Programowanie pod Windows
Zestaw 4

Język C# 3.0

2023-03-21

Liczba punktów do zdobycia: **10/36**
Zestaw ważny do: 2023-04-04

Uwaga! W zadaniach w których polecenie brzmi "dany jest plik tekstowy ..." należy sobie we własnym zakresie przygotować taki przykładowy plik tekstowy.

1. (**1p**) Zaimplementować metodę `bool IsPalindrome()` rozszerzającą klasę `string`. Implementacja powinna być niewrażliwa na białe znaki i znaki przestankowe występujące wewnątrz napisu ani na wielkość liter. Klient tej metody powinien wywołać ją tak:

   ```
   string s = "Kobyła ma mały bok.";
   bool ispalindrome = s.IsPalindrome();
   ```

2. (**1p**) Dany jest plik tekstowy zawierający zbiór liczb naturalnych w kolejnych liniach. Napisać wyrażenie LINQ, które odczyta kolejne liczby z pliku i wypisze tylko liczby większe niż 100, posortowane malejąco.

   ```
   from liczba in [liczby]
     where ...
       orderby ...
         select ...
   ```

   Przeformułować wyrażenie LINQ na ciąg wywołań metod LINQ to Objects:

   ```
   [liczby].Where( ... ).OrderBy( ... )
   ```

   Czym różnią się parametry operatorów **where/orderby** od parametrów funkcji **Where, OrderBy**?

3. (**1p**) Dany jest plik tekstowy zawierający zbiór nazwisk w kolejnych liniach.

   Napisać wyrażenie LINQ, które zwróci zbiór **pierwszych** liter nazwisk uporządkowanych w kolejności alfabetycznej. Na przykład dla zbioru (Kowalski, Malinowski, Krasicki, Abacki) wynikiem powinien być zbiór (A, K, M).

   Wskazówka: zgodnie z tytułem zadania użyć operatora GroupBy

4. (**1p**) Napisać wyrażenie LINQ, które dla zadanego foldera wyznaczy sumę długości plików znajdujących się w tym folderze.

   Do zbudowania sumy długości plików użyć funkcji `Aggregate`. Listę plików w zadanym folderze wydobyć za pomocą odpowiednich metod z przestrzeni nazw `System.IO`.

5. (**1p**) Dane są dwa pliki tekstowe, pierwszy zawierający zbiór danych osobowych postaci (Imię, Nazwisko, PESEL), drugi postaci (PESEL, NumerKonta). Kolejność danych w zbiorach jest przypadkowa.

   Napisać wyrażenie LINQ, które połączy oba zbiory danych i zbuduje zbiór danych zawierający rekordy postaci (Imię, Nazwisko, PESEL, NumerKonta). Do połączenia danych należy użyć operatora `join`.

6. (**2p**) Rejestr zdarzeń serwera IIS ma postać pliku tekstowego, w którym każda linia ma postać:

   ```
   08:55:36 192.168.0.1 GET /TheApplication/WebResource.axd 200
   ```

   gdzie poszczególne wartości oznaczają czas, adres klienta, rodzaj żądania HTTP, nazwę zasobu oraz status odpowiedzi.

   Napisać aplikację która za pomocą jednego (lub wielu) wyrażeń LINQ wydobędzie z przykładowego rejestru zdarzeń IIS listę adresów IP trzech klientów, którzy skierowali do serwera aplikacji największą liczbę żądań.

   Wynikiem działania programu powinien być przykładowy raport postaci:

   ```
   12.34.56.78 143
   23.45.67.89 113
   123.245.167.289 89
   ```

   gdzie pierwsza kolumna oznacza adres klienta, a druga liczbę zarejestrowanych żądań.

7. (**1p**) Listy generyczne ukonkretnieniamy typem elementów:

   ```
   List<int>    listInt;
   List<string> listString;...
   ```

   Z drugiej strony, w C# 3.0 mamy typy anonimowe, które nie są nigdy jawnie nazwane:

   ```
   var item = new { Field1 = "The value", Field2 = 5 };
   Console.WriteLine( item.Field1 );
   ```

   Czy możliwe jest zadeklarowanie i korzystanie z listy generycznej elementów typu anonimowego?

   ```
   var item = new { Field1 = "The value", Field2 = 5; };
   List<?> theList = ?
   ```

W powyższym przykładzie, jak utworzyć listę generyczną, na której znalazłby się element **item** w taki sposób, by móc następnie do niej dodawać nowe obiekty takiego samego typu?

Obiekty typu anonimowego mają ten sam typ, jesli mają tę samą liczbę składowych tego samego typu w tej samej kolejnosci.

8. (**2p**) Cechą charakterystyczną anonimowych delegacji, bez względu na to czy zdefiniowano je przy użyciu słowa kluczowego **delegate**, czy też raczej jako lambda wyrażenia, jest brak "nazwy", do której można odwołać się w innym miejscu kodu.

   Zadanie polega na zaproponowaniu takiego tworzenia anonimowych delegacji, żeby w jednym wyrażeniu możliwa była rekursja. W szczególności, poniższy fragment kodu powinien się kompilować i zwracać wynik zgodny ze specyfikacją.

```
List<int> list = new List<int>() { 1,2,3,4,5 };

foreach ( var item in
  list.Select( i => [....] ) )

  Console.WriteLine( item );
}
```

   W powyższym fragmencie kodu, puste miejsce ([....]) należy zastąpić definicją ciała anonimowej delegacji określonej rekursywnie:

$$f(i) = \begin{cases} 1 & i \leqslant 2 \\ f(i-1) + f(i-2) & i > 2 \end{cases}$$

   Wskazówka W języku C# można z powodzeniem zaimplementować operator punktu stałego **Y**, wykorzystywany do definicji funkcji rekurencyjnych. Zadanie to można rozwiązać więc definiując taki operator i za jego pomocą implementując funkcję rekurencyjną. Istnieje jednak zaskakujący i o wiele prostszy sposób rozwiązania wymagający jednak trochę nagięcia specyfikacji. Oba rozwiązania będą przyjmowane.

Wiktor Zychla