Составление разветвляющихся алгоритмов

Операция «,» (запятая)

Данная операция используется при организации строго гарантированной последовательности вычислений (обычно используется там, где по синтаксису допустима только одна операция, например, в операторах цикла). Форма записи:

***Выражение*_1, … , *Выражение_N*;**

Выражения 1, 2, …, *N* вычисляются последовательно друг за другом до последнего *Выражения_N*,

Пример 1:

***i* = 1, *j* = *i* + 1, *k* = 6, *n* = *i* + *j* + *k* ;**

получим последовательность вычислений, принятых как один оператор.

Пример 2:

***Операнд* = *Выражение*_1, … , *Выражение_N*;**

Используя «запятую» в операции присваивания, нужно помнить, что операция присваивания выполняется справа на лево и, не смотря на то, что ВСЕ выражения БУДУТ выполнены, результатом будет *Выражение*_1.

Пример ошибки. Вычислить

$$t = \frac{2\cos(x - \pi/6)}{0,5 + \sin y}$$

Обозначив числитель ***a***, знаменатель ***b*** запишем:

a = 2 * cos (*x* – M_PI / 6);

b = 0,5 + sin (*y*);

t = *a* / *b*;

В последней операции будет ошибка «Деление на ноль», т.к. в *b* = 0,5 + sin (*y*);

*Выражение*_1 = 0, *Выражение*_2 = 5 + sin (*y*).

В результате *b* = 0;

Операции сравнения

В языке Си используются следующие операции сравнения (отношения между объектами):

== Равно; **!=** Не равно;

< Меньше; **<=** Меньше либо равно;

> Больше; **>=** Больше либо равно.

Пары символов разделять нельзя.

Общий вид операций отношений:

Операнд_1 Знак_Операции Операнд_2

Операндами могут быть любые арифметические выражения и указатели.

Арифметические выражения перед сравнением вычисляются и преобразуются к одному типу.

Результат операции имеет значение **1** (***true***), если отношение ***Истинно***, или получено ***не нулевое значение***, в противном случае **0**, т.е. ***Ложно*** (***false***).

В языке Си нет логического типа данных, в C++ это тип ***bool*** – значения ***true*** (1) ***false*** (0).

Операции сравнения на равенство и неравенство имеют меньший приоритет, чем остальные операции отношений.

Арифметические операции имеют больший приоритет над операциями сравнения.

Примеры. Какие будут результаты в следующих выражениях:

```
int  a = 2, b = 5, x;
    1)  x = a % b == 0;
    2)  x = a * b > 3;
    3)  x = (a != 0) + (b == 5);
```

Логические операции

Логические операции в порядке убывания приоритета:

! – Логическое «НЕ», отрицание (инверсия);

&& – Логическое «И», конъюнкция (умножение);

|| – Логическое «ИЛИ», дизъюнкция (сложение).

Операндами логических операций могут быть любые скалярные объекты. ***Ненулевое*** значение операнда ***Истинно*** **(1)**, ***нулевое*** – ***Ложь*** **(0)**. Результат логической операции 1, или 0.

Общий вид ***унарной*** операции ***отрицания***, которая изменяет значение ***Операнда*** на противоположный:

! *Операнд*

Примеры (! *Истина* – *Ложь*):

1) ! 0 → 1

2) ! 5 → 0

Общий вид ***бинарных*** операций ***И*** и ***ИЛИ***

*Выражение*_1 ***Операция*** *Выражение*_2

1. В операции ***И*** (&&) результат **1** только если ***Истинны*** оба *Выражения*, иначе – 0 (умножение).

2. В операции ***ИЛИ*** (||) результат **0** только если ***Ложны*** оба *Выражения*, иначе – 1 (сложение).

Особенность операций ***И*** и ***ИЛИ*** – экономное последовательное вычисление *Выражений-Операндов*:

1) если *Выражение*_1 операции ***И*** ***Ложно***, то результат операции – **0** и *Выражение*_2 не вычисляется;

2) если *Выражение*_1 операции ***ИЛИ*** ***Истинно***, то результат операции – **1** и *Выражение*_2 не вычисляется.

