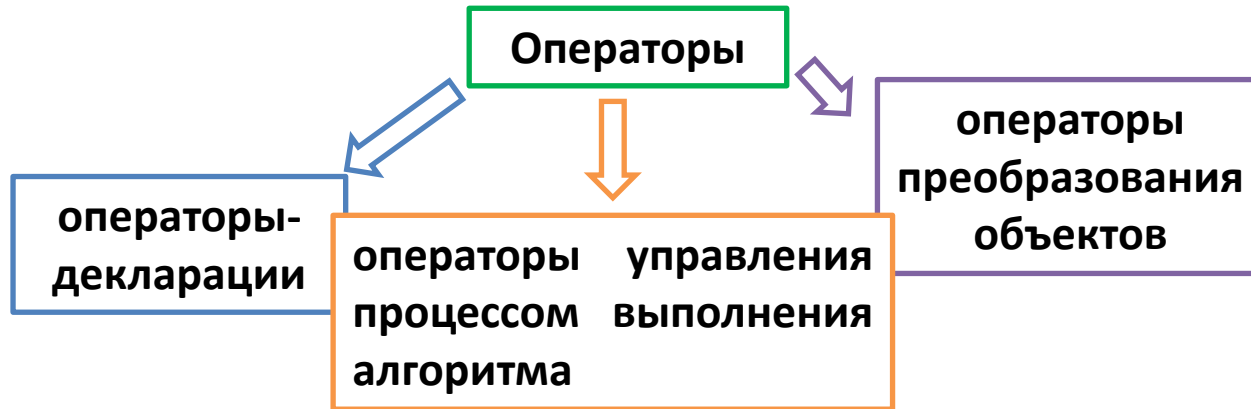


Основы алгоритмизации и программирования

Лекция 6

Составление разветвляющихся алгоритмов

Операторы языка Си



Программирование процесса преобразования объектов производится **посредством записи операторов (инструкций)**

Простейший вид операторов – **выражение**, заканчивающееся символом «**;**» (точка с запятой). Выполнение такого оператора заключается в вычислении некоторого выражения

К **управляющим операторам** относятся: операторы **условного** и **безусловного переходов**, оператор **выбора альтернатив (переключатель)**, операторы **организации циклов** и **передачи управления (перехода)**.

Операторы языка Си записываются в свободном формате с использованием разделителей между ключевыми словами. Любой оператор может помечаться меткой – идентификатор и символ «**:**» (двоеточие). Область действия метки – функция, где эта метка определена

Условный оператор if

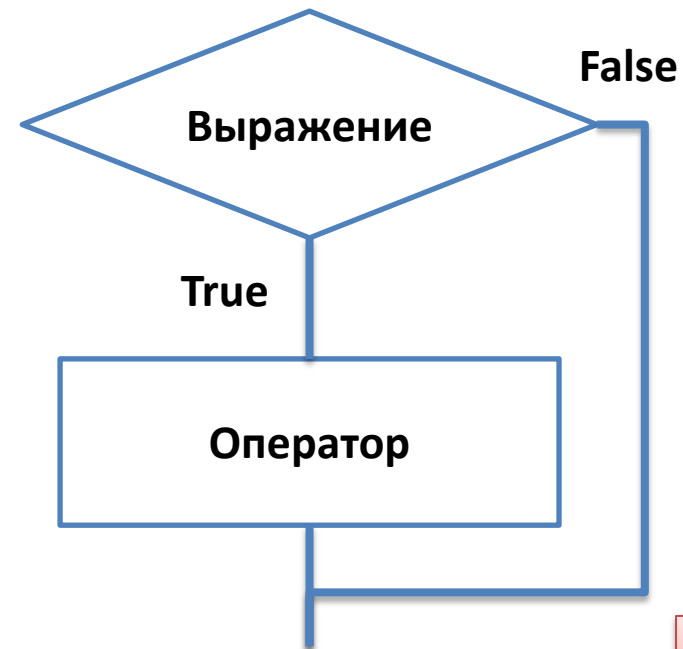
Условный оператор **if** используется для разветвления процесса выполнения кода программы на **два направления**.

