Основы программирования

ОГЛАВЛЕНИЕ

Оформление решений

Важно, чтобы решённая задача была правильно оформлена. Это предполагает:

- использование несущих смысловую нагрузку имен переменных, констант и функций;
- применение отступов при записи инструкций программы;
- использование комментариев.

Правильно оформленную программу легче отлаживать, кроме того, она производит хорошее впечатление.

Объявление переменных

Общие замечания

Приступая к решению задач этого раздела, следует вспомнить, что:

- каждая переменная программы должна быть объявлена;
- объявления переменных обычно помещают в начале функции, сразу за заголовком. Следует обратить внимание, что хотя язык C++ допускает объявление переменных практически в любом месте функции, объявлять переменные лучше все-таки в начале функции, снабжая инструкцию объявления кратким комментарием о назначении переменной;
- инструкция объявления переменной выглядит так:

  Тип ИмяПеременной;
- инструкцию объявления переменной можно использовать для инициализации переменной. В этом случае объявление переменной записывают следующим образом:

  Тип ИмяПеременной = НачальноеЗначение;
- В имени переменной можно использовать буквы латинского алфавита и цифры (первым символом должна быть буква);
- компилятор C++ различает прописные и строчные буквы, поэтому, например, имена Summa и summa обозначают разные переменные;
- основными числовыми типами языка C++ являются:

  int (целый) и float (дробный);
- после инструкции объявления переменной рекомендуется указывать назначение переменной.

Инструкция присваивания

Общие замечания

Приступая к решению задач этого раздела, следует вспомнить, что:

- инструкция присваивания предназначена для изменения значений переменных, в том числе и для вычислений "по формуле";
- в отличие большинства языков программирования, в C++ инструкция присваивания, выполняющая некоторое действие, может быть записана несколькими способами, например, вместо x=x+dx можно

записать x+=dx, а вместо i=i+1 воспользоваться оператором инкремента и записать i++;

- значение выражения в левой части инструкции присваивания зависит от типа операндов и операции, выполняемой над операндами. Целочисленное сложение и вычитание выполняется без учета перепонения. Например, если пременная n,
- объявленная как int , имеет значение 32767, то в результате выполнения инструкции n=n+1 значение переменной n будет равно -32768;
- результатом выполнения операции деления над целыми операндами является целое, которое получается отбрасыванием дробной части результата деления.

Вывод

Общие замечания

Приступая к решению задач этого раздела, следует вспомнить, что:

- функция printf обеспечивает вывод на экран монитора сообщений и значений переменных;
- первым параметром функции printf является строка вывода, определяющая выводимый текст и формат вывода значений переменных, имена которых указаны в качестве остальных параметров функции;
- формат вывода значений переменных задается при помощи спецификатора преобразования – последовательности символов, начинающейся с символа %;
- при выводе числовых значений наиболее часто используются следующие спецификаторы: %i – для вывода целых со знаком, %u – дня вывода беззнаковых целых, %f – для вывода дробных, в виде числа с плавающей точкой, %n.mf – для вывода дробных в формате с фиксированной точкой, где *n* – количество цифр целой части, *m* – дробной;
- некоторые символы могут быть помещены в строку вывода только как последовательность других, обычных символов: \n – новая строка, \t – табуляция, \" – двойная кавычка, \\ – символ \;
- наряду с функцией printf, для вывода на экран сообщений можно использовать функцию puts, которая после вывода текста автоматически переводит курсор в начало следующей строки;
- чтобы сразу после окончания работы программы окно, в котором программа работала, не было автоматически перекрыто другим окном, в конец программы нужно вставить следующие две инструкции:

```
printf("Для завершения нажмите клавишу <Enter>");
getch(); //  #include <conio.h>
```

Ввод

Общие замечания

Приступая к решению задач, следует вспомнить, что:

- для ввода исходных данных с клавиатуры предназначена функция scanf;
- первым параметром функции scanf является управляющая строка, остальные параметры – адреса переменных, значения которых должны быть введены;
- управляющая строка представляет собой заключенный в двойные кавычки список спецификаторов: %d – для ввода целых чисел, %f – для ввода дробных чисел, %c – для ввода символа, %s – для ввода строки;
- использование имени переменной, а не ее адреса в качестве параметра функции scanf является типичной ошибкой начинающих программистов. Кстати, компилятор эту ошибку не обнаруживает.

Примеры программ

Пример 1

```
// Программа для преобразования
// градусов Цельсия в градусы Фаренгейта:
// Fahrenheit = NCelsius * (212 - 32)/100 + 32

#include <iostream>
#include <conio.h>
int main()
{
   using namespace std;
   setlocale(LC_ALL, "Russian");
   // Введите температуру в градусах Цельсия
   int nNCelsius;
   cout << "Введите температуру по Цельсию: ";
   cin >> nNCelsius;
   // для приведенной формулы преобразования
   // вычислим преобразующий множитель
   int nNFactor;
   nNFactor = 212 - 32;
   // используем вычисленный множитель для
   // преобразования градусов Цельсия в
   // градусы Фаренгейта
   int nFahrenheit;
   nFahrenheit = nNFactor * nNCelsius/100 + 32;
   // вывод результатов
   cout << "Температура по Фаренгейту: ";
   cout << nFahrenheit;
   getch();
   return 0;
}
```

Пример 2

```
#include <stdio.h>
#include <conio.h>
#include <cmath>

int main(void)
{
    float x,y;
    x = 3.0;
    printf("x = ");
    scanf("%f", &x);
    y = pow(x,2) + 2*x + 5;
    printf("x =%5.2f, y =%6.2f\n",x,y);
    getch();
    return 0;
}
```

Лабораторная работа №1

Задачи

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

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

3. Запишите в виде инструкции присваивания формулу вычисления площади прямоугольника.

4. Запишите в виде инструкции присваивания формулу вычисления площади треугольника: $s = ah/2$, где a – длина основания; h – высота треугольника.

5. Запишите в виде инструкции присваивания формулу вычисления площади трапеции: $s =(a+b)h/2$, где a *и* b – длины оснований, h – высота трапеции.

6. Запишите в виде инструкции присваивания формулу вычисления площади круга: $S = \pi R^2$.

7. Запишите в виде инструкции присваивания формулы вычисления площади поверхности и объема цилиндра.

$$S = 2\pi R(h+R)$$

$$V = \pi R^2 h$$

8. Запишите в виде инструкции присваивания формулу вычисления объема параллелепипеда.

9. Объявите необходимые переменные и запишите в виде инструкции присваивания формулы вычисления объема и площади поверхности шара.

$$V =(3/4)\ \pi R^3$$

$$S = 4\pi R^2$$

10. Запишите в виде инструкции присваивания формулу вычисления объема цилиндра.

11. Запишите в виде инструкции присваивания формулу вычисления тока, по известным значениям напряжения и сопротивления электрической цепи.

12. Запишите в виде инструкции присваивания формулу вычисления сопротивления электрической цепи по известным значениям напряжения и силы тока.

13. Запишите в виде инструкции присваивания формулу вычисления сопротивления электрической цепи, состоящей из трех последовательно соединенных резисторов.

14. Запишите в виде инструкции присваивания формулу вычисления сопротивления электрической цепи, состоящей из двух параллельно соединенных резисторов: $R = R_1 \cdot R_2 / (R_1 + R_2)$

15. Объявите необходимые переменные и запишите в виде инструкции присваивания формулу вычисления стоимости покупки, состоящей из нескольких тетрадей, обложек к ним и карандашей.

16. Объявите необходимые переменные и запишите в виде инструкции присваивания формулу вычисления стоимости покупки, состоящей из помидоров, огурцов и нескольких пучков укропа.

17. Написать программу, которая выводит на экран ваши имя и фамилию.

18. Написать инструкции вывода значений дробных переменных $\mathbf{x_1}$ и $\mathbf{x_2}$. На экране перед значением переменной должен быть выведен поясняющий текст, представляющий собой имя переменной, за которым следует знак "равно".

19. Написать программу вычисления площади параллелограмма. Ниже приведен рекомендуемый вид экрана во время выполнения программы (данные, введенные пользователем, выделены полужирным шрифтом).

```
Вычисление площади прямоугольника
Введите исходные данные:
Длина (см) -> 9
Ширина (см) -> 7.5
Площадь параллелограмма: 67.50 кв.см
```

20. Написать программу вычисления объема параллелепипеда. Ниже приведен рекомендуемый вид экрана во время выполнения программы (данные, введенные пользователем, выделены полужирным шрифтом).

```
Вычисление объема параллелепипеда.
Введите исходные данные:
Длина (см) -> 9
Ширина (см) -> 7.5
Высота (см) -> 5
Объем: 337.50 куб.см
```

21. Написать программу вычисления площади поверхности параллелепипеда. Ниже приведен рекомендуемый вид экрана во время выполнения программы (данные, введенные пользователем, выделены полужирным шрифтом).

```
Вычисление площади поверхности параллелепипеда
Введите исходные данные:
Длина (см) -> 9
Ширина (см) -> 7.5
Высота (см) -> 5
Площадь поверхности: 90.00 кв.см.
```

22. Написать программу вычисления объема куба. Ниже приведен рекомендуемый вид экрана во время выполнения программы (данные, введенные пользователем, выделены полужирным шрифтом).

```
Вычисление объема куба.
Введите длину ребра (см) и нажмите клавиигу <Enter>
-> 9.5
Объем куба: 857.38 куб.см.
```

23. Написать программу вычисления объема цилиндра. Ниже приведен рекомендуемый вид экрана во время выполнения программы (данные, введенные пользователем, выделены полужирным шрифтом).

```
Вычисление объема цилиндра.
Введите исходные данные:
радиус основания (см) -> 5
высота цилиндра (см) -> 10
Объем цилиндра 1570.80 см. куб.
Для завершения нажмите <Enter>
```

Выбор

Инструкция *if*

Общие замечания

Приступая к решению задач этого раздела, следует вспомнить, что:

- инструкция **if** используется для выбора одного из двух направлений дальнейшего хода программы;
- выбор последовательности инструкций осуществляется в зависимости от значения *условия* – заключенного в скобки выражения, записанного после **if**;
- инструкция, записанная после **else**, выполняется в том случае, если значение выражения *условие* равно нулю, во всех остальных случаях выполняется инструкция, следующая за условием;
- если при соблюдении или несоблюдении условия надо выполнить несколько инструкций программы, то эти инструкции следует объединить в группу – заключить в фигурные скобки;
- при помощи вложенных одна в другую нескольких инструкций **if** можно реализовать множественный выбор.

