Федеральное государственное образовательное бюджетное
учреждение высшего образования
«Финансовый университет при Правительстве Российской Федерации»
(Финансовый университет)
Махачкалинский филиал Финуниверситета

ФОНД ОЦЕНОЧНЫХ СРЕДСТВ

по учебной дисциплине

ОП.04 ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ

09.02.07 Информационные системы и программирование

Махачкала
2023

ОДОБРЕН Предметной (цикловой) комиссией информационных систем и программирования	Разработан на основе Федерального государственного образовательного стандарта по специальности среднего профессионального образования 09.02.07 Информационные системы и программирование
Протокол № 1 от «29» марта 2023 г.	
Председатель предметной (цикловой) комиссии _____ / Расулова П.Г. *(подпись) Ф.И.О.*	Заместитель директора по учебно-методической работе _____ / Легашова О.Н. *(подпись)*

Составитель: Далгатова Якут Абдулмуслимовна, Заслуженный учитель РД, преподаватель ВКК, Махачкалинский филиал Финуниверситета.

I. ПАСПОРТ
ФОНДА ОЦЕНОЧНЫХ СРЕДСТВ
по учебной дисциплине
ОП.04 ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ
09.02.07 Информационные системы и программирование

Результаты обучения (усвоенные знания, освоенные умения)	ПК, ОК	Наименование темы	Наименование оценочного средства	
			Текущий контроль	Промежуточная аттестация
Освоенные знания: – эволюцию языков программирования, их классификацию, понятие системы программирования; – использовать программы для графического отображения алгоритмов; – определять сложность работы алгоритмов. **Освоенные умения:** – разрабатывать алгоритмы для конкретных задач.	ОК 01, ОК 02, ОК 04, ОК 05, ОК 08, ОК 09, ПК 1.1, ПК 1.2, ПК 1.3, ПК 1.4, ПК 1.5, ПК 2.4, ПК 2.5.	**РАЗДЕЛ 1. Введение в программирование.** **Тема1.1.** Языки Программирования.	– Компьютерное тематическое тестирование – Устный и письменный опрос – Оценка выполнения практических заданий – Дискуссия, обсуждение ситуационных заданий	Вопросы для проведения экзамена
Освоенные знания: – основные элементы языка, структуру программы, операторы и операции, управляющие структуры, структуры данных, файлы, классы памяти **Освоенные умения:** – работать в среде программирования; – реализовывать построенные алгоритмы в виде программ на конкретном языке программирования.	ОК 01, ОК 02, ОК 04, ОК 05, ОК 08, ОК 09, ПК 1.1, ПК 1.2, ПК 1.3, ПК 1.4, ПК 1.5, ПК 2.4, ПК 2.5.	**Тема1.2.** Типы данных.	– Компьютерное тематическое тестирование – Устный и письменный опрос – Оценка выполнения практических заданий – Дискуссия, обсуждение ситуационных заданий	
Освоенные знания: – понятие алгоритмизации, свойства алгоритмов, общие принципы построения алгоритмов, основные алгоритмические конструкции;	ОК 01, ОК 02, ОК 04, ОК 05, ОК 08, ОК 09,	**РАЗДЕЛ 2. Основы языка программирования Python.**	– Компьютерное тематическое тестирование – Устный и письменный опрос – Оценка	

– основные элементы языка, структуру программы, операторы и операции, управляющие структуры, структуры данных, файлы, классы памяти **Освоенные умения:** – разрабатывать алгоритмы для конкретных задач; – использовать программы для графического отображения алгоритмов; – определять сложность работы алгоритмов.	ПК 1.1, ПК 1.2, ПК 1.3, ПК 1.4, ПК 1.5, ПК 2.4, ПК 2.5.	**Тема 2.1**. Операторы языка программирования	выполнения практических заданий – Дискуссия, обсуждение ситуационных заданий Подготовка и выступление с сообщением (докладом, рефератом)	
Освоенные знания: – понятие алгоритмизации, свойства алгоритмов, общие принципы построения алгоритмов, основные алгоритмические конструкции; – подпрограммы, составление библиотек подпрограмм объектно-ориентированную модель программирования **Освоенные умения:** – разрабатывать алгоритмы для конкретных задач; – работать в среде программирования; – реализовывать построенные алгоритмы в виде программ на конкретном языке программирования.	ОК 01, ОК 02, ОК 04, ОК 05, ОК 08, ОК 09, ПК 1.1, ПК 1.2, ПК 1.3, ПК 1.4, ПК 1.5, ПК 2.4, ПК 2.5.	**РАЗДЕЛ 3. Подпрограммы С++. Понятие модуля** **Тема 3.1.** Процедуры и функции.	– Компьютерное тематическое тестирование – Устный и письменный опрос – Оценка выполнения практических заданий – Дискуссия, обсуждение ситуационных заданий	
Освоенные знания: – понятие алгоритмизации, свойства алгоритмов, общие принципы построения алгоритмов, основные алгоритмические конструкции; – эволюцию языков программирования, их классификацию, понятие системы программирования **Освоенные умения:** – работать в среде программирования;	ОК 01, ОК 02, ОК 04, ОК 05, ОК 08, ОК 09, ПК 1.1, ПК 1.2, ПК 1.3, ПК 1.4, ПК 1.5, ПК 2.4, ПК 2.5.	**Тема 3.2.** Структуризация в программировании.	– Компьютерное тематическое тестирование – Устный и письменный опрос – Оценка выполнения практических заданий – Дискуссия, обсуждение ситуационных заданий	

– оформлять код программы в соответствии со стандартом кодирования; выполнять проверку, отладку кода программы.				
Освоенные знания*:* – понятие алгоритмизации, свойства алгоритмов, общие принципы построения алгоритмов, основные алгоритмические конструкции; – эволюцию языков программирования, их классификацию, понятие системы программирования **Освоенные умения:** – разрабатывать алгоритмы для конкретных задач; – использовать программы для графического отображения алгоритмов; – определять сложность работы алгоритмов.	ОК 01, ОК 02, ОК 04, ОК 05, ОК 08, ОК 09, ПК 1.1, ПК 1.2, ПК 1.3, ПК 1.4, ПК 1.5, ПК 2.4, ПК 2.5.	**Тема 3.3.** Модульное программирование.	– Устный и письменный опрос – Оценка выполнения практических заданий – Дискуссия, обсуждение ситуационных заданий	
Освоенные знания: – основные элементы языка, структуру программы, операторы и операции, управляющие структуры, структуры данных, файлы, классы памяти; – подпрограммы, составление библиотек подпрограмм. **Освоенные умения:** – работать в среде программирования; – реализовывать построенные алгоритмы в виде программ на конкретном языке программирования.	ОК 01, ОК 02, ОК 04, ОК 05, ОК 08, ОК 09, ПК 1.1, ПК 1.2, ПК 1.3, ПК 1.4, ПК 1.5, ПК 2.4, ПК 2.5.	**Раздел 4. Динамические данные.** **Тема 4.1.**Указатели.	– Компьютерное тематическое тестирование – Устный и письменный опрос – Оценка выполнения практических заданий – Дискуссия, обсуждение ситуационных заданий	
Освоенные знания: – эволюцию языков программирования, их классификацию, понятие системы программирования; – основные элементы языка, структуру программы, операторы	ОК 01, ОК 02, ОК 04, ОК 05, ОК 08, ОК 09, ПК 1.1,	**Раздел 5. Объектно-ориентированное программирование.** **Тема 5.1.**	– Компьютерное тематическое тестирование – Устный и письменный опрос – Оценка выполнения	

и операции, управляющие структуры, структуры данных, файлы, классы памяти **Освоенные умения:** – работать в среде программирования; – оформлять код программы в соответствии со стандартом кодирования; выполнять проверку, отладку кода программы.	ПК 1.2, ПК 1.3, ПК 1.4, ПК 1.5, ПК 2.4, ПК 2.5.	Основные принципы объектно-ориентированного программирования (ООП)	практических заданий – Дискуссия, обсуждение ситуационных заданий	
Освоенные знания: – подпрограммы, составление библиотек подпрограмм объектно-ориентированную модель программирования, основные принципы объектно-ориентированного программирования на примере алгоритмического языка: понятие классов и объектов, их свойств и методов, инкапсуляция и полиморфизма, наследования и переопределения **Освоенные умения:** – работать в среде программирования; реализовывать построенные алгоритмы в виде программ на конкретном языке программирования.	ОК 01, ОК 02, ОК 04, ОК 05, ОК 08, ОК 09, ПК 1.1, ПК 1.2, ПК 1.3, ПК 1.4, ПК 1.5, ПК 2.4, ПК 2.5.	**Тема 5.2.** Интегрированная среда разработчика.	– Устный и письменный опрос – Оценка выполнения практических заданий – Дискуссия, обсуждение ситуационных заданий	
Освоенные знания: – эволюцию языков программирования, их классификацию, понятие системы программирования **Освоенные умения:** – использовать программы для графического отображения алгоритмов; – работать в среде программирования.	ОК 01, ОК 02, ОК 04, ОК 05, ОК 08, ОК 09, ПК 1.1, ПК 1.2, ПК 1.3, ПК 1.4, ПК 1.5, ПК 2.4, ПК 2.5.	**Тема 5.3.** Визуальное событийно-управляемое программирование.	– Устный и письменный опрос – Оценка выполнения практических заданий – Дискуссия, обсуждение ситуационных заданий	
Освоенные знания:	ОК 01, ОК 02,	**Тема 5.4.** Разработка	– Компьютерное тематическое	

– эволюцию языков программирования, их классификацию, понятие системы программирования. **Освоенные умения:** – работать в среде программирования.	ОК 04, ОК 05, ОК 08, ОК 09, ПК 1.1, ПК 1.2, ПК 1.3, ПК 1.4, ПК 1.5, ПК 2.4, ПК 2.5.	оконного приложения	тестирование – Устный и письменный опрос – Оценка выполнения практических заданий – Дискуссия, обсуждение ситуационных заданий	
Освоенные знания: – основные элементы языка, структуру программы, операторы и операции, управляющие структуры, структуры данных, файлы, классы памяти **Освоенные умения:** – разрабатывать алгоритмы для конкретных задач; – оформлять код программы в соответствии со стандартом кодирования; выполнять проверку, отладку кода программы.	ОК 01, ОК 02, ОК 04, ОК 05, ОК 08, ОК 09, ПК 1.1, ПК 1.2, ПК 1.3, ПК 1.4, ПК 1.5, ПК 2.4, ПК 2.5.	**Тема 5.5.** Этапы разработки приложений	– Компьютерное тематическое тестирование – Устный и письменный опрос – Оценка выполнения практических заданий – Дискуссия, обсуждение ситуационных заданий Подготовка и выступление с сообщением (докладом, рефератом)	
Освоенные знания: – эволюцию языков программирования, их классификацию, понятие системы программирования; – основные элементы языка, структуру программы, операторы и операции, управляющие структуры, структуры данных, файлы, классы памяти **Освоенные умения:** – работать в среде программирования; – реализовывать построенные алгоритмы в виде программ на	ОК 01, ОК 02, ОК 04, ОК 05, ОК 08, ОК 09, ПК 1.1, ПК 1.2, ПК 1.3, ПК 1.4, ПК 1.5, ПК 2.4, ПК 2.5.	**Тема 5.6.** Иерархия классов.	– Устный и письменный опрос – Оценка выполнения практических заданий – Дискуссия, обсуждение ситуационных заданий	

конкретном языке программирования; – оформлять код программы в соответствии со стандартом кодирования; выполнять проверку, отладку кода программы.				

Форма промежуточной аттестации по дисциплине

Дисциплина	Формы промежуточной аттестации
Основы алгоритмизации и программирования	экзамен

II. Комплект оценочных средств.
Пакет заданий для текущего контроля знаний и умений

1. Примерные вопросы компьютерного тематического тестирования
Тема 1.1. Языки программирования

1. Алгоритм — это:

1) указание на выполнение действий

2) процесс выполнения вычислений, приводящих к решению задачи.

3) система правил, описывающая последовательность действий, которые необходимо выполнить для решения задачи

2. Свойствами алгоритма являются:

1) информативность

2) массовость

3) оперативность

4) определенность

5) дискретность

6) цикличность

7) результативность.

3. Алгоритм может быть задан следующими способами:

1) словесным

2) на алгоритмическом языке

3) графическим

4) формально-словесным

5) словесно-графическим

6) последовательностью байтов.

4. Программа — это:

1) система правил, описывающая последовательность действий, которые необходимо выполнить для решения задачи

2) указание на выполнение действий из заданного набора

3) область внешней памяти для хранения текстовых, числовых данных и другой информации

4) последовательность команд, реализующая алгоритм решения задачи.

5. Программа-интерпретатор выполняет:

1) поиск файлов на диске

2) пооператорное выполнение программы

3) полное выполнение программы.

6. Программа-компилятор выполняет:

1) переводит исходный текст в машинный код

2) записывает машинный код в форме загрузочного файла.

3) формирует текстовый файл

7. QBASIC — это

1) алгоритмический язык, использующий команды MS-DOS

2) алгоритмический язык программирования, работающий в режиме интерпретации

3) алгоритмический язык, работающий только в среде Windows.

8. Алфавит языка QBASIC включает:

1) буквы латинского алфавита
2) буквы греческого алфавита
3) буквы русского алфавита
4) цифры+
5) знаки арифметических операций: +, -, /, «
6) знаки операций отношений: >, <, =, >=, <=, <>
7) специальные знаки:!,?, #, %,&, $,«,«,,.,,
8) круглые скобки () и) квадратные скобки.

9. В QBASIC существуют следующие типы данных:

1) числовые
2) текстовые
3) указатели
4) типы данных
5) записи.

10. Верно ли утверждение? В написании имен допускаются как строчные (маленькие)

1) , так и заглавные (большие)
2) буквы и QBASIC не делает между ними различия:
3) верно+
4) не верно.

Тема 1.2. Типы данных

1. Числовые данные могут быть представлены как:

1) целые
2) с фиксированной запятой
3) в виде строк
4) с плавающей запятой

2. Выберите правильно представленные числовые данные на QBASIC:

1) +B, -14, 21.5E2, 0.05
2) 3.4*E8, 45.E2, -16
3) 18.2, .05E1, -18
4) 0.05E5, ±16, -21,5
5) 21-Ю2, -18, 45.2

3. Запись числа в форме с плавающей точкой — это экспоненциальная форма записи:

1) верно
2) не верно.

4. Если тип данных несет текстовую информацию, то он должен быть заключен в кавычки:

1) верно
2) не верно.

5. Арифметические выражения состоят из:

1) чисел
2) констант
3) команд MS-DOS
4) машинных команд
5) переменных
6) функций

7) круглых скобок
8) квадратных скобок.

6. Переменная — это:

1) служебное слово на языке QBASIC
2) область памяти, в которой хранится некоторое значение+
3) значение регистра.

7. Имя переменной — это:

1) любая последовательность любых символов
2) последовательность латинских букв, цифр, специальных знаков (кроме пробел)
3) , которая всегда должна начинаться с латинской буквы
4) последовательность русских, латинских букв, начинающихся с латинской буквы и из специальных знаков, допускающая знак подчеркивания.

8. Для обозначения строковых переменных:

1) рядом с именем слева ставится знак $
2) рядом с именем справа ставится знак $
3) имя переменной записывается в кавычках.

9. Для обозначения целочисленных переменных:

1) рядом с именем слева ставится знак %
2) рядом с именем слева ставится знак #
3) рядом с именем справа ставится знак %.

10. Для обозначения действительных переменных с двойной точностью:

1) рядом с именем слева ставится знак #
2) рядом с именем справа ставится знак #
3) рядом с именем справа ставятся знаки ##.

Тема 2.1. Операторы языка программирования

1. Языком программирования называется:

1) совокупность средств и правил перевода текста с естественного языка на формальный.
2) совокупность средств и правил перевода текста с формального языка на естественный.
3) совокупность средств и правил представления алгоритма в виде, пригодном для выполнения вычислительной машиной.
4) язык, понятный программистам.
5) язык, понятный компьютеру.

2. Система программирования – это:

1) устройство для создания компьютерных программ.
2) специальная программа, предназначенная для создания компьютерных программ.
3) операционная система компьютера.
4) программное обеспечение компьютера, предназначенное для разработки, отладки и исполнения программ, записанных на определённом языке программирования.
5) совокупность программ на определённом языке программирования.

3. Константами называются

1) элементы данных, не имеющие значений.
2) элементы данных, обозначаемые словом cont.
3) элементы данных, значения которых в процессе выполнения программы могут изменяться или не изменяться в зависимости от условия.
4) элементы данных, значения которых в процессе выполнения программы изменяются.

5) элементы данных, значения которых в процессе выполнения программы не изменяются.

4. В языке Turbo Pascal используются константы следующих видов:

1) числовые, логические, символьные и строковые.
2) постоянные и переменные.
3) числовые и буквенные.
4) логические и булевские.
5) символьные и строковые.

5. Символьные и строковые константы – это

1) буквы и строки.
2) буквы и слова.
3) цифры и буквы.
4) цифры и строки букв.
5) отдельные символы и их последовательности.

6. Тип данных определяет…

1) множество значений, которые могут принимать объекты программы.
2) множество значений, допустимых для операций программы.
3) множество значений, которые могут принимать объекты программы, а также совокупность операций, допустимых над этими значениями.
4) совокупность операций, допустимых над этими значениями.
5) совокупность операций, допустимых над объектами программы.

7. Целочисленные типы, символьный, логический и пользовательские типы данных образуют группу

1) подпорядковых типов.
2) порядковых типов.
3) порядочных типов.
4) непорядочных типов.
5) произвольных типов.

8. Слова языка программирования Turbo Pascal подразделяются на:

1) зарезервированные слова, стандартные идентификаторы и идентификаторы пользователя.
2) зарезервированные слова и имена.
3) идентификаторы и резервные слова.
4) стандартные идентификаторы.
5) идентификаторы пользователя.

9. Метод решения задачи, записанный по определённым правилам, обеспечивающим однозначность его понимания и механического исполнения при всех значениях исходных данных (из некоторого множества значений), называется…

1) планом.
2) законом.
3) блок-схемой.
4) алгоритмом.
5) программой.

10. Алгоритмы делятся на три основных типа:

1) линейные, разветвляющиеся и цикловые.
2) линейные, разветвляющиеся и циклические.
3) линейные, ветвические и циклические.
4) прямолинейные, разветвляющиеся и циклические

5) прямолинейные, разветвляющиеся и цикловые.

Тема 3.1. Процедуры и функции

1. Какой вид программы применяют, если в результате получается несколько величин:

1) и подпрограмму-функцию и подпрограмму-процедуру
2) подпрограмму-процедуру
3) нельзя использовать подпрограмму
4) подпрограмму-функцию

2. Определите какие величины являются результатом работы подпрограммы в следующем заголовке

procedure kvadr(a,b,c:real; var x1,x2:real; var y:string);

1) y
2) x1, x2
3) a, b, c
4) x1, x2, y

3. Заголовок подпрограммы-функции начинается со слова ...

1) function
2) program
3) procedure
4) нет правильного ответа

4. Программа, предназначенная для решения какой-то конкретной частной задачи и к которой обращаются из другой программы называется

1) подпрограмма
2) нет правильного ответа
3) функция
4) модуль

5. Величины, описанные в подпрограмме называют

1) глобальные
2) фактические
3) локальные
4) формальные

6. Фактические параметры записываются

1) в команде вызова подпрограммы
2) в заголовке подпрограммы-процедуры
3) в заголовке подпрограммы-функции
4) после var

7. Заголовок подпрограммы-процедуры начинается со слова ...

1) program
2) procedure
3) function
4) нет правильного ответа

8. Определите тип значения функции в следующем заголовке

function nok(x,y:integer):real;

1) integer
2) real

9. Подпрограмма-функция возвращает в основную программу

1) все варианты правильные
2) 0 значений
3) единственное значение
4) несколько значений

10. Определите как обозначено значение функции в следующем заголовке
function nok(x,y:integer):real;

1) x
2) real
3) nok
4) y

Тема 3.2. Структуризация в программировании

1. Алгоритм это ...

• Последовательность команд, выполнение которых приводит нас к решению поставленной задачи.
• Последовательность действий, выполнив которые мы можем запустить программу.
• Задача, которую можно решить.

2. В рабочем окне системы программирования PascalABC есть:

• Строка заголовка окна
• Строка меню
• Рабочая область экрана
• Панель инструментов
• Строка прокрутки

3. Какого пункта нет в структуре программы?

• Раздел описания переменных
• Начало программы
• Конец программы
• Раздел поключения модулей (библиотек).
• Раздел пояснения программы

4. С помощью какой команды мы можем вывести на экран текст?

• write('текст')
• read('текст')
• написать('текст')
• вывести('текст')
• отобразить на экран('текст')

5. Каждое выражение(каждый оператор) в программе отделяется друг от друга ...

• точкой с запятой
• точкой
• запятой
• тире
• дефисом

6. Укажите последовательность действий выполняемых при сохранении готовой программы

• Нажать Файл
• Выбрать Сохранить Как
• Выбрать место сохранения и имя файла

• Нажать сохранить

7. Раздел var это ...

• Раздел описания переменных
• Начало программы
• Конец программы
• Раздел имя программы
• Раздел подключения библиотек

8. При выводе текста командой write он ...

• Берется о квадратные скобки
• заключается в одинарные кавычки
• заключается между двумя запятыми
• заключается между двумя точками

9. Установите соответствие. Напротив, слова на русском языке поставьте цифру, соответствующую номеру английской команды.

1.Program
2.Uses
3.Var
4.Begin
5.End

Укажите порядок следования вариантов ответа:

• Программа;
• Использовать;
• Описание;
• Начало;
• Конец;

10. Система программирования Pascal ABC относится к ...

• Системному программному обеспечению
• Прикладному программному обеспечению
• Инструментальному программному обеспечению

Тема 4.1. Указатели.

1. Один из наиболее известных языков программирования, используется для обучения программированию в старших классах и на первых курсах вузов, является базой для ряда других языков?

1) Pascal
2) Basic
3) Fortran
4) Cobol

2. Кто является автором языка программирования Паскаль?

1) Андрей Ершов
2) Блез Паскаль
3) Никлаус Вирт
Николай Паскаль

3. В какой период времени был разработан язык Паскаль?

