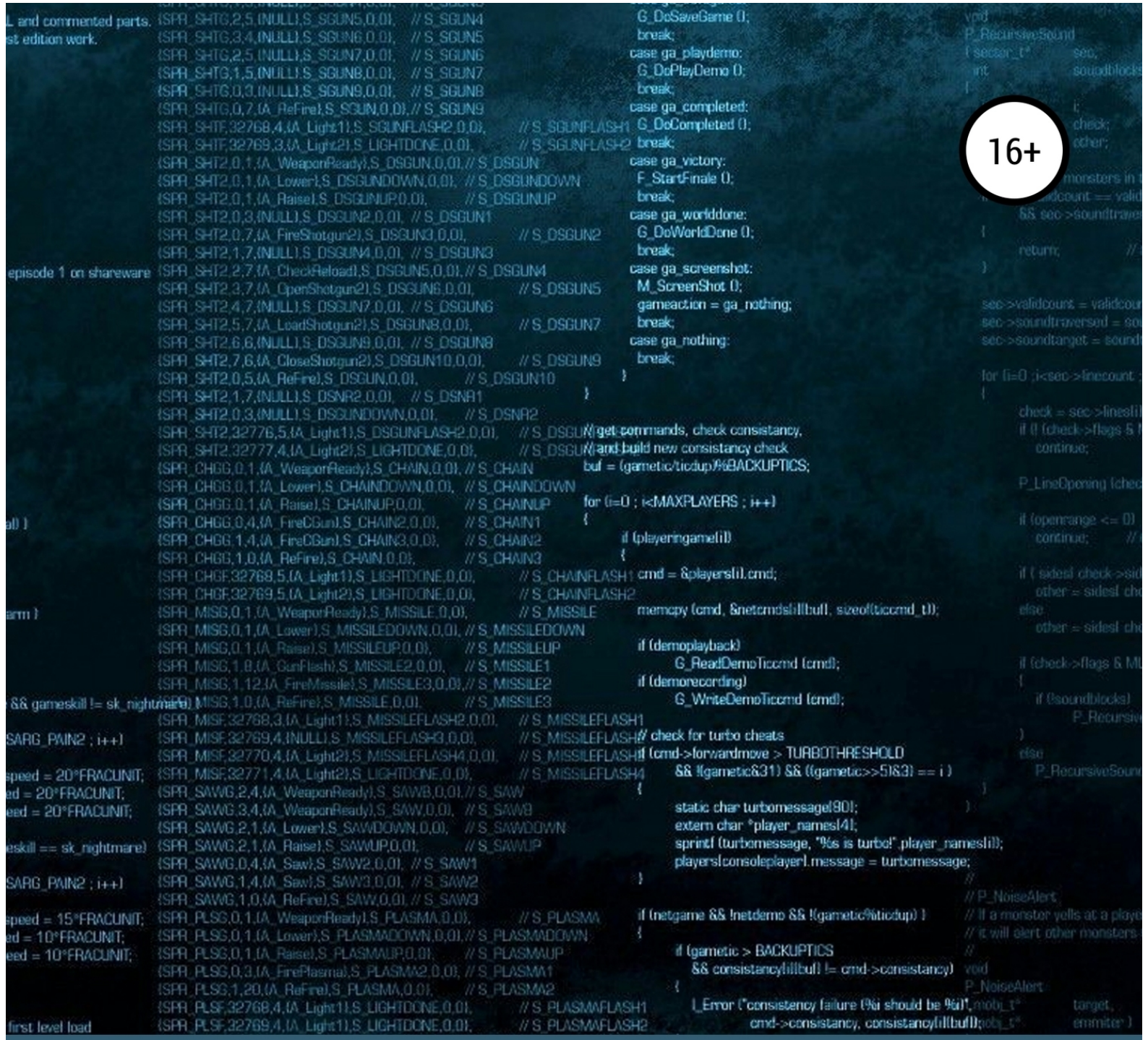

Станислава Солнечная

Программирование на C, C++

Станислава Солнечная

Программирование на C, C++

«ЛитРес: Самиздат»

2020

Солнечная С.

Программирование на С, С++ / С. Солнечная — «ЛитРес: Самиздат», 2020

Задача данной книги простым и доступным языком объяснить примеры использования С, С++ и основные возможности С, С++. Изложено кратко о некоторых инструментах и их использовании на практике. Также даны сведения об аппаратном обеспечении вычислительной техники, для представления механизма программирования и управления компьютера, необходимо дать понять как работает компьютер, компилятор, отладчик и т.д. с языком программирования . Каждый раздел книги наделен примерами. Дополненный материал в следующих изданиях. Кратко даны различные возможности для ознакомления, изучение их за пределами книги, так как они заслуживают тщательного и глубокого погружения.

Начало

Вычислительная техника создавалась для обработки информации. Информация бывает звуковая, графическая, текстовая и т.д. Компьютер не общается на естественном нам языке. Он общается последовательностями 0 и 1. Язык программирования – это команды компьютеру, что-то выполнить, сделать. Есть языки высокого уровня и машинные языки, например. Мы изучим язык С, С++. Почему С, С++? На языке С написаны большинство операционных систем и языков программирования. Зная один язык, легко освоить другой язык.

