Podstawy programowania

mgr inż. Katarzyna Dadek

2024

Lista 4
Podstawy C# cz.2

Platforma .NET umożliwia korzystanie z zaawansowanych funkcji języka C#, takich jak tablice, struktury, wyjątki oraz mechanizmy obsługi błędów. Dzięki nim aplikacje mogą przetwarzać i przechowywać dane, reagować na sytuacje wyjątkowe i wyświetlać komunikaty na konsoli. Wprowadzenie do tych zagadnień pozwoli na bardziej kompleksowe zrozumienie możliwości języka C# i platformy .NET.

4.1 Tablice

Tablice to struktury danych, które przechowują zbiór elementów **tego samego typu**. Są szczególnie użyteczne do przechowywania dużych ilości danych w jednym miejscu. Tablice w C# mogą mieć jeden lub więcej wymiarów i są indeksowane od **zera**.

- Deklaracja i inicjalizacja tablicy jednowymiarowej:

```
int[] numbers = new int[5];
int[] initializedNumbers = {1, 2, 3, 4, 5};
```

- Tablice wielowymiarowe:

```
int[,] matrix = new int[2, 3];
```

4.2 Struktury

Struktury (ang. **struct**) to typy wartościowe pozwalające grupować dane różnego typu. Są często wykorzystywane do definiowania prostych obiektów, które nie wymagają pełnej funkcjonalności klas.

- Definicja struktury:

```
struct Point {
 public int X;
 public int Y;

 public Point(int x, int y) {
  X = x;
  Y = y;
 }
}
```

4.3 We/Wy w Konsoli

Wejście i wyjście w konsoli pozwalają na interakcję z użytkownikiem poprzez wprowadzanie danych oraz ich wyświetlanie. W C# standardowe wejście to metoda `Console.ReadLine()`, a wyjście to `Console.WriteLine()`.

```
Console.Write("Your name: ");
string name = Console.ReadLine();
Console.WriteLine("Hello, " + name + "!");
```

4.3.1 Obsługa Błędów We/Wy

Podczas korzystania z We/Wy mogą wystąpić błędy, takie jak brak dostępu do pliku czy błędny format danych. Obsługa błędów umożliwia reagowanie na te sytuacje i zapobiega awarii programu. W C# do obsługi błędów stosuje się bloki **`try-catch`**.

```
try {
 int value = int.Parse(Console.ReadLine());
} catch (FormatException) {
 Console.WriteLine("Invalid number!");
}
```

4.4 Wyjątki

Wyjątki to specjalne obiekty przechowujące informacje o błędach w kodzie. Wyjątki można przechwytywać i obsługiwać za pomocą instrukcji **`try-catch`**. C#

obsługuje różne typy wyjątków, takie jak *NullReferenceException*, *IndexOutOfRangeException* i inne.

4.5 Kod nienadzorowany

Kod nienadzorowany (ang. *unsafe code*) pozwala na bezpośrednią manipulację wskaźnikami i jest używany w sytuacjach wymagających wysokiej wydajności, np. w pracy z pamięcią. Kod taki musi być oznaczony słowem kluczowym `unsafe` i może być kompilowany tylko w odpowiednio skonfigurowanym środowisku.

4.6 Linki

Poniżej znajdują się przydatne linki do stron z przykładami i dodatkowymi informacjami:

- `https://www.w3schools.com/cs/cs_arrays.php`
- `https://www.w3schools.com/cs/cs_user_input.php`
- `https://www.w3schools.com/cs/cs_exceptions.php`
- `https://learn.microsoft.com/pl-pl/dotnet/csharp/linq/get-started/introduction-to-linq-queries`

4.7 Zadania

Z4.1. Zapoznaj się z materiałami powyżej - wstępem teoretycznym oraz linkami.

Z4.2. Zadeklaruj tablicę liczb całkowitych i wypełnij ją kilkoma przykładowymi danymi. Następnie wyświetl jej zawartość na konsoli.

