МИНИСТЕРСТВО ТРАНСПОРТА РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ АГЕНТСТВО ЖЕЛЕЗНОДОРОЖНОГО ТРАНСПОРТА
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
САМАРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ПУТЕЙ СООБЩЕНИЯ

Приложение
к рабочей программе дисциплины

ОЦЕНОЧНЫЕ МАТЕРИАЛЫ ДЛЯ ПРОМЕЖУТОЧНОЙ АТТЕСТАЦИИ ПО ДИСЦИПЛИНЕ (МОДУЛЮ)

Алгоритмизация и программирование

(наименование дисциплины(модуля)

Направление подготовки / специальность

09.03.02 Информационные системы и технологии

(код и наименование)

Направленность (профиль)/специализация

Информационные системы и технологии на транспорте

(наименование)

Содержание

1. Пояснительная записка

Цель промежуточной аттестации – оценивание промежуточных и окончательных результатов обучения по дисциплине, обеспечивающих достижение планируемых результатов освоения образовательной программы.

Формы промежуточной аттестации: зачет – **1 семестр**, контрольная работа, экзамен**- 2 семестр**

Код и наименование компетенции	Код достижения индикатора компетенции
ОПК-6: Способен разрабатывать алгоритмы и программы, пригодные для практического применения в области информационных систем и технологий;	ОПК-6.1: Разрабатывает алгоритмы и программы, пригодные для практического применения в области информационных систем и технологий
	ОПК-6.2: Использует технологию программирования для написания программ, пригодных для практического применения

Результаты обучения по дисциплине, соотнесенные с планируемыми результатами освоения образовательной программы

Код и наименование индикатора достижения компетенции	Результаты обучения по дисциплине	Оценочные материалы
ОПК-6.1: Разрабатывает алгоритмы и программы, пригодные для практического применения в области информационных систем и технологий	Обучающийся знает: Основные структуры данных и методы их обработки, различия между языками программирования высокого и низкого уровня	Вопросы тестирования№(1-10)
	Обучающийся умеет: программировать алгоритмы, используя средства языков высокого уровня, разрабатывать тестовые случаи и сценарии.	Задания №(1-3)
	Обучающийся владеет: навыками алгоритмизации и программной реализации на языке высокого уровня решений практических задач	Задания №4
ОПК-6.2: Использует технологию программирования для написания программ, пригодных для практического применения	Обучающийся знает: язык программирования Си, набор функций стандартных библиотек.	Вопросы тестирования №(11-26)
	Обучающийся умеет: формализовать прикладную задачу, выбирать для неё подходящие структуры данных и алгоритмы обработки	Задания №5
	Обучающийся владеет: документирования исходного кода;	Задания №(6-7)

Промежуточная аттестация (зачет) проводится в одной из следующих форм:

1) проводиться в форме устного ответа на вопросы из перечня
2) выполнение заданий в ЭИОС СамГУПС.

Промежуточная аттестация (экзамен) проводится в одной из следующих форм:
1) проводиться в форме устного ответа на вопросы из перечня
2) выполнение заданий в ЭИОС СамГУПС.

Промежуточная аттестация (курсовая работа) проводится в одной из следующих форм:
1) Собеседование
2) выполнение заданий в ЭИОС СамГУПС.

2. Типовые[1] контрольные задания или иные материалы для оценки знаний, умений, навыков и (или) опыта деятельности, характеризующих уровень сформированности компетенций

2.1 Типовые вопросы (тестовые задания) для оценки знаниевого образовательного результата

Проверяемый образовательный результат:

Код и наименование индикатора достижения компетенции	Образовательный результат
ОПК-6.1: Разрабатывает алгоритмы и программы, пригодные для практического применения в области информационных систем и технологий	Обучающийся знает: Основные структуры данных и методы их обработки, различия между языками программирования высокого и низкого уровня
ОПК-6.2: Использует технологию программирования для написания программ, пригодных для практического применения	Обучающийся знает: язык программирования Си, набор функций стандартных библиотек.

1. Класс - это:

- любой тип данных, определяемый пользователем

- * тип данных, определяемый пользователем и сочетающий в себе данные и функции их обработки

- структура, для которой в программе имеются функции работы с нею

2.Тест. Членами класса могут быть

- * как переменные, так и функции, могут быть объявлены как private и как public

[1] Приводятся типовые вопросы и задания. Оценочные средства, предназначенные для проведения аттестационного мероприятия, хранятся на кафедре в достаточном для проведения оценочных процедур количестве вариантов. Оценочные средства подлежат актуализации с учетом развития науки, образования, культуры, экономики, техники, технологий и социальной сферы. Ответственность за нераспространение содержания оценочных средств среди обучающихся университета несут заведующий кафедрой и преподаватель – разработчик оценочных средств.

- только переменные, объявленные как private

- только функции, объявленные как private

- только переменные и функции, объявленные как private

- только переменные и функции, объявленные как public

3. Что называется конструктором?

- * метод, имя которого совпадает с именем класса и который вызывается автоматически при создании объекта класса

- метод, имя которого совпадает с именем класса и который вызывается автоматически при объявлении класса (до создания объекта класса)

- метод, имя которого необязательно совпадает с именем класса и который вызывается при создании объекта класса

- метод, имя которого совпадает с именем класса и который необходимо явно вызывать из головной программы при объявлении объекта класса

4. Объект - это

- переменная, содержащая указатель на класс

- * экземпляр класса

- класс, который содержит в себе данные и методы их обработки

5. Отметьте правильные утверждения

- * конструкторы класса не наследуются

- конструкторов класса может быть несколько, их синтаксис определяется программистом

- * конструкторов класса может быть несколько, но их синтаксис должен подчиняться правилам перегрузки функций

- конструктор возвращает указатель на объект

- * конструктор не возвращает значение

6. Что называется деструктором?

- метод, который уничтожает объект

- метод, который удаляет объект

- * метод, который освобождает память, занимаемую объектом

- системная функция, которая освобождает память, занимаемую объектом

7. Выберите правильные утверждения

- * у конструктора могут быть параметры

- конструктор наследуется, но должен быть перегружен

- конструктор должен явно вызываться всегда перед объявлением объекта

- * конструктор вызывается автоматически при объявлении объекта

- объявление каждого класса должно содержать свой конструктор

- * если конструктор не создан, компилятор создаст его автоматически

8. Выберите правильные утверждения

- деструктор - это метод класса, применяемый для удаления объекта

- * деструктор - это метод класса, применяемый для освобождения памяти, занимаемой объектом

- деструктор - это отдельная функция головной программы, применяемая для освобождения памяти, занимаемой объектом

- * деструктор не наследуется

- деструктор наследуется, но должен быть перегружен

Тест - 9. Что называется наследованием?

- * это механизм, посредством которого производный класс получает элементы родительского и может дополнять либо изменять их свойства и методы

- это механизм переопределения методов базового класса

- это механизм, посредством которого производный класс получает все поля базового класса

- это механизм, посредством которого производный класс получает элементы родительского, может их дополнить, но не может переопределить

10. Выберите правильное объявление производного класса

- class MoreDetails:: Details;

- class MoreDetails: public class Details;

- * class MoreDetails: public Details;

- class MoreDetails: class(Details);

11. Выберите правильные утверждения:

- если элементы класса объявлены как private, то они доступны только наследникам класса, но не внешним функциям

- * если элементы класса объявлены как private, то они недоступны ни наследникам класса, ни внешним функциям

- если элементы объявлены как public, то они доступны наследникам класса, но не внешним функциям

- * если элементы объявлены как public, то они доступны и наследникам класса, и внешним функциям

12. Возможность и способ обращения производного класса к элементам базового определяется

- ключами доступа: private, public, protected в теле производного класса

- только ключом доступа protected в заголовке объявления производного класса

- * ключами доступа: private, public, protected в заголовке объявления производного класса

- ключами доступа: private, public, protected в теле базового класса

13. Выберите правильные соответствия между спецификатором базового класса, ключом доступа в объявлении производного класса и правами доступа прозводного класса к элементам базового

- ключ доступа - public; в базовом классе: private; права доступа в производном классе - protected

- * ключ доступа - любой; в базовом классе: private; права доступа в производном классе - нет прав

- * ключ доступа - protected или public ; в базовом классе: protected; права доступа в производном классе - protected

- ключ доступа - private; в базовом классе: public; права доступа в производном классе - public

- * ключ доступа – любой; в базовом классе: public; права доступа в производном

классе – такие же, как ключ доступа

14. Дружественная функция - это

- функция другого класса, среди аргументов которой есть элементы данного класса

- * функция, объявленная в классе с атрибутом friend, но не являющаяся членом класса;

- функция, являющаяся членом класса и объявленная с атрибутом friend;

- функция, которая в другом классе объявлена как дружественная данному

15. Выберите правильные утверждения:

- * одна функция может быть дружественной нескольким классам

- дружественная функция не может быть обычной функцией, а только методом другого класса

- * дружественная функция объявляется внутри класса, к элементам которого ей нужен доступ

- дружественная функция не может быть методом другого класса

16. Шаблон функции - это...

- * определение функции, в которой типу обрабатываемых данных присвоено условное обозначение

- прототип функции, в котором вместо имен параметров указан условный тип

- определение функции, в котором указаны возможные варианты типов обрабатываемых параметров

- определение функции, в котором в прототипе указан условный тип, а в определении указаны варианты типов обрабатываемых параметров

17 Выберите правильные утверждения:

- * по умолчанию члены класса имеют атрибут private

- по умолчанию члены класса имеют атрибут public;

- члены класса имеют доступ только к элементам public;

- * элементы класса с атрибутом private доступны только членам класса

18. Переопределение операций имеет вид:

- имя_класса, ключевое слово operation, символ операции

- * имя_класса, ключевое слово operator, символ операции, в круглых скобках могут быть указаны аргументы

- имя_класса, ключевое слово operator, список аргументов

- имя_класса, два двоеточия, ключевое слово operator, символ операции

Тест - 19. Для доступа к элементам объекта используются:

- * при обращении через имя объекта – точка, при обращении через указатель – операция «->»

- при обращении через имя объекта – два двоеточия, при обращении через указатель – операция «точка»

- при обращении через имя объекта – точка, при обращении через указатель – два двоеточия

- при обращении через имя объекта – два двоеточия, при обращении через указатель – операция «->»

20. Полиморфизм – это :

- * средство, позволяющее использовать одно имя для обозначения действий, общих для родственных классов

- средство, позволяющее в одном классе использовать методы с одинаковыми именами;

- средство, позволяющее в одном классе использовать методы с разными именами для выполнения одинаковых действий

- средство, позволяющее перегружать функции для работы с разными типами или разным количеством аргументов.

21. Полиморфизм реализован через механизмы:

- * перегрузки функций, виртуальных функций, шаблонов

- перегрузки функций, наследования методов, шаблонов;

- наследования методов, виртуальных функций, шаблонов

- перегрузки функций, наследования, виртуальных функций.

22. Виртуальными называются функции:

- * функции базового класса, которые могут быть переопределены в производном

классе

- функции базового класса, которые не используются в производном классе;

- функции базового класса, которые не могут быть переопределены в базовом классе;

- функции производного класса, переопределенные относительно базового класса

23. Выберите правильный вариант выделения динамической памяти под переменную X типа float:

- * float *ptr = new float; X = *ptr;

- float & ptr = new float; X = & ptr;

- float * ptr = &X; X = new float;

24. Полиморфизм в объектно-ориентированном программировании реализуется:

- * через механизмы перегрузки (функций и операций), виртуальные функции и шаблоны

- через механизмы перегрузки (функций и операций) и шаблоны;

- через виртуальные функции и шаблоны;

- через механизмы перегрузки (функций и операций) и виртуальные функции

25. Дано определение класса

```
class monstr {

int health, armo;

monstr(int he, int arm);

public:

monstr(int he=50, int arm=10);

int color;

}
```

26.Тест. Укажите свойства и методы, доступные внешним функциям

- health, armo
monstr(int he, int arm);
monstr(int he=50, int arm=10);

- * int color;
monstr(int he=50, int arm=10);

- health, armo, color
monstr(int he=50, int arm=10);

- int color;
monstr(int he, int arm);

2.2 Типовые задания для оценки навыкового образовательного результата

Проверяемый образовательный результат:

Код и наименование индикатора достижения компетенции	Образовательный результат
ОПК-6.1: Разрабатывает алгоритмы и программы, пригодные для практического применения в области информационных систем и технологий	Обучающийся умеет: программировать алгоритмы, используя средства языков высокого уровня, разрабатывать тестовые случаи и сценарии.
	Обучающийся владеет: навыками алгоритмизации и программной реализации на языке высокого уровня решений практических задач
ОПК-6.2: Использует технологию программирования для написания программ, пригодных для практического применения	Обучающийся умеет: формализовать прикладную задачу, выбирать для неё подходящие структуры данных и алгоритмы обработки
	Обучающийся владеет: документирования исходного кода;

Работа № 1

Тема: "Программирование алгоритмов линейной структуры"

Цель работы: изучение основных типов данных, способов описания переменных различных типов, операторов присваивания и организации ввода – вывода.

Краткие теоретические сведения:

Алгоритм – это последовательность действий, выполняемых по строго определенным правилам, однозначно определяющая процесс решения задачи и заведомо приводящая к её решению за некоторое количество шагов. Алгоритмизация – разработка формального метода решения практической задачи с возможностью реализации в виде программы для ЭВМ.

Изображение алгоритма в виде схемы выполняется в соответствии с ГОСТ 19.701–90 Единой системы программной документации «Схемы алгоритмов, программ, данных и систем».

Структура программ для Microsoft Visual Studio.

```
// struct_program.cpp: определяет точку входа для консольного
1  приложения.
2  #include "stdafx.h"
```

```
3 //здесь подключаем все необходимые препроцессорные
4 директивы
5 int main() { // начало главной функции с именем main
6 //здесь будет находится ваш программный код
  }
```

В строке 1 говорится о точке входа для консольного приложения, это значит, что данную программу можно запустить через командную строку Windows указав имя программы, к примеру, такое struct_program.cpp. Строка 1 является однострочным комментарием, так как начинается с символов //. В строке 2 подключен заголовочный файл "stdafx.h". Данный файл похож на контейнер, так как в нем подключены основные препроцессорные директивы (те, что подключил компилятор, при создании консольного приложения), и вспомогательные (подключенные программистом).

include — директива препроцессора, т. е. сообщение препроцессору. Строки, начинающиеся с символа # обрабатываются препроцессором до компиляции программы.

заголовочные файлы:

Библиотека cmath определяет набор функций для выполнения общих математических операций и преобразований. Математические функции:

1) тригонометрические функции:

cos-вычисление косинуса угла, переведенного в радианы; sin-вычисление синуса угла, переведенного в радианы; tan-вычисление тангенса угла, переведенного в радианы;

acos-вычисление арккосинуса, результат будет в радианах; asin-вычисление арксинуса, результат будет в радианах;

atan-вычисление арктангенса, возвращаемый результат будет в радианах; atan2-вычисление арктангенса и квадранта по координатам x и

y, возвращаемый результат будет в радианах;

2) гиперболические функции:

cosh-вычисление гиперболического косинуса; sinh-вычисление гиперболического синуса; tanh-вычисление гиперболического тангенса;

3) экспоненциальные и логарифмические функции: exp-вычисление экспоненты;

frexp-получить мантиссу и показатель степени двойки;

ldexp-генерация числа по значению мантиссы и показателю степени; log-натуральный логарифм;

log10-десятичный логарифм;

modf-разделение вещественного значения на дробную и целую части;

4) функции степени:

pow-возведение числа в степень. Пример использования функции; sqrt-корень квадратный;

5) округление, модуль и другие функции

ceil-округление до наименьшего целого значения; fabs-вычислить модуль значения;
floor-округление до наибольшего целого значения; fmod-остаток от деления числителя на знаменатель.

Пример использования функций:

```
#include <iostream> // для оператора
cout #include <cmath>     // для функции
pow


int main()

{

  std::cout << "5.0 ^ 4 = "       << pow (5.0, 4)       << std::endl;

  std::cout << "2.77 ^ 9 = "     << pow (2.77, 9)     << std::endl;

  std::cout << "12.01 ^ 11.54 = " << pow (12.01, 11.54) << std::endl;
  std::cout << "sqrt(" << param << ") = "

          << sqrt(param)                // вычисляем корень квадратный

          << std::endl;
```

```
double param = 60.0;          // угол 60
 градусов std::cout << "Косинус "
                                  << param

          << " градусов = " << cos(param * PI / 180)  // вычисляем косинус угла,

               //переведённого в радианы

          << std::endl;
double param =
0.5;

 std::cout << "Арксинус " << param

          << " = " << (asin (param) * 180.0 / PI) // вычисляем арксинус

          << " градусов " << std::endl;
double val = 5.5, result;

 result = log (val);                  // вычисляем натуральный
 логарифм std::cout << "ln(" << val << ") = "

          << result << std::endl;
 return 0;
```

}

Файлы кода C++ (с расширением .cpp) не являются единственными файлами в проектах и программах. Есть еще один тип файлов, который называется **заголовочный файл** (файл заголовка, подключаемый файл или header file**)**. Они имеют расширение .h, но иногда их можно увидеть и с расширением .hpp или вообще без расширения. Целью заголовочных файлов является удобное хранение предварительных объявлений для использования другими файлами. Всё содержимое из заголовочного файла копируется в файл *.cpp, т.е. всё содержимое становится доступным для использования.

Заголовочные файлы:

— cstdio (stdio.h)-заголовочный файл для выполнения операций ввода/вывода;
— cstring (string.h)-заголовочный файл для работы со строками;
— iostream-заголовочный файл с классами, функциями и переменными для организации ввода-вывода. Для удобства в библиотеке определены три стандартных объекта-потока:

cin – объект класса istream, соответствующий стандартному вводу. В общем случае он позволяет читать данные с терминала пользователя;

cout – объект класса ostream, соответствующий стандартному выводу. В общем случае он позволяет выводить данные на терминал пользователя;

cerr – объект класса ostream, соответствующий стандартному выводу для ошибок. В этот поток мы направляем сообщения об ошибках программы.

