Министерство образования Республики Беларусь

БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ
ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Кафедра «Технология и методика преподавания»

КОНСТРУИРОВАНИЕ ПРОГРАММ И ЯЗЫКИ ПРОГРАММИРОВАНИЯ

Учебное пособие для студентов специальности
1-08 01 01 «Профессиональное обучение (по направлениям)»

Минск
БНТУ
2020

УДК 004.432 (075.8)
ББК 32.973 – 018.1
Д 75

Дробыш, А.А.
Д 75 Конструирование программ и языки программирования: указания к лабораторным работам для студентов специальности 1-08 01 01 «Профессиональное обучение (по направлениям)» направления специальности 1-08 01 01-07 «Профессиональное обучение (информатика)» / сост.: А.А. Дробыш. – Минск: БНТУ, 2019. – 45 с.

Издание содержит лабораторные работы, направленные на изучение языка программирования C++. Пособие предназначено для студентов специальности 1-08 01 01 «Профессиональное обучение (по направлениям)» направления специальности 1-08 01 01-07 «Профессиональное обучение (информатика)», а также может быть использовано для самостоятельного изучения языка программирования C++.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Лабораторные работы посвящены изучению синтаксических и семантических конструкций языка программирования высокого уровня C++.

В пособии рассматриваются принципы программирования разветвляющихся и циклических вычислительных процессов, обработки массивов и файлов, использования структурированных типов данных, подпрограмм с особенностями передачи в них параметров.

Приводятся различные приемы программирования, в том числе использование указателей и ссылок, динамически распределяемой памяти.

В каждой лабораторной работе приводится пример выполнения типового задания с учетом предъявляемых требований. Для получения глубоких знаний по теме необходимо внимательно ознакомиться с порядком выполнения работы и выполнить все указанные требования – от изучения теоретического материала до требований, предъявляемых к алгоритму решения задачи.

Основная цель лабораторных работ – сформировать у студентов фундаментальные основы знаний, необходимые при проектировании программ для вычислительных систем, привить навыки системного подхода к решению поставленной задачи, на практике познакомить с этапами решения практических задач на ЭВМ, научить оформлять сопроводительную документацию.

СОДЕРЖАНИЕ ОТЧЕТОВ

Каждая лабораторная работа включает в себя оформление соответствующей документации. В документации обязательно должны быть представлены следующие пункты:

1) текст индивидуального задания по варианту;

2) описания всех разработанных процедур и/или функций;

3) листинг программы решения задачи на языке высокого уровня C++;

4) ответы на контрольные вопросы;

5) выводы по работе.

Все представленные пункты должны быть отражены в отчете по каждой лабораторной работе.

ПОДГОТОВКА К ЛАБОРАТОРНЫМ РАБОТАМ

Для выполнения лабораторных работ потребуется среда разработки, поддерживающая язык C++. В данном пособии будем ориентироваться на Microsoft Visual Studio, версии не ниже 8.0.

Все лабораторные работы данного цикла ориентированы на создание консольных приложений.

Для создания проекта в среде Microsoft Visual Studio выполните следующее:

1. Запустите Microsoft Visual Studio.

2. Выберите в меню создание нового проекта: File-> New -> Project...

3. В открывшемся окне:

а) выберите Visual C++;

б) выберите *Пустой проект*;

в) в строке ввода Name введите имя проекта, в строке Location укажите место расположения будущего проекта на диске (рабочая директория проекта), в строке Solution name будет то же самое имя проекта, что и в Name.

4. Нажмите кнопку ОК.

В результате открывается совокупность окон редактора текста, дерева файлов проекта, отладчика и других, в которых располагается шаблон приложения.

В редакторе видим текст созданного файла с именем, как у проекта, и расширением .cpp. Для компиляции приложения выберите в меню *Build->Build Solution* (или нажмите F7).

Если компиляция выполнилась без ошибок, то в рабочей директории проекта будет создан исполняемый файл с именем проекта и расширением .exe. Если компилятор обнаружил ошибки, то их список будет располагаться в окне Output, там же появляется сообщение об успешной («Build succeeded»), либо неуспешной («Build FAILED») компиляции. Для отладки программы можно выполнять ее по шагам, для этого выберите в меню *Debug-> Step Over* (или Step Into) или нажмите F10 (F11).

ЛАБОРАТОРНАЯ РАБОТА № 1.

Основы программирования на языке С++

Цель работы: получить базовые умения работы в среде MS Visual Studio, написания простых консольных приложений.

Порядок выполнения работы: изучить теоретическую часть, выполнить практические задания, оформить отчет, осуществить защиту лабораторной работы.

Теоретическая часть

Структура программы на С++ состоящих из следующих элементов:

#include <заголовочный файл>

int main(){

операторы;

return 0;}

где, *#include <заголовочный файл>* – это директива препроцессора, которая представляет собой инструкцию, записанную в тексте программы на С++ и выполняемую до трансляции программы.

Директива *#include* включает в текст программы содержимое указанного файла.

Имеет две формы:

#include "имя_файла"

#include <имя_файла>

Если имя файла задано в угловых кавычках, то поиск файла производится в стандартных каталогах ОС, задаваемых командой PATH. Если же имя указано в кавычках, то поиск файла осуществляется в соответствии с заданным маршрутом, а при его отсутствии – в текущем каталоге.

int main(){

return 0;}

Это означает, что перед нами функция. Вся совокупность приведенных строк называется определением функции. Это определение состоит из двух частей: первой строки *int main ()*, которая называется заголовком функции и остальной части, заключенной в фигурные скобки, которая является телом функции.

Заголовок функции определяет интерфейс между функцией и остальной частью программы, а тело функции содержит инструкции для компьютера, т.е. определяет то, что собственно делает функция.

Заключительный оператор *return 0; (оператор возврата)*.

Синтаксис языка С++ требует, что определение функции main начиналось с int main(). Информация в круглых скобках называется списком аргументов или списком параметров.

Алфавит языка С++

В С++ используются следующие символы:

Знаки препинания: . , : ;

Скобки: () {} []

Знаки: | ^ ? _

Двойные и одинарные кавычки: “ ‘

Комментарии

Переменные

Каждая переменная имеет имя, тип, размер и значение.

Имя переменной в прямом смысле является ее названием. Имя переменной, или идентификатор, может состоять из латинских букв, цифр и символа подчеркивания.

Тип определяет, какие символы или числа записаны в ячейку памяти под этим именем.

Размер указывает максимальную величину или точность задания числа.

Описание переменных имеет вид:

имя_типа список_переменных

Примеры описаний:

char symbol, cc;

int number, row;

Типы данных

Данные в языке С++ разделяются на две категории: простые (скалярные), будем их называть базовыми, и сложные (составные) типы данных.

Основные типы базовых данных: целый *int* (integer), вещественный с одинарной точностью *float* и символьный -*char* (character).

В свою очередь, данные целого типа могут быть короткими *short*, длинными *long* и беззнаковыми *unsigned*, а вещественные с удвоенной точностью *double*.

Арифметические операции. К ним относятся:

– вычитание или унарный минус;

+ сложение или унарный плюс;

* умножение;

/ деление;

% деление по модулю;

++ унарная операция увеличения на единицу (инкремент);

-- унарная операция уменьшения на единицу (декремент).

Таблица 1. – Типы данных языка С++

Тип данных	Объем памяти (байт)	Диапазон значений
char	1	-128 …127
int	2	-32768…32767
short	2(1)	-32768…32767(-128…127)
long	4	-2147483648…2147483647
unsigned int	4	0…65535
unsigned long	4	0…4294967295
float	4	3,14*10-38…3,14*1038
double	8	1,7 *10-308 1,7 *10308

Пример 1. Программа расчета среднего арифметического трех введенных значений.

```
#include <iostream>
using namespace std;
int main ()
{
double a, b, c, sr;
cout<< "Input a, b, c";
cin>> a>>b>>c;
sr=(a+b+c)/3;
cout<<sr;
return 0;
}
```

Программы линейной структуры представляют собой одну из разновидностей базовых конструкций структурного программирования, так называемое следование.

