**Conditional Operators (Умовні оператори)**

**Умови та оператори If в C#**

C# підтримує звичайні логічні умови з математики:

* **Менше: a < b**
* **Менше або дорівнює: a <= b**
* **Більше ніж: a > b**
* **Більше або дорівнює: a >= b**
* **Рівне: a == b**
* **Не дорівнює: a != b**

Ви можете використовувати ці умови для виконання різних дій для різних рішень.

У мові C# є наступні умовні оператори:

* **Використовуйте *if*, щоб вказати блок коду, який буде виконано, якщо вказана умова є істинною.**
* **Використовуйте else, щоб вказати блок коду, який потрібно виконати, якщо умова *if* є хибною.**
* **Використовуйте *else if*, щоб вказати нову умову для перевірки, якщо перша умова є хибною**
* **Використовуйте *switch*, щоб вказати багато альтернативних блоків коду для виконання**

**Оператор if (The if Statement)**

Використовуйте інструкцію if, щоб вказати блок коду C#, який буде виконано, якщо умова має значення True.

**Syntax:**

if (*condition*)

{

*// block of code to be executed if the condition is True*

}

У прикладі нижче ми тестуємо два значення, щоб з'ясувати, чи є 20 більшим за 18. Якщо умова істинна, то вивести деякий текст:

if (20 > 18)

{

Console.WriteLine("20 is greater than 18");

}

int x = 20;

int y = 18;

if (x > y)

{

Console.WriteLine("x is greater than y");

}

**Оператор else(The else Statement)**

Використовуйте оператор else, щоб вказати блок коду, який буде виконано, якщо умова має значення False.

### **Syntax:**

if (*condition*)

{

*// block of code to be executed if the condition is True*

}

else

{

*// block of code to be executed if the condition is False*

}

int time = 20;

if (time < 18)

{

Console.WriteLine("Good day.");

}

else

{

Console.WriteLine("Good evening.");

}

// Outputs "Good evening."

У наведеному вище прикладі час (20) більший за 18, тому умова має значення False. Тому ми переходимо до умови else і виводимо на екран "Добрий вечір". Якби час був меншим за 18, програма вивела б "Доброго дня".

**Оператор else if (The else if Statement)**

Використовуйте оператор else if, щоб вказати нову умову, якщо перша умова має значення False.

### **Syntax:**

if (*condition1*)

{

*// block of code to be executed if condition1 is True*

}

else if (*condition2*)

{

*// block of code to be executed if the condition1 is false and condition2 is True*

}

else

{

*// block of code to be executed if the condition1 is false and condition2 is False*

}

Namespace MyApplication

{

class Program

{

static void Main(string[] args)

{

int time = 21;

if (time < 10)

{

Console.WriteLine("Good morning.");

}

else if (time < 15)

{

Console.WriteLine("Good day.");

}

else if (time < 20)

{

Console.WriteLine("Good evening.");

}

else

{

Console.WriteLine("Good night.");

}

}

}

}

У наведеному вище прикладі час (22) більший за 10, тому перша умова рівна False. Наступна умова в операторі else if також має значення False, тому ми переходимо до умови else, оскільки умова1 (condition1) і умова2 (condition2) мають значення False - і виводимо на екран "Добрий вечір". Однак, якби час був 14, наша програма вивела б "Добрий день".

## **Оператори Switch(C# Switch Statements)**

Використовуйте оператор switch для вибору одного з багатьох блоків коду для виконання.

### **Syntax:**

switch(*expression*)

{

case x:

*// code block*

break;

case y:

*// code block*

break;

default:

*// code block*

break;

}

Ось як це працює:

* Вираз switch обчислюється один раз
* Значення виразу порівнюється зі значеннями кожного випадку
* Якщо є збіг, виконується відповідний блок коду

### **Example:**

int day = 4;

switch (day)

{

case 1:

Console.WriteLine("Monday");

break;

case 2:

Console.WriteLine("Tuesday");

break;

case 3:

Console.WriteLine("Wednesday");

break;

case 4:

Console.WriteLine("Thursday");

break;

case 5:

Console.WriteLine("Friday");

break;

case 6:

Console.WriteLine("Saturday");

break;

case 7:

Console.WriteLine("Sunday");

break;

}

## **Ключове слово break (The break Keyword)**

Якщо C# досягає ключового слова break, вона виходить з блоку switch. Це зупиняє виконання більшої частини коду і тестування варіанту всередині блоку. Коли збіг знайдено і роботу виконано, настає час перерви. Немає необхідності в подальшому тестуванні.

Break може заощадити багато часу на виконанні, оскільки він "ігнорує" виконання решти коду у блоці switch.

**Ключове слово default** (**The default Keyword**)

The default keyword is optional and specifies some code to run if there is no case match.

Ключове слово default є необов'язковим і вказує на певний код, який буде запущено, якщо немає збігу регістру:

int day = 4;

switch (day)

{

case 6:

Console.WriteLine("Today is Saturday.");

break;

case 7:

Console.WriteLine("Today is Sunday.");

break;

default:

Console.WriteLine("Looking forward to the Weekend.");

break;

}

# **C# Break and Continue**

## 

## **C# Break**

Ви вже бачили оператор break, який використовувався у попередньому розділі цього підручника. Він використовувався для того, щоб "вискочити" з оператора switch.

Оператор break також можна використовувати для виходу з циклу.

У цьому прикладі ми вийдемо з циклу, коли i стане рівним 4:

for (int i = 0; i < 10; i++)

{

if (i == 4)

{

break;

}

Console.WriteLine(i);

}

## **C# Continue**

Оператор continue перериває одну ітерацію (у циклі), якщо виникає вказана умова, і продовжує виконання наступної ітерації у циклі.

У цьому прикладі пропускається значення 4:

for (int i = 0; i < 10; i++)

{

if (i == 4)

{

continue;

}

Console.WriteLine(i);

}

## **Break and Continue in While Loop**

Ви також можете використовувати break та continue у циклах while:

### **Break Example:**

int i = 0;

while (i < 10)

{

Console.WriteLine(i);

i++;

if (i == 4)

{

break;

}

}

### **Continue Example:**

int i = 0;

while (i < 10)

{

if (i == 4)

{

i++;

continue;

}

Console.WriteLine(i);

i++;

}

# **C# For Loop**

Коли ви точно знаєте, скільки разів ви хочете пройти через блок коду, використовуйте цикл for замість циклу while:

### **Syntax:**

for (*statement 1*; *statement 2*; *statement 3*)

{

*// code block to be executed*

}

* **Оператор 1 виконується (один раз) перед виконанням блоку коду.**
* **Оператор 2 визначає умову виконання блоку коду.**
* **Оператор 3 виконується (кожного разу) після виконання блоку коду.**
* **У наведеному нижче прикладі буде виведено числа від 0 до 4:**

for (int i = 0; i < 5; i++)

{

Console.WriteLine(i);

}

Пояснення до прикладу:

* **Оператор 1 встановлює змінну перед початком циклу (int i = 0).**
* **Оператор 2 визначає умову для запуску циклу (i має бути менше 5). Якщо умова істинна, то цикл почнеться знову, якщо хибна, то цикл завершиться.**
* **Оператор 3 збільшує значення (i++) кожного разу, коли виконується блок коду в циклі.**

## **Nested Loops**

Також можна розмістити цикл всередині іншого циклу.

Це називається вкладеним циклом.

"Внутрішній цикл" буде виконуватися один раз для кожної ітерації "зовнішнього циклу":

// Outer loop

for (int i = 1; i <= 2; ++i)

{

Console.WriteLine("Outer: " + i); // Executes 2 times

// Inner loop

for (int j = 1; j <= 3; j++)

{

Console.WriteLine(" Inner: " + j); // Executes 6 times (2 \* 3)

}

}