Пример

Написать программу, которая проверяет, является ли год високосным.

```
#include <iostream>
#include <conio.h>
int main()
{
   using namespace std;
   int Year;
   setlocale(LC_ALL, "Russian");
   cout << "Введите год: ";
   cin >> Year;
   if ((Year % 400 == 0) || (Year % 100 > 0) && (Year % 4 == 0))
      cout << "Год високосный\n";
   else
      cout << "Год невисокосный\n";
}
```

Лабораторная работа №2

Задачи

1. Написать программу, которая вычисляет частное двух чисел. Программа должна проверять правильность введенных пользователем данных и, если они неверные (делитель равен нулю), выдавать сообщение об ошибке. Ниже приведен рекомендуемый вид экрана во время выполнения программы (данные, введенные пользователем, выделены полужирным шрифтом).

```
Вычисление частного.
Введите в одной строке делимое и делитель,
затем нажмите <Enter>.
-> 12  0
Вы ошиблись. Делитель не должен быть равен нулю.
```

2. Написать программу вычисления площади кольца. Программа должна проверять правильность исходных данных. Ниже приведен рекомендуемый вид экрана во время выполнения программы (данные, введенные пользователем, выделены полужирным шрифтом).

```
Вычисление площади кольца.
Введите исходные данные:
Радиус кольца (см) -> 3.5
Радиус отверстия (см) -> 7
Ошибка! Радиус отверстия не может быть больше радиуса кольца.
```

3. Написать программу, которая переводит время из минут и секунд в секунды. Программа должна проверять правильность введенных пользователем данных и в случае, если данные неверные, выводить соответствующее сообщение. Рекомендуемый вид экрана во время выполнения программы приведен ниже.

```
Введите время (минут.секунд) -> 2.90
Ошибка! Количество секунд не может быть больше 60
Для завершения нажмите <Enter>
```

4. Написать программу вычисления сопротивления электрической цепи, состоящей из двух сопротивлений. Сопротивления могут быть соединены последовательно или параллельно. Ниже приведен рекомендуемый вид экрана во время выполнения программы (данные, введенные пользователем, выделены полужирным шрифтом).

```
Вычисление сопротивления электрической цепи.
Введите исходные данные:
Величина первого сопротивления (Ом) -> 15
Величина второго сопротивления (Ом)-> 27.3
Тип соединения (1 – последовательное, 2 – параллельное) -> 2
Сопротивление цепи: 9.68 Ом
```

5. Написать программу решения квадратного уравнения. Программа должна проверять правильность исходных данных и в случае, если коэффициент при второй степени неизвестного равен нулю, выводить соответствующее сообщение. Ниже приведен рекомендуемый вид экрана во время выполнения программы (данные, введенные пользователем, выделены полужирным шрифтом).

```
Решение квадратного уравнения
Введите в одной строке значения коэффициентов и нажмите
<Enter>
-> 1 2 -3
Корни уравнения:
x1 = 1.00000, x2 = -3.00000
```

6. Написать программу вычисления стоимости покупки с учетом скидки. Скидка в 10% предоставляется, если сумма покупки больше 1000 руб. Ниже приведен рекомендуемый вид экрана во время выполнения программы (данные, введенные пользователем, выделены полужирным шрифтом).

```
Вычисление стоимости покупки с учетом скидки
Введите сумму покупки и нажмите <Enter>
-> 1200
Вам предоставляется скидка 10%
Сумма покупки с учетом скидки: 1080.00 руб.
```

7. Написать программу вычисления стоимости покупки с учетом скидки. Скидка в 3% предоставляется, если сумма покупки больше 500 руб, в 5% – если сумма больше 1000 руб. Ниже приведен рекомендуемый вид экрана во время выполнения программы (данные, введенные пользователем, выделены полужирным шрифтом).

```
Вычисление стоимости покупки с учетом скидки.
Введите сумму покупки и нажмите <Enter>
-> 640
Вам предоставляется скидка 3%
Сумма с учетом скидки: 620.80 руб.
```

8. Написать программу, которая сравнивает два введенных с клавиатуры числа. Программа должна указать, какое число больше, или, если числа равны, вывести соответствующее сообщение. Ниже приведен рекомендуемый вид экрана во время работы программы.

```
Введите в одной строке два целых числа и нажмите <Enter>.
-> 34 67
34 меньше 67
```

9. Написать программу, которая вычисляет оптимальный вес для пользователя, сравнивает его с реальным и выдает рекомендацию о необходимости поправиться или похудеть. Оптимальный вес вычисляется по формуле: Рост (см) – 100. Рекомендуемый вид экрана во время выполнения программы приведен ниже (данные, введенные пользователем, выделены полужирным шрифтом).

```
Введите в одной строке через пробел
рост (см) и вес (кг), затем нажмите <Enter>
->170 68
Вам надо поправиться на 2.00 кг.
```

Инструкция *switch*

Общие замечания

Приступая к решению задач этого раздела, следует вспомнить, что:

- инструкция **switch** предназначена для выбора одного из нескольких возможных направлений дальнейшего хода программы;
- выбор последовательности инструкций осуществляется в зависимости от равенства значения переменной-селектора константе, указанной после слова **case**;
- если значение переменной-селектора не равно ни одной из констант, записанных после **case**, то выполняются инструкции, расположенные после слова **default**;
- в качестве переменной-селектора можно использовать переменную целого (int) или символьного (char) типа.

Циклы

for

Общие замечания

Приступая к решению задач этого раздела, следует вспомнить, что:

- инструкция **for** используется для организации циклов с фиксированным, известным во время разработки программы, числом повторений;
- количество повторений цикла определяется начальным значением переменной-счетчика и условием завершения цикла;
- переменная-счетчик должна быть целого (int) типа и может быть объявлена непосредственно в инструкции цикла.

do while

Общие замечания

Приступая к решению задач этого раздела, следует вспомнить, что:

- число повторений инструкций цикла **do while** определяется ходом выполнения программы;
- инструкции цикла **do while** выполняются до тех пор, пока значение выражения, записанного после слова **while,** не станет равным нулю;
- после слова **while** надо записывать условие выполнения инструкций цикла;
- для завершения цикла **do while** в теле цикла обязательно должны быть инструкции, выполнение которых влияет на условие завершения цикла;
- цикл **do while** – это цикл с постусловием, т. е. инструкции тела цикла будут выполнены хотя бы один раз;
- цикл **do while,** как правило, используется для организации приближенных вычислений, в задачах поиска.

while

Общие замечания

Приступая к решению задач этого раздела, следует вспомнить, что:

- число повторений инструкций цикла **do whil**e определяется ходом выполнения программы;
- инструкции цикла **while** выполняются до тех пор, пока значение выражения, записанного после слова **while,** не станет равным нулю;
- после слова **while** надо записывать условие выполнения инструкций цикла;
- для завершения цикла **while** в теле цикла обязательно должны быть инструкции, выполнение которых влияет на условие завершения цикла;
- цикл **while** – это цикл с предусловием, т. е. возможна ситуация, при которой инструкции тела цикла ни разу не будут выполнены;
- цикл **while,** как правило, используется для организации приближенных вычислений, в задачах поиска и обработки данных, вводимых с клавиатуры или из файла.

Примеры программ

Пример 1

```
// Инструкция выбора switch
#include <iostream>
#include <conio.h>
int main()
{
   using namespace std;
   setlocale(LC_ALL, "Russian");
   int choice;
   cout << "Введите 1, 2 или 3: ";
```

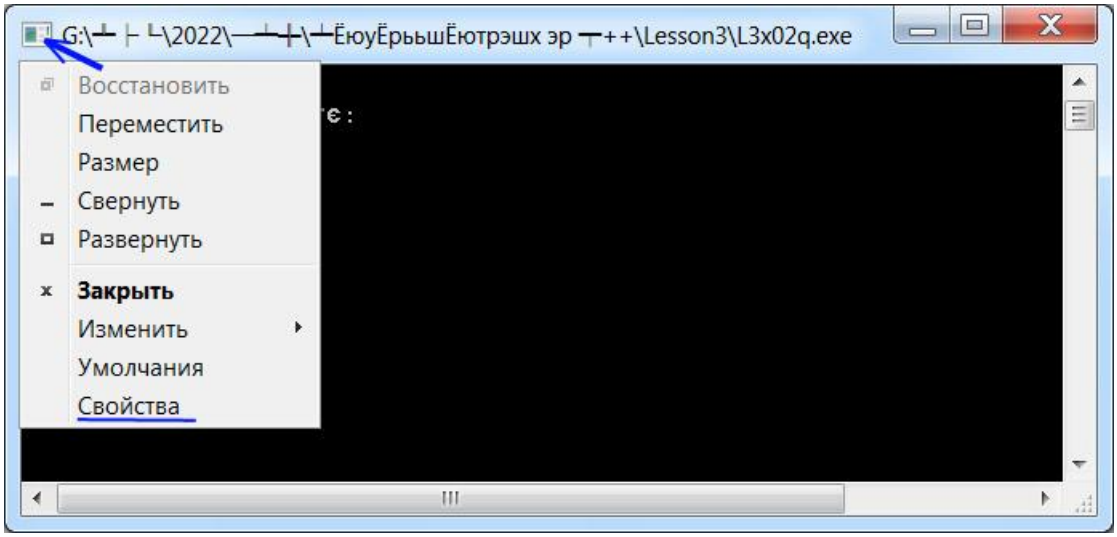
```
   cin >> choice;
   switch (choice)
   {
      case 1:
        //обработка случая "1"
         cout << "Ваш выбор 1";
         break;
      case 2:
         //обработка случая "2"
         cout << "Ваш выбор 2";
         break;
      case 3:
         // обработка случая "3"
         cout << "Ваш выбор 3";
         break;
     default:
         cout << "Некорректный ввод\n";
   }
   getch();
   return 0;
}
```

