Лекция 3. Оператор выбора switch

Цель лекции: Формирование у студентов знаний, связанных с использованием оператора выбора на языке C#.

В результате изучения данной лекции студенты будут способны:

- Продемонстрировать знание синтаксиса оператора switch
- Выбирать необходимый оператор для использования при программировании алгоритма ветвления

Вторым оператором выбора в C# является оператор switch, который обеспечивает многонаправленное ветвление программы. Следовательно, этот оператор позволяет сделать выбор среди нескольких альтернативных вариантов дальнейшего выполнения программы. Несмотря на то что многонаправленная проверка может быть организована с помощью последовательного ряда вложенных операторов if, во многих случаях более эффективным оказывается применение оператора switch. Этот оператор действует следующим образом. Значение выражения последовательно сравнивается с константами выбора из заданного списка. Как только будет обнаружено совпадение с одним из условий выбора, выполняется последовательность операторов, связанных с этим условием.

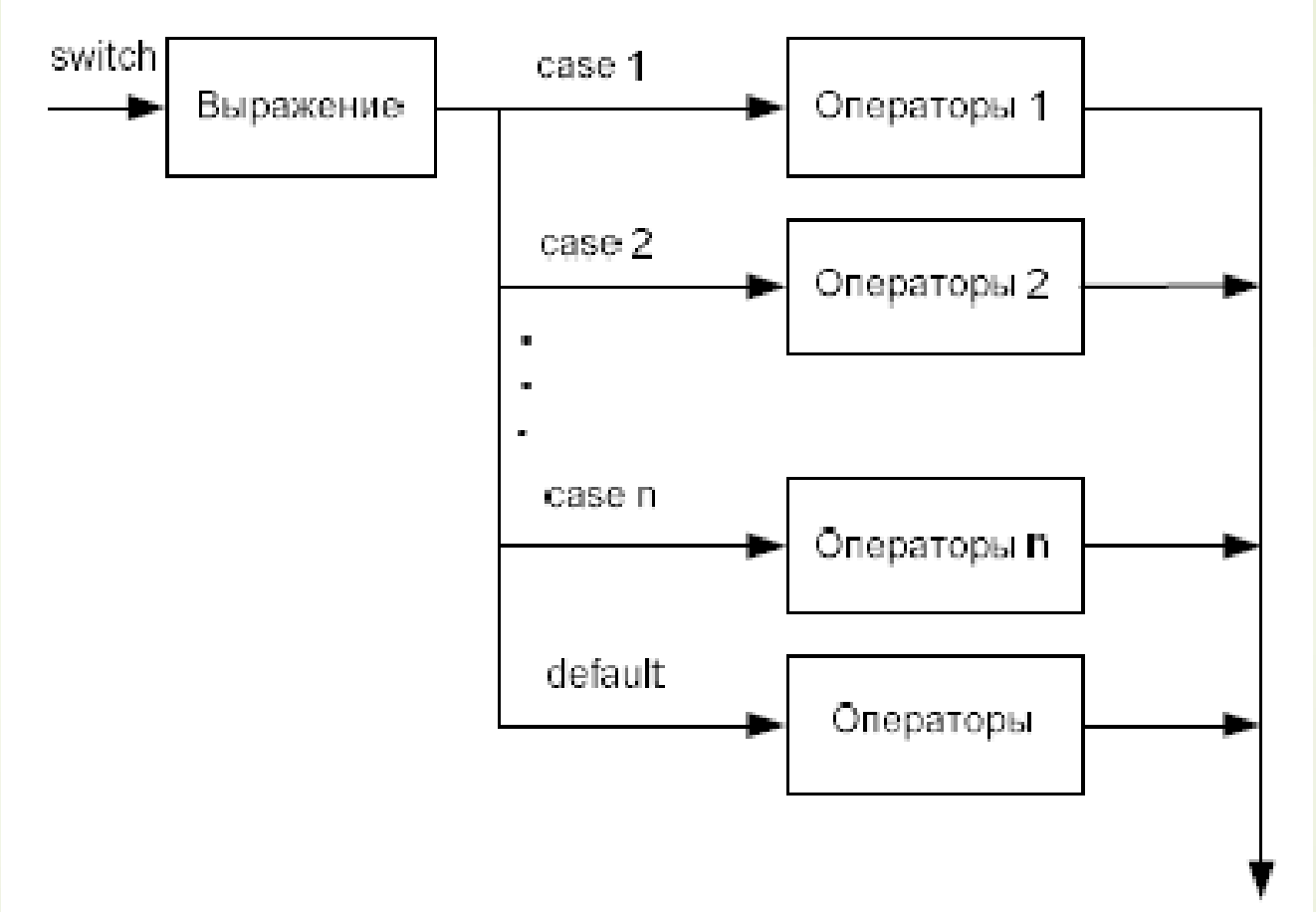

Рисунок 3.1 – Структурная схема оператора switch

Ниже приведена общая форма оператора switch.

```
switch(выражение) {
  case константа1:
    последовательность операторов
    break;
  case константа2:
    последовательность операторов
    break;
  case константа3:
    последовательность операторов
    break;
  default:
    последовательность операторов
    break;
}
```

Заданное *выражение* в операторе switch должно быть целочисленного типа (char, byte, short или int), перечислимого или же строкового. А выражения других типов, например с плавающей точкой, в операторе switch не допускаются. Зачастую выражение, управляющее оператором switch, просто сводится к одной переменной. Кроме того, константы выбора должны иметь тип, совместимый с типом выражения. В одном операторе switch не допускается наличие двух одинаковых по значению констант выбора. Последовательность операторов из ветви default выполняется в том случае, если ни одна из констант выбора не совпадает с заданным выражением. Ветвь default не является обязательной. Если же она отсутствует и выражение не совпадает ни с одним из условий выбора, то никаких действий вообще не выполняется. Если же происходит совпадение с одним из условий выбора, то выполняются операторы, связанные с этим условием, вплоть до оператора break.

Ниже приведен пример программы, в котором демонстрируется применение оператора switch.

// Продемонстрировать применение оператора switch.

```
using System;
class SwitchDemo {
  static void Main() {
    int i;
    for(i=0; i<10; i++)
      switch(i) {
        case 0:
          Console.WriteLine("i равно нулю");
          break;
        case 1:
          Console.WriteLine("i равно единице");
          break;
        case 2:
          Console.WriteLine("i равно двум");
          break;
        case 3:
          Console.WriteLine("i  равно трем");
          break;
        case 4:
          Console.WriteLine("i равно четырем");
          break;
        default:
          Console.WriteLine("i  равно или больше пяти");
          break;
    }
  }
```

Как видите, на каждом шаге цикла выполняются операторы, связанные с совпадающей константой выбора, в обход всех остальных операторов. Когда же значение переменной i становится равным или больше пяти, то оно не совпадает ни с одной из констант выбора, а следовательно, выполняются операторы из ветви default.

В приведенном выше примере оператором switch управляла переменная i типа int. Как пояснялось ранее, для управления оператором switch может быть использовано выражение любого целочисленного типа, включая и char. Ниже приведен пример применения выражения и констант выбора типа char в операторе switch.

```
// Использовать элементы типа char для управления оператором switch.
using System;
class SwitchDemo2 {
  static void Main() {
    char ch;
    for(ch='A'; ch<= 'E'; ch++)
      switch(ch) {
        case 'A':
          Console.WriteLine("ch содержит A");
          break;
        case 'B':
          Console.WriteLine("ch содержит B");
          break;
        case 'C':
          Console.WriteLine("ch содержит C");
          break;
        case 'D':
          Console.WriteLine("ch содержит D");
          break;
        case 'E':
          Console.WriteLine("ch содержит E");
          break;
  } } }
```

Вот какой результат дает выполнение этой програ

```
ch содержит A
ch содержит B
ch содержит C
ch содержит D
ch содержит E
```

Обратите в данном примере внимание на отсутствие ветви default в операторе switch. Напомним, что ветвь default не является обязательной. Когда она не нужна, ее можно просто опустить.

Переход последовательности операторов, связанных с одной ветвью case, в следующую ветвь case считается ошибкой, поскольку в C# должно непременно соблюдаться правило недопущения "провалов" в передаче управления ходом выполнения программы. Именно поэтому последовательность операторов в каждой ветви case оператора switch оканчивается оператором break. Когда в последовательности операторов отдельной ветви case встречается оператор break, происходит выход не только из этой ветви, но из всего оператора switch, а выполнение программы возобновляется со следующего оператора, находящегося за пределами оператора switch. Последовательность операторов в ветви default также должна быть лишена "провалов", поэтому она завершается, как правило, оператором break.

Правило недопущения "провалов" относится к тем особенностям языка C#, которыми он отличается от C, C++ и Java. В этих языках программирования одна ветвь case может переходить (т.е. "проваливаться") в другую. Данное правило установлено в C# для ветвей case по двум причинам. Во-первых, оно дает компилятору возможность свободно изменять порядок следования последовательностей операторов из ветвей case для целей оптимизации. Такая реорганизация была бы невозможной, если бы одна ветвь case могла переходить в другую. И во-вторых, требование завершать каждую ветвь case явным образом исключает непроизвольные ошибки программирования, допускающие переход одной ветви case в другую. Несмотря на то что правило недопущения "провалов" не допускает переход одной ветви case в другую, в двух или более ветвях case все же разрешается ссылаться с помощью меток на одну и ту же кодовую последовательность, как показано в следующем примере программы

```
// Пример "проваливания" пустых ветвей case.
using System;
class EmptyCasesCanFall {
  static void Main() {
    int i;
    for(i=1; i < 5; i++)
      switch(i) {
        case 1:
        case 2:
        case 3: Console.WriteLine("i равно 1, 2 или 3");
          break;
        case 4: Console.WriteLine("i равно 4");
          break;
    }
  }
}
```

Ниже приведен результат выполнения этой программы.

```
i равно 1, 2 или 3
i равно 1, 2 или 3
i равно 1, 2 или 3
i равно 4
```

Если значение переменной i в данном примере равно 1, 2 иди 3, то выполняется первый оператор, содержащий вызов метода WriteLine(). Такое расположение нескольких меток ветвей case подряд не нарушает правило недопущения "провалов"; поскольку во всех этих ветвях используется одна и та же последовательность операторов.

Расположение нескольких меток ветвей case подряд зачастую применяется в том случае, если у нескольких ветвей имеется общий код. Благодаря этому исключается излишнее дублирование кодовых последовательностей.

Вложенные операторы switch

Один оператор switch может быть частью последовательности операторов другого, внешнего оператора switch. И такой оператор switch называется *вложенным*. Константы выбора внутреннего и внешнего операторов switch могут содержать общие значения, не вызывая никаких конфликтов. Например, следующий фрагмент кода является вполне допустимым.

```
switch(ch1) {
  case 'A': Console.WriteLine("Эта ветвь A — Часть " +
"внешнего оператора switch.");
  switch(ch2) {
    case 'A':
      Console.WriteLine("Эта ветвь A — часть " +
"внутреннего оператора switch");
      break;
```

Заключение

В рамках данной лекции был изучен оператор выбора `switch`, предназначенный для организации многонаправленного ветвления в языке C#. Были подробно рассмотрены его синтаксис, требования к типам управляющих выражений и констант выбора, назначение ветви `default`, а также правило недопущения «провалов» между ветвями `case` и возможности создания вложенных конструкций `switch`.

Грамотное применение оператора `switch` делает программный код более читаемым, структурированным и эффективным по сравнению с громоздкими цепочками вложенных операторов `if-else-if` в тех случаях, когда выбор зависит от точного совпадения значения переменной с одним из множества дискретных вариантов. Понимание строгих правил языка C# (таких как обязательное завершение непустых блоков `case` оператором `break`) помогает избежать распространенных логических ошибок и повышает надежность разрабатываемых алгоритмов.

Вопросы для самопроверки

1. **Назначение оператора:** В каких ситуациях использование оператора `switch` является более предпочтительным и эффективным, чем применение многоступенчатой конструкции `if-else-if`?

2. **Типы данных:** Перечислите типы данных, к которым может относиться управляющее выражение в операторе `switch`. Допускается ли использование чисел с плавающей точкой (`float`, `double`)?

3. **Ветвь `default`:** Какую роль выполняет ветвь `default` в операторе `switch`? Является ли она обязательной, и как поведет себя программа при ее отсутствии, если совпадений ни с одной из констант `case` не найдено?

4. **Правило недопущения «провалов»:** Почему в языке C# запрещен прямой переход выполнения из одной непустой ветви `case` в другую? Назовите две основные причины введения этого правила.

5. **Группировка меток `case`:** Как в языке C# организовать выполнение одного и того же блока кода для нескольких разных значений констант выбора, не нарушая правило недопущения «провалов»?

6. **Оператор `break`:** Что происходит с ходом выполнения программы в тот момент, когда внутри ветви `case` срабатывает оператор `break`?

Список используемых источников

1. **Шилдт Г.** C# 4.0: полное руководство. — М.: Вильямс, 2019. — 1056 с. *(Глава 5: Управляющие операторы — подробный разбор синтаксиса `switch`, правила отсутствия «провалов» и вложенных операторов).*

2. **Троелсен Э., Джепикс Ф.** Язык программирования C# и платформа .NET. — СПб.: Диалектика, 2022. — 1312 с. *(Раздел: Конструкции принятия решений и оператор `switch`).*

3. **Албахари Д., Албахари Б.** C#. Справочник. Полное описание языка. — СПб.: БХВ-Петербург, 2023. — 1088 с. *(Спецификация инструкций ветвления и допустимых типов выражений в `switch`).*

4. **Павловская Т. А.** C#. Программирование на языке высокого уровня: Учебник для вузов. — СПб.: Питер, 2020. — 432 с. *(Глава: Операторы выбора и программирование разветвляющихся алгоритмов).*

5. **Официальная документация Microsoft Learn:** Инструкция `switch` (справочник по C#) [Электронный ресурс]. — URL: microsoft.com (дата обращения: 13.11.2026).