Лекция 2

Содержание

1. Блоки операторов и пробелы в написании кода на C++
2. Описание констант на C++.
3. Логическая функция и логические операторы
4. Область видимости имён и переменных
5. Оператора выбора - переключатель.

1. Блоки операторов и пробелы в написании кода на C++

1. Код программы на Си не чувствителен к пробелам: между ключевыми словами операторов можно ставить *любое количество пробелов*:

```
int a=2,     b=3,c=10;
```

Операторы можно записывать в одной строке или разделать по разным – значения не имеет:

```
int a=2,
  b=3,c=10;
```

2. В средах разработки по умолчанию блоки вложенных операторов автоматически отодвигаются от левой границы консоли, но это не является обязательным, как в Python.

```
if(2*a == b || a>b) cout<<a; else  if (b/a>c) cout<< b; else cout << a+b+c <<endl; // можно так
```

```
if(2*a == b || a>b) // или так
cout<<a;
else
  if (b/a>c) cout<< b;
  else cout << a+b+c <<endl;
```

Для улучшения читабельности кода программист сам может использовать пробел или табуляцию.

3. Пробелами и Enter ***нельзя разрывать ключевые слова*** и константы.

Правильно:	Неправильно:
y=sin(x)*atan (z) /log10 (x) *2.e-2; или y=sin(x) *atan(z) /log10 (x) *2.e-2;	y=s in(x)*atan(z) /log 10 (x) *2.e-2; или y=sin(x)*atan(z)/log10 (x) *2. e-2;

Ошибка трансляции

4. При записи строковых констант разрыв строк не допустим

Правильно:

```
16  string a="A very-very-very-very-very long sentence";
17  cout<< a;
18  return 0;
19  }
20
```

input

```
A very-very-very-very-very long sentence

...Program finished with exit code 0
Press ENTER to exit console.
```

Неправильно:

```
16  string a="A very-very-very-very-very
17  long sentence";
18  cout<< a;
19  return 0;
20  }
```

input

Compilation failed due to following error(s).

```
   16 | string a="A very-very-very-very-very
      |          ^
main.cpp:16:10: error: missing terminating " character
```

5. Для объединения операторов в блок используются фигурные скобки **{ }**.

```
bool x=true;
while(x) cout << x; x=x-1; x+=3; cout << x;
while(x) {cout << x; x=x-1; } x+=3;cout << x;
```

```
13  int main()
14  {
15  int x=1;
16  while(x) {cout << x; x--;} x+=3;cout << x;
17
18
19      return 0;
20  }
```

```
13
...Program finished with exit code 0
Press ENTER to exit console.
```

```
13  int main()
14  {
15  bool x=true;
16  while(x) cout << x; x=x-1; x+=3;cout << x;
17
18
19      return 0;
20  }
```

```
11111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111
```

```
13  int main()
14  {
15  bool x=true;
16  while(x) {cout << x; x=x-1; } x+=3;cout << x;
17
18
19      return 0;
20  }
```

```
11
...Program finished with exit code 0
Press ENTER to exit console.
```

Особенности описания констант на C++

1. Определение констант

Константами в программе называются параметры, значения которых не меняются в процессе выполнения программы.

Задание значений констант осуществляется, как правило, в начале программы.

Каждое из определений констант имеет следующий вид:

const <тип> < имя константы > = < значение константы > ;

Примеры:

```
const   int   N = 25;            // константа целого типа
const   double   G = 9.81;    // константы вещественного типа
const double  Avogadro=6.022e23;  // вещественного типа
```

Использование в программе имен констант (вместо указания их численных значений) делает программу более наглядной.

Кроме того, это позволяет программисту сгруппировать в начале программы все постоянные величины: здесь они более заметны и их легче изменить, при необходимости.

7

1.1 Определение констант с помощью суффиксов и префиксов

По умолчанию компилятор присваивает: целочисленной константе тип int (4 байта) вещественному числу тип double (8 байт): y =4+sin(x)*atan(z) /log10(2.e3)-3.;

Кроме естественного представления числовых констант в виде целого или вещественного числа языки программирования допускают различные добавки в начале ("префиксы ") или конце ("суффиксы ") числа, определяющие способы преобразования и хранения данных в памяти компьютера.

