ЛАБОРАТОРНАЯ РАБОТА № 3

ВВОД-ВЫВОД ДАННЫХ

Цель работы: освоение простейшей структуры программы, получение навыков в организации ввода/вывода значений стандартных типов данных, получение практических навыков работы в диалоговом режиме.

Краткие теоретические сведения

Приступая к выполнению данной работы, следует вспомнить, что :

• каждая переменная программы должна быть объявлена;

• объявления переменных обычно помещают в начале функции, сразу за заголовком;

• инструкция объявления переменной выглядит так:

тип имя_переменной;

• инструкцию объявления переменной можно использовать для инициализации переменной. В этом случае объявление переменной записывается следующим образом:

тип имя_переменной = начальное_значение;

• инструкция присваивания предназначена для изменения значений переменных, в том числе и для вычислений “по формуле”.

Элементы языка

Символы – это основные знаки, с помощью которых пишутся все тексты программы:

– *прописные и строчные латинские буквы и знак подчеркивания;*

– *арабские цифры от 0 до 9;*

– *специальные знаки:*

{ } , | [ ] ( ) + – / % * . \ ‘ : ; & ? < > = ! # ^ “

– *пробельные символы* (пробел, символы табуляции, символы перехода на новую строку).

Лексема образуется из символов и имеет самостоятельный смысл:

– *Идентификаторы* – имена объектов программ. *Идентификатор* – это последовательность латинских букв, цифр и знака подчеркивания. Первым символом должна быть буква или знак подчеркивания (но не цифра). Пробелы и ключевые слова в идентификаторах не допускаются. Прописные и строчные буквы различаются.

– *Ключевые слова* – это зарезервированные идентификаторы, которые имеют специальное значение для компилятора.

– *Знаки операций* – один или более символов, определяющих действия над операндами. Операции бывают унарные, бинарные и тернарные. Все знаки операций кроме [], () и ?: являются отдельными лексемами.

– *Константы* – это неизменные величины. Константы бывают строковые (“Hello\t World!\n”), целые (123, 020, 0xA), вещественные (5.7, .45, 0.2E6) и символьные (‘\n’, ‘/0’, ‘\x07’).

– *Разделители* – (скобки, точка, запятая, пробельные символы).

Выражение – это правило вычисления некоторого действия. Оно состоит из операндов, знаков операций и скобок, которые используются для вычисления некоторого значения. *Операнд* – это выражение, константа или переменная.

Оператор – это задание законченного описания действия. Выполнение оператора – это вычисление данного выражения.

Составной оператор – это последовательность операторов, заключенная в фигурные скобки.

```
{  i++;
   sum+=i;
}
```

Блок – это последовательность операторов, заключенная в фигурные скобки, а также наличие определений переменных.

```
{  int i=0, sum=0;
   i++;
   sum+=i;
}
```

Типы данных

Каждый тип данных имеет определенный размер, т.е. сколько байтов выделяется в оперативной для записи переменной данного типа, и диапазон значений.

В языке Си применяются данные двух категорий: ***простые*** (скалярные) и ***сложные*** (составные) типы данных.

К ***простым типам данных*** относятся символы, указатели, перечисления, целые, вещественные.

К ***сложным типам данных*** относятся массивы, структуры, объединения, битовые поля.

Целый тип данных

Данные целого типа могут быть короткими – ***short***, длинными – ***long*** и беззнаковыми – ***unsigned***.

Таблица 2. Целые типы данных

Тип	**Размер, в байтах**	**Специ-фикация**	**Диапазон значений**
1	**2**	**3**	**4**
short int	2	%hd	-32768..32767
int	2	%d	-32768..32767
	4		-2 147 483 648..2 147 483 647
long int	4	%ld	-2 147 483 648..2 147 483 647
unsigned int	2	%d	0..65 536
	4		0..4 294 967 295

Вещественный тип данных

Внутренне представление вещественного числа состоит из 2 частей: *мантиссы* и *порядка*. *Мантисса* – это численное значение со знаком, *порядок* – это целое со знаком, определяющее значимость мантиссы.

