Лекция 28_09_2021
Основы программирования

Задание : выбрать и нарисовать в конспекте один из 3-х вариантов (слайды 26-28) изображения оператора switch-case в блок-схеме (или свой вариант)

4.2. Унарные

Оператор побитового отрицания – «~».

x	~x
0	1
1	0

Пример 21.

```
int x=6;        /* 0110 */
int y=x&~3; /* 0110&~0011=0110&1100=0100 */
```

5. Операторы присваивания

Простейший оператор присваивания:
<v>=<e>
<v> - переменная
<e> - выражение

Оператор присваивания – бинарный, инфиксный.

Семантика: значение выражения, в которое входит присваивание, является значением левого операнда после присваивания.

Пример 22.

```
b=a;
a=b=c=0; // многократное присваивание
                // (a=(b=(c=0)))
```

Остальные операторы присваивания имеют следующий вид:

<e1><op>=<e2>

<op> - бинарный оператор: +, -, /, *, %, &, |, ^, <<, >>.

Выполнение:

<e1>=<e1><op><e2>

<e1> модифицируется с помощью оператора и <e2> и записывается в <e1>.

Значки операторов присваивания: +=, /=, *=, -=, %=, <<=, >>=, |=, ^=, &=.

Пример 23.

int x=6, y=10;
y=x; //y=60*

```
int z, x8, y8, z8;
x=15;                x8=015;
y=8;               y8=07;
z=5;               z8=05;

x+=2;                /*x=17*/
y-=7;                /*y=1*/
z*=4;                /*z=20*/
z/=2;              /*z=10*/
z%=3;                /*z=1*/
x8<<=2;              /*x8=064*/
y8>>=1;              /*y8=3*/
x8|=03;              /*x8=067*/
x8^=07;              /*x8=060*/
```

!!! Все операторы присваивания правоассоциативные.

x/=a+b x=x/(a+b)

prio(=) < prio(всех ранее рассмотренных операторов).

6. Инкрементные и декрементные операторы

Инкрементный – «++»: добавляет 1 к операнду.

Декрементный – «--»: вычитает 1 из операнда.

Префиксный «++» («--») увеличивает (уменьшает) операнд до его использования.

Постфиксный «++» («--») увеличивает (уменьшает) операнд после его использования.

Пример 24.

int x=2, y, z;
y=x++; *//1) y=x; 2) x=x+1; y=2, x=3*
z=++x; *//1) x=x+1; 2) z=x; x=3, z=3;*

!!! Эти операторы можно применять только к переменным, но не к выражениям и константам.

Пример 25.

y++; --x; //верно
(y+x)++; //неверно
1++; //неверно

7. Условный оператор (?) (условное выражение)

Синтаксис:

<e1>?<e2>:<e3>

Семантика:

1) Вычисляется значение первого выражения **<e1>**

2) Если **<e1>=«истина»,** то значение оператора = **<e2>**

3) Иначе значение оператора = **<e3>**

Пример 26. Присвоить в переменную abs_x модуль переменной x.

abs_x = (x>=0)? x : -x;

7. Преобразование типов

Преобразование (приведение) типов: приведение операндов разных типов к некоторому общему.

Правила приведения операндов к общему типу:

1) **для бинарных операторов**

тип результата определяется типом того операнда, который занимает наибольшее место в памяти:

а) если один из операндов имеет тип long double, то и второй операнд приводится к этому типу;

б) если … double, то … double;

в) если … float, то … float;

г) если … long, то … long;

д) если … short или int, то … int;

Библиотека <math.h> работает с типом данных double.

2) **при присваивании**

значение правой части приводится к типу левой части, это и будет типом результата:

а) char -> int (размножением знакового разряда)

б) int -> short
int -> char
long -> short
long -> int
long -> char

Отбрасыванием старших разрядов

в) float -> int
int -> float

Преобразованием типа: ПТ <-> целое

г) double -> float (округлением/ отбрасыванием)

3) при вызове функции;

4) явное задание приведения типов:

Когда ни одно из вышеуказанных правил не выполняется, то используется оператор <тип>, который задаёт явное преобразование к явному типу данных.

Используется оператор (<имя_типа>)e, где e – выражение.

Свойства оператора явного приведения типа:

- унарный;
- префиксный;
- правоассоциативный.

Пример 27.

```
int x=16;
…
double y=sqrt((double)x);
```

На самом деле по правилам вызова функций целочисленный аргумент x автоматически будет приведен к типу данных, который требует аргумент функции sqrt.

Сводная таблица приоритетов операторов

Операторы	Ассоциативность
() [] . ->	Л
! ~ ++ -- sizeof + - * & (унарные)	П
* / % (бинарные)	Л
+ - (бинарные)	Л
<< >>	Л
< <= >= >	Л
== !=	Л

Операторы	Ассоциативность
& (бинарный)	Л
^ (исключающее или)	Л
| (побитовое сложение)	Л
&&	Л
||	Л
?	Л
= += -= *= /= %= &= != ^= <<= >>=	П
, (запятая) – последовательность	Л

В таблице:

- операторы в одной строке с одинаковым приоритетом;
- строки – по убыванию приоритетов;
- ассоциативность:

1) Л – левоассоциативный (выполнение слева направо);
2) П – правоассоциативный (выполнение справа налево);

УПРАВЛЕНИЕ В Си

Управление – определенный порядок выполнения вычислений в программе.

1. Инструкции и блоки

Если в конце выражения поставить «;», то выражение становится инструкцией.

<инструкция>::=<выражение>;

Пример 28:

```
x=0       j++              /* выражения  */
x=0;      j++;             /* инструкции  */
```

Составная инструкция (блок) – это последовательность описаний (деклараций) и инструкций.

Заключаются внутрь { }.

<блок>::= {
[<декларация>]
<инструкция1>;
.....
<инструкция n> ;
}

Семантически блок воспринимается как единая инструкция.

2. Инструкция if – else

Синтаксис:

if(<выражение>) <инструкция1>;
else <инструкция2>;

или

if(<выражение>) <инструкция1>;

Семантика:

Вычисляется значение выражения <выражение>. Если значение выражения истинно, то выполняется <инструкция1>, если ложно – то <инструкция2>.

Пример 29. Вариант 1.

```
int a,b,c;        /* Исходные данные */
int x;            /* Результат */

……..
if (a>b)
    if (a>c)
       x=a;
    else
       x=c;
else
    if(b>c)
       x=b;
    else
       x=c;
```

Пример 30. Вариант 2.

```
int a,b,c,x;
……..
x=a;
if (b>x) x=b;
if (c>x)  x=c;
```

3. Переключатель switch

switch (<выражение>) {
case <константное выражение1>:
<инструкции 1>;
case < константное выражение n>:
<инструкции n>;
default: <инструкции>;
}

Используется для разветвления алгоритма более чем на 2 направления.

Каждая ветвь **case** помечена одной или несколькими целочисленными константами или константными выражениями.

Исполнение всей конструкции **switch** начинается с той ветви **case**, в которой константное выражение совпадает со значением выражения, записанного после слова **switch**.

Если ни одна из констант не подходит, то выполняется ветвь, помеченная **default**.

Ветвь default необязательна, и если ее нет, то ничего не вычисляется.

Ветви **case** и **default** можно размещать в любом порядке.

Поскольку выбор любой из ветвей case выполняется как переход на метку, то после выполнения одной ветви case, программа переходит к выполнению следующей ветви.

Если нужно завершить выполнение оператора **switch**, то ветвь case следует завершить оператором **break**.

Пример 31. Вариант 1.

```
if (j==1 || j==3 || j==5 || j==7 || j==8 || j==10 || j==12)
        kd=31;
else
   if (j==2) kd=28;
   else kd=30;
```

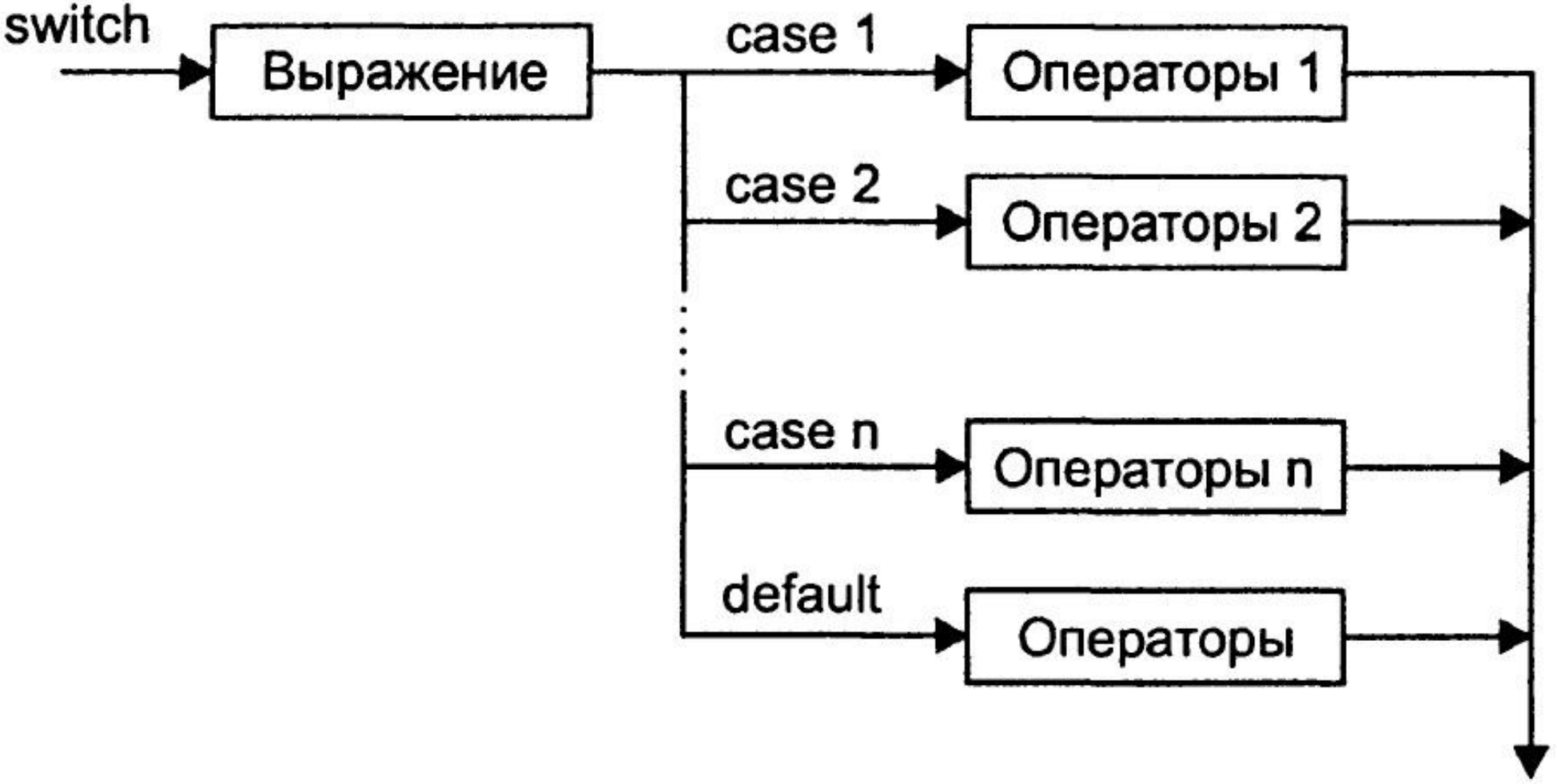
Пример 32. Вариант 2.

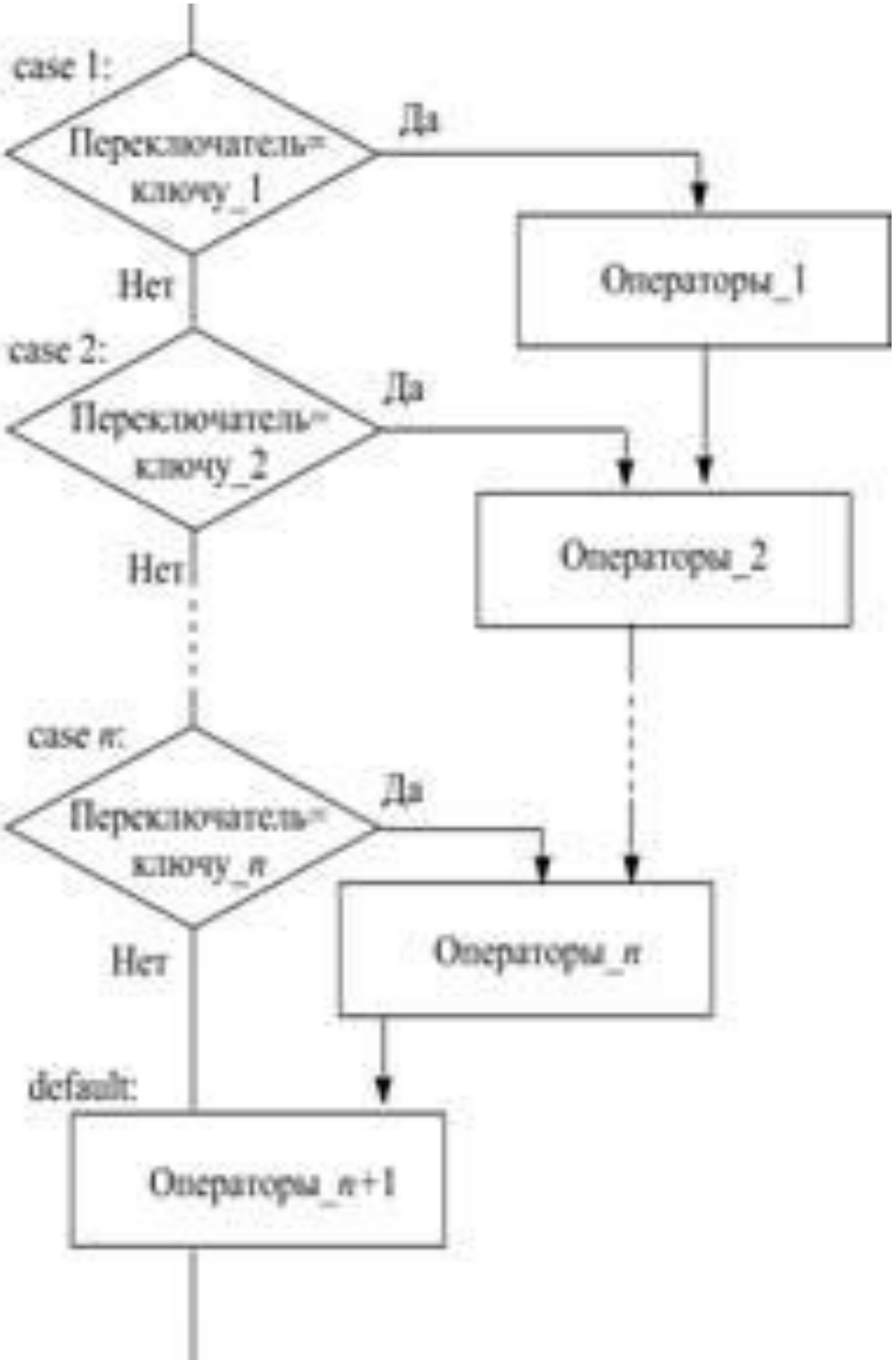
```
switch (j) {
    case 1: kd=31; break;
    case 2: kd=28; break;
    case 3: kd=31; break;
……..……..
    case 12: kd=31; break;
    default: printf("Ошибка"); }
```

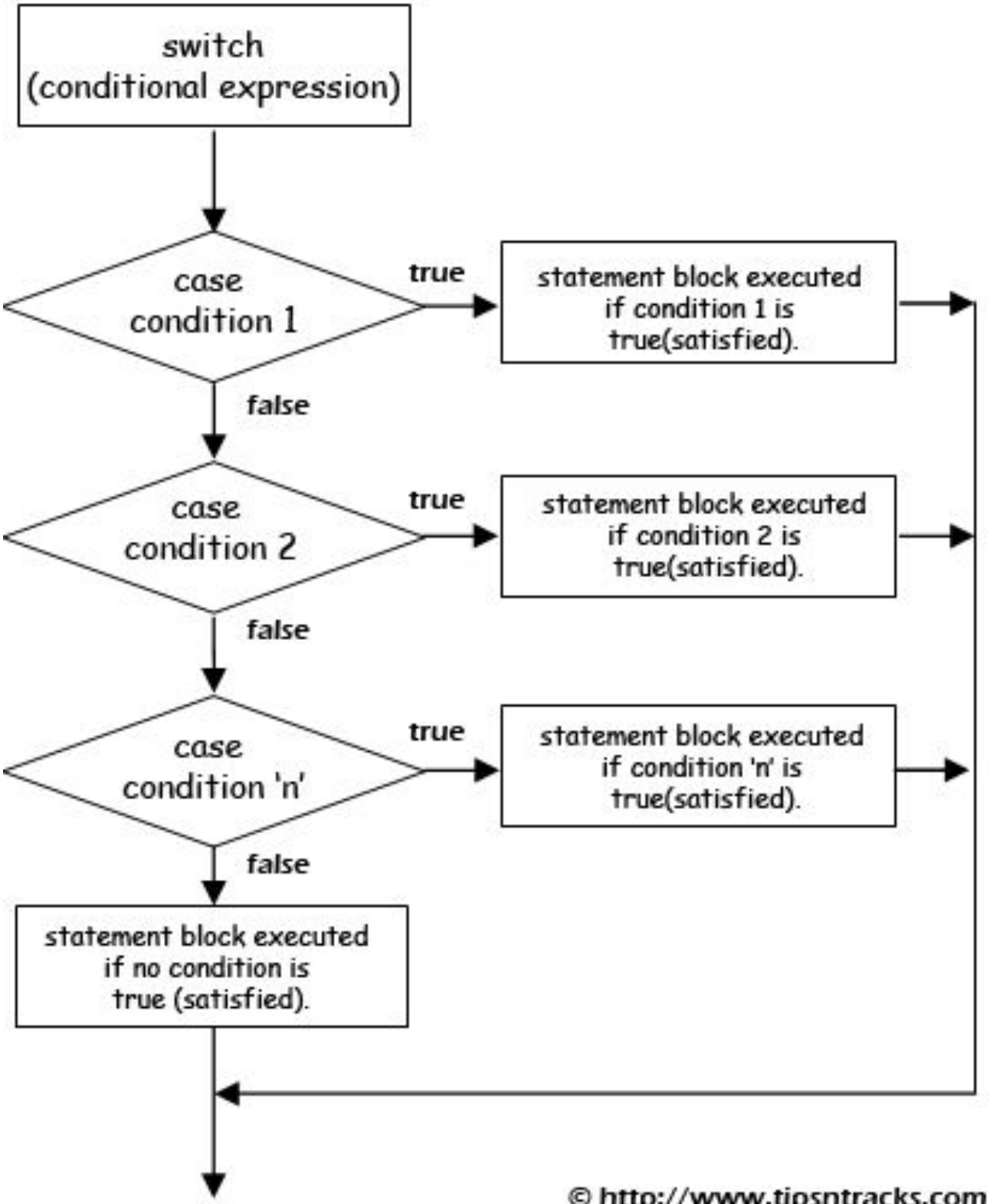
Пример 33. Вариант 3.

```
switch (j)
{
    case 1: case 3: case 5: case 7: case 8:
    case 10: case 12:
            kd=31;
            break;
    case 4: case 6: case 9: case 11:
            kd=30;
            break;
    case 2:
            kd=28;
            break;
    default: printf("Ошибка");
}
```

На блок-схеме оператор switch изображается следующим образом:




case 1:
Переключатель= ключу_1
Да
Операторы_1
Нет
case 2:
Переключатель= ключу_2
Да
Операторы_2
Нет
case n:
Переключатель= ключу_n
Да
Операторы_n
Нет
default:
Операторы_n+1


switch
(conditional expression)
case
condition 1
true
statement block executed
if condition 1 is
true(satisfied).
false
case
condition 2
true
statement block executed
if condition 2 is
true(satisfied).
false
case
condition 'n'
true
statement block executed
if condition 'n' is
true(satisfied).
false
statement block executed
if no condition is
true (satisfied).
© http://www.tipsntracks.com

6. Цикл while

Это цикл с неизвестным количеством повторений, с предусловием.

while (<выражение>)
    <инструкция>;

Вычисляется <выражение>.

Если оно отлично от 0 (истинно), то выполняется <инструкция> и переход на проверку выражения.

Как только <выражение> станет равным 0, цикл while завершается.

Тело цикла может быть блоком.

Пример 34. e в степени x (с точностью eps)

```
float exp,s,eps,x;
int i,n;
…… /* Ввод x */
exp=1; n=1; i=1; S=x;
while (s/n>eps)
{
    exp+=s/n;
    i++;
    n*=i;
    s*=x;
}
```

7. Цикл for

Это цикл с известным числом повторений

for (<выражение 1>; <выражение 2>; <выражение 3>)
    <инструкция>;

Семантика:

```
< выражение 1>;
while(<выражение 2>)
{
   < инструкция>;
   < выражение 3>;
}
```

<выражение 1> - инициализация счётчика цикла и других переменных.