- `5U,5u` - **целое число без знака (суффикс -** `u` **или** `U`**, от** `Unsigned` **);**
- `5L,5l` - **длинное целое число (суффикс -** `l` **или** `L`**, от** `Long` **);**
- **НО:**
- `5.L,5.0l` - **длинное вещественное число (**`Long Double`**);**
- `5LU,5lu,5Lu,5lU` - **длинное целое число без знака;**
- `5.0f,5.F` - **короткое вещественное число (суффикс -** `f` **или** `F`**, от** `Float` **);**
- `0x5,0X5` - **шестнадцатеричное целое число (префикс -** `0x` **или** `0X` **);**
- `05` - **восьмеричное целое число (префикс - незначащий нуль в начале); (056=?$_{10}$)**
- **Двоичная система представления данных непосредственно в языке Си не поддерживается.**

8

Целочисленные гонстанты на C++

Целочисленные данные в языке Си могут быть представлены в одной из следующих систем счисления:

Десятичные	Последовательность цифр (`0 – 9`), которая начинаются с цифры, отличной от нуля. Пример: 1, -29, 385. Исключение — число 0.
Восьмеричные	Последовательность цифр (`0 – 7`), которая всегда начинается с `0`. Пример: 00, 071, -052, -03.
Шестнадцатеричные	Последовательность шестнадцатеричных цифр (`0 – 9` и `A – F`), которой предшествует присутствует `0x` или `0X`. Пример: 0x0, 0x1, -0x2AF, 0x17.

По умолчанию целочисленные константы имеют тип int.

БАЗОВЫЕ АЛГОРИТМИЧЕСКИЕ СТРУКТУРЫ, РЕАЛИЗУЕМЫЕ ОПЕРАТОРАМИ С++

АЛГОРИТМ ВЕТВЛЕНИЯ

1. Условная операция.

2. Условный оператор.

- 2.1. Сокращенная форма условного оператора
- 2.2. Полная форма условного оператора
- 2.3. Технология программирования задачи с разветвлением на три направления

3. Оператор выбора.

- 3.1. Общий вид оператора
- 3.2. Пример

1. Условная операция (тернарная операция)

<переменная> = <логическое выражение> ? <выражение1> : <выражение2>;

Если **<логическое выражение>** истинно,
то **<переменная>** получает значение **<выражение1>**
иначе **<переменная>** получает значение **<выражение2>**

Пример 1: записать на языке программирования следующую функцию:

$$y = \begin{cases} a - 2 \cdot b, & если \quad a \cdot b < 5 \\ \sqrt{a \cdot b}, & если \quad a \cdot b \geq 5 \end{cases}$$

<выражение1>, <выражение2> НЕ ДОПУСКАЮТ ПРИМЕНЕНИЕ БЛОКОВ ОПЕРАТОРОВ

```
y = a * b < 5 ? a – 2 * b : sqrt(a * b);
cout<< a * b < 5 ? a – 2 * b : sqrt(a * b)<<endl;
z=3.0*(a * b < 5 ? a – 2 * b : sqrt(a * b));
z=3.0*(a * b < 5 ? { a – 2 * b } : {sqrt(a * b};
```

Допустимо применение условной операции в условной операции

Задача: Составить алгоритм находящий значение y, если

y=x, при x<0;

y=0, при 0<=x<30;

$y=x^2$, при x>=30;

```
int x;
   cout << "Enter x: ";
   cin >> x; // вводим значение икса
   y=x < 0 ? x : (x >= 0) && (x < 30) ? 0 : x * x  ); // две тернарные операции
```

Пример 2: определить наибольшее значение из a и b:

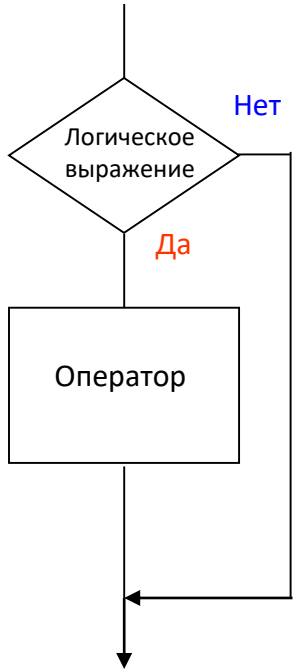
```
int   max = b > a ?   b  :  a;
```

Пример 3: требуется, чтобы некоторая целая величина увеличивалась на 1, если ее значение не превышает n, а иначе принимала значение 1.

```
int   i =  (i <= n)  ?  i + 1  :  1;
```

2. Условный оператор

2.1. Сокращенная форма условного оператора



if (<логическое_выражение>) <оператор> ;

Сначала определяется значение логического выражения.

Если оно равно **true**, то выполняется оператор,

иначе – пропускается (не выполняется).

После этого управление передается на оператор, следующий за условным.

Если в качестве оператора требуется выполнить несколько операторов, то необходимо заключить их в блок, т.е в фигурные скобки {}.