Мы научимся давать компьютеру простые команды. Напишем первую программу, см. Листинг 1.

Листинг 1

Первая программа

```
1 #include<stdio.h>
2 main()
3 {
4 printf(“Привет!”);
5 }
```

В 1 строке мы подключаем библиотеку. В программах есть функции. В библиотеке хранятся самые употребляемые функции. Во второй строке объявляем функцию main (). Далее с 3 по 5 строку тело функции, оно взято в фигурные скобки: ‘{‘, ‘}’. В четвертой строке функция библиотеки из файла stdio.h. Эта функция выводит на экран строку с символами: “Привет!”. Все строки заключаются в двойные кавычки, например, “слова”, литералы в одинарные кавычки, например, ‘в’.

Рекомендуется выучить наизусть написание простой программы. Для того, чтобы легче выучить, следует пописать похожие простые программы, на практике быстрее учиться.

Каждый оператор заканчивается точкой с запятой, делается это для компилятора, которому объявляется, где конец одного оператора и начало другого.

Если при выполнении программы, русская кодировка выдается у вас в консоли белибердой, то возможно добавить строки 2 и 5:

Листинг 2

Первая программа

```
1 #include<stdio.h>
2 #include<stdlib.h>
3 main()
4 {
5 system("chcp 1251 > nul");
6 printf(“Привет!”);
7 }
```

Функция system заголовочного файла stdlib.h передает строку "chcp 1251 > nul" в операционную систему для выполнения. Возможны и другие настройки.

Усложним первую программу, научим компьютер обращаться к нам по имени, см. Листинг 3.

Листинг 3

Программа: “Знакомство”

```
1 #include<stdio.h>
2 main()
3 {
4 char b[10];
5 printf(“Привет! Как тебя зовут?\n”);
6 scanf(“%s”,&b);
```

```
7 printf(“Привет! %s”,b);
8 }
```

В 4 строке объявляется массив из 10 переменных символьного типа b. Переменная – это имя какого-то участка памяти. В 5 строке оператор выводит на консоль строку, формат задан, переводит на новую строку: ‘\n’ – управляющий символ, символ перевода строки. В 6 строке считывается ввод с консоли, задается формат считывания – %s, означает, что считывается строка, &b – адрес, по которому будет сохранен массив символов. В строке 7 вывод на консоль форматированной строки, %s – означает, что будет выведена строка, b – это та строка, которую выведут на экран (форматирование %s).

Задания:

Написать простой диалог: Привет! Как тебя зовут! – Как дела?

Выучить написание простой программы.

Оператор if-else

Формальный синтаксис:

```
if (выражение)
оператор1
else
оператор2
```

Напишем следующую программу с использованием оператора if-else. Пользователь вводит число, компьютер сравнивает с 10, и выводит результат на экран.

Листинг 4

Программа “Сравнение”

```
1 #include<stdio.h>
2 main()
3 {
4 int a;
5 printf(“Vvedite 4islo \n”);
6 scanf(“%d”,&a);
7 if(a>10)
8 printf(“%d > 10”,a);
9 else
10 printf(“%d<=10”,a);
11 }
```

Задания:

1. Пользователь вводит число. Компьютер сравнивает с 20 и выводит результат.

В операторе if-else есть условие, правила составления условий рассказано в алгебре логике. Рассмотрим логическое или и логическое и в С:

&& – логическое И,

|| – логическое ИЛИ.

Таблица истинности для них, смотри Таблица 1-Таблица 2.

То есть ветвь if(условие) оператор1 выполняется , если условие равно 1.

Например,

```
if((a>10)&&(a<20))
оператор1
```

Если a>10 – истинно, в Таблице 1, это 1, если a>10 – ложь, это 0.

Таблица 1

a	b	a&&b
0	0	0
0	1	0
1	0	0
1	1	1

Таблица 2

a	b	a||b
0	0	0
0	1	1
1	0	1
1	1	1

Оператор switch, цикл while

Один из требований к программе, это удобство использования пользователем, поэтому напишем программу с меню, см. Листинг 5.

Строка 15, оператор break, он прерывает цикл и т.п.

Листинг 5

Программа с меню

```
1 #include<stdio.h>
2 main()
3 {
4 int a, d;
5 do
6 {
7      printf("      MENU      \n1. Kvadrat chisla\n2. Kub chisla\n3. Vuhod\nVuberite punkt menu\n");
8     scanf("%d",&d);
9     switch(d)
10      {
11       case 1:
12      printf("Vvedite chislo\n");
13      scanf("%d",&a);
14      printf("Kvadrat chisla raven %d\n",a*a);
15      break;
16 case 2:
17 printf("Vvedite chislo\n");
18 scanf("%d",&a);
19 printf("Kub chisla raven %d\n",a*a*a);
20 break;
21 case 3:
22      break;
23 default:
24 printf("Nevernui vvod!\n\n");
25      }
26 }
27 while(d!=3);
28 }
```

Оператор switch используется для выбора одного из вариантов, указанных в case. Также для написания программы мы использовали цикл do-while. Цикл проверяет условие в конце.