Следованием называется конструкция, представляющая собой последовательное выполнение двух или более операторов (простых или составных).

Целью использования базовых конструкция является получение программы простой структуры. Такую программу легко читать, отлаживать и при необходимости вносить в нее изменения.

Пример 2.

$$\sin\sqrt{x+1} - y^3$$

```
#include <iostream>
#include <math.h>
using namespace std;
int main ()
{
double x, y, F
cout<<"x=";
cin>>x;
cout<<"y=";
F=sin(sqrt(x+1++-pow(y,3);
cout<<"F="<<F<<endl;
}
```

Для вычисления корней, степени, синуса, косинуса и т.д. в С++ используется библиотека math.h.

Функция вычисления степени

pow(число_возводимое_в_степень, степень)

Функция вычисления корня

sqrt(выражение)

Функция вычисления синуса

sin(число)

Условные операторы

Ветвление в простейшем случае описывается в языке С++ с помощью условного оператора, имеющего вид:

if (выражение) оператор_1;

else оператор_2;

где часть *else оператор_2* может и отсутствовать.

Сначала вычисляется «выражение» в скобках; если оно истинно то выполняется оператор_1. Если «выражение» ложно (равно нулю – NULL), то оператор_1 пропускается, а выполняется оператор_2. Если на месте условно выполняемых операторов должна располагаться группа из нескольких операторов языка, то они заключаются в фигурные скобки – { }.

Часто «выражение» в скобках представляет условие, заданное с помощью операций отношений и логических операций.

Операции отношения обозначаются в С++ следующим образом:

== равно;
!= не равно;
< меньше;
> больше;
<= меньше или равно;
>= больше или равно.

Символ ! в языке С++ обозначает логическое отрицание.

Есть еще две логические операции: || означает или, а && – логическое И. Операции отношения имеют приоритет ниже арифметических операций, так что выражение вида k>n%i вычисляется как k>(n%i).

Приоритет && выше, чем у ||, но обе логические операции выполняются после операций отношения и арифметических.

В сомнительных случаях лучше расставлять скобки.

Пример 3.

```
if (num < 10)
{ // если введенное число меньше 10
cout << "Это число меньше 10." << endl;
}
else
{ // иначе
cout << "Это число больше либо равно 10" << endl;
}
```

Условный оператор с одной ветвью:

```
if (num!= 10) // если введенное число не равно 10
cout << "Это число не равно 10." << endl;
```

Распространенные ошибки:

1. Использование в выражении при проверке равенства вместо оператора (==) простое присваивание (=).

2. Неверная запись проверки на принадлежность диапазону. Условие *0 < x < 1* необходимо записать следующим образом: *if (0 <x&&x< 1).*

Оператор множественного выбора switch

Общая форма оператора следующая:

```
switch(переменная выбора)
{
case const 1: операторы 1; break;
...
case const N: операторы N; break;
```

default: операторы N+1;
}

При использовании оператора switch сначала анализируется переменная выбора и проверяется, совпадает ли её значение со значением одной из констант. При совпадении выполняются операторы этого case. Конструкция default (может отсутствовать) выполняется, если результат выражения не совпал ни с одной из констант.

Пример 4.

```
int a=1;
switch(a)
{
   case 1:
      a++;
   case 2:
      a++;
   case 3:
      a++;
}
cout<<"a= "<<a;
```

Данная программа выведет a = 4.

Практическая часть

Задание 1. Создайте программу решения уравнения (x и y вводятся пользователем во время выполнения программы), учтите возможные условия арифметических операций.

Варианты заданий:

1. $\sqrt{x-y}/3x^2$
2. $\left(x^2-4x+y\right)/3$
3. $\sqrt{3x+y}/3x^2$
4. $\left(x^2-4y+y\right)/3$
5. $\sqrt{|x-y|}/3x$
6. $\left(x\ +4y-6\right)/3$
7. $\sqrt{10x-5}/3y$

8. $\left(x^2-7x+y\right)/y$
9. $\left(8x-4y-90\right)/x$
10. $x+\frac{3y}{x^2}$
11. $x^2+\frac{3y}{x}$
12. $\left(y^2-56\right)x\cdot y$
13. $x\cdot y-y^2+78/x$
14. $x\left(y-2\right)+\frac{3y}{x}$
15. $y\left(y-2\right)+\frac{3y}{x}$

Задание 2. Создайте программу решения уравнения вида $x^2-a\cdot x+b-c$, где:

1. A=5, b=7, c=7 – Это константы, x вводит пользователь с клавиатуры;
2. A=5, b=7 – Это константы, c и x вводит пользователь с клавиатуры;
3. A=5, c=7 – Это константы, b и x вводит пользователь с клавиатуры;
4. b=7, c=7 – Это константы, a и x вводит пользователь с клавиатуры;
5. A=15, b=7, c=27 – Это константы, x вводит пользователь с клавиатуры;
6. A=3, b=6 – Это константы, c и x вводит пользователь с клавиатуры;
7. A=1, c=2 – Это константы, b и x вводит пользователь с клавиатуры;
8. b=3, c=3 – Это константы, a и x вводит пользователь с клавиатуры;
9. A=11, b=22, c=33 – Это константы, x вводит пользователь с клавиатуры

10. A=7, b=8 – Это константы, c и x вводит пользователь с клавиатуры;
11. A=8, c=6 – Это константы, b и x вводит пользователь с клавиатуры;
12. b=4, c=4 – Это константы, a и x вводит пользователь с клавиатуры;
13. A=2, b=2, c=4 – Это константы, x вводит пользователь с клавиатуры;
14. A=4, b=3 – Это константы, c и x вводит пользователь с клавиатуры;
15. A=2, c=2 – Это константы, b и x вводит пользователь с клавиатуры.

Задание 3. Решите задачу (*Номер варианта определяется по списку группы кратно количеству вариантов в задании*).

1. Пользователь вводит порядковый номер пальца руки. Необходимо показать его название на экран
2. Дано значение n = 1..7, которое есть номером дня недели. По значению n определить, выходной этот день или рабочий. Результат вывести на экран.
3. Даны три целых числа *a, b, c*. Разработать программу, которая находит минимальное значение между этими числами. Результат вывести на экран.
4. Дан номер месяца года *n*. По номеру месяца определить, сколько дней в этом месяце. Фрагмент кода, который решает данную задачу. Принять, что в феврале 28 дней. Результат вывести на экран.
5. Даны три целых числа: A, B, C. Проверить истинность высказывания: «Справедливо двойное неравенство A < B < C». Результат вывести на экран.
6. Даны три целых числа. Найти количество положительных и количество отрицательных чисел в исходном наборе. Результат вывести на экран.
7. Даны три целых числа. Определите, сколько среди них совпадающих. Программа должна вывести одно из чисел: 3 (если все совпадают), 2 (если два совпадает) или 0 (если все числа различны).

Контрольные вопросы:

1. Простые типы данных языка C++.

2. Структура программы на языке С++.
3. Стандартные библиотеки и их подключение.
4. Что такое идентификатор, переменная, константа?
5. Что такое совместимость типов?
6. Состав языка С++.
7. Синтаксис условного оператора if.

ЛАБОРАТОРНАЯ РАБОТА № 2.
Массивы

Цель работы: получить навыки создания и обработки одномерных и двухмерных массивов.

Порядок выполнения работы: изучить теоретическую часть, выполнить практические задания, оформить отчет, осуществить защиту лабораторной работы.

Теоретическая часть

Наиболее часто работа с массивами организуется посредством циклов.

Оператор цикла for

Общий вид оператора:

for (инициализирующее выражение; условие; инкрементирующее выражение)

{

тело цикла

}

Инициализирующее выражение выполняется только один раз в начале выполнения цикла и, как правило, инициализирует счетчик цикла.

Условие содержит операцию отношения, которая выполняется в начале каждого цикла. Если условие равно 1 (true), то цикл повторяется, иначе выполняется следующий за телом цикла оператор. Инкрементирующее выражение, как правило, предназначено для изменения значения счетчика цикла.

