Laboratorul 10

1. Exemplu de programare.

Problema

Pentru o statistică, un lanţ de magazine de mobilă doreşte să facă o comparaţie a vânzărilor din două magazine aflate în acelaşi oraş. Comparaţia se va face sub forma unui grafic ce va afişa vânzările din raioanele similare ale celor două magazine. Pentru fiecare magazin este disponibil câte un fişier de date cu următoarea structură:

```
ID-ul raionului
Nr. de zile lucratoare pentru care s-au inregistrat vanzarile
Vanzarile pt. ziua 1
Vanzarile pt. ziua 2
...
Vanzarile pt. ultima zi
ID-ul raionului
Nr. de zile lucratoare pentru care s-au inregistrat vanzarile
Vanzarile pt. ziua 1
...
```

Conţinului unui fişier de date pentru unul dintre magazine ar putea fi:

```
101
3
5388.99
2647.69
6108.05
102
3
2692.17
10922.69
7933.78
```

Programul afişează rezultatele în forma următoare:

```
Grafic comparativ pentru raioanele din magazinele 1 si 2

Magazin  Vanzari pe raion in mii de lei
         0         5        10        15        20        25
         |.........|.........|.........|.........|.........|

         Raionul 101
   1     ****************************
         Raionul 101
   2     *************************
         Raionul 102
   1     ******************************************
         Raionul 102
   2     ******************************************
```

Intrări

Cele două fişiere de date (`magazin1` şi `magazin2`).

Ieşire

Graficul prezentat mai sus.

Discuţie

Citirea datelor din fişiere se face după metoda cunoscută deja. Se deschid fişierele, se citesc ID-urile raionului, numărul de zile pentru care s-au facut înregistrări şi vânzările zilnice. După procesarea datelor pentru raion, în ambele magazine, se poate trece la raionul următor până la epuizarea datelor de intrare. Pentru ca citirea celor două fişiere se face similar, putem folosi o funcţie comună pentru citirea lor. Tot ce trebuie să facem este să transmitem numele fişierului ca parametru. Funcţia ne va furniza vânzările totale pentru fiecare raion.

O altă funcţie va realiza afişarea rezultatelor în forma dorită. În acest program sunt trei bucle: una în funcţia `main` (pentru citirea şi procesarea datelor), a doua în funcţia de citire a datelor pentru un raion şi una în funcţia de afişare. Bucla din funcţia `main` testează `EOF` pentru ambele fişiere de date. Bucla din funcţia `TiparesteDate` este puţin mai specială. Variabila care controlează bucla este decrementată cu 500 la fiecare iteraţie.

Ipoteză

Fiecare fişier de date este ordonat după ID-ul raionului.

Programul

```
//*********************************************
//Grafic.cpp
//Programul genereaza graficul vanzarilor lunare
//pe raioane pentru doua magazine ale unei retele
//de magazine de mobila, permitand comparatia
//pe raioane similare din cele doua magazine
//*********************************************
#include <iostream>
#include <iomanip>  //Pentru setw()
#include <fstream>  //Pentru operatii I/O cu fisiere
using namespace std;

void CitesteDate(ifstream&, int&, double&);
void TiparesteDate(int, int, double);
void TiparesteHeader();

int main()
{
     int raionID1;      //ID-ul raionului pt. magazinul 1
     int raionID2;      //ID-ul raionului pt. magazinul 2
     double vanzari1;   //Vanzarile din magazinul 1
     double vanzari2;   //Vanzarile din magazinul 2
     ifstream magazin1; //Fisierul de date pt. magazinul 1
     ifstream magazin2; //Fisierul de date pt. magazinul 2
```

```
     magazin1.open("magazin1.dat");
     magazin2.open("magazin2.dat");
     if(!magazin1 || !magazin2)
                    //Verifica deschiderea fisierelor de date
     {
          cout << "Eroare: Nu se pot deschide "
               << "fisierele de intrare" << endl;
          return 1;
     }

     TiparesteHeader();
     //Prima citire
     CitesteDate(magazin1, raionID1, vanzari1);
     CitesteDate(magazin2, raionID2, vanzari2);
     while(magazin1 && magazin2)
     {
          cout << endl;
          //Procesarea pt. mag.1
          TiparesteDate(raionID1, 1, vanzari1);
          //Procesarea mag.2
          TiparesteDate(raionID2, 2, vanzari2);
          CitesteDate(magazin1, raionID1, vanzari1);
          CitesteDate(magazin2, raionID2, vanzari2);
     }
     return 0;
}

void TiparesteHeader()
{
     cout << "Grafic comparativ pentru raioanele din "
          << "magazinele 1 si 2" << endl << endl
          << "Magazinul  Vanzari pe magazin in "
          << "sute de milioane de lei"
          << endl
     << " 0         5        10        15        20        25"
     << endl
     << " |.........|.........|.........|.........|.........|"
     << endl;
}

void CitesteDate(
        /*inout*/ ifstream& fisierDate, //Fisier de intrare
        /*out  */ int& raionID,    //Numarul raionului
        /*out  */ double& vanzariRaion) //Vanzarile lunare
{
     int numZile;    //Numarul de zile lucrate din luna
     int zi;         //Controloeaza bucla de citire
     double vanzare; //Vanzarile pe o zi

     fisierDate >> raionID;
```

```
     if(!fisierDate) //Verifica daca fisierul a fost vid (EOF)
          return;    //Daca da, iese din functie

     fisierDate >> numZile;
     vanzariRaion = 0.0;
     //Initializeaza controlul buclei de citire
     zi = 1;
     while(zi <= numZile)
     {
          fisierDate >> vanzare;
          vanzariRaion = vanzariRaion + vanzare;
          //Actualizeaza variabila de control al buclei
          zi++;
     }
}

void TiparesteDate(
     /*in*/ int raionID, //ID-ul raionului
     /*in*/ int numMagazin,   //Numarul magazinului
     /*in*/ double vanzariRaion)   //Vanzarile totale/raion
{
     cout << setw(12) << "Raion " << raionID << endl;
     cout << setw(3) << numMagazin << "  ";
     while(vanzariRaion > 500.0)
     {
          //Tipareste un '*' pt. fiecare 500 de lei
          cout << '*';
          //Actualizeaza variabila de control
          vanzariRaion = vanzariRaion - (float)500.0;
     }
     cout << endl;
}
```