В default мы указали случай, в котором пользователь вводит цифру, не являющуюся номером пункта меню. А что будет если пользователь введет букву. Возможно зацикливание и т.п. Чтобы этого не было. Нам необходимо предусмотреть ввод не цифр, а букв.

Листинг 6

Программа с меню

```
1 #include<stdio.h>
2 main()
3 {
4 int a;
5 char d;
6 do
7 {
8       printf("       MENU       \n1. Kvadrat chisla\n2. Kub chisla\n3. Vuhod\nVuberite punkt menu\n");
9      scanf("%s",&d);
10      switch(d)
11       {
12        case ‘1’:
13       printf("Vvedite chislo\n");
14       scanf("%d",&a);
15       printf("Kvadrat chisla raven %d\n",a*a);
16       break;
17 case ‘2’:
18 printf("Vvedite chislo\n");
19 scanf("%d",&a);
20 printf("Kub chisla raven %d\n",a*a*a);
21 break;
22 case ‘3’:
23       break;
24 default:
25 printf("Nevernui vvod!\n\n");
26       }
27 }
28 while(d!=3);
```

Задания:

Спроектировать меню. Написать программу.

Оператор for

Оператор for – это оператор цикла, обобщение оператора while [4]. Оператор for:

```
for (int i=0; i<n; i++)
{
….
}
```

int i=0 – инициализация,

i<n – проверка условия,

i++ – модификация.

Каждый раз выполняется тело цикла, в конце идет модификация (в данном случае i++), поэтому цикл будет выполняться столько раз, сколько указано в условиях, если условие верно, то выполняется тело еще раз.

Задание: вычислить значение функции y=x*x-4*x при x от 100 до 200 включительно, начиная от 100 с шагом 20. Решение задачи в Листинге 7, результат решения Листинга 7 на Рисунке 1.

Листинг 7

```
#include<iostream>
using namespace std;
int main()
{
int y;
for(int x=100;x<=200;x=x+20)
{
      y=x*x-4*x;
      cout<<"y("<<x<<")="<<y<<endl;
}
}
```