15

Пример 4: вычислить значение функции:

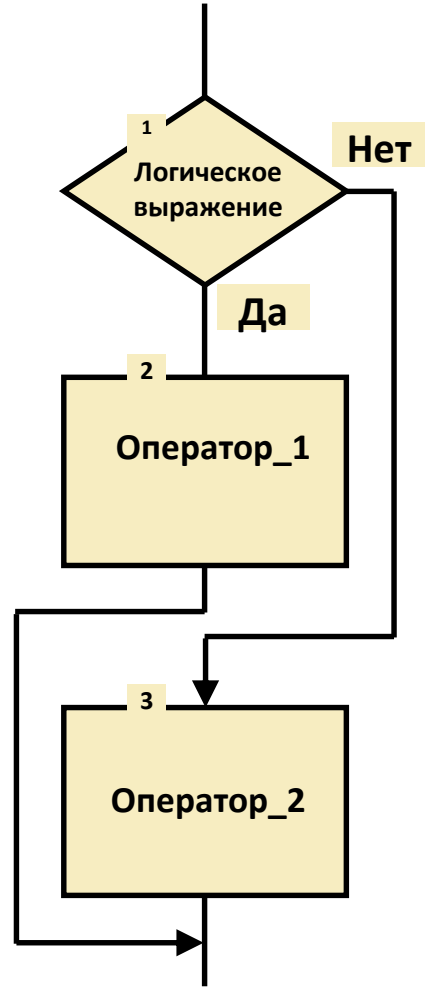
$$y = \begin{cases} a - e^{b/a}, & если \quad a \ кратно \ b \\ \sqrt{a \cdot \sqrt{b}}, & в \ противном \ случае \end{cases}$$

```
if(a % b  ==  0)   y = a – exp(b/a);

if(a % b  !=  0)   y = sqrt(a * sqrt(b));
```

2.2. Полная форма условного оператора

```
if ( <логическое_выражение> ) <оператор_1> ;
else   <оператор_2> ;
```



Здесь else - служебное слово (иначе).

Сначала определяется значение логического выражения.

Если оно равно true, то выполняется оператор_1,

иначе – выполняется оператор_2.

После этого управление передается на оператор, следующий за условным.

Если в качестве оператора требуется выполнить несколько операторов, то необходимо заключить их в блок: поставить операторы в фигурные скобки.

17

Пример 5: вычислить значение функции:

$$y = \begin{cases} a - e^{b/a}, & \text{если } a \text{ кратно } b \\ \sqrt{a \cdot \sqrt{b}}, & \text{в противном случае} \end{cases}$$

```
if(a % b  ==  0)   y = a – exp(b/a);
else   y = sqrt(a * sqrt(b));
```

Пример 6: наименьшее из двух значений разделить на три, а большее - удвоить.

```
if(a > b )  {a*=2.;  b/=3.;}  else {b*=2.;  a/=3.;}
```

2.3. Технология программирования задачи с разветвлением на несколько направлений

sin(x)|cos(x)|tan(x)|log(x))|abs(x)
pow(x, a) => x^a

$$z = \begin{cases} sin^2 x, & если \quad x \in \left[-\frac{\pi}{2};+\frac{\pi}{2}\right] \\ cos x^3, & если \quad -\pi \le x < -\frac{\pi}{2} \\ tg\sqrt[4]{\ln|x|}, & если \quad x \in \left(-\infty;-\frac{3\pi}{2}\right) \cup \left(\frac{3\pi}{2};+\infty\right) \\ 1, & во\ всех\ остальных\ случаях \end{cases}$$

&& - And
|| - Or
! -Not

```
const double pi = acos(-1.);
double x=pi/3;
if(x >= - pi/2 &&  x <= pi/2) z = sin(x)*sin(x);
else
  if(x >= - pi &&  x< - pi/2)  z = cos(pow(x,3));
  else
     if (x< - 3*pi / 2 II x > 3*pi/2) z= tan(pow(log(abs(x)),1./4));
     else z=1;
```

```
{double x = log(abs(x)), px = 1./4; z = tan(pow(x, px));}
```

Область видимости переменных **x, px** только текущий блок – они исчезнут из ОП после завершения выполнения операторов блока

3. Оператор выбора

3.1. Назначение оператора

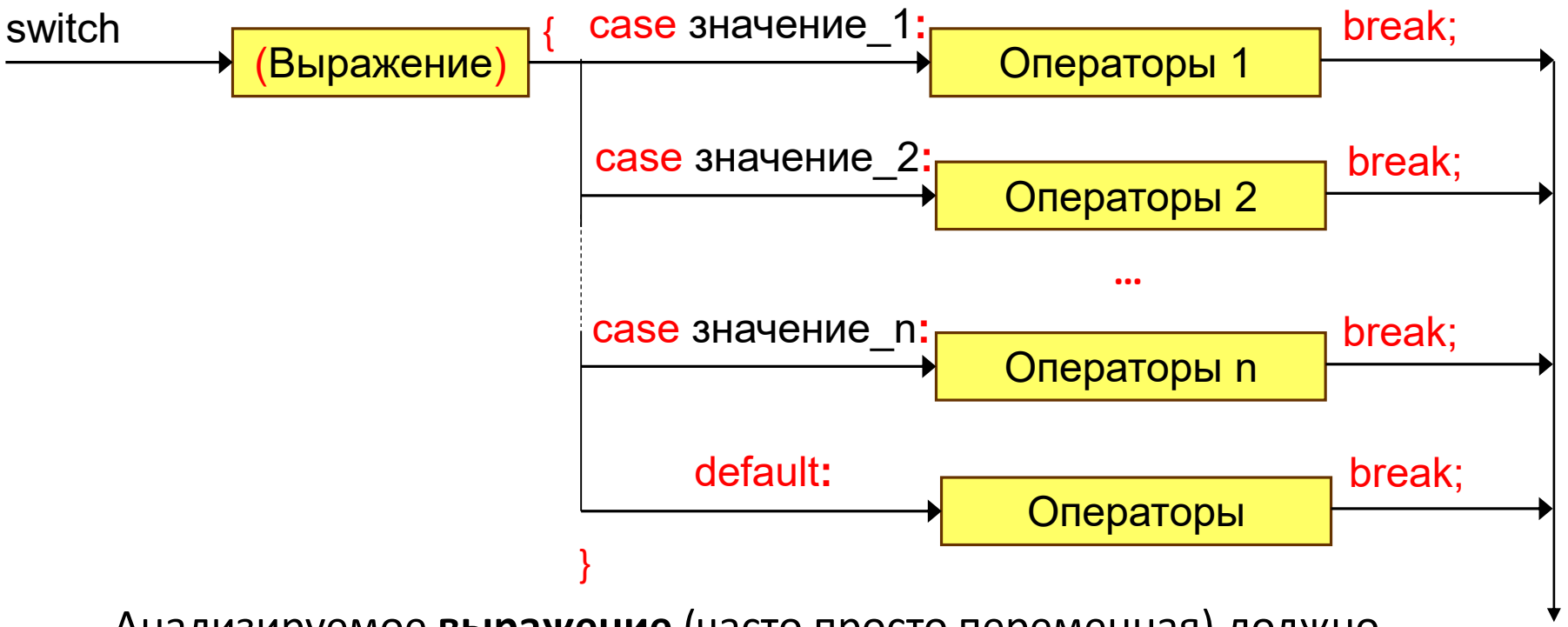
Оператор switch предназначен для разветвления процесса вычислений на несколько направлений.

В отличие от оператора if , switch позволяет сразу проверять переменную на предмет соответствия множеству различных значений, а не с помощью множества отдельных условий.

В таких проверках разрешено использовать только дискретные значения, а не конструкции вроде "больше чем X", поэтому и способ применения этого оператора немного отличается.