1. Простой условный оператор

if (выражение) оператор;

выражение – логическое или арифметическое выражение, вычисляемое перед проверкой, и, если выражение **истинно**, то **выполняется оператор**, иначе он игнорируется;

оператор – простой или составной (блок) оператор языка Си. Если в случае истинности выражения необходимо выполнить **несколько операторов** (более одного), их необходимо заключить в **фигурные скобки**.



Пример

```
if (x > 0) x = 0;  
if (i != 1) j++, s = 1; – операция «запятая»;  
if (i != 1) {  
    j++; s = 1;  
}  
if (i>0 && i<n) k++;  
if (a++) b++;
```

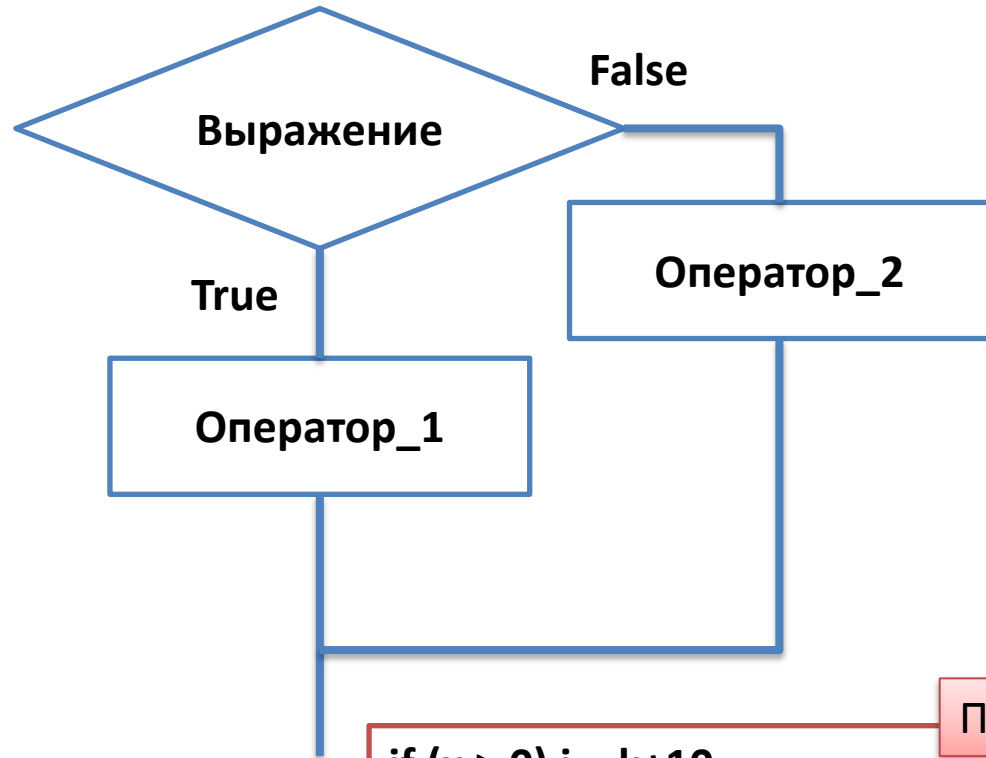
Условный оператор if

2. Полный условный оператор

```
if (выражение) оператор 1 ;  
else оператор 2 ;
```

Если **выражение** не равно нулю (**истина**), то выполняется **оператор 1**, иначе – **оператор 2**. Операторы 1 и 2 могут быть простыми или составными (блоками).

Наличие символа «**;**» перед словом **else** в языке Си обязательно.



Пример

```
if (x > 0) j = k+10;  
    else m = i+10;
```


```
if ( x>0 && k!=0 ) {  
    j = x/k;  
    x += 10;  
}  
else m = k*i + 10;
```

Условный оператор if

Операторы 1 и 2 могут быть **любыми операторами**, в том числе и **условными**. Тогда, если есть вложенная последовательность операторов **if – else**, то фраза **else** связывается с ближайшим к ней предыдущим **if**, не содержащим ветвь **else**.