Модификация счетчика происходит после каждого выполнения тела цикла.

Оператор цикла while

Оператор цикла с предусловием. Общий вид оператора:

while (условие)

{

тело цикла

}

Организует повторение операторов тела цикла до тех пор, пока условие истинно.

Оператор цикла do

Оператор цикла с постусловием. Общий вид оператора:

do {

тело цикла

}

while (условие);

Организует повторение операторов тела цикла до тех пор, пока условие истинно.

Массивы

Массив представляет нечто целое, что содержит целый набор однотипных пронумерованных элементов.

Объявляется создаваемый одномерный массив так:

int A[100]; //Массив A под 100 элементов тип массива int (т.е. это 100 целых чисел)

char S[256]; //Массив S типа char В массиве всего 256 элементов

В квадратных скобках указывается число элементов в массиве.

Присвоение первому элемента массива значения 333: *A[0]=333*;

Особенности работы с массивами:

Для создания массива компилятору необходимо знать тип данных и количество элементов в массиве.

Массивы могут иметь те же типы данных, что и простые переменные.

Квадратные скобки это своеобразный индикатор того, что происходит работа с массивом.

При объявлении массива, внутри квадратных скобок указывается число элементов для массива.

При использовании массива, внутри квадратных скобок указывается номер элемента из массива.

Номер элемента массива называется индексом массива.

Внешние и статические массивы можно инициализировать.

Автоматические и регистровые массивы инициализировать нельзя.

Любой массив требует такой же инициализации как и переменные, иначе в него может попасть информационный мусор.

Пример инициализации массива:

int A[10]={10,222,3,444,5,55,34,4,43,4};

Если у вас большой массив, но надо установить все элементы в нулевое значение, то не обязательно приписывать в фигурных скобках нули столько раз, сколько элементов в массиве, можно использовать простую конструкцию int A[1000]={0,};

Синтаксис С++ позволяет писать пустые квадратные скобки при объявлении массива. В этом случае компилятор сам определяет, сколько памяти нужно выделить массиву.

*int A[]={0,}; // Выделилось (1 байт * sizeof(int) = 2 байта) – под 2 элемента*

*int A[]={0,0,0,};//Выделилось (3 байта * sizeof(int) =12 байт) – под 4 элемента*

Узнать сколько байт съедает один элемент из массива, можно командой *sizeof(A[0])***.** Узнать сколько элементов может поместиться в массив командой *sizeof(A)/sizeof(A[0])***;** – применимо именно к массиву, у указателя узнать сколько вмещает элементов указатель не выйдет.

Пример 1. Создать массив из 10 чисел и вывести его на экран

```
#include <iostream>
using namespace std;
void main ()
{
const int N=10; //размер обычного массива можно определить константой
int A[N]={0,}; //массив из 10 элементов, каждый элемент равен 0
A[5]=100; // шестому элементу присвоили значение 100
for (int i=0;i<N; i++) cout<<A[i]<<" "; // вывод массива на экран поэлементно
cin.get();
}
```

Пример 2. Удалить из одномерного массива все отрицательные элементы

```
for (i=0; i<N; i++)
if (a[i]<0
{
for (j=i+1; j<N; j++) a[j-1]=a[j];
n--; i--;
}
```

C++ позволяет создавать многомерные массивы. Простейшим видом многомерного массива является двумерный массив. Двумерный массив - это массив одномерных массивов. Двумерный массив объявляется следующим образом:

тип имя_массива[размер второго измерения][размер первого измерения];

Следовательно, для объявления двумерного массива целых с размером 10 на 20 следует написать:

int d[10] [20];

Посмотрим внимательно на это объявление. В противоположность другим компьютерным языкам, где размерности массива отделяются запятой, C++ помещает каждую размерность в отдельные скобки.

Для доступа к элементу с индексами 3, 5 массива d следует использовать:

d[3] [5]

Двумерные массивы сохраняются в виде матрицы, где первый индекс отвечает за строку, а второй – за столбец. Это означает, что правый индекс изменяется быстрее левого, если двигаться по массиву в порядке расположения элементов в памяти.

Число байт в памяти, требуемых для размещения двумерного массива, вычисляется следующим образом:

*число байт = размер второго измерения * размер первого измерения * sizeof (базовый тип)*

Предполагая наличие в системе 2-байтных целых, целочисленный массив с размерностями 10 на 5 будет занимать 10×5×2, то есть 100 байт.

Пример 3. В двумерный массив заносятся числа от 1 до 12, после чего массив выводится на экран.

```
#include <stdio.h>
int main(void)
{
int t,i, num[3][4];
/* загрузка чисел */
for(t=0; t<3; ++t)
for (i=0; i<4; ++i)
num[t][i] = (t*4)+i+1;
/* вывод чисел */
for (t=0; t<3; ++t)
```

```
{
for (i=0; i<4; ++i)
printf("%d  ",num[t][i]);
printf ("\n");
}
return 0;
}
```

Практическая часть

Задание 1. Выполните задание согласно варианта.

1. Задан массив из k символов. Преобразовать массив следующим образом: сначала должны стоять цифры, входящие в массив, а затем все остальные символы. Взаимное расположение символов в каждой группе не должно изменяться.

2. Задан массив из k символов. Преобразовать массив следующим образом: расположить символы в обратном порядке.

3. Задан массив из k чисел. Найти число, наиболее часто встречающееся в этом массиве.

4. Задан массив из k чисел. Отсортировать элементы массива по возрастанию.

5. Задан массив из k чисел. Найти числа, входящие в массив только один раз.

6. Задан массив из k чисел. Сдвинуть элементы массива циклически на n позиций влево.

7. Задан массив из k чисел. Сдвинуть элементы массива циклически на n позиций вправо.

8. Задан массив из k чисел. Преобразовать массив следующим образом: все отрицательные элементы массива перенести в начало, а все остальные – в конец, сохранив исходное взаимное расположение, как среди отрицательных, так и среди положительных элементов.

9. Задан массив из k символов. Создать два новых массива: в первый перенести все цифры из исходного массива, во второй – все остальные символы.

10. Задан массив из k символов. Определить, симметричен ли он, т. е. читается ли он одинаково слева направо и справа налево.

11. Задано два массива. Найти наименьшие среди элементов первого массива, которые не входят во второй массив.

12. Задан массив из k чисел. Определить количество инверсий в массиве (т. е. таких пар элементов, в которых большее число находится слева от меньшего).

13. Задан массив из k символов. Удалить из него повторные вхождения каждого символа.

14. Задан массив из k символов. Определить количество различных элементов в массиве.

15. Задан массив из k символов латинского алфавита. Вывести на экран в алфавитном порядке все символы, которые входят в этот массив по одному разу.

Задание 2. Создайте приложение в котором пользователь вводит двумерный массив размером N×M, выводит его на экран, а так же выполняет дополнительное действие, описанное ниже:

1. N=3, M=4, выводит сумму элементов второй строки.
2. N=4, M=3, выводит сумму элементов первой строки.
3. N=3, M=4, выводит сумму элементов первого столбца.
4. N=3, M=4, выводит сумму элементов второго столбца.
5. N=5, M=2, выводит сумму элементов третьей строки.
6. N=4, M=2, выводит сумму элементов четвертой строки.
7. N=5, M=5, выводит сумму элементов третьего столбца.
8. N=5, M=5, выводит сумму элементов четвертого столбца.
9. N=4, M=3, выводит сумму элементов второй строки.
10. N=3, M=4, выводит сумму элементов первой строки.
11. N=4, M=3, выводит сумму элементов первого столбца.
12. N=4, M=3, выводит сумму элементов второго столбца.
13. N=5, M=3, выводит сумму элементов третьей строки.
14. N=5, M=2, выводит сумму элементов четвертой строки.
15. N=5, M=4, выводит сумму элементов третьего столбца.

Задание 3. Выполните задание согласно варианта.

1. Дана прямоугольная таблица, которая содержит не более 10 строк и не более 10 столбцов. Найти сумму элементов, которые расположены в строках с нечетными номерами.