20

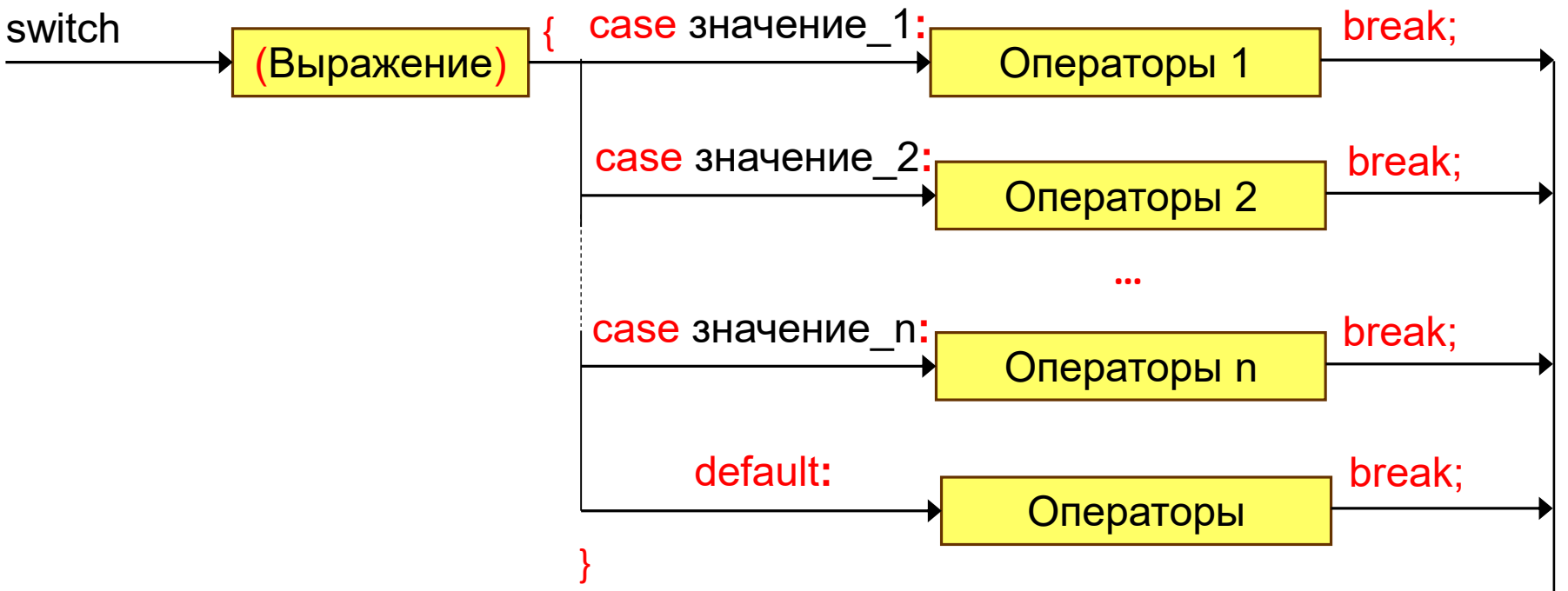
Структурная схема оператора switch.



Анализируемое **выражение** (часто просто переменная) должно давать в результате либо символ, либо целочисленное значение (выражение с вещественным значением недопустимо).

Значение <**выражение**> последовательно сравнивается с каждым из значений <значение_X> (задаваемых с помощью операторов case).

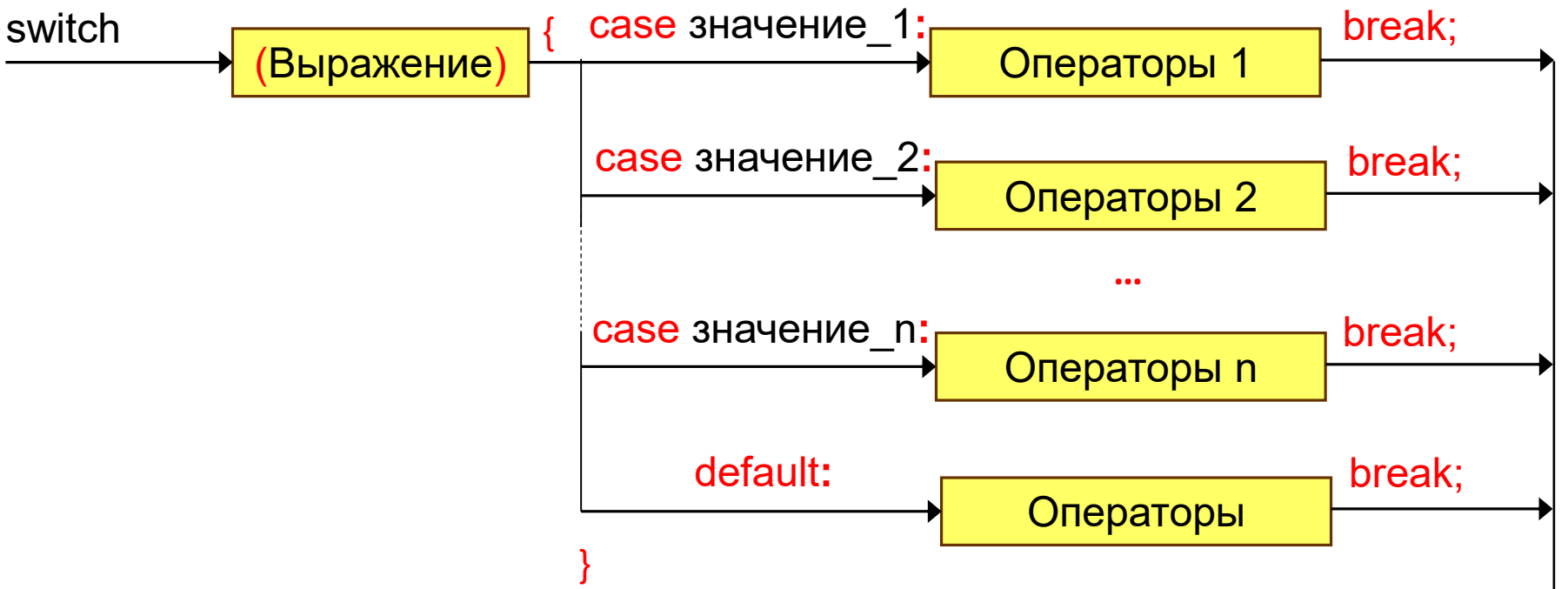
Структурная схема оператора switch.