Sugestii pentru test şi debugging

- La începutul programului scrieţi câte un prototip pentru fiecare funcţie folosită în program.
  Nu uitaţi să puneţi semnul ; după fiecare prototip. Nu puneţi, însă, acest semn după heading-ul funcţiei.
- Lista parametirlor formali trebuie să cuprindă tipul de dată pentru fiecare parametru.
- Folosiţi parametri valoare atunci când rezultatul nu trebuie returnat printr-un parametru.
- Asiguraţi-vă că în lista parametrilor formali tipurile de dată ale parametrilor referinţă au ataşat semnul &.
- Lista parametrilor formali trebuie să corespundă cu cea a parametrilor actuali în număr şi tipuri de date. Compilatorul semnalizează dacă numărul parametrilor nu corespunde, însă nu semnalizează nepotrivirile tipurilor de dată.
- Parametrii referinţă cer o variabilă ca parametru actual, în timp ce parametrilor valoare le pot corespunde şi expresii ale căror rezultate au ca tip de dată tipul parametrului valoare.

2. Date fiind declaraţiile

```
const int ANGLE = 90;
char letter;
int number;
```

arătaţi care dintre următorii parametri actuali pot fi transmişi prin valoare, prin referinţă sau amândouă:

a) `letter`
b) `ANGLE`
c) `number`
d) `number+3`
e) `23`
f) `ANGLE * number`
g) `abs(number)`

3. Având următoarele date de intrare:

```
3 2 4
```

arătaţi ce va tipări următorul program:

```
#include <iostream>
using namespace std;

void Test( int&, int&, int& );
int main()
{
     int var1;
     int var2;
     int var3;
     Test(var1, var2, var3);
     var2 = var2 + 10;
     cout << "Raspunsurile sunt " <<
var2 << ' ' << var3 << ' ' << var1;
     return 0;
}

void Test( int& var2, int& var3, int&
var1)
{
     cin >> var2 >> var3 >> var1;
     var1 = var2 * var3 + var1;
}
```

4. Numerotaţi instrucţiunile marcate şi arătaţi care este ordinea în care sunt executate:

```
#include <iostream>
using namespace std;
void Functie( int&, int& );
int main()
{
     int numar1;
     int numar2;
___  cout << "Exercitiul ";
```

```
___  Functie(numar1, numar2);
___  cout << numar1 << ' ' << numar2 << endl;
     return 0;
}

void Functie( int& valoare1, int& valoare2 )
{
     int valoare3;
___  cin >> valoare3 >> valoare1;
___  valoare2 = valoare1 + 10;
}
```