Логические операции «И» и «ИЛИ» имеют меньший приоритет, чем операции сравнения.

Примеры:

1) $y > 0$ && $x == 7$ → **1**, если оба выражения ***Истинны***;

2) $y > 0$ || $x == 7$ → **1**, если хотя бы одно выражение ***Истинно***.

3) чтобы записать $x \in [a, b]$ $(a \le x \le b)$:

```
x >= a && x <= b
```

4) чтобы записать $x \in [a, b]$ или $x \in [c, d]$:

```
(x >= a && x <= b) || (x >= c && x <= d)
```

***Побитовые логические операции* (~, &, |, ^) и *операции над битами* (>>, <<)**

В языке Си предусмотрен набор операций для работы с отдельными битами. Побитовые логические операции нельзя применять к переменным вещественного типа.

Обозначения операций:

~ Дополнение (унарная операция);

& Побитовое «И»;

| Побитовое включающее «ИЛИ»;

^ Побитовое исключающее «ИЛИ» – сложение по модулю 2;

>> Сдвиг вправо;

<< Сдвиг влево.

Условный оператор

Условный оператор ***if*** (***если***) используется для разветвления процесса на ***два*** направления.

Условный оператор может быть простым и полным.

Форма ***простого*** оператора:

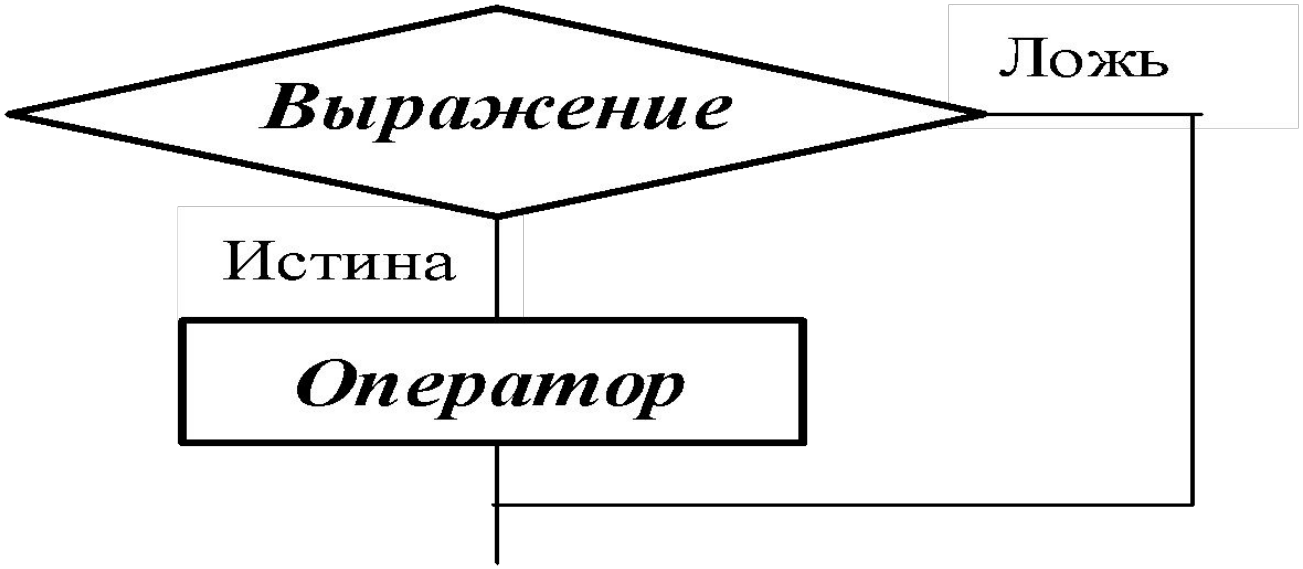
if (***Выражение***) ***Оператор***;

Выражение – логическое или арифметическое выражение, вычисляемое перед проверкой.