Если удается обнаружить совпадение, тогда выполняется тот код, который содержится в разделе, соответствующем обнаруженному совпадению.

Если не удается обнаружить ни одного совпадения, выполняется тот код, который содержится в разделе default, при условии, что такой раздел существует.

Структурная схема оператора switch.



В конце кода в каждом разделе добавляется команда break, исключающая возможность перехода после обработки одного блока case к обработке следующего оператора case, потому что это недопустимо.

В данном случае оператор break просто завершает работу оператора switch, после чего процесс обработки продолжается с оператора, следующего после данной структуры.

3.2. Пример:

Составить программу, реализующую

простейший калькулятор

на четыре действия:

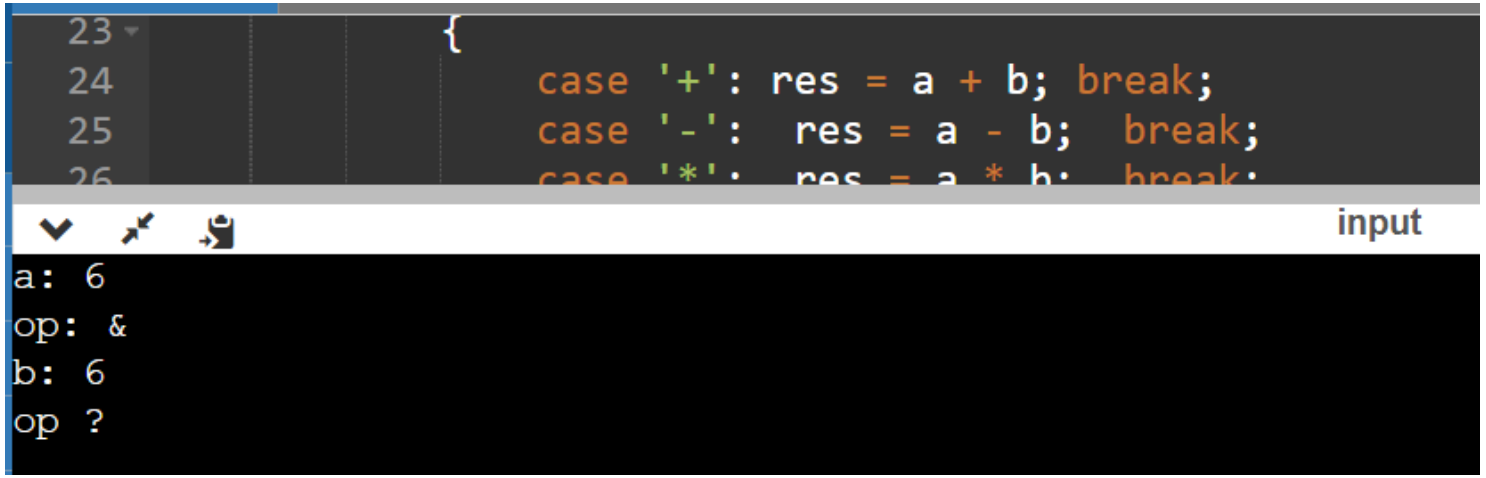
сложение, вычитание,

умножение и деление.

```
double   a, b, res;
char   op;
cout  <<   "a: " ;    cin  >> a;
cout  <<   "op: " ;   cin  >> op;
cout  <<   "b: " ;    cin  >> b;

bool ok = true;
        switch (op)
         {
            case '+': res = a + b; break;
            case '-':  res = a - b;  break;
            case '*':  res = a * b;  break;
            case '/':  res = a / b;   break;
            default: res = 999; ok = false; break;
             }

if (ok) cout <<  "res ="  << res << endl;
else cout << "op ?"  << endl;
```



Форматированный консольный вывод

Функция форматированного вывода printf() имеет следующий прототип:

int printf(const char **форматная_строка*,...);

Первый аргумент *форматная_строка* определяет способ вывода последующих аргументов. Он содержит два типа элементов: *символы*, выводимые на экран, и *спецификаторы формата*, определяющие способ вывода аргументов, следующих за форматной строкой.