Пример 2

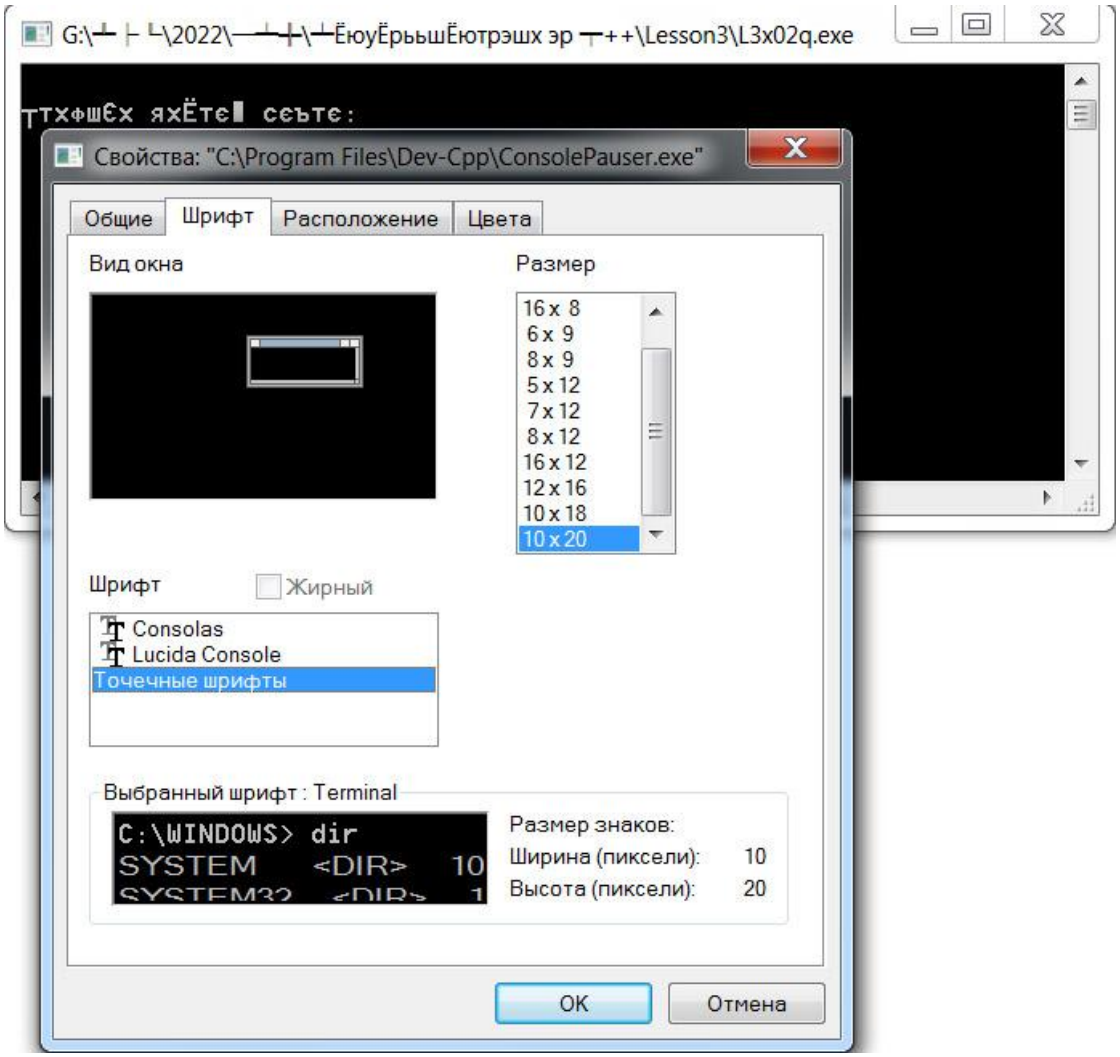
```
// Выбор названия животного по первой букве
#include <stdio.h>
#include <conio.h>
#include <iostream>
#include <Windows.h>
using namespace std;

main()
{
     setlocale(LC_ALL, "Russian");  // in <iostream>
     SetConsoleCP(1251);      // in  <Windows.h>
     SetConsoleOutputCP(1251);
     char c;
     printf("\nВведите первую букву:\n");
     scanf("%c", &c); // ввести букву
     printf("%c\n", c);
     switch (c) // заголовок оператора выбора
     {
          case 'а':
          case 'А': printf("\nАнтилопа"); break;
          case 'б':
          case 'Б': printf("\nБарсук"); break;
          case 'в':
          case 'В': printf("\nВолк"); break;
          default: printf("\nНе знаю я таких!"); // по умолчанию
     }
     getch();  // <conio.h>
}
```

Здесь строчки `SetConsoleCP(1251)` и `SetConsoleOutputCP(1251)` потребовались для корректного ввода с клавиатуры русских букв. Кроме того, возможно, потребуется установить шрифт для консольного окна. Для этого, когда при работе программы появится это окно, надо в настройках вызвать пункт «Свойства» этого окна. Сначала щёлкнуть мышкой по иконке окна в левом верхнем углу (см. рисунок ниже):



Затем в пункте «Шрифт» выбрать шрифт «Consolas» или «Lucida Console» (см. рисунок ниже).



Пример 3

```
// ForDemo. Вводится счетчик цикла.
// На экран выводится количество выполненных циклов for
#include <iostream>
#include <conio.h>
int main()
{
    using namespace std;
    setlocale(LC_ALL, "Russian");
    // ввод счетчика цикла
    int loopCount;
    cout << "Введите loopCount: " ;
    cin >> loopCount;
    // работаем, пока не нарушится условие
    for (int i = loopCount; i > 0; i--)
    {
       cout << "Осталось выполнить " << i-1 << " циклов\n";
    }
    getch();
```

```
    return 0;
}
```

Пример 4

```
// ForDemo. Вводится счетчик цикла.
// На экран выводится количество выполненных
// циклов for
#include <iostream>
#include <conio.h>
int main()
{
    using namespace std;
    setlocale(LC_ALL, "Russian");
    // ввод счетчика цикла
    int loopCount;
    cout << "Введите loopCount: " ;
    cin >> loopCount;
    // работаем, пока не нарушится условие
    for (int i = loopCount; i > 0; i--)
    {
       cout << "Осталось выполнить " << i-1 << " циклов\n";
    }
    getch();
    return 0;
}
```

Пример 5

```
// BreakDemo - зводим множество чисел.
// Суммируем эти числа, пока пользователь не введет
// отрицательное число
#include <iostream>
#include <conio.h>
int main()
{
   using namespace std;
   setlocale(LC_ALL, "Russian");
   // введите счетчик цикла
   int accumulator = 0;
   cout << "Эта программа суммирует числа, "
   << "введенные пользователем\n";
   cout << "Выполнение цикла "
   << "заканчивается после "
   << "введения отрицательного числа\n";
   // бесконечный цикл
   for(;;)
   {
      // ввод следующего числа
      int value = 0;
      cout << "Введите следующее число: ";
      cin >> value;
      // если оно отрицательно...
      if (value < 0)
```

```
        {
            // ...тогда ВЫХОДИМ из цикла
            break;
        }
        // ...иначе добавляем число к общей сумме
        accumulator = accumulator + value;
        // после выхода из цикла
        // выводим результат суммирования
        cout << "\nОбщая сумма равна "
        << accumulator << endl;
    }
    getch();
    return 0;
}
```

Лабораторная работа №3

Задачи

1. Напишите программу, которая запрашивает у пользователя номер дня недели, затем выводит название дня недели или сообщение об ошибке, если введены неверные данные.

2. Написать программу, которая вычисляет стоимость междугородного телефонного разговора (цена одной минуты определяется расстоянием до города, в котором находится абонент). Исходными данными для программы являются код города и длительность разговора. Ниже приведены коды некоторых городов и рекомендуемый вид экрана во время работы программы (данные, введенные пользователем, выделены полужирным шрифтом).

