Арифметические действия в языке Си

Программы работают с данными. Зачастую данные представляют собой числа. В этом уроке, как вы наверное догадались, мы будем заниматься изучением того, как и что в языке Си можно делать с числами. Начнём с арифметики.

Компилятор языка Си понимает все основные арифметические операции, которые вам известны со школы. Плюс есть несколько дополнительных.

Основные арифметические операторы языка Си.
`+` оператор сложения
`-` оператор вычитания
`*` оператор умножения
`%` оператор взятия остатка от деления
`/` оператор деления

Следующая программа иллюстрирует использование первых четырёх из них. Кстати, обратите внимание на то, как с помощью функции **`printf`** вывести на экран символ `%`.

Листинг 1.

```
#include <stdio.h>
int main(void){
  int a=7, b=2;
  int res;
  res = a+b;
  printf("%d + %d = %d\n",a,b,res);
  res = a-b;
  printf("%d - %d = %d\n",a,b,res);
  res = a*b;
  printf("%d * %d = %d\n",a,b,res);
  res = a%b;
  printf("%d %% %d = %d\n",a,b,res);
  return 0;
}
```

Результат работы этой программы представлен на следующем рисунке.

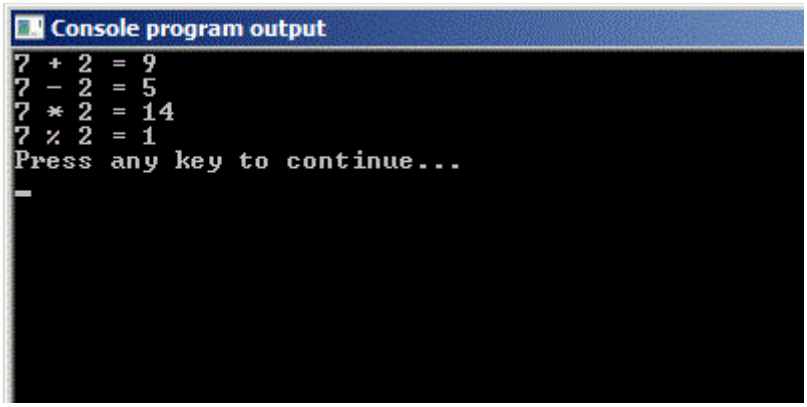

Рис.5 Использование арифметических действий в Си.

Всё чётко и понятно. Никаких неожиданностей. А теперь попробуем получить частное двух чисел. Т.к. результат должен получиться 3.5, то **res** объявим как **float**.

Листинг 2.

```
#include <stdio.h>
int main(void){
  int a=7, b=2;
  float res;

  res = a/b;
  printf("%d / %d = %f\n",a,b,res);

  return 0;
}
```

Как видите, результат получился не тот, что мы ожидали. Это одна из особенностей оператора деления в языке Си.

Целочисленное деление.

При делении значение целого типа на значение целого типа результат тоже получается целого типа.

Так уж устроен язык Си. Поэкспериментируйте, попробуйте любые другие целые числа.

Вычислить результат целочисленного деления легко. Поделите числа и отбросьте всё, что получилось в дробной части.

Пример: Как получить результат целочисленного деления

```
7/2  = 3.5 → 3
11/3 = 3.66 → 3
2/5  = 0.4 → 0
```

Для того чтобы получить тот результат, который мы в данном случае ожидаем, одно из значений нужно сделать вещественным. Сделать это проще простого. Для этого необходимо рядом с ним в скобках записать **float**.

Посмотрим на нашем примере:

Листинг 3.

```
#include <stdio.h>
int main(void){
  int a=7, b=2;
  float res;

  res = (float)a/b;
  printf("%d / %d = %f\n",a,b,res);

  return 0;
}
```

Теперь результат будет тот, что мы ожидали. Проделанный нами трюк называется явным преобразованием типа.

Явное преобразование (приведение) типа

Если какое-то значение нужно привести к другому типу, нужно перед этим значением в скобках написать название требуемого типа.

Листинг 4. Примеры явного преобразования типа

```
int a=7, b;
float g= 9.81,v;

b = (int) g; //приводим значение 9.81 к типу int, получим 9
v= (float)a; // приводим значение 7 к типу float, получим 7.0
```

Важный момент: преобразуется не тип исходной переменной, а только лишь значение, которое используется в выражении. В видео-фрагменте об этом говорится подробнее.

Обратите внимание, что, когда мы преобразовываем целое значение в вещественное, ничего особенного не происходит, т.к. вещественные числа включают в себя целые.

Совсем иная ситуация, когда мы от вещественного переходим к целому. При этом переходе у нас теряется вся дробная часть. Не забывайте об этом.

Картинка, показывающая различия между операциями взятие остатка, целочисленного деления и обычного деления.

Рис.2 Деление, целочисленное деление и остаток от деления.

Практика.

1. Решите предложенные задачи с автоматической проверкой решения.

Исследовательские задачи для хакеров:

1. Подумайте и приведите примеры, когда обычное деление не имеет смысла. Например, деление трёх лицензионных ключей от программы между двумя людьми. Зачем кому-то нужна половина лицензионного ключа? (если, конечно, он не занимается reverse engineering).

Интернет версия: http://youngcoder.ru/less4/