2. Дан двумерный массив, который содержит не более 10 строк и не более 10 столбцов. Найти максимальный по абсолютной величине элемент и поменять его местами с последним элементом массива.

3. Даны оценки, полученные на 4 экзаменах во время сессии студентами одной группы, по 10 бальной системе. Определить сколько студентов не сдали сессию.

4. Дана прямоугольная таблица, которая содержит не более 10 строк и не более 10 столбцов. Найти сумму элементов, которые делятся на данное число X.

5. Дан двумерный массив, который содержит не более 10 строк и не более 10 столбцов. Найти минимальный элемент и поменять его местами с тем элементом, который стоит в конце массива.

6. Даны оценки, полученные на 4 экзаменах во время сессии студентами одной группы, по 10 бальной системе. Определить сколько студентов сдали сессию на 10 и 9 баллов.

7. Дана прямоугольная таблица, которая содержит не более 10 строк и не более 10 столбцов. Найти сумму элементов, у которых сумма делителей меньше данного числа X.

8. Дан двумерный массив, который содержит не более 10 строк и не более 10 столбцов. Создать новый массив, элементами которого являются суммы цифр каждого числа старого массива.

9. Даны оценки, полученные на 4 экзаменах во время сессии студентами одной группы, по 10 бальной системе. Определить сколько студентов сдали сессию только на 7.

10. Дана прямоугольная таблица, которая содержит не более 10 строк и не более 10 столбцов. Найти сумму отрицательных и сумму положительных элементов и сравнить их по модулю.

11. Дан двумерный массив, который содержит не более 10 строк и не более 10 столбцов. Создать новый массив, элементами которого являются суммы делителей каждого числа старого массива.

12. Даны оценки, полученные на 4 экзаменах во время сессии студентами одной группы, по 10 бальной системе. Определить сколько студентов сдали сессию на балл не ниже 6.

13. Дана прямоугольная таблица, которая содержит не более 10 строк и не более 10 столбцов. Найти и вывести те элементы, которые больше предыдущего, стоящего с ним в одной строке.

14. Дан двумерный массив, который содержит не более 10 строк и не более 10 столбцов. Создать новый массив, элементами которого являются суммы первой и последней цифры каждого числа старого массива.

15. Есть группа спортсменов из 7 человек. Для каждого спортсмена приводится его рост и вес. Вес спортсмена считается нормальным, если от роста отнять 100 и полученное число отличается от веса не более чем на 3. Вывести номера тех спортсменов, чей вес превышает норму.

Задание 4. Выполните задание согласно варианта.

1. Сформируйте массив L(I, J) с помощью датчика случайных чисел. Увеличить каждый элемент массива в 3 раза и поменяйте знак на противоположный. Массив выведите на экран в виде таблицы.
2. Дана матрица целых чисел размера N×M. Вывести номер строки, содержащей минимальное число одинаковых элементов.
3. Дана целочисленная прямоугольная матрица размера M×N. Сформировать одномерный массив, состоящий из элементов, лежащих в интервале [1, 10]. Найти произведение элементов полученного одномерного массива.
4. Дана целочисленная матрица размера 8×5. Определить: а) сумму всех элементов второго столбца массива; б) сумму всех элементов 3-й строки массива.
5. Дан двумерный массив. Вставьте первую строку после строки, в которой находится первый встреченный минимальный элемент.
6. Вычислить средние арифметические значения неотрицательных элементов каждой строки матрицы [N; M].
7. Преобразовать исходную матрицу A(M×N) так, чтобы последний элемент каждой строки был заменен суммой предыдущих элементов той же строки.
8. Задана матрица A (n, n) действительных чисел. «Перевернуть» в ней главную и побочную диагонали (переписать цифры в обратном порядке). Для того, чтобы «перевернуть» диагонали, не надо перебирать всю матрицу. Достаточно перебрать половину
9. В матрице найти элементы (их позицию), которые являются одновременно минимальными в строке и столбце.
10. В заданной матрице найти произведение ненулевых элементов.
11. Задан двухмерный массив чисел. Значение элементов матрицы вводятся с клавиатуры. Вычислить сумму элементов матрицы, индексы которых составляют в сумме заданное число K (это число вводится пользователем). Вывести результат.

12. Задан двухмерный массив чисел. Значение элементов матрицы вводятся с клавиатуры. Определить сумму одинаковых элементов матрицы и вывести те из них, которые находятся на нечетных столбцах. Вывести результат.

13. Посчитать суммы каждой строки и каждого столбца матрицы. Вывести суммы строк в конце каждой строки, а суммы столбцов под соответствующими столбцами.

14. Матрицу 10×20 заполнить случайными числами от 0 до 15. Вывести на экран саму матрицу и номера строк, в которых число 5 встречается три и более раз.

15. Заполнить матрицу случайными целыми неотрицательными числами. Вывести на экран матрицу в табличном виде так, чтобы сначала шли нечетные числа, затем четные. Также вывести количество четных и нечетных чисел в матрице.

Контрольные вопросы:

1. Что такое массив?
2. Опишите, как объявляется массив
3. Опишите, что такое инициализация массива и как она выполняется.
4. Какие типы данных может содержать массив?
5. Какие отличия имеет двухмерный массив от одномерного?
6. Как вычисляется число байт, требуемых для хранения в памяти массива?

ЛАБОРАТОРНАЯ РАБОТА № 3.
Перечисления. Структуры. Объединения

Цель работы: получить навыки работы с перечислениями, структурами и объединениями.

Порядок выполнения работы: изучить теоретическую часть, выполнить практические задания, оформить отчет, осуществить защиту лабораторной работы.

Теоретическая часть
Перечисления

Перечисления (enum) представляют еще один способ определения своих типов. Их отличительной особенностью является то, что они содержат набор числовых констант.

Определим простейшее перечисление:

```
enum seasons
{
spring,
summer,
autumn,
winter
};
```

Для определения перечисления применяется ключевое слово enum, после которого идет название перечисления. Затем в фигурных скобках идет перечисление констант через запятую. Каждой константе по умолчанию будет присваиваться числовое значение, начиная с нуля. То есть в данном случае spring=0, а winter=3.

Пример 1. Используем перечисление

```
#include <iostream>
enum seasons
{
spring,
summer,
autumn,
winter
};
int main()
{
seasons currentSeason = autumn;
std::cout << "Season: " << currentSeason << std::endl;
return 0;
}
```

Мы можем определить переменную типа seasons и присвоить этой переменной значение одной из констант, объявленных в перечислении. Но фактически это будет числовое значение. В частности, консольный вывод данной программы: *Season: 2*.

В то же время перечисления – это отдельный тип, поэтому мы не можем присвоить переменной напрямую числовое значение:

```
seasons currentSeason = 2;  // ошибка
```

Если нас не устраивают значения по умолчанию для констант, то мы можем явным образом задать значения. Например, установить начальное значение:

```
enum seasons
```

```
{
spring = 1,
summer, //2
autumn, //3
winter  //4
};
```

В этом случае значения второй и последующих констант будет увеличиваться на единицу.

Также можно задать значение для каждой константы:

```
enum seasons
{
spring = 1,
summer = 2,
autumn = 4,
winter = 8
};
```

Перечисления могут использоваться, когда у нас есть ряд логически связанных констант, которые естественно лучше определить в одном общем типе данных.

Структуры

Структура представляет собой переменную, группирующую связанные части информации, называемые элементами, типы которых могут различаться. Группируя данные в одну переменную подобным образом, вы упрощаете ваши программы, снижая количество переменных, которыми необходимо управлять, передавать в функции и т. д.

Концепции структур:

– Структуры позволяют вашим программам группировать в одной переменной связанные данные, типы которых могут различаться.

– Структура состоит из одной или нескольких частей данных, называемых элементами.

– Для определения структуры внутри программы следует указать имя структуры и ее элементы.

– Каждый элемент структуры имеет тип, например char, int и float, и имя каждого элемента должно быть уникальным.

– После того как ваша программа определит структуру, она может объявить переменные типа этой структуры.

– Для изменения элементов структуры внутри функции ваши программы должны передать структуру в функцию с помощью адреса.

Объявление структуры

Структура определяет шаблон, с помощью которого ваша программа может позднее объявить одну или несколько переменных. Другими словами, ваша программа сначала определяет структуру, а затем объявляет переменные типа этой структуры. Для определения структуры ваши программы используют ключевое слово struct, за которым обычно следует имя и левая фигурная скобка. Следом за открывающей фигурной скобкой вы указываете тип и имя одного или нескольких элементов. За последним элементом вы размещаете правую закрывающую фигурную скобку. В этот момент вы можете (необязательно) объявить переменные данной структуры:

```
struct name
{
int member_name_1 ;// Объявления элементов структуры
float member_name_2;
 } variable; //Объявление переменной
```

Пример 1. Объявление и заполнение структуры.

```
#include <iostream>
using namespace std;
struct Struct{
int a;
};
void main(){
Struct a;
a.a=4;
cout<<a.a<<endl;
}
```

Пример 2. Объявление и заполнение структуры.

```
struct database {
int id_number;
int age;
float salary;
};
int main()
{
```

```
database employee;
employee.age = 22;
employee.id_number = 1;
employee.salary = 12000.21;
}
```

Структура database содержит три переменные доступ к которым осуществляется с помощью оператора «.».

Указатели

Если вы работаете с указателем на структуру, то для доступа к переменным надо использовать оператор «->» вместо точки. Все свойства указателей не изменяются.

Пример 3

```
#include <iostream>
using namespace std;
struct xampl {
int x;
};
int main()
{
xampl structure;
xampl *ptr;
structure.x = 12;
ptr = &structure;
cout<< ptr->x;
cin.get();
}
```

Объединения

Объединение – это объект, позволяющий нескольким переменным различных типов занимать один участок памяти. Объявление объединения похоже на объявление структуры:

```
union union_type {
int i; char ch;
};
```

Когда объявлено объединение, компилятор автоматически создает переменную достаточного размера для хранения наибольшей переменной, присутствующей в объединении.

Для доступа к членам объединения используется синтаксис, применяемый для доступа к структурам – с помощью операторов «точка» и

«стрелка». Чтобы работать с объединением напрямую, надо использовать оператор «точка». Если к переменной объединения обращение происходит с помощью указателя, надо использовать оператор «стрелка».

Например, для присваивания целого числа 10 элементу i объединения *cnvt* следует написать: *cnvt.i = 10*;

Использование объединений помогает создавать машинно-независимый (переносимый) код. Поскольку компилятор отслеживает настоящие размеры переменных, образующих объединение, уменьшается зависимость от компьютера. Не нужно беспокоиться о размере целых или вещественных чисел, символов или чего-либо еще.

Объединения часто используются при необходимости преобразования типов, поскольку можно обращаться к данным, хранящимся в объединении, совершенно различными способами.

Практическая часть

Задание 1. Создайте программу, реализующую работу со структурой (ввод-вывод данных) согласно варианта.

1. Структура «Класс»: Номер по списку, Фамилия ученика, Имя ученика, Отчество ученика, адрес жительства, домашний телефон.
2. Структура «Кладовка»: Код товара, Наименование товара, количество штук, вес одной штуки.
3. Структура «Прокат DVD»: код диска, Название диска, количество штук, стоимость проката, год выпуска.
4. Структура «Расписание занятий»: Код дня недели, День недели, 1-я пара, 2-я пара, 3-я пара.
5. Структура «Библиотека»: Код книги, название книги, автор, год издания.
6. Структура «Учебная группа»: Номер по списку, Фамилия студента, Имя студента, Отчество студента, адрес жительства, мобильный телефон.
7. Структура «Класс»: Номер по списку, Фамилия ученика, Имя ученика, Отчество ученика, адрес жительства, домашний телефон.
8. Структура «Кладовка»: Код товара, Наименование товара, количество штук, вес одной штуки.
9. Структура «Прокат DVD»: код диска, Название диска, количество штук, стоимость проката, год выпуска.
10. Структура «Расписание занятий»: Код дня недели, День недели, 1-я пара, 2-я пара, 3-я пара.

11. Структура «Библиотека»: Код книги, название книги, автор, год издания.

12. Структура «Учебная группа»: Номер по списку, Фамилия студента, Имя студента, Отчество студента, адрес жительства, мобильный телефон.

13. Структура «Класс»: Номер по списку, Фамилия ученика, Имя ученика, Отчество ученика, адрес жительства, домашний телефон.

14. Структура «Кладовка»: Код товара, Наименование товара, количество штук, вес одной штуки.

15. Структура «Прокат DVD»: код диска, Название диска, количество штук, стоимость проката, год выпуска.

Контрольные вопросы:

1. Понятия «структура», «перечисление» и «объединение».
2. Опишите концепции структур.
3. Перечислите операторы для доступа к структурам, объединениям.

ЛАБОРАТОРНАЯ РАБОТА № 4.
Битовые поля. Указатели

Цель работы: получить навыки работы с битовыми полями.

Порядок выполнения работы: изучить теоретическую часть, выполнить практические задания, оформить отчет, осуществить защиту лабораторной работы.

Теоретическая часть

Битовое поле – это своеобразная структура, которая позволяет работать с отдельными битами. Причины использования битовых полей:

Если ограничено место для хранения информации, можно сохранить несколько логических (истина/ложь) переменных в одном байте.

Некоторые интерфейсы устройств передают информацию, закодировав биты в один байт.

Некоторым процедурам кодирования необходимо получить доступ к отдельным битам в байте.

Синтаксис объявления битовых полей следующий:

```
struct [имя_структуры] {
 тип [имя_битового_поля1]: длина;
 тип [имя_битового_поля2]: длина;
 . . . . .
```

тип [имя_битового_поляN]: длина;
} [имя_объекта];

В языке С в качестве типа битовых полей обязательно нужно использовать *int* или *unsigned* или *signed* . В С++ разрешено использовать кроме перечисленных выше любой тип, интерпретируемый как целый: *char, bool, short, long* и перечисления. Длина битового поля задается целочисленным значением, которое определяет, сколько битов необходимо выделить указанному полю.

Пример 1. Битовое поле:

```
struct {
unsigned name1: 1;
unsigned name2: 3;
unsigned name3: 5;
} obj;
obj.name1 = 1;
obj.name3 = 30;
```

В приведенном примере мы создали три битовых поля. Для хранения элемента *name1* выделено 1 бит, для хранения элемента *name2* - 3 бита, а для хранения *name3* выделено 5 бит. После объявления битовых полей в нашем примере происходит присвоение значений битовым полям. При этом удостоверьтесь, что присваиваемые значения не превышают размер поля.

Размер одного битового поля не должен превышать размер типа данного поля. То есть, если битовое поле объявлено как *unsigned* или *int*, то размер такого поля не должен превышать 32 бита.

В следующем примере объявляется структура, которая содержит битовые поля:

```
struct Date {
unsigned short nWeekDay  : 3;    // 0..7   (3 bits)
unsigned short nMonthDay : 6;    // 0..31  (6 bits)
unsigned short nMonth    : 5;    // 0..12  (5 bits)
unsigned short nYear     : 8;    // 0..100 (8 bits)
};
```

Ссылка на битовое поле выполняется по *имени_битового_поля*. Если имя поля не указано, запрошенные биты все равно выделяются, но доступ к ним невозможен. Такое поле называется неименованным битовым полем. Существует множество ситуаций, в которых оправдано использование неименованных битовых полей: они ничем не хуже

неименованных параметров функций объявление которых можно отделить от определения (инициализации) случаи:

float sum (float a, float b) и *float sum (float, float);*.

```
struct Example1 {
nsigned n1: 6;
unsigned : 2;
unsigned n2: 14;
unsigned : 2;
unsigned n3: 5;
};
```

Неименованные битовые поля с шириной поля 0 задают выравнивание следующего поля по границе максимального типа элементов структуры.