Если ***Выражение*** истинно (не равно **0**), то выполняется ***Оператор***, иначе он игнорируется.

Оператор – простой или составной (***блок***) оператор.

Блок создаем, если необходимо выполнить более одного оператора (заключаем их в фигурные скобки).



Структурная схема простого оператора ***if***

Примеры записи:

1) if (x > 0) y = 0;

2) if (i != 1) { – Создаем ***Блок***
   j++; s = 1;
   }

3) if (i != 1) j++, s = 1; – Используем ***Запятую***

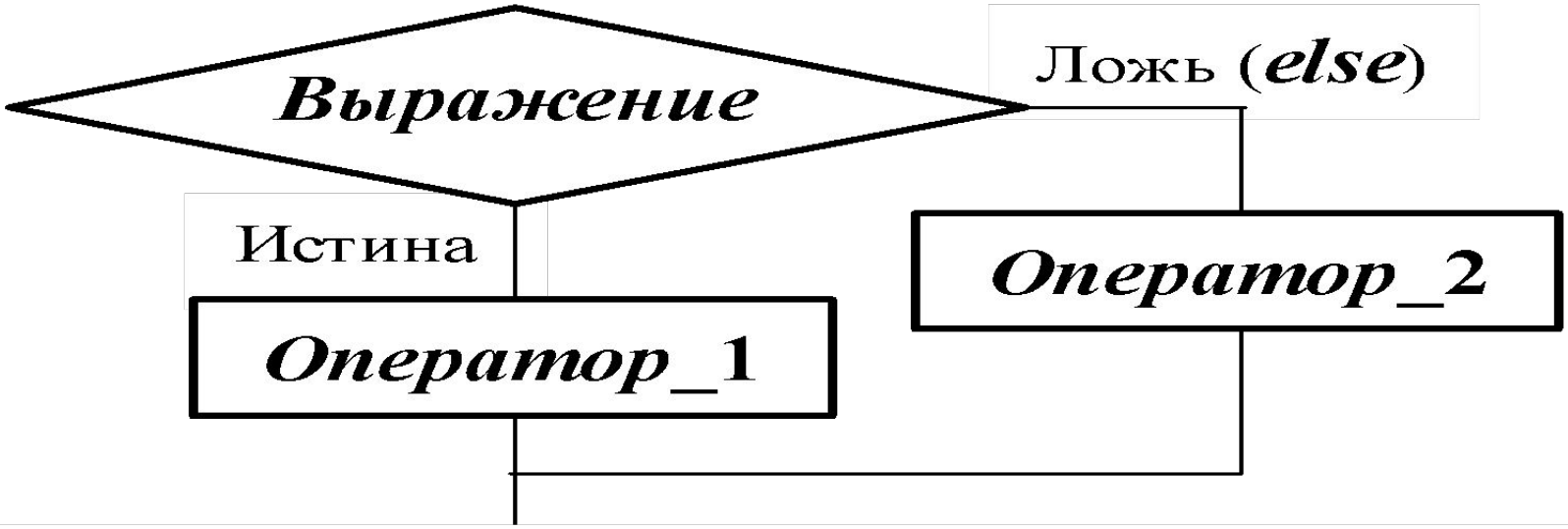
Форма ***полного*** (составного) оператора ***if – else***:

if (***Выражение***) ***Оператор_1*** ;

else ***Оператор_2*** ;

Если ***Выражение*** ***Истинно***, то выполняется ***Оператор_1***, иначе (***else***) – ***Оператор_2***. Операторы 1, 2 могут быть простыми или составными (блоками).

Символ **«;»** перед словом ***else*** в языке Си обязателен.



Структурная схема полного оператора ***if***

Операторами 1, 2 могут быть любые, в том числе и условные. Если есть вложенные операторы ***if-else***, то текущий ***else*** принадлежит ближайшему ***if***, не имеющему ***else***. Например:

```
if (n > 0)
    if(a > b)  z = a;
    else   z = b;
```

Здесь ***else*** связан со вторым *if* (a > b).