Таблица 3. Вещественные типы данных

Тип	Размер, в байтах	Спецификация	Диапазон значений
float	4	%f	1.17E-38..3.37E+38
double	8	%lf	2.23E-308..1.67E+308

Символьный тип

Значениями символьного типа являются элементы конечного упорядоченного множества символов. Каждому символу ставится в соответствие число, которое называется кодом символа. Символы с кодами от 0 до 31 относятся к служебным и имеют самостоятельное значение только в операторах ввода-вывода.

Код символа является его номером во множестве ASCII кодов.

Таблица 4. Символьный тип данных

Тип	Размер, в байтах	Спецификация	Диапазон значений
char	1	%c	-128..127
unsigned char	1	%c	0..255

Использование функций printf() и scanf() для форматного вывода и ввода информации

Важная составляющая часть решения любой задачи – представление результатов.

При запуске программы к ней автоматически присоединяются три потока. Стандартный поток ввода **stdin**

обычно присоединяется к клавиатуре, а стандартный поток вывода **stdout** – к устройству вывода информации на экран монитора. Третий поток – стандартный поток ошибок **stderr** – также присоединяется к экрану. В него выводятся сообщения об ошибках.

Функции **printf()** и **scanf() (**заголовочный файл stdio.h) позволяют пользователю общаться с программой. Они называются функциями вывода/ввода (output/input).

Форматированный вывод на экран осуществляется с помощью функции **printf(),** имеющей следующую структуру:

printf(”форматная строка”, список_вывода);

форматная строка ограничена двойными кавычками (т.е. является текстовой константой) и может включать в себя *произвольный текст, управляющие символы и спецификаторы формата*. Список аргументов может отсутствовать или же состоять из выражений, значения которых выводятся на экран (в частном случае из констант и переменных).

Признаком *управляющего символа* является значок \. Ниже приводится их список:

\n – перевод строки;
\t – горизонтальная табуляция;
\r – возврат курсора к началу новой строки;
\b – возврат на один символ (одну позицию);
\f – перевод (прогон) страницы;
\v – вертикальная табуляция.

Спецификатор формата определяет форму внешнего представления выводимой величины. Вот некоторые спецификаторы формата:

%c – символ;
%s – строка;
%d (%i) – целое десятичное число (тип int);
%u – целое десятичное число без знака (тип unsigned);
%f – вещественные числа в форме с фиксированной точкой;
%e – вещественные числа в форме с плавающей точкой (с мантиссой и порядком);

%ld – вывод long int;
%lf – вывод double.

К спецификатору формата могут быть добавлены числовые параметры: ширина поля и точность. Ширина – это число позиций, отводимых на экране под величину, а точность – число позиций под дробную часть (после точки). Параметры записываются между значком % и символом формата и отделяются друг от друга точкой.

Пример:

```
#include <stdio.h>
void main()
{
  int a;
  char b;
  float x,y;
  a=2; x=3.14; y=0.0317; b=’A’;
  printf("a=%2d\t x=%4.2f\t y=%6.2e\t b=%c",a,x,y,b);
}
```

На экране дисплея высветится:

a= 2 x=3.14 y=3.17e-2 b=A

Здесь трижды используемый управляющий символ табуляции \t отделил друг от друга выводимые значения. Из этого примера видно, что соответствие между спецификаторами формата и элементами списка аргументов устанавливается в порядке их записи слева направо.

Если в пределы указанной ширины поля выводимое значение не помещается, то этот параметр игнорируется и величина будет выводиться полностью.

Форматированный ввод с клавиатуры осуществляется с помощью оператора вызова функции scanf(), имеющего следующую структуру:

scanf(форматная_строка, список_аргументов);

Данная функция осуществляет чтение символов, вводимых с клавиатуры, и преобразование их во внутреннее представление в соответствии с типом величин. В функции scanf() форматная строка и список аргументов присутствуют обязательно.

Символьную последовательность, вводимую с клавиатуры и воспринимаемую функцией scanf(), принято называть входным потоком. Функция scanf() разделяет этот поток на отдельные вводимые величины, интерпретирует их в соответствии с указанным типом и форматом и присваивает переменным, содержащимся в списке аргументов.

Список аргументов – это перечень вводимых переменных, причем перед именем каждой переменной ставится значок &. Это знак операции «взятие адреса переменной».