1) 40-х - начале 50-х годов
2) 60-х -начале 70-х годов

3) 80-х - начале 90-х годов
4) 90-х - начале 95-х годов

4. В какие годы программы писались на языке машинных кодов из 0 и 1?

1) 40-ые годы
2) 50-ые годы
3) 30-ые годы
4) 80-ые годы

5. Назовите годы появления программы, называемые ассемблерами?

1) 20-е годы
2) 30-е годы
3) 40-е годы
4) 50-е годы

6. Языки близкие к процессору называются языками....

1) среднего уровня;
2) низкого уровня;
3) высокого уровня;
4) максимального уровня.

7. Работа программ рассматривается как последовательное выполнение операторов в.....

1) объектно-ориентированных;
2) логических;
3) процедурных;
4) математических.

8. работа программы рассматривается как последовательность событий и соответствующих реакций различных объектов на эти события в ....

1) объектно-ориентированных языках;
2) процедурных языках;
3) логических;
4) смешанных.

9. Работа программы рассматривается как преобразование в соответствии со строгими логическими правилами в .....

1) логических языках;
2) математических;
3) смешанных;
4) процедурных.

10. Кто был разработчиком алголо-паскалеподобный "Учебно- алгоритмического языка"?

1) Андрей Ершов
2) Никлаус Вирт
3) Сергей Синицин
4) Брайан Керниган

Тема 5.1. Основные принципы объектно-ориентированного программирования (ООП)

1. С помощью чего реализуется принцип полиморфизма в С ++?

A. наличия множественного наследования.
B. наличия виртуальных методов.
C. Использование виртуального наследования.

D. наличия абстрактных классов.

2. В программе описано класс и объект class A {public: int a, b, c; }; A * obj; Как обратиться к атрибуту c?

A. obj.c

B. obj-> c

C. obj A -> -> c

D. obj-> A.c

3. Какая из перечисленных функций не может быть конструктором?

A. void String ()

B. String ();

C. String (String & s)

D. String (const int a)

4. Отметьте правильное утверждение для абстрактного класса для языка C ++.

A. Класс, у которого все методы чисто виртуальные, называется абстрактным.

B. Абстрактный базовый класс навязывает определенный интерфейс всем производным из него классам.

C. Невозможно создать объект абстрактного класса.

D. В абстрактном классе не описываются методы вообще.

5. Если в программе на языке C ++ в производном классе переопределена операция new то …

A. все объекты этого класса и все объекты классов, выведенных из него, будут использовать эту операцию независимо от зоне видимости, в которой она переопределена.

B. производные от этого класса могут использовать глобальную операцию применив операцию базовый класс: new.

C. операцию new нельзя переопределить.

D. в любом случае эта операция будет доступна только в пределах класса-потомка.

6. Какой из перечисленных методов может быть конструктором для класса String в языке C ++?

A. String * String ();

B. void String ();

C. String (String & s);

D. const String (int a);

7. Какая функция, не будучи компонентом класса, имеет доступ к его защищенным и внутренних компонентов?

A. Шаблонная.

B. Полиморфная.

C. Дружеская.

D. Статическая.

8. Вызовет данный код ошибку компиляции? class Rectangle public: int a, b; int sum (); int square (); ~ Rect (); };

A. Ошибки нет, все записано верно.

B. Ошибка являются: имя деструктора должно совпадать с именем класса.

C. Ошибка являются: имя деструктора не может начинаться с маленькой буквы.

D. Ошибка являются: никакой идентификатор в C ++ не может начинаться со знака «~».

9. Укажите правильное объявление виртуального метода, который принимает одно целочисленное значение и возвращаетvoid.

A. virtual void SomeFunction (int x);
B. void SomeFunction (int x) virtual;
C. virtual SomeFunction (int x);
D. virtual void SomeFunction (int * x);

10. Укажите правильное использование оператора friend.

A. class A {int_friend CountPass (); private: short i;};
B. class A {public: friend int H :: CountPass (); private: short i;};
C. class A {public: int A1 :: CountPass (); friend: short i;};
D. class A {public: friend int H :: q; short i;};

Тема 5.4. Разработка оконного приложения

1. Операционная система – это:

А) прикладная программа;
Б) системная программа;
В) система программирования;
Г) текстовый редактор.

2. Драйвер – это:

А) устройство компьютера;
Б) программа для работы с устройствами компьютера;
В) прикладная программа;
Г) язык программирования.

3. Программа, работающая под управлением Windows, называется:

А) приложение;
Б) документ;
В) среда;
Г) как-то иначе.

4. Операционную систему с диска загружает в ОЗУ:

А) BIOS;
Б) драйвер;
В) загрузчик операционной системы;
Г) сервисная программа.

5. Свойствами Рабочего стола является:

А) оформление Рабочего стола;
Б) ярлыки, папки, файлы, расположенные на Рабочем столе;
В) дата изготовления Рабочего стола;
Г) имя пользователя, работающего с Рабочим столом.

6. Активизировать или выделить файл, или папку можно:

А) двойным щелчком мыши;
Б) щелчком;
В) протаскиванием;
Г) указыванием.

7. На панели задач находятся:

А) кнопки свернутых программ;
Б) только ярлыки;
В) кнопка Пуск;
Г) кнопка Пуск и значки свернутых и работающих программ.

8. Главное меню открывается:
А) щелчком по значку Мой компьютер;
Б) кнопкой Пуск;
В) контекстным меню;
Г) щелчком на Панели задач.
9. Окно – это:
А) рабочая область;
Б) основное средство общения с Windows;
В) приложение Windows;
Г) событие Windows.
10. Где расположена строка меню окна:
А) сверху;
Б) снизу;
В) слева;
Г) справа.

Тема 5.5. Этапы разработки приложений

1. Информационная система включает в себя следующие компоненты:
а) база данных;
б) программное обеспечение базы данных;
в) прикладное программное обеспечение;
г) аппаратное обеспечение (в том числе устройства хранения);
д) жизненный цикл информационной системы
ж) персонал, разрабатывающий и использующий эту систему.
2. Планирование разработки базы данных - это
а) процесс создания проекта базы данных, предназначенный для поддержки функционирования предприятия и способствующий достижению его целей.
б) подготовительные действия, позволяющие с максимально возможной эффективностью реализовать этапы жизненного цикла приложений баз данных.
в) выбор СУБД подходящего типа, предназначенной для поддержки создаваемого приложения базы данных.
г) проектирование интерфейса пользователя и прикладных программ, предназначенных для работы с базой данных.
3. Проектирование базы данных -
а) процесс создания проекта базы данных, предназначенный для поддержки функционирования предприятия и способствующий достижению его целей.
б) подготовительные действия, позволяющие с максимально возможной эффективностью реализовать этапы жизненного цикла приложений баз данных.
в) выбор СУБД подходящего типа, предназначенной для поддержки создаваемого приложения базы данных.
г) проектирование интерфейса пользователя и прикладных программ, предназначенных для работы с базой данных.
4. Эксплуатация и сопровождение -
а) процесс создания проекта базы данных, предназначенный для поддержки функционирования предприятия и способствующий достижению его целей.

б) подготовительные действия, позволяющие с максимально возможной эффективностью реализовать этапы жизненного цикла приложений баз данных.
в) выбор СУБД подходящего типа, предназначенной для поддержки создаваемого приложения базы данных.
г) проектирование интерфейса пользователя и прикладных программ, предназначенных для работы с базой данных.
д) наблюдение за системой и поддержка ее нормального функционирования по окончании развертывания.

5. Планирование разработки базы данных включает в себя

а) Проектирование базы данных
б) Определение требований к системе
в) Выбор целевой СУБД
г) Сбор и анализ требований пользователей

6. Администрирование баз данных включает в себя

а) Проектирование базы данных
б) Определение требований к системе
в) Выбор целевой СУБД
г) Сбор и анализ требований пользователей

7. Создание прототипа — это

а) Физическая реализация базы данных и разработанных приложений.
б) Перенос любых существующих данных в новую базу данных и загрузка, и модификация любых существующих приложений с целью организации совместной работы с новой базой данных.
в) создание рабочей модели приложения баз данных.
г) проектирование интерфейса пользователя и прикладных программ, предназначенных для работы с базой данных.

8. Расположите в правильном порядке

а) Выбор СУБД
б) Логическое проектирование
г) Физическое проектирование
д) Концептуальное проектирование

9) Вставьте пропущенное слово

_____подход проектирования баз данных начинается с разработки моделей данных, которые содержат несколько высокоуровневых сущностей и связей, затем работа продолжается в виде серии нисходящих уточнений низкоуровневых сущностей, связей и относящихся к ним атрибутов.

10. На каком этапе производят оценку показателей уже существующих информационных систем с целью выявления их сильных и слабых сторон?

а) Проектирование базы данных
б) Администрирование баз данных
в) Тестирование
г) Планирование разработки базы данных.

Критерии оценки:

– оценка «отлично» выставляется обучающемуся за 86-100% правильных ответов на вопросы теста;

– оценка «хорошо» выставляется обучающемуся за 70-85% правильных ответов на вопросы теста;

– оценка «удовлетворительно» выставляется обучающемуся за 50-69% правильных ответов на вопросы теста;

– оценка «неудовлетворительно» выставляется обучающемуся, если правильных ответов на вопросы теста менее 50%.

2. Примерные вопросы для проведения устного и/или письменного опроса

Тема 1.1. Языки программирования

- Области применения языков программирования.
- Стандарты языков программирования.
- Жизненный цикл программы.

Тема 1.2. Типы данных

- Программный продукт и его характеристики.
- Основные этапы решения задач на компьютере.
- Типы данных.

Тема 2.1. Операторы языка программирования

- Структура программы.
- Оператор присваивания. Составной оператор.
- Условный оператор. Оператор выбора.

Тема 3.1. Процедуры и функции

- Циклы: с постусловием, с предусловием, с параметром, вложенные.
- Массивы.
- Понятие и особенности использования подпрограмм.

Тема 3.2. Структуризация в программировании

- Методы структурного программирования.

Тема 3.3. Модульное программирование

- Модульное программирование: понятие, назначение, возможности.

Тема 4.1. Указатели.

- Структуры данных на основе указателей.

Тема 5.1. Основные принципы объектно-ориентированного программирования (ООП)

- Основные принципы объектно-ориентированного программирования.
- Интерфейс среды разработчика.

Тема 5.2. Интегрированная среда разработчика.

- Настройка среды и параметров проекта.

Тема 5.3. Визуальное событийно-управляемое программирование

- Визуальное событийно-управляемое программирование.

Тема 5.4. Разработка оконного приложения

- Особенности разработки оконного приложения.

Тема 5.5. Этапы разработки приложений

- Этапы разработки приложений.

Тема 5.6. Иерархия классов.