Вывод осуществляется, как правило, с помощью перегруженного оператора сдвига влево (<<), а ввод – с помощью оператора сдвига вправо (>>).

Основные типы данных представлены в таблице 1.

Таблица 1 — Типы данных C++

Тип	байт	Диапазон принимаемых значений
целочисленный (логический) тип данных		
bool	1	0 / 255
целочисленный (символьный) тип данных		
char	1	0 / 255
целочисленные типы данных		
short int	2	-32 768 / 32 767
unsigned short int	2	0 / 65 535
int	4	-2 147 483 648 / 2 147 483 647
unsigned int	4	0 / 4 294 967 295
long int	4	-2 147 483 648 / 2 147 483 647
unsigned long int	4	0 / 4 294 967 295
типы данных с плавающей точкой		
float	4	-2 147 483 648.0 / 2 147 483 647.0
long float	8	-9 223 372 036 854 775 808 .0/ 9 223 372 036 854 775 807.0

double	8	-9 223 372 036 854 775 808 .0 / 9 223 372 036 854 775 807.0

Способы ввода данных в языке возможно двумя способами: форматированные ввод-вывод или потоковый.

При форматированном способе используются операторы ввода scanf вывода printf. Синтаксис операторов имеет вид:

scanf(<строка описания форматов> [, <список ввода>]); printf(<строка описания форматов> [, <список вывода>]);

Строка описания форматов состоит из обычных символов, специальных управляющих последовательностей символов и спецификаций формата. Обычные символы и управляющие последовательности просто копируются в стандартный выходной поток в порядке их появления. Спецификации формата начинаются с символа % и заканчиваются символом, определяющим тип выводимого значения. Кроме того, спецификации формата могут содержать символы и цифры для управления видом выводимого значения (подробно см. ниже). Список вывода состоит из переменных и/или констант, значения которых должны быть выведены. Количество спецификаций формата должно быть равно количеству выводимых значений, которые указываются в списке вывода.

К управляющим последовательностям относятся последовательности символов, представленных в таблице 2.

Таблица 2 — Управляющие символы

Последовательность	Действие
\a	Звуковой сигнал
\b	Удаление предыдущего символа
\n	Новая строка
\r	Возврат каретки
\t	Горизонтальная табуляция
\v	Вертикальная табуляция
\'	Апостроф
\"	Кавычки
\\	Обратный слеш
ooo	ASCII символ в восьмеричной нотации
\x*ooo*	ASCII символ в шестнадцатеричной нотации

Иногда при работе операторов используются спецификаторы форматов (таблица 3-5).

Таблица 3 — Спецификаторы формата для оператора printf

Символ	Назначение
%c	символ
%d	целое десятичное число
%i	целое десятичное число
%e	десятичное число в виде x.xx e+xx
%E	десятичное число в виде x.xx E+xx
%f	десятичное число с плавающей запятой xx.xxxx
%F	десятичное число с плавающей запятой

	xx.xxxx
%g	%f или %e, что короче
%G	%F или %E, что короче
%o	восьмеричное число
%s	строка символов
%u	беззнаковое десятичное число
%x	шестнадцатеричное число
%X	шестнадцатеричное число
%%	символ %
%p	указатель
%n	указатель

Кроме того, к командам формата могут быть применены модификаторы l и h (таблица 4).

Таблица 4

Обозначение	Назначение
%ld	печать long int
%hu	печать short unsigned
%Lf	печать long double

Таблица 5 — Спецификаторы формата для оператора scanf

Символы	Назначение
%c	чтение символа
%d	чтение десятичного целого
%i	чтение десятичного целого
%e	чтение числа типа float (плавающая запятая)
%h	чтение short int
%o	чтение восьмеричного числа
%s	чтение строки
%x	чтение шестнадцатеричного числа
%p	чтение указателя
%n	чтение указателя в увеличенном формате

Примеры использования спецификаторов и модификаторов в операторах ввода-вывода:

```
int    m, n, x;
double y;
char  c = '&';

char  str[] = "String";
```

```
scanf("%d%d", &m, &n);        // Ввод десятичных целых чисел в переменные m и n
```

```
printf("m = %5d\nn = %5d\n", m, n);          // Вывод переменных m и n в десятичном целом формате, используются как минимум 5 знаков

scanf("%d", &x);        // Ввод десятичного целого числа в переменную x

printf("%#010x\n", x);          // Вывод переменной x в шестнадцатеричной системе, используются 10 знаков,

// впереди добавляются нули и символы 0x

scanf("%lf", &y);        // Ввод вещественного числа в переменную y

printf("y = %7.2lf\n", y);        // Вывод вещественной переменной, используются как минимум 7 знаков, из них 2 – после точки

printf("c = %c\n", c);          // Вывод одного
символа printf("%.4s\n", str);

}
```

Составим программу на языке С++ для умножения двух чисел.

```
//Подключение стандартных библиотек:

#include<iostream>

#include <math.h>  // для работы с мат.функциями acos, log10, log, atan

using namespace std; // пространство имен, где определяются идентификаторы

void main()                    //заголовок главной функции программы

{                          //операторные скобки

  setlocale(0, “Russian”); //установка вывода сообщений на русском
  языке float x,y,z;                  //описание переменных (имя и тип)
  system(“cls”);              //функция очистки экрана

  cout<<"Введите числа x,y: "<<endl;  //вывод сообщения на экран

  cin>>x>>y;                 //считать 2 вещ.числа, записать их по адресу перем. x,y

  z=x*y;

  cout<<"Произведение="<<z;            //вывести результат на экран

 system(“pause”);

}                                                  //конец главной функции
```

Пример выполнения задания.

Задание: вычислить значение функции $y = \frac{\sin^3 c \cdot \cos^2 a}{2} +$

$5\sin^d b$ 15

при

$A = 9{,}5$; $B = 1{,}365$; $C = 6{,}5$; $D = 5$. Использовать два варианта ввода исходных данных и вывода результатов: возможности библиотеки функций языка С и библиотеки классов языка С++.

Схема программы для задания представлена на рисунке 3:

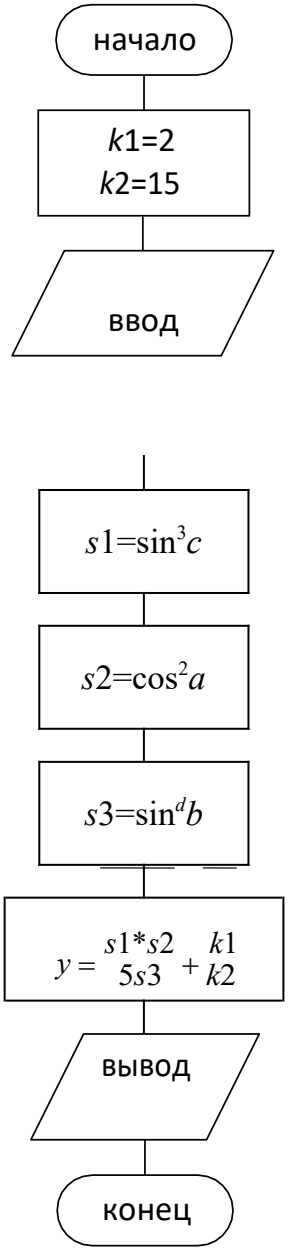


Рисунок 3

Текст программы с использованием библиотеки функций языка С (форматированный ввод-вывод) для ввода исходных данных и вывода результатов:

```
//Лабораторная работа 1

//с использованием библиотеки классов языка С++

// форматированный ввод-
вывод # include <iostream>

# include <math.h>
using namespace
std; void main()

{
```

```
setlocale(0, “Russian”); //Вывод сообщений на русском языке
const double k1=2.0;

const double k2= 15.0;
double a, b, c, d, s1, s2, s3, y;

system(“cls”);                    //Очистка экрана
printf(“Введите значения переменных a, b, c, d:\
n”); scanf(“%lf%lf%lf%lf”, &a, &b, &c, &d);
s1=pow(sin(c), 3);

s2=pow(cos(a), 2 );
```

```
s3=pow(sin(b), d );
y=(s1*s2)/(5*s3) + k1/k2;

printf(“Искомое значение y=%lf”, y);
system(“pause”);                    //Пауза

}
```

Текст программы с использованием библиотеки классов языка С++ (потоковый ввод-вывод) для ввода исходных данных и вывода результатов:

```
//Лабораторная работа 1

//с использованием библиотеки классов языка С++

// потоковый ввод-
вывод # include
<iostream>

# include <math.h>
using namespace
std; void main()

{

setlocale(0, “Russian”); //Вывод сообщений на русском языке
const double k1= 2.0;

const double k2= 15.0;
double a, b, c, d, s1, s2, s3, y;

system(“cls”);                //Очистка экрана
cout<<“Введите значения переменных a, b, c, d:”<<endl;
cin>>a>>b>>c>>d;

s1=pow(sin(c), 3);

s2=pow(cos(a), 2 );

s3=pow(sin(b), d  );
y=(s1*s2)/(5*s3) + k1/k2;
cout<<“Искомое значение y=”<<y;
system(“pause”);        //Пауза

}
```

Контрольные вопросы

1. Структура программы на языке С++

2. Директивы препроцессора, заголовочные файлы, прототипы библиотечных функций, их вызовы.

3. Этапы обработки текста программы. Включение текстов из заголовочных файлов.

4. Главная функция программы. Структура функции, ее заголовок.

5. Определение переменных в программе. Типы переменных.

6. Функции форматного ввода-вывода в стиле С.

7. Ввод-вывод данных потоком в стиле С++.

8. Операции, выражения. Оператор присваивания.

Варианты заданий

Вычислить выражение, использовать два варианта ввода исходных данных и вывода результатов: возможности библиотеки функций языка С и библиотеки классов языка С++:

1. $y = \arccos\left(\frac{A}{B} - B\right) - \frac{B}{\left|\left(\frac{B}{}\right)\right|^2} \cdot \lg\left(A^2 + B^2\right)$ при $A = 5, B = 2{,}35$ ·

2. $\left(A+B\right)^2$ при $A=6{,}84;\ B=3{,}22;\ C=4;\ D=2{,}5$.

$Y=\left|\overline{}\right|+B$ (под корнем $C+D$)

3. $Y=\lg\left(\sqrt{B}+\mathrm{tg}A\right)$ при $A=4;\ B=13{,}6$.

4. $Y=\dfrac{\sin^3 c\cdot\cos^2 d}{5\sin b}$ при $A=9.5;\ B=1{,}365;\ C=6{,}6;\ D=3$.

5. $Y=2A\cdot\arcsin\left(3{,}14\right)$ при $A=8;\ B=5{,}6$.

$\left|\overline{}\right|$ (под корнем B)

6. $Y=\dfrac{2B\cdot\ln(A-B)}{D^2}$ при $A=5{,}6;\ B=2{,}8;\ D=3$.

7. $Y=2\left|\dfrac{A^D+4C^2}{\sqrt{3}}\right|$ при $A=5{,}95;\ C=3{,}6;\ D=-3$.

8. $Y=0{,}5\lg\dfrac{1+\sin B}{1-\sin A}\cdot(C-D)$ при $A=3{,}14;\ B=-1{,}57;\ C=10.5;\ D=8$.

9. $Y=\dfrac{1}{2}\cdot\left|\dfrac{B}{10}\right|^3\cdot(C+D)^2$ при $A=2{,}6;\ B=56{,}6;\ C=35;\ D=-20{,}3$.

10. $\left(A/B-1\right)^2$ при $A=-6;\ B=1{,}64;\ C=0{,}16;\ D=0{,}8$.

$Y=\left|\overline{}\right|$ (под корнем $C/D-1$)

11. $Y=\dfrac{A+B}{C+\dfrac{D}{A+C}}$ при $A=25;\ B=8{,}5;\ C=0{,}56;\ D=0{,}01$.

12. $Y=\sin(A+B+C+D)^2-\sqrt{tg(A-C)}$ при $A=8{,}4;\ B=-6{,}4;\ C=4;\ D=-7$.

13. $Y=\sqrt{(A+B+C)^2-(A-B-C)^2}$ при $A=25;\ B=8;\ C=12{,}5$.

14. $Y = \dfrac{A^2 + B^2 + C^2}{\ln(A \cdot B \cdot C)}$ при $A = 0{,}6;\ B = 0{,}5;\ C = 6$.

15. $Y = A \cdot \dfrac{D^3}{3} + B \cdot \dfrac{C^2}{2}$ при $A = 0{,}3;\ B = 2{,}8;\ C = -4{,}5;\ D = 1{,}35$.

16. $Y = 2\sin(3\pi - 2\alpha)\cos^2(5\pi + 2\alpha)$ при $\alpha = 1{,}57$.

17. $Y = 2\left(2\cos\alpha \cdot \sin\left(\dfrac{\pi}{2\alpha} + \right.\right.$ при $\alpha = 2{,}09$.

$\sqrt{\quad}\ \left|\ \dfrac{\quad}{4}\ \right|$

18. $Y = \dfrac{\sin 2\alpha + \sin 5\alpha - \sin 3\alpha}{\cos\alpha + 1 - 2\sin^2 2\alpha}$ при $\alpha = 4{,}71$.

19. $Y = 1 - \dfrac{1}{2\alpha}\sin^2 2\alpha + \cos\dfrac{\alpha}{4}$ при $\alpha = 1{,}57$.

20. $Y = 4\cos\dfrac{\alpha}{2} \cdot \cos\dfrac{5}{2}\alpha \cdot \cos 4\alpha$ при $\alpha = 2{,}09$.

21. $Y = \arccos\left(\dfrac{A}{B}^2 - B\right) - \dfrac{B}{2} \cdot \lg\left(A^2 + B^2\right)$ при $A = 5,\ B = 2{,}35$.

$\left|\ \dfrac{\quad}{B}\ \right|$

22. $\left(\dfrac{A + B}{C + D}\right)^2 + B$ при $A = 6{,}84;\ B = 3{,}22;\ C = 4;\ D = 2{,}5$.

23. $Y = \lg\left(\sqrt{B} + \operatorname{tg}A\right)$ при $A = 4;\ B = 13{,}6$.

24. $Y = \dfrac{\sin^3 c \cdot \cos^2 d}{5\sin b}$ при $A = 9.5;\ B = 1{,}365;\ C = 6{,}6;\ D = 3$.

25. $Y = 2A \cdot \arcsin\left(\dfrac{3{,}14}{B}\right)$ при $A = 8;\ B = 5{,}6$.

$\left|\ \dfrac{\quad}{B}\ \right|$

Работа № 2

Тема: "Программирование алгоритмов разветвленной структуры"

Цель работы – изучение условного оператора и приобретение навыков программирования разветвляющихся алгоритмов.

Краткие теоретические сведения.

Синтаксис записи условного оператора if else:

— сокращенная запись: если условие истинно, т.е. выполняется, то выполняется и тело оператора выбора, иначе выполняется оператор, стоящий следом за if, т.е. оператор n. Графически запись представлена на рисунке 4.

```
if (/*проверяемое условие*/)
{
  /*тело оператора выбора 1*/;
}
оператор n;
```

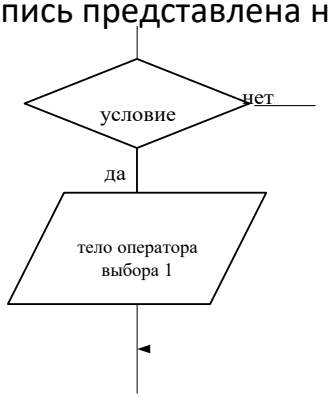


Рисунок 4

— полная запись: если проверяемое условие истинно, то выполняется тело оператора выбора 1, иначе, т. е. проверяемое условие ложно, выполняется тело оператора выбора 2. Графически запись представлена на рисунке 5.

```
if (/*проверяемое условие*/)
{
  /*тело оператора выбора 1*/;
} else
    {
      /*тело оператора выбора 2*/;
    }
```

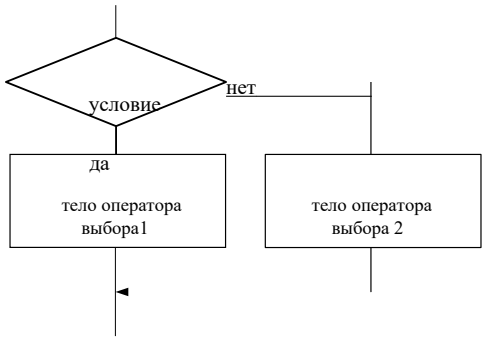


Рисунок 5

Пример выполнения задания.

Задание: по заданным координатам *x* и *y* определить, где находится точка (рисунок 6): внутри заштрихованной области; вне заштрихованной области; на границе этой области.

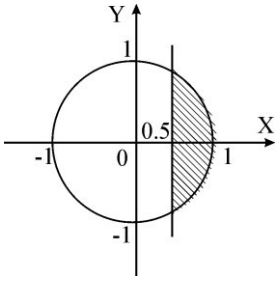


Рисунок 6 — Постановка задачи Метод решения задачи:

1. Для решения задачи будем использовать уравнение окружности $x^2+y^2=R^2$. Так как R=1, то уравнение принимает вид $x^2+y^2=1$.

2. Определяем условие, при котором точка будет находиться внутри заштрихованной области: (x>0.5) и (x^2+y^2<1).

3. Определяем условие, при котором точка будет находиться вне заштрихованной области: (x<0.5) или (x^2+y^2>1).

Схема программы представлена на рисунке 7.

