Конструкция **switch/case** оценивает некоторое выражение и сравнивает его значение с набором значений. И при совпадении значений выполняет определенный код.:

Конструкция switch имеет следующее формальное определение:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | switch (выражение)  {  case значение1:  код,выполняемый если выражение имеет значение1  break;  case значение2:  код,выполняемый если выражение имеет значение2  break;  //.............  case значениеN:  код, выполняемый если выражение имеет значениеN  break;  default:  код, выполняемый если выражение не имеет ни одно из выше указанных значений  break;  } |

После ключевого слова **switch** в скобках идет сравниваемое выражение. Значение этого выражения последовательно сравнивается со значениями, помещенными после оператора **сase**. И если совпадение будет найдено, то будет выполняться определенный блок **сase**.

В конце каждого блока сase должен ставиться один из операторов перехода: **break**, **goto case**, **return** или **throw**. Как правило, используется оператор **break**. При его применении другие блоки case выполняться не будут.

Например:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | string name = "Tom";    switch (name)  {  case "Bob":  Console.WriteLine("Ваше имя - Bob");  break;  case "Tom":  Console.WriteLine("Ваше имя - Tom");  break;  case "Sam":  Console.WriteLine("Ваше имя - Sam");  break;  } |

В данном случае конструкция switch последовательно сравнивает значение переменной name с набором значений, которые указаны после операторов case. Поскольку здесь значение переменной name - строка "Tom", то будет выполняться блок

|  |  |
| --- | --- |
| 1  2  3 | case "Tom":  Console.WriteLine("Ваше имя - Tom");  break; |

Соответственно мы увидим на консоли

Ваше имя - Tom

Если значение переменной name не совпадает ни с каким значением после операторов case, то ни один из блоков case не выполняется. Однако если даже в этом случае нам все равно надо выполнить какие-нибудь действия, то мы можем добавить в конструкцию **switch** необязательный блок **default**. Например:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | string name = "Alex";    switch (name)  {  case "Bob":  Console.WriteLine("Ваше имя - Bob");  break;  case "Tom":  Console.WriteLine("Ваше имя - Tom");  break;  case "Sam":  Console.WriteLine("Ваше имя - Sam");  break;  default:  Console.WriteLine("Неизвестное имя");  break;  } |

В данном случае никакое из значений после операторов case не совпадает со значением переменной name, поэтому будет выполняться блок default:

|  |  |
| --- | --- |
| 1  2  3 | default:  Console.WriteLine("Неизвестное имя");  break; |

Однако если мы хотим, чтобы, наоборот, после выполнения текущего блока case выполнялся другой блок case, то мы можем использовать вместо break оператор goto case:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | int number = 1;  switch (number)  {  case 1:  Console.WriteLine("case 1");  goto case 5; // переход к case 5  case 3:  Console.WriteLine("case 3");  break;  case 5:  Console.WriteLine("case 5");  break;  default:  Console.WriteLine("default");  break;  } |

### Возвращение значения из switch

Конструкция **switch** позволяет возвращать некоторое значение. Для возвращения значения в блоках **case** может применятся оператор **return**. Например, определим следующий метод:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | int DoOperation(int op, int a, int b)  {  switch (op)  {  case 1: return a + b;  case 2: return a - b;  case 3: return a \* b;  default: return 0;  }  } |

В метод DoOperation() передается числовой код операции и два операнда. В зависимости от кода операции над операндами выполнется определенная операция и ее результат возвращается из метода. Для примера при по умолчанию из метода возвращается 0, если код операции не равен 1, 2 или 3.

Затем мы можем вызвать этот метод:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | int DoOperation(int op, int a, int b)  {  switch (op)  {  case 1: return a + b;  case 2: return a - b;  case 3: return a \* b;  default: return 0;  }  }    int result1 = DoOperation(1, 10, 5); // 15  Console.WriteLine(result1); // 15    int result2 = DoOperation(3, 10, 5); // 50  Console.WriteLine(result2); // 50 |

### Получение результата из switch

Хотя конструкция **switch** в примере выше прекрасно работает, тем не менее мы ее можем сократить и получить результат неосредственно из конструкции **switch**:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | int DoOperation(int op, int a, int b)  {  int result = op switch {  1 => a + b,  2 => a - b,  3 => a \* b,  \_ => 0  };  return result;  } |

Теперь не требуется оператор **case**, а после сравниваемого значения ставится оператор стрелка **=>**. Значение справа от стрелки выступает в качестве возвращаемоего значения. Кроме того, вместо оператора **default** используется почерк \_. В итоге результат конструкции switch будет присвиваиваться переменной result.

Естестввенно, мы можем сразу возвратить из метода результат без присвоения переменной результата конструкции switch:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | int DoOperation(int op, int a, int b)  {  return op switch  {  1 => a + b,  2 => a - b,  3 => a \* b,  \_ => 0  };  } |

Или сделать метод еще короче:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | int DoOperation(int op, int a, int b) => op switch  {  1 => a + b,  2 => a - b,  3 => a \* b,  \_ => 0  }; |

Обращаю внимание, что данное упрощение касается лишь таких конструкций **switch**, которые **возвращают** некоторые значения, как в примере выше.

Стоит отметить, что при возвращении значения из метода, метод должен в любом случае возвращать значение. Например, следующая версия метода не будет работать

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | int DoOperation(int op, int a, int b)  {  return op switch  {  1 => a + b,  2 => a - b,  3 => a \* b  };  } |

Эта версия метода возвращает значение, если код операции равен 1, 2 или 3. Но что, если будет передано значение 4 или какое-то другое? Поэтому данная версия метода даже не скомпилируется. Поэтому нам надо предусмотреть возвращение значения из метода при всех возможных вариантах. То есть, мы можем, как в примере выше, добавить в конструкцию switch блок default, в котором будет возвращаться значение при всех остальных случаях.