- Иерархия классов

Критерии оценки:

– оценка **«отлично»** выставляется обучающемуся, если полно излагает изученный материал, правильно воспроизводит определения понятия, термины; обнаруживает понимание материала, может обосновать свои суждения, применить знания на практике, привести необходимые примеры не только по учебнику, но и самостоятельно составленные; излагает материал последовательно и правильно с точки зрения норм литературного языка;

– оценка **«хорошо»** выставляется обучающемуся, если студент даёт ответ, удовлетворяющий тем же требованиям, что и для оценки "5", но допускает 1-2 ошибки, которые сам же исправляет, и 1-2 недочёта в последовательности и языковом оформлении излагаемого;

– оценка **«удовлетворительно»** выставляется обучающемуся, если излагает материал неполно и допускает неточности в определении понятий или формулировке правил; не умеет достаточно глубоко и доказательно обосновать свои суждения и привести свои примеры; излагает материал непоследовательно и допускает ошибки в языковом оформлении излагаемого;

– оценка **«неудовлетворительно»** выставляется обучающемуся, если студент обнаруживает незнание большей части соответствующего раздела изучаемого материала, допускает ошибки в формулировке определений и правил, искажающие их смысл, беспорядочно и неуверенно излагает материал.

3. Примерная тематика рефератов, докладов, сообщений

Тема 1.1. Языки программирования

– Обзор языков программирования», «Модели жизненного цикла

Тема 2.1. Операторы языка программирования

– Алгоритмические конструкции», «Блок-схемы

Тема 5.1. Основные принципы объектно-ориентированного программирования (ООП)

– Обзор языков ООП

Тема 5.6. Иерархия классов.

– Этапы проектирования интерфейсов

Критерии оценки:

– оценка **«отлично»** выставляется обучающемуся, если работа выполнена полностью и правильно, сделаны правильные выводы; работа выполнена по плану с учетом техники безопасности.

– оценка **«хорошо»** выставляется обучающемуся, если работа выполнена правильно с учетом нескольких несущественных ошибок, исправленных самостоятельно по требованию преподавателя;

– оценка **«удовлетворительно»** выставляется обучающемуся, если работа выполнена правильно не менее чем на половину или допущена существенная ошибка;

– оценка **«неудовлетворительно»** выставляется обучающемуся, если допущены существенные ошибки в ходе работы, которые студент не может исправить даже по требованию преподавателя либо работа не выполнена.

4. Примерные задания практических работ и ситуационные задания

Тема 2.1. Операторы языка программирования.

Составление программ линейной структуры.

1. Ввести программу, которая использует потоковый вывод данных. Проанализировать ее текст и выполнить.

```
#include <iostream>
void main()
{
setlocale (LC_CTYPE, "Russian");
using namespace std;
cout<<"Тип Размер в байтах" << endl;
cout<<" int: "<<sizeof(int)<<"\n";
cout<<" char: "<<sizeof(char)<<"\n";
cout<<" float: "<<sizeof(float)<<"\n";
cout<<" double: "<<sizeof(double)<<"\n";
// sizeof определяет размер объекта в байтах
}
```

setlocale (LC_CTYPE, "Russian"); используется для вывода русского текста.

Библиотека потоков ввода/вывода определяет: cout – стандартный поток вывода (на экран дисплея); cin – стандартный поток ввода (с клавиатуры); cerr, clog – стандартный поток сообщений об ошибках. Вывод осуществляется с помощью оператора cout <<…, ввод с помощью операторacin>> ….

Оператор endl переводит строку на уровень ниже.

Если в программе записано using namespace std; (используя пространство имен), то перед каждым оператором вывода можно не писать std:

// – начало комментария, конец которого определяется концом строки.

2. Добавить в программу п.1 ввод переменных различных типов. Выполнить вывод этих переменных, используя потоковый вывод с манипуляторами. Опробовать программу, приведенную ниже.

Выражение cin >> x; используется для ввода целого числа с клавиатуры в переменную *x* (при выполнении программы нужно набрать число на клавиатуре и нажать клавишу Enter).

```
#include <iostream>
#include <stdio.h>
#include <iomanip>
void main()
{ setlocale(LC_CTYPE, "Russian");
using namespace std;
char c, probel; probel = ' ';
cout << "Введите символ "; cin >> c;
cout << setw(35) << setfill(probel) << probel;
cout << setw(10) << setfill(c) << c << endl;
cout << setw(34) << setfill(probel) << probel;
cout << setw(12) << setfill(c) << c << endl;
cout << setw(33) << setfill(probel) << probel;
```

cout << setw(14) << setfill(c) << c << endl;
}

При выводе данных могут использоваться *манипуляторы*: endl – перейти на новую строку; ends – вывести нулевой байт (признак конца строки символов); dec – вывод числа в десятичной системе; oct – вывод числа в восьмеричной системе; hex – вывод числа в шестнадцатеричной системе счисления; setw(int n) – установить ширину поля вывода в n символов; setfill(int n) – символ-заполнитель; setprecision(int n) – установить количество цифр после запятой при выводе вещественных чисел; setbase(int n) – установить систему счисления для вывода чисел (n может принимать значения 0, 2, 8, 10, 16, причем 0 означает систему счисления по умолчанию, т.е. 10). Манипуляторы определены в заголовочном файле <iomanip>, поэтому при их использовании надо добавлять директиву #include <iomanip>.

Пример вывода числа из 6 символов (3 до запятой и 2 после):

using namespace std;

……

double x;

cout << setw(6) << setprecision(2) << x << endl;

3. Выполнить программу, записанную в правой части, которая использует форматированный ввод-вывод данных. Проанализировать текст программы. Изменить программу так, чтобы выводилась своя фамилия, имя и отчество, факультет, номер группы русскими буквами. При выводе использовать управляющие коды.

Пример программы, выводящей слово "Privet".

#include <stdio.h>
#include <conio.h>
void main()
{
printf("\n\t Privet\n");
printf("\n... Press key");
_getch();
}
	Заголовочный файл с именем stdio.h –стандартный ввод-вывод. Файл conio.h. поддерживает функцию _getch(), которая предназначена для приема сообщения о нажатии клавиши на клавиатуре. Функция printf() печатает аргументы. С помощью символа \n осуществляется перевод на новую строку. Символ \tосуществляет табуляцию, т.е. начало вывода результатов с отступом вправо. Функция _getch() ожидает нажатия клавиши.

Можно управлять перемещением курсора на экране и выполнять некоторые другие функции, используя управляющие коды: \n - перемещает курсор в начальную позицию следующей строки; \t– перемещает курсор в следующую позицию табуляции экрана; \r – выполняет «возврат каретки», перемещая курсор к началу той же строки без перехода на следующую; \b – передвигает курсор только на одну позицию влево.

4. Написать программу, реализующую диалог, используя пример, записанный ниже.

Функция puts() осуществляет вывод символов на экран. Требует подключения #include <stdio.h>.

Функция gets() вводит строку в переменную, записанную в скобках. Параметром puts() может быть строковая переменная: char a[] = “Привет!”; puts (a);

Функция putchar() предназначена для вывода единичного символа на экран.

П

```
#include <iostream>
#include <stdio.h>
void main()
{
setlocale (LC_CTYPE, "Russian");
char name[60];
puts("Как вас зовут? ");
gets(name);
printf ("Привет, %s\n", name);
}
```

араметром функции может быть символ (например:putchar (‘H’);) или символьная переменная:char letter; letter=’G’; putchar (letter);

Функция getchar() вводит с клавиатуры единичный символ:

```
int letter; letter = getchar();
```

5. В соответствии со своим вариантом написать программу по условию, приведенному в таблице ниже.

№ варианта	Условие задачи
1	Вывести полукруг на консольное окно, закрашенное введенным символом.
2	Вывести шестиугольник по центру консольного окна, закрашенный введенным символом.
3	Вывести елочки, закрашенные введенным символом.
4	Вывести круг по центру консольного окна, закрашенный введенным символом
5	Вывести треугольник по центру консольного окна, закрашенный введенным символом
6	Вывести звезду, закрашенную введенным символом.
7	Вывести трапецию по центру консольного окна, закрашенную введенными символами
8	Вывести человечка на консольное окно, закрашенного введенным символом.
9	Вывести разнонаправленные стрелки (→↑↓←), состоящие из введенного символа.
10	Вывести овал на консольное окно, закрашенный введенным символом.
11	Вывести квадрат на консольное окно, закрашенный введенным символом.
12	Вывести знак + на консольное окно, закрашенный введенным символом.
13	Вывести сердце ♥ на консольное окно, закрашенное введенным символом.
14	Вывести ромб ♦ на консольное окно, закрашенный введенным символом.
15	Вывести домик на консольное окно, закрашенный введенным символом.
16	Вывести грибок, закрашенный введенным символом.

Составление программ разветвляющейся структуры.

1. Выполнить программу несколько раз с различными значениями переменной j. Переделать программу с тем, чтобы ввод и вывод осуществлялся с помощью потоковых операторов ввода-вывода.

Пример. Вычислить значение z по формулам:

$$z = \begin{cases} \sqrt{x + 0{,}3j} + b, & b < 1{,}5 \\ |xj + b|, & b \geq 1{,}5 \end{cases},$$

b = s + (5*x + j),

где $x = 4 \cdot 10^{-3}$; s = 1,1; j = {2; 6,8; 0,03; 55; …}.

Здесь в программе для организации разветвлений используется условный оператор if.

Формат оператора: if <условие> оператор-1; [else оператор-2;] или if <условие> оператор-1;

Выполнение оператора if начинается с проверки условия. Если условие истинно, то выполняется оператор 1, если условие ложно, то выполняется оператор 2.

Если условие ложно и отсутствует оператор 2, то выполняется следующий за if оператор.

П

```
#include <iostream>
#include <stdio.h>
void main ()
{
setlocale (LC_CTYPE, "Russian");
float b, z, x = 4e-3, s = 1.1, j;
printf ("Введите j ");
scanf("%f", &j);
b = s + (5*x + j);
if (b < 1.5)
z = sqrt(x + 0.3 * j) + b;
else
z = abs(x * j + b);
printf("j=%f\t", j);
printf("z=%f\n", z);
}
```

При каждом повторении операторов цикла если выражениеb <1,5 истинно, то z вычисляется по первой формуле. В противном случае – по второй.

2. Выполнить программу. Изменить программу с тем, чтобы ввод и вывод осуществлялся с помощью потоковых операторов ввода-вывода.

```
#include <iostream>
#include <stdio.h>
void main()
{
setlocale (LC_CTYPE, "Russian");
float x, y, z, min, max;
printf("Введите x "); scanf("%f", &x);
printf("Введите y "); scanf("%f", &y);
```

```
printf("Введите z "); scanf("%f", &z);
if ((x + y + z) < (x * y * z))
max = (x * y * z);
else
max = (x + y + z);
if (z < y) min = z; else min = y;
if (x < min) min = x;
printf("max(x + y + z, xyz) * min(x, y, z) = %f \n", min*max);
}
```

Выполнить задания:

1. Ввести с клавиатуры вещественные числа x, y, z, используя функции форматированного ввода. Вычислить max(x + y + z, xyz) * min(x, y, z).

2. Выполнить программу с использованием оператора switch. Записать условие задачи. Изменить программу с тем, чтобы в программе присутствовал оператор goto.

