

Оператор **switch**

Операторы
управления потоком
выполнения
программ

Остановка

Прыжок

Условные
ветвления

Циклы

Исключения

Функция
exit(0).
Подключаем
<stdlib >.

Ключевые
слова goto,
break и
continue .

Операторы
if, else, else if.
**Ключевое
слово switch**

?

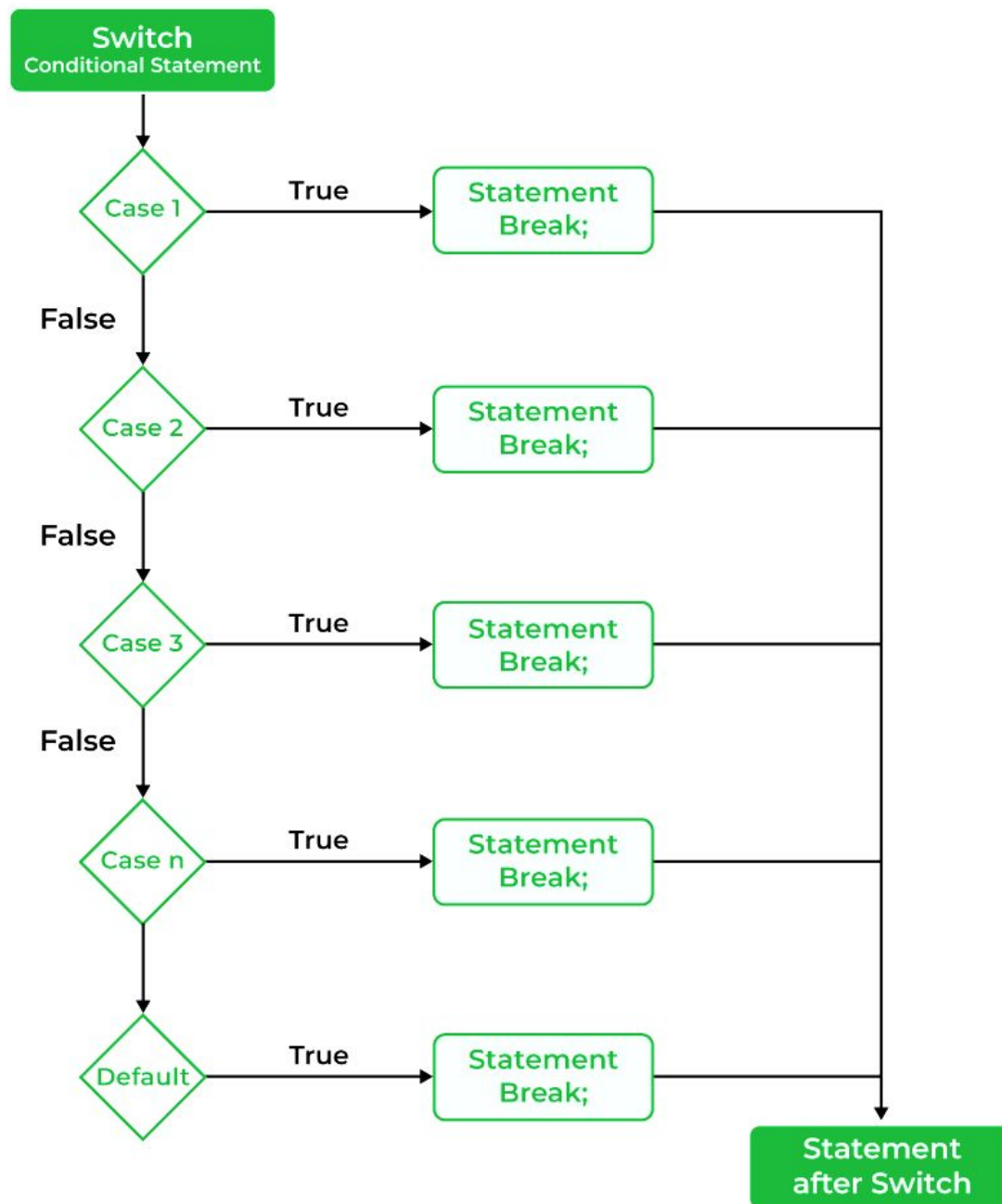
?

