Модуль 2.2. Ввод-вывод на консоль

Темы модуля

- Синтаксис функций ввода-вывода символов, строк
- Форматированный ввод-вывод на консоль

Планируемые результаты обучения

После изучения данного модуля Вы должны уметь:

на уровне знаний:

- воспроизводить алфавит и лексику языка
- воспроизводить типы данных языка программирования
- воспроизводить правила записи выражений и операций
- воспроизводить синтаксис простых операторов
- описывать структуру программы на языке С

на уровне понимания:

- объяснять применение типов данных

на уровне применения:

- использовать по назначению базовые типы данных языка программирования при объявлении переменных
- записывать в соответствии с правилами языка программирования выражения и операции
- записывать действия алгоритма на языке С в соответствии с синтаксическими правилами записи операторов

на уровне анализа:

- анализировать разработанную программу с целью выявления логических ошибок;

на уровне синтеза:

- использовать математические методы и вычислительные алгоритмы для решения практических задач
- проектировать структуру программы
- организовать работу в группе при совместном решении задачи
- проектировать тестирование программы
- защищать выполненную самостоятельную работу
- принимать верное решение при коллективном решении задачи

Ввод-вывод на консоль

- В языке С не определено никаких ключевых слов, с помощью которых можно выполнять ввод/вывод. Вместо них используются библиотечные функции.
- Система ввода/вывода языка С — это элегантная конструкция, которая обеспечивает гибкий и в то же время слаженный механизм передачи данных от одного устройства к другому. Эта система достаточно большая и состоит из нескольких различных функций
- Заголовочным файлом для функций ввода/вывода является <stdio.h>.
- Имеются как консольные, так и файловые функции ввода/вывода
- Консольные функции ввода/вывода выполняют ввод с клавиатуры и вывод на экран
- В действительности же эти функции работают со стандартным потоком ввода и стандартным потоком вывода

Функции ввода/вывода на консоль стандарта языка С

- В стандарте языка С не определены никакие функции, предназначенные для выполнения различных операций управления экраном (например, позиционирования курсора) или вывода на него графики
- В стандарте языка С не определены никакие функции, которые выполняют операции вывода в обычном или диалоговом окне, создаваемом в среде Windows
- Функции ввода/вывода на консоль выполняют всего лишь телетайпный вывод
- Однако в библиотеках большинства компиляторов имеются функции графики и управления экраном, предназначенные для той среды, в которой как раз и должны выполняться программы. И на языке С можно писать Windows-программы. Просто в С не определены функции, которые выполняли бы эти задачи напрямую.

Чтение и запись символов

getchar() - читает символ с клавиатуры. Она ожидает, пока не будет нажата клавиша, а затем возвращает значение этой клавиши. Кроме того, при нажатии клавиши на клавиатуре на экране дисплея автоматически отображается соответствующий символ

putchar() - отображает символ на экране в текущей позиции курсора

Прототипы функций getchar() и putchar():

```
int getchar(void);
int putchar(int c);
```

Как видно из прототипа, считается, что функция getchar() возвращает целый результат. Однако возвращаемое значение можно присвоить переменной типа char, что обычно и делается, так как символ содержится в младшем байте. В случае ошибки getchar() возвращает EOF. (Макрос EOF определяется в <stdio.h> и часто равен -1.)

Функция putchar(), то несмотря на то, что объявлена как принимающая целый параметр, обычно вызывается с символьным аргументом. На самом деле из ее аргумента на экран выводится только младший байт. Функция putchar() возвращает записанный символ или, в случае ошибки, EOF.

Пример использования `getchar()`

- В этой программе с клавиатуры вводятся символы, а затем они отображаются на другом регистре. То есть символы, вводимые на верхнем регистре, выводятся на нижнем, а вводимые на нижнем — выводятся на верхнем. Чтобы остановить программу, нужно ввести точку.

```
#include <stdio.h>
#include <ctype.h>
 int main(void) {
char ch;
printf("Введите какой-нибудь текст (для завершения
  работы введите точку).\n");
do {
ch = getchar();
if(islower(ch)) ch = toupper(ch);
else ch = tolower(ch); putchar(ch);
} while (ch != '.');
return 0;
}
```

Альтернативы `getchar()`

У двух из самых распространенных альтернативных функций

getch() и getche()

имеются следующие прототипы:

```
int getch(void);
int getche(void);
```

В библиотеках большинства компиляторов прототипы таких функций находятся в заголовочном файле <conio.h>.

В библиотеках некоторых компиляторов имена этих функций начинаются со знака подчеркивания (_)

Например, в Visual C++ компании Microsoft они называются

_getch()

_getche()

Форматный ввод / вывод на консоль

- Функции printf() и scanf() выполняют форматный ввод и вывод, то есть они могут читать и писать данные в разных форматах
- printf() – выводит данные на консоль
- scanf(), наоборот — считывает данные с клавиатуры

Обе функции могут работать с любым встроенным типом данных, а также с символьными строками, которые завершаются символом конца строки ('0').

Функция printf():

Вот прототип функции printf():

```
int printf(const char *управляющая_строка, ...);
```

Функция printf() возвращает число выведенных символов или отрицательное значение в случае ошибки.