Для организации выбора из множества различных вариантов используется оператор выбора switch. Формат оператора:

```
switch (выражение)
  { [ case  константное выражение 1]: [ список операторов 1];
……..
  [ case  константное выражение n]: [ список операторов n];
   [ default: [ список операторов ];] }
```

В

```
#include <iostream>
int main()
{
setlocale (LC_CTYPE, "Russian");
int k;
puts ("Вы хотите купить автомобиль? (1-да, 2-нет)");
std::cin>>k;
switch (k)
{
case 2: puts ("Правильно. Полезно ходить пешком");
break;
case 1: puts ("Какой? (1- Майбах, 2-Тойота, 3-Лада)");
{ std::cin>>k;
switch (k)
{
case 1: puts ("Не слабо"); break;
case 2: puts ("Хороший выбор"); break;
case 3: puts ("Сомнительный выбор"); break;
default: ;
}
}
default: ;
```

```
}
return 0;
}
```

Выражение в круглых скобках может быть целой или символьной константой. Выполняется оператор следующим образом: вычисляется выражение в круглых скобках, и полученное значение последовательно сравнивается с константными выражениями, следующими за ключевыми словами case. Если одно из константных выражений совпадает со значением константного выражения, то управление передается на оператора, помеченный соответствующим ключевым словом case. Если совпадения нет, то управление передается на оператора, помеченный ключевым словом default, а в случае его отсутствия управление передается на следующий после switch оператор.

Оператор break изменяет поток управления, он передает управление оператору, следующему за switch.

Оператор return 0; завершает выполнение функции и возвращает системе значение 0.

В некоторых случаях приходится использовать оператор goto, который передает управление на оператор, помеченный меткой (например, оператор goto a; передает управление оператору: a:std::cin>>k; Здесь a – метка оператора).

3. Написать программу, реализующую диалог на любую тему, с использованием оператора switch.

Составление программ циклической структуры.

1. Выполнить программу с использованием оператора for. Проанализировать ее.

```
#include <iostream>
#include <stdio.h>
#include <cmath>
void main()
{ setlocale (LC_CTYPE, "Russian");
float z, y, x, st;
st = 1.0 / 3.0;
for (int n = 1; n <= 4; n++)
{ printf ("Введите x ");
scanf("%f", &x);
z = 2 * pow(x, 2);
y = z + pow(x, st);
printf("x = %5.2f\t", x);
printf("y = %5.2f\n", y); }
}
```

2. Вычислить значение y по формулам: $z = 2*x^2, y = z + \sqrt[3]{x}$, где x = {4; 3,5; 7; 1}.

Математические функции декларированы в файлах <cmath>. Тело цикла for заключается в фигурные скобки. Вычисляется выражение n=1; проверяется условие n<=4; если оно истинно, то выполняются операторы цикла. Затем вычисляется n=n+1, вновь проверяется условие n<=4; если оно истинно, то выполняются операторы цикла. Так продолжается до тех пор, пока n не станет больше 4. Тогда управление передается оператору, следующему за телом цикла.

Результаты выводятся в виде действительных чисел, занимают 5 символов, из которых 2 отводятся для записи дробной части.

3. Выполнить программу с использованием оператора while. Проанализировать ее.

```
#include <stdio.h>
#include <cmath>
void main()
{
float z, y, x = 3;
while (x <= 4)
{ z = 2 * pow(x,2);
y = z + pow(x,1 / 3);
printf("x=%5.2f\t", x);
printf("y=%5.2f\n", y);
x = x + 0.1; }
}
```

4. Вычислить значениеу по формулам: $z=2*x^2, y=z+\sqrt[3]{x}$, гдех = 3(0,1)4 – меняется от 3 до 4 с шагом 0,1. Тело цикла while заключается в фигурные скобки. Проверяется условие в скобках (x<=4), если оно истинно, то выполняются операторы цикла до тех пор, пока оно не станет ложным. Тогда управление передается оператору, следующему за телом цикла. Если в программе будет отсутствовать оператор x=x+0.1, то программа зациклится. Надо прервать выполнение программы, нажав Ctrl – Break, и исправить ошибку.

Выполнить задания:

1. Выполнить программу с использованием оператора do while. Проанализировать ее**.**

```
#include <stdio.h>
#include <cmath>
void main()
{ float z, y, x = 3;
do
{ z = 2 * pow(x, 2);
y = z + pow(x, 1 / 3);
printf("x=%5.2f\t", x);
printf("y=%5.2f\n", y);
x = x + 0.1; }
while (x <= 4);
}
```

2. Вычислить значение y по формулам: $z=2*x^2, y=z+\sqrt[3]{x}$, где x = 3(0,1)4 – меняется от 3 до 4 с шагом 0,1. Оператор цикла do while называется оператором цикла с постусловием. Сначала выполняется тело цикла, затем проверяется условие, записанное в скобках (x <=4), если оно истинно, то выполняются операторы цикла до тех пор, пока оно не станет ложным. Тогда управление передается оператору, следующему за телом цикла.

3. Выполнить программу, содержащую двойной цикл. Записать условие задачи. Оформить вывод результатов, используя различные возможности операторов вывода.

```
#include <stdio.h>
using namespace std;
void main()
{
int n, i, j;
printf ("Enter n: ");
scanf ("%d", &n);
for(i = 1; i <= n ; i++)
{ for(j = 1; j <= n ; j++)
printf ("%5d", i * j);
printf("\n"); }
}
```

Добавить в заголовочную часть:

```
#include <iostream>
#include <iomanip>
```

Перед printf("\n"); разместить операторы:

```
if (i == 1)
{ cout<<endl;
cout << setw(5*n)<<setfill('-') << '-'<<endl; }
```

Пояснить результат.

4. В таблице приведены формулы и два варианта исходных данных, по которым надо составить две циклические программы с одними и теми же расчетными формулами. Для каждой из задач составить блок-схему алгоритма. При наличии ошибок из-за некорректных исходных данных, выполнить вычисления с другими числами.

№	Формулы для вычислений	Исходные данные (for)	Исходные данные (while)
1	$h = (10r - j)/(c^2 + e^{-m})$ $y = (h \cdot m - j^2) + (0{,}1c)^2$	$c = 2{,}1;\ r = 4 \cdot 10^{-4}$ $m = 7;$ $j = \{4{,}2;\ 0{,}3;\ 1{,}7\}$	$j =$ $= 0(0{,}1)1{,}7$
2	$y = \sqrt{i \cdot b - b^2 \cdot a}$ $z = y \cdot \mathrm{tg}(n/4) - e^{1+b}$	$a = 2 \cdot 10^{-3};$ $b = 8{,}5;\ n = 2;$ $i = \{2;\ 1;\ 8{,}3\}$	$i =$ $= 1(0{,}5)3$
3	$z = \sqrt{a+1} - \mathrm{tg}(j + y)$ $q = e^{-j\,0{,}1y} + (3 \cdot z)^2$	$y = -1{,}55;\ a = 6$ $j = \{8;\ -0{,}6;\ 1;\ 6{,}4\}$	$j =$ $-2(0{,}2)1$
4	$y = (m \cdot j)/\mathrm{tg}\,a - e^{10m}$ $z = 2y \cdot b + \sqrt{a+b})$	$b = 2{,}5;\ m = 3$ $a = 1{,}4 \cdot 10^{-3}$ $j = \{0{,}7;\ -2;\ 2{,}9\}$	$j =$ $= b(0{,}1)3$
5	$c = i/k - \sqrt{y}/0{,}4$ $d = e^{1-c} + 4{,}9(x^2 + 1)$	$i = \{0{,}9;\ 8{,}4;\ 2\}$ $y = 1{,}6 \cdot 10^{-4}$ $x = -1;\ k = 6$	$i =$ $= 0(0{,}5)3$

№	Формулы для вычислений	Исходные данные (for)	Исходные данные (while)
6	$v = u + b - 2\sqrt{0{,}7k + m}$ $w = 100 \cdot e^{-0{,}2u} - \ln(8{,}1u)$	$b = -5{,}4;\ m = 4$ $u = 0{,}05 \cdot 10^{-4}$ $k = \{6;\ 4;\ 0{,}3;\ -7\}$	$k =$ $= 1(0{,}5)3$
7	$z = t \cdot y^2 - \sqrt{i + x} \cdot \operatorname{tg} y$ $q = \sqrt{z^2 + 5z} \cdot \ln(y)$	$x = 0{,}9;\ y = 2;$ $t = 6{,}96 \cdot 10^{-5}$ $i = \{9;\ -1{,}4;\ 5\}$	$i =$ $= 1(0{,}2)2$
8	$w = 0{,}6j + e^{t \cdot j} \cdot (4y / n)^2$ $s = \sqrt{w - 0{,}1t} / (2 + n^2)$	$y = 5;\ t = -7{,}4;$ $n = 9;$ $j = \{0{,}5;\ 8{,}4;\ 0{,}3\}$	$j =$ $= 0(0{,}1)2$
9	$y = (s + c) / \ln(f) / e^{-s}$ $h = (y - m) / \ln(y)$	$s = 5{,}9;\ m = 6$ $f = 1{,}2 \cdot 10^{-3}$ $c = \{2;\ 0{,}5;\ -4\}$	$c =$ $= 0(0{,}1)2$
10	$y = s / \ln(5{,}2f) / (e^{-s} + 2)$ $v = (1 + m \cdot y - m \cdot k) / \ln(y)$	$s = 7{,}4;\ m = 10$ $f = 3{,}2 \cdot 10^{-4}$ $k = \{4;\ 0{,}5;\ 8\}$	$k =$ $= 0(0{,}5)4$
11	$w = \operatorname{tg}(a / 3) + e^{a/m}$ $r = 0{,}9\sqrt{w + j} + \lvert a^2 - 1 \rvert$	$a = -1{,}4;\ m = 16$ $j = \{0{,}5;\ 9{,}1;\ 5\}$	$j =$ $= 1{,}8(0{,}2)3$
12	$y = (m - b) / (\sin a - e^a)$ $z = 3y + \sqrt{a - 4b}$	$b = 0{,}5;\ m = 8$ $a = 2{,}4 \cdot 10^{-4}$ $j = \{1{,}7;\ 2{,}9;\ -8\}$	$j =$ $= 2(0{,}2)3$
13	$d = \sin(k / a) / \cos(m \cdot b)$ $c = d / (d^2 + 1) / (1 - e^k)$	$a = 8;\ m = 6$ $b = 5 \cdot 10^{-3}$ $k = \{1{,}6;\ 9{,}1;\ 8\}$	$k =$ $= a(-0{,}5)3$
14	$y = a / (b + e^b) / (1 + j \cdot i)$ $t = \cos(y + 1) / \sqrt{\lvert -2j \rvert}$	$b = 2;\ j = 6$ $a = 1{,}5 \cdot 10^{-8}$ $i = \{7;\ 4;\ 2;\ 6\}$	$i =$ $= 3(-0{,}1)2$
15	$z = \sqrt{t \cdot a + y} + 4e^{-2 \cdot j}$ $w = \ln(0{,}4y) / (7a - z)$	$a = -1;\ y = 0{,}4;$ $t = 5 \cdot 10^{-5};$ $j = \{5;\ 3;\ 1;\ 7;\ 3\}$	$j =$ $= 1(0{,}2)2$
16	$w = \cos(a / 3) + t + e^{a/m}$ $r = 0{,}7\sqrt{3w + j} + \lvert a^2 - 1 \rvert$	$a = -1{,}4;\ m = 6$ $t = 15 \cdot 10^{-5}$ $j = \{0{,}5;\ 9{,}1;\ 5\}$	$j =$ $= 1(0{,}1)2$

Обработка одномерных массивов.

