# **try-catch**

Оператор **try-catch** складається з блоку try, за яким слідує один або декілька операторів catch, які визначають обробники для різних винятків.

Коли виникає виключення, середовище виконання звичайної мови (**Common Language Runtime**) шукає оператор **catch**, який обробляє це виключення. Якщо поточний метод, що виконується, не містить такого блоку перехоплення, **CLR** шукає метод, який викликав поточний метод, і так далі вгору по стеку викликів. Якщо блок **catch** не знайдено, то **CLR** виводить користувачеві повідомлення про необроблену виняткову ситуацію і зупиняє виконання програми.

Блок try містить захищений код, який може спричинити виключення. Блок виконується до тих пір, поки не буде згенеровано виключення або поки він не буде успішно завершений.

**Try-catch в C#** - це механізм обробки винятків, який дозволяє програмі перехопити та обробити помилки під час виконання.

* У блоку **try** розміщується код, який потенційно може викликати помилки. Якщо помилка виникає в цьому блоку, виконання коду припиняється та виконується блок **catch**, де визначена логіка обробки помилки.
* У блокі **catch** можна вказати, який тип помилок має бути перехоплено, та що робити в разі їх виникнення.

Наприклад, наступний код перевіряє, чи може змінна x бути поділена на змінну y. Якщо y дорівнює нулю, виникає помилка, яка перехоплюється блоком **catch**:

try

{

int x = 10;

int y = 0;

int z = x / y;

}

catch (DivideByZeroException ex)

{

Console.WriteLine("Помилка: " + ex.Message);

}

У цьому випадку, якщо y дорівнює нулю, виконання коду припиняється та виводиться повідомлення про помилку "Помилка: Attempted to divide by zero.".

Розглянемо наступний приклад, де ми створюємо масив із трьох цілих чисел:

This will generate an error, because **myNumbers[10]** does not exist.

int[] myNumbers = {1, 2, 3};

Console.WriteLine(myNumbers[10]); // error!

The error message will be something like this:

System.IndexOutOfRangeException: 'Index was outside the bounds of the array.'

Якщо виникає помилка, ми можемо використати **try...catch** щоб перехопити помилку і виконати деякий код для її обробки.

У наступному прикладі ми використовуємо змінну всередині блоку **catch (e)** разом з вбудованою властивістю **Message**, яка виводить повідомлення, що описує виключення:

try

{

int[] myNumbers = {1, 2, 3};

Console.WriteLine(myNumbers[10]);

}

catch (Exception e)

{

Console.WriteLine(e.Message);

}

**try-finally**

**Try-finally в C#** - це механізм обробки винятків, який дозволяє виконати певний код незалежно від того, чи виникла помилка під час виконання програми.

* У блоку **try** розміщується код, який потенційно може викликати помилки.
* Якщо помилка виникає в цьому блоку, виконання коду припиняється та виконується блок **finally**, де визначена логіка, яка повинна бути виконана, незалежно від того, чи виникла помилка.
* Якщо помилка не виникає, блок **finally** виконується після блоку **try**.

Блок **finally** виконується, коли блок **try/catch** завершує виконання, незалежно від того, яка умова його викликала. Він завжди виконується незалежно від того, чи блок **try** завершується нормально, чи завершується через виключення. Основна мета блоку **finally** - звільнити системні ресурси. Блок **finally** слідує за блоком **try/catch.**

Наприклад, наступний код записує вміст файлу у змінну та закриває файл незалежно від того, чи виникає помилка:

FileStream file = null;

try

{

file = new FileStream("file.txt", FileMode.Open);

byte[] buffer = new byte[file.Length];

file.Read(buffer, 0, (int)file.Length);

Console.WriteLine(Encoding.Default.GetString(buffer));

}

finally

{

if (file != null)

{

file.Close();

}

}

У цьому випадку, якщо файл ***"file.txt"*** успішно відкрито і прочитано, блок **finally** виконається після блоку **try** та закриє файл. Якщо виникає помилка під час виконання блоку **try**, блок **finally** також виконається, щоб закрити файл перед виходом з програми.

У наступному прикладі некоректний оператор перетворення викликає виключення **System.InvalidCastException.** Виключення не обробляється.

public class ThrowTestA

{

public static void Main()

{

int i = 123;

string s = "Some string";

object obj = s;

try

{

// Invalid conversion; obj contains a string, not a numeric type.

i = (int)obj;

// The following statement is not run.

Console.WriteLine("WriteLine at the end of the try block.");

}

finally

{

// To run the program in Visual Studio, type CTRL+F5. Then

// click Cancel in the error dialog.

Console.WriteLine("\nExecution of the finally block after an unhandled\n" +

"error depends on how the exception unwind operation is triggered.");

Console.WriteLine("i = {0}", i);

}

}

// Output:

// Unhandled Exception: System.InvalidCastException: Specified cast is not valid.

//

// Execution of the finally block after an unhandled

// error depends on how the exception unwind operation is triggered.

// i = 123

}

**Важливі моменти:**

* У мові C# заборонено використовувати декілька блоків **finally** в одній програмі.
* Блок **finally** не містить операторів return, continue, break, оскільки він не дозволяє елементам управління залишати блок **finally**.
* Ви також можете використовувати блок finally тільки з блоком **try** без блоку **catch**, але в цьому випадку виключення не обробляються.
* Блок **finally** буде виконано після блоків **try** і **catch**, але до того, як елемент керування повернеться до свого початкового стану.

**Try-catch-finally**

Try-catch-finally в C# - це механізм обробки винятків, який дозволяє виконувати певні дії при виникненні помилки та після її виникнення, незалежно від того, чи була помилка виконана.

* У блоку try розміщується код, який потенційно може викликати помилки. Якщо помилка виникає в цьому блоку, виконання коду припиняється та виконується блок catch, де визначена логіка обробки помилки.
* У блоці catch можна вказати, який тип помилок має бути перехоплено, та що робити в разі їх виникнення.
* Після блоку try-catch може бути блок finally, в якому можна визначити логіку, яка повинна бути виконана незалежно від того, чи виникала помилка в блоку try.
* Блок finally завжди виконується після блоку try-catch, незалежно від того, чи виникла помилка, чи ні.

Наприклад, наступний код записує вміст файлу у змінну та закриває файл, незалежно від того, чи виникає помилка під час відкриття файлу чи читання даних з файлу:

FileStream file = null;

try

{

file = new FileStream("file.txt", FileMode.Open);

byte[] buffer = new byte[file.Length];

file.Read(buffer, 0, (int)file.Length);

Console.WriteLine(Encoding.Default.GetString(buffer));

}

catch (Exception ex)

{

Console.WriteLine("Помилка: " + ex.Message);

}

finally

{

if (file != null)

{

file.Close();

}

}

У цьому випадку, якщо файл "file.txt" успішно відкрито і прочитано, блок **finally** виконається після блоку **try** та закриє файл. Якщо виникає помилка під час виконання блоку **try**, блок **catch** перехоплює помилку та виводить повідомлення про помилку, а блок **finally** закриє файл перед виходом з програми.