Пример

```
y=3;
printf("\n Значение переменной y=%10.7f",x);
```

Результат работы программы

```
 Значение переменной y=3.0000000
```

В программе 10 — общее количество позиций под значение переменной; 7 — количество позиций после десятичной точки.

Функция форматированного ввода данных с клавиатуры scanf выполняет чтение кодов, вводимых с клавиатуры, преобразует их во внутренний формат и передаёт программе. При этом программист может повлиять на правила интерпретации входных кодов с помощью спецификаций форматной строки.

Общая форма записи функции *scanf*

```
scanf ("строка_форматов", адрес_объекта_1,
адрес_объекта_2, ..., адрес_объекта_n);
```

Строка форматов аналогична функции *printf*. Адрес объекта генерируется следующим образом: *&имя_объекта*. Строка форматов и список аргументов для функции обязательны.

Пример программы

```
scanf("%d", &m);
/* Ввести целое число и присвоить */
/* его значение переменной m.     */


scanf("%lf", &x1);
/* Ввести значение переменной x1, */
/* имеющей тип double.            */
```

§1.7. Операторы

Условный оператор if

Общая форма записи

```
if(< выражение>)
   <оператор 1>;
```

```
[else
   <оператор 2>;]
```

Если выражение истинно, то выполняется <*оператор* 1>, если выражение ложно, то выполняется <*оператор* 2> (при наличии опции *else*).

Оператор *if* может быть вложенным.

Пример

```
if (key == 1)
   printf("\n Выбран первый пункт");
else
   if (key == 2)
       printf("\n Выбран второй пункт");
   else
       printf("\n Первый и второй пункты не
                выбраны");
```

Возможно использование оператора *if* без опции *else*. При использовании обеих форм оператора *if* опция *else* связывается с последним оператором *if*.

Пример

```
if (key!= 1)
   if (key == 2)
       printf( "\n Выбран второй пункт");
   else
       printf("\n Первый и второй пункты не
       выбраны");
```

Если <*оператор1*> или <*оператор2*> должны состоять из нескольких операторов, то необходимо использовать составной оператор (блок).

Пример

```
if (key == 1)
{
   n=n+1;
   m=l+r;
}
else
{
   m=m-1;
```

```
    n=l-r;
}
```

Оператор ветвления switch

Оператор *if* позволяет осуществить выбор только между двумя вариантами. Для того, чтобы производить выбор одного из нескольких вариантов используется оператор *switch*.

Общая форма записи

```
switch( <целое выражение> )
{
    case <константное выражение1>: <оператор1>; break;

    case <константное выражение2>: <оператор2>; break;

                    . . .

    default:                       <оператор n+1>;
}
```

Оператор выполняется следующим образом:

1) вычисляется выражение в скобках оператора *switch*;

2) полученное значение сравнивается с метками (константными выражениеми) в опциях *case*;

3) сравнение производится до тех пор, пока не будет найдена метка, соответствующая данному значению, после этого выполнится оператор соответствующей ветви;

4) если соответствующая метка не найдена, то выполнится оператор в опции *default*.

Альтернатива *default* может отсутствовать, тогда не будет произведено никаких действий.

Опция *break* осуществляет выход из оператора *switch* и переход к следующему за ним оператору. При отсутствии опции *break* будут выполняться все операторы, начиная с помеченного данной меткой и кончая оператором в опции *default*.

Константые выражения (выражения, операнды которого константы) должны быть целого типа (включая *char*).

Пример. Разработать программу, определяющую день недели по его введенному номеру. Программа должна реагировать на неверно введенный номер дня недели.

```
main( )
{
   int i;

   printf("\nВведите номер дня недели: ");
   scanf("%u", &i);
   switch( i )
   {
      case 1: printf("\n Понедельник."); break;
      case 2: printf("\n Вторник.");      break;
                    . . .
      case 7: printf("\n Воскресенье.");  break;
      default:
              {
                   printf("\n Неверно введен ");
                   printf("номер дня недели.");
              }
   }
}
```

Операторы цикла

В языке Си реализованы три вида операторов цикла:
1) *while* — цикл с предусловием;
2) *do...while* — цикл с постусловием;
3) *for* — цикл с заданным числом повторений (цикл с предусловием).

Цикл while

Общая форма записи

```
while(<выражение>)
   <оператор> ;
```

Если выражение истинно (т. е. не равно нулю), то выполняется оператор или группа операторов, входящих в цикл *while*; затем выражение проверяется снова. Последовательность действий, состоящая из проверки и выполнения оператора, повторяется до тех пор, пока выражение не станет ложным (т. е. равным нулю). После этого происходит выход из цикла, и далее выполняется оператор, стоящий после оператора цикла. При построении цикла *while*, в него необходимо включить конструкции, изменяющие величину проверяемого выражения так, чтобы в конце концов оно стало ложным. Иначе выполнение цикла никогда не завершится.