Город	Код	Цена минуты (руб.)
Владивосток	423	2,2
Москва	495	1,0
Мурманск	815	1,2
Самара	846	1,4

```
Вычисление стоимости разговора по телефону.
Введите исходные данные:
Код города -> 423
Длительность (целое количество минут)-> 3
Город: Владивосток
Цена минуты: 2.20 руб.
Стоимость разговора: 6.60 руб.
```

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

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

```
Таблица квадратов
```

```
- - - - - - - - - -
Число      Квадрат
1            1
2            4
3            9
4           16
5           25
6           36
7           49
8           64
9           81
10         100
- - - - - - - - - -
```

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

```
Таблица квадратов нечётный чисел
- - - - - - - - - -
Число      Квадрат
1            1
3            9
5           25
7           49
9           81
- - - - - - - - - -
```

6. Написать программу, которая вычисляет сумму первых **n** целых положительных целых чисел. Количество суммируемых чисел должно вводиться во время работы программы. Ниже приведен рекомендуемый вид экрана (данные, введенные пользователем, выделены полужирным шрифтом).

```
Вычисление суммы положительных чисел.
Введите количество суммируемых чисел -> 20
Сумма первых 20 положительных чисел равна 210
```

7. Написать программу, которая вычисляет сумму первых **n** целых положительных четных целых чисел. Количество суммируемых чисел должно вводиться во время работы программы. Ниже приведен рекомендуемый вид экрана во время работы программы (данные, введенные пользователем, выделены полужирным шрифтом).

```
Вычисление суммы четных положительных чисел.
Введите количество суммируемых чисел и нажмите <Enter>
-> 12
Сумма первых 12 положительных четных чисел равна 156
```

8. Написать программу, которая вычисляет сумму первых **n** членов ряда: 1, 3, 5, 7 ... Количество суммируемых членов ряда задается во время работы программы. Ниже приведен рекомендуемый вид экрана во время работы программы (данные, введенные пользователем, выделены полужирным шрифтом).

```
Вычисление частичной суммы ряда: 1,3,5,7 ...
Введите количество суммируемых членов ряда -> 15
Сумма первых 15 членов ряда равна 330
```

9. Написать программу, которая вычисляет сумму первых **n** членов ряда: 1 + 1/2 + 1/3 + 1/4 + … Количество суммируемых членов ряда задается во время работы программы. Ниже приведен рекомендуемый вид экрана (данные, введенные пользователем, выделены полужирным шрифтом).

```
Вычисление частичной суммы ряда: 1 + 1/2 + 1/3 + 1/4 + …
Введите кол-во суммируемых членов ряда -> 15
Сумма первых 15 членов ряда равна 3.3182
```

10. Написать программу, которая выводит таблицу степеней двойки от нулевой до десятой. Ниже приведен рекомендуемый вид экрана во время работы программы.

```
Таблица степеней двойки
- - - - - - - - -- - - -
0             1
1             2
2             4
3             8
4            16
5            32
6            64
7           128
8           256
9           512
10         1024
```

11. Написать программу, которая вычисляет факториал введенного с клавиатуры числа. (Факториалом числа **n** называется произведение целых чисел от 1 до **n**. Например, факториал 1 равен 1, 8 – 40320).

```
Вычисление факториала.
Введите число, факториал которого надо вычислить
-> 7
Факториал 7 равен 5040
```

12. Написать программу, которая выводит таблицу значений функции $y = -2{,}4x^2+5x-3$ В диапазоне от –2 до 2, с шагом 0,5. Ниже приведен рекомендуемый вид экрана во время работы программы.

```
--------------
   x  |   y
--------------
  -2  | -22.60
-1.5  | -15.90
  -1  | -10.40
-0.5  |  -6.10
   0  |  -3.00
 0.5  |  -1.10
   1  |  -0.40
 1.5  |  -0.90
   2  |  -2.60
---------------
```

13. Написать программу, которая вычисляет среднее арифметическое вводимой с клавиатуры последовательности дробных чисел. Количество чисел должно задаваться во время работы программы. Рекомендуемый вид экрана приведен ниже.

```
    Вычисление среднего арифметического последовательности дробных чисел.
    Введите количество чисел последовательности -> 5
    Вводите последовательность. После ввода каждого числа нажимайте <Enter>
    -> 5.4
    -> 7.8
    -> 3.0
    -> 1.5
    -> 2.3
    Среднее арифметическое введенной последовательности: 4.00
    Для завершения нажмите <Enter>
```

14. Написать программу, которая вычисляет среднее арифметическое последовательности дробных чисел, вводимых с клавиатуры. После ввода последнего числа программа должна вывести минимальное и максимальное число последовательности. Количество чисел последовательности должно задаваться во время работы программы. Рекомендуемый вид экрана приведен ниже. Данные, введенные пользователем, выделены полужирным шрифтом.

```
Обработка последовательности дробных чисел.
Введите количество чисел последовательности -> 5
Вводите последовательность. После ввода каждого числа нажимайте
<Enter>
-> 5.4
-> 7.8
-> 3.0
-> 1.5
-> 2.3
Количество чисел: 5
Среднее арифметическое: 4.00
Минимальное число: 1.5
Максимальное число: 7.8
```

```
Для завершения нажмите <Enter>
```

15. Написать программу, которая выводит таблицу значений функции y=|x-2|+|x+1|. Диапазон изменения аргумента от -4 до 4, шаг приращения аргумента 0,5.

16. Напишите программу, которая выводит на экран квадрат Пифагора – таблицу умножения. Рекомендуемый вид экрана во время выполнения программы приведен ниже.

```
   1  2  3  4  5  6  7  8  9 10
1  1  2  3  4  5  6  7  8  9 10
2  2  4  6  8 10 12 14 16 18 20
3  3  6  9 12 15 18 21 24 27 30
4  4  8 12 16 20 24 28 32 36 40
5  5 10 15 20 25 30 35 40 45 50
6  6 12 18 24 30 36 42 48 54 60
7  7 14 21 28 35 42 49 56 63 70
8  8 16 24 32 40 48 56 64 72 80
9  9 18 27 36 45 54 63 72 81 90
```

Лабораторная работа №4

Задачи

1. Написать программу, вычисляющую сумму и среднее арифметическое последовательности положительных чисел, которые вводятся с клавиатуры. Ниже приведен рекомендуемый вид экрана во время выполнения программы (данные, введенные пользователем, выделены полужирным шрифтом).

```
Вычисление среднего арифметического последовательности
положительных чисел.
Вводите после стрелки числа. Для завершения ввода введите ноль.
-> 45
-> 23
-> 15
-> 0
Введено чисел: 3
Сумма чисел: 83
Среднее арифметическое: 27.67
```

2. Написать программу, которая определяет максимальное число из введенной с клавиатуры последовательности положительных чисел (длина последовательности неограниченна). Ниже приведен рекомендуемый вид экрана во время выполнения программы (данные, введенные пользователем, выделены полужирным шрифтом).

```
Определение максимального числа последовательности положительных
чисел.
Вводите после стрелки числа. Для завершения ввода введите ноль.
-> 56
-> 75
-> 43
-> 0
Максимальное число: 75
```

3. Напишите программу, которая проверяет, является ли введенное пользователем целое число простым. Рекомендуемый вид экрана во время выполнения программы приведен ниже. Данные, введенные пользователем, выделены полужирным шрифтом.

```
Введите целое число и нажмите <Enter>
-> 45
45 – не простое число.
```

4. Написать программу, которая "задумывает" число в диапазоне от 1 до 10 и предлагает пользователю угадать число за 5 попыток. Ниже приведен рекомендуемый вид экрана во время выполнения программы (данные, введенные пользователем, выделены полужирным шрифтом).

```
Игра "Угадай число".
Компьютер "задумал" число от 1 до 10.
Угадайте его за 5 попыток.
Введите число и нажмите <Enter>
-> 5
Нет.
-> 3
Вы выиграли! Поздравляю!
```

5. Напишите программу, которая вычисляет число π с заданной пользователем точностью. Для вычисления значения числа π воспользуйтесь тем, что значение частичной суммы ряда 1-1/3+1/5-1/7+1/9-... при суммировании достаточно большого количества членов приближается к значению $\pi/4$. Рекомендуемый вид экрана во время выполнения программы приведен ниже (данные, введенные пользователем, выделены полужирным шрифтом).

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

6. Написать программу, которая вычисляет наибольший общий делитель двух целых чисел. Рекомендуемый вид экрана во время выполнения программы приведен ниже.

Массивы

Общие замечания

Приступая к решению задач этого раздела, следует вспомнить, что:

- массив – это структура данных, представляющая собой набор, совокупность элементов одного типа;
- в инструкции объявления массива указывается количество элементов массива;
- элементы массива нумеруются с нуля;
- доступ к элементу массива осуществляется путем указания индекса (номера) элемента. В качестве индекса можно использовать выражение

целого типа – константу или переменную. Индекс может меняться от 0 до n – 1, где *n* – количество элементов массива;

- доступ к элементам массива можно осуществить при помощи указателя;
- в инструкции объявления массива удобно использовать именованную константу, объявленную в директиве #define;
- для ввода, вывода и обработки массивов удобно использовать инструкции циклов (for, while);
- типичной ошибкой при использовании массивов является обращение к несуществующему элементу (выход индекса за допустимое значение).

Лабораторная работа №5

Задачи

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

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

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

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

3. Написать программу, которая вычисляет среднюю за неделю температуру воздуха. Исходные данные должны вводиться во время работы программы. Рекомендуемый вид экрана приведен ниже (данные, введенные пользователем, выделены полужирным шрифтом).

```
Введите температуру воздуха за неделю.
Понедельник -> 12
Вторник -> 10
Среда -> 16
Четверг -> 18
Пятница ~> 17
Суббота -> 16
Воскресенье -> 14
Средняя температура за неделю: 14.71 град.
```

4. Написать программу, которая проверяет, представляют ли элементы введенного с клавиатуры массива возрастающую последовательность.

5. Написать программу, которая методом обмена ("пузырька") сортирует по убыванию введенный с клавиатуры одномерный массив.

6. Написать программу, которая объединяет два упорядоченных по возрастанию массива в один, также упорядоченный массив. Рекомендуемый вид экрана во время работы программы приведен ниже, данные, введенные пользователем, выделены полужирным шрифтом.

```
Объединение двух упорядоченных по возрастанию массивов.
Введите в одной строке элементы первого массива, (5 целых чисел)
 -> 1 3 5 7 9
Введите в одной строке элементы второго массива, (5 целых чисел)
 -> 2 4 6 8 10
Массив – результат: 1 2 3 4 5 6 7 8 9 10
Для завершения работы нажмите <Enter>.
```

7. Написать программу, которая, используя метод бинарного поиска, выполняет поиск в упорядоченном по возрастанию массиве.

8. Написать программу, которая вводит по строкам с клавиатуры двумерный массив и вычисляет сумму его элементов по столбцам.

9. Написать программу, которая определяет номер строки квадратной матрицы, сумма элементов которой максимальна.

10. Написать программу, которая проверяет, является ли введенная с клавиатуры квадратная матрица "магическим" квадратом. "Магическим" квадратом называется матрица, у которой сумма чисел в каждом горизонтальном ряду, в каждом вертикальном и по каждой из диагоналей одна и та же (см. приведенный ниже рисунок).

2	9	4
7	5	3
6	1	8

13	8	12	1
2	11	7	14
3	10	6	15
16	5	9	4

Символы и строки

Общие замечания

Приступая к решению задач этого раздела, следует вспомнить, что:

- Каждому символу соответствует число – код символа.
- В C/C++ строка – это массив символов.
- Последним символом строки обязательно должен быть нуль-символ, код которого равен 0, и который в тексте программы изображается так: \0.

- Сообщения или подсказки, используемые в программе, удобно представить как массив указателей на строки и инициализировать массив, задать сообщения, в инструкции объявления массива:
  `char *mes[] = {"Сообщение 1","Сообщение 2", ... };`
- Если вводимая во время работы программы строка содержит пробелы, то функция scanf вводит только часть строки, до первого пробела, а функция gets – всю строку, в том числе и соответствующий клавише <Enter> символ "" \n.

Пример 1

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

```
Как Вас зовут?
Введите свое имя и фамилию, затем нажмите <Enter>
-> Вася Иванов
Здравствуйте, Вася Иванов!
```

```
// Приветствие
#include <conio.h>
#include <iostream>
#include <Windows.h>
using namespace std;

main()
{
    setlocale(LC_ALL, "Russian");  // in <iostream>
    SetConsoleCP(1251);       // in  <Windows.h>
    SetConsoleOutputCP(1251);
    char name[15]; // имя
    char fam[20]; // фамилия
    printf("Как Вас зовут?\n");
    printf("Ведите свое имя и фамилию, затем нажмите <Enter>");
    printf("-> ");
    scanf("%s", &name);
    scanf("%s", &fam);
    // функция scanf читает из буфера клавиатуры символы
    // до разделителя - пробела
    printf("Здравствуйте, %s %s!\n", name, fam);
    printf("\nДля завершения нажмите <Enter>");
    getch();
}
```

Пример 2

Написать программу, которая запрашивает у пользователя имя и отчество, затем здоровается с ним. Для ввода используйте функцию `getch()`.

```
// Приветствие (посимвольный ввод строки)
#include <conio.h>
#include <iostream>
#include <Windows.h>
using namespace std;

main()
{
   setlocale(LC_ALL, "Russian");  // in <iostream>
   SetConsoleCP(1251);        // in  <Windows.h>
   SetConsoleOutputCP(1251);
   char name[40]; // имя и отчество пользователя
   char ch;
   int i;
   printf("Как Вас зовут?\n");
   printf("(введите свое имя, отчество и нажмите <Enter>");
   printf("-> ");
   i = 0;
   while ((ch=getch()) != 13 && i < 40) // пока не <Enter>
   {
      putch(ch);
      name[i++] = ch;
   }
   name[i] = '\0';
   printf("\nЗдравствуйте, %s!\n", name);
   printf("\nДля завершения нажмите <Enter>");
   getch();
}
```

Пример 3

Написать программу, которая выводит на экран сообщение в "телеграфном" стиле: буквы сообщения должны появляться по одной, с некоторой задержкой.

```
// Посимвольный вывод сообщения
#include <stdio.h>
#include <conio.h>
#include <iostream>
#include <Windows.h> // для доступа к функции delay
main()
{
```

```
     setlocale(LC_ALL, "Russian"); // in <iostream>
     char msg[] = "\n\rПриветствую великого программиста!\0";
     int i; // номер символа
     i = 0;
     while(msg[i])
     {
          putch(msg[i++]);
          Sleep(150);
     }
     printf("\n\nДля завершения нажмите <Enter>");
     getch();
}
```

Пример 4

Написать программу, которая выводит на экран печатную часть таблицы кодировки символов (символы с кодами от 32 до 255). Таблица должна состоять из восьми колонок (*Код: Символ*).

```
// ASCII таблица кодировки символов
#include <stdio.h>
#include <conio.h>
#include <iostream>
main()
{
     // Если ch объявить как char, то буквам русского
     // алфавита будут соответствовать отрицательные числа
     unsigned char ch; // символ
     setlocale(LC_ALL, "Russian"); // in <iostream>
     int i,j;
     printf("\nASCII таблица кодировки символов\n");
     for (i = 0; i < 28; i++) // 28 строк
     {
          ch = i + 32;
          for (j = 1; j <= 8; j++) // восемь колонок
          {
               printf("%3c -%4i", ch, ch);
               ch += 28;
          }
          printf("\n");
     }
     printf("\nДля завершения нажмите <Enter>");
     getch();
}
```

Пример 5

Написать программу, которая в введенной с клавиатуры строке преобразует строчные буквы русского алфавита в прописные (учтите, что стандартная функция `upcase` с символами русского алфавита не работает).

```
// Преобразование прописных букв в строчные
#include <conio.h>
#include <iostream>
#include <cstdio>
#include <Windows.h>
main()
{
     setlocale(LC_ALL, "Russian"); // in <iostream>
     SetConsoleCP(1251); // in <Windows.h>
     SetConsoleOutputCP(1251);
     char st[80]; // строка текста
     int i; // номер обрабатываемого символа
     printf("\nВведите строку текста и нажмите <Enter>");
     printf("->");
     gets(st);  // in <cstdio>
     i = 0;
     while ( st[i] )
     {
          if ((st[i] >= 'a' && st[i] <= 'z')
             || (st[i] >= 'a' && st[i] <= 'я'))
               st[i] -= 32;
          else if (st[i] == 'ё')
               st[i] = 'Ё';
          i++;
     }
     printf("\n%s\n", st);
     printf("\nДля завершения нажмите <Enter>");
     getch();
}
```

Лабораторная работа №6

Задачи

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

```
Введите символ и нажмите <Enter>.
Для завершения введите точку.
```

```
-> 1
Символ: 1 Код: 49
-> 2
Символ: 2 Код: 50
-> ы
Символ: ы Код: 235
- > .
```

2. Написать программу, которая проверяет, является ли введенная с клавиатуры строка целым числом. Рекомендуемый вид экрана во время выполнения программы приведен ниже (данные, введенные пользователем, выделены полужирным шрифтом).

```
Введите число и нажмите <Enter>
-> 23.5
Введенная строка не является целым числом.
```

3. Написать программу, которая проверяет, является ли введеннаяс клавиатуры строка шестнадцатеричным числом.

4. Написать программу, которая проверяет, является ли введеннаяс клавиатуры строка дробным числом.

5. Написать программу, которая преобразует введенное с клавиатуры восьмиразрядное двоичное число в десятичное. Рекомендуемый вид экрана во время выполнения программы приведен ниже (введенные пользователем данные выделены полужирным шрифтом).

```
Введите восьмиразрядное двоичное число
и нажмите <Enter>
-> 11101010
Двоичному числу 11101010 соответствует десятичное 234
Для завершения нажмите <Enter>
```

6. Написать программу, которая преобразует введенное с клавиатуры двухразрядное шестнадцатеричное число в десятичное.

7. Написать программу, которая преобразует введенное пользователем десятичное число в число в указанной системе счисления (от 2 до 10). Рекомендуемый вид экрана во время выполнения программы приведен ниже.

```
Введите целое число -> 67
Введите основание системы счисления -> 2
Десятичному числу 67 соответствует число 100011 по основанию 2
```

8. Написать программу, которая преобразует введенное пользователем десятичное число в шестнадцатеричное.

9. Написать программу, которая вычисляет значение выражения $N_0O_1N_2...O_kN_k...$, где N_k – целое одноразрядное число, O_k – один из двух знаков простейших арифметических действий: сложения (+) или вычитания.

Ниже приведен рекомендуемый вид экрана во время работы программы (данные, введенные пользователем, выделены полужирным шрифтом).

```
Введите арифметическое выражение,
например, 4+5-3-5+2, и нажмите <Enter>
-> 9-5+4+2-6
Значение введенного выражения: 4
Для завершения программы нажмите <Enter>
```

Функции

Общие замечания

Приступая к решению задач этого раздела, следует вспомнить, что:

- Передавать данные в функцию следует только с помощью параметров. Глобальные переменные, т. е. переменные, объявленные вне функции, использовать не рекомендуется.
- Тип каждого фактического параметра (константы или переменной) в инструкции вызова функции должен соответствовать типу соответствующего формального параметра, указанного в инструкции объявления функции.
- Если параметр функции служит для возврата результата, то в объявлении функции этот параметр должен быть ссылкой, а в инструкции вызова функции в качестве фактического параметра должен использоваться адрес переменной.

