведомости содержит сведения об одном студенте: первые две позиции = порядковый номер студента, третья позиция = "пробел", с поз. 4 начинается фамилия студента длиной не более 10 символов, поз. 16...20,—

оценки по трем предметам (математике, программированию и физике). Каждой оценке предшествует пробел, а первой оценке может предшествовать и большее число "пробелов".

Написать: 1) определение массива структур для хранения указанной ведомости и фрагмент программы для чтения ведомости и файла "Task4.in"; 2) фрагмент программы для нахождения и печати списка студентов, сдавших физику на "отлично" в файл "Task4.out".

Вариант 15. В текстовом файле "Task4.in" содержатся сведения о предприятиях сферы обслуживания районов города, имеющих следующий вид:

I. Калининский 10 20 7
10. Выборгский 15 10 9

Каждая строка содержит сведения об одном районе: первые две или три позиции -номер района (с точкой), далее следует один или два "пробела", поз. 5..19 = название района длиной не более 14 символов, далее следуют три целых числа, каждому из которых предшествуют один или более пробелов. Первое число задает количество аптек, второе — универсамов, а третье = химчисток.

Написать: 1) определение массива структур для хранения указанной информации и фрагмент программы для чтения данных из файла "Task4.in"; 2) фрагмент программы для нахождения и печати в файл "Task4.out" названия района (или районов), в котором (в которых) находится больше всего аптек.

Вариант 16. В текстовом файле "Task4.in" содержится информация о квартире, имеющая следующий вид:

01 Комната 15
05 Кухня 5

Каждая строка содержит сведения об одной комнате: первые две позиции = порядковый номер комнаты, третья позиция = "пробел", с поз. 4 начинается название комнаты длиной не более 15 символов, с поз. 21 = метраж комнаты.

Написать: 1) определение массива структур для хранения указанных данных и фрагмент программы для чтения данных о квартире из файла "Task4.in"; 2) фрагмент программы для нахождения и печати в файл "Task4.out" метража самой большой по площади комнаты в квартире.

Вариант 17. В текстовом файле "Task4.in" содержится список книг библиотеки, имеющий следующий вид:

01 Иванов Программирование 20.500
15 Петров Архиваторы 7.200

Каждая строка списка содержит сведения об одной книге: первые две позиции = порядковый номер книги, третья позиция = "пробел", с поз. 4 начинается фамилия автора длиной не более 12 символов, поз. 18...34 = название книги (из одного слова), поз. 35 = "пробел", с поз. 36 = стоимость книги.

Написать: 1) определение массива структур для хранения указанного списка и фрагмент программы для чтения списка из файла

*Task4.in"; 2) фрагмент программы, вычисляющей и печатающий полные данные (номер, автор, название и цена) самой дорогой КНІ в библиотеке в файл "Task4.out".

Вариант 18. В текстовом файле "Task4.in" содержится ведомость сдачи экзаменов студентами некоторой группы, имеющая с дующий вид:

01 Андреев 5 4 5

,16. Петров 4 5.4

Каждая строка ведомости содержит сведения об одном студенте: первые две позиции и порядковый номер студента, третья позиция = "пробел", с поз. 4 начинается фамилия студента длиной не лее 11 символов, поз. 16..20 = оценки по трем предметам. Каждой оценке предшествует пробел, а первой оценке может предшествовать и большее число "пробелов".

Написать: 1) определение массива структур для хранения указанной ведомости и фрагмент программы для чтения ведомости файла "Task4.in"; 2) фрагмент Программы для нахождения и печати списка студентов-троечников (сдавших экзамены на одни тройки файл "Task4.out".

Вариант 19. В файле операционной системы "f.in" имеется строк, каждая из которых содержит длины сторон прямоугольника (значения длин задаются в формате с плавающей точкой и разделены пробелами).

Написать: 1) определение массива структур для хранения указанных длин сторон прямоугольников, их площадей и периметр 2) фрагмент программы для чтения длин Сторон прямоугольников файла операционной системы "//и"; 3) фрагмент программы, вычисляющий и печатающий длины сторон прямоугольников, имеющих максимальные периметр и площадь в файл операционной системе "f.out".

Вариант 20. В файле операционной системы "Task4.in" хранится в текстовой форме ведомость со сведениями о продуктах. Каждая строка этого файла содержит сведения об одном виде продукта, представленные в следующем формате: позиции 1...2 = порядковый номер продукта; позиция 3 = пробельная литера; позиции 4 = название продукта длиной не более 11 символов, в произвола месте поля; позиция 16 = пробельная литера; позиции 17:,. 19 содержание белка в 100 граммах продукта (целое); Позиция 20 - пробельная литера; позиции 21...23 = калорийность 100 грамм продукта (целое). Количество продуктов в ведомости равно 16. Пример строк указанного файла:

1 минтай 20 100

2 щука 21 120

16 сметана 15 150

Написать: 1) определение массива структур для хранения указанной ведомости и фрагмент программы, который заполнит экзаменационную ведомость данными, вводимыми из файла операционной системы "Task4.in" (ввод данных должен осуществляться в текстовом режиме); 2) фрагмент

программы для нахождения и печати (в файл "Task4.out") названия продукта (продуктов) с наибольшей калорийностью.

П.1.1.7. Функции. Варианты тестов

В ответах на приведенные ниже варианты тестов выполнить следующее.

Для решения указанных в вариантах тестов задач написать прототип, определение функции и пример ее вызова.

Для передачи в функцию исходных данных и получения из нее ответов использовать список параметров. Единственный ответ лучше получать из функции как возвращаемое значение

Вариант 1. Вычислить max :=наиб{a,b,c}. Исходные данные имеют тип с плавающей точкой.

Вариант 2. В массиве целого типа определить количество положительных, отрицательных и нулевых элементов.

Вариант 3. Вычислить max := наиб{a,b} и min:=наим{a,b}.

Вариант 4. Подсчитать в одномерном массиве целого типа размером 100 элементов наименьшее значение среди положительных элементов.

Вариант 5. Подсчитать в одномерном массиве целого типа размером 100 элементов среднее арифметическое значение. Постарайтесь не потерять в ответе дробную часть.

Вариант 6. Подсчитать в одномерном массиве целого типа размером 100 элементов индекс и значение последнего из положительных элементов.

Вариант 7. Подсчитать в одномерном массиве целого тип размером 100 элементов количество нулевых значений.

Вариант 8. Сформировать одномерный массив с элементами

z[i] (0 <= i < N), N=20 из двух заданных массивов целого типа x[i], y[i] по правилу:

z[i] := min{ x[i], y[i] }, i = 0, 1, . . . , N-l

Вариант 9. Вычислить сумму квадратов элементов двух одномерных массивов вещественного типа размером по 40 элементов и получить ее из функций как возвращаемое значение

39

Сумма { x[i] * x[i] + y[i] * y[i])

i=0

Вариант 10. Найти индекс максимального элемента в массиве целого типа из 30 элементов. Результат получить из функции как возвращаемое значение.

Вариант 11. Написать функцию с двумя параметрами логического типа, возвращающую значение в соответствии со следующей таблицей истинности:

Параметры		Возвратаемый
Первый	Второй	результат
false	false	false
false	true	false
true	false	true
true	true	false

Параметры и результат - целого типа: false соответствует нулевому и true - ненулевому значениям.

Вариант 12. Получить одномерный массив из двух заданных массивов вещественного типа x, y по правилу:

z[i] := (x[i] + y[i]) / 2, i = 0, 1, ..., 29.

Вариант 13. Найти величину и номер первого отрицательного и последнего положительного элементов в массиве вещественного типа заданного размера.

Вариант 14. Поменять местами первый и последний элемент, второй и предпоследний и т.д. в одномерном массиве вещественного типа заданного размера.

Вариант 15. Вычислить среднее арифметическое для положительных чисел в одномерном массиве целого типа заданного размера. Постарайтесь не потерять дробную часть результата и избежать возможного деления на нуль.

Вариант 16. Написать функцию нахождения минимального элемента среди отрицательных и максимального элемента среди положительных в одномерном массиве целого типа заданного размера.

Вариант 17. Написать функцию нахождения максимального положительного числа кратного пяти в одномерном массиве целого типа заданного размера.

Вариант 18. Найти количество нулевых элементов в одномерном массиве целого типа заданного размера и сформировать новый массив из ненулевых элементов исходного массива.

Вариант 19. В одномерном массиве вещественного типа заданного размера найти сумму элементов, расположенных между максимальным и минимальным элементами. Указанный результат получить из функции как возвращаемое значение.

Вариант 20. Сжать одномерный массив вещественного типа заданного размера. С этой целью удалить из массива все элементы, абсолютное значение которых меньше единицы. Освободившиеся в конце массива элементы заполнить нулями.

П.1.1.8. **Области действия определений.** Варианты тестов

В ответах на приведенные ниже варианты тестов укажите, как будут выглядеть строки, выведенные на экран в результате выполнения программы, приведенной в соответствующем варианте. В ответе укажите также местоположение пробелов.

Вариант 1. Что напечатает следующая программа?

```
#include <stdio.h>
int i = 0, j = 2;
int main( void )
{
auto int i = 0;
361
```