Логические и поразрядные операции

Логическое отрицание

Оператор НЕ (NOT) используется для того, чтобы инвертировать значение аргумента. Т.е., если ему передали истину, то он вернёт ложь, если получил ложь в качестве аргумента, то вернёт истину.

В СИ отрицание представлено оператором **!**.

Логическое отрицание

Здесь действует закон двойного отрицания – отрицание отрицания можно опустить.

```c
#include <stdio.h>

int main()
{    int i = 0;
    if (i)
        { printf("i is true\n"); }
    if (!i)
        { printf("i is not true\n"); }
    if (!!i)
        { printf("i is not not true\n");}
    if (!!!i)
        { printf("i is not not not true\n"); }
    return 0; }
```

```
i is not true
i is not not not true

Process returned 0 (0x0)   execution time : 0.172 s
Press any key to continue.
```

Логическое И

Оператор И (AND, логическое умножение) возвращает истину тогда и только тогда, когда оба аргумента являются истиной.

В СИ логическое умножение представлено оператором &&.

Оператор И может применяться последовательно к нескольким аргументам.

Логическое И

Пример программы, которая отбирает анкеты мужчин старше 21 года и ростом более 180 см.

```c
#include <stdio.h>

int main() {
    char gender;
    unsigned int age;
    unsigned int height;

    printf("Enter gender ('M' or 'W')\n");
    scanf("%c", &gender);
    printf("Enter age\n");
    scanf("%u", &age);
    printf("Enter height\n");
    scanf("%u", &height);

    if (gender == 'M' && age > 21 && height >= 180)
        { printf("Wellcome"); }
        else { printf("Go away");}
    return 0;}
```

```
Enter gender ('M' or 'W')
M
Enter age
34
Enter height
182
Wellcome
Process returned 0 (0x0)   execution time : 13.126 s
Press any key to continue.
```

Логическое И

Пример программы, которая отбирает анкеты мужчин старше 21 года и ростом не менее 180 см.

```c
#include <stdio.h>

int main() {
    char gender;
    unsigned int age;
    unsigned int height;

    printf("Enter gender ('M' or 'W')\n");
    scanf("%c", &gender);
    printf("Enter age\n");
    scanf("%u", &age);
    printf("Enter height\n");
    scanf("%u", &height);

    if (gender == 'M' && age > 21 && height >= 180)
        { printf("Wellcome"); }
        else { printf("Go away");}
    return 0;}
```

```
Enter gender ('M' or 'W')
M
Enter age
34
Enter height
182
Wellcome
Process returned 0 (0x0)   execution time : 13.126 s
Press any key to continue.
```

Логическое ИЛИ

Оператор логическое ИЛИ (логическое сложение, OR) истинен тогда, когда истиной является хотя бы один его аргумент. В СИ ИЛИ представлен оператором ||. Например, усовершенствуем предыдущую программу: пол можно вводить как большой, так и маленькой буквой

Логическое ИЛИ

```c
#include <stdio.h>

int main() {
    char gender;
    unsigned int age;
    unsigned int height;

    printf("Enter gender ('M' or 'W')\n");
    scanf("%c", &gender);
    printf("Enter age\n");
    scanf("%u", &age);
    printf("Enter height\n");
    scanf("%u", &height);

    if ((age > 17 && height >= 180) && (gender == 'M' || gender == 'm'))
    { printf("Wellcome");}
    else { printf("Go away"); }
   return 0;
}
```

```
Enter gender ('M' or 'W')
m
Enter age
34
Enter height
178
Go away
Process returned 0 (0x0)   execution time : 12.157 s
Press any key to continue.
```

Порядок выполнения логических операторов

Оператор отрицания имеет больший приоритет, чем И или ИЛИ, поэтому будет выполняться в первую очередь. Если может случиться ситуация, когда порядок выполнения не ясен, определите его с помощью скобок.

Рассмотрим выражение

a && b && c && d где a, b, c, d – логические значения. Всё выражение равно истине тогда и только тогда, когда все операнды истинны. Если хотя бы один из операндов ложь, то остальные уже не важны. Поэтому, для оптимизации работы, вычисление происходит слева направо и останавливается, как только был найден первый операнд, равный нулю.

То же самое происходит и при выполнение ||. Выражение

a || b || c || d выполняется слева направо до тех пор, пока не встретит первое ненулевое значение. После этого выполнение останавливается, так как известно, что всё выражение равно

Тернарная операция ? (вопросительный знак) в СИ

- *Условная операция* ? работает не с двумя, а с тремя операндами (является *тернарной*). Она имеет следующий формат:
- операнд1 ? операнд2 : операнд3.
- Операнд1 вычисляется и сравнивается с нулем, при этом он может иметь целый, плавающий тип, либо быть указателем. Если операнд1 не равен 0 (истина), вычисляется операнд2 и результатом операции является его значение. В противном случае вычисляется операнд3 и результатом является его значение. В любом случае вычисляется только один из операндов 2 или 3, но не оба.

Тернарная операция ? (вопросительный знак) в СИ

- В формуле:
- y = x ? a: b;
- y = a, если x не равно нулю (истинно), и y = b, если x равно нулю (ложно).
- Следующее выражение:
- y = x<0 ? -x : x ;
- вычисляется абсолютное значение x (|x|).