Пример 1

Написать функцию пересчета температуры из градусов Фаренгейта в градусы Цельсия ($C° = 5/9 \cdot (F° – 32)$) и программу, использующую эту функцию, которая выводит на экран таблицу соответствия температур в шкалах Фаренгейта и Цельсия.

```
// Функция Fahr2Cels пересчитывает температуру
// из градусов Фаренгейта в градусы Цельсия
#include "stdio.h"
#include "conio.h"
#include <iostream>

float Fahr2Cels(float f)
{
     float c;
     c = (float) 5/9*(f - 32);
     return (c);
```

```
    // Вместо приведенных выше инструкций
    // можно написать:
    // return ( (float)5/9*(f - 32));
}
int main()
{
    using namespace std;
    setlocale(LC_ALL, "Russian");
    float f; // температура в градусах Фаренгейта
    float c; // температура в градусах Цельсия
    float f1,f2; // диапазон изменения температуры
    float df; // шаг изменения температуры
    f1 = 0;
    f2 = 5;
    df = 0.5;
    printf("\n----------------");
    printf("\n F   C");
    printf("\n----------------");
    f = f1;
    do
    {
        c = Fahr2Cels(f);
        printf("\n%5.2f %5.2f", f, c);
        f = f + df;
    }
    while ( f <= f2 );
    printf("\n----------------");
    printf("\nДля завершения нажмите <Enter>");
    getch();
    return 0;
}
```

Пример 2

Написать функцию, которая вычисляет объем цилиндра.

```
#include <stdio.h>
#include <conio.h>
#include <iostream>
#include <math.h> // для доступа к M_PI
// объем цилиндра
float vcil(float h, float r)
{
    return(M_PI*r*r*h);
}
```

```
int main()
{
     using namespace std;
     setlocale(LC_ALL, "Russian");
     float r,h; // высота и радиус основания цилиндра
     float v; // объем цилиндра
     puts("Вычисление объема цилиндра");
     printf("Введите высоту и радиус основания ->");
     scanf("%f%f", &h, &r);
     v = vcil(h, r);
     printf("Объем цилиндра %3.2f\n", v);
     printf("Для завершения нажмите <Enter>");
     getch();
     return 0;
}
```

Лабораторная работа №7

Задачи

1. Написать функцию пересчета длины из дюймов в миллиметры (1 дюйм = 2,54 см).

2. Написать функцию пересчета расстояния из миль в километры (1 миля = 1,60094 км).

3. Написать функцию пересчета цены нефти за баррель в цену за тонну (1 нефтяной баррель марки Urals равен 136,4 кг). Для проверки работоспособности функции написать программу, использующую эту функцию для пересчета цены за баррель в цену за тонну.

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

5. Написать функцию **profit**, которая вычисляет доход по вкладу. Исходные данные для функции: величина вклада, процентная ставка (годовых) и срок вклада (количество дней).

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

7. Написать функцию, которая выводит на экран строку символов. Длина строки и символ являются параметрами функции.

8. Чтобы из трех отрезков можно было составить треугольник, необходимо и достаточно, чтобы сумма длин любых двух отрезков была строго больше третьего.

Напишите функцию **triangle**, которая принимает на вход три длины отрезков и определяет, можно ли из этих отрезков составить треугольник. Ваша функция должна печатать «Это треугольник», если составить треугольник можно, и «Это не треугольник», если нельзя.

9. Напишите функцию **num_digits**, возвращающую число цифр в десятичном натуральном числе.

Файлы

Общие замечания

Приступая к решению задач этого раздела, следует вспомнить, что:

- В программе, которая выполняет операции чтения из файла или запись в файл, должна быть объявлена переменная-указатель на тип FILE.
- Для того чтобы файл стал доступен, его нужно открыть, указав, для выполнения какой операции открывается файл (чтение, запись или обновление данных) и тип файла (двоичный или текстовый).
- При работе с файлами возможны ошибки. Поэтому рекомендуется проверять результат выполнения потенциально опасных, с точки зрения возникновения ошибок, операций с файлами (например, fopen).
- Если в инструкции открытия файла путь к файлу не указан, то по умолчанию предполагается, что файл находится в том же каталоге, что и выполняемый файл программы.
- При записи в тексте программы пути к файлу, следует помнить, что символ \ в строковых константах обозначает начало специальной последовательности символов (например, \n). Поэтому, чтобы путь к файлу был интерпретирован правильно, символ \ следует продублировать (например, c:\\temp).
- Чтение данных из текстового файла обеспечивает функция fscanf, запись – fprintf.
- По завершении работы с файлом его нужно обязательно закрыть (функция fclose).

Функции работы с файлами

fopen

Синтаксис:

```
FILE* fopen(const char * Имя, const char* Режим)
```

Открывает файл с указанным именем для действия, которое задается параметром *Режим*.

Режим	Действие
r	Только чтение. Файл открывается только для чтения
w	Только запись. Файл открывается для записи. Если файл с именем, указанным в качестве первого параметра функции fopen, уже существует, то новые данные записываются поверх старых, т. е. старый файл фактически уничтожается
A	Добавление. Файл открывается для записи данных в конец существующего файла. Если файл с именем, указанным в качестве первого параметра функции fopen, не существует, то он будет создан

Если файл открывается как текстовый, то после символьной константы, определяющей режим открытия файла, нужно добавить символ **t.** Например, строка **rt** задает, что для чтения открывается текстовый файл.

В случае успешного открытия файла функция `fopen` возвращает указатель на поток, из которого можно читать или в который можно записывать. Если по какой-либо причине операция открытия файла не была выполнена, `fopen` возвращает *NULL*.
В этом случае, чтобы получить информацию о причине ошибки, следует обратиться к функции `ferror`.
Заголовочный файл: `<stdio.h>`

fprintf

Синтаксис:

```
int fprintf(FILE *Поток, Формат, СписокПеременых);
```

Выполняет форматированный вывод (см. printf) в файл, связанный с потоком, указанным в качестве первого параметра.
Файл, связанный с потоком, должен быть открыт как текстовый, в режиме, допускающем запись (см. fopen).
Заголовочный файл: `<stdio.h>`

fscanf

Синтаксис:

```
int fscanf(FILE *Поток, const char* Формат, СписокАдр);
```

Выполняет форматированное (см. scanf) чтение значений переменных из файла, связанного с потоком, указанным в качестве первого параметра.
Файл, связанный с потоком, должен быть открыт как текстовый, в режиме, допускающем чтение (см. fopen).
Заголовочный файл: `<stdio.h>`

fgets

Синтаксис:

```
char* fgets(char *Строка, int КолСимволов, FILE *Поток)
```

Читает из указанного потока символы и записывает их в строку, указанную при вызове функции. Чтение заканчивается, если прочитан символ с номером *КолСимволов-1* или если очередной символ является символом новой строки. Прочитанный из файла символ новой строки заменяется нулевым символом.
Файл, связанный с потоком, должен быть открыт как текстовый в режиме, допускающем чтение (см. fopen).
Заголовочный файл: `<stdio.h>`

fputs

Синтаксис:

```
char* fputs(char *Строка, FILE *Поток)
```

Записывает в указанный поток строку символов. Символ конца строки, нуль-символ, в поток не записывается.
Файл, связанный с потоком, должен быть открыт как текстовый в режиме, допускающем запись (см. fopen).
Заголовочный файл: `<stdio.h>`

ferror

Синтаксис:

```
int ferror(FILE* Поток)
```

Возвращает ненулевое значение, если последняя операция с указанным потоком завершилась ошибкой.
Заголовочный файл: `<stdio.h>`

feof

Синтаксис:

```
int feof(FILE* Поток)
```

Возвращает ненулевое значение, если в результате выполнения последней операции чтения из потока достигнут конец файла.
Заголовочный файл: `<stdio.h>`

fclose

Синтаксис:

```
int fclose(FILE* Поток)
```

Закрывает указанный поток.
Заголовочный файл: `<stdio.h>`

Пример 1

Написать программу, которая создает на диске компьютера файл numbers.txt и записывает в него 5 целых чисел, введенных пользователем с клавиатуры. Откройте созданный программой файл и убедитесь, что каждое число находится в отдельной строке.

```
// Создает на диске файл
```

```
#include <stdio.h>
#include <conio.h>
#include <Windows.h>
#include <iostream>
#define FNAME "numbers.txt\0" // имя файла
#define N 5 // количество чисел
// Создает на диске файл и записывает в него
// целые числа, введенные пользователем
int main()
{
     setlocale(LC_ALL, "Russian"); // in <iostream>
     SetConsoleCP(1251); // in <Windows.h>
     SetConsoleOutputCP(1251);
     char fname[20] = FNAME;
     FILE *f; // файл чисел
     int n; // число
     puts("\nСоздание файла");
     printf("Веденные числа будут записаны в файл %s\n",
     fname);
     puts("После ввода каждого числа нажимайте <Enter>\n");
     // Открыть файл в режиме записи (w) текста (t)
     // Если файл с таким именем уже есть, то новые
     // данные будут записаны поверх старых
     // Для добавления в конец файла, используйте
     // режим добавления (a)
     if ((f = fopen(fname, "wt")) == NULL)
     {
          printf("Ошибка открытия файла для записи");
          getch();
          return 1;
     }
     for (int i = 0; i < N; i++)
     {
          printf("->");
          scanf("%i", &n);
          fprintf(f,"%i\n",n);
     }
     fclose(f); // закрыть файл
     printf("Введенные числа записаны в файл %s\n", fname);
     puts("\nДля завершения нажмите <Enter>");
     getch();
}
```