Должно быть *полное соответствие* между числом аргументов и числом спецификаторов формата, а также спецификаторы формата и аргументы должны соответствовать друг другу слева направо.

Функция printf() возвращает число напечатанных символов. В случае ошибки она возвращает EOF.

EOF - выдаёт истину, если файл пуст, а если файл не пуст, то выдаётся ложь. Эта функция часто применяется, при работе с вводом/выводом из файлов с использованием операций ">>" и "<<«.

Спецификаторы формата вывода

Спецификаторы формата начинаются со знака процент (%), за которым следует код формата.

Код	**Формат**
%c	Символ
%d	Знаковое десятичное целое число
%i	Знаковое десятичное целое число
%e|E	Научная нотация (e|E - строчнаяIпрописная) – для чисел в формате плавающей точки
%f	Десятичное число с плавающей точкой
%s	Строка символов
%u	Беззнаковое десятичное число
%o	Беззнаковое восьмеричное число
%x|X	Беззнаковое шестнадцатеричное число (строчные буквы|прописные буквы)
%p	Вывод указателя
%%	Выводит знак %

Примеры форматного вывода

```
int num;
num = 5;
printf(“%d”, num);
```

сразу несколько переменных:

```
int num_i;
float num_f;
num_i = 5;
num_f = 10.5;
printf(“num_i = %d, num_f = %f”, num_i, num_f);
```

Результат выполнения программы будет выглядеть так:

num_i = 5, num_f = 10.5

Примеры форматного вывода

```
unsigned short int n, k, n1;
        cout << "Input n k:";    cin >> n >> k;
        printf("\nn= %X  k= %d", n,k);
        //формируем число из единиц нужного порядка k
        n1 = 0xFFFE;
        printf("\nn1= %X", n1);
```

```
Input n k:135 2

n= 87  k= 2
n1= FFFE
```

Форматированный консольный ввод

Стандартный ввод с консоли, как правило, осуществляется с помощью функции **scanf()**. Она читает все стандартные типы данных и автоматически преобразует числа к правильному внутреннему формату:

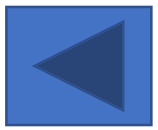
int scanf(const char * форматная_строка,..);

Функция scanf() возвращает число введенных полей. Она возвращает EOF, если обнаруживается преждевременный конец файла.

В ряде версий C++ используется **scanf_s** вместо **scanf**

Спецификаторы формата

Код	Формат
%c	Читает одиночные символы
%d	Читает десятичное число
%i	Читает десятичное число
%e	Читает число с плавающей запятой
%f	Читает число с плавающей запятой
%g	Читает число с плавающей запятой
%o	Читает восьмеричное число
%s	Читает строку
%x	Читает шестнадцатеричное число
%p	Читает указатель
%n	Получает целочисленное значение, равное числу прочитанных символов
%u	Читает беззнаковое целое
%[ ]	Сканирует множество символов



Применение форматного ввода

Для чтения десятичного числа используются спецификаторы %d или %i.

Для чтения числа с плавающей точкой, представленного в стандартной или научной нотации, используются %e, %f или %g.

Можно использовать scanf() для чтения целых чисел в восьмеричном или шестнадцатеричном формате, используя спецификаторы %o и %x соответственно. %x может быть записан как строчными, так и прописными буквами.

При вводе шестнадцатиричных чисел можно вводить буквы от A до F как прописными, так и строчными.

Примеры форматного ввода

```
int main(void)
{
int i, j;
scanf("%o%x", &i,&j);
printf("%o %x", i, j);
return 0;
}
```

Программа осуществляет ввод восьмеричного и шестнадцатеричного чисел.

Функция scanf() останавливает чтение чисел при обнаружении первого нечислового символа.

Примеры форматного ввода

Программа читает беззнаковое целое и помещает его значение в num.

```
unsigned num;
scanf ("%u", &num);
```

при вводе «x y» фрагмент кода

```
scanf ("%c%c%c", &a, &b, &c);
```

вернет, символы x в *a*, пробел в *b* и y в *c*.

