Содержание

операторы

Арифметические операторы

Бинарными[1)] арифметическими операторами являются +, -, *, /, а также оператор деления по модулю %. Деление целых сопровождается отбрасыванием дробной части, какой бы она ни была.

Оператор % к операндам типов float и double не применяется. В какую сторону (в сторону увеличения или уменьшения числа) будет усечена дробная часть при выполнении / и каким будет знак результата операции % с отрицательными операндами, зависит от машины.

Операторы отношения и логические операторы

Операторами отношения являются:

- > - больше
- >= - больше или равно
- < - меньше
- ⇐ - меньше или равно

Операторы сравнения на равенство:

- == - равно
- != - не равно

Логические операторы:

- && - или
- || - и

Выражения, между которыми стоят операторы && или ||, вычисляются слева направо. Вычисление прекращается, как только становится известна истинность или ложность результата

Унарный оператор:

- ! - не

Операторы присвоения

представляют из себя сокрашенные выражения

список

- += - присваивание после сложения. Присваивает левому операнду сумму левого и правого операндов: A += B эквивалентно A = A + B
- -= - присваивание после вычитания. Присваивает левому операнду разность левого и правого операндов: A -= B эквивалентно A = A - B
- *= - присваивание после умножения. Присваивает левому операнду произведение левого и правого операндов: A *= B эквивалентно A = A * B
- /= - присваивание после деления. Присваивает левому операнду частное левого и правого операндов: A /= B эквивалентно A = A / B
- %= - присваивание после деления по модулю. Присваивает левому операнду остаток от целочисленного деления левого операнда на правый: A %= B эквивалентно A = A % B
- «= - присваивание после сдвига разрядов влево. Присваивает левому операнду результат сдвига его битового представления влево на определенное количество разрядов, равное значению правого операнда: A «= B эквивалентно A = A « B
- »= - присваивание после сдвига разрядов вправо. Присваивает левому операнду результат сдвига его битового представления вправо на определенное количество разрядов, равное значению правого операнда: A »= B эквивалентно A = A » B
- &= - присваивание после поразрядной конъюнкции. Присваивает левому операнду результат поразрядной конъюнкции его битового представления с битовым представлением правого операнда: A &= B эквивалентно A = A & B
- |= - присваивание после поразрядной дизъюнкции. Присваивает левому операнду результат поразрядной дизъюнкции его битового представления с битовым представлением правого операнда: A |= B эквивалентно A = A | B
- ^= - присваивание после операции исключающего ИЛИ. Присваивает левому операнду результат операции исключающего ИЛИ его битового представления с битовым представлением правого операнда: A ^= B эквивалентно A = A ^ B

Операторы сдвига

Каждое целое число в памяти представлено в виде определенного количества разрядов. И операции сдвига позволяют сдвинуть битовое представление числа на несколько разрядов вправо или влево. Операции сдвига применяются только к целочисленным операндам.

- « - Сдвигает битовое представление числа, представленного первым операндом, влево на определенное количество разрядов, которое задается вторым операндом.
- » - Сдвигает битовое представление числа вправо на определенное количество разрядов.

```
int a = 2 << 2;  // 0b10 на два разрядов влево = 0b1000 = 8
int b = 16 >> 3; // 0b10000 на три разряда вправо = 0b10 = 2
```

Поразрядные операции

Поразрядные операции также проводятся только над разрядами целочисленных операндов.

- & - поразрядная **конъюнкция** (операция И или поразрядное умножение). Возвращает 1, если оба из соответствующих разрядов обоих чисел равны 1
- | - поразрядная **дизъюнкция** (операция ИЛИ или поразрядное сложение). Возвращает 1, если хотя бы один из соответствующих разрядов обоих чисел равен 1
- ^ - поразрядное исключающее ИЛИ. Возвращает 1, если только один из соответствующих разрядов обоих чисел равен 1
- ~ - поразрядное отрицание. Инвертирует все разряды операнда. Если разряд равен 1, то он становится равен 0, а если он равен 0, то он получает значение 1.

```
int a = 5 | 2; // 0b101 | 0b010 = 0b111 = 7
int b = 6 & 2; // 0b110 & 0b010 = 0b10 = 2
int c = 5 ^ 2; // 0b101 ^ 0b010 = 0b111 = 7

int f = 12; // 0b00001100
int d = ~f; // 0b11110011 или -13
```

Преобразования типов

В ряде случаев преобразования сопровождаются потерей информации, например, когда числа большей разрядности (скажем размером 4 байт) получаем число меньшей разрядности (например, в 2 байта). Без потери информации проходят следующие цепочки преобразований:

- `char → short → int → long`
- `unsigned char → unsigned short → unsigned int → unsigned long`
- `float → double → long double`

пример потери данных

```
#include <stdio.h>

int main(void){
  int number1 = 300;
  char code = number1; // потеря точности - число  number1  усекается до 1 байта
  printf("code = %d \n", code); // code = 44

  short number2 = 100000; // потеря точности - число 100000 усекается до 2 байт
  printf("number2 = %d \n", number2); // number2 = -31072

  return 0;
}
```

Число 300 в двоичной системе = 0000000100101100, оставляем только первый младший байт: 00101100, и у нас получается число 44 в десятичной системе.

Неявное преобразование

Преобразования, применяемые компилятором при арифметических операциях:

1. Если один из операндов имеет тип long double, то второй операнд тоже будет преобразован в тип long double
2. Если предыдущий пункт не выполняется и если один из операндов имеет тип double, то второй операнд тоже будет преобразован к типу double
3. Если предыдущий пункт не выполняется и если один из операндов имеет тип float, то второй операнд тоже будет преобразован к типу float
4. Если предыдущий пункт не выполняется и если один из операндов имеет тип unsigned long int, то второй операнд тоже будет преобразован к типу unsigned long int
5. Если предыдущий пункт не выполняется и если один из операндов имеет тип long, то второй операнд тоже будет преобразован к типу long
6. Если предыдущий пункт не выполняется и если один из операндов имеет тип unsigned, то второй операнд тоже будет преобразован к типу unsigned
7. Если предыдущий пункт не выполняется то оба операнда приводятся к типу int

Явное преобразование

Явное преобразование происходит через операторы преобразования:

```
int a = 10;
int b = 4;
int c = a / b;
double d = a / b;
double e = (double)a / (double)b;
printf("c = %d \n", c); // 2
printf("d = %f \n", d); // 2.00000
printf("e = %f \n", e); // 2.50000

int number = 70;
char symbol = (char) number;
printf("symbol = %c \n", symbol); //  F
printf("symbol (int code) = %d \n", symbol); // 70
```

Приоритет и очередность

Операторы	Выполняются
() [] -> .	слева направо
! ~ ++ -- + - * & (тип) sizeof	справа налево
* / %	слева направо
+ -	слева направо
<< >>	слева направо
< <= > >=	слева направо
== !=	слева направо
&	слева направо

Операторы	**Выполняются**
^	слева направо
|	слева направо
&&	слева направо
||	слева направо
?:	справа налево
= += -= *= /= %= &= ^= |= <<= >>=	справа налево
,	слева направо

Примечание. Унарные операторы +, -, * и & имеют более высокий приоритет, чем те же бинарные операторы.

1)

т.е. с двумя операндами