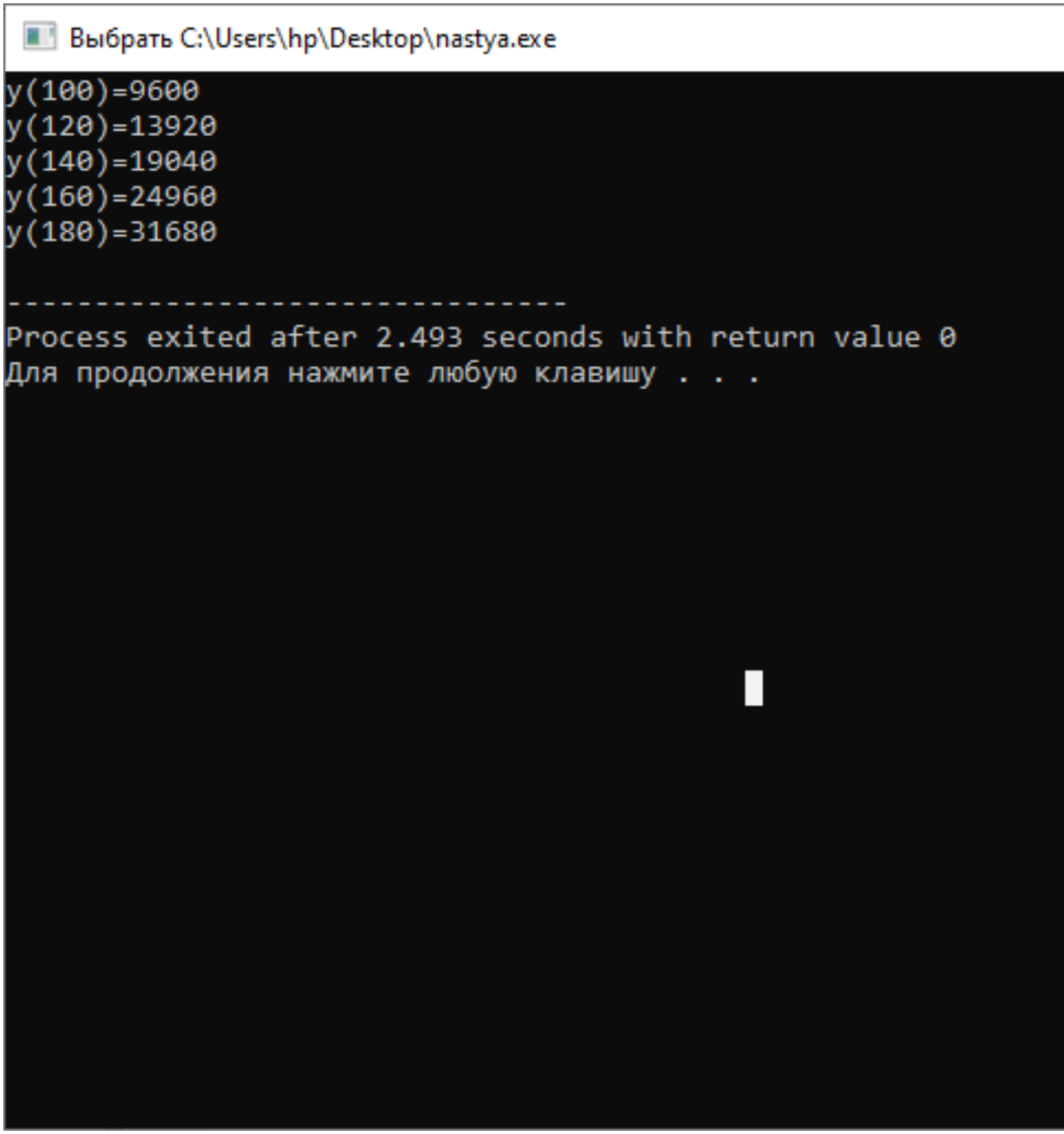

Рисунок 1

Типы данных

Типы данных:

char хранит символ (один байт),

int – целочисленные значения,

float – вещественные значения с одинарной точностью,

double – вещественное значения с двойной точностью и т.д.

Модификаторы:

short – короткое целое,

long – длинное целое.

Диапазон значений зависит от аппаратной и системной платформы.

Например, int будет или 16, или 32. Узнать это возможно, используя операцию sizeof(), она возвращает размер в байтах, смотри Листинг 8, Рисунок 2.

Листинг 8

```
#include<iostream>
using namespace std;
int main()
{
cout<<sizeof(int);
}
```

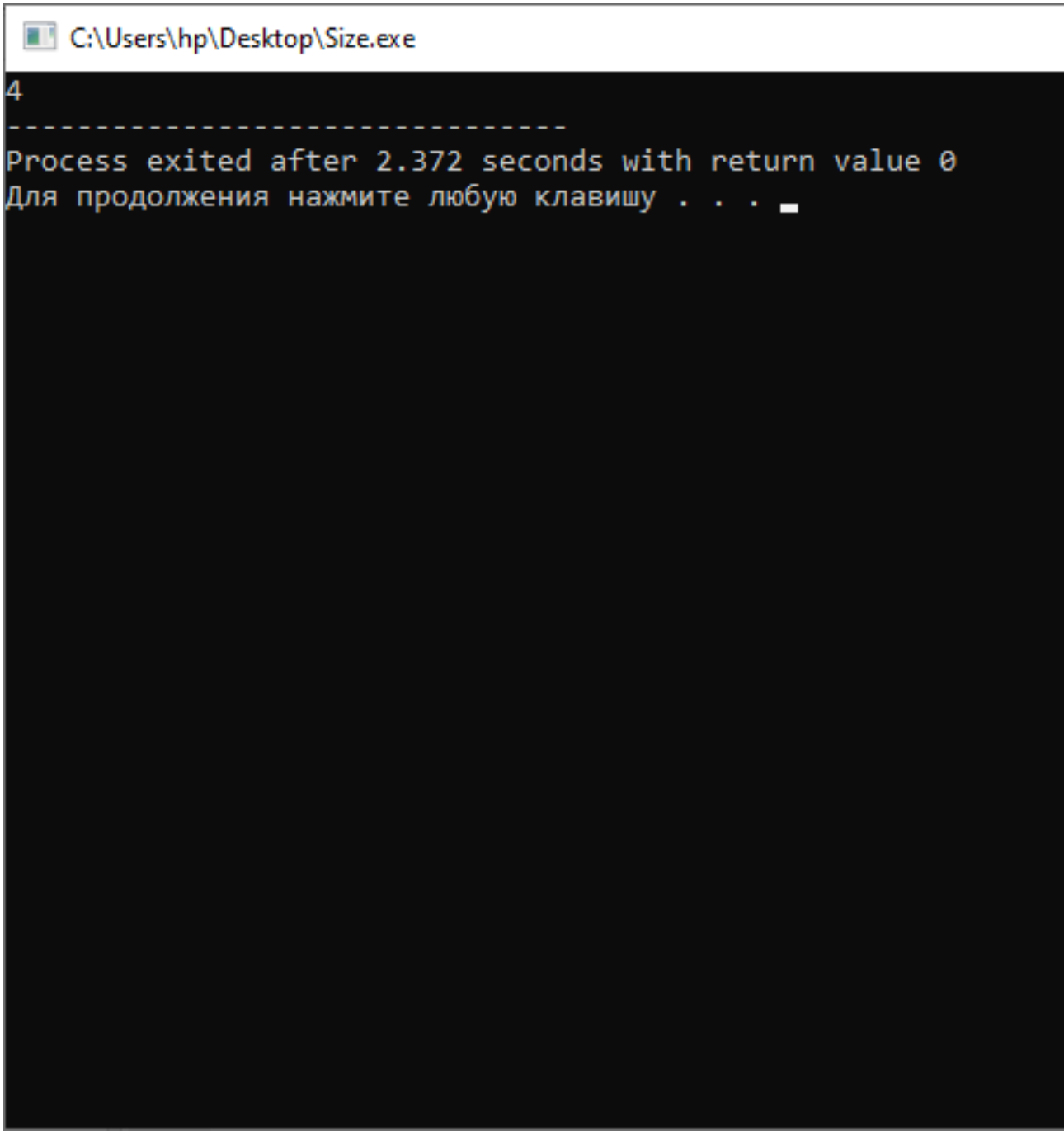

Рисунок 2

Время жизни и область видимости переменной

Программный блок – это часть программы между фигурными скобками.

