# Тема 4. Условный оператор. Оператор выбора. Логические операторы.

# Учебные вопросы:

1. Условный оператор if-else.
2. Вложенные условные операторы
3. Оператор выбора switch-case.
4. Логические операторы.

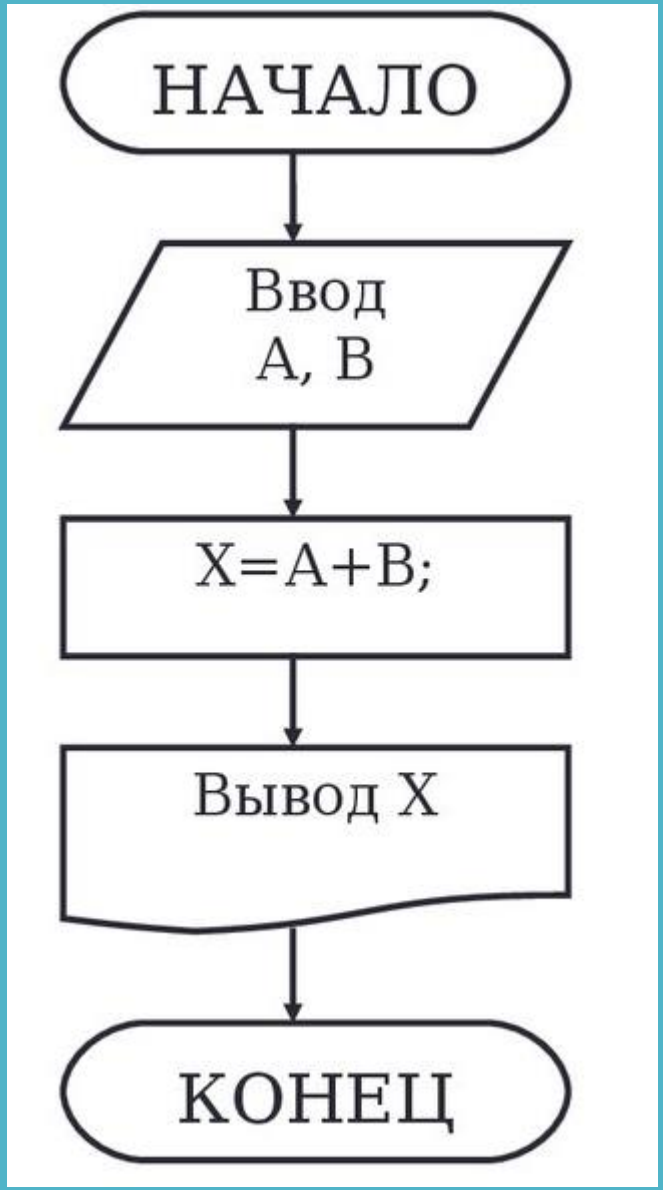
# 1. Условный оператор if-else.

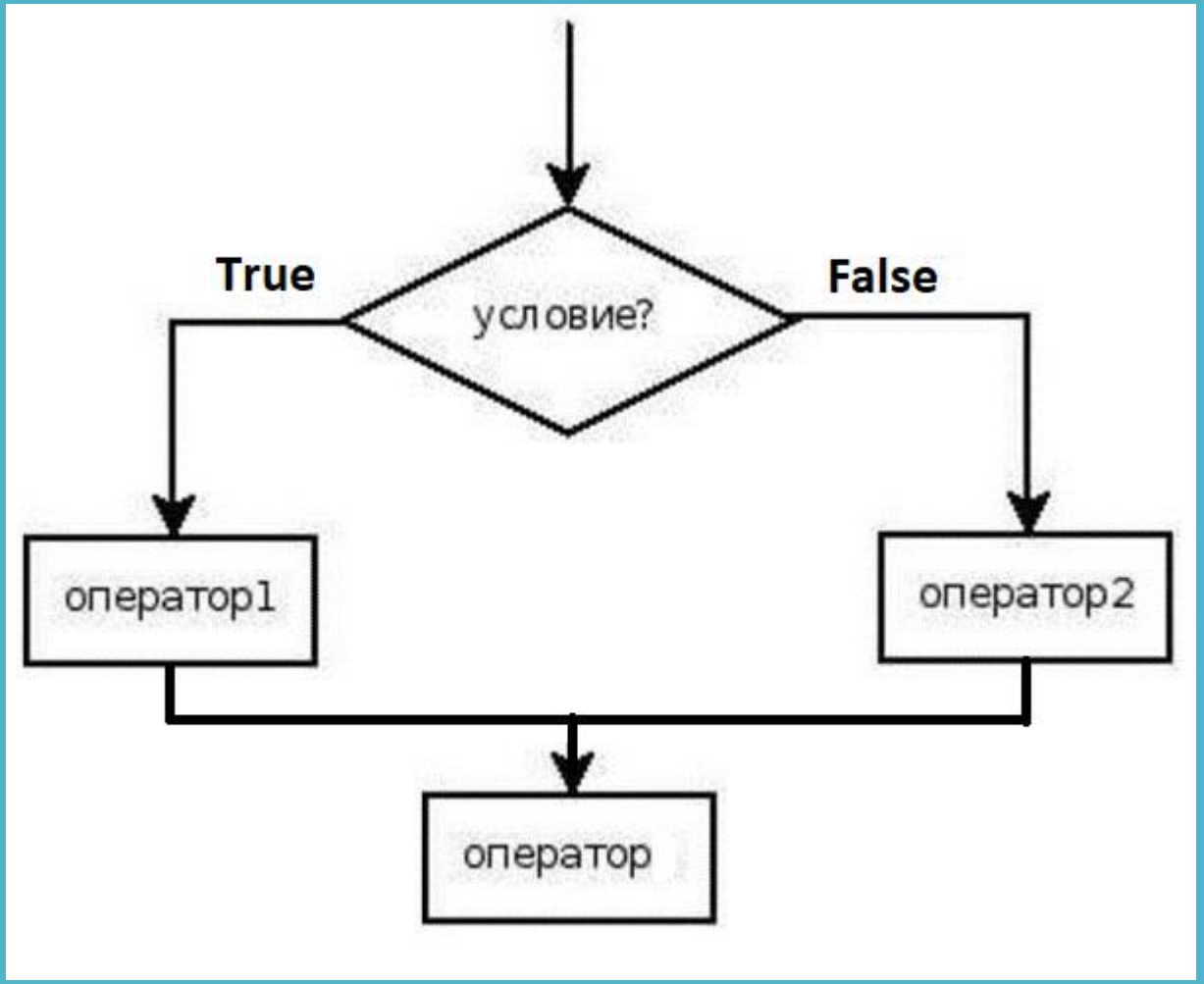
**Условный оператор** — это управляющая структура в программировании, которая позволяет выполнять определенные действия в зависимости от выполнения или невыполнения определенного условия.

Условные операторы применяются для принятия решений внутри программы. Они позволяют программе реагировать на различные входные данные или ситуации и изменять свое поведение на основе этих данных.

Условный оператор проверяет логическое выражение или условие. Если это условие истинно (**true**), программа выполняет один блок кода. Если условие ложно (**false**), может выполняться другой блок кода или программа может перейти к следующей инструкции.

# Линейный и разветвляющийся алгоритм



**Типы условных операторов.**

В большинстве языков программирования, включая C#, существуют различные виды условных операторов:

1. Оператор **if**: Проверяет условие и выполняет блок кода, если условие истинно. Если ложно, не выполняется ничего.

```
if (x > 0)
{
    Console.WriteLine("x положительное число");
}
```

Основной синтаксис оператора **if** в C# выглядит следующим образом:

```
if (условие) {
    // Код, который выполнится, если условие истинно
}
```

**Условие** — это логическое выражение, которое возвращает значение true или false.

**Код в фигурных скобках** — это блок кода, который выполняется, если условие истинно.

Если код умещается в одной строке, фигурные скобки можно не использовать.

```
if (x > 0)
{
    Console.WriteLine("x положительное число");
}
```

```
if (x > 0)
    Console.WriteLine("x положительное число");
```

```
if (x > 0) Console.WriteLine("x положительное число");
```

```
bool isLogin = true;

if (isLogin == true)
    Console.WriteLine("Вы вошли в систему!");

// Если переменная имеет булевый тип можно записать короче:
if (isLogin)
    Console.WriteLine("Вы вошли в систему!");

if (isLogin == false)
    Console.WriteLine("Неверный логин или пароль!");

if (!isLogin)
    Console.WriteLine("Неверный логин или пароль!");
```

2. Оператор **if-else**: Проверяет условие и выполняет один блок кода, если условие истинно, и другой блок кода, если условие ложно.

```
if (x > 0)
{
    Console.WriteLine("x положительное число");
}
else
{
    Console.WriteLine("x неположительное число");
}
```

3. Оператор **else if**: Позволяет проверить несколько условий последовательно.

```
if (x > 0)
{
    Console.WriteLine("x положительное число");
}
else if (x < 0)
{
    Console.WriteLine("x отрицательное число");
}
else
{
    Console.WriteLine("x равно нулю");
}
```

Условия проверяются последовательно, сверху вниз, выполнится только от блок кода, в котором условие выполнится первым.

4. Тернарный оператор **?**: Условный оператор в одной строке, который используется для простых условий.

```
string result = (x > 0) ? "положительное" : "отрицательное";
```

В переменную **result** запишется строка «положительное», если условие выполнится, в противном случае запишется «отрицательное».

# 2. Вложенные условные операторы

Условные операторы могут быть вложены друг в друга для создания более сложных логических конструкций:

```
if (условие1) {
    if (условие2) {
        // Код, который выполнится, если условие1 и условие2 истинны
    } else {
        // Код, который выполнится, если условие1 истинно, а условие2 ложно
    }
} else {
    // Код, который выполнится, если условие1 ложно
}
```

# 3. Оператор выбора switch-case.

3. Оператор **switch-case** используется для выбора одного из нескольких возможных блоков кода для выполнения на основе значения выражения. Он может быть полезен, когда одна переменная или выражение должны быть проверены на несколько различных значений.

# Синтаксис оператора swich-case:

```
switch (выражение) {
    case значение1:
        // Код, который выполнится, если выражение равно значение1
        break;
    case значение2:
        // Код, который выполнится, если выражение равно значение2
        break;
    // Другие case
    default:
        // Код, который выполнится, если выражение не совпало ни с одним значением case
        break;
}
```

```
string command = "start";
switch (command)
{
    case "start":
        Console.WriteLine("Запуск программы");
        break;
    case "stop":
        Console.WriteLine("Остановка программы");
        break;
    case "pause":
        Console.WriteLine("Пауза программы");
        break;
    default:
        Console.WriteLine("Неизвестная команда");
        break;
}
```

# 4. Логические операторы.

**Логические операторы** используются для выполнения логических операций над булевыми значениями (true или false). Они позволяют создавать сложные логические выражения, которые могут быть использованы в условных операторах, циклах и других конструкциях.

Основные логические операторы в C#

Оператор И (&&)

Оператор ИЛИ (||)

Оператор НЕ (!)

Оператор ИСКЛЮЧАЮЩЕЕ ИЛИ (^)

Оператор **И (&&)** возвращает **true**, если оба операнда истинны. Если хотя бы один операнд ложен, результат будет **false**.

```
bool isAdult = true;
bool hasTicket = true;


if (isAdult && hasTicket) {
    Console.WriteLine("Вход разрешен.");
} else {
    Console.WriteLine("Вход запрещен.");
}
```

Оператор **ИЛИ (||)** возвращает **true**, если хотя бы один из операндов истинен. Если оба операнда ложны, результат будет **false**.

```
bool isWeekend = true;
bool isHoliday = false;


if (isWeekend || isHoliday) {
    Console.WriteLine("Сегодня выходной.");
} else {
    Console.WriteLine("Сегодня рабочий день.");
}
```

Оператор **НЕ (!)** инвертирует значение операнда. Если операнд равен **true**, результат будет **false**, и наоборот.

```
bool isRaining = false;


if (!isRaining) {
    Console.WriteLine("Сегодня не идет дождь.");
} else {
    Console.WriteLine("Сегодня идет дождь.");
}
```

Оператор **ИСКЛЮЧАЮЩЕЕ ИЛИ (^)** возвращает **true**, если одно из значений истинно, а другое ложно. Если оба значения одинаковы (оба истинны или оба ложны), результат будет **false**.

```
bool isMember = true;
bool hasCoupon = false;


if (isMember ^ hasCoupon) {
    Console.WriteLine("Вы получите дополнительную скидку.");
} else {
    Console.WriteLine("Скидка не предоставлена.");
}
```

# Таблица истинности.

| Операция | Операнд1 | Операнд2 | Результат |
|---|---|---|---|
| И (&&) | TRUE | TRUE | TRUE |
| | TRUE | FALSE | FALSE |
| | FALSE | TRUE | FALSE |
| | FALSE | FALSE | FALSE |
| ИЛИ (\|\|) | TRUE | TRUE | TRUE |
| | TRUE | FALSE | TRUE |
| | FALSE | TRUE | TRUE |
| | FALSE | FALSE | FALSE |
| ИСКЛЮЧАЮЩЕЕ ИЛИ (^) | TRUE | TRUE | FALSE |
| | TRUE | FALSE | TRUE |
| | FALSE | TRUE | TRUE |
| | FALSE | FALSE | FALSE |
| НЕ (!) | TRUE | | FALSE |
| | FALSE | | TRUE |

# Комбинирование логических операторов

Логические операторы можно комбинировать для создания более сложных условий:

```
int age = 20;
bool hasLicense = true;


if ((age >= 18 && hasLicense) || age >= 65) {
    Console.WriteLine("Вы можете управлять автомобилем.");
} else {
    Console.WriteLine("Вы не можете управлять автомобилем.");
}
```

**Приоритет операций**

Оператор **!** имеет более высокий приоритет по сравнению с **&&** и **||**.

Оператор **&&** имеет более высокий приоритет по сравнению с **||**.

Это означает, что выражения с **!** будут вычислены первыми, затем выражения с **&&**, а потом с **||**. При необходимости можно использовать скобки для явного указания порядка выполнения операций.

# Контрольные вопросы:

- Что такое условный оператор и для чего он используется в программировании?
- Опишите синтаксис оператора if в языке C#. Когда можно опустить фигурные скобки?
- Чем отличается оператор if-else от простого if? Приведите пример.
- Как используются вложенные условные операторы? Приведите пример их применения.
- В каких случаях предпочтительнее использовать оператор выбора switch-case вместо if-else?
- Перечислите основные логические операторы в C# и кратко опишите их функции.
- Каков результат применения логического оператора И (&&), если один из операндов ложен?
- Что делает логический оператор НЕ (!)? Приведите пример.
- Объясните, что означает приоритет операций в контексте логических операторов. Почему он важен?
- Как можно комбинировать логические операторы для создания более сложных условий? Приведите пример.

# Материалы лекций:

https://github.com/ShViktor72/Education2025