C# / .NET

Ogólne

Opracowana przez Microsoft platforma programistyczna ***.NET Framework*** jest odpowiednikiem platformy Java. Obejmuje gotowe biblioteki, kompilatory oraz środowisko uruchomieniowe ***CLR*** (*Common Language Runtime*), które jest odpowiednikiem wirtualnej maszyny Java i zajmuje się „odśmiecaniem pamięci" po nieużywanych obiektach. Środowiskiem programistycznym jest *Visual Studio*. Aplikacje .NET nie będą działały bez środowiska .NET. Istnieje wersja .NET dla systemu Linux i jest to ***MONO*** (projekt firmy *Novell* na licencji *open source* bazujący na środowisku programistycznym *MonoDevelop*), ***dotGNU*** lub ***.NET Core***.

Aplikacje do .NET można pisać w językach *C++/CLI*, *C#* (*si szarp* - ulepszona wersja C++, możliwy jest kod asynchroniczny, nie ma funkcji globalnych), *F#*, *Delphi.NET*, *Perl*, *Phyton*. Kod (plik z rozszerzeniem *.cs) przekształcany jest do postaci kodu pośredniego ***CIL*** (*Common Intermediate Language*), a pliki zawierające taki kod mają rozszerzenie *.*il*. Kod pośredni kompilowany jest do postaci binarnej „w locie" (*just-in-time*) przy pierwszym wywołaniu.

Bezpośrednie uruchomienie wersji CIL (z rozszerzeniem *.*exe*) w systemie bez zainstalowanego *.NET Framework, .NET Core*[1] lub otwartego *Mono*, *DotGNU* – nie jest możliwe.

Aplikacje mogą działać albo stacjonarnie w systemie albo po stronie serwera IIS (*ASP.NET*).

Visual Studio

Visual Studio Community jest darmowym[2] środowiskiem programistycznym dla języka C# (także innych języków), obsługującego system kontroli wersji GIT. Istnieje także wersja dla systemu Linux o nazwie *Visual Studio Code,* lecz jest znacznie uboższa w funkcje (w zasadzie jest to notatnik z funkcją kolorowania składni).

Strukturą nadrzędną wobec projektu jest „Rozwiązanie" („Solution"), które może zawierać w sobie wiele projektów. Aby stworzyć nowe „Solution" wraz z nowym projektem: Plik / Nowy projekt. Aby dodać do istniejącego „Solution" nowy projekt: PPM na nazwie „Solution" (menadżer po prawej stronie) / Dodaj / Nowy projekt.

Po napisaniu kodu należy go skompilować (kompilujemy całe „Solution", bądź pojedynczy projekt). Ścieżka do skompilowanego pliku to:

~/source/repos/nazwaSolution/nazwaProjektu/bin/Debug/program.exe.

Skompilowany projekt można uruchomić: Debugowanie / Uruchom bez debugowania. Można także użyć jednocześnie kompilacji i uruchomienia poprzez kliknięcie w przycisk „Rozpocznij".

Aby kompilowany i uruchamiany był konkretny projekt należy PPM kliknąć na jego nazwie i włączyć „Ustaw jako projekt startowy".

Aby dodać do projektu Nugeta (cały pakiet bibliotek o odpowiedniej wersji): PPM na nazwieProjektu / Zarządzaj pakietami NuGet...

Aby pisać aplikację pod .NET w języku C++: Plik / Nowy projekt / Online / Szablony / Visual C++ / Rozszerzenia programu Visual Studio / C++/CLR Windows Forms.

- Wybór języka C++: *New Peoject / Other Language / CLR*.
- Ustawianie okna startowego: *PPM na WindowsFormsApp1*.
- Dodanie projektu do istniejącego „rozwiązania" (solution): *PPM → Add Project*.
- Dodanie bibliotek: *PPM na programie / Referencje (odwołania) / Dodaj odwołanie*.

[1] Wersja *.NET* dla systemu *Linux* utworzona przez firmę *Microsoft*.

[2] Darmowym dla maksymalnie 5-osobowego zespołu programistycznego (jeden z warunków) lub dla dochodów poniżej miliona dolarów.

- Podczas pisania kodu, działa mechanizm IntelliSense (podpowiada kod zależnie od kontekstu).
- Kompilowanie całego projektu: *Kompiluj rozwiązanie (solution)*. Można także kliknąć w długi przycisk "*Local Windows Debugger*".
- Uruchamianie skompilowanego programu: *Debugowanie → Uruchom bez debugowania*.
- Plik binarny znajduje się w folderze: *bin/Debug/program.exe*.
- Nuget – pakiet bibliotek; aby nimi zarządzać: PPM na projekcie.
- Garbage Collector - mechanizm, który automatycznie zwalnia obiekty z pamięci (w C++ trzeba to robić samemu, ręcznie).
- Aplikacje okienkowe możemy tworzyć jako stare "*Windows Form*" (sięgające jeszcze czasów *Windows 3.11*) lub jako lepsze wektorowe "*WPF*" (z kodem XAML).