Локальные переменные живут только во время программного блока. Локальные переменные объявлены внутри блока.

Глобальная переменная живет на протяжении всей жизни программы. Глобальная переменная объявляется вне блоков программы.

Пример в Листинге 9. Если убрать скобки в строках 8 и 9, компилятор выдаст ошибку.

Листинг 9

```
1#include<iostream>
2using namespace std;
3
4 int main()
5 {
6     int i=10;
7     cout<<i;
8     {
9          int i=11;
10 cout<<endl<<i;
11     }
12}
```

Преобразование типов

В зависимости от версии компилятора, округление будет либо в большую, либо в меньшую сторону. Смотри Листинг 10, Рисунок 3.

Листинг 10

```
#include<iostream>
using namespace std;
int main()
{
int y;
y=(int)(3/10);
cout<<y<<endl;
y=(int)(10/3);
cout<<y;
}
```

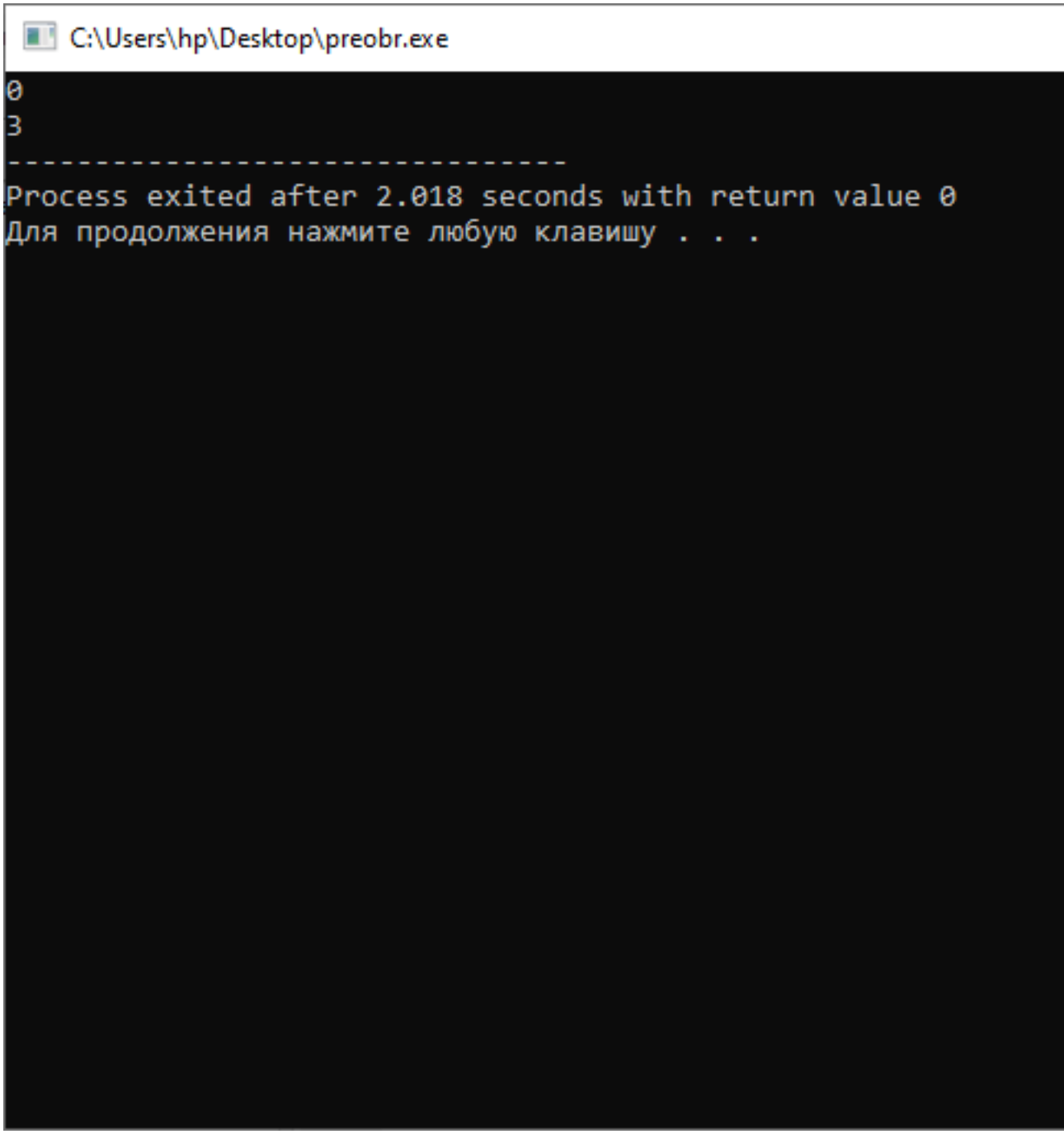

Рисунок 3

Поразрядные операции

Напишите программу, которая переводит введенное положительное число в двоичное систему счисления и выводит на экран результат перевода.