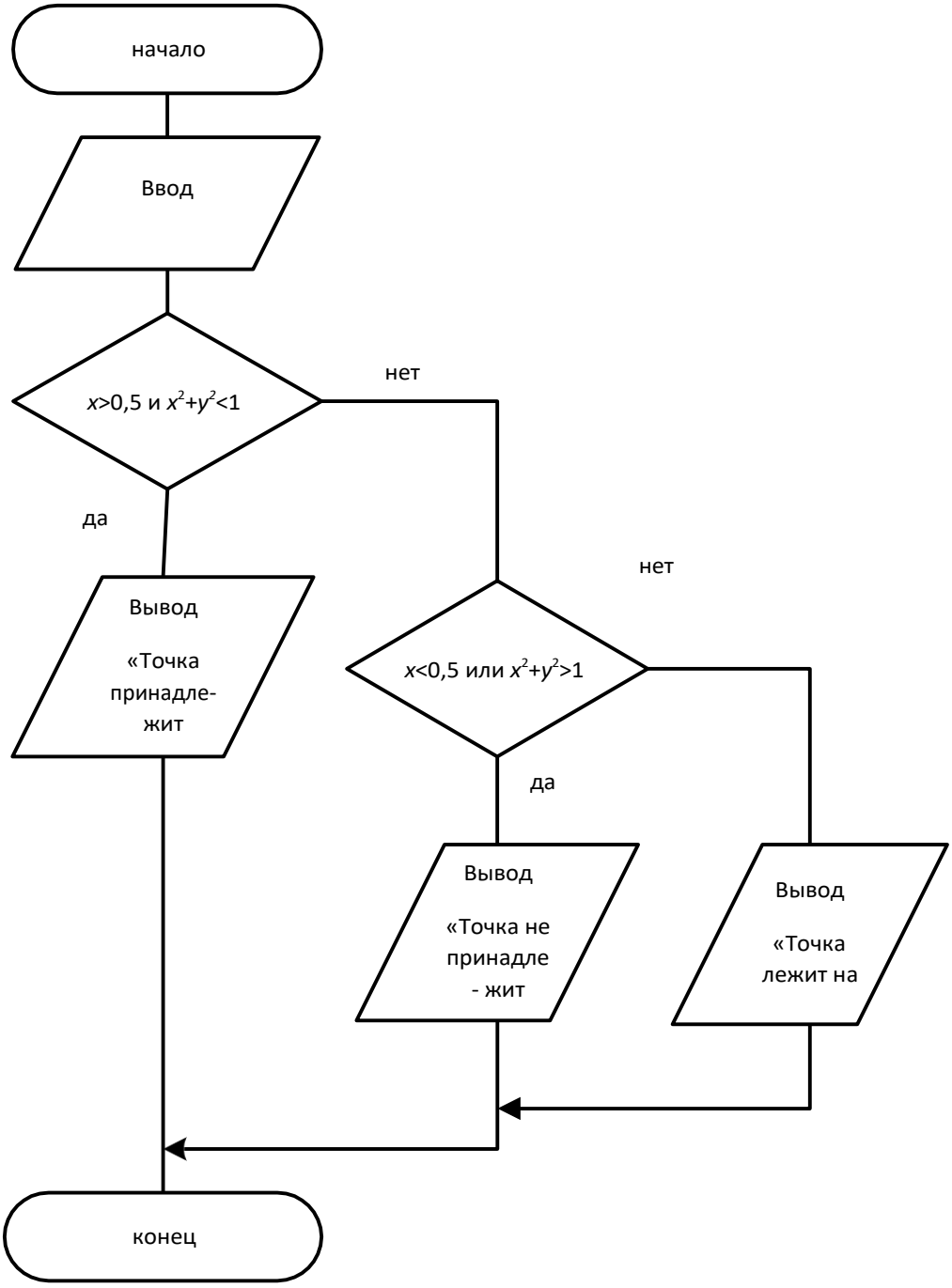


Рисунок 7

Текст программы:

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

{

  double  x,y;

  cout<<”Введите координаты точки: ”;
  cin>>x>>y;

  if ((x>0.5)&&(x*x+y*y<1)) cout<<”Точка принадлежит области”;
  else if ((x<0.5)| |(x*x+y*y>1))

        cout<<”Точка не принадлежит области”;
     else cout<<”Точка лежит на границе области”;

}
```

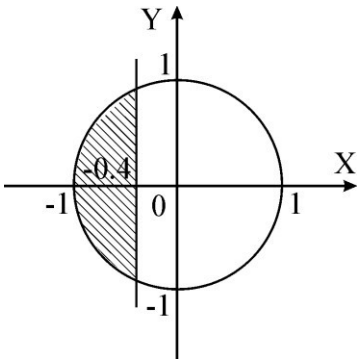
Контрольные вопросы

1. В каких случаях используются условные операторы? Как изображаются условные операторы на схеме программы?
2. Условный оператор в языке С++. Форма записи. Правила выполнения.
3. Истинность и ложность выражений. Значение NULL.
4. Операции конъюнкции, дизъюнкции, отрицания. Знаки операций, их назначение. Какие знаки используются в операциях сравнения?
5. Использование составного оператора в языке С++. Отличие блока от составного оператора.
6. Вложенные операторы if–else.

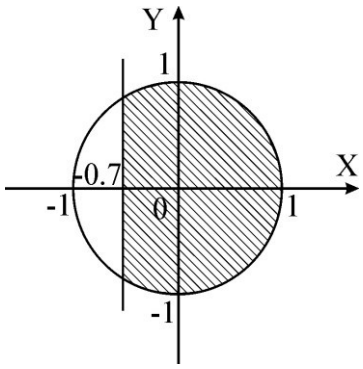
Варианты заданий

По заданным координатам точки определить, где находится точка:

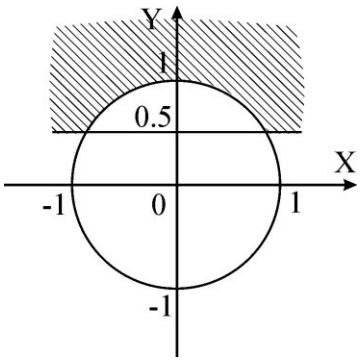
1) внутри заштрихованной области;
2) вне заштрихованной области;
3) на границе этой области.



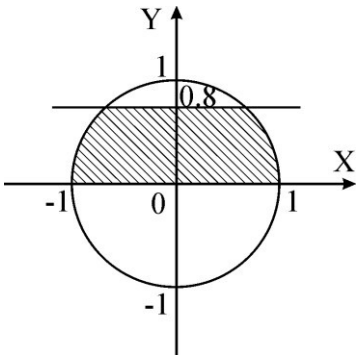
1.



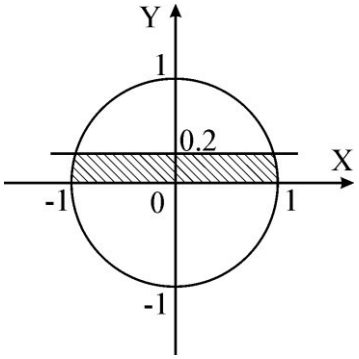
2.



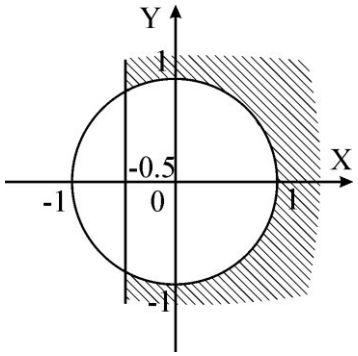
3.



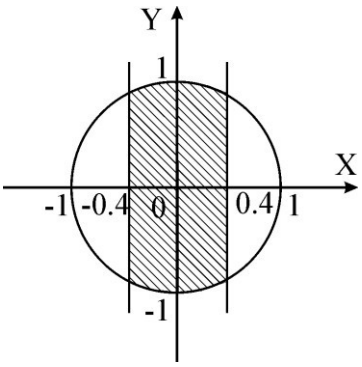
4.



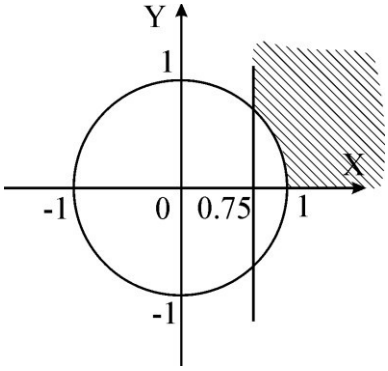
5.



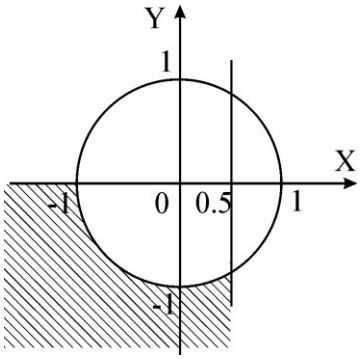
6.



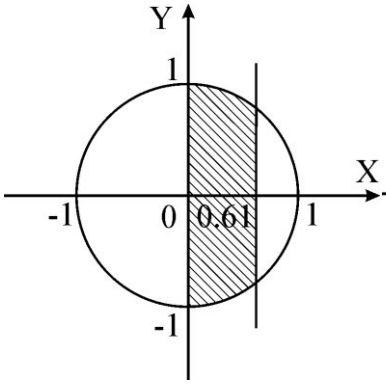
7.



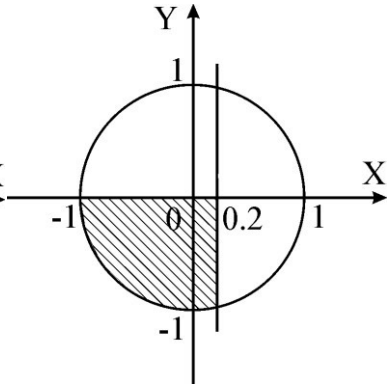
8.



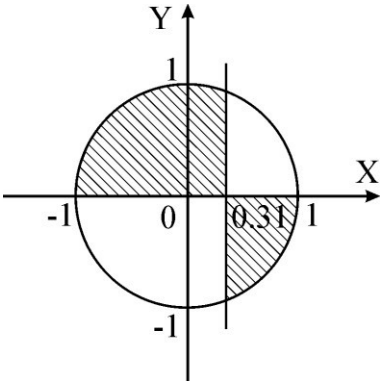
9.



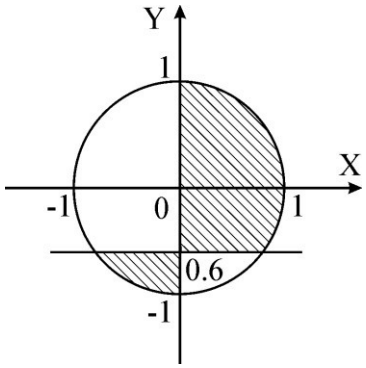
10.



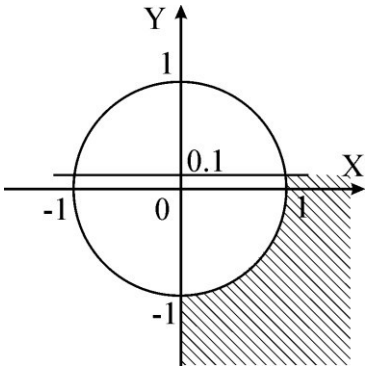
11.



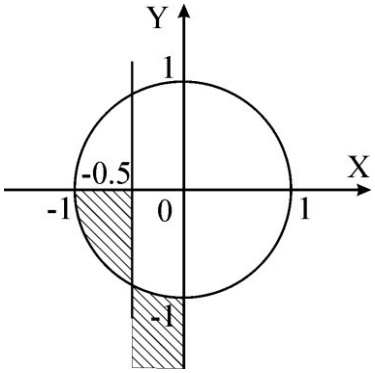
12.



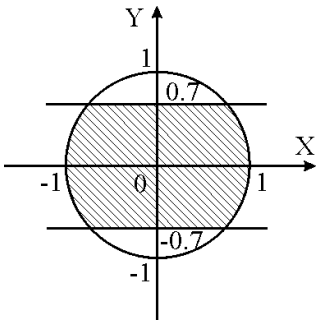
13.



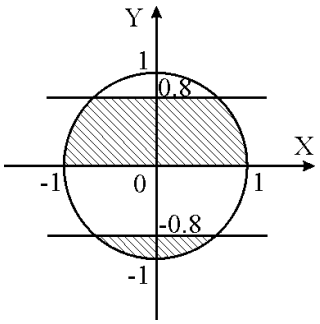
14.



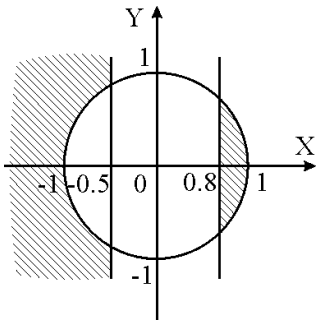
15.



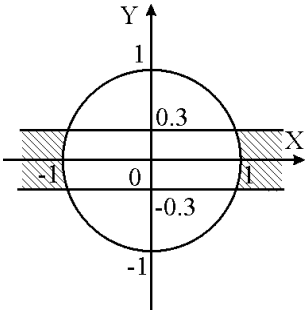
16.



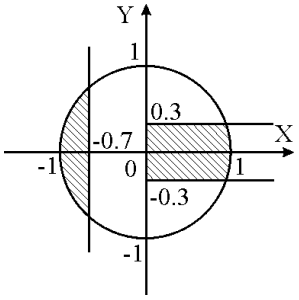
17.



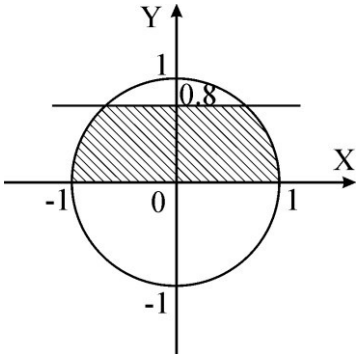
18.



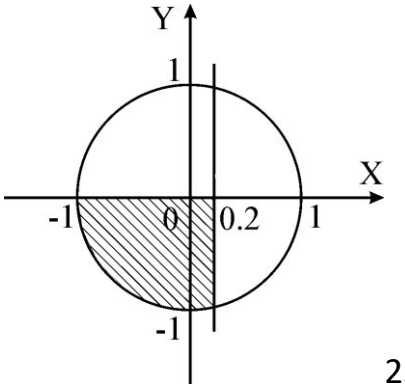
19.



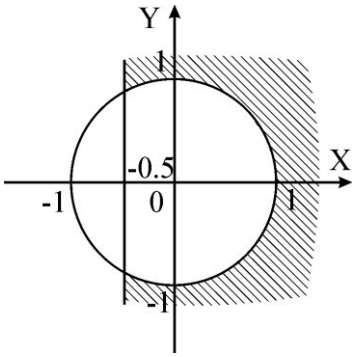
20.



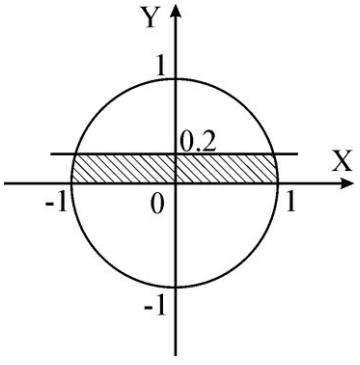
21.



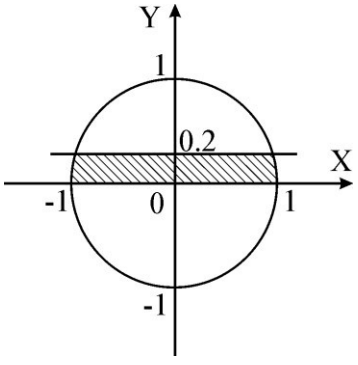
22.



23.



5.



24. 2

Работа № 3

Тема: "Программирование циклов с неизвестным заранее числом повторений"
Цель работы: освоение средств языка С++ для описания итерационных циклов и закрепление навыков использования их при программировании.

Краткие теоретические сведения.

Иногда необходимо повторять одно и то же действие несколько раз подряд. Для этого используют циклы. Итерационные циклы — это циклы с заранее неизвестным числом шагов. Различают циклы с пред и пост условием. Графически цикл с предусловием представлен на рисунке 8.

Цикл с предусловием будет выполняться, пока условие, указанное в круглых скобках будет истинным. Синтаксис цикла с предусловием:

```
while (Условие)
{
   тело цикла;
}
```

Рисунок 8

Синтаксис цикла с постусловием:

```
// форма записи оператора цикла do while:
do // начало цикла do while
{
/*блок операторов*/;
}
while (/*условие выполнения цикла*/); // конец цикла do while
```

Рисунок 9

Цикл с постусловием do while отличается от цикла while тем, что в do while сначала выполняется тело цикла, а затем проверяется условие продолжения цикла. Из-за такой особенности do while называют циклом с постусловием. Таким образом, если условие do while заведомо ложное, то хотя бы один раз блок операторов в теле цикла do while выполнится. В итоге do while отличается от цикла while структурой. Если в while сначала выполняется проверка условия продолжения цикла, и если условие истинно, то только тогда выполняется тело цикла. Цикл do while работает с точностью до наоборот, сначала выполняется тело цикла, а потом проверяется условие, вот почему тело цикла do while, хотя бы раз, выполнится. Графическое изображение цикла представлено на рисунке 9.

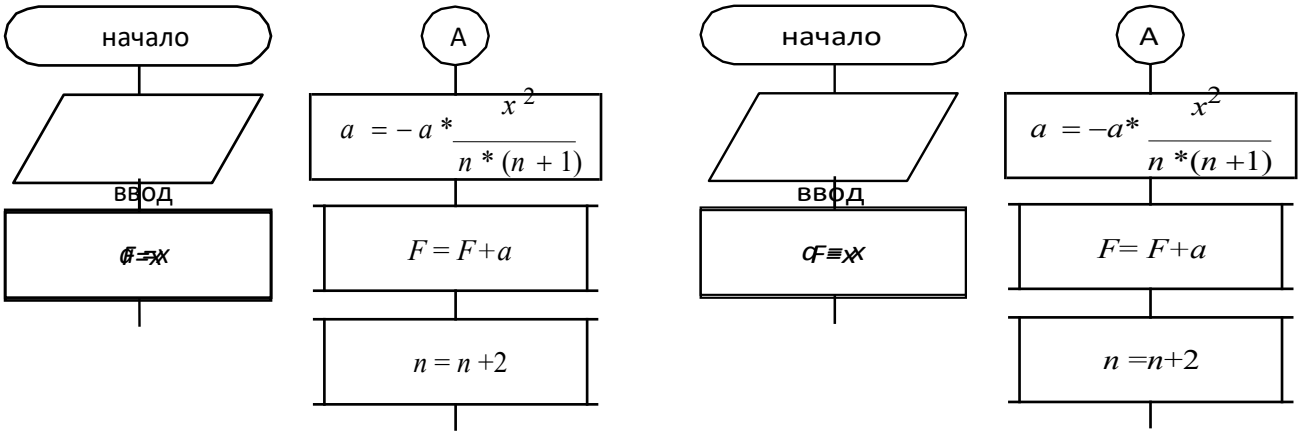
Пример выполнения задания.

Задание: составить схему программы и программу на языке С++ с использованием оператора цикла с предусловием и оператора цикла с постусловием

для вычисления функции sin x с заданной точностью ε с использованием разложения в ряд:

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots$$

Схема программы представлена на рисунке 10:



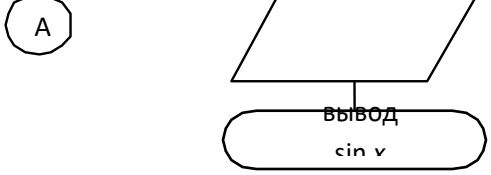


Рисунок 10

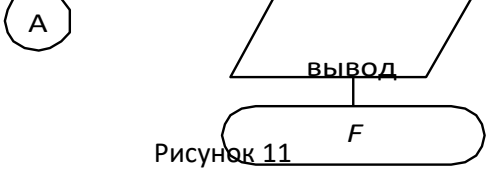


Рисунок 11

Текст программы с использованием оператора **цикла с предусловием**:

```
//Программа вычисления sin
x # include <iostream>

# include <math.h>
void main ()

{

  double x, eps;

  cout<<”Введите значения аргумента и точности\n”;
  cin>>x>>eps;

  double F=x, a=x;
  int n=2;

  while (fabs(a)>=eps)

   {

    a*= –x*x/(n*(n+1));
    F+=a;
```

```
        n+=2;
        }
      cout<<”Приближенное значение sin x=”<<F<<endl;
      cout<<”Точное значение sin x=”<<sin(x);
    }
```

Схема программы с использованием цикла с постусловием представлена на рисунке 11.

Текст программы с использованием оператора **цикла с постусловием**:

```
//Программа вычисления sin
x # include <iostream.h>

# include <math.h>
void main ()

{

  double x, eps;

  cout<<”Введите значения аргумента и точности\n”;
  cin>>x>>eps;

  double F=x, a=x;
  int n=2;

  do

   {

    a*= –x*x/(n*(n+1));
    F+=a;

    n+=2;

    }

  while (fabs(a)>=eps)

  cout<<”Приближенное значение sin x=”<<F<<endl;
  cout<<”Точное значение sin x=”<<sin(x);

}
```

Контрольные вопросы

1. Какие операторы языка C++ используются для организации итерационных циклов?
2. Синтаксис оператора цикла с предусловием.
3. Как выполняется оператор цикла с предусловием?
4. Синтаксис оператора цикла с постусловием.
5. Как выполняется оператор цикла с постусловием?
6. Чем отличаются операторы цикла с предусловием и с постусловием?
7. В каких случаях в операторе цикла используется составной оператор или блок?

Варианты задание

Варианты зданий представлены в таблице 6 Таблица

6.

№ вариант а	Функция	Разложение в ряд	Область сходимости
1	e^x	$1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + ...$	$x < \infty$
2	$\sin(x + a)$	$\sin(a) + x\cos a - \frac{x^2 \sin(a)}{2!} - \frac{x^3 \cos(a)}{3!} + \frac{x^4 \sin(a)}{4!} + ..$	$\lvert x \rvert < \infty$
3	$\cos x$	$1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + ...$	$x < \infty$
4	$\operatorname{arctg} x$	$x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + ...$	$\lvert x \rvert < 1$
5	$\ln(1 + x)$	$x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + ...$	$\lvert x \rvert < 1$
6	$\operatorname{sh} x$	$x + \frac{x^3}{3!} + \frac{x^5}{5!} + \frac{x^7}{7!} + ...$	$x < \infty$
7	$\operatorname{ch} x$	$1 + \frac{x^2}{2!} + \frac{x^4}{4!} + \frac{x^6}{6!} + ...$	$x < \infty$
8	$a^2 + b$	$a\left[1 + \frac{1}{2}\cdot\frac{b}{a^2} - \frac{1\cdot 1}{2\cdot 4}\cdot\left(\frac{b}{a^2}\right)^2 + \frac{1\cdot 3}{2\cdot 4\cdot 6}\cdot\left(\frac{b}{a^2}\right)^3 + ...\right]$	$-1 < \frac{b}{a^2} < 1$
9	$\sqrt[3]{a^3 + b}$	$a\left[1 + \frac{1}{3}\cdot\frac{b}{a^3} - \frac{2}{3\cdot 6}\cdot\left(\frac{b}{a^3}\right)^2 + \frac{2\cdot 5}{3\cdot 6\cdot 9}\cdot\left(\frac{b}{a^3}\right)^3 + ...\right]$	$-1 < \frac{b}{a^3} < 1$
10	$\lg(a + b)$	$\lg a + 2M\left[\frac{b}{2a+b} + \frac{1}{3}\left(\frac{b}{2a+b}\right)^3 + \frac{1}{5}\left(\frac{b}{2a+b}\right)^5 + ...\right]$ $M = 0{,}434294482\ ...$	$a > 0,\ b > 0$
11	$\ln x$	$2\left[\frac{x-1}{x+1} + \frac{(x-1)^3}{3(x+1)^3} + \frac{(x-1)^5}{3(x+1)^5} + ...\right]$	$x > 0$
12	$\sqrt{\ln x}$	$(x-1) - \frac{(x-1)^2}{2} + \frac{(x-1)^3}{3} - \frac{(x-1)^4}{4} + ...$	$0 < x \le 2$
13	$\arcsin x$	$x + \frac{x^3}{2\cdot 3} + \frac{1\cdot 3\cdot x^5}{2\cdot 4\cdot 5} + \frac{1\cdot 3\cdot 5\cdot x^7}{2\cdot 4\cdot 6\cdot 7} + \frac{1\cdot 3\cdot 5\cdot 7\cdot x^9}{2\cdot 4\cdot 6\cdot 8\cdot 9} + ...$	$\lvert x \rvert < 1$

14	$\arccos x$	$\frac{\pi}{2} - \left(x + \frac{x^3}{2 \cdot 3} + \frac{1 \cdot 3 \cdot x^5}{2 \cdot 4 \cdot 5} + \frac{1 \cdot 3 \cdot 5 \cdot x^7}{2 \cdot 4 \cdot 6 \cdot 7} + \ldots \right)$	$\lvert x \rvert < 1$
15	$\operatorname{arctg} x$	$\pm \frac{\pi}{2} - \frac{1}{x} + \frac{1}{3x^3} - \frac{1}{5x^5} + \frac{1}{7x^7} - \ldots$	$\lvert x \rvert > 1$
16	$\operatorname{arsh} x$	$x - \frac{1}{2 \cdot 3} x^3 + \frac{1 \cdot 3}{2 \cdot 4 \cdot 5} x^5 - \frac{1 \cdot 3 \cdot 5}{2 \cdot 4 \cdot 6 \cdot 7} x^7 + \ldots$	$\lvert x \rvert < 1$
17	$\operatorname{arth} x$	$x + \frac{x^3}{3} + \frac{x^5}{5} + \frac{x^7}{7} + \ldots$	$\lvert x \rvert < 1$
18	$\operatorname{arcth} x$	$\frac{1}{x} + \frac{1}{3x^3} + \frac{1}{5x^5} + \frac{1}{7x^7} + \ldots$	$\lvert x \rvert > 1$
19	$\ln x$	$\frac{x-1}{x} + \frac{(x-1)^2}{2x^2} + \frac{(x-1)^3}{3x^3} + \ldots$	$x > \frac{1}{2}$
20	$\operatorname{arcctg} x$	$\frac{\pi}{2} - \left(x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \ldots \right)$	$\lvert x \rvert < 1$
21	$-\ln(1-x)$	$x + \frac{x^2}{2} + \frac{x^3}{3} + \frac{x^4}{4} + \ldots$	$-1 > x > 1$
22	$\frac{1}{(1-x)^2}$	$1 + 2 \cdot x + 3 \cdot x^2 + 4 \cdot x^3 \ldots$	$\lvert x \rvert < 1$
23	$\frac{1}{(1-x)}$	$1 + x + x^2 + x^3 \ldots$	$-1 > x > 1$
24	$\frac{1}{(1+x)}$	$1 - x + x^2 - x^3 \ldots$	$-1 > x > 1$
25	$\ln\left(\frac{1+x}{1-x}\right)$	$2 \cdot \left(x + \frac{x^3}{3} + \frac{x^5}{5} + \frac{x^7}{7} + \ldots\right)$	$0 < x < 1$

Работа № 4

Тема: "Программирование циклов с параметром.

Одномерные массивы"

Цель работы: освоение средств языка С++ для описания циклов с параметром и закрепление навыков использования их при программировании, изучение способов описания, ввода-вывода и обработки одномерных массивов.

Краткие теоретические сведения.

Цикл с параметром используется в том случае если известно число шагов обработки, т.е. если известно число итераций. Итерацией цикла называется один проход цикла.

Синтаксис оператора цикла имеет вид:

```
for (действие до начала цикла; условие
    продолжения цикла;
    действия в конце каждой итерации цикла) { инструкция
        цикла;
        инструкция цикла 2;
        инструкция цикла N;
}
```

Существует частный случай этой записи:

```
for (счетчик = значение; счетчик < значение; шаг цикла)
{
    тело цикла;
}
```

Описание синтаксиса:

1) сначала присваивается первоначальное значение счетчику;

2) затем задается конечное значение счетчика цикла. Счетчик цикла — это переменная, в которой хранится количество проходов данного цикла. После того, как значение счетчика достигнет указанного предела, цикл завершится;

3) задаем шаг цикла. Шаг цикла — это значение, на которое будет увеличиваться или уменьшаться счетчик цикла при каждом проходе.

Одномерный массив — массив, с одним параметром, характеризующим количество элементов одномерного массива. Фактически одномерный массив — это массив, у которого может быть только одна строка, и n-е количество столбцов. Столбцы в одномерном массиве — это элементы массива. На рисунке 12 показана структура целочисленного одномерного массива *a*. Размер этого массива — 16 ячеек.

5	-12	-12	9	10	0	-9	-12	-1	23	65	64	11	43	39	-15
a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]	a[10]	a[11]	a[12]	a[13]	a[14]	a[15]

Рисунок 12 — Одномерный массив

Максимальный индекс одномерного массива *a* равен 15, но размер массива 16 ячеек, потому что нумерация ячеек массива всегда начинается с 0. Индекс ячейки – это целое неотрицательное число, по которому можно обращаться к каждой ячейке массива и выполнять какие-либо действия над ней (ячейкой). Синтаксис объявления одномерного массива в C++:

тип данных имя одномерного массива [размерность одномерного массива];

//пример объявления одномерного массива, изображенного на рисунке 12:

int a[16];

где, int — целочисленный тип данных; a — имя одномерного массива; 16 — размер одномерного массива, 16 ячеек.

Всегда сразу после имени массива идут квадратные скобочки, в которых задаётся размер одномерного массива, этим массив и отличается от всех остальных переменных.

Рассмотрим на примерах способы ввода-вывода одномерного массива:

1) размер массива можно не указывать только при его инициализации, при обычном объявлении массива обязательно нужно указывать размер массива

```
#include "stdafx.h"
#include <iostream>
using namespace std;


int main(int argc, char* argv[])

{

   cout << "obrabotka massiva" << endl;

// объявление и инициализация одномерного массива

   int array1[16] = { 5, -12, -12, 9, 10, 0, -9, -12, -1, 23, 65, 64, 11, 43, 39, -15 };

   cout << "indeks" << "\t\t" << "element massiva" << endl; // печать
   заголовков for (int counter = 0; counter < 16; counter++)  //начало цикла

   {

   //вывод на экран индекса ячейки массива, а затем содержимого этой
    ячейки cout << "array1[" << counter << "]" << "\t\t" << array1[counter] <<
    endl;

   }

   system("pause")
   ; return 0;
```

```
}
```

Результат работы представлен на рисунке 13.

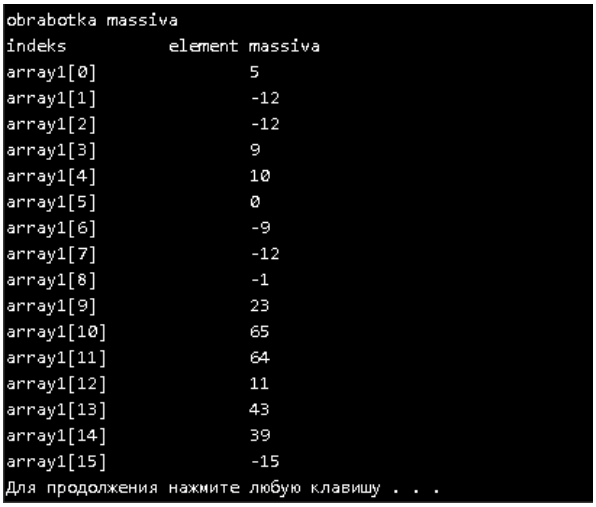


Рисунок 13

2) программа должна последовательно считывать десять введённых чисел с клавиатуры. Все введённые числа просуммировать, результат вывести на экран. #include "stdafx.h"

```
#include <iostream> using
namespace std;


int main(int argc, char* argv[])
{
    int array1[10]; // объявляем целочисленный массив cout <<
    "Enter elementi massiva: " << endl;
    int sum = 0;
    for ( int counter = 0; counter < 10; counter++ ) // цикл для считывания чисел cin >>
        array1[counter];  // считываем вводимые с клавиатуры числа
    cout << "array1 = {";
    for ( int counter = 0; counter < 10; counter++ ) // цикл для вывода элементов массива cout <<
        array1[counter] << " ";  // выводим элементы массива на стандартное
устройство вывода
    for ( int counter = 0; counter < 10; counter++ ) // цикл для суммирования чисел массива
        sum += array1[counter]; // суммируем элементы массива cout << "}\
    nsum = " << sum << endl;
    system("pause");
    return 0;
}
```

Результат представлен на рисунке 14.

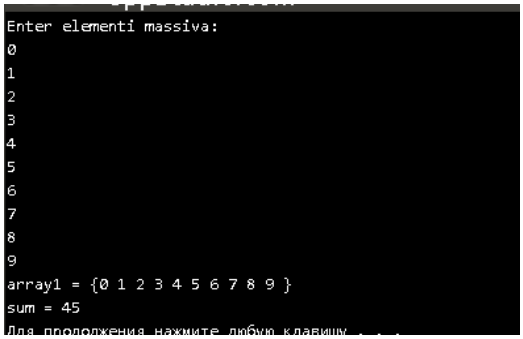


Рисунок 14.

Пример выполнения задания.

Задание: произвести следующую обработку 15 целых чисел: подсчитать сумму чисел, входящих в диапазон [-5..5] и количество нечетных чисел.

Схема программы к данному заданию с использованием одного цикла представлена на рисунке 15.

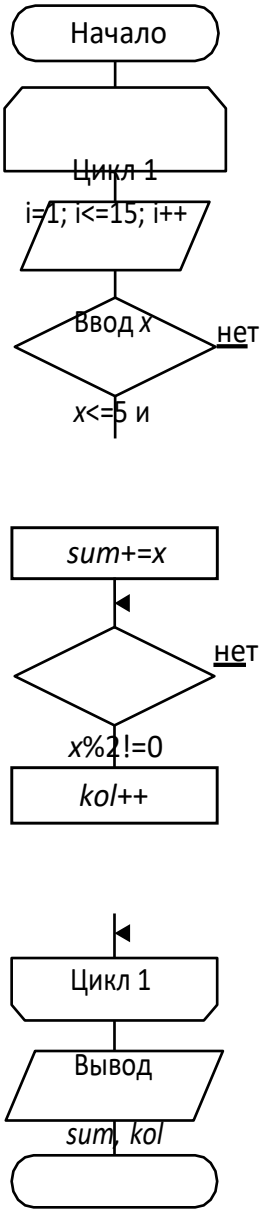


Рисунок 15 Текст

программы без использования массива:

```
#include <iostream>


void main()

{

int x,sum=0,i,kol=0;
printf("Enter numbers\
n"); for (i=1;i<=15;i++)

{

scanf("%d",&x);

if ((x>=-5)&&(x<=5)) sum+=x; if
(x%2!=0) kol++;

}

printf("Summa v diapazone [-5,5]=%d\n", sum);
printf("Kolichestvo nechetnih=%d", kol);

}
```

Схема программы с циклами для ввода и обработки массива (рисунок 16).

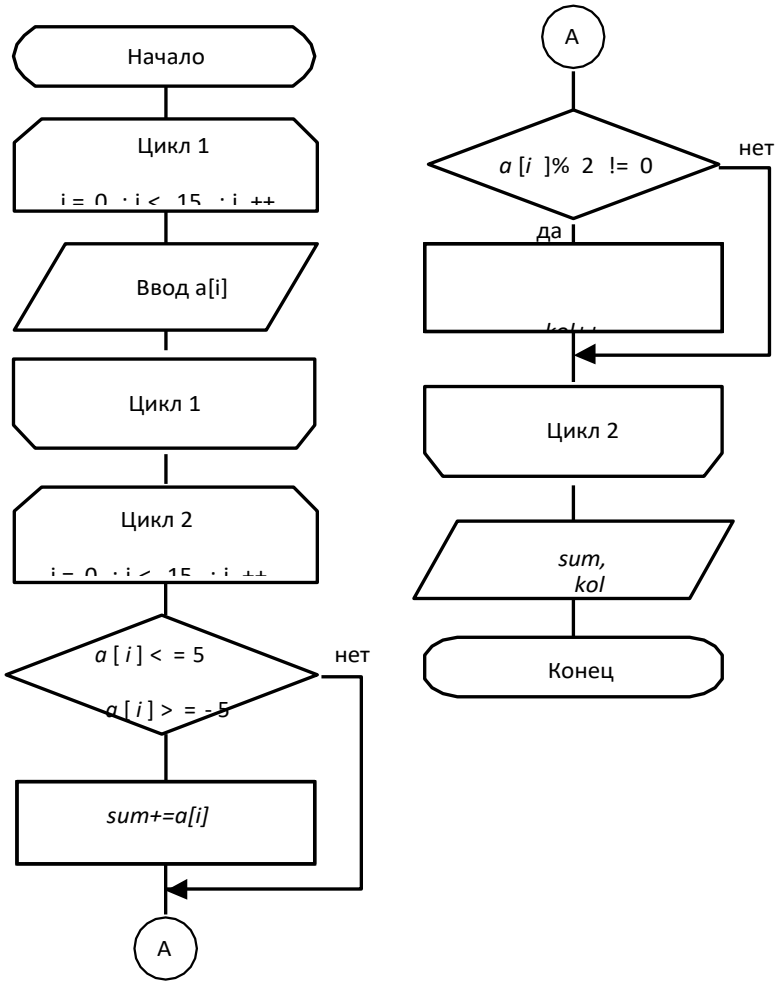


Рисунок 16

Пример программы с использованием одномерного массива

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

{

int a[15],sum=0,i,kol=0;
printf("Enter numbers\
n");
```

```
for (i=0;i<15;i++)
scanf("%d",&a[i]);
for (i=0;i<15;i++)

{

if ((a[i]>=-5)&&(a[i]<=5)) sum+=a[i]; if
(a[i]%2!=0) kol++;

}

printf("Summa v diapazone [-5,5]=%d\n", sum);
printf("Kolichestvo nechetnih=%d", kol);

}
```

Контрольные вопросы

1. Массивы в языке С++: понятие массива в языке С++, описание массива в программе, представление элементов массива в памяти, обращение к элементам массива.
2. Оператор цикла for в языке С++. Форма записи. Правила выполнения.