Если же необходимо **связать фразу else** с внешним **if**, то используются операторные скобки:

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



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

Здесь ветвь **else** связана со вторым **if (a > b)**

В следующей цепочке операторов выражения просматриваются последовательно:

```
if (выражение 1) оператор 1;  
else  
    if (выражение 2) оператор 2;  
    else  
        if (выражение 3) оператор 3;  
        else оператор 4 ;
```

Условный оператор if

В следующей цепочке операторов выражения просматриваются последовательно:

```
if (выражение 1) оператор 1;  
else  
    if (выражение 2) оператор 2;  
    else  
        if (выражение 3) оператор 3;  
        else оператор 4 ;
```

Пример

```
if (x < 0) printf("\n X<0 \n");  
else if(x==0) printf ("\n X =0 \n");  
else printf("\n X >0 \n");
```

Распространенной ошибкой при создании условных операторов является использование в выражении операции присваивания «=» вместо операции сравнения «==». Здесь синтаксической ошибки нет:

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

но значение **a** будет увеличено на единицу независимо от значения переменной **x**, т.к. **x = 5** является – **true**

Если какое-то выражение оказывается **истинным**, то **выполняется** относящийся к нему **оператор** и этим вся цепочка заканчивается. Каждый оператор может быть либо отдельным оператором, либо группой операторов в фигурных скобках. **Оператор 4** будет **выполняться** только тогда, когда **ни одно из** проверяемых **условий не подходит**. Иногда при этом не нужно предпринимать никаких явных действий, тогда последний **else** может быть опущен или его можно использовать для контроля, чтобы зафиксировать «невозможное» условие (**экономия на проверке условий**).

Условная операция «? :»

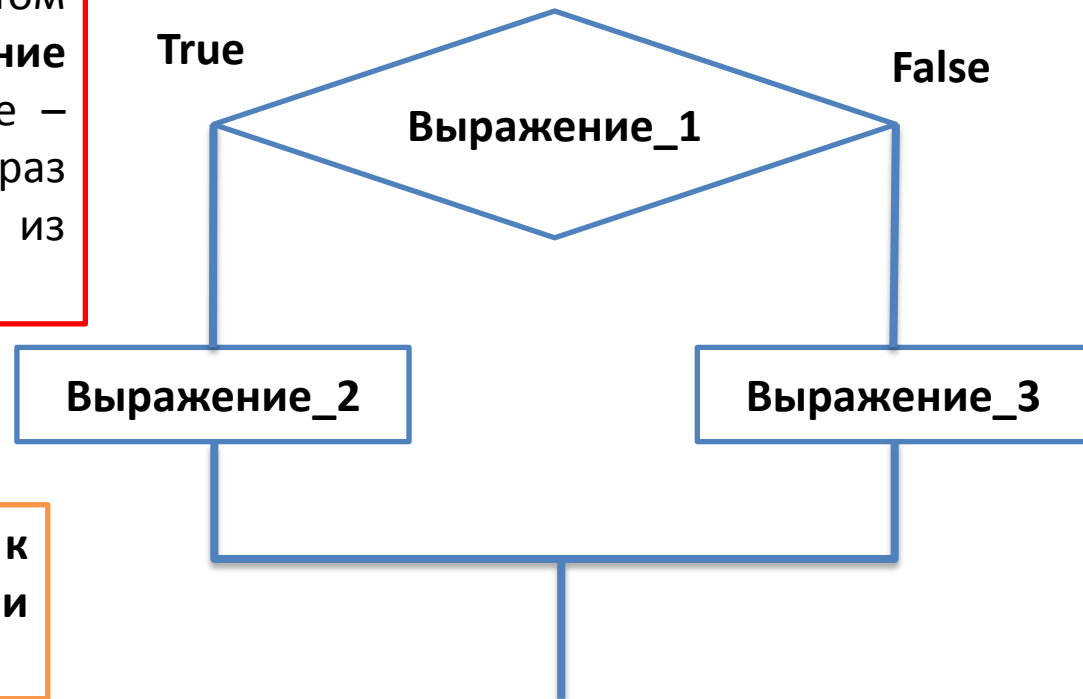
Условная операция – *тернарная*, т.к. в ней участвуют три операнда

Формат условной операции

Выражение 1 ? выражение 2 : выражение 3;

Если **выражение 1** (условие) **отлично от нуля (истинно)**, то результатом операции является **значение выражения 2**, в противном случае – **значение выражения 3**. Каждый раз вычисляется только одно из выражений **2** или **3**.