Пример решения задания в Листинге 11.

Листинг 11

```
#include <iostream>
using namespace std;
void dv(int a)
{
int b[100];
int i=0;
while(a>1)
```

```
{
b[i]=a%2;
    a=(a-a%2)/2;
    i++;
}
b[i]=a;
for(int j=i;j>=0;j–)
cout<<b[j];
}
int main()
{
int a;
cin>>a;
dv(a);
return 0;
}
```

Ниже даны тесты для проверки задач программы.

Тест 1

a=10

Результат

1010

Тест 2

a=2

Результат

10

Тест 3

a=8

Результат

1000

Поразрядные операции применимы только к целочисленным аргументам (char, short, int и long).

& – поразрядное И

| – поразрядное включающее ИЛИ

^ – поразрядное исключающее ИЛИ

<< – сдвиг влево

>> – сдвиг вправо

~ – одноместное поразрядное дополнение до единицы

В побитовых операциях работа идет над каждым битом.

Поразрядное включающее ИЛИ

Пример:

8 | 10 = 10

8 – это 1000 в двоичной системе счисления, 10 – это 1010 в двоичной системе счисления. С каждым битом числа выполняется операции логическое ИЛИ, и вместо этого бита ставится результат этой операции, смотри Рисунок 4.

1-е число	1	0	1	0
2-е число	1	0	0	0
	1 | 1	0 | 0	1|0	0|0
Результат	1	0	1	0

Рисунок 4

Поразрядное И

Пример:

8 & 10 = 8

8 – это 1000 в двоичной системе счисления, 10 – это 1010 в двоичной системе счисления. С каждым битом числа выполняется операции логическое И, и вместо этого бита ставится результат этой операции.

Пример программы в Листинге 12.

Листинг 12

```
#include <iostream>
using namespace std;
void dv(int a)
{
int b[100];
int i=0;
while(a>1)
{
b[i]=a%2;
    a=(a-a%2)/2;
    i++;
}
b[i]=a;
for(int j=i;j>=0;j–)
cout<<b[j];
}
int main()
{
int a,b;
int c;
cin>>a>>b;
c=a|b;
cout<<endl;
dv(a);
cout<<" | ";
dv(b);
cout<<" = ";
dv(c);
c=a&b;
cout<<endl;
dv(a);
cout<<" & ";
dv(b);
cout<<" = ";
dv(c);
return 0;
}
```

Ниже даны тесты для проверки задач программы.

Тест 1

a=10 b=8

Результат

1010 | 1000 =1010
1010 & 1000 =1000
Тест 2
a=11 b=3
Результат
1011 | 11 = 1011
1011 & 11 = 11

Сдвиг влево

Пример:
10 << 2 = 1000

Двоичная запись числа передвинется на 2 знака влево, на их место проставятся 0. Необходимо быть внимательными, так как в типе int и т.д. хранится ограниченное количество бит.

Сдвиг вправо

Пример:
100 >> 2 = 1

Двоичная запись числа передвинется на 2 знака вправо. 2 бита исчезнут.

Одноместное поразрядное дополнение до единицы

С каждым битом выполняется инверсия.

Пример:
x = ~8;

8 – это 1000 в двоичной системе счисления, после инверсии с каждым битом: 1 меняется на 0, 0 на 1. При хранении числа, один бит отвечает за знак, поэтому знак числа тоже меняется. ~x=|~x|-1. Результат: ~8=-9.

Комментарии

Для красивого стиля и правил оформления кода, необходимо, чтобы в коде все функции, блоки и т.д. были расшифрованы. Комментарии бывают /*….*/ (все, что между косыми чертами и звездочкой есть комментарий), // (все, что после // и на одной строке есть комментарий), смотри Листинг 13.

Листинг 13

```
/*демонстративная программа*/
#include<iostream>
using namespace std;
int main()
{
cout<<"Hello!"; //Вывод «Hello!» в консоль
}
```

Строки

Во второй программе мы уже использовали строки. Специальные функции для работы со строками определены в библиотечном файле <string.h>.

Некоторые функции для работы со строками, представлены ниже.

char* strcpy(str1,str2) – копирует строку str2 в строку str1 с ‘\0’, возвращает str1.
char* strcat(str1,str2) – присоединяет str2 в конец строки str1, возвращает str1.

Листинг 14

Работа со строками

```
#include<stdio.h>
#include<string.h>
int main()
{
char str1[100];
```

```
char str2[100];
printf("Vvedite stroky: \n");
scanf("%s",str1);
printf("Vvedenai stroka:\n%s \n",str1);
printf("Vvedite stroky: \n");
scanf("%s",str2);
printf("Vvedenai stroka:\n%s\n",str2);
strcat(str1,str2);
printf("Vvedenai stroka:\n%s\n",str1);
}
```

Задания:

Написать программу, в которой пользователь дописывает фразу, которую вывел компьютер, результат вывести на экран.

Закрепление материала

Операции

Присваивать значение переменной

a=10;

Вычислить значение выражения a^3+a^2-10.

Листинг 15

```
#include “stdio.h”
int main()
{
int a,s;
printf(“Vvedite zna4enie a\n”);
scanf(“%d”,&a);
s=a*a*a+a*a-10;
printf(“Rezultat: %d”,s);
}
```

Операция инкрементирования и декрементирования

++ – операция увеличения на 1,

– – – операция уменьшения на 1.

Операции ++ и – – бывают постфиксные и префиксные.

Пример:

```
N++;
++N;
–-N;
N–;
```

Разница в постфиксной и префиксной форме в том, что ++N – прибавление 1 до того, как переменная используется, N++ после того. Аналогично, с операцией –. Смотри Листинг 16, результат на Рисунке 5.

Листинг 16

```
#include<iostream>
using namespace std;
int main()
{
int y=3;
cout<<y++;
y=3;
cout<<endl<<++y;
```

```
}
```

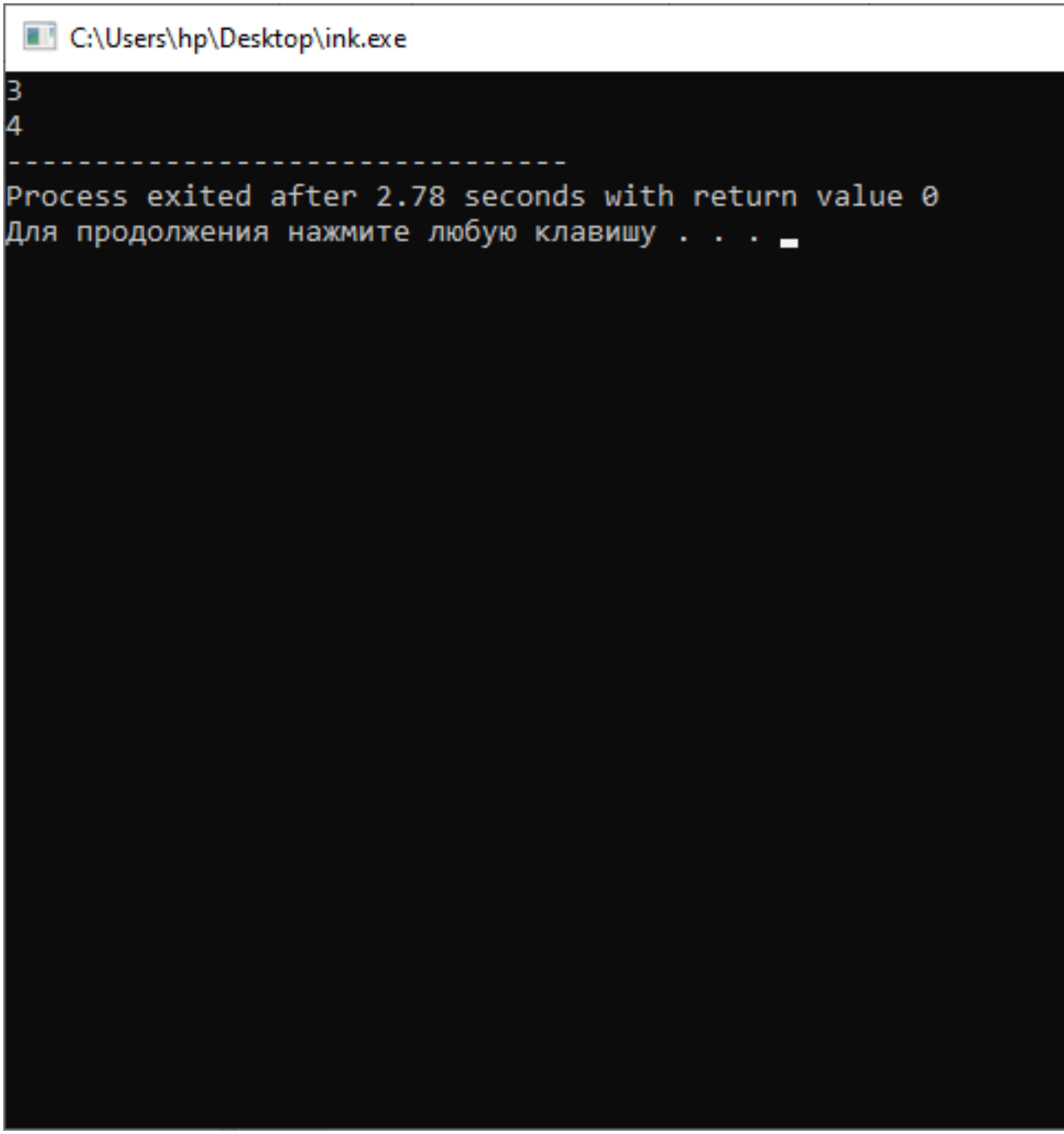

Рисунок 5

Приоритет операций и порядок выполнения

В любой операции важен приоритет, как в вычислительном примере порядок действий. Также порядок действий зависит от аппаратно-системной архитектуры, поэтому нужно быть аккуратными.