Пример 2. Структура битовых полей.

```
struct Example2 {
unsigned n1: 12;
unsigned : 0;
unsigned n2: 8;
};
```

использует неименованное поле нулевой ширины, чтобы пропустить оставшиеся биты в том элементе памяти, в котором хранится *n1*, и выровнять *n2* по границе следующего элемента памяти. Элементом памяти выступает в данном случае 4 байта, то есть размер *unsigned int*. В общей сложности данная структура будет занимать в памяти 8 байт.

К битовым полям не может быть применена операция получения адреса (&) и поэтому не существует указателей на поля.

Доступ к элементам битового поля осуществляется так же, как и доступ к обычным членам структуры. Например:

```
#include struct demo
{
unsigned a: 1;
signed b: 3;
int c: 6; };
int main () {
struct demo s;
s.a = 1;
s.b = -1;
s.c = -4;
```

```
printf("s.a = %u\n", s.a); // печать: s.a = 1
printf("s.b = %d\n", s.b); // печать: s.b = -1
printf("s.c = %d\n", s.c); // печать: s.c = -4
return 1;
}
```

Указатели

Указатели представляют собой объекты, значением которых служат адреса других объектов (переменных, констант, указателей) или функций. Как и ссылки, указатели применяются для косвенного доступа к объекту. Однако в отличие от ссылок указатели обладают большими возможностями.

Для определения указателя надо указать тип объекта, на который указывает указатель, и символ звездочки *. Например, определим указатель на объект типа *int*: *int *p;*

Пока указатель не ссылается ни на какой объект. При этом в отличие от ссылки указатель необязательно инициализировать каким-либо значением. Теперь присвоим указателю адрес переменной:

```
int x = 10;     // определяем переменную
int *p;         // определяем указатель
p = &x;         // указатель получает адрес переменной
```

Для получения адреса переменной применяется операция &. Что важно, переменная x имеет тип *int*, и указатель, который указывает на ее адрес, тоже имеет тип *int*. То есть должно быть соответствие по типу.

Если мы попробуем вывести адрес переменной на консоль, то увидим, что он представляет шестнадцатиричное значение:

```
#include <iostream>
int main()
{
int x = 10;     // определяем переменную
int *p;         // определяем указатель
p = &x;         // указатель получает адрес переменной
std::cout << "p = " << p << std::endl;
return 0;
}
```

Практическая часть

Задание 1. Создайте структуру с битовыми полями, описывающую текущий момент времени от секунды до года. Обеспечьте ввод/вывод информации.

Задание 2. Решите задачу.

1. Ввести значение 2-х целых переменных a и b. Направить два указателя на эти переменные. С помощью указателя увеличить значение переменной a в 2 раза. Затем поменять местами значения переменных a и b через их указатели.

2. Ввести значение 2-х целых переменных a и b. Направить два указателя на эти переменные. С помощью указателя увеличить значение переменной a в 2 раза если a>b иначе b уменьшить в 2 раза.

3. Ввести значение 2-х вещественных переменных a и b. Направить два указателя на эти переменные. С помощью указателя увеличить значение переменной a в 3 раза. Затем поменять местами значения переменных a и b через их указатели.

4. Ввести значение 2-х вещественных переменных a и b. Направить два указателя на эти переменные. Если a>b, то с помощью указателя увеличить значение переменной a на 3 и b уменьшить в 3 раза, в противном случае a уменьшить в 2 раза и b увеличить на 3.

5. Ввести значение 2-х символьных переменных a и b. Направить два указателя на эти переменные. С помощью указателя изменить значение переменной a. Затем поменять местами значения переменных a и b через их указатели.

6. Ввести значение 2-х целых переменных a и b. Направить два указателя на эти переменные. Большее из них с помощью указателя увеличить в 5 раз и меньшее уменьшить на 5.

7. Ввести значение 3-х целых переменных a и b и c. Направить указатели на эти переменные. С помощью указателя увеличить значение переменной a в 2 раза. Затем поменять местами значения переменных c и b через их указатели.

8. Ввести значение 3-х вещественных переменных a и b и c. Направить указатели на эти переменные. С помощью указателя увеличить значение переменной c в 3 раза. Затем поменять местами значения переменных a и c через их указатели.

9. Ввести значение 2-х вещественных переменных a и b. Направить два указателя на эти переменные. Большее из них с помощью указателя увеличить на 7 и меньшее уменьшить на 3.

10. Ввести значение 2-х символьных переменных a и b. Направить два указателя на эти переменные. Затем поменять местами значения переменных a и b через их указатели.

11. Ввести значение 2-х целых переменных a и b. Направить два указателя на эти переменные. Затем поменять местами значения переменных a и b через их указатели.

12. Ввести значение 2-х вещественных переменных a и b. Направить два указателя на эти переменные. Затем поменять местами значения переменных a и b через их указатели.

13. Ввести значение 2-х целых переменных a и b. Направить два указателя на эти переменные. С помощью указателя увеличить значение переменной a в 2 раза, a b уменьшить в 2 раза.

14. Ввести значение 2-х вещественных переменных a и b. Направить два указателя на эти переменные. С помощью указателя увеличить значение переменной a в 3 раза, a b уменьшить в 3 раза

15. Ввести значение 2-х вещественных переменных a и b. Направить два указателя на эти переменные. С помощью указателя увеличить значение переменной a в 3 раза, a b уменьшить в 3 раза.

ЛАБОРАТОРНАЯ РАБОТА № 5.

Функция

Цель работы: получить навыки использования пользовательских функций.

Порядок выполнения работы: изучить теоретическую часть, выполнить практические задания, оформить отчет, осуществить защиту лабораторной работы.

Теоретическая часть

Функцией называется выделенная последовательность инструкций, предназначенных для решения определенной задачи.

Есть несколько причин для использования пользовательских функций, во-первых, программа приобретает некоторую структуру и, тем самым, становится более понятной и упорядоченной, во-вторых, исключаются повторы похожих участков текста, то есть текст программы оптимизируется.

Функция может многократно вызываться из различных частей программы, в общем случае она выполняет следующие действия:

– получает параметры;

– выполняет инструкции, согласно заложенному алгоритму;
– может возвращать результат в вызывающую программу.

С использованием функций в языке С++ связаны понятия, которые условно можно разделить на две группы.

В первую группу входят определение, прототип и вызов функции – все три понятия связанны с подготовкой функции к работе.

Вторая группа, параметры и возвращаемое значение, обеспечивает связь функции с «внешней средой». Функция может многократно вызываться из различных частей программы, при этом необходимо обеспечить её связь с вызывающей программой, из вызывающей программы в функцию передать необходимые для работы данные, а по окончанию работы принять результат.

Определение функции – это описание действий, выполняемых функцией согласно требованиям алгоритма. Именно эта часть программы будет впоследствии многократно вызываться из других частей программы.

Прототип функции (объявление) используется в том случае, если вызов функции предшествует её определению или если определение и вызовы функции находятся в разных файлах.

Вызов функции обеспечивает связь с вызывающей программой. При вызове:

– передаются параметры из вызывающей программы в функцию
– управление передается первой инструкции в теле функции,
– после завершения работы функции в вызывающую программу передается возвращаемое значение, управление возвращается в точку вызова.

Определение функции состоит из заголовка и тела, например :

double f1 (int a, int f) //заголовок

{ … } // тело

В данном примере определена функция *f1* с двумя параметрами *int a* и *int f* , возвращающая значение типа double.

Тип функции (в нашем примере double) определяет тип значения, которое возвращает функция. Если тип не указан, то предполагается, что функция возвращает целое значение, типа *int*. Если функция не должна возвращать значение, то используется тип *void*, который в данном случае означает отсутствие значения.

В заголовке функции параметры называются формальными, и служат для её связи с вызывающей программой. Формальные параметры создаются в начале работы функции – это локальные переменные, которые инициализируются значениями, полученными из вызывающей программы при вызове функции. Параметры при вызове функции получают конкретные значения и называются фактическими параметрами, например, вызов функции может выглядеть так:

```
…
double z;
int s1=10;
…
z = f1(s1, 5); //вызов функции f1, s1 и 5 фактические параметры
```

Передача параметров в функцию и возврат значений

Параметры позволяют передать информацию из вызывающей программы в функцию. В теле функции параметрами можно пользоваться так же, как и локальными переменными. При вызове функции:

– для каждого формального параметра создаётся локальная переменная;

– начальными значениями созданных переменных являются фактические параметры, определяемые при вызове функции.