Оператор if/else

```
1 #include <iostream>
2 using namespace std;
3
4 int main()
5 {
6     setlocale(LC_ALL, "rus");
7
8     int NumberOfFinger = 0;
9
10    cout << "Введите номер: ";
11    cin >> NumberOfFinger;
12
13    if (NumberOfFinger == 1)
14        cout << "\nРезультат: Большой палец\n";
15    else if (NumberOfFinger == 2)
16        cout << "\nРезультат: Указательный палец\n";
17    else if (NumberOfFinger == 3)
18        cout << "\nРезультат: Средний палец\n";
19    else if (NumberOfFinger == 4)
20        cout << "\nРезультат: Безымянный палец\n";
21    else if (NumberOfFinger == 5)
22        cout << "\nРезультат: Мизинец\n";
23    else
24        cout << "\nНет соответствий!\n\n";
25
26    return 0;
27 }
```

Оператор switch

```
4     using namespace std;
5
6     int main()
7     {
8         setlocale(LC_ALL, "rus");
9
10        int NumberofFinger = 0;
11
12        cout << "Введите номер: ";
13        cin >> NumberofFinger;
14
15        switch (NumberofFinger)
16        {
17            case 1:
18                cout << "\nРезультат: Большой палец\n";
19                break;
20            case 2:
21                cout << "\nРезультат: Указательный палец\n";
22                break;
23            case 3:
24                cout << "\nРезультат: Средний палец\n";
25                break;
26            case 4:
27                cout << "\nРезультат: Безымянный палец\n";
28                break;
29            case 5:
30                cout << "\nРезультат: Мизинец\n";
31                break;
32            default:
33                cout << "\nРезультат: Нет соответствий!\n";
34                break;
35        }
36        return 0;
37    }
38 }
```



2.
Лейблы

1.
Оператор

```
4  using namespace std;
5
6  int main()
7  {
8      setlocale(LC_ALL, "rus");
9
10     int NumberofFinger = 0;
11
12     cout << "Введите номер: ";
13     cin >> NumberofFinger;
14
15     switch (NumberofFinger)
16     {
17     case 1:
18         cout << "\nРезультат: Большой палец\n";
19         break;
20     case 2:
21         cout << "\nРезультат: Указательный палец\n";
22         break;
23     case 3:
24         cout << "\nРезультат: Средний палец\n";
25         break;
26     case 4:
27         cout << "\nРезультат: Безымянный палец\n";
28         break;
29     case 5:
30         cout << "\nРезультат: Мизинец\n";
31         break;
32     default:
33         cout << "\nРезультат: Нет соответствий!\n";
34         break;
35     }
36     return 0;
37 }
38
```

Лейблы case

Первый вид лейбла — это `case` (или просто *"кейс"*), который объявляется с использованием **ключевого слова** `case` и имеет константное выражение.

Константное выражение, находящееся после ключевого слова `case`, проверяется на равенство с выражением, находящимся возле ключевого слова `switch`. Если они совпадают, то тогда выполняется код под соответствующим кейсом.

```
1. switch (z)
2. {
3.     case 3:
4.         case 3: // нельзя, значение 3 уже используется!
5.         case COLOR_PURPLE: // нельзя, COLOR_PURPLE вычисляется как 3!
6. };
```

Все выражения case должны производить уникальные значения!

Лейбл по умолчанию

Второй тип лейбла — это **лейбл по умолчанию** (так называемый "*default case*"), который объявляется с использованием **ключевого слова default**. Код под этим лейблом выполняется, если ни один из кейсов не соответствует выражению switch. Лейбл по умолчанию является необязательным.

В одном switch может быть только один default.

Обычно его объявляют самым последним в блоке switch.

```
switch (NumberofFinger)
{
    case 1:
        cout << "\nРезультат: Большой палец\n";
        break;
    case 2:
        cout << "\nРезультат: Указательный палец\n";
        break;
    case 3:
        cout << "\nРезультат: Средний палец\n";
        break;
    case 4:
        cout << "\nРезультат: Безымянный палец\n";
        break;
    case 5:
        cout << "\nРезультат: Мизинец\n";
        break;
    default:
        cout << "\nРезультат: Нет соответствий!\n";
        break;
}
return 0;
```

Выполнение начинается с первого кейса, который находится после соответствующего кейса и продолжается до тех пор, пока не будет выполнено одно из следующих условий завершения:

- Достигнут конец блока `switch`.
- Выполняется оператор `return`.
- Выполняется оператор `goto`.
- Выполняется оператор `break`.