Тернарная операция ? (вопросительный знак) в СИ

Пример использования операции ? для определения максимального значения

```c
#include <stdio.h>
float max (float x, float y)
{
    return (y <= x) ? x : y;
}

int main() {
    float a,b,c,d,e,f ;
    a=3.4; b=1.9; c= -7.6;
    d = max (a,b); e=max (a,c); f=max(b,c);
    printf("d=%f\t e=%f\t f=%f\n",d,e,f);
   return 0;}
```

```
d=3.400000      e=3.400000      f=1.900000

Process returned 0 (0x0)   execution time : 0.177 s
Press any key to continue.
```

Оператор , (запятая) в СИ

Его назначение – связать определенным образом несколько выражений. Значение списка выражений, разделенных запятыми, определяется крайним справа выражением.

Например, при выполнении инструкции

```
var= (count=19, incr=10, count+1);
```

переменной count присваивается сначала значение 19, переменной incr – значение 10, потом значение count увеличивается на 1 и присваивается переменной var (var=20). Круглые скобки обязательны, т.к. оператор «запятая» имеет более низкий приоритет по сравнению с оператором присваивания.

Оператор , (запятая) в СИ

Рассмотрим следующую программу

```c
#include <stdio.h>

int main() {
    int i,j;
    j=10;
    i=(j++,j+100,999+j);     //оператор запятая означает сделать и то и другое
    printf("i=%d\n",i);
   return 0;}
```

```
i=1010

Process returned 0 (0x0)   execution time : 0.141 s
Press any key to continue.
```

Операции сравнения

Приведен пример пользовательской функции, которая определяет наибольшее число из трех, переданных в функцию

```
#include <stdio.h>

 int max3 (int a, int b, int c)
{
   if (a >= b && a >= c)
      return a;
   if (b > a && b >= c)
      return b;
   if (c > a && c > b)
      return c;
}
int main()
{
     int a,b,c,M ;
     printf("a:  "); scanf("%d",&a);
     printf("b:  "); scanf("%d",&b);
     printf("c:  "); scanf("%d",&c);
     M = max3(a,b,c);
        printf("\t  M =%d \n", M);
              return 0;
}
```

Операции сравнения

То, что истинное выражение считается равным 1, ложное выражение считается равным 0, можно использовать при вычислении значений функции не используя структуру **if…..else**

Операции сравнения

- **Пример. Вычислить значение функции** $y(x) = \begin{cases} x+1, x \geq 0, \\ x^2, x < 0. \end{cases}$

```
#include <stdio.h>
int main( )

{   float x,y;
    scanf("%f",&x);
    y = (x + 1)*(x >= 0) + (x * x)*(x < 0);
    printf("x=%f\t y=%f\n",x,y);
    return 0;
}
```

```
5
x=5.000000       y=6.000000

Process returned 0 (0x0)   execution time : 5.641 s
Press any key to continue.
```

Операции сравнения

Пример. Вычислить значение функции знака числа:

$$\operatorname{sgn}(x) = \begin{cases} 1, x > 0, \\ 0, x = 0, \\ -1, x < 0. \end{cases}$$

Запишем функцию в виде:

sgn(x) = 1 * (x > 0) + 0 * (x = 0) + (-1) * (x < 0) = (x > 0) – (x < 0)

Функция sng(x)

```c
#include <stdio.h>
int sign(float x)
{
return (x > 0) - (x < 0);
}
int main( )
{  float x,y,z;
  x = 23;    y = -12; z = 0;
  printf("f(x)=%d\t f(y)=%d \tf(z)=%d\n",sign(x),sign(y),sign(z));
  return 0;
}
```

```
f(x)=1   f(y)=-1        f(z)=0

Process returned 0 (0x0)   execution time : 0.109 s
Press any key to continue.
```

Поразрядный сдвиг

Поразрядный сдвиг на 2 позиции соответствует делению на 4 .

```c
#include <stdio.h>

int main()
{
unsigned char a=36;
unsigned char b;
b = a >> 2;
printf ("b=%d\n",b);
return 0;
  }
```

```
b=9

Process returned 0 (0x0)   execution time : 0.172 s
Press any key to continue.
```

Побитовые операции. Маски

Побитовые операции позволяют осуществлять установку и сброс отдельных битов числа. С этой целью используется маскирование битов. Маски, соответствующие установке каждого бита в байте, представлены в таблице

Бит	Маска
0	0x01
1	0x02
2	0x04
3	0x08
4	0x10
5	0x20
6	0x40
7	0x80

Маски

Для установки определенного бита необходимо соответствующий бит маски установить в 1 и произвести операцию побитового логического ИЛИ с константой, представляющей собой маску.

Для сброса определенного бита необходимо соответствующий бит маски сбросить в 0 и произвести операцию побитового логического И с константой, представляющей собой инверсную маску.

Маски

```c
#include <stdio.h>

int main() {
    unsigned char a = 3;  unsigned char b = 3;
    a = a | 0x04;  // a = 7, бит 2 установлен
    b = b & (~0x02);  // b = 1, бит 1 сброшен
    printf("a=%d\t b=%d\n",a,b);
   return 0;
}
```

```
a=7      b=1

Process returned 0 (0x0)   execution time : 0.094 s
Press any key to continue.
```