Пример 2

Написать программу, которая дописывает в файл numbers.txt три целых числа, введенных пользователем. Убедитесь, открыв файл при помощи редактора текста, что в файле находятся 8 чисел.

```
// добавляет данные в файл
#include "stdio.h"
#include "conio.h"
#include <Windows.h>
#include <iostream>

#define FNAME "numbers.txt\0" // имя файла
#define N 3 // количество чисел
// Дописывает в находящийся на диске файл numbers.txt
// целые числа, введенные пользователем
int main()
{
     setlocale(LC_ALL, "Russian"); // in <iostream>
     SetConsoleCP(1251); // in <Windows.h>
     SetConsoleOutputCP(1251);
     char fname[20] = FNAME;
     FILE *f; // файл чисел
     int n; // число
     puts("\nДобавление в файл");
     printf("Веденные числа будут добавлены в файл %s\n",
fname);
     puts("После ввода каждого числа нажимайте <Enter>\n");
     // Открыть файл в режиме добавления (a) текста (t)
     // Если файла с таким именем нет, то он будет создан
     if ((f = fopen(fname, "at")) == NULL)
     {
          printf("Ошибка открытия файла для добавления");
          getch();
          return 1;
     }
     for (int i = 0; i < N; i++)
     {
          printf("->");
          scanf("%i", &n);
          fprintf(f,"%i\n",n);
     }
     fclose(f); // закрыть файл
     printf("Введенные числа добавлены в файл %s\n", fname);
```

```
    puts("\nДля завершения нажмите <Enter>");
    getch();
}
```

Пример 3

Написать программу, которая выводит на экран содержимое файла numbers.txt.

```
// Выводит на экран содержимое файла
#include "stdio.h"
#include "conio.h"
#include <Windows.h>
#include <iostream>
#define FNAME "numbers.txt\0" // имя файла
int main()
{
    setlocale(LC_ALL, "Russian"); // in <iostream>
    SetConsoleCP(1251); // in <Windows.h>
    SetConsoleOutputCP(1251);
    char fname[20] = FNAME;
    FILE *f; // текстовый файл
    char st[80]; // строка из файла
    printf("\nСодержимое файла %s\n", fname);
    puts("---------------------------------");
    // Открыть файл в режиме чтения (r) текста (t)
    if ((f = fopen(fname, "rt")) == NULL)
    {
        printf("Ошибка открытия файла для чтения");
        getch();
        return 1;
    }
    while (!feof(f))
    {
        fscanf(f,"%s", &st);
        if (feof(f)) break;
        printf("%s\n", st);
    }
    fclose(f); // закрыть файл
    puts("---------------------------------");
    puts("\nДля завершения нажмите <Enter>");
    getch();
}
```

Лабораторная работа №8

Задачи

1. Написать программу, которая вычисляет среднее арифметическое чисел, находящихся в файле numbers.txt.

2. Написать программу, которая записывает в файл данные, находящиеся в двумерном массиве дробного типа.

3. Написать программу, которая загружает из файла данные в двумерный массив дробного типа.

4. Написать программу, которая позволяет просматривать текстовые файлы (выводит на экран их содержимое), например, файлы исходных программ C++. Имя просматриваемого файла должно вводиться пользователем во время работы программы.

5. Написать программу, которая дописывает в находящийся на диске компьютера файл contacts.txt имя, фамилию и номер телефона, например, вашего товарища. Если файла на диске нет, то программа должна создать его. В файле каждый элемент данных (имя, фамилия, телефон) должен находиться в отдельной строке. Ниже приведен рекомендуемый вид экрана во время работы программы.

```
Добавление информации в телефонный справочник
Фамилия -> Сидоров
Имя -> Вася
Телефон -> 234-84-37
Информация добавлена
Для завершения нажмите <Enter>
```

6. Усовершенствуйте программу работы с телефонным справочником (см. задачу 5) так, чтобы за один сеанс работы в файл contacts.txt можно было добавить информацию о нескольких людях. Рекомендуемый вид экрана во время работы программы приведен ниже.

```
Добавление информации в телефонный справочник
Для завершения вместо ввода фамилии нажмите <Enter>
Фамилия -> Иванов
Имя -> Иван
Телефон -> 234-84-37
Информация добавлена
Фамилия -> Орлов
Имя -> Андрей
Телефон -> 552-18-40
Информация добавлена
Фамилия ->
Ввод завершен
Для завершения нажмите <Enter>
```

7. Написать программу, которая позволяет найти в телефонном справочнике (в файле contacts.txt) нужную информацию. Программа должна

запрашивать у пользователя фамилию человека и выводить информацию о нем. Если в справочнике есть люди с одинаковыми фамилиями, то программа должна вывести список всех этих людей. Рекомендуемый вид экрана во время работы программы приведен ниже.

```
Телефонный справочник
Введите фамилию и нажмите <Enter>. Для завершения работы с про-
граммой сразу после приглашения нажмите <Enter>
-> Петров
В справочнике данных о Петров нет.
-> Иванов
Иванов Вася 578-12-45
Иванов Сергей 244-34-02
->
```

8. Написать программу, которая объединяет возможности программ "Добавление в телефонный справочник" и "Поиск в телефонном справочнике". При запуске программы на экране должно отображаться меню, вид которого приведен ниже.

```
*** Телефонный справочник ***
1 - Добавление
2 - Поиск
0 - Завершение работы
Ваш выбор ->
```

Структуры в C++. Двоичные файлы

Структуры очень удобно записывать в двоичные файлы, поскольку можно за 1 раз прочитать или записать сразу одну или даже несколько структур. Вспомним, что при чтении из двоичного файла функции **fread** надо передать адрес нужной области памяти (куда записать прочитанные данные), размер одного блока и количество таких блоков. Для того, чтобы не вычислять размер структуры вручную (можно легко ошибиться), применяют оператор **sizeof**.

Пример 1

```
Book b;  // Book – название структуры
int n;
FILE *fp;
fp = fopen("books.dat", "rb");
n = fread(&b, sizeof(Book), 1, fp);
if (n == 0)
{
     printf("Ошибка при чтении из файла");
}
fclose (fp);
```

Можно было также вместо **sizeof(Book)** написать **sizeof(b),** поскольку запись **b** – это как раз один экземпляр структуры **Book**. Функция **fread** возвращает число удачно прочитанных элементов (в нашем случае – структур). Поэтому если в примере переменная **n** равно нулю, чтение закончилось неудачно и надо вывести сообщение об ошибке.

Для записи структуры в двоичный файл используют функцию **fwrite**. Ее параметры – те же, что и у **fread**. Пример ниже показывает добавление структуры в конец двоичного файла **books.dat**.

Пример 2

```
Book b;
FILE *fp;
// здесь надо заполнить структуру
// . . . . . .
fp = fopen("books.dat ", "ab");
fwrite(&b, sizeof(Book), 1, fp);
fclose (fp);
```

Копирование

Задача. Пусть в памяти выделено две структуры одного типа и в одну из них записаны какие-то данные. Требуется скопировать все данные из первой структуры во вторую.

Пусть структуры имеют тип **Book** и называются **b1** и **b2**. Существуют три способа решения этой задачи. Самый сложный – копирование каждого поля отдельно:

```
Book b1, b2;
// здесь заполняем структуру b1
// . . . . . .
strcpy ( b2.author, b1.author );
strcpy ( b2.title, b1.title );
b2.year = b1.year;
b2.pages = b1.pages;
```

Можно использовать специальную функцию **memcpy**, которая умеет копировать блоки памяти. Для ее использования надо подключить к программе заголовочный файл **mem.h**.

```
memcpy(&b2, &b1, sizeof(Book)); // куда, откуда, сколько байт
```

Самый простой способ – третий. Достаточно просто написать

```
b2 = b1;
```

При этом программа скопирует одну структуру в другую «бит в бит».

Массивы структур

Структуры служат для обработки большого объема информации, поэтому чаще всего в программе используются массивы структур. Они

объявляются так же, как обычно, но предварительно (выше) надо объявить саму структуру как новый тип данных.

Для обращения к полю структуры также используют точку, но теперь надо указать в квадратных скобках еще номер нужной структуры, например

```
Book A[20];
...
A[12].pages = 50;
for ( i = 0; i < 20; i ++ ) // цикл по всем структурам в массива
     puts(A[i].title); // вывести название книги
```

Если вы работаете с двоичными файлами, то чтение и запись всего массива структур выполняется в одну строчку. Покажем приемы работы с двоичным файлом на примере задачи.

Задача. В файле **books.dat** записаны структуры типа **Book**. Известно, что их не больше **100**.Требуется прочитать их в память, у всех книг установить **2009** год издания и записать обратно в тот же файл.

Поскольку по условию известно, что структур не больше 100, заранее выделяем в памяти массив на 100 структур.

```
Book b[100];
```

При чтении из файла пытаемся читать все 100 структур:

```
n = fread ( &b[0], sizeof(Book), 100, fp );
```

Чтобы определить, сколько структур было в действительности прочитано, используем значение n, которое функция fread возвращает в качестве результата. Вот полная программа:

```
#include <stdio.h>
struct Book
{ // объявление нового типа данных
    char author[40];
    char title[80];
    int year;
    int pages;
};
int main()
{
    Book b[100]; // выделение памяти под массив структур
    int i, n;
    FILE *fp;
    fp = fopen("books.dat", "rb"); // читаем 100 структур
    n = fread(&b[0], sizeof(Book), 100, fp); // прочитали n шт.
    fclose ( fp );
```

```
    for ( i = 0; i < n; i ++ )
       // обрабатываем все, что прочитали
    b[i].year = 2009;
    fp = fopen("books.dat", "wb"); // записываем n шт.
    fwrite ( b, sizeof(Book), n, fp );
    fclose ( fp );
}
```

Динамическое выделение памяти

Предположим, что надо создать массив структур, размер которого выясняется только во время работы программы. Для этого надо

1) объявить переменную типа указатель на нужный тип данных;

2) выделить память с помощью оператора **new** и запомнить адрес выделенного блока;

3) использовать новую область как обычный массив.

```
Book *B;
int n;
printf("Сколько у вас книг? ");
scanf ( "%d", &n ); // вводим размер массива
B = new Book[n]; // выделяем память
// здесь работаем с массивом B, как обычно
delete B; // освобождаем память
```

Иногда требуется выделить в памяти одну структуру. При этом мы получаем ее адрес, который записываем в переменную-указатель. Как получить доступ к полю структуры?