Условное вычисление применимо к арифметическим операндам и операндам-указателям



Условная операция «? :»

Рассмотрим участок программы для нахождения максимального значения z из двух чисел a и b , используя оператор **if** и **условную операцию**.

1. При помощи **if** :

```
if ( $a > b$ )  $z = a$ ;  
else  $z = b$ ;
```

2. Используя **условную операцию**:

```
 $z = (a > b) ? a : b$ ;
```

Условную операцию можно использовать так же, как и **любое другое выражение**. Если **выражения 2 и 3** имеют **разные типы**, то **тип результата** определяется по **правилам преобразования**. Например, если f имеет тип **double**, а $n - int$, то результатом операции $(n > 0) ? f : n$; по правилам преобразования типов будет **double**, независимо от того, положительно n или нет.

Использование **условных выражений** позволяет во многих случаях значительно упростить программу. Например:

```
int a, x;  
...  
 $x = (a < 0) ? -a : a$ ;  
printf("\n Значение %d  %s нулевое !", x, (x ? "не" : " "));
```


Оператор выбора альтернатив

Оператор **switch** (переключатель) предназначен для разветвления процесса вычислений на несколько направлений.

Общий вид оператора

```
switch ( выражение ) {  
  case константа1:           список операторов 1  
  case константа2:           список операторов 2  
                               ...  
  case константаN:           список операторов N  
  default: список операторов N+1 – необязательная ветвь;  
}
```

Выполнение оператора начинается с вычисления **выражения**, значение которого должно быть **целого или символьного типа**. Это значение **сравнивается** со значениями **констант** и используется для **выбора ветви**, которую нужно выполнить

Оператор выбора альтернатив

В данной конструкции **константы** фактически **выполняют роль меток**. Если значение выражения **совпало** с одной из перечисленных **констант**, то **управление передается** в **соответствующую ветвь**. После этого, **если выход** из переключателя в данной ветви **явно не указан**, последовательно **выполняются все остальные ветви**

Все **константы** должны иметь **разные значения**, но быть **одного** и того же **типа**. Несколько меток могут следовать подряд, и тогда переход в указанную ветвь будет происходить при совпадении хотя бы одной из них. Порядок следования ветвей не регламентируется.

В случае **несовпадения** значения выражения **ни с одной из констант выбора** происходит **переход** на метку **default** либо, при ее отсутствии, к оператору, следующему за оператором **switch**

