Fisiere Text

În C# fişierele text sunt gestionate, ca şi în C++, prin stream-uri (fluxuri de date structurate pe octeţi). Pentru lucrul cu fişiere text trebuie să utilizaţi spaţiul de nume System.IO, deci veţi include în antetul programului linia: `using System.IO;`

Cele două clase pe care le vom utiliza sunt StreamReader (pentru citirea datelor din fişier) şi StreamWriter (pentru scrierea datelor în fişier).

P1. În fişierul c:\test\date.txt pe prima linie se găsesc mai multe cuvinte formate din litere. Aceste cuvinte sunt separate prin spaţii. Să se afişeze cuvintele în fişierul date.out, fiecare cuvânt pe câte o linie.
Dacă nu este specificată o cale completă către fişier ci doar numele fişierului, atunci date.txt trebuie să se afle în acelaşi folder cu executabilul. Acest folder este de regulă **bin\release**. Mai jos este programul complet.

```
using System;
using System.IO;
namespace Fisiere
{
  class Program
  {
    static void Main(string[] args)
    {
      // citire din fisier
      StreamReader fin = new StreamReader("c:\\test\\date.txt");
      string linie = fin.ReadLine();
      fin.Close();

      // separare in cuvinte
      char[] separator = { ' ' };
      string[] cuvinte = linie.Split(separator,
StringSplitOptions.RemoveEmptyEntries);

      // tiparirea cuvintelor
      StreamWriter fout = new StreamWriter("c:\\test\\date.out");
      foreach (string cuv in cuvinte)
          fout.WriteLine(cuv);
      fout.Close();
    }
  }
}
```

De menţionat că fişierul de intrare trebuie să existe şi să nu fie vid! Fişierul de ieşire este creat automat, iar dacă există este suprascris. Dacă fişierul nu există atunci programul furnizează una din aşa-numitele **excepţii**. Mai multe despre excepţii vom discuta într-o lecţie ulterioară. Există însă o modalitate simplă pentru a verifica dacă un fişier există la o anumită locaţie. Acest lucru se face cu funcţia statică **Exists** din clasa **File**. Este returnează valoarea **true** dacă fişierul există sau valoarea **false** în caz contrar. Iată codul mai jos.

```
if (File.Exists("c:\\test\\date.txt"))
{
   // citire din fisierul date.txt,
   // scriere in fisierul de iesire
}
else
   Console.WriteLine("Fisier de intrare inexistent!");
```

P2. În folderul c:\test se află mai multe fişiere. Se cere să se scrie în fisierul c:\test\lista.txt numele fişierelor din acest folder care au extensia .in.

```
StreamWriter f = new StreamWriter("c:\\test\\lista.txt");
string[] numeFisiere = Directory.GetFiles("c:\\test", "*.in");
foreach (string s in numeFisiere)
   f.WriteLine(s);
f.Close();
```

Folosind funcţia **GetFiles** din clasa **Directory** se preiau automat fişierele din folderul c:\test care au extensia ".in". Pattern-ul de căutare, "*.in" este binecunoscut.

P3. În fişierul numere.txt se află mai multe numere naturale, dispuse pe mai multe linii. Să se afişeze în fişierul rez.txt suma lor.

```
int suma = 0;
string linie;
char[] sep = {' '};
StreamReader f = new StreamReader("c:\\test\\numere.txt");
while ((linie = f.ReadLine()) != null)
{
  string[] numere = linie.Split(sep, StringSplitOptions.RemoveEmptyEntries);
  foreach (string x in numere)
      suma += int.Parse(x);
}
Console.WriteLine(suma);
f.Close();
```

Observaţi modul în care se citesc mai multe linii dintr-un fişier dacă nu se cunoaşte dinainte numărul acestora. Este cumva asemănător cu modul de citire cu stream-uri din C++: `while (f >> linie) ...`

Probleme propuse

P4. Se citeşte un număr natural n. Să se scrie în fişierul **patrate.in** toate pătratele numerelor naturale cuprinse între 1 şi n.
P5. Fişierele **unu.txt** şi **doi.txt** conţin pe mai multe linii câte un cuvânt. Să se concateneze conţinutul celor două fişiere în fişierul **trei.txt**.
P6. Fişierul **date.in** conţine mai multe numere pe mai multe linii. Să se ordoneze crescător şirul de numere şi să se tipărească în fişierul **date.out**.
P7. Fişierul **date.in** conţine pe prima linie n, iar pe linia a doua n numetre naturale separate prin cate un spatiu. Scrieti in fisierul **date.out** toate numerele prime.

Agenda

Descriere

Aplicaţia va permite gestionarea unei agende personale. Agenda este compusă din activităţi organizate pe domenii. Activităţile gestionate de către aplicaţie pot fi simple (care se execută o singură dată) sau recurente (care se execută periodic). Pentru toate activităţile se vor reţine codul, titlul, categoria asociată, durata şi data de început. Pentru activităţile recurente se vor specifica în plus numărul de programări şi intervalul dintre două programări succesive. Nu se pot programa mai multe activităţi simultan.

Aplicaţia va permite:

- gestionarea categoriilor (adăugare / modificare / ştergere); - introducerea, modificarea şi ştergerea de activităţi; - vizualizarea grafică a datelor pentru o săptămână; - obţinerea de situaţii referitoare la activităţile programate pentru o perioadă sau pentru o categorie de activităţi.

Implementare

Pentru construirea modelului vom avea nevoie de următoarele clase: Categorie: va conţine datele referitoare la o categorie: Cod, Denumire şi Culoare; ActivitateSimpla: va conţine datele despre o activitate simplă (Cod, Titlu, CodCategorie, Data, Durata); ActivitateRecurenta: va conţine datele despre o activitate recurentă (Cod, Titlu, Descriere, CodCategorie, Data, Durata, NumarProgramari, Interval); Model: va conţine colecţiile de categorii şi activităţi, precum şi operaţiile de modificare pentru acestea.

Specificaţii

Pentru a se uşura lucrul cu activităţi cele două clase ActivitateSimpla şi ActivitateRecurenta pot fi derivate dintr-o clasă de bază comună sau pot fi grupate într-o singură clasă. Operaţiile asupra datelor trebuie să verifice corectitudinea operaţiilor (exemplu: nu se pot programa două activităţi în acelaşi timp). Salvarea datelor din model în fişiere text se poate face într-un format de tipul: numar_categorii,numar_activitati cod_1,denumire_1,culoare_1 .... cod_n,denumire_n,culoare_n tip_act_1, titlu_1, codcat_1, data_1, durata_1, nrprog_1,interval_1 .... tip_act_k,titlu_k,codcat_k,data_k,durata_k,nrprog_k,interval_k Numărul de programări şi intervalul dintre acestea vor fi specificate doar în cazul în care activitatea este de tip recurent.