Форматная строка заключается в кавычки (как и для printf) и состоит из списка спецификаций. Каждая спецификация начинается со знака %, после которого могут следовать

*ширина_поля модификатор спецификатор

Из них обязательным элементом является лишь спецификатор. Для ввода числовых данных используются такие же спецификаторы, как и для вывода.

Звездочка в спецификации позволяет пропустить во входном потоке определенное количество символов. Ширина поля — целое положительное число, позволяющее определить число символов из входного потока, принадлежащих значению соответствующей вводимой переменной.

Функции для ввода и вывода символов

Функция ***int getchar(void)***; //заголовочный файл stdio.h

Возвращает значение считанного символа (если он есть) из потока stdin или константу EOF, если обнаружен конец файла или ошибка. Как правило, EOF= –1. После ввода символа пользователь должен нажать клавишу <Enter>. Функция не имеет параметров.

Функция ***int getch(void)***; (заголовочный файл conio.h) возвращает код символа нажатой клавиши (сам символ на экран не выводится). Если нажата служебная клавиша, то функция

возвращает 0. В этом случае, для того, чтобы определить, какая служебная клавиша нажата, нужно обратиться к функции getch еще раз.

Пример:

```
#include<stdio.h>
#include<conio.h>
void main()
{
  char s;
  clrscr();
  printf("vvedite simvol s=");
  s=getchar();
  printf("\nsimvol s=%c",s);
  getch();
}
```

Функции:

int putchar(int c); //заголовочный файл stdio.h

int putch(int c); //заголовочный файл conio.h

выводят на экран (в поток stdout) символ. В случае успеха возвращает с, а иначе – возвращает EOF. Функции имеют параметр **int c –** символ, записываемый в stdout.

Пример:

```
#include<stdio.h>
#include<conio.h>
void main()
{
  char c;
  clrscr();
  printf("vvedite simvol c=");
  c=getchar();
  putchar(c);
  getch();
 }
```

Неформатный ввод-вывод строк

Функция ***char *gets(char *s);*** //заголовочный файл stdio.h

Читает из стандартного входного потока символы, включая пробелы и табуляции, до тех пор, пока не встретится символ новой строки (\n), который заменяется нулевым символом (\0). Функции gets() нельзя указать максимальное число читаемых символов. Результирующая строка завершается нулевым символом (\0).

Функция ***int puts(const char *s);*** /*заголовочный файл stdio.h */

Эта функция записывает в стандартный вывод заданную строку. Параметр const char *s является указателем на завершающуюся нулем строку. Строка s должна иметь ограничитель '\0'. Функция записывает в поток всю строку, за исключением символа '\0', и добавляет в конец символ '\n'.

Пример:

```
#include <stdio.h>
void main()
{
  char buf[128];
  puts ("Введите строку: ");
  gets(buf);
  printf ("\n Вы ввели строку: %s \ n", buf);
}
```

Функция ***cputs(const char *s);*** (заголовочный файл conio.h) выводит на экран строку. Цвет выводимых символов можно задать при помощи функции *textcolor*, цвет фона – при помощи функции *textbackground*. Для перехода к началу следующей строки вместо \n следует использовать символы \n\r, иначе курсор лишь переводится на новую строку, но не возвращается к левой границе окна. То же самое относится и к функции *cprintf*.

Функция ***cprintf***, как и функция *printf*, используется для вывода на экран сообщений и значений переменных, но при этом

имеется возможность задать цвет выводимых символов и цвет фона. Заголовочный файл conio.h.

Функция ***void textcolor(int цвет);*** (заголовочный файл conio.h) задает цвет для выводимого функциями *cputs* и *cprintf* текста. В качестве параметра *цвет* обычно используют одну из перечисленных ниже именованных констант.

Таблица 5

Цвет	**Константа**	**Значение константы**
1	**2**	**3**
Черный	BLACK	0
Синий	BLUE	1
Зеленый	GREEN	2
Бирюзовый	CYAN	3
Красный	RED	4
Сиреневый	MAGENTA	5
Коричневый	BROWN	6
Светло-серый	LIGHTGRAY	7
Серый	DARKGRAY	8
Голубой	LIGHTBLUE	9
Светло-зеленый	LIGHTGREEN	10
Светло-бирюзовый	LIGHTCYAN	11
Алый	LIGHTRED	12
Светло-сиреневый	LIGHTMAGENTA	13
Желтый	YELLOW	14
Белый (яркий)	WHITE	15

Функция ***void textbackground(int цвет);*** (заголовочный файл conio.h) задает цвет фона, на котором появляется текст, выводимый функциями *cputs* и *cprintf*. В качестве параметра *цвет* обычно используют одну из перечисленных ниже именованных констант.