Управляющая_строка состоит из элементов двух видов:

- символы, которые предстоит вывести на экран;
- *спецификаторы преобразования*, которые определяют способ вывода стоящих за ними аргументов. Каждый такой спецификатор начинается со знака процента, за которым следует код формата. Аргументов должно быть ровно столько, сколько и спецификаторов, причем спецификаторы преобразования и аргументы должны попарно соответствовать друг другу в направлении слева направо

Например, в результате такого вызова printf()

```
printf("Мне нравится язык %c %s", 'C', "и к тому же
                    очень сильно!");
```

Будет выведено

```
Мне нравится язык C и к тому же очень сильно!
```

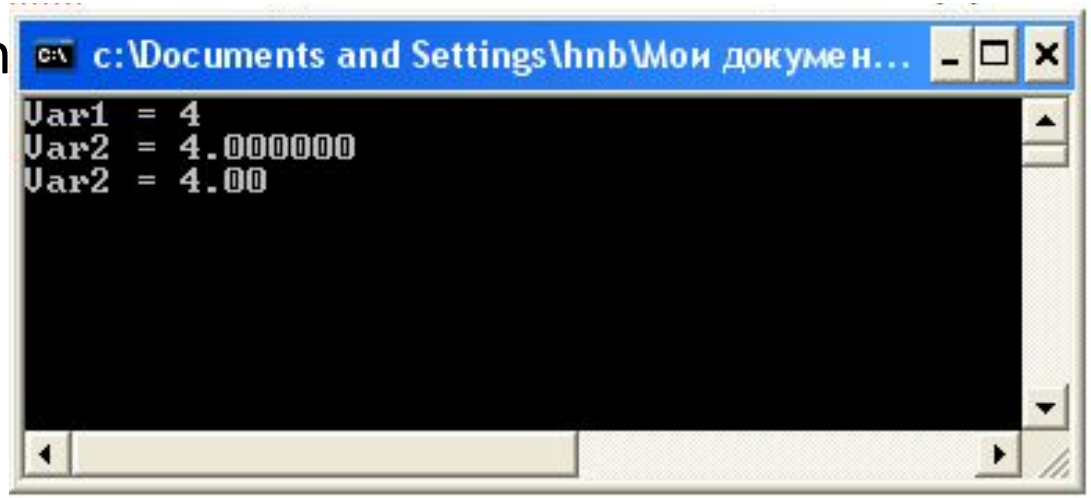
В этом примере первому спецификатору преобразования (%c), соответствует символ 'C', а второму (%s), — строка "и к тому же очень сильно!".

Спецификаторы преобразования для функции printf()

Код	*Формат*
%a	Шестнадцатеричное в виде *0xh.hhhhp+d* (только C99)
%A	Шестнадцатеричное в виде *0Xh.hhhhP+d* (только C99)
%c	Символ
%d	Десятичное целое со знаком
%i	Десятичное целое со знаком
%e	Экспоненциальное представление ('e' на нижнем регистре)
%E	Экспоненциальное представление ('E' на верхнем регистре)
%f	Десятичное с плавающей точкой
%g	В зависимости от того, какой вывод будет короче, используется %e или %f
%G	В зависимости от того, какой вывод будет короче, используется %E или %F
%o	Восьмеричное без знака
%s	Строка символов
%u	Десятичное целое без знака
%x	Шестнадцатеричное без знака (буквы на нижнем регистре)
%X	Шестнадцатеричное без знака (буквы на верхнем регистре)
%p	Выводит указатель
%n	Аргумент, соответствующий этому спецификатору, должен быть указателем на целочисленную переменную. Спецификатор позволяет сохранить в этой переменной количество записанных символов (записанных до того места, в котором находится код %n)
%%	Выводит знак %

Примеры

```
int Var1 = 4; //объявляем и инициализируем переменную
    Var1
printf("Var1 = %d\n", Var1);
float Var2 = 4;//объявляем и инициализируем переменную
    Var2
printf("Var2 = %f\n", Var2);
prin
```



Функция scanf()

scanf() - это программа ввода общего назначения, выполняющая ввод с консоли

Она может читать данные всех встроенных типов и автоматически преобразовывать числа в соответствующий внутренний формат, scanf() во многом выглядит как обратная к printf().

Прототип функции scanf():

```
int scanf(const char *управляющая_строка, ...);
```

Эта функция возвращает количество тех элементов данных, которым было успешно присвоено значение. В случае ошибки scanf() возвращает EOF, *управляющая_строка* определяет преобразование считываемых значений при записи их переменные, на которые указывают элементы списка аргументов

Управляющая строка

Управляющая строка состоит из символов трех видов:

- спецификатор преобразования - начинается со знака %, и сообщает функции scanf() тип считываемых данных
- разделитель - дает scanf() указание пропустить в потоке ввода один или несколько начальных разделителей. Разделителями являются пробелы, табуляции, вертикальные табуляции, подачи страниц и разделители строк
- символы, не являющиеся разделителями - если в управляющей строке находится символ, не являющийся разделителем, то функция scanf() прочитает символ из входного потока, проверит, совпадает ли прочитанный символ с указанным в управляющей строке, и в случае совпадения пропустит прочитанный символ

Функции scanf() необходимо передавать адреса переменных

Для всех переменных, которые должны получить значения с помощью scanf(), должны быть переданы адреса переменных

Для этого используется оператор *&*, это унарный оператор, возвращающий адрес операнда в памяти

Например, для считывания целого значения в переменную count можно использовать такой вызов функции scanf()

```
scanf("%d", &count);
```

Пример

```
int Var1; //объявляем и инициализируем переменную
    Var1
scanf("%d", &Var1);
printf("Var1 = %d\n", Var1);
```