

ЛЕКЦИИ 7 – 8

Оператор вызова функций.

Функции форматированного ввода и вывода.

Оператор присвоения. Математические и логические операции. Построение выражений.

Условная операция.

Оператор вызова функций

Оператор вызова функций просто определяет синтаксис описания вызова любой функции в тексте программы:

```
имя_функции(параметры);
```

Особенностью оператора вызова функций является то, что он может содержать в себе и сам может быть включен в другие операторы языка C.

Пример: вычисление квадратного корня куба вещественной переменной x :

```
sqrt(pow(x,3.0));
```

Функции форматированного ввода и вывода

Функции форматированного ввода и вывода описаны в библиотеке **stdio.h**:

- `scanf` – функция форматированного ввода;
- `printf` – функция форматированного вывода

Функция форматированного ввода

Функция **scanf** предназначена для форматированного ввода данных и имеет следующий заголовок:

```
int scanf(const char * restrict format [,addresses,...]);
```

Пример ввода двух переменных целого типа и переменной вещественного типа:

```
int a,b;
```

```
double x;
```

```
...
```

```
scanf(“%d %d %lf”,&a,&b,&x);
```

Спецификатор типа

Сочетание `%d` или любое другое называется спецификатором типа. В общем случае он имеет следующий формат:

`% [*] [WIDTH] [hh|h|l|ll|L] символ_типа`

* – указывает запрещенные для ввода символы.

WIDTH – задает максимальную длину поля ввода.

Спецификатор типа

Преобразователи типа:

hh - преобразование целочисленных типов к типам **char** или **unsigned char**.

h - преобразование целочисленных типов к типам **short int** или **unsigned short int**.

l - преобразование к типу **long int** всех целочисленных типов и к типу **double** всех вещественных типов.

ll - преобразование целочисленных типов к типам **long long int** или **unsigned long long int**.

L - преобразование к типу **long double** всех вещественных типов.

Спецификатор типа

Спецификаторы типа:

d, i – знаковое целое число в десятичной системе

o – знаковое целое число в восьмеричной системе

X, x – знаковое целое число в шестнадцатеричной системе

u – незначающее целое число в десятичной системе исчисления

f, e и g – вещественное число в десятичной системе исчисления

a - вещественное число в шестнадцатеричной системе исчисления

c – символ

s – строка

p – указатель

% - ввод символа '%'

Функция форматированного вывода

Функция **printf** предназначена для форматированного вывода данных и имеет следующий заголовок:

```
int printf(const char * restrict format [,variables,...]);
```

Пример вывода двух переменных целого типа и переменной вещественного типа:

```
int a,b;
```

```
double x;
```

```
...
```

```
printf(“%d %d %5.2lf”,a,b,x);
```

Спецификатор типа

Сочетание `%d` или любое другое называется спецификатором типа. В общем случае он имеет следующий формат:

`% [flag] [WIDTH][.PREC] [h|hh|l|ll|L] символ_типа`

`flag` – для чисел указывает на необходимость вывода знака ‘+’ для положительных чисел, для строк управляет форматированием – по левому или по правому краю.

`WIDTH` – задает длину поля.

`.PREC` – задает количество символов после запятой для вещественных чисел и минимальное количество знаков для целых чисел.

Спецификатор типа

`%c` – вывод или ввод
символа

```
char ch;  
scanf("%c",&ch);  
printf("%c",ch);
```

`%hhc` – ввод или
вывод целого
значения размера 1
байт

```
char val;  
scanf("%hhd",&ch);  
printf("%hhd",ch);
```