Варианты заданий

1. Произвести следующую обработку 15 целых чисел: найти количество отрицательных чисел, количество нулевых и подсчитать сумму положительных чисел.
2. Произвести следующую обработку 15 целых чисел: найти количество четных чисел, а нечетные числа, входящие в диапазон [1..11] возвести в квадрат.
3. Произвести следующую обработку 15 вещественных чисел: найти количество отрицательных чисел, а числа, входящие в диапазон [0..10] возвести в квадрат.
4. Произвести следующую обработку 10 вещественных чисел: найти количество чисел, больших или равных 1,5, и подсчитать сумму отрицательных чисел.
5. Произвести следующую обработку 10 вещественных чисел: найти количество чисел, равных нулю, и найти сумму чисел, входящих в диапазон [-15..15].
6. Произвести следующую обработку 15 целых чисел: найти количество отрицательных чисел и подсчитать разность положительных чисел.
7. Произвести следующую обработку 15 вещественных чисел: найти среднее

арифметическое положительных чисел и подсчитать количество чисел, входящих в диапазон [-15..5].

8. Произвести следующую обработку 10 целых чисел: найти количество отрицательных чисел и подсчитать сумму положительных чисел, делящихся без остатка на 3.

9. Произвести следующую обработку 10 целых чисел: найти количество отрицательных чисел, а числа, входящие в диапазон [0..10], умножить на 10.

10. Произвести следующую обработку 10 целых чисел: найти количество отрицательных чисел, а числа, входящие в диапазон [0..10], умножить на 3.

11. Произвести следующую обработку 15 вещественных чисел: найти среднее арифметическое отрицательных чисел и подсчитать количество чисел, входящих в диапазон [0..5].

12. Произвести следующую обработку 15 вещественных чисел: найти среднее арифметическое нечетных чисел и подсчитать сумму чисел, входящих в диапазон [-15..5].

13. Произвести следующую обработку 10 вещественных чисел: найти количество чисел, равных нулю, и найти синус чисел, входящих в диапазон [-15..15].

14. Произвести следующую обработку 10 целых чисел: подсчитать сумму положительных чисел и определить номера отрицательных чисел.

15. Произвести следующую обработку 15 вещественных чисел: найти количество отрицательных чисел и номера нулевых чисел.

16. Произвести следующую обработку 12 целых чисел: подсчитать количество чисел, делящихся без остатка на 5, и сумму чисел, входящих в диапазон [-5..5].

17. Произвести следующую обработку 10 вещественных чисел: подсчитать количество чисел, отличающихся от числа 3 не более чем на 0.5, и сумму отрицательных чисел.

18. Произвести следующую обработку 15 целых чисел: подсчитать количество нулевых чисел и вычислить квадраты чисел, входящих в диапазон [-5..5].

19. Произвести следующую обработку 12 целых чисел: подсчитать количество нечетных чисел и сумму отрицательных чисел.

20. Произвести следующую обработку 15 вещественных чисел: подсчитать количество чисел, отличающихся от заданного не более чем на 0.5, и сумму положительных чисел.

21. . Произвести следующую обработку 10 вещественных чисел: найти количество отрицательных чисел, находящихся в диапазоне от -5 до 5 и подсчитать сумму положительных чисел.

22. Произвести следующую обработку 15 целых чисел: найти количество чисел, входящих в диапазон [1..11] и каждое число возвести в квадрат.

23. Произвести следующую обработку 15 вещественных чисел: найти количество чисел, равных 0, а числа, входящие в диапазон [-1..1] возвести в куб.

24. Произвести следующую обработку 10 вещественных чисел: найти количество чисел, больших или равных 1,5, и подсчитать сумму отрицательных чисел, входящих в диапазон [-1..0].

25. Произвести следующую обработку 10 вещественных чисел: найти количество чисел, меньших 15, и найти произведение чисел, входящих в диапазон [10..15].

Работа № 5

Тема:"Обработка двумерных массивов. Указатели"

Цель работы: изучение способов описания, ввода-вывода и обработки двумерных массивов, использование указателей при работе с массивами.

Краткие теоретические сведения.

Двумерный массив — это обычная таблица, со строками и столбцами. Фактически двумерный массив — это одномерный массив одномерных массивов. Структура двумерного массива, с именем *a*, размером *m* на *n* показана ниже (см. рисунок 17).

Рисунок 17 — Двумерный массивы в C++

a[0][0]	a[0][1]	a[0][2]	a[0][3]	. . .	a[0][n]
a[1][0]	a[1][1]	a[1][2]	a[1][3]	. . .	a[1][n]
a[2][0]	a[2][1]	a[2][2]	a[2][3]	. . .	a[2][n]
. . .	. . .	. . .	. . .	. . .	. . .
a[m][0]	a[m][1]	a[m][2]	a[m][3]	. . .	a[m][n]

где, *m* — количество строк двумерного массива; *n* — количество столбцов двумерного массива;

m x *n* — количество элементов массива.

Наиболее распространенный синтаксис объявления двумерного массива:

/*тип данных*/ /*имя массива*/[/*количество строк*/][/*количество столбцов*/];

Примеры инициализации двумерного массива:

— способ 1:

```
int arr[5][3];
```

— способ 2:

```
int arr[5][3] = { {4, 7, 8}, {9, 66, -1}, {5, -5, 0}, {3, -3, 30}, {1, 1, 1} };
```

В последнем случае представление массива и обращение к элементу массива имеет вид, показанный на рисунке 18,19.

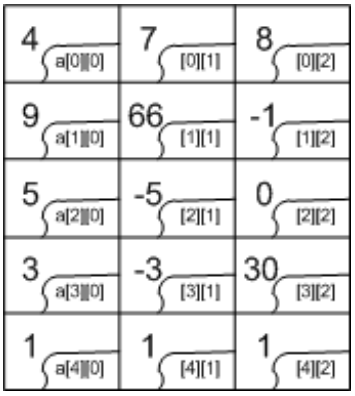


Рисунок 18 — Двумерный массив в C++

Рисунок 19 — Обращение в элементу массива

Ввод массива с клавиатуры и его вывод на экран выполняется следующим образом:

```
int m,n,i,j;

 cout <<"Введите размеры матрицы:"<<endl;
 cin>>m;

 cin>>n;
int
elem;

int ar [m][n];

 for (i=0; i<m; i++)

 {for (j=0; j<n; j++){ cout<<"Введите
   элемент:"<<endl; cin>>elem;

   ar [i][j]=elem;

   }

}

for (i=0; i<m; i++)

{for (j=0; j<n; j++){
   cout<<ar[i][j]<<endl;

   }

}

system("pause")
; return 0;

}
```

При работе с массивами можно использовать указатели. Указатель – переменная, значением которой является адрес ячейки памяти. То есть указатель ссылается на блок данных из области памяти, причём на самое его начало. Указатель может ссылаться на переменную или функцию. Для этого нужно знать адрес переменной или функции. Так вот, чтобы узнать адрес конкретной переменной в C++ существует унарная операция взятия адреса &. Такая операция извлекает адрес объявленных переменных, для того, чтобы его присвоить указателю.

Указатели используются для передачи по ссылке данных, что намного ускоряет процесс обработки этих данных (в том случае, если объём данных большой), так как их не надо копировать, как при передаче по значению, то есть, используя имя переменной. В основном указатели используются для организации динамического распределения памяти, например при объявлении массива, не надо будет его ограничивать в размере. Ведь программист заранее не может знать, какого размера нужен массив тому

или иному пользователю, в таком случае используется динамическое выделение памяти под массив. Любой указатель необходимо объявить перед использованием, как и любую переменную:

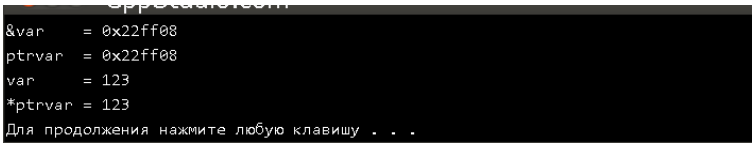
/*тип данных*/ * /*имя указателя*/;

Работу с указателями можно представить следующим образом:

```
#include "stdafx.h"

#include

<iostream>
```

```
using namespace std;

 int main(int argc, char* argv[])

{

    int var = 123; // инициализация переменной var числом 123

    int *ptrvar = &var; // указатель на переменную var (присвоили адрес переменной

    указателю) cout << "&var        = " << &var << endl;// адрес переменной var содержащийся в

    памяти,
извлечённый операцией взятия адреса

    cout << "ptrvar  = " << ptrvar << endl;// адрес переменной var, является значением указателя

    ptrvar cout << "var        = " << var << endl; // значение в переменной var

    cout << "*ptrvar = " << *ptrvar << endl; // вывод значения содержащегося в переменной var через
указатель, операцией разименования указателя

    system("pause")

    ; return 0;

}
```

Результат работы представлен ниже на рисунке 20.



Пример выполнения задания.

Задание. Найти максимальную сумму элементов строк матрицы 3×5. Написать программы без использования указателей и с использованием указателей.

Схема программы без использования указателей представлена на рисунке 20:

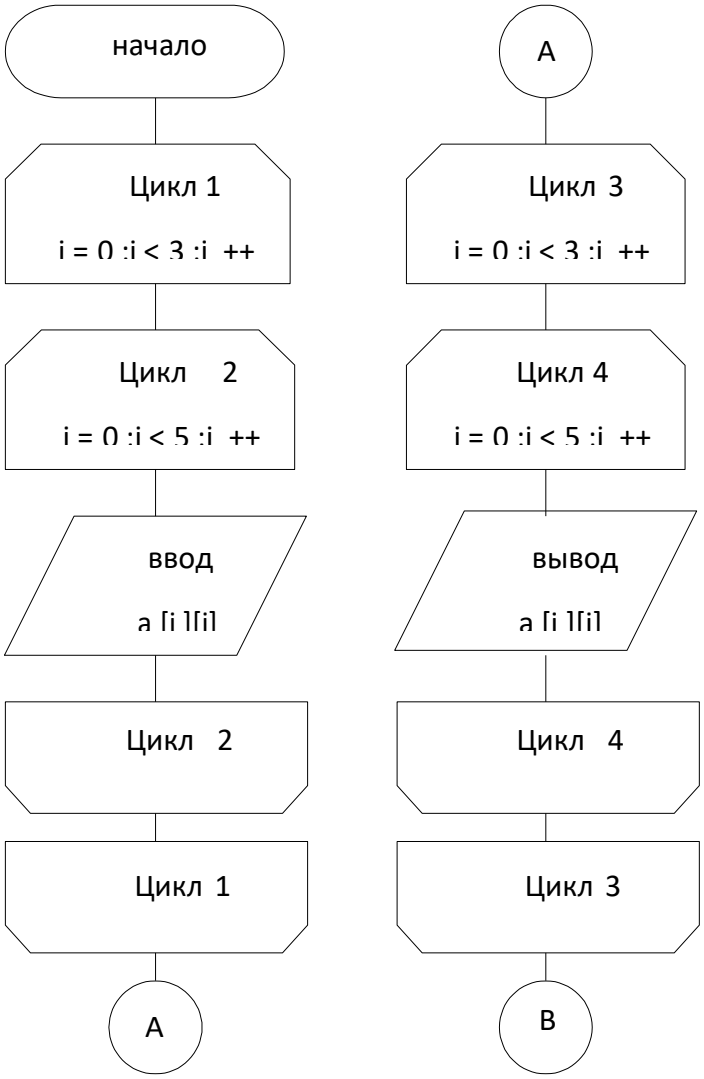
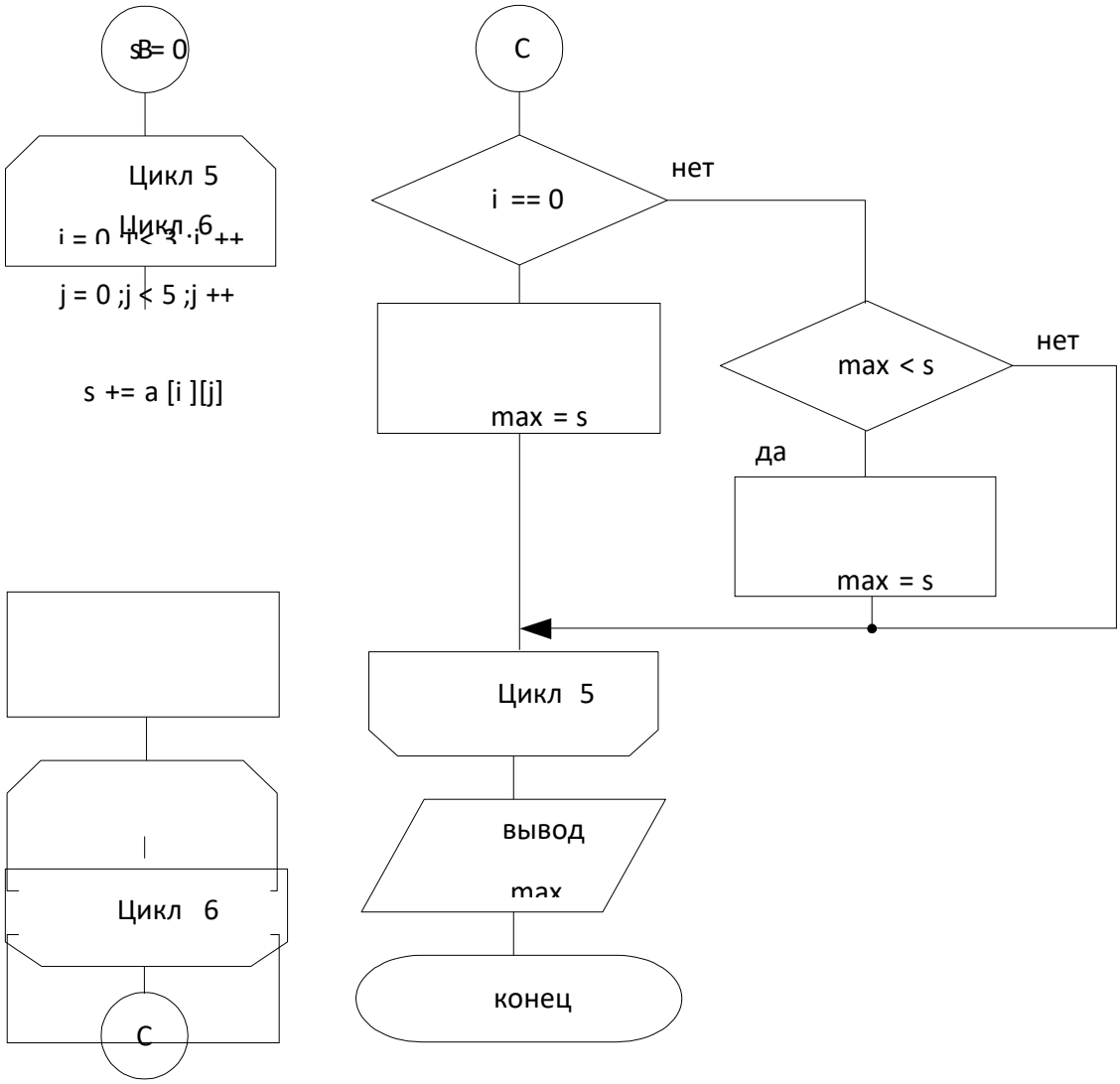

начало
Цикл 1
i = 0 ;i < 3 ;i ++
Цикл 2
i = 0 ;i < 5 ;i ++
ввод
a [i][i]
Цикл 2
Цикл 1
A
A
Цикл 3
i = 0 ;i < 3 ;i ++
Цикл 4
i = 0 ;i < 5 ;i ++
вывод
a [i][i]
Цикл 4
Цикл 3
B