Если нужно связать ***else*** с внешним ***if***, то используем операторные скобки (блок):

```
if (n > 0)  {
if (a > b)  z = a;
}
else   z = b;
```

Обратите внимание, что после заголовка оператора if (Выражение) «;» не ставится!!!

В следующей цепочке операторов ***if-else-if*** *Выражения* 1,2,3 просматриваются последовательно:

```
if (Выражение_1)  Оператор_1;
else
if (Выражение_2)  Оператор_2;
    else
    if (Выражение_3)  Оператор_3;
        else   Оператор_4 ;
```

Если какое-то ***Выражение*** оказывается истинным, то выполняется относящийся к нему ***Оператор*** и этим вся цепочка заканчивается.

Оператор_4 будет выполнен только, если все ***Выражения*** (1, 2, 3) ***Ложны***.

Замечание* 1**. Наиболее распространенной ошибкой является использование в ***Выражении операции присваивания «=» вместо операции сравнения на равенство «==» (два знака равно). Например, в следующем операторе синтаксической ошибки нет

***if* (*x* = 5) *a*++;**

но значение ***a*** будет увеличено на единицу независимо от значения переменной ***x***, т.к. результатом операции присваивания *x* = 5 в круглых скобках является значение

5 ≠ 0 – ***Истина***.

Замечание* 2**. Вычисления по равенству вещественных величин рекомендуются задавать в условии ***else.

Пример 1:

```
if (x < 0) cout << " X отрицательное " << endl;
    else
    if(x > 0) cout << " X положительное " << endl;
else
        cout << " X равно нулю "  << endl;
```

Пример 2: вычислить

$$x=\begin{cases} z+1; & z \geq 1; \\ z^2; & z<1. \end{cases}$$

Более корректной для ***double x, z***; будет следующая запись

```
    if (z < 1) x = z * z;
        else
        x = z + 1;
```

Условная операция «?:»

Условная операция – ***тернарная***, т.к. в ней участвуют три операнда. Ее форма:

*Выражение*_1 **?** *Выражение*_2 **:** *Выражение*_3

если *Выражение*_1 ***Истинно***, то результат операции *Выражение*_2, иначе – *Выражение*_3. Каждый раз вычисляется только одно из *Выражений* 2 или 3.

Схема работы аналогична схеме полного оператора ***if*** :

Пример 1: найти максимальное между ***a*** и ***b***.

1. Используя оператор ***if*** :

```
if (a > b)   max = a;
    else  max = b;
```

2. Используя условную операцию:

```
max = (a > b) ? a : b;
```

Условную операцию можно использовать также как и любое другое выражение.

Пример 2: найти y = max(a, b) / max(c, d);

```
y = (a > b ? a : b) / (c > d ? c : d);
```

Оператор безусловного перехода

goto Метка ; (перейти на …)

передает управление оператору с указанной ***Меткой***. ***Метка*** – идентификатор, записанный по синтаксису языка Си с символом «двоеточие» после него. Например, пустой оператор «**;**» с меткой ***m*1**

***m*1 : ;**

Область действия ***Метки*** – функция (блок), где она определена.

Имена меток не декларируются.

Операторы (циклы, ***switch***) могут быть вложены вдруг в друга и ***наиболее оправданный случай*** использования оператора ***goto*** – выход во вложенной структуре.

Например, выход из двух (или более) вложенных операторов при возникновении грубых ошибок:

```
Оператор 1 (...) {
    Оператор 2(...) {
        ...
                    if (Ошибка)  goto error;
    }
                ...
}
    error :        - Устранение ошибки
```

Второй случай: переход из нескольких мест функции в одно, например, когда перед завершением работы функции необходимо сделать одни и те же действия.

Оператор альтернативного выбора

Составной оператор ***switch*** (переключатель) предназначен для разветвления процесса вычислений на несколько направлений (**три и более**).