Z4.3. Utwórz program, który deklaruje tablicę liczb całkowitych i pozwala użytkownikowi na wprowadzenie pięciu liczb. Następnie program wyświetli te liczby w odwrotnej kolejności.

Z4.4. Napisz program, który tworzy macierz o wymiarach 3x3, wypełnia ją liczbami od 1 do 9, a następnie wyświetla ją w postaci tabeli.

Z4.5. Utwórz strukturę `Punkt`, która przechowuje współrzędne `X` i `Y`. Napisz program, który tworzy dwa punkty, pobiera ich współrzędne od użytkownika i oblicza odległość między nimi.

Z4.6. Napisz program, który wczytuje nazwisko i imię użytkownika, a następnie wyświetla wiadomość w formacie: `Witaj, [Imię] [Nazwisko]`. Użyj interpolacji stringów.

Z4.7. Utwórz program, który prosi użytkownika o podanie liczby N. Następnie prosi po podanie N liczb całkowitych, a następnie wyświetla ich średnią.

Z4.8. Napisz program, który wczytuje liczbę od użytkownika, sprawdza, czy jest poprawna i wyświetla ją na ekranie. W przypadku błędnego formatu wyświetla komunikat o błędzie. Zastanów się jak najbezpieczniej sprawdzić czy podana fraza jest liczbą. Uwzględnij różne przypadki.

Z4.9. Stwórz program, który wczytuje liczbę całkowitą i dzieli ją przez liczbę podaną przez użytkownika. Obsłuż wyjątek dzielenia przez zero.

Z4.10. Utwórz program, który pobiera wartość z tablicy, której rozmiar zdefiniowany jest na stałe. Obsłuż wyjątek, jeśli użytkownik poda indeks spoza zakresu.

Z4.11. Utwórz program, który tworzy tablicę liczb całkowitych i oblicza sumę jej elementów przy użyciu wskaźników. Program powinien działać w trybie `unsafe`.

Z4.12. Zadeklaruj tablicę 15 liczb całkowitych i wypełnij ją kilkoma przykładowymi danymi. Następnie napisz program, który oblicza sumę wszystkich elementów w danej tablicy. Użyj metody .Sum() z Linq.

Z4.13. Zadeklaruj tablicę 5 liczb całkowitych i wypełnij ją kilkoma przykładowymi danymi. Następnie oblicz średnią arytmetyczną wszystkich elementów w tablicy. Użyj metody .Average() z Linq.

Z4.14. Narysuj w konsoli wypełniony prostokąt z gwiazdek o szerokości 5 i wysokości 3.

Z4.15. Narysuj w konsoli wypełniony prostokąt z gwiazdek o szerokości i wysokości podanej przez użytkownika ale nie większych niż 100 i nie mniejszych niż 3.

4.8 Zadanie domowe

Na następnych zajęciach odbędzie się pierwsze kolokwium. Proszę zapoznać się z materiałami, linkami i zadaniami jakie do tej pory pojawiły się na zajęciach. Dodatkowo przydatne mogą okazać się poniższe linki - w ramach powtórki materiału.

- Value types

- Integral numeric types
- Floating-point numeric types
- Numeric conversions
- bool
- char
- enum
- struct
- ref struct
- Value tuples
- Nullable value types
- Built-in types
- Unmanaged types
- Default values
- Operators overview
- Arithmetic operators
- Boolean logical operators
- Bitwise and shift operators
- Collection expressions
- Equality operators
- Comparison operators
- Member access operators
- Type testing and cast operators
- User-defined conversion operators
- Pointer-related operators
- Assignment operator

- Lambda expressions
- Subtraction operator
- Addition operator
- Conditional operator
- Null-coalescing operator
- Lambda operator
- is operator
- new operator
- sizeof operator
- true/false operators
- Operator overloading
- String interpolation
- Unsafe code
- Exception overview
- Exception class and properties
- Using try-catch
- Specific exceptions
- Throwing exceptions
- User-defined exceptions
- Finally blocks
- Best practices for exceptions