Таблица 6

Цвет	**Константа**	**Значение константы**
Черный	BLACK	0
Синий	BLUE	1
Зеленый	GREEN	2
Бирюзовый	CYAN	3
Красный	RED	4
Сиреневый	MAGENTA	5
Коричневый	BROWN	6
Светло-серый	LIGHTGRAY	7

Функция ***void gotoxy(int x, int y);*** (заголовочный файл conio.h) переводит курсор в позицию с указанными координатами. Координата *x* задает номер колонки, координата *y* – номер строки, на пересечении которых находится знакоместо, куда переводится курсор.

Функция ***void clrscr(void);*** (заголовочный файл conio.h) очищает экран и закрашивает его цветом, заданным функцией *textbackground*.

Функция ***void window(int x1, int y1, int x2, int y2);*** (заголовочный файл conio.h) определяет окно – область экрана. параметры *x1, y1* задают координаты левого верхнего угла окна относительно экрана, параметры *x2, y2* – правого нижнего.

Пример.

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

```
void main()
{
 char s;
 textbackground(BLUE);
 textcolor(LIGHTRED);
 clrscr();
 cprintf("vvedite s=\n\r");
 s=getchar();
 gotoxy(5,3);
 putch(s);
 getch();
}
```

Пример.

```
#include<conio.h>
void main()
{
 clrscr();
 window(30,10,60,11);
 textbackground(GREEN);
 textcolor(BLACK);
 cprintf("Hello, World!");
 getch();
}
```

Внимание!!!

Для работы с цветным текстом используйте следующую библиотеку и функцию:

```
#include <windows.h>
        void SetColor(int ForgC)
         {
         WORD wColor;
```

```
        HANDLE hStdOut =
GetStdHandle(STD_OUTPUT_HANDLE);
        CONSOLE_SCREEN_BUFFER_INFO csbi;

                //We use csbi for the wAttributes word.
        if(GetConsoleScreenBufferInfo(hStdOut, &csbi))
        {
                //Mask out all but the background attribute,
and add in the forgournd color
            wColor = (csbi.wAttributes & 0xF0) + (ForgC &
0x0F);
            SetConsoleTextAttribute(hStdOut, wColor);
        }
        return;
}
```

Выбор цвета происходит с использованием следующей функции: SetColor(12);

где цифра соответствует цвету текста, исходя из следующей таблицы:

Name	Value
Black	0
Blue	1
Green	2
Cyan	3
Red	4

Magenta | 5

Brown | 6

Light Gray | 7

Dark Gray | 8

Light Blue | 9

Light Green | 10

Light Cyan | 11

Light Red | 12

Light Magenta| 13

Yellow | 14

White | 15

Программа для вывода на экран слова «привет» ярко-красным цветом будет выглядеть следующим образом:

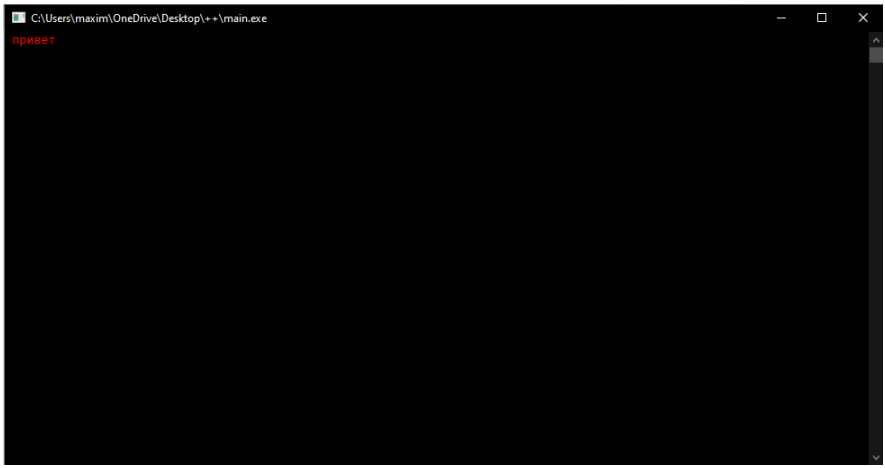
main.c

```c
#include <stdio.h>
#include <windows.h>
#include <locale.h>
int main(void)
{
  char *locale = setlocale(LC_ALL, "");
  SetColor(12);
  printf(" привет \n ");
  getch();
  return 0;
}

void SetColor(int ForgC)
 {
 WORD wColor;

  HANDLE hStdOut = GetStdHandle(STD_OUTPUT_HANDLE);
  CONSOLE_SCREEN_BUFFER_INFO csbi;

                       //We use csbi for the wAttributes word.
 if(GetConsoleScreenBufferInfo(hStdOut, &csbi))
 {
                //Mask out all but the background attribute, and add in the forgournd color
      wColor = (csbi.wAttributes & 0xF0) + (ForgC & 0x0F);
      SetConsoleTextAttribute(hStdOut, wColor);
 }
 return;
}
```



