Programowanie pod Windows
Zestaw 5

.NET Base Class Library

ogłoszenie listy: 05-04-2005
ważność listy: 19-04-2005

Streszczenie

Biblioteka standardowa platformy .NET zawiera komplet funkcji do komunikacji z usługami systemu operacyjnego Windows. Ponieważ w przyszłych wersjach systemu operacyjnego Windows interfejs BCL ma szansę stać się natywnym interfejsem programowania Windows, warto szczegółowo zapoznać się z jego możliwościami.

1. **Wielojęzykowość .NET**
   Napisać program złożony z co najmniej 3 modułów, z których **co najmniej jeden** będzie napisany w innym języku niż C#.
   [**1p**]
2. Napisać klasę do obsługi liczb zespolonych. Dodać odpowiednie konstruktory, przeciążyć odpowiednie operatory. Porównać wydajność obliczeń z użyciem zaprojektowanej klasy z obliczeniami przy użyciu szablonu `complex` z C++ (napisać podobny kawałek kodu z przykładowymi obliczeniami i porównać czas wykonania).
   Rozszerzyć tę klasę o własne formatowane. Ściślej, zaimplementować interfejs `IFormattable` i obsługiwać dwa rodzaje formatowania:
   - domyślne (brak formatowania lub `d`) powinno dawać wynik $a + bi$
   - wektorowe (format `w`) powinno dawać wynik $[a, b]$.

   Przykładowy kawałek kodu:

   ```
   Complex z = new Complex( 4, 3 );
   Console.WriteLine( String.Format( "{0}", z ) );
   Console.WriteLine( String.Format( "{0:d}", z ) );
   Console.WriteLine( String.Format( "{0:w}", z ) );
   ```

   powinien dać wynik

   ```
   4+3i
   4+3i
   [4,3]
   ```

   [**1p**]
3. Zmierzyć czas działania kodu:

   ```
   int    k = 1000;
   string s = string.Empty;

   for ( int i=0; i<k; i++ )
     s += i.ToString();
   ```

dla rosnących `k` (powiedzmy co 1000 do 50000).

Jakie zjawisko możemy zaobserwować? Jak mu zaradzić?

[**1p**]

4. **Własne kolekcje**

   Zaimplementować kolekcję `Set` działającą jak zbiór, odrzucający duplikaty elementów. Które kolekcje wbudowane mogą być użyte jako bazowe dla `Set`?

   [**1p**]

5. Przetestować w praktyce składanie enumeratorów opakowujących. Ściślej, uzupełnić szkic kodu ze str. 211 z podręcznika, tak aby kod zadziałał zgodnie z sugestią.

   [**2p**]

6. **Zaawansowane struktury danych**

   Napisać klasę `TArray`, która będzie pewną specjalną implementacją tablicy elementów.

   Wewnątrz obiektu klasy dane powinny być przechowywane na drzewie, w którym każdy węzeł ma 10 synów, oznaczonych indeksami od 0 do 9. Aby dostać się do elementu o indeksie $i = \sum_{j=0}^{k} 10^j * i_j$ przechodzimy drzewo, na poziomie j przechodząc do syna i_j.

   Na przykład chcąc uzyskać dostęp do elementu o indeksie 7 wybieramy 7 syna korzenia drzewa i odczytujemy zapamiętany w nim element. Aby uzyskać dostęp do elementu o indeksie 145 przechodzimy kolejno przez 5-ego, 4-ego i 1-ego syna kolejnych węzłów począwszy od korzenia.

   Odpowiednie gałęzie drzewa powinny być budowane tylko wtedy, kiedy do tablicy dodawny jest element o odpowiednim indeksie. Na przykład dodanie do tablicy elementu o indeksie 10000000000 powinno spowodować powstanie tylko jednej długiej gałęzi od 0-ego syna korzenia, przez dziesięciu kolejnych synów kolejnych węzłów. Bezpośrednio po zainicjowaniu korzeń drzewa powinien mieć tylko 10 pustych referencji na kolejnych synów.

   Dzięki takiej konstrukcji użytkownik będzie mógł dodać na przykład element o indeksie 1 i element o indeksie 10000, a w drzewie będą przechowane tylko te 2 elementy (plus oczywiście puste referencje na pozostałe elementy w kolejnych węzłach). Tablica nie będzie więc (jak zwykła tablica liniowa) zużywać miejsca na wszystkie brakujące elementy między 1 a 10000.

   Zdefiniować odpowiedni indekser, tak aby do elementów tablicy można było odwoływać się w "zwykły" sposób, na przykład:

```
TArray a   = new TArray();
a[17]      = 5;
a[1000000] = 176;
```

   Zdefiniować odpowiedni enumerator, tak aby elementy tablicy można było przeglądać w "zwykły" sposób, na przykład:

```
TArray a   = new TArray();
a[17]      = 5;
a[1000000] = 176;
foreach ( int i in a )
...
```

   Porównać wydajność (tworzenie, przeglądanie):

   - TArray
   - ArrayList
   - zwykłych tablic

   [**4p**]