Один из вариантов – «разыменовать» указатель, то есть обратиться к той структуре, на которую он указывает. Если p – указатель на структуру типа Book, то обратиться к ее полю author можно как (*p).author. Эта запись означает «мне нужно поле author той структуры, на которую указывает указатель p».

В языке Си существует и другой способ обратиться к полю структуры: можно написать и p->author, что значит то же самое, что и (*p).author, но более понятно. В следующем примере динамически выделяется память на 1 структуру, ее поля считываются с клавиатуры, и затем структура записывается в конец текстового файла books.txt.

```
    Book *p;
    FILE *fp;
    p = new Book; // выделить память на 1 структуру
    printf("Автор "); // ввод полей через указатель
```

```
gets ( p->author );
printf("Название книги ");
gets ( p->title );
printf("Год издания, кол-во страниц ");
scanf("%d%d", &p->year, &p->pages);
fp = fopen("books.txt", "a"); // дописать в конец файла
fprintf("%s\n%s\n%d %d\n", // обращение через указатель
p->author, p->title, p->year, p->pages);
fclose ( fp );
delete p; // освободить память
```

Лабораторная работа №9

Задание:

В программах необходимо использовать только динамические структуры.

Выполнить следующие действия:

1. С помощью текстового редактора создать текстовый документ, в который занести исходную информацию. Рекомендуется вначале занести количество записей, а затем последовательно все записи, причем тестовые поля вводить в отдельной строке, а числовые – можно в отдельной строке или в одной через пробел или разделитель табуляцию.

Например, содержимое файла, который будет включать 7 записей – фамилия, группа, 5 оценок :

```
7
Иванов
Ф01-а
5 4 5 3 5
Петров
УА01-б
5 4 5 3 5
```

2. Написать первую программу, которая считывает информацию из созданного текстового файла и записывает ее в двоичный файл

3. Написать вторую программу, которая считывает информацию из двоичного файла, реализует поставленную задачу. После повторного чтения двоичного файла результат работы выводится в результирующий текстовый файл.

Все текстовые и двоичный файлы должны располагаться или в личной папке или в папке проекта.

Варианты заданий

1. Создать двоичный файл с информацией о перенесенных инфекционных заболеваниях учащимися средней школы (табл. 9.1).

Таблица 9.1 Статистика заболеваемости по школе

Болезнь	Количество больных					
	Январь	Февраль	Март	Апрель	Май	Июнь
Грипп	120	132	97	54	12	3
...	...	...	...	...	...	...

Добавить в файл поле «Средняя заболеваемость за полугодие». Упорядочить информацию в файле в порядке возрастания средней заболеваемости.

2. Создать двоичный файл с информацией о перенесенных инфекционных заболеваниях учащимися средней школы (см. табл. 9.1). Добавить в файл поле «Минимальная заболеваемость за полугодие». Упорядочить файл в порядке возрастания заболеваемости в январе.

3. Создать двоичный файл с информацией, приведенной в табл. 9.2. Добавить поле «Средний прирост населения». Упорядочить информацию, расположив названия областных центров в алфавитном порядке.

4. Создать двоичный файл с информацией, приведенной в табл. 9.2. Упорядочить информацию, расположив названия областных центров в порядке убывания среднего прироста населения.

Таблица 9.2. Прирост населения в крупных городах СНГ

Город	Прирост населения, в тыс. чел.				
	1996	1997	1998	1999	2000
Москва	50	54	60	49	63
Донецк	-14	13	-10	-8	-5
...	...	...	...	...	...

5. Создать двоичный файл с информацией об успеваемости студентов некоторого факультета за все время обучения (табл. 9.3).

Таблица 14.3 Средняя успеваемость

Название банка	Выданные ссуды, тыс. руб.				
	1999	2000	2001	2002	2003
Гамма:Банк	20	35	56	70	120
...	...	...	...	...	...

Добавить в файл поле «Средняя успеваемость». Удалить из файла информацию о студентах с средним баллом менее 3.5.

6. Создать двоичный файл с информацией об успеваемости студентов некоторого факультета за все время обучения (см. табл. 9.3). Переписать информацию в другой файл, расположив фамилии в алфавитном порядке и добавив поле «Средняя успеваемость».

7. Создать двоичный файл с информацией, приведенной в табл. 9.4.

Таблица 9.4 Ссуды, выданные банками

Название банка	Выданные ссуды, тыс. руб.				
	1999	2000	2001	2002	2003
Гамма:Банк	20	35	56	70	120
...	...	...	...	...	...

Добавить в файл поле «Общая сумма ссуд, выданных каждым банком», упорядочить информацию в файле в алфавитном порядке названий банков.

8. Создать двоичный файл с информацией, приведенной в табл. 9.4. Переписать информацию в другой двоичный файл, исключив информацию о банках с общей суммой ссуд менее 100 тыс. Упорядочить информацию в новом файле, расположив банки в алфавитном порядке.

9. Создать двоичный файл с информацией о продаже путевок некоторой туристической фирмой (табл. 9.5). Добавить поле «Продано путевок на сумму». Вывести на экран информацию о странах, в которые продано больше всего путевок.

Таблица 9.5. Продажа путевок

Страна	Цена путевки, долл.	Количество проданных путевок					
		Апрель	Май	Июнь	Июль	Август	Сентябрь
Греция	546	75	120	150	158	160	130
...	...	...	...	...	...	...	...

10.Создать двоичный файл с информацией о продаже путевок некоторой туристической фирмой (см. табл. 9.5). Добавить поле «Среднее количество проданных путевок». Упорядочить файл в порядке убывания информации в поле «Продано путевок на сумму».

Лабораторная работа №10

Приём командно-строковых аргументов

Командно-строковые аргументы передаются непосредственно в функции main: чтобы воспользоваться ими, необходимо полностью объявить её (все функции main, которые вы видели ранее, имели пустой список аргументов). На самом деле функция main принимает два параметра: количество аргументов в командной строке и полный список аргументов.

Полное объявление функции выглядет следующим образом:

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

Целое число argc хранит количество аргументов, переданное программе через командную строку, с учётом имени программы.

Массив указателей на символы содержит все аргументы. Элемент argv[0] хранит имя программы, а последующие элементы с номерами меньшими, чем argc, – командно-строковые аргументы. Все элементы массива можно использовать как строки.

Пример

Вывод на печать всех аргументов, переданных программе

```
#include <iostream>
using namespace std;
int main(int argc, char *argv[])
{
     setlocale(LC_ALL, "Russian");
     for (int i=0; i<argc; i++)
     {
          cout << i << ". " << argv[i]<< endl;
     }
     system("pause");
     return 0;
}
```

Сохраните файл программы под именем **Ex10_00.cpp**. Скомпилируйте его и затем в командной строке, находясь в той же папке, где и файл **Ex10_00.cpp,** наберите и запустите на выполнение Ex10_00.exe первый второй третий.

Задачи

1. Напишите программу, которая берет из параметров командной строки произвольное количество параметров – чисел. При этом параметры могут быть ошибочными – соответственно ошибки надо правильно обработать.

Вам необходимо посчитать сумму всех параметров, причем с чередующимися знаками. Это означает, что первое встретившееся число будет учтено в сумме с плюсом, второе – с минусом, третье – опять с плюсом и т.д.

Если параметры не было переданы, то следует вывести фразу **NO PARAMS**, а в случае других ошибок – название ошибки.

2. Напишите программу, которая разбирает аргументы командной строки вида "ключ=значение" и выводит их на отдельной строке в виде "Key: ключ Value: значение".

Кроме описанных аргументов программа может принимать необязательную опцию --sort, которая позволяет отсортировать выводимые значения по ключу.

Пример

Ввод

```
solution.exe --sort surname=Ivanov name=Vasya age=42
```

Вывод

```
Key: age     Value: 42
Key: name    Value: Vasya
Key: surname   Value: Ivanov
```

3. Напишите программу, которая в качестве аргумента принимает имя файла (не указан файл или указан несуществующий – ошибка) и выводит его содержимое на экран.

В добавок, программа может принимать дополнительные агрументы:

- «--count» для вывода кол-ва строк в конце сообщения,
- «--num» для вывода порядкового номера с пробелом в начале каждой строки,
- «--sort» для сортировки строк в алфавитном порядке перед выводом.

Пусть файл **text1.txt** содержит строки:

Houston
we have
a problem

Пример 1

Ввод

```
solution.exe --num text1.txt
```

Вывод

```
0 Houston
1 we have
2 a problem
```

Пример 2

Ввод

```
solution.exe --count --sort text1.txt
```

Вывод

```
Houston
a problem
we have
rows count: 3
```

Пример 3

Ввод	Вывод
`solution.exe --count --sort textX.txt`	`ERROR`

Примечание

При наличии ошибки необходимо вывести слово **ERROR**