PWr Spotkanie 4
Aplikacje webowe na platformie .NET
Laboratorium – **Lista 4**

Wstęp.

W języku C# Istnieje wiele instrukcji obecnych w innych językach o podobnej składni. Istnieją jednak również różnice, tak jak specjalna postać instrukcji **`switch`**/**`case`** tworząca dopasowanie do przypadku **`case`** zależne od typu argumentu.

Inną ciekawą różnicą jest przekazywanie parametrów do metod. O ile w wielu językach jedyną możliwością jest przekazywanie kopii wartości/referencji, o tyle w C# można przekazać argument przez referencję (za pomocą słówka kluczowego **`ref`** lub **`out`**). Parametry mogą mieć swojej wartości domyślne, przez co stają się wartościami opcjonalnymi. Nie trzeba też pamiętać kolejności parametrów, pod warunkiem, że pamiętamy ich nazwy (z nagłówka metody). Łącząc te dwa ostatnie podejścia można tworzyć wieloparametrowe metody z parametrami domyślnymi, w których podczas wywołania ustawienie tylko kilku wybranych parametrów jest bardzo wygodne. Oznaczając ostatni parametr, który musi być tablicą, słowem kluczowym **`params`**, umożliwiamy podczas użycia stosować notację, gdzie elementy tabeli zapisujemy po prostu jak kolejne parametry oddzielone przecinkiem.

Język C# pozwala na sterowanie procesem kompilacji poprzez wstawienie dyrektyw kompilatora. Pozwalają one na warunkową kompilację, deklaracje identyfikatorów używanych np. właśnie do warunkowej kompilacji. Część z tych identyfikatorów kompilator pobiera ze środowiska developerskiego (np. wersja języka, wersja platformy .NET itd.). Pozwala generować własne błędy kompilacji z komentarzem nt. logiki aplikacji zamiast błędów ogólnych.

List zadań

Założenie: użytkownik podaje jako dane ciągi znaki, które są poprawnymi wartościami dla konkretnego zadania. Do każdej metody program demonstrujący jej użycie.

1. Napisać metodę `GetFromConsoleXY`, która otrzyma w parametrach dwie linie komentarza (**`string`**). Metoda wypisze pierwszą linię, wczyta liczbę całkowitą. Wypisze drugą linię komentarz i wczyta drugą liczbę całkowitą.
    a. Pierwsza realizacja – te dwie liczby zwracane jako para (krotka).
    b. Druga realizacja – metoda typu **`void`** zwracająca dwie liczby w parametrach.
2. Napisać metodę `DrawCard` „rysującą” wizytówkę. Metoda otrzymuje w parametrach pierwszą linię wizytówki, drugą linię, znak obramowania, szerokość obramowania, minimalną szerokość całej wizytówki. Oprócz pierwszego parametru pozostałe są opcjonalne (wybrać własne domyślne wartości). Zademonstrować różne wywołania metody w tym z użyciem parametrów nazwanych. Wizytówka to prostokątna ramka z wycentrowanymi napisami np. dla zestawu danych
   `DrawCard („Ryszard”,”Rys”,”X”,2,20)` „narysuje”:
   ```
   XXXXXXXXXXXXXXXXXXXX
   XXXXXXXXXXXXXXXXXXXX
   XX      Ryszard     XX
   XX        Rys       XX
   XXXXXXXXXXXXXXXXXXXX
   XXXXXXXXXXXXXXXXXXXX
   ```
3. Stworzyć metodę `CountMyTypes`, w której parametrach można podać dowolny ciąg elementów rozdzielonych przecinkami, a metoda ta zliczy ile wśród parametrów było parzystych liczb całkowitych, liczb rzeczywistych dodatnich, napisów co najmniej 5-znakowych i elementów innych typów niż w/w, zwracając te cztery wartości w wybrany sposób. Użyć instrukcji `switch/case`.
4. Powyższe zadania uzupełnić dyrektywami kompilatora pozwalającymi w łatwy sposób tak uruchomić aplikację, by zobaczyć na konsoli więcej informacji o przebiegu jej działania (np. w zadaniu 3 informować, który typ został właśnie wykryty).

Data I: Spotkanie 5 (max 100 punktów)
Data II: Spotkanie 6 (max 80 punktów)
Data III: Spotkanie 7 (max 50 punktów)