Рисунок 20



Продолжение рисунка 20

Текст программы:

```
#include <stdio.h>
void main()

{

int a[3][5], i, j, s, max;

printf (“Введите 3 строки по 5 чисел”);
for (i=0;i<3;i++)

 for (j=0;j<5;j++)
scanf("%d",&a[i][j]);
printf (“Матрица a :\n”);
for (i=0; i<3; i++)
```

```
  {for (j=0; j<5; j++) printf
    (“%5d”, a[i][j]); printf
    (“\n”);

  }

for(i=0;i<3;i++)

  {s=0;

    for (j=0;j<5;j++)
      s+=a[i][j];

    if (i==0) max=s;

    else if (max<s) max=s;
```

```
 }
printf("Максимальная сумма строки = %d",max);
}
```

Схема программы с использованием указателей представлена на рисунке 21:

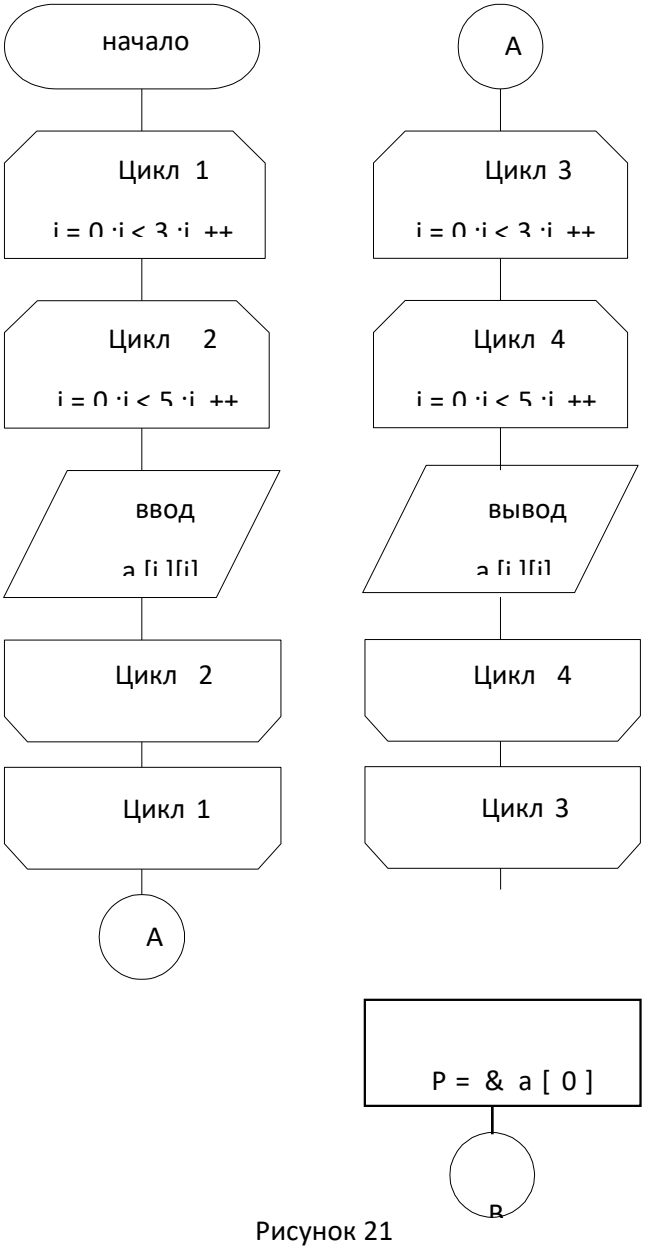
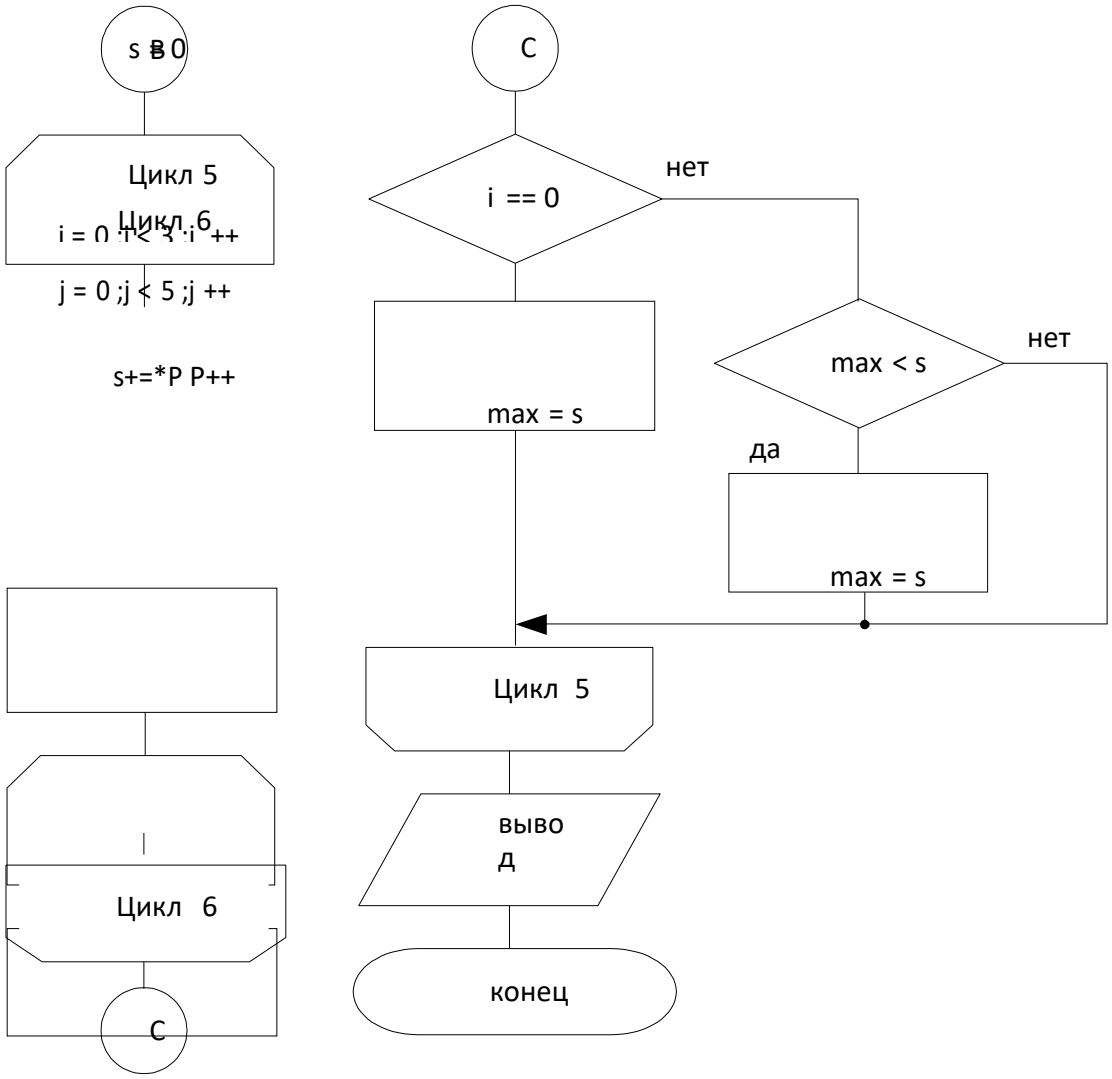


Рисунок 21



Продолжение рисунка 21. Пример программы с использованием указателей:

```
#include <stdio.h>
void main()

{

int a[3][5], *P, i, j, s, max;

printf (“Введите 3 строки по 5 чисел”);
for (i=0;i<3;i++)

 for (j=0;j<5;j++)
scanf("%d",&a[i][j]);
printf (“Матрица a :\n”);
for (i=0; i<3; i++)

 {for (j=0; j<5; j++) printf
  (“%5d”, a[i][j]); printf
  (“\n”);
```

```
  } P=&a[0]
[0];

for(i=0;i<3;i++)

  {s=0;

    for (j=0;j<5;j++)

      {s+=*P;
        P++;

      }
```

```
    if (i==0) max=s;
    else if (max<s) max=s;
  }
printf("Максимальная сумма строки = %d",max);
}
```

Контрольные вопросы

1. Правила организации вложенных циклов.
2. Особенности организации двумерных массивов: понятие массива в языке C++, описание массива в программе, представление элементов массива в памяти, обращение к элементам массива.
3. Указатели в языке C++: понятие указателя, описание указателя в программе.
4. Операции над указателями.
5. Связь массивов и указателей.

Варианты заданий

1. Вычислить сумму положительных элементов каждого столбца матрицы A(m×n).
2. Из матрицы *X* (m×n) построить матрицу *Y*, поменяв местами строки и столбцы.
3. Найти наименьший элемент матрицы *X* (m×n) и записать нули в ту строку и столбец, где он находится.
4. Переписать первые элементы каждой строки матрицы *A* (m×n), большие *C*, в массив *B***.** Если в строке нет элемента, большего *C*, то записать ноль в массив *B*.
5. Дана действительная матрица размера *m*×*n*. Найти сумму наибольших значений элементов ее строк.
6. В данной действительной матрице размера m×n поменять местами строку, содержащую элемент с наибольшим значением, со строкой, содержащей элемент с наименьшим значением. Предполагается, что такой элемент единственный.
7. В данной действительной квадратной матрице порядка *n* найти сумму элементов строки, в которой расположен элемент с наименьшим значением. Предполагается, что такой элемент единственный.

8. Дана действительная матрица размера $m \times n$, все элементы которой различны. В каждой строке выбирается элемент с наименьшим значением, затем среди этих чисел выбирается наибольшее. Указать индексы элемента с найденным значением.

9. Дана целочисленная матрица размера m×n. Найти матрицу, получающуюся перестановкой столбцов (первого с последним, второго с предпоследним и т.д.).

10. Дана целочисленная матрица размера m×n. Найти матрицу, получающуюся перестановкой строк (первой с последней и т.д.).

11. Дана действительная матрица $[a_{ij}]$, где i, j = 1..*n.* Получить действительную матрицу $[b_{ij}]$, где i, j = 1..*n,* элемент b_{ij} которой равен сумме элементов данной

матрицы, расположенных в области, определяемой индексами i, j (область заштрихована на рисунке 22):

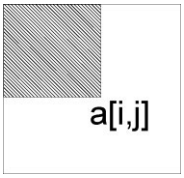


Рисунок 22

12. Дана действительная квадратная матрица порядка *n*. Преобразовать матрицу по правилу: строку с номером *n* сделать столбцом с номером *n,* а столбец с номером *n* сделать строкой с номером *n.*

13. Просуммировать элементы матрицы X (4,5), сумма индексов которых равна заданной константе К.

14. Дана матрица M (4×5). Вычислить вектор D, компоненты которого равны сумме элементов строк матрицы.

15. Дана матрица M (6×6). Вычислить сумму элементов главной диагонали.

16. Дана матрица N (6×5). Найти столбец с минимальной суммой элементов.

17. Дана матрица M (4×5) и константа C. Вычислить матрицу D, равную произведению элементов матрицы M на константу.

18. Дана матрица M (4×6). Вычислить вектор D, компоненты которого равны сумме элементов столбцов матрицы.

19. Дана действительная квадратная матрица порядка *n*, все элементы которой различны. Найти наибольший элемент, среди стоящих на главной и побочной диагоналях и поменять его местами с элементом, стоящим на пересечении этих диагоналей.

20. Дана действительная квадратная матрица порядка n. Найти наибольшее из значений элементов, расположенных в заштрихованной части матрицы (рисунок 23):

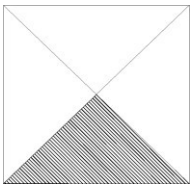


Рисунок 23

21. Дана матрица M (2×5), определить максимальный и минимальный элементы. Поменять местами максимальный и минимальный элементы.

22. В матрице A(n×n) вычислить сумму элементов матрицы (n-2×n-2) и определить максимальный элемент в ней.

23. Дана матрица M (6×6). Вычислить произведение элементов главной диагонали с константой C.

24. Дана матрица N (6×5). Найти строку с минимальной суммой элементов, а элемент с номером n_{ij} возвести в квадрат.

25. Дана матрица вещественных чисел A (m×n). В сроке m определить максимальный элемент, а в столбце n количество элементом, меньших порога k.

Работа № 6

Тема:"Строки"

Цель работы: изучение правил описания, ввода-вывода и основных функций обработки символьных данных.

Краткие теоретические сведения.

Строки в С++ позволяют работать с символьными данными. С помощью строк возможно осуществить чтение с клавиатуры текста, его обработка и вывод.

В С++ существует 2 типа строк:

1. Массив переменных типа char, заканчивающийся нуль-терминатором \0. Символьные строки состоят из набора символьных констант заключённых в двойные кавычки. При объявлении строкового массива необходимо учитывать наличие в конце строки нуль-терминатора, и отводить дополнительный байт под него.

char string[10]; // string – имя строковой переменной, 10 – размер массива, (в строке может поместиться 9 символов, последнее место отводится под нуль-терминатор)

2. Специальный класс string

Для его работы необходимо в начале программы подключить заголовочный файл string: *#include <string>*

Для создания строки необходимо в начале программы написать

using namespace std;

Теперь чтоб создать строку достаточно написать: *string s;*

Для записи в строку можно использовать оператор =

s="Hello";

Доступ к i-му элементу строки **s** типа string осуществляется стандартным образом s[i]. Над строками типа string определены следующие операции:

- присваивания, например s1=s2;
- объединения строк (s1+=s2 или s1=s1+s2) — добавляет к строке s1 строку s2, результат храниться в строке s1, пример объединения строк:
- сравнения строк: s1=s2, s1!=s2, s1<s2, s1>s2, s1<=s2, s1>=s2 — результатом будет логическое значение.

Существует множество функций для работы со строками (таблица 7). Таблица 7 – Функции для работы со строками

Функция	Объяснение
s.append(str)	добавляет в конец строки строку str
s.assign(str)	присваивает строке s значение строки str
s.clear()	отчищает строку, т.е. удаляет все элементы в ней

s.compare(str)	сравнивает строку s со строкой str и возвращает 0 в случае совпадение
s.copy(C, I, N)	копирует из строки s в строку C(может быть как строка типа string, так и строка типа char) I символов, начиная с N-го символа
bool b=s.empty()	если строка пуста, возвращает true, иначе false
s.erase(I,N)	удаляет N элементов с I-го символа

s.find(str,I)	ищет строку str начиная с I- го символа
s.insert(pos, s1)	вставляет строку **s1** в строку **s**, начиная с позиции **pos**
int len=s.length()	записывает в len длину строки
s.push_back(symbol)	добавляет в конец строки символ
s.replace(index, n,str)	берет n первых символов из str и заменяет символы строки s на них, начиная с позиции index
str=s.substr(n,m)	возвращает m символов начиная с позиции n
s.swap(str)	меняет содержимое s и str местами.
s.size()	возвращает число элементов в строке.

Функции для работы со строками, прототипы которых описаны в заголовочном файле string.h:

1. Сравнение строк. Для сравнения строк используются функции strcmp и strncmp. Функция

```
int strcmp ( const char *str1, const char *str2);
```

лексикографически сравнивает строки str1, str2 и возвращает –1, 0 или 1, если строка str1 соответственно меньше, равна или больше строки str2.

Функция

```
int strncmp ( const char *str1, const char *str2, size_t n);
```

лексикографически сравнивает не более чем n первых символов из строк str1 и str2. Функция возвращает –1, 0 или 1, если первые n символов из строки str1 соответственно меньше, равны или больше первых n символов из строки str2.

Пример:

```
// пример сравнения строк
#include #include int
main() {
char str1[] = "aa bb"; char
str2[] = "aa aa"; char str3[] =
"aa bb cc";
printf("%d\n", strcmp(str1, str3)); // печатает: -1 printf("%d\n",
strcmp(str1, str1)); // печатает: -0 printf("%d\n", strcmp(str1,
str2)); // печатает: 1 printf("%d\n", strncmp(str1, str3, 5)); //
печатает: 0 return 1;
 }
```

2. Копирование строк. Для копирования строк используются функции strcpy и strncpy. Функция

```
char *strcpy ( char *str1, const char *str2 );
```

копирует строку str2 в строку str1. Строка str2 копируется полностью, включая завершающий нулевой байт. Функция возвращает указатель на str1. Если строки перекрываются, то результат непредсказуем.

Функция

```
char *strncpy ( char *str1, const char *str2, size_t n );
```

копирует n символов из строки str2 в строку str1. Если строка str2 содержит меньше чем n символов, то последний нулевой байт копируется столько раз, сколько нужно

для расширения строки str2 до n символов. Функция возвращает указатель на строку str1.

Пример:

```
char  str1[80];
char str2 = "Copy string."; strcpy (
str1, str2 );
printf ( str1 ); // печатает: Copy string.
```

4. Соединение строк. Для соединения строк в одну строку используются функции strcat и strncat. Функция

```
char*  strcat ( char  *str1, const  char  *str2 );
```

присоединяет строку str2 к строке str1, причем завершающий нулевой байт строки str1 стирается. Функция возвращает указатель на строку str1.

Функция

```
char*  strncat ( char  *str1, const  char  *str2, size_t  n );
```

присоединяет n символов из строки str2 к строке str1, причем завершающий нулевой байт строки str1 стирается. Функция возвращает указатель на строку str1. если длина строки str2 меньше n, то присоединяются только символы, входящие в строку str2. После соединения строк к строке str1 всегда добавляется нулевой байт. Функция возвращает указатель на строку str1.

