Лекция №3. Преобразование типов при вычислении выражения.

Преобразование типов делится на явные и неявные. Явные выполняет программист, неявные выполняет вычислительная система.
Для выполнения явного преобразования программист перед преобразуемым выражением должен указать имя требуемого типа в круглых скобках.

```
14.0 / 5 → 2.8

int a = 14, b = 5;
(double)a / b → 2.8
(double)(a / b) → 2.0

(int)2.8 → 2
```

Неявные преобразования выполняются по определенным правилам, например, рассмотрим данные правила для арифметических операций.

1. Операнды типа float преобразуются к типу *double*.
2. Если один операнд long double, то второй преобразуется к этому типу.
3. Если один из операндов double, то и второй преобразуется к этому типу.
4. Char преобразуется к типу Int.
5. Unsigned char и unsigned short преобразуется к unsigned int.
6. Если один из операндов unsigned long, то и другой преобразуется к этому типу.
7. Если один из операндов long, то и второй преобразуется к этому типу.
8. Если один из операндов unsigned int, то и второй преобразуется к этому типу.

Структура программы.

С-программы состоят из групп, которые вызывают друг друга. Функции могут располагаться в различных файлах. Вне функций разрешается определять типы данных и переменные. Выполнение программы начинается с функции именем main. Все операторы кроме составных заканчиваются точкой с запятой. Любое выражение заканчивающееся точкой с запятой есть оператор выражения.

Обычно данное выражение имеет побочный эффект, заключающееся в изменении значения переменной.

```
y = sin(x)
a++;
printf("123\n")
z + 8;
```

Составной оператор.

Составной оператор представляет собой группу операторов и объявлений, заключенных в фигурные скобки.

```
{
        int tmp = a
        a = b
        b = tmp
}
```

Ввод/вывод.

Средства ввода/вывода не являются составной частью языка С. Рассмотрим стандартные способы ввода и вывода.

```
#include <stdioh>
```

Для форматного ввода/вывода используется функция printf(), которая имеет следующий формат:

```
printf(control[Arg1, Arg2 ... ArgN])
```

Функция print преобразует свои аргументы и передает в стандартный вывод под управление строки и control. Управляющая строка содержит два вида символов: обычные символы, которые просто копируются в выходной поток; спецификации преобразования, каждый из которых управляет печатью очередного аргумента. Начальные преобразования начинаются с символа % и заканчивается символом преобразования.
Между ними могут находиться…

- **Знак минус**, который обеспечивает выравнивание по левому краю поля
- **Строка цифр**, задающая минимальную ширину поля, если преобразуемый элемент имеет большую ширину, то поле автоматически расширяется. Если ширина аргумента меньше ширины поля, то выводятся дополняющие символы (обычно это пробел, но если ширина поля указывается с лидирующим нулем, то дополняющим символом является ноль).
- **Точка и строка цифр**, которые указывают количество знаков дробной части.

Рассмотрим некоторые символы преобразования.

Код	Формат
%c	Символ типа char
%d	Десятичное число целого типа со знаком
%i	Десятичное число целого типа со знаком
%e	Научная нотация (e нижнего регистра)
%E	Научная нотация (E верхнего регистра)
%f	Десятичное число с плавающей точкой
%g	Использует код %e или %f — тот из них, который короче (при использовании %g используется e нижнего регистра)
%G	Использует код %E или %f — тот из них, который короче (при использовании %G используется E верхнего регистра)
%o	Восьмеричное целое число без знака
%s	Строка символов
%u	Десятичное число целого типа без знака
%x	Шестнадцатиричное целое число без знака (буквы нижнего регистра)
%X	Шестнадцатиричное целое число без знака (буквы верхнего регистра)
%p	Выводит на экран значение указателя
%n	Ассоциированный аргумент — это указатель на переменную целого типа, в которую помещено количество символов, записанных на данный момент
%%	Выводит символ %

```
int a = 1, b = 2;

printf("Пример;\n");                  // Вывод: Пример;
printf("%d + %d = %d\n", a, b, a+b); // Вывод: 1 + 2 = 3

int day = 20, month = 9, year = 2024;
printf("%02d.%02d.%d\n", day, month, year); // Вывод: 20.09.2024

char g = 'a';
printf("%c %d\n", g, g);              // Вывод: a 97 (тк код ASCII буквы 'a' =
97)
```

```
double pi = 3.14159;
printf("%7s %6.3lt", "abcde", pi);   // Вывод: abcde   3.141
```

Для ввода используется функция scan, которая имеет следующий формат...

```
scanf(control[Arg1, Arg2 ... ArgN])
```

Аргументами функции scan являются адреса переменных.

```
int a, b;
scanf("%d%d", 8a, 8b)
```

При вводе строк операция взятия адреса не используется, т.к. адрес строки находится у соответствующего массива.

```
char but[64]
scanf("%s", but)
```