Выполнить программу. Опробовать второй вариант генерации чисел в 13 строке. Добавить в программу операторы вычисления суммы элементов массива A. Произвести отладку.

```
#include <iostream>
#include <time.h>
void main()
{ setlocale(LC_CTYPE, "Russian");
using namespace std;
const int N = 100;
int i, sz, A[N]; int rmn = 0, rmx = 99;
cout << "Введите размер массива ";
cin >> sz;
cout << "Массив A:" << endl;
srand((unsigned)time(NULL));
for ( i = 0; i < sz; i++)
{ A[i] = (int) ((( double)rand () / (double)RAND_MAX)*(rmx-rmn)+rmn);
// или A[i]=rand()%99;
cout << A[i] << endl; }
}
```

1. Сформировать одномерный статический массив целых чисел A, используя датчик случайных чисел (диапазон значений от 0 до 99). Массив заполняется случайными числами с помощью функции rand из стандартной библиотеки. Она генерирует целое число в диапазоне от 0 до RAND_MAX (символическая константа, определенная в заголовочном файле <stdlib>). При этом необходимо использование стандартной библиотечной функции srand, которая получает целый аргумент unsigned и при каждом выполнении программы задает начальное число, которое функция rand использует для генерации последовательности квазислучайных чисел. Чтобы не вводить каждый раз начальное число, можно использовать оператор: srand(unsigned)time(NULL);. Здесь для автоматического получения начального числа считываются показания часов с помощью функции time (с аргументом NULL, как записано в указанном выше операторе). Функция возвращает текущее «календарное время» в секундах, которое преобразуется в беззнаковое целое число. При использовании функции time нужно включить в заголовок #include <time.h> или #include <locale>

2. Выполнить программу. Записать ее условие. Добавить в программу операторы вычисления среднего значения исходного массива. Произвести отладку.

```
#include <iostream>
void main()
{
setlocale(LC_CTYPE, "Russian");
using namespace std;
int i, k, sz=4;
float A[ ] = {5, -4, 17.1, 9, 1};
cout << "Введите номер элемента (от 0 до 4) ";
cin >> k;
for ( i = k; i <=sz; i++)
```

```
A[i] = A[i+1];
sz--;
for ( i = 0; i <= sz; i++)
cout << A[i] << endl;
}
```

Инициализация массива означает присвоение начальных значений его элементам при объявлении. Массивы можно инициализировать списком значений, заключенных в фигурные скобки:

```
float A[ ] = {5, -4, 17.1, 9, 1};
```

Длина массива вычисляется компилятором по количеству значений, перечисленных в фигурных скобках.

3. Выполнить прокрутку программы, на любых конкретных числах и записать условие задачи. Опробовать программу. Реализовать условие задачи для массивов A и B размера 7. Произвести отладку.

```
#include <stdio.h>
void main()
{ int a[5], b[5], c[10]; int k = 0, l = 0, i = 0;
printf("A:\n");
for ( int n= 0; n < 5; n++) scanf("%d", &a[n]);
printf("B:\n");
for( int n = 0; n < 5; n++) scanf("%d", &b[n]);
do
{ if (a[k] <= b[l])
{ c[i] = a[k]; k++; }
else
{ c[i] = b[l]; l++; }
i++;
if (k == 5)
for (; l < 5; l++)
{ c[i] = b[l]; i++; }
if (l == 5)
for(; k < 5; k++)
{ c[i] = a[k]; i++; }
}
while (i < 10);
printf("\n");
for (i = 0; i < 10; i++) printf("%d ", c[i]);
printf("\n");
}
```

Выполнить задания:

1. Даны два массива целых чисел A и B размера 5, элементы которых предварительно упорядочены по возрастанию. Сформировать массив C …

2. Выполнить программу. Внести изменения с тем, чтобы вычислялся минимальный элемент массива. Произвести отладку.

Пример. Сформировать массив целых чисел в количестве не более 30. Размерность массива ввести с клавиатуры.
Найти в массиве наибольший элемент.

```
#include <locale>
#include <iostream>
void main()
{
setlocale(LC_ALL, "rus");
using namespace std;
int n, i, a[30], kmax = 0;
cout << "Введите размер массива (не более 30)" << endl;
cin >> n;
srand((unsigned)time(NULL));
for ( i = 0; i < n; i++)
{
a[i] = rand() % 30;
cout << a[i] << " ";
}
cout << endl;
for (i = 1; i < n; i++)
if (a[i] > a[kmax]) kmax = i;
cout << "Максимальный элемент" << a[kmax] << endl;
}
```

3. Сформировать одномерный массив целых чисел, используя датчик случайных чисел (диапазон от 0 до 99). Размер массива ввести с клавиатуры. В соответствии со своим вариантом написать программу по условию, представленному в таблице ниже. Составить блок-схему алгоритма.

4. Внести изменения в программу с тем, чтобы исходные значения в количестве 10 элементов не формировались случайным образом, а инициализировались в программе.

№ варианта	Условие задачи
1	Удалить элемент с номером k. Добавить после каждого четного элемента массива элемент со значением 0.
2	Все четные элементы целочисленного массива K(n) поместить в массив L(n), а нечетные – в массив M(n). Подсчитать количество тех и других.

№ варианта	Условие задачи
3	Удалить элементы, индексы которых кратны 7. Добавить после каждого нечетного элемента массива элемент со значением 4.
4	Поменять местами минимальный и максимальный элементы массива.
5	В массиве С каждый третий элемент заменить полусуммой двух предыдущих. Дополнительный (рабочий) массив не использовать.
6	Удалить все элементы с заданным значением, если они имеются в массиве. Добавить перед каждым четным элементом массива элемент со значением 1.
7	Удалить из массива все элементы, совпадающие с его минимальным значением. Добавить в начало массива три элемента со значением, равным среднему арифметическому массива.
8	Найти максимальный элемент массива и заменить им нечетные по номеру элементы.
9	Найти в массиве элемент, наиболее близкий к среднему арифметическому суммы его элементов.
10	Найти в массиве элемент, если он существует, равный среднему арифметическому суммы трех его последних элементов.
11	Удалить пять первых нечетных элементов массива. Добавить в конец массива три новых нулевых элемента
12	Найти минимальный элемент массива Т и заменить им четные по номеру элементы.
13	В массиве A(n) каждый элемент, кроме первого, заменить суммой всех предыдущих.
14	В массиве найти первый и последний нулевые элементы. Вывести их индексы.
15	Удалить элементы, индексы которых кратны 3. Добавить после каждого отрицательного элемента массива элемент со значением 10|.
16	В массиве найти первый и последний минимальные элементы. Вывести их индексы.

Тема 4.1. Основные принципы объектно-ориентированного программирования (ООП)

Выполнить задания:

1. Изучить работу с указателями, выполнив программу

Указатель – это переменная, значением которой является адрес.

Пусть объявлена переменная a: int a = 0; Указатель на нее: int *ptr; Можно в указатель поместить адрес переменной и напечатать его: ptr = &a; cout << ptr;

Здесь & – операция получения адреса, * – операция разыменования. Через указатель можно изменять значение, хранящееся по адресу:

```
int a, *ptr; ptr = &a; a = 10;
cout << a<<" "<<*ptr<<endl;
*ptr = 20;
c
#include <iostream>
using namespace std;
void main()
{ int a = 10, *pa, b = 20, *pb;
pa = &a;
cout << &a<<" "<<a<<endl;
cout << pa<<" "<<*pa<<endl;
pb = &b;
cout << pb<<" "<<*pb<<endl;
*pa = *pb;
cout << &a<<" "<< a<<endl;
}
out << a<<" "<<*ptr<<endl;
```

Чтобы выполнить арифметические и логические операции над указателями или над объектами, на которые они указывают, необходимо при выполнении каждой операции явно определить тип объектов. При выполнении операций учитывается тип указателей.

Например: char *pz; int *pk, *pi; float *pf;

```
pz++; //значение указателя изменяется на 1
pk++; //значение указателя изменяется на 2
pf++; //значение указателя изменяется на 4
```

2. Ниже записаны фрагменты программ с использованием указателя на константу, константного указателя, константного указателя на константу. Убрать ошибочные операторы, дописать операторы вывода и выполнить программы на компьютере.

Указатель на константу	Константный указатель	Константный указатель на константу

{ int a = 9;
const int *pa;
pa = &a;
*pa = 12; // ошибка (нельзя менять значение переменной a)
	{ int a = 99;
	int* const pa = &a;
	int b = 44;
	pa = &b; // ошибка (константный указатель менять нельзя)
		{ int a = 99;
		const int* const pa = &a;
		*pa = 33; // ошибка (нельзя менять содержимое переменной a)
		int b = 44;

int b = 20;
pa = &b; }
	*pa = 12;
	}
		pa = &b; // ошибка (константный указатель на константу менять нельзя)
		}

3. Ниже записана программа суммирования значений a, которые вводятся с клавиатуры, с использованием указателя.

```
#include <iostream>
void main()
{
setlocale (LC_CTYPE, "Russian");
using namespace std;
float a, *pa, s=0; int i;
pa = &a;
for (i = 1; i <= 4; i++)
{
cout<<"Введите a" << i << endl;
cin >> a;
s = s + *pa;
}
cout <<"Ответ " << s << endl;
}
```

4. Написать программу генерации элементов массива A из случайных чисел и их вывода двумя способами (с указателями и без них).

Имя массива A без индекса является указателем-константой (не изменяется на протяжении всей работы программы), т. е. адресом первого элемента массива A[0].

Цикл, в котором генерируется и выводится массив A, содержащий sz элементов (от 0 до sz–1), можно реализовать двумя способами (во втором используются указатели):

1. for (i = 0; i < sz; i++) {A[i] = rand()%99; cout << A[i] << endl; }
2. for (i = 0; i < sz; i++) { *(A +i) = rand()%99; cout << *(A + i) << endl; }

5. Изучить отличия ссылок от указателей. Выполнить программы, записанные ниже.

Ссылку можно рассматривать как указатель, который всегда разыменовывается. Формат объявления ссылки: тип & имя; где тип – это тип величины, на которую указывает ссылка, & – оператор ссылки, означающий, что следующее за ним имя является именем переменной ссылочного типа, например: int kol; int &pal = kol;

Здесь pal – альтернативное имя для kol const char &CR = '\n '; // ссылка на константу

Использование ссылки	Использование указателя

#include <stdio.h>
void main()
{
int V = 1;
printf("V = %d\n", V);
int &rV = V;
	#include <stdio.h>
	void main()
	{
	int V = 1;
	printf("V = %d\n", V);
	int &rV = V;

rV = 5;
printf("V = %d\n", V);
}
	rV = 5; int *pV = &rV;
	printf("V = %d\n", *pV);
	}

Между ссылкой и указателем существуют два основных отличия: ссылка обязательно должна быть инициализирована в месте своего определения; всякое изменение ссылки преобразует не ее, а тот объект, на который она ссылается.