VS2013 scanf_s

Краткие итоги

- **Функция main** обязана присутствовать в каждой программе на языке С, С++. Именно с нее начинается выполнение программы, она – главная (именно так переводится служебное слово main). Предшествующее ей служебное слово int (от integer – целый) сообщает, что результатом работы функции main должно быть целое число. По возвращаемому функцией значению операционная система, запустившая программу main, может "сообразить", правильно или неправильно завершилась работа программы. По общепринятому соглашению нулевое значение, возвращаемое функцией main, свидетельствует о нормальном завершении работы программы.
- **using namespace** – директива, описывающая пространство имён и позволяющая исключить конфликты имен идентификаторов.
- **Именные константы** (вместо указания их численных значений) делают программу более наглядной и позволяют программисту сгруппировать в начале программы все постоянные величины.
- **При записи числовой константы** надо помнить, что целая константа имеет тип int и занимает 4 байта, а вещественная – тип double (8байт). Изменить типы можно с применением суффиксов.
- Помимо **десятичных числовых** констант на С++ допускается применение восьмеричных и шестнадцатеричных констант, компилятор их отличает от десятичных констант по соответствующим префиксами. Для двоичных чисел на С++ (в VS) не предусмотрено представление.
- **Потоки консольного ввода/вывода** позволяют выделить все операции при работе с консолью в отдельные библиотеки и стандартизировать многие однотипные процессы ввода /вывода**.**
- **Бесформатный ввод/вывод** с помощью операторов cin | cout достаточно прост, **форматированный** ввод/вывод гибче, но требует изучения специализированных форматов и грамотного их использования.

Контрольные вопросы

1. Как по-вашему – где надёжнее всего размещать директиву using namespace: до main или в любой другой функции?
2. Функция main оператором return возвращает 0. А что этот ноль означает? А что будет означать, например, 1? Смысл этих констант?
3. Видите ли Вы ошибки в следующем фрагменте программы? Чему равен размер массива b в байтах?

```
int const na =2.5L;
int b(na);
```

4. Какой результат будет выведен на консоль после выполнения ниже представленного фрагмента:

```
int x = 1.2f, y = 2l, z = 034, u = 0x23, w; w = y / x * u / z; cout << "\nx = " << x << "\n
z = " << z << "\n w = " << w << endl;
```
?

5. Какой результат будет выведен на консоль после выполнения ниже представленного фрагмента:

```
int p = 25l; printf("\n %d \t %x  %o \n", p, p, p);
```
?

Проверьте свои оценки на компьютере.

Краткие итоги

- **Оператор if на** C++ отличается от аналогичного оператора на Python лишь синтаксически. Условие всегда размещается в скобках. Если за if или else стоит не один, а несколько операторов, их помещают в блок – обрамляют фигурными скобками.
- **Логическая (условная) функция** – возвращает значение и может быть встроена в вычислительные выражения. Необходимо точно соблюдать её синтаксис. Функция не поддерживает блоки операторов.
- **Оператора выбора switch** не такой гибкий, как if, но позволяет строить наглядные коды и , главное, обладает бОльшей производительностью.

Тесты на проверку знаний логических операций

- Чему равно значение выражения (! a && (b || c)), где a, b и c - величины типа bool, имеющие значения true, true и false соответственно?

(Отметьте один правильный вариант ответа.)

Вариант 1 false

Вариант 2 true

- Чему равно значение выражения (a && ! b || c), где a, b и c -величины типа bool, имеющие значения false, true и true соответственно?

(Отметьте один правильный вариант ответа.)

Вариант 1 false

Вариант 2 true

Задания на самостоятельную работу

1.Можно ***самостоятельно*** выбрать варианты решения из Л/Р 2 в moodle – задания 1 и 2 – линейный алгоритм; 3и 4 – ветвление/переключатель.

Или можно сделать эти задания:

- **Задание 3.** ***Составить программу на C++ для вычисления указанных математических функций, предусмотрев случаи разрыва функций: определить ОДЗ и запрограммировать вывод результата, если он может быть получен, в противном случае – запрограммировать сообщение, почему вычисление функции невозможно. Аргумент x вводится с консоли, константы записываются в формате с плавающей точкой, параметры (a, b, c) инициализируются в теле программы. Аналитического упрощения функций НЕ делать.*** ***(По аналогии с примером 5 методички к Л/Р2.)***

$\sqrt{\frac{log_a \sin x}{\cos \sqrt{x}}}$, a $= 2{,}35 \cdot 10^{-3}$, x – любое число, задаётся в градусах.

Задание 4. ***Используя сочетание операторов if и оператора выбора switch, составить программу по заданному условию.***

В 3-х шкатулках лежат A ≠ B ≠ C денег. Если игрок называет цифру 1, открывается шкатулка с наибольшим количеством денег, если цифру 3, то с наименьшим, если 2 то со средним значением денег.