В языке С функция может возвращать только одно значение, для этого её выполнение следует завершить оператором return, содержащим некоторое выражение. Следует отметить, что тип функции в определении должен соответствовать типу выражения оператора return в её теле.

Пример 1. Функция нахождения суммы двух чисел

```
…
int sum(int a, int b)
{ return a+b;}
int _main()
{ int x =5,y=10,z;
z = sum(x,y);
…
}
```

При вызове функции создаются две локальные *a* и *b* переменные, которые инициализируются фактическими параметрами *x* и *y* :

```
a=x
```

b=y

Функция возвращает значение типа *int*, которое записывается в переменную *z*.

Использование прототипа функции

В языке Си определения функций могут следовать за определением функции main, перед ним, или находится в другом файле.

Положение определения функции :

– за функцией main;
– перед функцией main;
– в другом модуле (файле).

Однако во всех случаях к моменту вызова функция должна быть определена или объявлена. Это требование обусловлено тем, что компилятор должен осуществить проверку корректности вызова функции (проверку соответствия количества и типов фактических параметров, количеству и типам формальных параметров).

Когда вызов функции предшествует её определению, эта проверка выполняется по прототипу.

Прототип напоминает заголовок в определении функции:

– Тело функции отсутствует;
– Имена формальных параметров могут быть опущены (типы параметров опускать нельзя!).

Прототипы:

int power (int base, int n);

или

int power (int , int);

Если функция определена до функции main() – прототип не обязателен.

Практическая часть

Задание 1. Решите задачу.

1. Определить функцию для вычисления расстояния между двумя точками, заданными своими координатами (вычисление длины отрезка). Создать программу ввода информации о нескольких отрезках и вычислении их длин.

2. Определить функцию возможности построения треугольника по его сторонам. Определить функцию вычисления площади треугольника по его сторонам.

3. Создать программу многократного ввода сторон треугольника и вычисление его площади предусмотреть вывод сообщения о невозможности построения треугольника.

4. Определить функцию вычисления объема куба, заданного стороной. Создать программу многократного ввода стороны куба и вычисление его объема.

5. Определить функцию вычисления площади прямоугольника по его сторонам.

6. Создать программу многократного ввода сторон прямоугольника и вычисления его площади.

7. Определить функцию вычисления площади поверхности цилиндра, заданного радиусом и высотой.

8. Создать программу ввода радиуса и высоты и вычисления площади поверхности цилиндра.

9. Определить функцию вычисления периметра квадрата, заданного диагональю.

10. Создать программу многократного ввода диагонали квадрата и вычисления его периметра.

11. Определить функцию вычисления объема конуса, заданного радиусом высотой.

12. Создать программу многократного ввода радиуса и высоты и вычисления объема конуса.

13. Определить функцию вычисления площади прямоугольника по его сторонам.

14. Создать программу многократного ввода информации о прямоугольниках и определить их площадь.

15. Определить функцию вычисления объема куба, заданного стороной. Создать программу многократного ввода стороны куба и вычисление его объема.

16. Определить функцию вычисления периметра прямоугольника по его сторонам.

17. Создать программу многократного ввода сторон прямоугольника и вычисления его периметра.

18. Определить функцию вычисления площади круга, заданного радиусом.

19. Создать программу многократного ввода двух радиусов и вычисления площади «бублика» между двумя кругами.

20. Определить функцию возможности построения треугольника по его сторонам. Определить процедуру определения типа треугольника (равносторонний, равнобедренный, разносторонний)

21. Создать программу многократного ввода сторон треугольника и определения его типа, предусмотреть вывод сообщения о невозможности построения треугольника.

22. Определить функцию вычисления площади круга, заданного радиусом.

23. Создать программу многократного ввода радиуса и вычисления площади круга.

24. Определить функцию вычисления площади квадрата, заданного диагональю. Создать программу многократного ввода диагонали квадрата и вычисления его площади.

25. Определить функцию возможности построения треугольника по его сторонам. Определить процедуру вычисления периметра треугольника.

26. Создать программу многократного ввода сторон треугольника и определения его периметра, предусмотреть вывод сообщения о невозможности построения треугольника.

Задание 2. Решите задачу.

1. Создать и распечатать четыре исходных массива целых двухзначных чисел X(N) , Y(M), Z(K), Q(L). Сформировать и распечатать результирующий массив R(4), содержащий максимальные элементы исходных массивов. В массиве с максимальным значением максимума заменить все нули максимумом, результат вывести на экран. Использовать функции: инициализации, вывода, поиска максимального элемента в произвольном массиве целых чисел.

2. Создать и распечатать исходную матрицу вещественных двухзначных чисел N×M (S знаков после запятой). Вычислить и отобразить суммы строк. Определить номер строки с максимальной суммой, вывести сообщение. Использовать функции: инициализации, вывода, вычисления суммы заданной строки матрицы.

3. Создать и распечатать два исходных массива вещественных чисел X(N) , Y(M).

4. (S знаков после запятой). Определить массив, в котором количество отрицательных элементов меньше. Использовать функции:

инициализации, вывода, определения количества отрицательных элементов в произвольных массивах целых чисел.

5. Создать и распечатать две исходных матриц вещественных трехзначных чисел N×M (S знаков после запятой). Определить матрицу с большей суммой выше главной диагонали. Использовать функции: инициализации, вывода, определения суммы выше главной диагонали в матрице вещественных чисел.

6. Создать и распечатать исходную матрицу вещественных трехзначных чисел N×M (S знаков после запятой). Вычислить и отобразить суммы столбцов. Определить номер столбца с минимальной суммой, вывести сообщение. Использовать функции: инициализации, вывода, вычисления суммы заданного столбца произвольной матрицы.

7. Создать два исходных массива целых чисел X(N) , Y(M). Определить массив с меньшим значением минимума, вывести сообщение. В массиве с большим значением минимума подсчитать количество нулей. Использовать функции: инициализации, вывода, поиска минимального элемента в произвольном массиве вещественных чисел.

8. Создать и распечатать исходную матрицу вещественных двухзначных чисел N×M (S знаков после запятой). Вычислить и отобразить максимумы столбцов. Определить номер столбца с минимальным максимумом, вывести сообщение. Использовать функции: инициализации, вывода, вычисления максимума заданного столбца произвольной матрицы.

9. Создать и распечатать две исходных матрицы целых двухзначных чисел N×N. Определить матрицу с большей суммой побочной диагонали, вывести сообщение. Использовать функции: инициализации, вывода, определения суммы побочной диагонали произвольной матрицы.

10. Создать и распечатать исходную матрицу целых двузначных чисел N×M. Определить столбец с максимальным числом элементов, кратных K, (K вводится с клавиатуры). Использовать функции: инициализации, вывода, вычисления числа элементов, кратных K в заданном столбце матрицы.

11. Создать и распечатать два исходных массива вещественных чисел X(N) , Y(M) (S знаков после запятой). Определить массив с большим числом отрицательных элементов, вывести сообщение. В массиве с меньшим числом отрицательных элементов определить минимальный отрицательный элемент. Использовать функции: инициа-

лизации, вывода, поиска минимального отрицательного элемента, поиска количества отрицательных элементов в произвольном массиве вещественных чисел.

12. Создать и распечатать две исходных матрицы целых двухзначных чисел N×M. Вычислить и отобразить суммы столбцов. Определить номер столбца с минимальной суммой, вывести сообщение

13. Определить матрицу с максимальной суммой столбца, вывести сообщение (указать матрицу и номер столбца). Использовать функции: инициализации, вывода, определения суммы заданного столбца произвольной матрицы.