6. Выполнить программу, которая разработана с использованием указателей. Внести изменения с тем, чтобы программа стала содержать ошибки. Исследовать их с помощью отладки.

```
#include <iostream>
void main()
{ setlocale(LC_CTYPE, "Russian");
using namespace std;
int i, k, sz=5;
float A[ ] = {5, -4, 17.1, 9, 1};
cout << "Введите номер элемента (от 0 до 4) " << endl;
cin >> k; cout << endl;
for ( i = k; i < sz-1; i++)
*(A +i) = *(A +i+1);
sz--;
for ( i = 0; i < sz; i++)
cout << *(A +i) << endl;
}
```

7. Пусть имеется массив A. Значения элементов массива инициализируются в программе. Удалить элемент с номером, который вводится с клавиатуры.

Тема 5.1. Интегрированная среда разработчика.

1. Изучить способы преобразования символов, выполнив программу в правой части. Внести изменения в программу с тем, чтобы продемонстрировать, как прописная буква выводится в виде строчной.

```
#include <stdio.h>
void main()
{ int n = 5; char c, t;
c = n + '0';
printf("%c\n", c);
if (c >= '0' && c <= '9') n = c - '0';
printf(" %d\n", n);
c = 'b';
if (c >= 'a' && c <= 'z') t = c-'a' + 'A';
printf(" %c\n", t);
}
```

Заменить в программе четвертую и пятую строки на:

```
char *pc;
pc = &c;
```

```
*pc = n + '0';
printf("%c\n", *pc);
```
Объяснить результат.

2. Выполнить задание из таблицы ниже двумя способами: используя индексы и используя указатели. При написании программ не использовать стандартные операции и функции для строк символов.

№ варианта	Условие задачи
1	Написать программу, реализующую вставку в строку **n** символов, начиная с позиции **k.**
2	Написать программу, реализующую выделение подстроки ***S***1 длиной **k** с позиции номер **n** из строки.
3	В строке есть два символа *****. Получить все символы между первым и вторым символом ***.**
4	Написать программу, которая удаляет в строке все буквы **b** в тексте, написанном латинскими буквами.
5	Исключить из строки группы символов, расположенные между скобками вместе со скобками. Предполагается, что нет вложенных скобок.
6	В строке есть символы *****. Преобразовать строку следующим образом: удалить все символы *****, и повторить каждый символ, отличный от ***.**
7	Преобразовать строку: после каждой буквы **a** добавить символ **!**
8	Написать программу, которая осуществляет сравнение двух строк и выводит сообщение о том, какие символы совпадают.
9	Написать программу, реализующую функцию вставки подстроки **S1** длиной **k** в строку **S** с позиции номер **n.**
10	Написать программу, которая записывает строку в обратном порядке.
11	Вывести текст, составленный из последних букв всех слов.
12	Зашифровать введенную с клавиатуры строку, поменяв местами первый символ со вторым, третий с четвертым и т. д.
13	Отредактировать заданное предложение, удаляя из него все слова с чётными номерами.
14	Найти самое длинное и самое короткое слово в заданном предложении.
15	Из предложения удалить все символы, совпадающие с символом, введенным с клавиатуры.
16	Из текста удалить те его части, которые заключены в кавычки (вместе с кавычками).

Тема 5.3. Визуальное событийно-управляемое программирование

Многомерные массивы

1. Выполнить программу, записанную в правой части. Внести изменения в программу с тем, чтобы инициализировался другой массив, например: D[2][4]. Осуществить вывод массива в виде матрицы.

```
#include <iostream>
void main ()
```

```
{ int a[3] [2]={{1,2}, {3,4}, {5,6 } };
int i, j;
for (i = 0; i < 2; i++)
for (j = 0; j < 3; j++)
std::cout <<"\n a["<< i <<"," << j <<"] ="<< a[i][j]; }
```

Пример программы, которая инициализирует массив и выводит его элементы на экран.

2. Выполнить задание из таблицы ниже двумя способами: используя индексы и используя указатели.

№ варианта	Условие задачи
1	Найти наибольший элемент матрицы A(N, M), а также номера строки и столбца, на пересечении которых он находится.
2	В каждой строке заданной матрицы A(N, M) вычислить сумму, количество и среднее арифметическое положительных элементов.
3	Для заданной целочисленной матрицы A(N, M) определить, является ли сумма её элементов чётным числом.
4	Дана матрица A(N, M). Найти количество элементов этой матрицы, больших среднего арифметического всех её элементов.
5	Дана целочисленная матрица A(N, M). Вычислить сумму и произведение тех её элементов, которые при делении на два дают нечётное число.
6	В заданной матрице A(N, M) поменять местами столбцы с номерами P и Q.
7	Дана матрица A(N, M). Поменять местами её наибольший и наименьший элементы.
8	Даны две целочисленные матрицы A(N, M) и B(N, M). Подсчитать количество тех пар (a_{ij} , b_{ij}) , для которых: а) $a_{ij} < b_{ij}$; б) $a_{ij} = b_{ij}$; в) $a_{ij} > b_{ij}$.
9	Дана матрица A(N, N). Переписать элементы её главной диагонали в одномерный массив Y(N) и разделить их на максимальный элемент главной диагонали.
10	Дана матрица B(n, m). Вычислить произведение чётных положительных элементов матрицы,
11	Найти наибольший элемент главной диагонали матрицы A(N, N) и вывести на печать всю строку, в которой он находится.
12	Дана целочисленная матрица A(N, M). Вычислить сумму и произведение нечётных отрицательных элементов матрицы, удовлетворяющих условию $\lvert a_{ij} \rvert < i$.
13	Найти наименьший элемент главной диагонали матрицы C(N, N) и вывести на печать столбец, в котором он находится.
14	Дана матрица A(N, N) и целое число m. Преобразовать матрицу по правилу: строку с номером M сделать столбцом с номером M, а столбец с номером M сделать строкой с номером M
15	В заданном массиве A(N, N) вычислить две суммы элементов, расположенных выше и ниже главной диагонали.
16	Найти наименьший элемент матрицы B, а также номера строки и столбца, на пересечении которых он находится.

Тема 5.4. Разработка оконного приложения

Структура и назначение функций

Функция – это совокупность объявлений и операторов, предназначенная для решения определенной задачи. Одно из главных преимуществ, предоставляемых функцией, состоит в том, что она может быть выполнена столько раз, сколько необходимо, в различных местах основной программы.

В любой программе C/C++ должна быть функция с именем main (главная функция или основная программа), именно с этой функции начинается выполнение программы. Другая функция (подпрограмма) может иметь любое имя, не совпадающее с ключевыми словами. После имени записываются формальные аргументы. Если надо передать результат из функции в основную программу, то завершается подпрограмма оператором return.

Чтобы обратиться к функции в основной программе записывается имя функции и фактические аргументы, которые должны соответствовать формальным по типу. При вызове функции управление передается первому оператору тела функции, и начинается выполнение функции, которое продолжается до тех пор, пока не встретится оператор return или последний оператор функции. Управление возвращается в точку основной программы, следующей за точкой вызова.

1. Изучить способы организации работы с функциями, выполнив программы. Опробовать работу программы, содержащей функцию, с разными паролями.

Пример программы, в которой происходит проверка пароля (например, qwerty123) в двух вариантах: без использования функции и с использованием функции check_pass.

Функция может быть записана как до основной программы, так и после. В последнем случае вызов функции следует предварять объявлением (прототипом) этой функции.

В том варианте программы, который содержит функцию, int main() – основная программа, а voidcheck_pass (string password) – функция. Перед int main() записан прототип, так как функция расположена после основной программы.

Поскольку в данном примере возвращать результат из функции не нужно, она не содержит оператора return.

```
include <iostream>
#include <string>
using namespace std;
int main()
{
setlocale (LC_CTYPE, "Russian");
string valid_pass = "qwerty123";
string user_pass;
cout << "Введите пароль: ";
getline(cin, user_pass);
if (user_pass == valid_pass)
{ cout << "Доступ разрешен" << endl; }
else
{ cout << "Неверный пароль!" << endl;
} return0;
}
```

```
#include <iostream>
#include <string>
using namespace std;
void check_pass (string password);
int main()
{ setlocale (LC_CTYPE, "Russian");
string user_pass;
cout << "Введите пароль: ";
getline (cin, user_pass);
check_pass (user_pass); return 0; }
void check_pass (string password)
{ string valid_pass = "qwerty123";
if (password == valid_pass)
{ cout<<"Доступ разрешен"<<endl; }
else
{ cout<<"Неверный пароль"<<endl; } }
```

2. Изучить использование рекурсий на примере программы, записанной в правой части.

```
#include<stdio.h>
double power(double x, long n)
{ if (n == 0) return 1.0;
if (n < 0) return 1.0 / (x*power (1.0 / x, n + 1));
return x*power(x, n - 1); }
void main()
{ double x; long n; printf("Enter x and n\n");
scanf ("%lf %ld", &x, &n);
printf("%16.16lf\n", power (x, n)); }
```

Любая функция в программе может быть вызвана рекурсивно, т.е. она может вызывать саму себя.

Пример программы, в которой вычисляется степень числа xn с использованием простой рекурсии для функции power.

Тема 5.5. Этапы разработки приложений

Массивы и ссылки при работе с функциями

1. В правой части записана программа в двух вариантах: без использования указателей и с указателями.

```
#include <iostream>
int pfmin(int p, int n)
{
int pmin;
for (pmin = p; n > 0; p++, n--)
if (p < pmin) pmin = p;
return pmin;
}
void main()
{
int B[5] = {4, 8, 2, 6, 4};
(pfmin(B, 5))++;
for (int i = 0; i < 5; i++)
printf("%d ", B[i]);
}
```

```
#include <iostream>
int *pfmin(int *p, int n)
{
int *pmin;
for (pmin = p; n > 0; p++, n--)
if (*p < *pmin) pmin = p;
return pmin;
}
void main()
{
int B[5] = {4, 8, 2, 6, 4};
(*pfmin(B, 5))++;
for (int i = 0; i < 5; i++)
printf("%d ", B[i]);
}
```

В функцию можно передаватьмассив, при этом массив не копируется.

Имя массива преобразуется в указатель, и копия указателя на начало массива передается в функцию по значению.

Пример. Пусть имеется массив B. С использованием функции надо определить минимальный элемент массива и в основной программе увеличить его значение на 1. Программа в левой части выполниться не сможет. Использование указателя в правой части позволяет решить задачу.

2. Изучить использование ссылок как формальных параметров и как результатов выполнения функций, выполнив программы, записанные в правой части. Опробовать работу программы, содержащей функцию, с разными значениями массива A.

Массив A содержит набор значений. Надо определить минимальный элемент и заменить его на другое значение.

При передаче фактического параметра по ссылке передаётся адрес объекта и, соответственно, работа внутри функции происходит не с копией, а с оригиналом объекта. Чтобы параметр передавался по ссылке, достаточно в прототипе функции поставить знак & после типа параметра. Например:

```
void func1(int val, int& ref)
{ val++; ref++; }
...
int a = 10, b = 10;
func1(a, b);
cout << a << endl; // 10, значение будет увеличено, но внутри функции, как локальное
cout << b << endl; // 11, будет увеличено значение внешней переменной b.
#include <iostream>
double& dmin (double A[], int n)
{
int i, j = 0;
for (i = 1; i < n; i++)
if (A[j] > A[i]) j = i;
return A[j]; //возвращается ссылка
}
void main( )
{
double s; double A[ ] = {5, 4.1, 3, 0.2, 11};
s = dmin(A, 5);
std::cout << s;
dmin(A, 5) = 1.0; // изменить минимум на 1.0
for ( int i = 0; i < 5; i++)
std::cout << " "<<A[i];
}
```

Тема 5.6. Иерархия классов.

