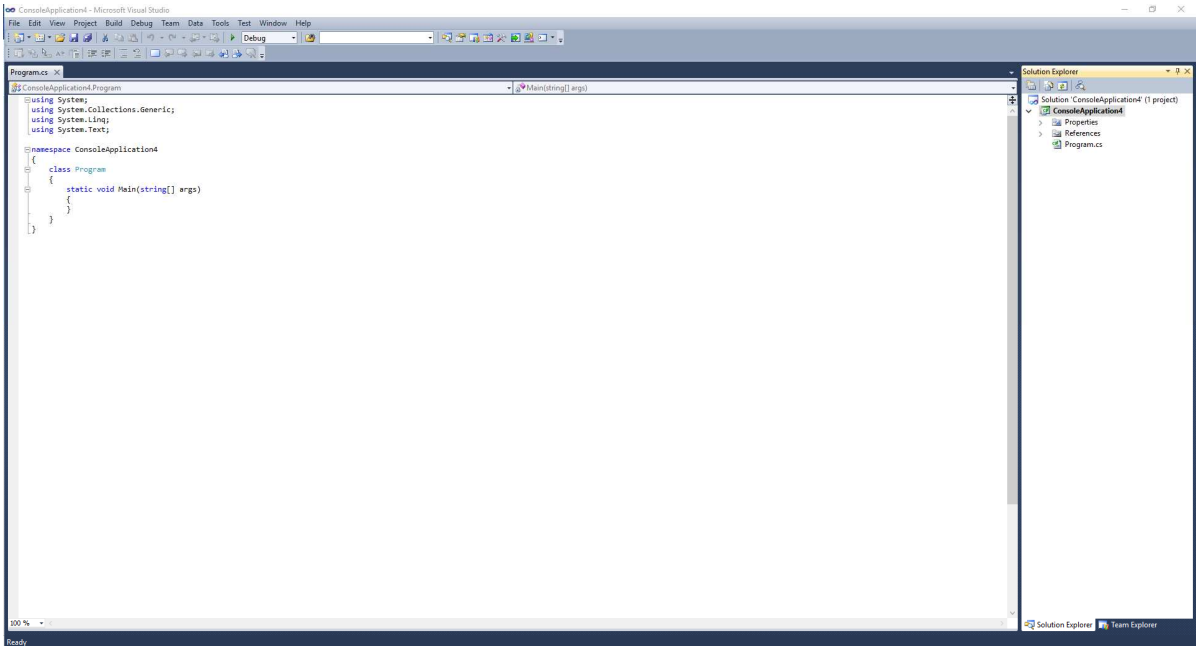
Podstawy programowania

Laboratorium

Ćwiczenie 1 – Wprowadzenie

Polecenie 1:

Uruchom środowisko programistyczne Microsoft Visual Studio 2010. Wybierz nowy projekt. Wybierz język programowania Visual C#. Wybierz rodzaj tworzonej aplikacji – Aplikacja konsolowa. Prawidłowe wykonanie czynności powinno skutkować pojawieniem się ekranu jak pokazano poniżej:



Polecenie 2:

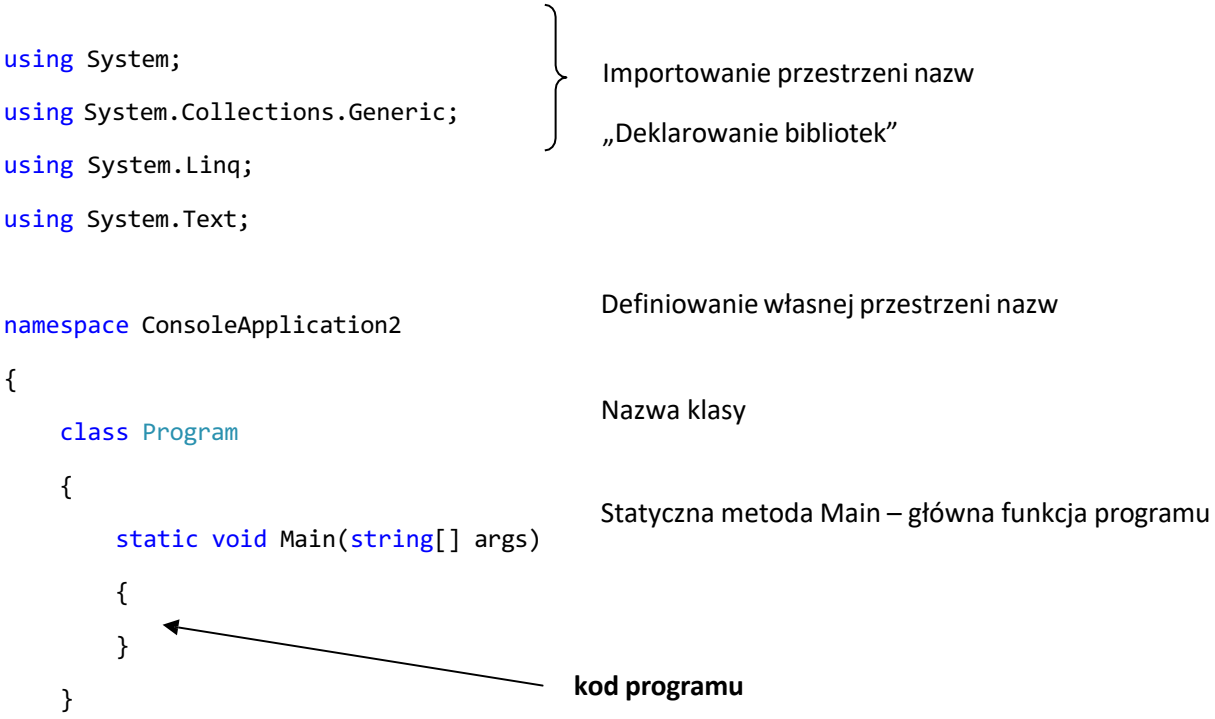
W menu **Debug** wybierz polecenie **Start Debugging**, co spowoduje kompilację oraz uruchomienie tzw. „pustego” programu. Jaki efekt na ekranie można zaobserwować i z czego on wynika?

Polecenie 3:

Zlokalizuj na dysku katalog, w którym znajdują się pliki powiązane z realizowaną aplikacją. Odnajdź plik wykonywalny *.exe realizowanej aplikacji i uruchom go.

Pamiętaj, że przy przenoszeniu na inny komputer plików projektu aplikacji należy kopiować całą zawartość katalogu aplikacji.

Struktura aplikacji konsolowej



Polecenie 4:

W menu **View** wybierz polecenie **Error List**, co spowoduje uruchomienie dodatkowego okna informującego o błędach.

W miejscu kodu programu umieść instrukcję

```
Console.ReadKey(true);
```

Skompiluj i uruchom program. Jaki jest efekt działania instrukcji i do czego ona służy? Czy w trakcie pisania kodu programu istotna jest wielkość liter? Co to jest konsola?

Polecenie 5:

Umieść kilka instrukcji `Console.ReadKey(true);` jedna pod drugą. Jaki jest efekt działania programu? Czy instrukcje można umieszczać jedna obok drugiej? Jaka jest funkcja średnika w zapisie kodu programu i czy zawsze jest on wymagany?

Wyświetlenie tekstu w konsoli

Do wyświetlenia tekstu w konsoli można użyć między innymi poleceń:

```
Console.WriteLine("Przykladowy tekst");
```

```
Console.Write("Przykladowy tekst");
```

Polecenie 6:

Napisz program wyświetlający pojedynczą linię przykładowego tekstu w konsoli z wykorzystaniem kolejno obu wymienionych powyżej instrukcji. Zatrzymaj działanie programu w taki sposób, aby można było odczytać (zauważyć) wyświetlony w konsoli tekst. Czy wypisywany tekst może zawierać polskie znaki diakrytyczne?

Polecenie 7:

Napisz program wyświetlający kilka przykładowych linii tekstu. Użyj różnych kombinacji poleceń WriteLine i Write. Jaka jest różnica w działaniu tych instrukcji?