<выражение 2> - условие завершения цикла.

<выражение 3> - изменение значения счётчика цикла (и других переменных).

!!! Любое из этих 3 выражений может отсутствовать, но ‘;’ должна быть обязательно.

Бесконечный цикл:

```
for( ; ;)
{……..}
```

Особенности цикла for в языке Си:

1) переменная цикла и ее предельное значение могут изменяться внутри цикла;

2) по завершению цикла переменная цикла определена;

3) в выражениях 1, 2 и 3 может быть использовано более одного выражения, разделенных оператором ‘,’.

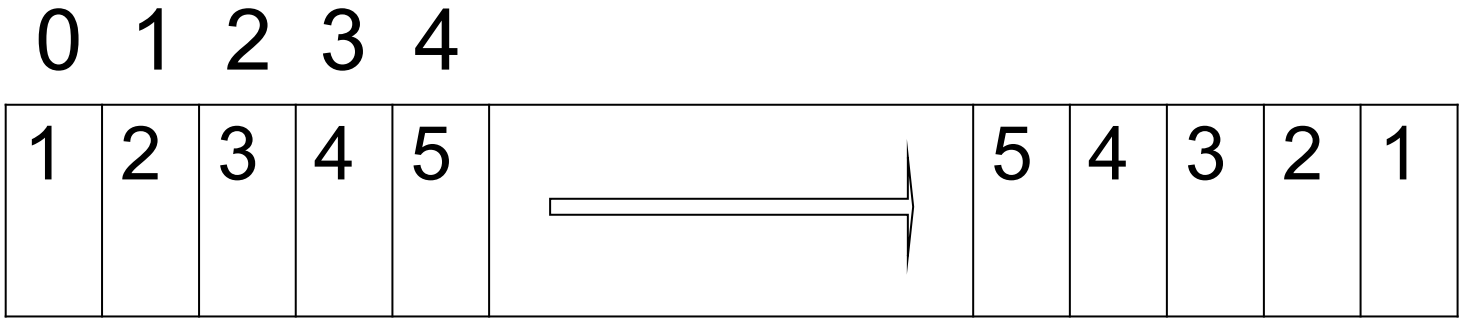
‘,’ – бинарный, инфиксный, левоассоциативный оператор.

Пример 35. Сумма элементов массива

```
#define N 10

float X[N], S;
int i;
……..
/* Вариант 1 */
S=0;
for (i=0; i<N; i++)
     S+=X[i];

/* Вариант 2 */
for (S=0, i=0; i<N; i++)
     S+=X[i];
```

```
/* Вариант 3 */
for (S=0, i=0; i<N; )
    S+=X[i++];

/* Вариант 4 */
i=S=0;
for (; i<N; )
    S+=X[i++];
```

Пример 36. Реверсирование элементов массива




```
#define N 10
……..
float  x[N];
int i, j;
float r;
……..
```

```
for (i=0, j=N-1; i<j; i++, j--)
{
        r=x[i];
        x[i]=x[j];
        x[j]=r;
}
```

8. Цикл do-while

Цикл с неизвестным числом повторений, с постусловием.

do <инструкция>; while (<выражение>);

Семантика.

1. Выполняется <инструкция>.

2. Вычисляется значение <выражения>.

3. Если <выражение> истинно (не равно 0), то снова переходим на пункт 1. Если <выражение> равно 0 (ложно) – конец цикла.

Пример 37. Проверка на корректность ввода

```
#include <stdio.h>
int a, er;
do {
    printf("Введите длину стороны
треугольгика: ");
```

```
    er=scanf("%d",&a);
    if(a<=0 || er==0)
        printf("Ошибка!!! Повторите ввод! \n");
    fflush(stdin);
}
while(a<=0 || er==0);
```

9. Инструкция break и continue

break используется в циклах **while**, **for**, **do while** и в конструкции **switch**.

В циклах использование **break** приводит к немедленному их завершению (до достижения предельного значения переменной цикла **for** или до достижения условия завершения цикла в **while** или **do while**).