Операторы ветвления. Приведение типов

Лекция 3

Приведение типов

Преобразования бывают *явными* и *неявными*.

Неявные преобразования производит сам компилятор без помощи программиста, их можно разделить на два класса:

1. Когда мы присваиваем результат вычисления какого-либо выражения переменной – этот результат преобразуется к типу переменной;

2. при вычислении выражения операнды каждой операции преобразуются к какому-то одному типу (если они различаются), при этом результат операции будет того же типа.

Для минимизации потерь меньший тип преобразуется к большему:

`char ≤ short ≤ float ≤ int ≤ long ≤ double`

Неявное преобразование при присваивание

Пример 1:

```
int a = 2.3 + 4.5;
```

По логике результат должен быть дробным

По факту он будет целочисленный

Пример 2:

```
float a = 5 / 6; 
```
-не правильно

```
float a = 5.0 / 6; 
```
-правильно

Неявное преобразование при присваивание

Пример 3:

```
int a, b;
double f1, f2;
f1 = a / (f2 + b) – b;
```

- 1. сложение преобразует тип значения переменной `b` к `double` и выдаст результат типа `double`,
- 2. деление преобразует значение `a` тоже к типу `double`, так как результат только что вычисленного второго операнда будет типа `double`,
- 3. наконец, вычитание сделает ровно то же самое, так как первый операнд будет иметь тип `double`.

Неявное преобразование при присваивание

Чему будет равно значение выражения 2 / 4 * 3.14?

Ожидается 1.57

По факту 0

2 и 4 оба целочисленные, результат деления 0

0 умножаем на 3.14 получаем 0

Явное преобразование типа

Пример 1:

Не правильно

```
int a = 1, b = 2;
double result = a / b;
```

Правильно

```
int a = 1, b = 2;
double result = (double)a / b;
```

Пример 2:

```
float a = 4.0, b = 2.0;
int result;
result = (int)((a * a) / (b * 6) + 4);
```

Операторы

; - пустой оператор

= - оператор присваивания

Идущие друг за другом операторы называются *последовательностью операторов*, что является первой базовой конструкцией любого структурированного языка программирования. При этом программа выполняется именно в этой последовательности – прямолинейно, оператор за оператором:

<оператор 1>;

<оператор 2>;

…

<оператор N>;

Блок операторов

Операторы можно объединять в группы (блоки) с помощью фигурных скобок, создавая тем самым *сложные (составные) операторы*:

```
{
  <оператор 1>;
  <оператор 2>;
  …
  <оператор N>;
}
```

Операторы ветвления. Условный оператор

- Операторы ветвления предназначены для управления ходом выполнения программы. Они выбирают, какую часть программы выполнять в зависимости от истинности или ложности заданных условий. Одним из типов ветвлений в Си является **if ... else**. Этот тип ветвления осуществляет выбор между двумя альтернативами.
- **if** (condition){
  statement1;
  statement2;
  }
  else{
  statement3;
  statement4;
  }

Пример 1

Написать программу, спрашивающую у пользователя два числа и выводящую «TRUE», если первое меньше второго. В противном случае вывести «FALSE».

Решение:

```
int a, b;
printf("Введите два числа: ");
scanf_s("%d %d", &a, &b);
if (a < b)
	printf("TRUE\n");
else
	printf("FALSE\n");
```

if…else Дополнительные сведения

- Ветвления могут быть вложенными.

```
if (condition1){
    if (conditon2)
          statement1;
}
else{
    if (condition3)    {
        statement3;
        statement4;
    }
}
```

Пример 2

Написать программу, спрашивающую у пользователя три числа и выводящую наименьшее и наибольшее из них.

Решение:

```
int a, b, c, max, min;
printf("Введите три числа: ");
scanf_s("%d %d %d", &a, &b, &c);
if (a >= b && a >= c){
        max = a;
}
else {
if (b >= a && b >= c)
        max = b;
else
        max = c;
}
```

```
if (a <= b && a <= c){
        min = a;
}
else {
if (b <= a && b <= c)
min = b;
else
min = c;
}
printf("Max = %d, min = %d\n", max, min);
```

Конструкция switch

```
switch(выражение)
{
    case константа_1: инструкции_1; break;
    case константа_2: инструкции_2; break;
     …
    default: инструкции;
}
```

Конструкция switch

```
int main()
{
        int day;
        while(1){
                day=getDayOfWeek;
                switch (day)
                {
                   case 1: printf(“Понедельник”);  break;
                   case 2: printf(“Вторник”);   break;
                   case 3: printf(“Среда”);   break;
                   case 4: printf(“Четверг”);  break;
                   case 5: printf(“Пятница”); break;
                   case 6: printf(“Суббота”); break;
                   case 7: printf(“Воскресенье”);
                }
        }
}
```

Конструкция switch

- После ключевого слова switch в скобках идет сравниваемое выражение. Значение этого выражения последовательно сравнивается со значениями после оператора case. И если совпадение будет найдено, то будет выполняться данный блок case.
- В качестве констант после оператора case могут выступать значения типов char, int и unsigned
- В конце конструкции switch может стоять блок default. Он необязателен и выполняется в том случае, если ни одна совпадения в блоках case не было найдено. Например, сравним значение переменной с набором значений: