Лекция 2. ОПЕРАТОРЫ ВЕТВЛЕНИЯ

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

В результате изучения данной лекции студенты будут способны:

- Продемонстрировать знание базовых конструкторов структурного программирования
- Продемонстрировать знание операторов сравнения и логических операторов
- Продемонстрировать понимание условных алгоритмов

Базовые конструкции структурного программирования

В этой лекции рассматриваются основные операторы C#, составляющие так называемые базовые конструкции структурного программирования. *Структурное программирование* – это технология создания программ, позволяющая путем соблюдения определенных правил сократить время разработки и уменьшить количество ошибок. Идеи структурного программирования получили свое дальнейшее развитие в *объектно-ориентированном программировании* – технологии, позволяющей достичь простоты структуры и управляемости очень больших программных систем. Несмотря на то, что язык C# реализует объектно-ориентированную парадигму, принципы структурного программирования лежат в основе кодирования каждого метода, каждого фрагмента алгоритма.

В C# идеи структурного программирования используются на самом низком уровне – при написании методов объектов. Доказано, что любой алгоритм можно реализовать только из трех структур, называемых *базовыми конструкциями* структурного программирования, – это следование, ветвление и цикл.

Следованием называется конструкция, реализующая последовательное выполнение двух или

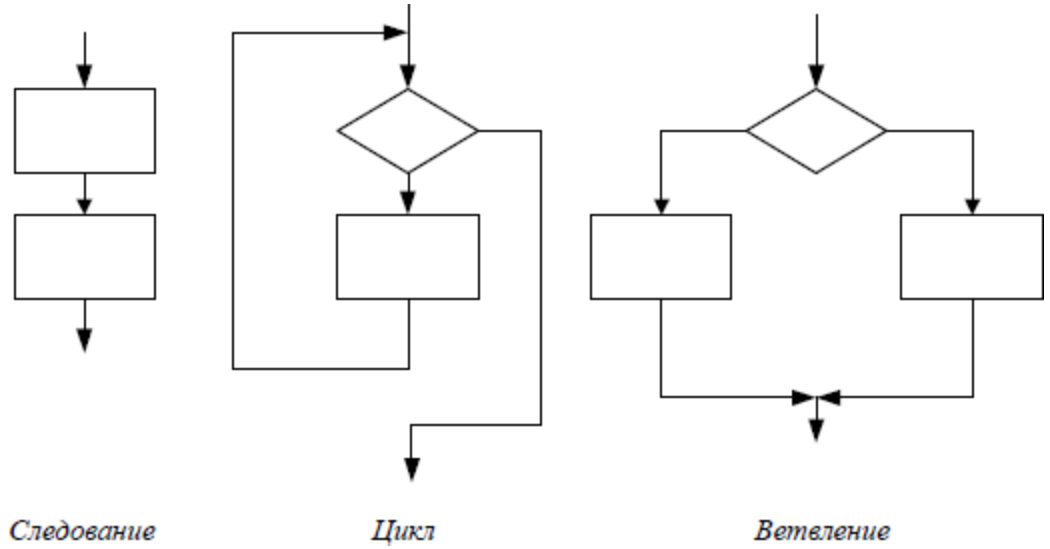


Рисунок 2.1 – Базовые конструкции структурного программирования

Конструкции могут вкладываться друг в друга произвольным образом, например, цикл может содержать следование из двух ветвлений, каждое из которых включает вложенные циклы.

Целью использования базовых конструкций является получение программы простой структуры. Такую программу легко читать, отлаживать и при необходимости вносить в нее изменения.

В большинстве языков высокого уровня существует несколько реализаций базовых конструкций; в C# есть четыре вида циклов и два вида ветвлений (на два и на произвольное количество направлений). В каждом случае программист выбирает наиболее подходящие средства. Главное, о чем нужно помнить при написании программ, – они должны состоять из четкой последовательности блоков строго определенной структуры.

Таблица 2.1 – Операторы сравнения

Оператор	Знак	Запись
равно	==	a==b
не равно	!=	a!=b
больше	>	a>b
меньше	<	a<b
больше или равно	>=	a>=b
меньше или равно	<=	a<=b

Таблица 2.2 – Логические операторы

Оператор	Значение
&	логическое И (AND)
|	логическое ИЛИ (OR)
^	логического исключающее ИЛИ (XOR)
!	логическое отрицание НЕ (NOT)
&&	условное И (AND)
||	условное ИЛИ (OR)

C# предоставляет несколько *логических операций*, которые можно использовать для образования сложных условий. Операнды логических операторов должны относиться к типу bool, а результат выполнения логической операции также относиться к типу bool.

Логические операторы &, |, ^ и ! поддерживают основные логические операции И, ИЛИ, исключающее ИЛИ и НЕ.

Логическое И (&) возвращает true, если оба операнда дают значение true. *Логическое ИЛИ* (|) возвращает true, если хотя бы один из операндов дает значение true. Логическое отрицание *НЕ* (!) возвращает true, если выражение дает false, или false, если выражение дает true. *Логическое исключающее ИЛИ* (^) возвращает true, если один изоперандов дает значение true, а другой – false. Если оба операнда истины или ложны, то все условие будет целиком ложным.

В C# предусмотрены также условные версии (&& и ||) логических операторов (& и |). Они называются *условными И* и *ИЛИ*, так как второй операнд в них вычисляется только при необходимости. Если значение левого операнда однозначно определяет значение всего выражения, то значение второго операнда не вычисляется. В выражении x&&y значение y

не вычисляется, если x имеет значение false. В выражении x||y значение y не вычисляется, если x равно true. Операции && и || предназначены для получения более эффективного кода.

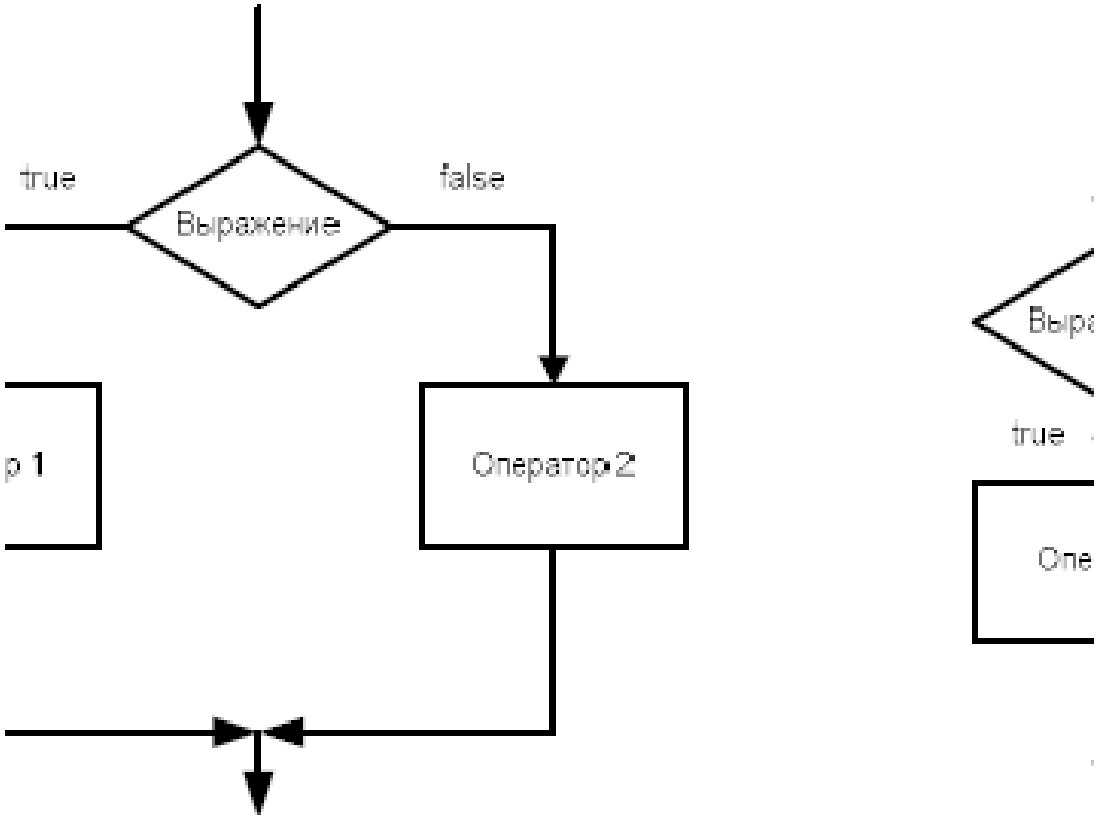
Приоритет выполнения операторов:

1) Унарные операции (операции с одним операндом, например изменение знака);

2) Операторы умножения / деления;

3) Операторы сложения / вычитания;

РИСУНОК 2.2 – СТРУКТУРНАЯ СХЕМА УСЛОВНОГО ОПЕРАТОРА



- **Операторы ветвления: оператор if и оператор switch**
- Операторы ветвления **if** и **switch** применяются для того, чтобы в зависимости от конкретных значений исходных данных обеспечить выполнение разных последовательностей операторов. Оператор **if** обеспечивает передачу управления на одну из двух ветвей вычислений, а оператор **switch** – на одну из произвольного числа ветвей.
- **Условный оператор if**
- Условный оператор if используется для разветвления процесса вычислений на два направления. Структурная схема оператора приведена на рис.2.2.

Синтаксис оператора:

if (условие) оператор_1; else оператор_2;

где if, else – зарезервированные слова, условие – это условное выражение, оператор_1, оператор_2 – любые исполняемые операторы или блоки операторов.

Условный оператор работает следующим образом. Вначале вычисляется логическое выражение. Если логическое выражение равно true, выполняется оператор_1. В противном случае, когда выражение равно false, выполняется оператор_2.

Ниже приведена общая форма оператора if, в котором используются блоки операторов.

```
 if (условие)
{
    последовательность операторов
}
else
{
    последовательность операторов
}
```

Допустима сокращенная форма условного оператора, в которой отсутствует else и оператор_2. В этом случае при ложности проверяемого условия никакие действия не выполняются, например:if(a>0) a=-a;

Ниже приведен пример простой программы, в которой операторы if и else используются для того, чтобы сообщить, является ли число положительным или отрицательным.

```
// Определить, является ли числовое значение положительным или
// отрицательным.
using System;
class PosNeg {
  static void Main() {
    int i;
    for(i=-5; i <= 5; i++) {
      Console.Write("Проверка " + i + ");
      if(i < 0) Console.WriteLine("отрицательное число”);
      else Console.WriteLine("положительное число");
    }
  }
}
```

Вложенные операторы if

Вложенным называется такой оператор if, который является адресатом другого оператора if или же оператора else. Вложенные операторы if очень часто применяются в программировании. Что же касается их применения в C#, то не следует забывать, что любой оператор else всегда связан с ближайшим оператором if, т.е. с тем оператором if, который находится в том же самом блоке, где и оператор else, но не с другим оператором else.

Рассмотрим следующий пример.

```
if (i == 10) {
  if(j < 20) a = b;
  if(k > 100) c = d;
  else a = c; // этот оператор else связан с оператором if(k > 100)
}
else a = d; // этот оператор else связан с оператором if(i == 10)
```

Как следует из комментариев к приведенному выше фрагменту кода, последний оператор else не связан с оператором if(j < 20), поскольку они не находятся в одном и том же блоке, несмотря на то, что этот оператор является для него ближайшим оператором if без вспомогательного оператора else. Напротив, последний оператор else связан с оператором if(i == 10). А внутренний оператор else связан с оператором if(k > 100), поскольку этот последний является для него ближайшим оператором if в том же самом блоке.

В приведенном ниже примере программы демонстрируется применение вложенного оператора if. В представленной ранее программе определения положительных и отрицательных чисел о нуле сообщалось как о положительном числе. Но, как правило, нуль считается числом, не имеющим знака. Поэтому в следующей версии данной программы о нуле сообщается как о числе, которое не является ни положительным, ни отрицательным.

```
// Определить, является ли числовое значение
// положительным, отрицательным или нулевым.
using System;
class  PosNegZero  {
  static void Main() {
    int i;
    for(i=-5; i <= 5; i++) {
      Console.Write("Проверка " + i + ": ");
      if(i < 0) Console.WriteLine("отрицательное число");
      else if(i == 0) Console.WriteLine("число без знака");
      else Console.WriteLine ("положительное число");
    }
  }
}
```

Конструкция if – else – if

В программировании часто применяется *многоступенчатая конструкция* if – else – if, состоящая из вложенных операторов if. Ниже приведена ее общая форма:

```
if (условие)
   оператор;
else if (условие)
   оператор;
else if (условие)
   оператор;
   ...
else
   оператор;
```

Логические выражения в такой конструкции вычисляются сверху вниз. Как только обнаружится истинное условие, выполняется связанный с ним оператор, а все остальные операторы в многоступенчатой конструкции опускаются. Если ни одно из условий не является истинным, то выполняется последний оператор else. Когда же последний оператор

else отсутствует, а все остальные проверки по условию дают ложный результат, то никаких действий вообще не выполняется.

В приведенном ниже примере программы демонстрируется применение многоступенчатой конструкции if-else-if. В этой программе обнаруживается наименьший множитель заданного целого значения, состоящий из одной цифры.

```
// Определить наименьший множитель заданного
// целого значения, состоящий из одной цифры.
using System;
class Ladder {
  static void Main() {
    int num;
    for(num = 2; num < 12; num++) {
      if((num % 2) == 0)
        Console.WriteLine("Наименьший множитель числа " + num + " равен 2.");
      else if((num % 3) == 0)
        Console.WriteLine("Наименьший множитель числа " + num + " равен 3.");
      else if((num % 5) == 0)
        Console.WriteLine("Наименьший множитель числа " + num + " равен 5.");
      else if((num % 7) == 0)
        Console.WriteLine("Наименьший множитель числа " + num + " равен 7.");
      else
        Console.WriteLine(num + " не делится на 2, 3, 5 или 7.");
    }
  }
```

Заключение

В рамках данной лекции были изучены базовые конструкции структурного программирования (следование, ветвление и цикл), а также практические инструменты языка C# для реализации условных алгоритмов: операторы сравнения, логические операторы и операторы ветвления (`if`, `if-else`, `if-else-if` и `switch`).

Понимание принципов построения условных конструкций и знание приоритета выполнения операций позволяют создавать гибкие и читаемые программы, способные изменять ход вычислений в зависимости от входных данных. Особое внимание при написании кода следует уделять использованию условных логических операторов (`&&`, `||`) для повышения производительности, а также корректному применению фигурных скобок при вложении операторов `if`, чтобы избежать логических ошибок при связывании веток `else`.

Вопросы для самопроверки

1. **Структурное программирование:** Назовите три базовые конструкции структурного программирования. Какова главная цель их применения при разработке программного обеспечения?

2. **Логические операторы:** В чем заключается принципиальное отличие между побитовыми/логическими операторами `&`, `|` и их условными версиями `&&`, `||`? Почему условные операторы считаются более эффективными?

3. **Исключающее ИЛИ:** При каких условиях логический оператор `^` (исключающее ИЛИ) возвращает значение `true`, а при каких — `false`?

4. **Приоритет операций:** В каком порядке выполняются унарные операции, арифметические операции, операции сравнения и логические операции в сложных выражениях без скобок?

5. **Оператор ветвления `if`:** Чем полная форма условного оператора `if-else` отличается от сокращенной? Что происходит в сокращенной форме, если условие оказывается ложным?

6. **Вложенные условия:** Каким правилом руководствуется компилятор C#, определяя, к какому именно оператору `if` относится оператор `else` во вложенных конструкциях? Какую роль здесь играют фигурные скобки `{}`?

Список используемых источников

1. **Троелсен Э., Джепикс Ф.** Язык программирования C# и платформа .NET. — СПб.: Диалектика, 2022. — 1312 с. *(Раздел: Управляющие конструкции и логические операторы).*

2. **Шилдт Г.** C# 4.0: полное руководство. — М.: Вильямс, 2019. — 1056 с. *(Глава 3: Операторы; Глава 5: Управляющие операторы — подробный разбор вложенных* `if` *и* `switch`*).*

3. **Албахари Д., Албахари Б.** C#. Справочник. Полное описание языка. — СПб.: БХВ-Петербург, 2023. — 1088 с. *(Таблица приоритетов операторов и спецификация инструкций ветвления).*

4. **Павловская Т. А.** C#. Программирование на языке высокого уровня: Учебник для вузов. — СПб.: Питер, 2020. — 432 с. *(Глава: Базовые конструкции структурного программирования и условные операторы).*

5. **Официальная документация Microsoft Learn:** Операторы выбора — `if`, `if-else` и `switch` в C# [Электронный ресурс]. — URL: microsoft.com (дата обращения: 13.11.2026).