Спецификатор типа

`%c` – ввод ASCII символа

`%hd` – ввод целого числа со знаком размером 1 байт в десятичной системе

`%hu` – ввод целого числа без знака размером 1 байт в десятичной системе

`%ho` – ввод целого числа со знаком размером 1 байт в восьмеричной системе

`%hx` – ввод целого числа со знаком размером 1 байт в шестнадцатеричной системе

Спецификатор типа

`%hd` – ввод целого числа со знаком размером 2 байта в десятичной системе

`%hu` – ввод целого числа без знака размером 2 байта в десятичной системе

`%ho` – ввод целого числа со знаком размером 2 байта в восьмеричной системе

`%hx` – ввод целого числа со знаком размером 2 байта в шестнадцатеричной системе

Спецификатор типа

`%d` – ввод целого числа со знаком размером 4 байта в десятичной системе

`%i` – ввод целого числа без знака размером 4 байта в десятичной системе

`%o` – ввод целого числа со знаком размером 4 байта в восьмеричной системе

`%x` – ввод целого числа со знаком размером 4 байта в шестнадцатеричной системе

Спецификатор типа

`%lld` – ввод целого числа со знаком размером 8 байт в десятичной системе

`%llu` – ввод целого числа без знака размером 8 байт в десятичной системе

`%llo` – ввод целого числа со знаком размером 8 байт в восьмеричной системе

`%llx` – ввод целого числа со знаком размером 8 байт в шестнадцатеричной системе

Спецификатор типа

`%f` – ввод или вывод вещественного числа типа `float` в десятичной системе

`%a` – ввод или вывод вещественного числа типа `float` в шестнадцатеричной системе

`%lf` – ввод или вывод вещественного числа типа `double` в десятичной системе

`%la` – ввод или вывод вещественного числа типа `double` в шестнадцатеричной системе

`%Lf` – ввод или вывод вещественного числа типа `long double` в десятичной системе

`%La` – ввод или вывод вещественного числа типа `long double` в шестнадцатеричной системе

Пример

Даны два вещественных числа, организовать их ввод в формате: (x,y) и вывести на экран в формате X = значение, Y = значение.

```
#include<stdio.h>
int main(int argc, char *argv[])
{
    double x,y;
    scanf("(%lf,%lf)",&x,&y);
    printf("X = %5.2lf, Y = %5.2lf",x,y);
    return 0;
}
```

Операторы присвоения

Синтаксис оператора присвоения языка C имеет вид:

LValue = RValue;

LValue – объект, в который будет записано присваиваемое значение. В качестве такого объекта в языке C может выступать только переменная.

RValue – объект, значение которого будет присвоено. В качестве такого объекта в языке C может выступать:

- переменная,
- константа,
- оператор вызова функции,
- математическое или логическое выражение.

Примеры присвоений

```
int a, b, c;
```

```
double x, y;
```

```
a = 5; b = 4; c = a + b;
```

```
x = 5.0; y = exp(x);
```

Усовершенствованные операторы присвоений

В языке C присутствуют усовершенствованные операторы присвоения, которые имеют следующий синтаксис:

LValue **X**= RValue;

где X – символ, означающий определенную математическую операцию из набора: + - * / % ^ & | << >>.

Усовершенствованные операторы присвоений

Использование усовершенствованного оператора присвоения аналогично записи:

$LValue = LValue \times RValue;$

Пример:

$a += b; \equiv a = a + b;$

$b -= c; \equiv b = b - c;$

$d *= a; \equiv d = d * a;$

Математические операции

В языке C математические операции делятся на две группы:

- математические операции для вещественных и целочисленных вычислений;
- математические операции только для целочисленных вычислений.

Математические операции для вещественных и целочисленных вычислений

К математическим операциям для вещественных и целочисленных вычислений языка C относят обычные арифметические операции:

- сложения (+),
- вычитания (-),
- умножения (*),
- деления (/).

Соответствие типа результата от типов операндов

Тип первого операнда	Тип второго операнда	Тип результата
Целый	Целый	Целый
Целый	Вещественный	Вещественный
Вещественный	Целый	Вещественный
Вещественный	Вещественный	Вещественный

Особенности языка C

Дан фрагмент программы:

```
int a,b; double c;
```

```
a = 10; b = 4;
```

```
c = a / b;           // c == 2
```

Дан фрагмент программы:

```
double x = 1 / 3; // x == 0
```

Операции для целочисленных вычислений

К операциям целочисленных вычислений относятся:

- операция взятия остатка от деления,
- побитовые операции,
- операции сдвигов,
- операции инкремента и декремента.

Остаток от деления

Операция взятия остатка от деления является бинарной операцией и в языке C обозначается символом процента (%).