Пример:

```
#include <stdio.h>
#include <string.h> int
main ( )
{
char str1[80] = "String "; char
str2 = "catenation "; char str3 =
"Yes No"; strcat  ( str1, str2 );
printf ("%s\n", str1 );          // печатает: String catenation
strncat ( str1, str3, 3 );
printf ("%s\n", str1 );          // печатает: String catenation Yes return
1;
}
```

5. Поиск символа в строке. Для поиска символа в строке используются функции strchr, strrchr, strspn, strcspn и strpbrk. Функция

```
char*  strchr ( const  char  *str, int  c );
```

ищет первое вхождение символа, заданного параметром c, в строку str. В случае успеха функция возвращает указатель на первый найденный символ, а в случае неудачи – NULL.

Функция

```
char*  strrchr ( const  char  *str, int  c );
```

ищет последнее вхождение символа, заданного параметром c, в строку str. В случае успеха функция возвращает указатель на последний найденный символ, а в случае неудачи – NULL.

Пример:

```
#include <stdio.h>
#include <string.h> int
main ( )

{

char  str[80] = "Char search";

printf ("%s\n", strchr ( str, 'r' ));                // печатает: r search
printf ("%s\n", strrchr ( str, 'r' )); // печатает: rch return 1;

}
```

Функция

```
size_t  strspn ( const  char  *str1, const  char  *str2 );
```

возвращает индекс первого символа из строки str1, который не входит в строку str2.

Функция

```
size_t  strcspn ( const  char  *str1, const  char  *str2 );
```

возвращает индекс первого символа из строки str1, который входит в строку str2.

Пример:

```
int  main ( )

{

char  str[80] = "123 abc";

printf ("n = %d\n", strspn ( str, "321" ); // печатает: n = 3 printf ("n = %d\
n", strcspn ( str, "cba" ); // печатает: n = 4 return 1;

}
```

Функция

```
char*  strpbrk ( const  char  *str1, const  char  *str2 );
```

находит первый символ в строке str1, который равен одному из символов в строке str2. В случае успеха функция возвращает указатель на этот символ, а в случае неудачи – NULL.

Пример.

```
char  str[80] = "123 abc";

printf ("%s\n", strpbrk ( str, "bca" )); // печатает: abc
```

6. Сравнение строк. Для сравнения строк используются функция strstr.

Функция

```
char*  strstr ( const  char  *str1, const  char  *str2 );
```

находит первое вхождение строки str2 (без конечного нулевого байта) в строку str1. В случае успеха функция возвращает указатель на найденную подстроку, а в случае неудачи – NULL. Если указатель str1 указывает на строку нулевой длины, то функция возвращает указатель str1.

Пример:

```
char  str[80] = "123 abc 456;

printf ("%s\n", strstr ( str, "abc"); // печать: abc 456
```

7. Разбор строки на лексемы. Для разбора строки на лексемы используется функция strtok. Функция

```
char*  strtok ( char  *str1, const  char  *str2 );
```

возвращает указатель на следующую лексему (слово) в строке str1, в которой разделителями лексем являются символы из строки str2. В случае если лексемы закончились, то функция возвращает NULL. При первом вызове функции strtok параметр str1 должен указывать на строку, которая разбирается на лексемы, а при последующих вызовах этот параметр должен быть установлен в NULL. После нахождения лексемы функция strtok записывает после этой лексемы на место разделителя нулевой байт.

Пример.

```
 #include        <stdio.h>
#include  <string.h>  int
main( )

{

char str[ ] = "12 34 ab cd"; char *p;

p = strtok ( str, " " );
while (p)

 {

printf ( "%s\n", p ); // печатает в столбик значения: 12 34 ab cd p = strtok (
NULL, " " );

}

return 1;

}
```

8. Определение длины строки. Для определения длины строки используется функция strlen. Функция

```
size_t  strlen ( const  char  *str);
```

возвращает длину строки, не учитывая последний нулевой байт. Например, char str[] = "123";

```
printf ("len = %d\n", strlen ( str )); // печатает: len = 3
```

Пример выполнения задания.

Задание. Найти слова во введенной с клавиатуры строке, вывести их на экран и подсчитать их количество.

Схема программы представлена на рисунке 24:

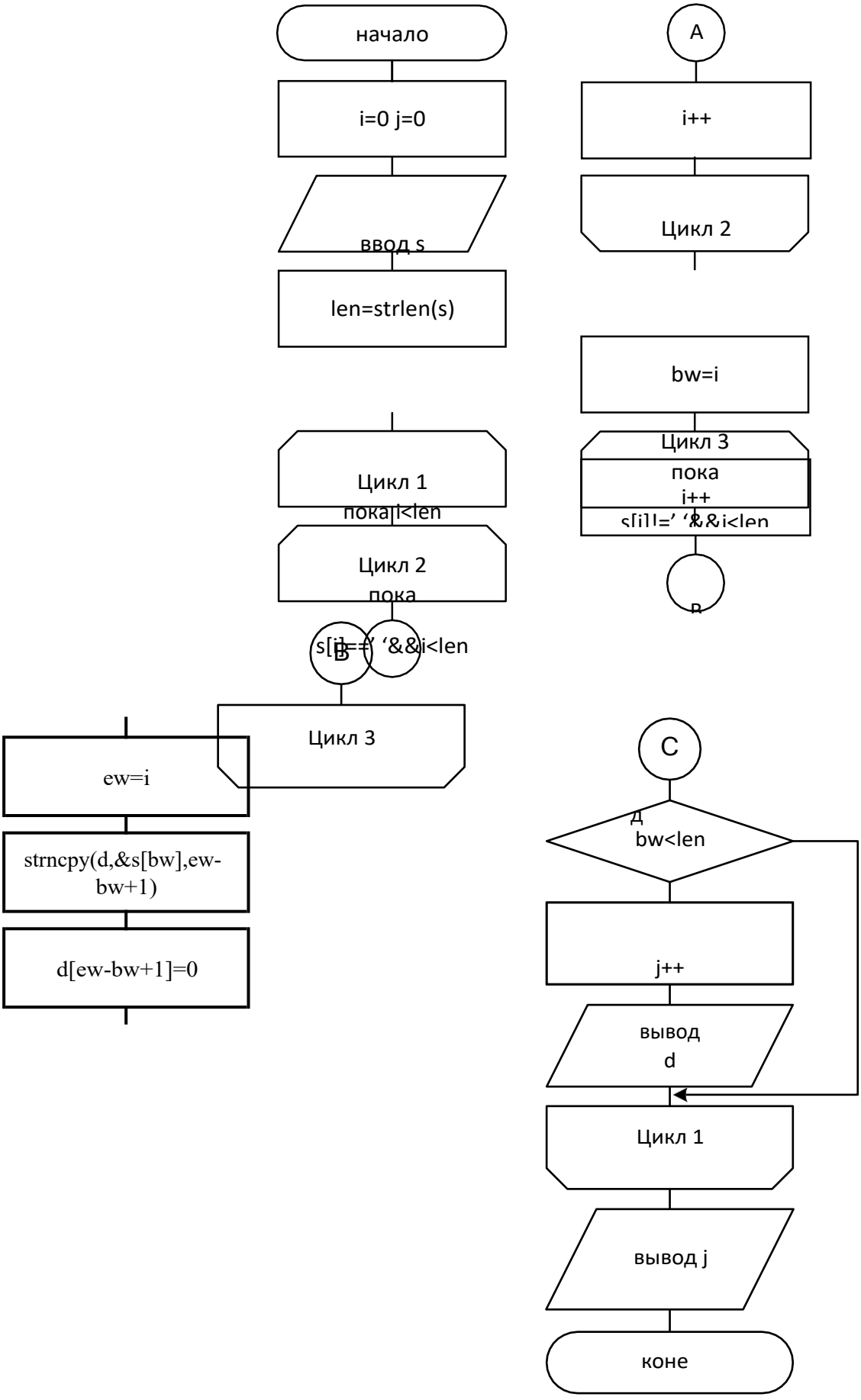

начало
i=0 j=0
ввод s
len=strlen(s)
Цикл 1
пока i<len
Цикл 2
пока
s[i]==' '&&i<len
B
Цикл 3
ew=i
strncpy(d,&s[bw],ew-bw+1)
d[ew-bw+1]=0
A
i++
Цикл 2
bw=i
Цикл 3
пока
i++
s[i]!=' '&&i<len
B
C
д
bw<len
j++
вывод d
Цикл 1
вывод j
коне

Рисунок 24

Пример программы:

```
#include<stdio.h>
#include<string.h>
```

```
vid main()
{
 char  s[100],d[100]; int
 i=0,j=0,bw,ew,len; gets(s);
 len=strlen(s); while (i<len)
 {
   while((s[i]==' ')&&(i<len)) i++; bw=i;
   while((s[i]!=' ')&&(i<len)) i++; ew=i;
   strncpy(d,&s[bw],ew-bw+1); d[ew-
   bw+1]=0;
   if (bw<len)
    { j++;
      printf("%s\n",d);}
 }
printf("Vsego slov %d\n", j);
}
```

Контрольные вопросы

1. Строки в языке С++: понятие строки, описание строк в программе, обращение к элементам строки.
2. Три способа ввода строк в С++.
3. Три способа вывода строк в С++.
4. Способы инициализации строк (задание значений в программе).
5. Стандартные функции для обработки строк.

Варианты заданий

Дана последовательность символов S_1, …, S_N. Группы символов, разделенные пробелом (одним или несколькими) и не содержащие пробелов внутри себя, будем называть словами.

1. Определить число символов в самом длинном слове строки. Слова отделяются знаком “/”.
2. В произвольном тексте выделить и отпечатать слова, начинающиеся с буквы А.

3. В произвольном тексте вставить между первым и вторым словом новое слово.

4. В произвольном тексте найти и отпечатать слова, содержащие букву Е.

5. Отпечатать второе и третье слова произвольного текста.

6. В произвольном тексте вставить между вторым и третьим словом новое слово.

7. В произвольном тексте найти и отпечатать все слова длиной 5 символов.

8. В произвольном тексте найти самое короткое слово.

9. В последовательности из 10 пятибуквенных слов найти и поменять местами пару слов, у которых первые три буквы одного совпадают с последними тремя буквами другого.

10. Упорядочить в алфавитном порядке последовательность из 10 пятибуквенных слов.

11. В строке из 50 символов отдельные слова разделены пробелом. Упорядочить строку так, чтобы каждое следующее слово было не короче предыдущего.

12. Расположить слова строки в порядке, обратным исходному.

13. Подсчитать количество букв ‘a’ в последнем слове строки.

14. Найти количество слов, у которых первый и последний символы совпадают между собой.

15. Исключить из строки слова, расположенные между скобками (,). Сами скобки должны быть исключены.

16. В произвольном тексте найти и отпечатать слова, содержащие букву А.

17. Отпечатать первое и второе слова произвольного текста.

18. В произвольном тексте вставить после первого слова новое слово.

19. В произвольном тексте найти и отпечатать все слова длиной 4 символа.

20. В произвольном тексте найти самое длинное слово.

21. Выполнить сравнение двух строк s и d. Результат вывести в виде сообщения «идентичны» или «не идентичны».

22. Добавить в конец строки новое слово, длинною 5 симвлов, иначе выдать сообщение об ошибке.

23. Добавить в начало строки новое слово, начинающееся с буквы *a,* иначе, если слово начинается с другой буквы вывести сообщение о невозможности добавления.

24. Посчитать какое количество раз встречается буква n (задается при каждом выполнении алгоритма).

25. Проанализировать массив символов, состоящий из n символов. Если массив состоит из n-5 символов, добавить в конец набор символов rrrrr.

Работа № 7

Тема: "Подпрограммы. Функции"

Цель работы: изучение правил составления и использования функций в программах на C++.

Краткие теоретические сведения.

В практике программирования часто складываются ситуации, когда одну и ту же группу операторов, реализующих определённую цель, требуется повторить без изменений в нескольких местах программы. Для избавления от столь нерациональной траты времени была предложена концепция подпрограммы.

Подпрограмма — именованная, логически законченная группа операторов языка, которую можно вызвать для выполнения любое количество раз из различных мест программы. В языке C++ подпрограммы реализованы в виде функций.

Функция — это поименованный набор описаний и операторов, выполняющих определённую задачу. Функция может принимать параметры и возвращать значение. Информация, передаваемая в функцию для обработки, называется параметром, а результат вычисления функции её значением. Обращение к функции называют вызовом.

Описание функции состоит из заголовка и тела функции:

```
тип имя_функции(список_переменных)
{
    тело_функции
}
```

Заголовок функции содержит:

– тип возвращаемого функцией значения, он может быть любым; если функция не возвращает значения, указывают тип void;

– имя_функции;

– список_переменных — перечень передаваемых в функцию величин (аргументов), которые отделяются друг от друга запятыми; для каждой переменной из списка указывается тип и имя; если функция не имеет аргументов, то в скобках указывают либо тип void, либо ничего.

Тело функции представляет собой последовательность описаний и операторов, заключённых в фигурные скобки.

В общем виде структура программы на C++ может иметь вид:

```
директивы компилятора
тип имя_1(список_переменных)
{
        тело_функции_1;
```

```
}
тип имя_2(список_переменных)
{
        тело_функции_2;
}
...
тип имя_n(список_переменных)
{
        тело_функции_n;
```

```
}

int main (список_переменных)

{

//Тело функции может содержать операторы вызова

//функций имя_1, имя_2, ...,
        имя_n
        тело_основной_функции;

}
```

Однако допустима и другая форма записи программного кода :

```
директивы компилятора

тип имя_1(список_переменных);
тип имя_2(список_переменных);

...

тип имя_n(список_переменных);
int main (список_переменных)

{

//Тело функции может содержать операторы вызова

// функций имя_1, имя_2, ...,
        имя_n
        тело_основной_функции;

}

тип имя_1(список_переменных)

{

        тело_функции_1;

}

тип имя_2(список_переменных)

{

        тело_функции_2;

}

...
```

```
тип имя_n(список_переменных)
{
        тело_функции_n;
}
```

Здесь функции описаны после функции main(), однако до неё перечислены заголовки всех функций. Такого рода опережающие заголовки называют прототипами функций. Прототип указывает компилятору тип данных, возвращаемых функцией, тип переменных, выступающих в роли аргументов, и порядок их следования. Прототипы используются для проверки правильности вызова функций в основной программе.

Вызвать функцию можно в любом месте программы. Для вызова функции необходимо указать её имя и в круглых скобках, через запятую перечислить имена или значения аргументов, если таковые имеются:

```
имя_функции(список_переменных);
```

Рассмотрим пример. Создадим функцию f(), которая не имеет входных значений и не формирует результат. При вызове этой функции на экран выводится строка символов "С Новым Годом, ".

```
#include <iostream>
using namespace std;

void f ( ) //Описание функции.

{

      cout << "С Новым Годом, ";
```

```
}

int main ( )

{

    f ( ); //Вызов функции.
    cout <<"Студент!" << endl;
    f ( ); //Вызов функции.

    cout <<"Преподаватель!" << endl;
    f ( ); //Вызов функции.

    cout <<"Народ!" << endl;

}
```

Результатом работы программы будут три строки:

```
С Новым Годом, Студент!

С Новым Годом, Преподаватель!
С Новым Годом, Народ!
```

Далее приведён пример программы, которая вычисляет значение выражения $\sin^2(\alpha)+\cos^2(\alpha)$ при заданном значении α. Здесь функция radian выполняет перевод градусной меры угла в радианную1.

```
#include <iostream>
#include <math.h>
#define PI 3.14159
using namespace std;

double radian ( int deg, int min, int sec )

{

    return ( deg * PI/180+min* PI /180/60+ sec * PI /180/60/60);

}

int main ( )

{

    int DEG, MIN, SEC; double RAD;

    //Ввод данных.

    cout<<" Input :"<<endl; //Величина угла:
    cout<<" DEG ="; cin >>DEG; //градусы,
    cout<<" MIN ="; cin >>MIN; //минуты,
    cout<<" SEC ="; cin >>SEC; //секунды.
```

```
    //Величина угла в радианах.

    RAD=radian (DEG, MIN, SEC); //Вызов функции.
    cout << " Value in radian A="<<RAD << endl;

    //Вычисление значения выражения и его
    вывод. cout << " sin (A) ^2+ cos (A) ^2= ";

    cout << pow( sin (RAD), 2 )+pow( cos (RAD), 2 ) << endl;
    return 0;

}
```

Переменные, описанные внутри функции, а также переменные из списка аргументов, являются локальными. Например, если программа содержит пять разных функций, в каждой из которых описана переменная N, то для C++ это пять различных переменных. Область действия локальной переменной не выходит за рамки функции. Значения локальных переменных между вызовами одной и той же функции не сохраняются.

Переменные, определённые до объявления всех функций и доступные всем функциям, называют глобальными. В функции глобальную переменную можно отличить, если не описана локальная переменная с теми же именем. Глобальные

переменные применяют для передачи данных между функциями, но это затрудняет отладку программы. Для обмена данными между функциями используют параметры функций и значения, возвращаемые функциями.

Пример выполнения задания.

Задание. Найти максимальную сумму элементов строк матрицы 3×5 с использованием функций.