Массивы и ссылки при работе с функциями

Выполнить задания:

1. Выполнить программу, содержащую функцию с переменным числом параметров. Записать условие задачи.

```
#include <iostream>
int sum (int n, ...)
{ int *p = &n; int s = 0;
for (int i = 1; i <= n; i++)
s+= *(++p);
return s;
}
void main()
{ std::cout << sum(6,4,5,1,2,3,0);
}
```

При вызове функции с переменным числом параметров задается любое требуемое число аргументов.

В объявлении и определении такой функции переменное число аргументов задается многоточием в конце списка формальных параметров или списка типов аргументов. При этом должен быть указан как минимум один обычный параметр.

2. В соответствии с вариантом написать программу по условию задачи из таблицы ниже. Программа должна содержать вызывающую функцию main и функцию с переменным числом параметров, к которой должно быть не менее трех обращений с различным количеством параметров.

№ **варианта**	**Условие задачи**
1	Написать функцию **fmin** с переменным числом параметров, которая находит минимальное из чисел типа **int.**
2	Написать функцию **fsum** с переменным числом параметров, которая находит сумму чисел типа int по формуле: ***S=a1*a2+a2*a3+a3*a4+ . . . .***
3	Написать функцию **fmax** с переменным числом параметров, которая находит минимальное из чисел типа **int** или из чисел типа **double**, тип параметров определяется с помощью первого параметра функции.
4	Написать функцию **days** с переменным числом параметров, которая находит количество дней, прошедших между двумя датами (параметрами функции являются даты в формате «дд.мм.гг».
5	Написать функцию **kvadr** с переменным числом параметров, которая определяет количество чисел, являющихся точными квадратами (2, 4, 9, 16,…) типа **int.**
6	Написать функцию **as** с переменным числом параметров, которая находит сумму чисел типа **int** по формуле: ***S=a1*a2-a2*a3+a3*a4-. . . .*** .
7	Написать функцию **mn** с переменным числом параметров, которая находит минимальное из чисел типа **int** или из чисел типа **double**, тип параметров определяется с помощью первого параметра функции.
8	Написать функцию **prost** с переменным числом параметров, которая находит все простые числа из нескольких интервалов. Интервалы задаются границами **a** и **b.**
9	Во введенном тексте подсчитать количество символов в слове максимальной длины (слова разделяются пробелами) с помощью функции с переменным числом параметров.
10	Написать функцию, которая находит в строке самое первое (по алфавиту) слово. С ее помощью реализовать размещение слов в выходной строке в алфавитном порядке.
11	Написать функцию **mult** с переменным числом параметров, которая находит произведение чисел типа **float**.
12	Написать функцию, проверяющую есть ли отрицательные элементы в заданном одномерном массиве размерностью **n**. Удалить из массива все отрицательные элементы, удаленный элемент заполняется нулем и переносится в конец массива.
13	Написать функцию для вычисления суммы элементов квадратной матрицы, которые расположены ниже главной диагонали. С ее помощью найти максимальное значение такой суммы в n матрицах.

№ варианта	Условие задачи
14	Написать и протестировать функцию **compr**, которая «сжимает» строку, удаляя из нее все пробелы.
15	Написать функцию с переменным числом параметров для перевода чисел из десятичной системы счисления в двоичную.
16	Выяснить, есть ли во введенном тексте слова, начинающиеся с буквы **А**, и сколько так

5. Примерный перечень вопросов и ситуационных заданий для подготовки к промежуточной аттестации

1. Области применения языков программирования.
2. Стандарты языков программирования.
3. Среда проектирования. Компиляторы и интерпретаторы.
4. Жизненный цикл программы.
5. Программа. Программный продукт и его характеристики.
6. Основные этапы решения задач на компьютере.
7. Составление программ линейной структуры.
8. Типы данных.
9. Операции и выражения.
10. Правила формирования и вычисления выражений.
11. Структура программы.
12. Ввод и вывод данных.
13. Оператор присваивания. Составной оператор.
14. Условный оператор. Оператор выбора.
15. Цикл с постусловием. Цикл с предусловием. Цикл с параметром. Вложенные циклы.
16. Массивы. Двумерные массивы.
17. Строки. Стандартные процедуры и функции для работы со строками.
18. Структурированный тип данных – множество. Операции над множествами.
19. Комбинированный тип данных – запись. Файлы последовательного доступа. Файлы прямого доступа
20. Обработка одномерных массивов.
21. Обработка двумерных массивов.
22. Определение и вызов подпрограмм.
23. Область видимости и время жизни переменной.
24. Механизм передачи параметров.
25. Организация функций.
26. Рекурсия. Программирование рекурсивных алгоритмов.
27. Методы структурного программирования.
28. Модульное программирование. Понятие модуля. Структура модуля. Компиляция и компоновка программы.
29. Указатели. Описание указателей.
30. Основные понятия и применение динамически распределяемой памяти.
31. Создание и удаление динамических переменных.
32. Структуры данных на основе указателей.

33. Использование указателей для организации связанных списков.
34. Базовые понятия ООП: объект, его свойства и методы, класс, интерфейс.
35. Основные принципы ООП: инкапсуляция, наследование, полиморфизм.
36. Классы объектов.
37. Компоненты и их свойства.
38. Событийно-управляемая модель программирования.
39. Компонентно-ориентированный подход.
40. Требования к аппаратным и программным средствам интегрированной среды разработчика.
41. Интерфейс среды разработчика: характеристика, основные окна, инструменты, объекты. Форма и размещение на ней управляющих элементов.
42. Панель компонентов и их свойства. Окно кода проекта.
43. Состав и характеристика проекта. Выполнение проекта. Настройка среды и параметров проекта.
44. Панель компонентов и их свойства.
45. Настройка среды и параметров проекта.
46. Основные компоненты (элементы управления) интегрированной среды разработки, их состав и назначение.
47. Дополнительные элементы управления. Свойства компонентов. Виды свойств. Синтаксис определения свойств. Назначения свойств и их влияние на результат. Управление объектом через свойства.
48. События компонентов (элементов управления), их сущность и назначение.
49. Создание процедур на основе событий.
50. Разработка функционального интерфейса приложения.
51. Создание интерфейса приложения.
52. Разработка функциональной схемы работы приложения.
53. Проектирование объектно-ориентированного приложения.
54. Создание интерфейса пользователя.
55. Тестирование, отладка приложения.
56. Классы ООП: виды, назначение, свойства, методы, события.
57. Перегрузка методов.
58. Тестирование и отладка приложения.

Примерные ситуационные задания:

1. Создать процедуру, которая заполняет массив Integer значениями от 1 до 20 с использованием цикла Repeat. Телом цикла является заполнение массива, вывод в текстовый редактор значений из массива и последовательное наращивание индекса массива. Условие окончания цикла - проверка выхода заграницы массива: i=21
2. Создать процедуру, которая заполняет массив Integer значениями от 1 до 20 с использованием цикла While. Телом цикла является заполнение массива, вывод в текстовый редактор значений из массива и последовательное наращивание индекса массива. Условие продолжения цикла - проверка границы массива: i<21.
3. Создать процедуру, которая показывает Ваше знание математических операций.
   - вычесть из одного числа другое;
   - произвести целочисленное деление
   - вычислить выражение – (21+6)/2*(12-5)
   - разделить 2 числа

Данные для операций можно задать через константы или непосредственно в операциях.

4. Создать процедуру. Даны три действительных числа x, y, z.
   Получить Max (x, y, z).
5. Создать процедуру, которая показывает Ваше знание математических операций.
   - сложить 2 вещественных числа;
   - определить остаток от деления 2 целых чисел
   - вычислить выражение – 4(10-3):(2+5)
   - умножить 2 вещественных числа

   Данные для операций можно задать через константы или непосредственно в операциях.
6. Создать процедуру, которая сравнивает два числа, введенных с клавиатуры. Программа должна указать какое число больше, или, если числа равны, вывести соответствующее сообщение.
7. Создать процедуру, которая проверяет является ли четным введенное пользователем целое число.
8. Создать процедуру, которая вычисляет частное от деления двух чисел. Программа должна проверять правильность введённых пользователем данных и, если они не верные (делитель равен 0), выдавать сообщения об ошибке.
9. Создать процедуру, которая используя массив констант, состоящий из 10 целых элементов, вычисляет сумму отрицательных элементов массива, массив обрабатывается через цикл For.
10. Создать процедуру, которая заполняет массив integer значениями от 20 до 1 с использованием цикла For. Телом цикла является заполнение массива, вывод в текстовый редактор значений из массива.
11. Создать процедуру, которая используя массив констант, состоящий из 10 вещественных элементов, вычисляет сумму отрицательных элементов массива, массив обрабатывается через цикл For.
12. Создать процедуру, которая находит сумму целых положительных чисел меньших 100, с использованием цикла While. Телом цикла является проверка счетчика цикла на чётность, если он чётный, собираем его в сумму, последовательно наращиваем счётчик цикла.
13. Создать процедуру, которая используя массив констант, состоящий из 12 целочисленных элементов, вычисляет произведение элементов массива с чётными номерами, массив обрабатывается через цикл For.
14. Создать процедуру, которая находит сумму целых положительных чисел, кратных 4 и меньших 100, с использованием цикла While. Телом цикла является проверка счётчика цикла на кратность 4, если он кратен 4, собираем его в сумму, последовательно наращиваем счётчик цикла.
15. Создать процедуру, которая находит сумму целых положительных чисел больших 20, меньших 100 и кратных 3, с использованием цикла While. Телом цикла является проверка счётчика цикла на кратность 3, если он кратен 3, собираем его в сумму, последовательно наращиваем счётчик цикла.
16. Создать процедуру, которая используя двумерный массив констант целого типа размерностью 3*3. Выводит на экран элементы массива и их произведение. Использовать вложенные циклы for.
17. Создать процедуру, которая используя двумерный массив констант целого типа размерностью на 4*2. Выводит на экран элементы массива и их сумму. Использовать вложенные циклы For.

18. Создать процедуру, которая, используя двумерный массив констант целого типа размерностью 4*4. Выводит на экран элементы массива, их сумму, и произведение. Использовать вложенные циклы For.

Критерии оценки результатов освоения учебной дисциплины:

«Отлично»

– теоретическое содержание курса освоено полностью, без пробелов, умения сформированы, все предусмотренные программой учебные задания выполнены, качество их выполнения оценено высоко.

«Хорошо»

– теоретическое содержание курса освоено полностью, без пробелов, некоторые умения сформированы недостаточно, все предусмотренные программой учебные задания выполнены, некоторые виды заданий выполнены с ошибками.

«Удовлетворительно»

– теоретическое содержание курса освоено частично, но пробелы не носят существенного характера, необходимые умения работы с освоенным материалом в основном сформированы, большинство предусмотренных программой обучения учебных заданий выполнено, некоторые из выполненных заданий содержат ошибки.

«Неудовлетворительно»

– теоретическое содержание курса не освоено, необходимые умения не сформированы, выполненные учебные задания содержат грубые ошибки.

Преподаватель _____Далгатова Я.А.