Пример вычисления:

```
int a = 10, b = 3, c;  
c = a % b;           // c == 1
```

Побитовые операции

Побитовые операции языка C представлены тремя бинарными и одной унарной операцией.

К бинарным побитовым операциям относятся:

- операция «И» (&),
- операция «ИЛИ» (|)
- операция «Исключающее ИЛИ» (^).

Побитовые операции

Первый операнд	Второй операнд	Операция		
		И	ИЛИ	Исключающее ИЛИ
0	0	0	0	0
1	0	0	1	1
0	1	0	1	1
1	1	1	1	0

Побитовые операции

Унарной побитовой операцией является операция отрицания, обозначаемая символом тильды (\sim).

Пример:

```
unsigned char a = 10, b; //a: 00001010 = 10
```

```
b = ~a;           //b: 11110101 = 245
```

Операции сдвига

Операции сдвига осуществляют побитовый сдвиг целого значения, указанного в первом операнде, вправо (символ `>>`) или влево (символ `<<`) на указанное во втором операнде целое число бит.

Пример:

```
unsigned char a = 10, b, c; //a: 00001010 = 10
```

```
b = a << 2;           //b: 00101000 = 40
```

```
c = a >> 1;          //c: 00000101 = 5
```

Операции инкремента и декремента

Операции инкремента (знак ++) и декремента (знак --) являются унарными и осуществляют увеличение и уменьшение целого значения на единицу соответственно.

```
int a = 10, b, c;
```

```
b = ++a; //пред- инкремент b == 11
```

```
c = a++; //пост- инкремент c == 11
```

Операции инкремента и декремента

В современных языках программирования (в том числе и языке C стандарта C99) данные операции могут использоваться и для вещественных значений.

Пример:

```
double x = 12.5;
```

```
x++;
```

```
printf(“%lf\n”,x); //вывод: 13.5
```

Операции отношения (сравнения)

В языках программирования операции отношения (сравнения) являются бинарными операциями, осуществляющими сравнение двух операндов и возвращающие результат сравнения в виде логического значения.

В языке C принято логические значения ИСТИНА и ЛОЖЬ интерпретировать посредством целочисленных значений:
0 – ЛОЖЬ, 1 – ИСТИНА.

Операции отношения (сравнения)

Обозначение

Название

$>$

Больше

$<$

Меньше

\geq

Больше или равно

\leq

Меньше или равно

$=$

Равно

\neq

Не равно

Примеры

Несколько примеров использования операций сравнения:

```
int a=5, b=4, c=10, x, y;
```

```
x = a > b;           //x == 1
```

```
y = c == a;         //y == 0
```

Логические операции

Логические операции – унарные или бинарные операции, осуществляющие действия над логическими значениями и возвращающие логическое значение.

Набор логических операций у разных языков программирования может быть различен.

Логические операции

Название	Обозначение	Описание
И	&&	Логическое «И» – возвращает значение ИСТИНА, если оба операнда имеют значение ИСТИНА
ИЛИ		Логическое «ИЛИ» - возвращает значение ИСТИНА, если хотя бы один из операндов имеет значение ИСТИНА
НЕ	!	Логическое «НЕ» - унарная операция, инвертирует логическое значение своего операнда

Примеры

Примеры логических операций:

```
int a=1, b=0, c, d; //a – ИСТИНА, b – ЛОЖЬ
```

```
c = a || b;           //c == 1
```

```
d = !b && a;         //d == 1
```

Приоритеты операций

++, --	Операции пост- инкремента и декремента
()	Вызов функции, группировка операций
[]	Обращение к элементу массива
->	Обращение к полю структуры или объединения через указатель
.	Обращение к полю структуры или объединения
++, --	Операции пред- инкремента и декремента
!	Логическое «НЕ»
~	Бинарное отрицание (инверсия)
+, -	Унарные плюс и минус
&	Операция взятия адреса
*	Разыменование указателя
sizeof	Оператор определения размера
(type)	Оператор преобразования типа

Приоритеты операций

*	Умножение
/	Деление
%	Взятие остатка от деления
+	Сложение
-	Вычитание
<<, >>	Побитовые сдвиги влево и вправо
<, <=, >, >=	Операции сравнения
==, !=	Операции сравнения
&	Побитовое «И»
^	Побитовое «Исключающее ИЛИ»
	Побитовое «ИЛИ»
&&	Логическое «И»
	Логическое «ИЛИ»
?:	Условная операция