Budowa języka

- Komentarze:

  // komentarz jednoliniowy

  / komentarz wielowierszowy */*
- Znak ucieczki: **@"Jakiś tekst/"** (małpa jest tutaj „znakiem ucieczki", dzięki któremu ukośnik będzie interpretowany jako zwykły znak tekstowy).
- Łączenie stringów ze zmiennymi: **„Jakiś tekst" + x**.
- Zasięg zmiennych: od **{** do **}** i wchodzi do podrzędnych sekcji. Zmienne mogą być definiowane na końcu i obowiązują w całej przestrzeni nazw.
- Implikacja: **x?y** (jeśli wystąpi x to wykonaj y).
- Delegacja: wskazuje na jakąś funkcję, definiuje nowy typ. Czyli: jedna funkcja przekazuje (deleguje) parametry do drugiej funkcji. Zamiast pisać kilka funkcji do różnych rzeczy, można napisać jedną funkcję, która będzie się różnie zachowywać w zależności od przekazanych parametrów.
- Rekurencja: funkcja sama siebie wywołuje (ciąg Fibonacciego, fraktale, itp.).
- Macierz (Matrix): tablica tablic (jest to tablica 2D lub 3D).
- Metoda: rodzaj funckji znajdującej się wewnątrz klasy, np.

```
class nazwaKlasy
{
     static void Powitanie();
     {
          jakieśInstrukcje;
     }
}
```

- Metoda typu „**void**" (zwracać) – zwraca jakieś dane; bez tego słowa, musimy na końcu zamieścić „*return*".
- Wywołanie metody: **nazwaMetody()**.
- Argumenty metody: słowo **args** w metodzie głównej jest tablicą zawierającą nazwy argumentów wywoływanego programu, np.: *program.exe -jakiśArgument*.
- Konstruktor ma taką samą nazwę jak klasa nadrzędna, np.

```
class Osoba {
     public Osoba       // To jest właśnie konstruktor
}
```

Przykłady

Hello World w C#!

```
namespace Kurs
{
  class Program
  {
    static void Main(string[] args)
    {
         Console.WriteLine("Hello World!");
         Console.ReadKey(); // Odpowiednik "pause" w BATCH.
    }
  }
}
```

Hello World w C++

```
#include "stdafx.h"
#include <iostream>
using namespace std;

int main()
{
    cout << "Hello world!\n";
    return 0;
// W tym miejscu warto wstawić "BreakPoint", aby okno konsoli nie zamknęło się.
}
```

Pobieranie imienia

```
using System;
public class PrzykladowaKlasa
{
    public static void Main()
    {
        Console.WriteLine("Podaj swoje imię:");
        string imie = Console.ReadLine();
        Console.WriteLine("Twoje imię to: " + imie);
        Console.WriteLine("Wciśnij dowolny klawisz by zakończyć program.");
        Console.ReadKey();
    }
}
```

Pole powierzchni prostokąta

```
using System;
public class PrzykladowaKlasa
{
    public static void Main()
    {
        Console.WriteLine("Podaj swoje imię:");
```

```
        string imie = Console.ReadLine();
        Console.WriteLine("Twoje imię to: " + imie);
        Console.WriteLine("Wciśnij dowolny klawisz by zakończyć program.");
        Console.ReadKey();
    }
}
```

Pole powierzchni koła

```
using System;
public class Pole_kola
{
    public static void Main()
    {
        // Wzór do zrealizowania: Pi_R^2.
        double Pi = 3,14;
        Console.WriteLine("Podaj srednice kola: ");
     // Konwertujemy stringa, który podał użytkownik, na typ double:
        double srednica = double.Parse(Console.ReadLine());
        // Obliczamy pole powierzchni:
        double srednicaDoKwadratu = Math.Pow(srednica,2);
        double powierzchnia = Pi*srednicaDoKwadratu;
        Console.WriteLine("Pole powierzchni tego koła wynosi: " + powierzchnia);
        Console.WriteLine("Wciśnij dowolny klawisz by zakończyć program.");
        Console.ReadKey();
    }
}
```

Młody czy stary?

```
public class Mlody_Stary
{
    public static void Main()
    {
        Console.WriteLine("Podaj swój wiek:");
        int wiek = int.Parse(Console.ReadLine());

        if ((wiek > 30) && (wiek < 40))
        {
            Console.WriteLine("Jesteś w średnim wieku");
        }
        else if (wiek < 30)
        {
            Console.Write("Młodziak z Ciebie jeszcze");
        }
        else
        {
            Console.WriteLine("Jesteś już naprawdę stary!");
```

```
      }
         Console.ReadKey();
    }
}
```

Ostatnia aktualizacja: 28 listopada 2018.