Общий вид:

switch (***Выражение***) { - Заголовок оператора

case Константа_1: *Список_1 Операторов*

case Константа_2: *Список_2*

...

case Константа_N : *Список_N*

default : *Список_N*+1 – Необязательная ветвь

}

Типы ***Выражения*** и ***Констант*** должны совпадать и могут быть только ***int*** (целый) или ***char*** (символьный).

Сначала вычисляется ***Выражение***, затем оно сравнивается с ***Константами***, которые фактически выполняют роль меток.

Если значение ***Выражения*** совпало с одной из ***Констант***, то выполняется соответствующий ***Список Операторов***. Далее, если выход (оператор ***break***) из ***switch*** в данной ветви не указан, последовательно выполняются все остальные ветви (до конца, или до первого ***break***).

Константы должны иметь разные значения одного типа. Несколько меток могут следовать подряд, тогда переход в указанную ветвь будет происходить при совпадении хотя бы одной из них. Порядок следования ветвей не важен.

Если ***Выражение*** не совпало ни с одной из ***Констант***, то выполняется ветвь ***default***, а при ее отсутствии – оператор, следующий за ***switch***.

Оператор ***break*** (разрыв, прерывание) выполняет досрочный выход из оператора ***switch*** .

Если по совпадению с каждой ***Константой*** должна быть выполнена только одна ветвь, в конце каждой из них используем ***break*.**

Пример ***switch*** с использованием оператора ***break***:

```
        . . .
int i = 2;
    switch ( i ) {
        case  1:   cout << " Case 1 " << endl;      break;
        case  2:   cout << " Case 2 " << endl;      break;
        case  3:   cout << " Case 3 " << endl;      break;
        default :   cout << " Default " << endl;    break;
     }
     cout << " End switch" << endl;
        . . .
```

Результат:

```
    Case 2
    End switch
```

Пример ***switch*** без ***break***:

```
        . . .
int i = 2;
    switch ( i ) {
            case  1:   cout << " Case 1 " << endl;
            case  2:   cout << " Case 2 " << endl;
            case  3:   cout << " Case 3 " << endl;
            default :   cout << " Default " << endl;
        }
    cout << " End switch " << endl;
        . . .
```

Результат в данном случае:

```
    Case 2
    Case 3
    Default
    End switch
```

Пример выбора и расчета функции ***f*(*x*): 2*x*, x^2, *x*/3** :

```
                    . . .
    int  kod;
    double f, x;
    cout << " 1 – 2x (Default)\n  2 – x*x \n  3 – x / 3 " << endl;
cin >> kod;
switch ( kod )  {
    case  1 :     default :
        f = 2 * x;   cout << " f = 2x ( Default ! ) " << endl;
    break;
    case  2 :
        f = x * x;   cout << " f = x*x  " << endl;
    break;
    case  3 :
        f = x / 3;   cout << " f = x/3 " << endl;
    break;
}
                    . . .
```

Можно сделать проще :

```
                    . . .
    int   code;
                    . . .
    cout << " 1 – 2x \n  2 – x*x \n  ELSE – x / 3 " << endl;
cin >> code;
switch ( code)  {
    case  1 :
        f = 2 * x;   cout << " f = 2x " << endl;
    break;
    case  2 :
        f = x * x;   cout << " f = x*x  " << endl;
    break;
    default :
        f = x / 3;   cout << " f = x/3 ( Default ! ) " << endl;
    break;
}
                    . . .
```

Можно сделать иначе и без дополнительной переменной ***code***:

```
                . . .

    cout << " Press 1 – 2x \t  2 – x*x \t  ELSE – x / 3 " << endl;

switch ( getch() )  {          // Или   _getch();
    case  '1' :         // Символьные константы
        f = 2 * x;   cout << " f = 2x " << endl;
    break;
    case  '2' :
        f = x * x;   cout << " f = x*x  " << endl;
    break;
    default :
        f = x / 3;   cout << " f = x/3 ( Default ! ) " << endl;
    break;
}
              . . .
```

В этом случае при нажатии **ЛЮБОЙ** клавиши будет **default** !!!