В Таблице 3 представлен приоритет с ассоциированием слева направо для ANSI C.

Таблица 3

Операция
() [] -> .
! ~ ++ -- + - * & (тип) sizeof
* / %
+ -
<< >>
< <= > >=
== !=
&
^
|
&&
||
?:
= += -= *= /= %= &= ^= |= <<= >>=
,

Работа с файлами

С писался для написания Unix, операционной системы. Все устройство Unix – это потоки. Также есть понятие файла. Файл – именованный памяти компьютера. «Поток» – это абстракция, все программирование – это абстракции.

Рассмотрим Листинг 17. 4 строка – это файловый указатель. Об указателях в части 3.

Строка 5.

```
fopen (“text”, “w+”);
 fopen (                “text”,                “w+”);
 1                      2                      3
```

1 – функция для открытия файла

2 – название файла

3 – режим доступа

Строка 6.

fwrite(“ura”,1,sizeof(char)*u,F);

```
fwrite(      “ura”,      1,      sizeof(char)*u,      F);
  1            2          3            4              5
```

1 – функция для записи в файл

2 – что записываем, строку

3 – сколько таких строк

4 – размер, функция sizeof() – вычисляет размер типа

5 – файловый указатель

После запуска программы, на компьютере в папке с программой будет текстовый файл «text».

Режимы доступа для функции fopen() приведены в Таблице 4.

Таблица 4

Режим	Описание
r	Режим открытия файла для чтения
w	Режим, при котором переписываем или создаем новый файл для записи
a	Дополняем или создаем файл для записи
r+	Открываем файл для чтения и записи
w+	Создаем новый файл для чтения и записи
a+	Дополняем или создаем файл для чтения и записи

Листинг 17

Ввод в файл. Способ первый

```
1 #include <stdio.h>
2 main()
3 {
4 File *F;
5 F=fopen(“text”, “w+”);
```

```
6 fwrite(“ura”,1,sizeof(char)*u,F);
7 fclose(F);
8 }
```

Рассмотрим второй способ записи в файл, Листинг 18.

Листинг 18

Ввод в файл. Способ второй

```
1 #include <stdio.h>
2 #include <string.h>
3 main()
4 {
5 File *F;
6 char text[100];
7 printf(«Vvedite text:/n»);
8 scanf(“%s”,&text);
9 F=fopen(“text”, “w+”);
10 fwrite(text,1,sizeof(char)*strlen(text), F);
11 fclose(F);
12 }
```

fwrite(text,1,sizeof(char)*strlen(text),F)

```
fwrite(text,1,sizeof(char)*strlen(text),F)
  fwrite(      text,      1,      sizeof(char)*strlen(text),      F);
    1          2          3                  4                   5
```

1 – функция для записи в файл

2 – что записываем, массив символов

3 – сколько таких строк

4 – размер, функция sizeof() – вычисляет размер типа, функция strlen() – вычисляет длину заполненного массива text.

5– файловый указатель

fclose(F);

```
fclose(F);
 fclose(                          F);
 1                                2
```

1 – функция для закрытия файла

2 – файловый указатель

Задания:

Записать в файл строку.

Прочитать из файла текст.

Структуры

«Структура – это совокупность нескольких переменных, часто различных типов, сгруппированных под единым именем для удобства обращения» [4].

Методов в структурах нет в стандарте ANSI C. Я бы не рекомендовала смешивать методы C, C++ и следить за версиями компилятора для красоты стиля программирования.

Несколько структур с одним набором данных.

struct {…} x,y,z;

Описание структуры.

```
struct point
{
int x;
```

```
int y;
};
```

Объявление структур: через точку, в начале название структуры, в конце имя переменной point.x;

Листинг 19

Создание структуры

```
1 #include<stdio.h>
2 struct point
3 {
4 int x;
5 int y;
6 };
7 int main ()
8 {
9 printf(“Введите координаты точки /n Введите абциссу точки”);
10 scanf(“%d”,&point.x);
11 printf(«Введите ординату точки»);
12 scanf(“%d”,&point.y);
13 printf(“/n (%d,%d)”,point.x,point.y);
14 }
```

Массивы структур

```
struct key
{
char* word;
int count;
}keytab[NKEYS];
```

Листинг 20

Работа со структурой

```
1 #include <stdio.h>
2 struct zapisi
3 {
4 char text[100];
5 char data[11];
6 };
7 main()
8{
9 struct zapisi x;
10 int d;
11 printf(«Vvedite datu, v formate dd.mm.yyyy: \n»);
12 scanf(“%s”,x.data);
13 printf(“Vvedite poslanie: \n”);
```

Конец ознакомительного фрагмента.