14. Создать и распечатать три исходных массива целых чисел X(N) , Y(M), Z(K). Сформировать и распечатать результирующий массив R(3), содержащий максимальные положительные элементы исходных массивов. Вывести на печать массив с минимальным количеством положительных элементов.

15. Использовать функции: инициализации, вывода, поиска максимального положительного элемента и функцию подсчета количества положительных элементов в произвольных массивах целых чисел.

16. Создать и распечатать две исходных матрицы целых трехзначных чисел N×N. Определить и распечатать матрицу с меньшей суммой главной диагонали. Использовать функции: инициализации, вывода, определения суммы главной диагонали произвольной матрицы целых чисел.

17. Создать и распечатать два исходных массива целых двузначных чисел X(N) , Y(M) Определить массив, в котором количество элементов кратных K больше (K ввести с клавиатуры). Использовать функции: инициализации, вывода, определения количества элементов, кратных K в массиве целых чисел.

18. Создать и распечатать исходную матрицу целых двухзначных чисел N×M. Вычислить и отобразить число нулей в каждой строке. Определить строку с максимальным числом нулей. Использовать функции: инициализации, вывода, определения числа нулей в заданной строке произвольной матрицы целых чисел.

Контрольные вопросы:

1. Поясните общие понятия, связанные с использованием функции: определение, вызов, параметры функции.
2. Что такое прототип функции, когда он используется.

3. Что такое тип функции?

4. Какую роль выполняют параметры в функции? Расскажите о формальных и фактических параметрах функции.

5. Расскажите об использовании переменных в функциях, какая разница между локальной и глобальной переменной?

6. Как передать массив в функцию?

ЛАБОРАТОРНАЯ РАБОТА № 6.

Переменные и классы

Цель работы: получить навыки работы с классами.

Порядок выполнения работы: изучить теоретическую часть, выполнить практические задания, оформить отчет, осуществить защиту лабораторной работы.

Теоретическая часть

В стандарте C++ под классом (class) подразумевается пользовательский тип, объявленный с использованием одного из ключевых слов class, struct или union, под структурой (structure) подразумевается класс, определённый через ключевое слово struct, и под объединением (union) подразумевается класс, определённый через ключевое слово union.

Особенности классов:

1. Класс является собственным типом данных;

2. Класс – это некоторая идея еще не существующего объекта, в которой воедино собраны все детали, все свойства и все нужные действия, необходимые для этого объекта. Например:

```
void main()
{
  int   //Вы еще не прописали переменной, но она задумана и тип
уже прописан.
  return;
}
```

3. Создается класс с помощью слова class, за которым следует открывающаяся и закрывающаяся фигурные скобки. После закрывающейся фигурной скобки ставится точка с запятой. Внутри фигурных скобок пишется вся информация для класса. После создания пустого класса внутри этого класса прописывается вся информация для выполнения поставленной задачи. Важные моменты то, что внутри класса используются слова public, private и protected;

4. По сути, класс есть структура. Отличается класс от структуры только модификатором доступа по умолчанию (private);

5. Функции внутри класса называют методами класса или полями класса;

6. Объект есть воплощение вашей идеи, описанной в классе во что-то реально существующее.

Пример 1.

Объявляем класс

```
#include <conio.h>
#include <iostream>
using namespace std;
class sum
{
};
void main ()
{
}
```

Описываем класс

```
#include <conio.h>
#include <iostream>
using namespace std;
class sum
{
int x, y;
public:
void get_xy()
{
cout<<”Input x”;
cin>>x;
cout<<”Input y”;
cin>>y;

}
int sum_xy() return x+y;
};
void main ()
{
}
```

Описываем работу с классом

```
#include <conio.h>
#include <iostream>
using namespace std;
class sum
{
int x, y;
public:
void get_xy()
{
cout<<”Input x”;
cin>>x;
cout<<”Input y”;
cin>>y;
}
int sum_xy() return x+y;
} s1;
void main ()
{
s1.get_xy();
cout<< s1.get_xy();
}
```

Практическая часть

Задание 1. Реализовать через класс расчет уравнения, X и Y вводятся программно.

1. X^2+Y^2, X=5, Y=6
2. $3X-Y^3$, X=10, Y=2
3. X^2+Y-2, X=7, Y=6
4. $2X^2+Y-2$, X=7, Y=6
5. $X^2+4Y-12$, X=7, Y=6
6. X+Y+, X=7, Y=6
7. $3X-Y^3$, X=11, Y=3
8. X^2+Y-2, X=17, Y=4
9. $2X^2+Y-2$, X=9, Y=7
10. $X^2+4Y-12$, X=10, Y=4
11. X^2+Y^2, X=2, Y=3
12. $3X-Y^3$, X=3, Y=2
13. X^2+Y-2, X=6, Y=8

14. $2X^2+Y-2$, X=8, Y=8
15. $X^2+4Y-12$, X=6, Y=9

Задание 2. Реализовать программу работы с классом, программа должна обеспечивать ввод вывод данных из объектов класса на примере одного объекта.

1. Класс Vizitka. Содержит: Фамилию, Имя, Номер телефона.
2. Класс Vid_zanatiy. Содержит: Номер, Название вида, Описание вида.
3. Класс Времена года. Содержит: Название времени года, Описание времени года.
4. Класс Zametki. Содержит: Номер заметки, дата заметки, текст заметки.
5. Класс Gruppa. Содержит: Номер группы, номер в группе, ФИО.
6. Класс Raspisanie. Содержит: Номер пары, Время начала, время окончания.
7. Класс Moi_mesta. Содержит: Номер, Название места, Мои комментарии.
8. Класс Zveri_lesa. Содержит: Номер, Название зверя, описание зверя.
9. Класс Riby. Содержит: Номер, Название рыбы, описание рыбы.
10. Класс Pamyatniki_Minska. Содержит: Название памятника, адрес, описание.
11. Класс Reki. Содержит: Название реки, длина, описание реки.
12. Класс Ozera. Содержит: Название озера, площадь, описание озера.
13. Класс Goroda. Содержит: Название города, количество жителей, год основания.
14. Класс Frukti. Содержит: Название фрукта, Описание, страна импортер.
15. Класс Ovoschi. Содержит: название овоща, описание, вес плода (средний).

Контрольные вопросы:

1. Понятие «Класс».
2. Особенности классов.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Дьюхарст Программирование на C++ / Дьюхарст, Старк Стефан , Кэти. – М.: ДиаСофт, 2015. – 272 с.
2. Мейерс, С. Эффективный и современный C++. 42 рекомендации по использованию C++11 и C++14 / С. Мейерс. – М.: Вильямс, 2015. – 304 с.
3. Секунов, Н.Ю. Самоучитель Visual C++ 6.0 / Н.Ю. Секунов. – М.: СПб: BHV, 2014. – 960 с.
4. Ашарина, И.В. Основы программирования на языках С и С++: Курс лекций для высших учебных заведений / И.В. Ашарина. – М.: Гор. линия-Телеком, 2018. – 208 с.
5. Дорогов, В.Г. Основы программирования на языке С: Учебное пособие / В.Г. Дорогов, Е.Г. Дорогова; Под общ. ред. проф. Л.Г. Гагарина. – М.: ИД ФОРУМ, НИЦ ИНФРА-М, 2017. – 224 с.
6. Страуструп, Б. Язык программирования С++: Специальное издание / Б. Страуструп; Пер. с англ. Н.Н. Мартынов. – М.: БИНОМ, 2017. – 1136 с.
7. Фридман, А.Л. Основы объектно-ориентированного программирования на языке Си++ / А.Л. Фридман. – М.: Гор. линия-Телеком, 2016. – 234 с.
8. Павловская, Т.А. С/ С++. Программирование на языке высокого уровня / Т.А. Павловская. – СПб.: Питер, 2011. – 461 с.
9. Шилдт, Г. С++: базовый курс / Г. Шилдт – М.: Издательский дом «Вильямс», 2008. – 624 с.
10. Васильев, А.Н. Самоучитель С++ с примерами и задачами/ А.Н. Васильев. – СПб.: Наука и Техника, 2010. – 480с.