Problematyka zmiennych liczbowych i tekstowych

Wybrane rodzaje zmiennych

Nazwa typu	Zakres wartości	Format
`Boolean`	1, 0 true, false	*wartości logiczne*
`Int16`	-32768...32767	*liczba całkowita*
`Int32`	-2,147,483,648 … 2,147,483,647	*liczba całkowita*
`Double`	$\pm 5.0 \times 10^{-324}$ do $\pm 1.7 \times 10^{308}$	*liczba rzeczywista*
`String`	do 2 miliardów znaków	*tekst*

Polecenie 8:

Napisz program o postaci jak podano poniżej i sprawdź działanie tego programu.

```
Int32 a, b;
a = 130;
b = 15;
Console.WriteLine("Wartość a i b wynosi odpowiednio: " + a + " oraz " + b + " i są to
liczby całkowite.");
Console.ReadKey(true);
```

Jakie są nazwy zmiennych wykorzystanych w programie i jakiego one są typu?
W jaki sposób można wyświetlić jednocześnie tekst oraz wartość zmiennej w ramach pojedynczej instrukcji Write (WriteLine)?
Zmodyfikuj zapis programu w taki sposób, aby przypisanie wartości zmiennej odbywało się już na etapie deklaracji zmiennej.

Uwaga! Istnieją alternatywne sposoby zapisu instrukcji do wyświetlania danych na ekranie, jak np.

```
Console.WriteLine("Wartość a i b wynosi odpowiednio: {0} oraz {1} i są to liczby
całkowite.", a, b);
```

Polecenie 9:

Zmodyfikuj zapis programu w taki sposób, aby wartości zmiennych **a** i **b** były liczbami rzeczywistymi o wartościach początkowych a= 12,34 oraz b=56,78. Uwaga! Konieczna jest zmiana rodzaju zmiennej. Przetestuj działanie programu.

Przypisanie łańcucha znaków z konsoli i konwersja na liczbę

W celu przypisania łańcucha znaków podanego przez użytkownika w konsoli oraz konwersji tego łańcucha na liczbę można posłużyć się kombinacją dwóch instrukcji:

```
1. x = Console.ReadLine();
2. y = Int32.Parse(x);
```

gdzie: x – jest zmienną tekstową typu String, Console.ReadLine() – reprezentuje ciąg znaków wpisany z klawiatury, y – zmienną liczbową typu Int32, Int32.Parse(x) – instrukcją powodującą przekonwertowanie ciągu tekstowego x do zmiennej y.

Można również stosować uproszczoną wersję zapisu instrukcji:

```
y = Int32.Parse(Console.ReadLine());
```

Należy zauważyć, że w przypadku drugiego wariantu zapisu, nie ma konieczności deklarowania i stosowania dodatkowej zmiennej tekstowej x.

Polecenie 10:

Napisz program do wyznaczania wartości pola **s** prostokąta o bokach **a** i **b.** Program powinien wyświetlać komunikaty zachęcające użytkownika do podania wartości długości poszczególnych boków prostokąta, a także komunikat wyświetlający wynik obliczeń. Zmienne **s**, **a** i **b** powinny być typu rzeczywistego. Przetestuj działanie programu. Jaki będzie efekt działa programu, jeżeli użytkownik zamiast wartości liczbowej poda przez pomyłkę znaki tekstowe?

Funkcje matematyczne

Zbiorem metod zawierającym podstawowe funkcje matematyczne jest standardowa klasa **Math.** Sprawdź np. z poziomu edytora aplikacji jakie funkcje matematyczne oferuje klasa Math.

Polecenie 11:

Napisz program, który wyświetla w konsoli wartość następującego wyrażenia z dokładnością do czterech miejsc po przecinku (prawidłowa odpowiedź 2,9995):

$$\frac{\sqrt{\pi}+e}{\log_{10}(\pi)+\ln(e)}$$

Uwaga! Ustalenie dokładności wyświetlania zmiennej liczbowej **x** można zrealizować stosując zapis,

```
Console.WriteLine("{0:#.######}",x);
```

co w podanym przykładzie oznacza, że na wyświetlenie wartości zmiennej x przeznaczono 7 pól, w tym 6 pól na część ułamkową.