Порядок выполнения работы

1. Изучить теоретические сведения по теме «Ввод-вывод данных».
2. Ответить на контрольные вопросы.
3. Выполнить задание.

Контрольные вопросы

1. Как осуществляется форматный вывод информации?
2. С помощью каких функций можно выводить символьные данные?
3. В чем различие вывода строк функциями *printf* и *puts*?
4. Какие функции используются для ввода данных?
5. С помощью каких функций можно задавать цвет фона и цвет шрифта?

Задание для выполнения

Перед написанием программы в IDE Visual Studio вставьте следующий код #define _CRT_SECURE_NO_WARNINGS

Вариант 1

Задача 1

Написать программу, которая выводит на экран ваше имя, отчество и фамилию (каждую часть имени с новой строки).

Задача 2

Написать инструкцию вывода значений переменных a, b и c (типа float) с 3я цифрами в целой части и 2я — в дробной. Значения должны быть выведены в виде: a = значение b = значение c = значение.

Задача 3

Написать инструкцию, которая выводит в одной строке значения переменных a и b целого типа (int).

Задача 4

Написать программу, которая выводит на синем фоне серыми буквами четверостишие:

Все деревья облетели.

Зеленеют только ели.

Днём и ночью дождик льёт.

Грязь и лужи у ворот.

автор: В. Мирович

Задача 5

Написать инструкции, которые обеспечивают ввод значений дробных переменных a и b (тип float). Предполагается, что пользователь после набора каждого числа будет нажимать клавишу <Enter> (каждое число вводить в отдельной строке).

Задача 6

Объявить необходимые переменные и написать фрагмент программы вычисления объема куба, обеспечивающий ввод исходных данных.

Вариант 2

Задача 1

Написать программу, которая выводит на экран приведенное далее четверостишие. Между последней строкой стихотворения и именем автора должна быть пустая строка.

Осень яркими цветами,
Всё раскрасило в саду,
Очень скоро за грибами,
Я с сестрёнкою пойду!

автор: Г. Шмонов

Задача 2

Написать инструкцию вывода значений переменных c и k (типа float), которые содержат значения высоты и длины прямоугольника. Перед значением переменной должен быть пояснительный текст (высота=, ширина=), а после — единица измерения (см).

Задача 3

Написать инструкцию вывода значений целых переменных q, w и p. Значение каждой переменной должно быть выведено в отдельной строке.

Задача 4

Написать программу, которая выводит на экран фразу "Каждый охотник желает знать, где сидят фазаны", позволяющую запомнить порядок следования цветов радуги (первая буква слова кодирует цвет: каждый — красный, охотник — оранжевый, желает — желтый, знать — зеленый, где — голубой, сидят — синий, фазаны — фиолетовый). Каждое слово фразы должно быть выведено наиболее подходящим цветом.

Задача 5

Написать инструкцию, которая обеспечивает ввод значений переменных t и y (тип float). Предполагается, что пользователь будет набирать числа в одной строке.

Задача 6

Объявить необходимые переменные и написать инструкции ввода исходных данных для программы вычисления дохода по вкладу. Предполагается, что процентную ставку программа определяет на основе данных о сумме и сроке вклада.