Оператор выбора альтернатив

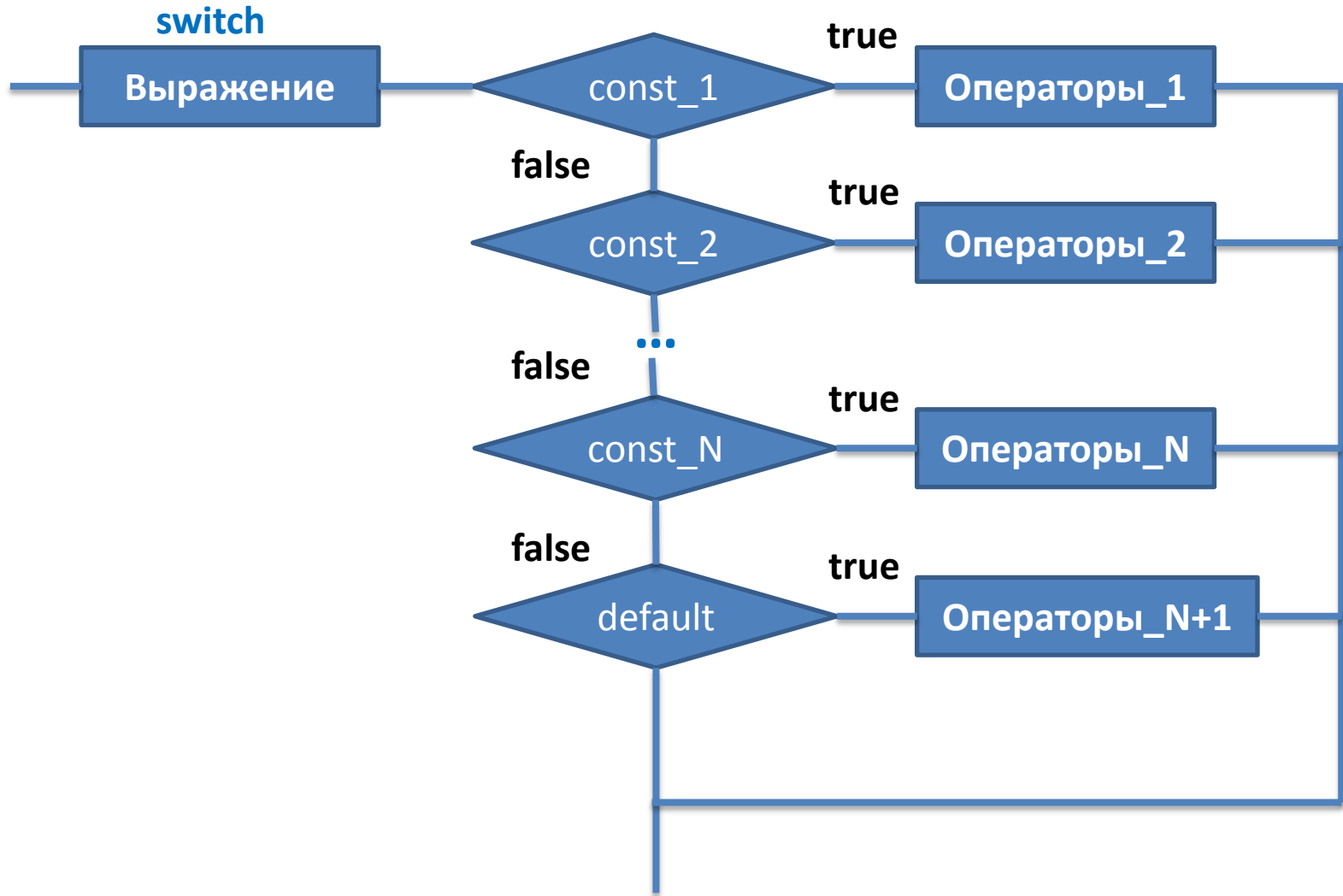
Управляющий оператор **break** (разрыв) выполняет выход из оператора **switch**. Если по совпадению с каждой константой должна быть выполнена одна и только одна ветвь, схема оператора **switch** следующая:

```
switch (выражение) {  
    case константа1: операторы 1; break;  
    case константа2: операторы 2; break;  
    ...  
    case константаN: операторы N; break;  
    default: операторы (N+1); break;  
}
```

```
void main(void)    {  
    int i = 2;  
  
    switch(i)    {  
        case 1: puts ( "Случай 1. "); break;  
        case 2: puts ( "Случай 2. "); break;  
        case 3: puts ( "Случай 3. "); break;  
        default: puts ( "Случай default. "); break;  
    }  
}
```

Оператор выбора альтернатив

Структурная схема рассмотренной конструкции (с использованием оператора *break*)



Оператор выбора альтернатив

Пример без использования оператора *break* :

```
void main()      {  
int i = 2;  
  
    switch(i)    {  
        case 1: puts ( "Случай 1. ");  
        case 2: puts ( "Случай 2. ");  
        case 3: puts ( "Случай 3. ");  
        default: puts ( "Случай default. ");  
    }  
}
```

В данном случае результат будет следующим:

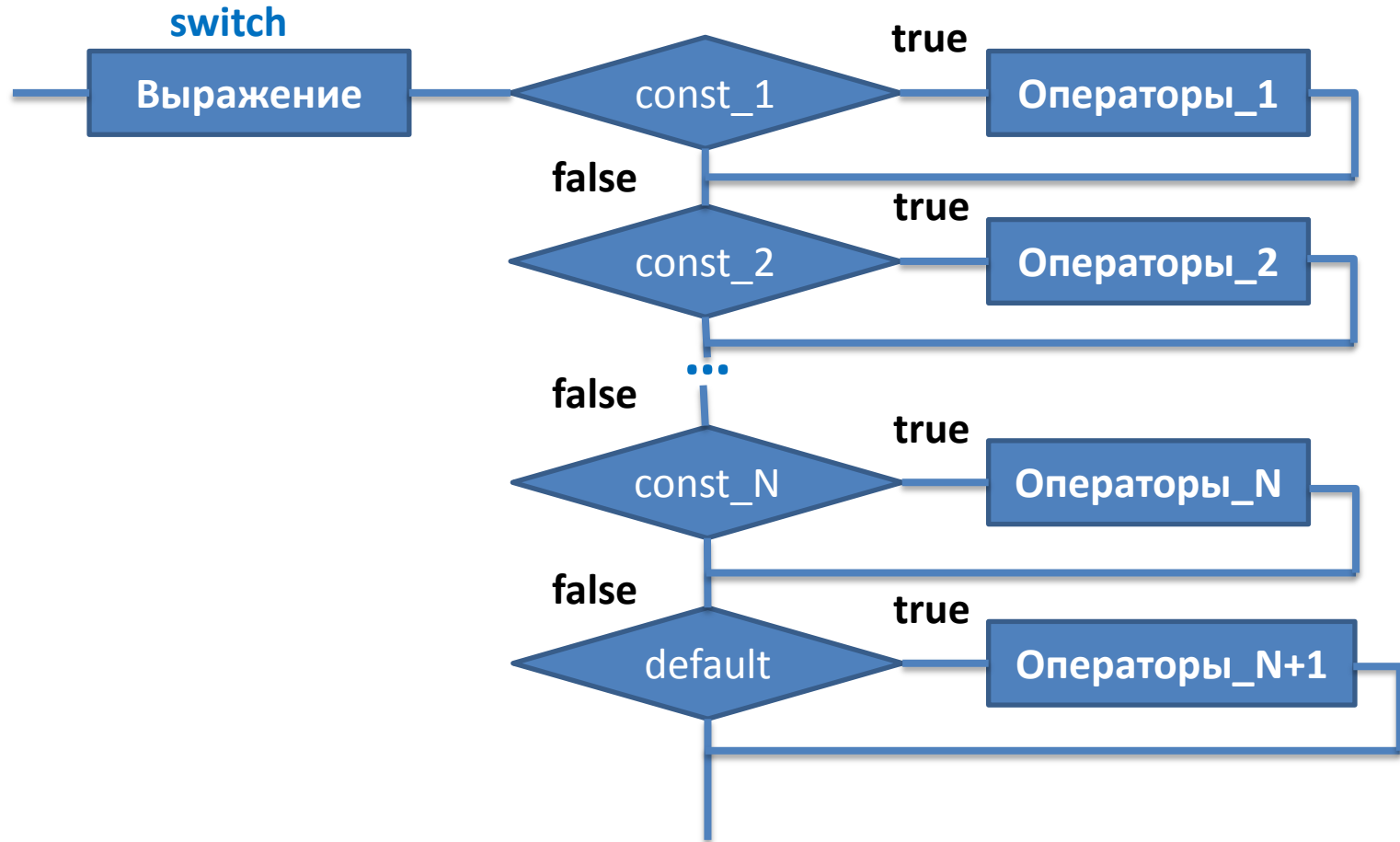
Случай 2.

Случай 3.

Случай default.

Оператор выбора альтернатив

Структурная схема рассмотренной конструкции (без использования оператора *break*)



Оператор выбора альтернатив

Пример

```
#include <stdio.h>
void main(void)
{
    double a, b, c;
    char s;
    m1: fflush(stdin);                // Очистка буфера ввода stdin
    printf("\n Введите операнд 1, символ операции, операнд 2:");
    scanf("%lf%c%lf", &a, &s, &b);
    switch(s) {
        case '+': c = a+b; break;
        case '-': c = a-b; break;
        case '*': c = a*b; break;
        case '/': c = a/b;           break;
        default: printf("\n Ошибка, повторите ввод! "); goto m1;
    }
    printf("\n a %c b = %lf ", s, c);
    printf("\n Продолжим? (Y/y) ");
    s = getch();
    if ( (s=='Y') || (s=='y') ) goto m1;
    printf("\n Good bye! ");
}
```