Приоритеты операций

=	Оператор простого присвоения
*=, /=, %=, +=, -=, <<=, >>=, &=, ^=, =	Усовершенствованные операторы присвоения
,	Запятая

Особенности трансляторов

Не определяется порядок, в котором вычисляются аргументы функции при ее вызове. Поэтому следующий оператор может дать различные результаты при трансляции разными компиляторами:

```
printf(“%d %lf\n”, ++n, pow(2.0,n));
```

Результат будет зависеть от того, получает ли n приращение до или после вызова функции `pow`. Чтобы решить проблему достаточно записать так:

```
n++;
```

```
printf(“%d %lf\n”, n, pow(2.0,n));
```

Схема автоматического приведения типа

- Если какой-либо из операторов имеет тип **long double**, то и другой приводится к **long double**.
- Иначе, если какой-либо из операторов имеет тип **double**, то и другой приводится к **double**.
- Иначе, если какой-либо из операторов имеет тип **float**, то и другой приводится к **float**.
- Иначе, для обоих операндов выполняется расширение целого типа; затем, если один из операндов имеет тип **unsigned long int**, то другой преобразуется в **unsigned long int**.

Схема автоматического приведения типа

- Иначе, если один из операндов имеет тип **long int**, а другой – **unsigned int**, то результат зависит от того, представляет ли **long int** все значения **unsigned int**; если это так, то операнд типа **unsigned int** приводится к типу **long int**; если нет, то оба операнда преобразуются в **unsigned long int**.
- Иначе, если один из операндов имеет тип **long int**, то и другой приводится к **long int**.
- Иначе, оба операнда имеют тип **int**.

Оператор приведения типа

```
int a = 15, b = 2;  
double r = 0.0;  
r = a / b; //r == 7.0
```

Оператор приведения типа:
(тип)выражение

```
r = (double)a / b; //Правильно
```

```
r = (double) (a / b); //Неправильно
```

Условная операция

В языке C присутствует так называемая условная операция, которая имеет следующий синтаксис:

условие ? выражение №1 : выражение №2;

Если условие истинно, то вычисляется выражение №1.

Если же условие ложно, то вычисляется выражение №2.

Пример условной операции

Необходимо ввести с клавиатуры два вещественных значения и вывести на экран максимальное из этих значений:

```
#include <stdio.h>
int main(int argc, char *argv[])
{
    double x,y;
    printf("Введите значения: ");
    scanf("%lf %lf",&x,&y);
    double max = (x > y) ? x : y;
    printf("Максимальное значение: %lf\n",max);
    return 0;
}
```

Пример условной операции

Необходимо ввести с клавиатуры три вещественных значения и вывести на экран максимальное из этих значений:

```
#include <stdio.h>
int main(int argc, char *argv[])
{
    double x, y, z;
    printf("Введите значения: ");
    scanf("%lf %lf %lf",&x,&y,&z);
    double max = (x > y) ?
                ((x > z) ? x : z):
                ((y > z) ? y : z);
    printf("Максимальное значение: %lf\n",max);
    return 0;
}
```

Пример 1

Вещественное число вводится с клавиатуры. Возвести число в четвертую степень, используя только две операции умножения.

```
#include <stdio.h>
int main(int argc, char *argv[])
{
    double a;
    printf("Введите значение: ");
    scanf("%lf",&a);
    a *= (a *=a);
    printf("Результат: %lf\n",a);
    return 0;
}
```

Пример 2

Квадратное уравнение вида $ax^2 + bx + c = 0$ задается коэффициентами А, В и С. Определить какое количество корней имеет данное уравнение.

```
#include <stdio.h>
int main(int argc, char *argv[])
{
    double a,b,c;
    printf("Введите коэффициенты А, В и С: ");
    scanf("%lf %lf %lf",&a,&b,&c);
    double d = b*b-4*a*c;
    int n = (d < 0.0)?0:(d > 0.0)?2:1;
    printf("Количество корней: %d\n",n);
    return 0;
}
```