Схемы программ функций ввода-вывода и обработки данных представлены на рисунках 25 и 26. Схема главной функции main представлена на рисунке 27:

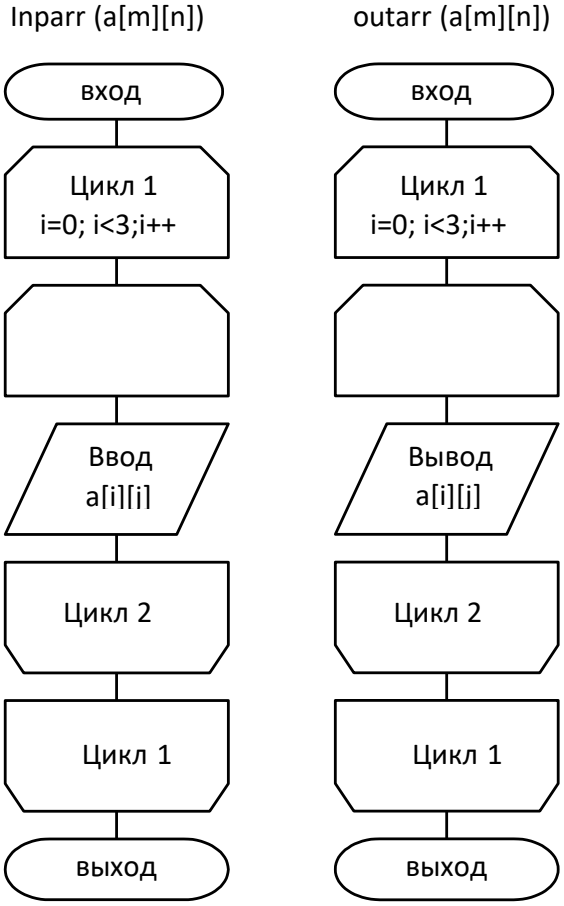


Рисунок 25 – Функции ввода и вывода массива данных

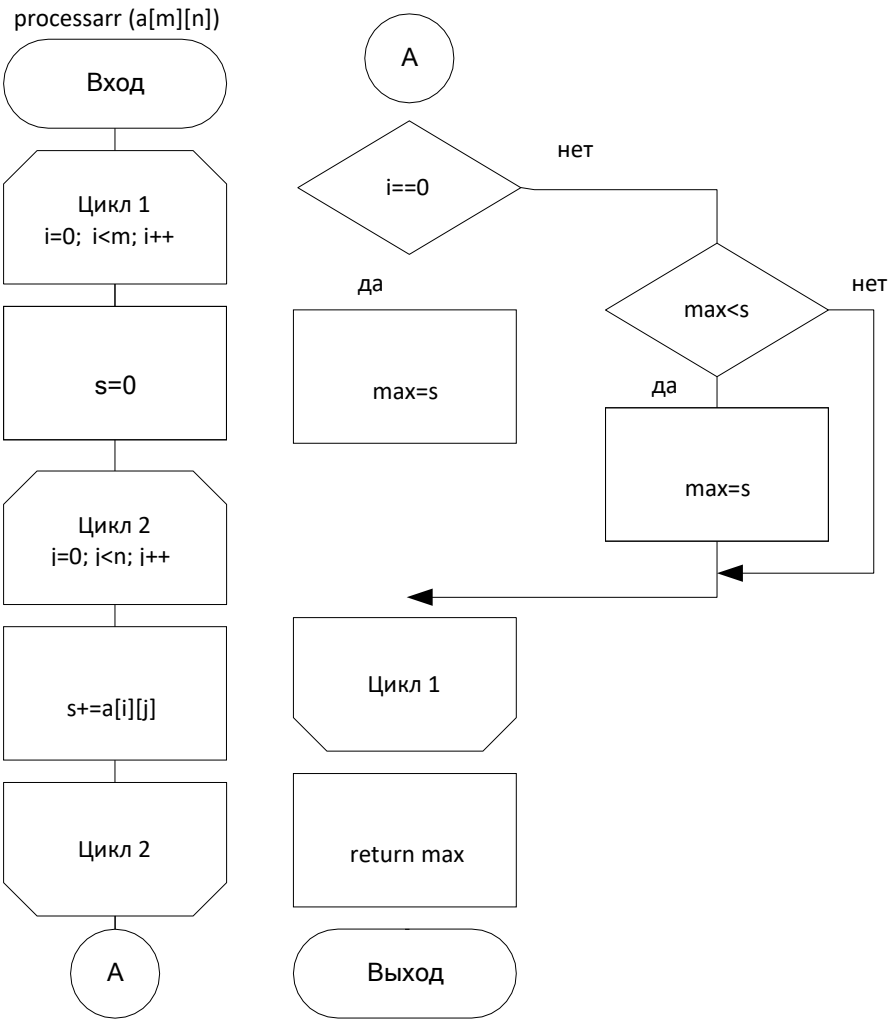
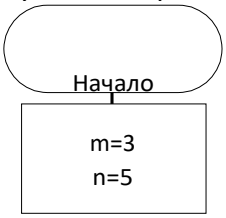


Рисунок 26 – Функция обработки массива данных



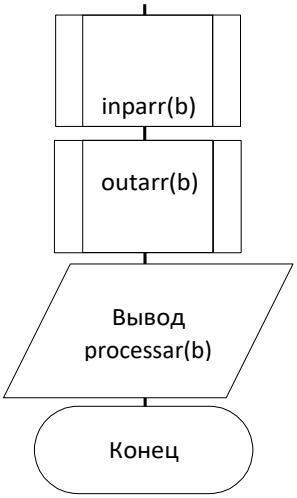


Рисунок 27 – Главная функция программы main

Пример программы:

```
#include <stdio.h>
const int m=3, n=5;

//функция ввода

void inparr(int a[m][n])

{

int i,j;

for (i=0;i<m;i++)
 for (j=0;j<n;j++)

 scanf("%d",&a[i][j]);

}

//функция вывода

void outarr (int a[m][n])

{

int i,j;
printf("Matrica:\
n"); for (i=0; i<m; i+
+)

{

for (j=0; j<n; j++)
printf("%5d", a[i][j]);
printf("\n");

}

}

int processarr(int a[m][n])

{

int i,j,s,max;
for(i=0;i<m;i++)

{

 s=0;
```

```
 for (j=0;j<n;j++)
   s+=a[i][j];

 if (i==0) max=s;

 else if (max<s) max=s;

}

return max;

}


void main()

{

int b[m][n];
inparr(b);
outarr(b);

printf("Maximalnaya summa stroki = %d", processarr(b));

}
```

Контрольные вопросы

1. Структура функции в языке C++, ее заголовок.
2. Вызов функции.
3. Способы передачи параметров.
4. Оператор return (2 формы записи).
5. Описание функции (прототип).

Варианты заданий

См. задание к лабораторной работе № 5.

2.3. Перечень вопросов для подготовки обучающихся к промежуточной аттестации (зачету)

1. Основы алгоритмизации и программирования
2. Методы формального описания алгоритмов. Схемы алгоритмов.
3. Основные характеристики алгоритмов и этапы их разработки.
4. Базовые разновидности программных алгоритмов. Принципы алгоритмизации. Разветвленные и циклические алгоритмы. Сложные циклы.
5. Алгоритмы с массивами. Взаимосвязь алгоритмов, моделей данных и постановок задач. Алгоритм и его программная реализация.
6. Понятие языка программирования.
7. Основные парадигмы программирования - процедурное, логическое, функциональное, объектно-ориентированное программирование.
8. Основные классификационные признаки и характеристики языков программирования. Синтаксис и семантика языка.
9. Понятие алгоритмического языка программирования и наиболее распространенные представители универсальных алгоритмических языков высокого уровня.
10. Система программирования и инструментальные средства поддержки основных этапов проектирования прикладных программных продуктов с использованием алгоритмического языка программирования.

11. Функциональное содержание процессов компиляции (трансляции, интерпретации) и построения загрузочных модулей, отладочных операций и тестирования.
12. Структурное программирование: общая характеристика языка C++
13. Место языка C++ в общей иерархии алгоритмических языков программирования.
14. Реализация языка для различных вычислительных платформ и операционных сред.
15. Интегрированная среда программирования системы MS Visual Studio C++.
16. Структурное программирование: структура программы на языке C++
17. Понятия программы, модуля, программной единицы. Общая структура программы. Пользовательские и библиотечные функции. Заголовочные файлы.
18. Препроцессор и его основные директивы.
19. Структурное программирование: основные элементы языка C++
20. Алфавит языка. Идентификаторы.
21. Ключевые слова и символы. Знаки операций. Синтаксис описания констант и переменных. Основные типы данных.
22. Структурное программирование: операции и выражения
23. Арифметические операции. Операции инкрементации и декрементации.

24. Логические операции и операции отношения. Операция условия.
25. Операция присваивания. Операция sizeof. Приоритет операций. Назначение выражений. Примеры выражений.
26. Структурное программирование: операторы управления
27. Основные виды операторов - операторы циклов, условных и безусловных переходов, оператор выбора. Вспомогательные операторы.
28. Простейшие операторы консольного ввода - вывода.
29. Структурное программирование: указатели, ссылки, массивы
30. Использование указателей как средства хранения адреса. Имена указателей. Операции над указателями. Оператор разыменования.
31. Использование оператора адреса (&) при работе со ссылками.
32. Возвращение значений с помощью ссылок.
33. Понятие массива. Синтаксис описания массивов. Обращение к элементам массива. Инициализация массивов.
34. Массивы и указатели.
35. Двумерные и одномерные массивы.
36. Ввод и вывод массивов.

Вопросы к экзамену

1. Понятие алгоритма.
2. Методы формального описания алгоритмов.
3. Схемы алгоритмов.
4. Основные характеристики алгоритмов и этапы их разработки.
5. Базовые разновидности программных алгоритмов.
6. Принципы алгоритмизации.
7. Разветвленные и циклические алгоритмы.
8. Сложные циклы.
9. Алгоритмы с массивами.
10. Взаимосвязь алгоритмов, моделей данных и постановок задач.
11. Алгоритм и его программная реализация.
12. Понятие языка программирования.
13. Основные парадигмы программирования - процедурное, логическое, функциональное, объектно-ориентированное программирование.
14. Основные классификационные признаки и характеристики языков программирования.
15. Синтаксис и семантика языка.
16. Понятие алгоритмического языка программирования и наиболее распространенные представители универсальных алгоритмических языков высокого уровня.
17. Система программирования и инструментальные средства поддержки основных этапов проектирования прикладных программных продуктов с использованием алгоритмического языка программирования.
18. Функциональное содержание процессов компиляции (трансляции, интерпретации) и построения загрузочных модулей, отладочных операций и тестирования.
19. Место языка C++ в общей иерархии алгоритмических языков программирования.
20. Реализация языка для различных вычислительных платформ и операционных сред.
21. Интегрированная среда программирования системы MS VS C++.
22. Понятия программы, модуля, программной единицы.
23. Общая структура программы.
24. Пользовательские и библиотечные функции.
25. Заголовочные файлы.
26. Препроцессор и его основные директивы.

20
27. Алфавит языка C++. Идентификаторы.
28. Ключевые слова и символы.
29. Знаки операций.
30. Синтаксис описания констант и переменных.
31. Основные типы данных.
32. Арифметические операции.
33. Операции инкрементации и декрементации.
34. Логические операции и операции отношения. Операция условия (?:). Операция присваивания.
35. Операция sizeof.
36. Приоритет операций.
37. Назначение выражений.
38. Операторы циклов,
39. Операторы условных и безусловных переходов,
40. Оператор выбора.
41. Вспомогательные операторы.
42. Операторы консольного ввода - вывода.
43. Использование указателей как средства хранения адреса.
44. Имена указателей.
45. Операции над указателями.
46. Оператор разыменования.
47. Использование оператора адреса (&) при работе со ссылками.
48. Возвращение значений с помощью ссылок.
49. Понятие массива. Синтаксис описания массивов.
50. Обращение к элементам массива. Инициализация массивов.
51. Массивы и указатели. Двумерные и одномерные массивы. Ввод и вывод массивов.
52. Переименование типов (typedel).
53. Перечисления (enum).
54. Структуры (struct).
55. Объединения (union).
56. Объявление и определение функций. Вызов функций.
57. Формальные и фактические параметры функций. Механизм передачи параметров по значению и по адресу.
58. Перегрузка функций.
21
59. Глобальные и локальные переменные. Область видимости и время жизни объектов.
60. Классы памяти.
61. Понятие рекурсии.
62. Модели памяти.
63. Статические и динамические данные.
64. Механизмы выделения, перераспределения и очистки динамической памяти.
65. Функции, поддерживающие основные операции с динамической памятью.
66. Операторы new и delete.
67. Динамические структуры данных.
68. Линейные списки, стеки, очереди, бинарные деревья.
69. Описание и внутреннее представление файлов.
70. Текстовые и бинарные файлы.
71. Базовые операции над файлами. Режимы доступа. Позиционирование в файле.
72. Библиотечные функции работы с файлами. Понятие потока.
73. Стандартные потоки в C++.
74. Функции работы с потоками

2.4. Перечень тем контрольных работ

1.Опишите структуру программы на языке С++.
2.Опишите императивную парадигму программирования. Приведите обзор языков.
3.Опишите объектно-ориентированную парадигму программирования. Приведите обзор языков.
4.Опишите функциональную парадигму программирования. Приведите обзор языков.
5.Опишите логическую парадигму программирования. Приведите обзор языков.
6.Целочисленные типы данных в языке С++.
7.Вещественные типы данных в языке С++.
10.Инструкция присваивания.
11.Приведение типов в языке С++.

3. Методические материалы, определяющие процедуру и критерии оценивания сформированности компетенций при проведении промежуточной аттестации

Критерии формирования оценок по ответам на вопросы, выполнению тестовых заданий

- оценка **«отлично»** выставляется обучающемуся, если количество правильных ответов на вопросы составляет 100 – 90 % от общего объёма заданных вопросов;

- оценка **«хорошо»** выставляется обучающемуся, если количество правильных ответов на вопросы – 89 – 76 % от общего объёма заданных вопросов;

- оценка **«удовлетворительно»** выставляется обучающемуся, если количество правильных ответов на тестовые вопросы –75–60 % от общего объёма заданных вопросов;

- оценка **«неудовлетворительно»** выставляется обучающемуся, если количество правильных ответов – менее 60 % от общего объёма заданных вопросов.

Критерии формирования оценок по результатам выполнения заданий

«Зачтено» – ставится за работу, выполненную полностью без ошибок и недочетов в соответствии с заданием. Обучающийся полностью владеет информацией по теме работы, решил все поставленные в задании задачи**.**

«Не зачтено» - ставится за работу, если обучающийся правильно выполнил менее 2/3 всего задания, использовал при выполнении неправильные алгоритмы, допустил грубые ошибки при программировании, сформулировал неверные выводы по результатам работы.

Виды ошибок:

- грубые ошибки: незнание основных понятий, правил, норм; незнание приемов решения задач; ошибки, показывающие неправильное понимание условия предложенного задания.

- негрубые ошибки: неточности формулировок, определений; нерациональный выбор хода решения.

- недочеты: нерациональные приемы выполнения задания; отдельные погрешности в формулировке выводов; небрежное выполнение задания.

Критерии формирования оценок по зачету

«Зачтено» - обучающийся демонстрирует знание основных разделов программы изучаемого курса: его базовых понятий и фундаментальных проблем; приобрел необходимые умения и навыки, освоил вопросы практического применения полученных знаний, не допустил фактических ошибок при ответе, достаточно последовательно и логично излагает теоретический материал, допуская лишь незначительные нарушения последовательности изложения и некоторые неточности.

«Не зачтено» - выставляется в том случае, когда обучающийся демонстрирует фрагментарные знания основных разделов программы изучаемого курса: его базовых понятий и фундаментальных проблем. У экзаменуемого слабо выражена способность к самостоятельному аналитическому мышлению, имеются затруднения в изложении материала, отсутствуют необходимые умения и навыки, допущены грубые ошибки и незнание терминологии, отказ отвечать на дополнительные вопросы, знание которых необходимо для получения положительной оценки.

Критерии формирования оценок по написанию и защите курсовой работы

«Отлично» (5 баллов) – получают обучающиеся студенты, оформившие курсовую работу в соответствии с предъявляемыми требованиями, в которой отражены все необходимые результаты проведенного анализа, сделаны обобщающие выводы и предложены рекомендации в соответствии с тематикой курсового проекта, а также грамотно и исчерпывающе ответившие на все встречные вопросы преподавателя.

«Хорошо» (4 балла) – получают обучающиеся, оформившие курсовой проект в соответствии с предъявляемыми требованиями, в которой отражены все необходимые результаты проведенного анализа, сделаны обобщающие выводы и предложены рекомендации в соответствии с тематикой курсового проекта. При этом при ответах на вопросы преподавателя обучающийся студент допустил не более двух ошибок.

«Удовлетворительно» (3 балла) – получают обучающиеся, оформившие курсовой проект в соответствии с предъявляемыми требованиями. При этом при ответах на вопросы преподавателя обучающийся студент допустил более трёх ошибок.

«Неудовлетворительно» (0 баллов) – ставится за курсовой проект, если число ошибок и недочетов превысило удовлетворительный уровень компетенции.

Критерии формирования оценок по экзамену

«Отлично» (5 баллов) – обучающийся демонстрирует знание всех разделов изучаемой дисциплины: содержание базовых понятий и фундаментальных проблем; умение излагать программный материал с демонстрацией конкретных примеров. Свободное владение материалом должно характеризоваться логической ясностью и четким видением путей применения полученных знаний в практической деятельности, умением связать материал с другими отраслями знания.

«Хорошо» (4 балла) – обучающийся демонстрирует знания всех разделов изучаемой дисциплины: содержание базовых понятий и фундаментальных проблем; приобрел необходимые умения и навыки, освоил вопросы практического применения полученных знаний, не допустил фактических ошибок при ответе, достаточно последовательно и логично излагает теоретический материал, допуская лишь незначительные нарушения последовательности изложения и некоторые неточности. Таким образом данная оценка выставляется за правильный, но недостаточно полный ответ.

«Удовлетворительно» (3 балла) – обучающийся демонстрирует знание основных разделов программы изучаемого курса: его базовых понятий и фундаментальных проблем. Однако знание основных проблем курса не подкрепляются конкретными практическими примерами, не полностью раскрыта сущность вопросов, ответ недостаточно логичен и не всегда последователен, допущены ошибки и неточности.

«Неудовлетворительно» (0 баллов) – выставляется в том случае, когда обучающийся демонстрирует фрагментарные знания основных разделов программы изучаемого курса: его базовых понятий и фундаментальных проблем. У экзаменуемого слабо выражена способность к самостоятельному аналитическому мышлению, имеются затруднения в изложении материала, отсутствуют необходимые умения и навыки, допущены грубые ошибки и незнание терминологии, отказ отвечать на дополнительные вопросы, знание которых необходимо для получения положительной оценки.