Последовательность выполнения кода

```
1. switch (2)
2. {
3.     case 1: // Не совпадает!
4.         std::cout << 1 << '\n'; // пропускается
5.     case 2: // Совпало!
6.         std::cout << 2 << '\n'; // выполнение кода начинается здесь
7.     case 3:
8.         std::cout << 3 << '\n'; // это также выполнится
9.     case 4:
10.        std::cout << 4 << '\n'; // и это
11.    default:
12.        std::cout << 5 << '\n'; // и это
13. }
```



**fall-through
(проваливаться)**

Оператор break

Когда компилятор встречает оператор `break`, то выполнение кода переходит из `switch` на следующую строку после блока `switch`. Рассмотрим вышеприведенный пример, но уже с корректно вставленными опер

```
1. switch (2)
2. {
3.     case 1: // не совпадает - пропускается
4.         std::cout << 1 << '\n';
5.         break;
6.     case 2: // совпало! Выполнение начинается со следующего кейс-а
7.         std::cout << 2 << '\n'; // выполнение начинается здесь
8.         break; // оператор break завершает выполнение switch-а
9.     case 3:
10.        std::cout << 3 << '\n';
11.        break;
12.    case 4:
13.        std::cout << 4 << '\n';
14.        break;
15.    default:
16.        std::cout << 5 << '\n';
17.        break;
18.}
19. // Выполнение продолжается здесь
```

Результат:

Оператор break

Предупреждение: Не забывайте использовать оператор break в конце каждого кейса. Его отсутствие — одна из наиболее распространенных ошибок новичков!

Объявление переменной и её инициализация внутри case

```
1. switch (x)
2. {
3.     case 1:
4.         int z; // ок, объявление разрешено
5.         z = 5; // ок, операция присваивания разрешена
6.         break;
7.
8.     case 2:
9.         z = 6; // ок, переменная z была объявлена выше, поэтому мы можем
            использовать её здесь
10.        break;
11.
12.    case 3:
13.        int c = 4; // нельзя, вы не можете инициализировать переменные внутри
            case
14.        break;
15.
16.    default:
17.        std::cout << "default case" << std::endl;
18.        break;
19.}
```

Совмещение условий

Можно определять для нескольких меток `case` один набор инструкций:

```
1 #include <iostream>
2
3 int main()
4 {
5     int x {2};
6
7     switch(x)
8     {
9         case 1:
10        case 2:
11            std::cout << "x = 1 or 2" << "\n";
12            break;
13        case 3:
14        case 4:
15            std::cout << "x = 3 or 4" << "\n";
16            break;
17        case 5:
18            std::cout << "x = 5" << "\n";
19            break;
20    }
21 }
```

Здесь если `x=1` или `x=2`, то выполняется одна и та же инструкция `std::cout << "x = 1 or 2" << "\n"`. Аналогично для вариантов `x=3` и `x=4` также определена общая инструкция.

Тест

10

Дано целое число (от пользователя) в диапазоне 1-7. Вывести строку – название дня недели, соответствующее данному числу (1 – понедельник, 2 – вторник и т.д.).

Дано целое число (от пользователя) в диапазоне 1-7. -> int num;

Вывести строку – название дня недели, соответствующее данному числу (1 – понедельник, 2 – вторник и т.д.). -> 7 лейблов кейсов + 1 лейбл по умолчанию

```
setlocale(0, "");  
int num;  
cin >> num;
```

*Объявление переменной и
получение её значения от
пользователя*

```
switch (num)  
{
```

*Для хранения дней недели
используем оператор switch*

```
...
```

```
return 0;
```

```
}
```

```
switch (num)
{
case 1:

case 2:

case 3:

case 4:

case 5:

case 6:

case 7:

default:

}
return 0;
}
```