5. Ce afişează programul de mai jos? De ce?

```
#include <iostream>
using namespace std;

void Test(int&,int);
int main()
{
     int variabila1;
     variabila1 = 12;
     int variabila2;
     variabila2 = 14;
     Test(variabila1, variabila2);
     cout << "In functia main, dupa primuul apel
     variabilele "
          << "au urmatoarele valori: "
          << variabila1 << ' ' << variabila2 << endl;
     variabila1 = 15;
     variabila2 = 18;
     Test(variabila2, variabila1);
     cout << "In functia main, dupa al doilea apel "
          << "variabilele au urmatoarele valori: "
          << variabila1 << ' ' << variabila2 << endl;
     return 0;
}
void Test(int& s, int t)
{
     s = 3;
     s = s+2;
     t = 4*s;
     cout << "In functia Test, variabilele au "
          << "urmatoarele valori: "
          << s << ' ' << t << endl;
}
```

6. Scrieţi o funcţie intitulată `CountUpper` care să contorizeze majusculele dintr-o linie de date de intrare. Funcţia trebuie să returneze acest număr în variabila `upCount`.

7. Scrieţi o funcţie numită `GetNonBlank` care să folosească funcţia `cin.get` pentru citirea caracterelor din stream-ul standard de intrare şi să returneze primul caracter diferit de blank pe care îl întâlneşte. Această funcţie ar trebui să aibă acelaşi comportament ca o citire care foloseşte operatorul `>>`.

8. Folosind funcţia `cin.get` pentru citirea caracterelor de pe stream-ul standard de intrare, scrieţi funcţia `SkipToBlank` care să aibă acelaşi comportament cu funcţia `cin.ignore`.

9. Scrieţi un program C++ care citeşte caractere reprezentând numere în baza 2 dintr-un fişier de date pe care le converteşte în baza 10. Numerele zecimale vor fi afişate pe coloane, cu un cap de tabel explicativ. Fiecare număr binar se găseşte în fişier inversat, adică prima cifră intâlnită este, de fapt, cifra cea mai din dreapta a numărului. Programul citeşte câte o cifră. Pe măsură ce se citeşte câte o cifră, ea este transformată în corespondentul zecimal înmulţind-o cu 2 ridicat la putere (puterea se stabileşte în funcţie de poziţia cifrei binare în număr). În fişier se va găsi câte un singur număr pe o linie.

10. Scrieţi un program care să numere cuvintele dintr-un text, eliminând spaţiile suplimetare dintre cuvinte, în cazul în care apare mai mult de unul (funcţia `WhitespaceToBlank`). Textul este citit dintr-un fişier şi este afişat pe ecran.

11. Calculaţi calendarul în functie de ziua săptămânii în care este 1 ianuarie din anul respectiv. Consideraţi că anul este bisect dacă el este divizibil cu 4 (deşi determinarea unui an bisect este puţin mai complicată). Numărul de luni pentru care se afişează calendarul este citit de la utilizator. Verificaţi ca toate datele de intrare să se încadreze în intervalele corecte. Sugestie: Puteţi folosi o funcţie de afişare a calendarului pentru o lună, având ca parametri ziua din săptămână în care începe acea lună şi numărul zilelor din lună. Zilele săptămânii pot fi numerotate de la 1 la 7.
Programul trebuie să se comporte astfel (datele introduse de utilizator sunt prezentate cu litere ingrosate):

```
Care este anul pentru care doriti afisarea calendarului?
2007
In ce zi din saptamana este ziua de 1 ianuarie 2007? (1 pentru
luni, 2 pentru marti, etc.)
2
Pentru cate luni din anul 2007 doriti afisarea calendarului?
2

2007
Ianuarie
  L  M  M  J  V  S  D
  1  2  3  4  5  6  7
  8  9 10 11 12 13 14
 15 16 17 18 19 20 21
 22 23 24 25 26 27 28
 29 30 31
Februarie
  L  M  M  J  V  S  D
           1  2  3  4
```

```
 5  6  7  8  9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28
```

12. Scrieţi o funcţie recursivă `power(base, exponent)` care returnează valoarea $base^{exponent}$. De exemplu, `power(3, 4)` = 3*3*3*3. Presupuneţi că `exponent` este un întreg mai mare sau egal cu 1. Indicaţie: Pasul de recursie foloseşte relaţia $base^{exponent} = base \cdot base^{exponent-}$ şi condiţia de terminare apare când `exponent` este 1.

13. *Generarea numerelor aleatoare*. Scrierea unui program care implementează distribuiţia cărţilor în jocul *Solitaire* presupune folosirea unui mecanism prin care cărţile sunt alese la întâmplare. Funcţia `rand()` din biblioteca `cstdlib` generează un număr întreg cu valoarea cuprinsă între `0` şi `RAND_MAX`, o constantă simbolică definită, de asemenea, în biblioteca `cstdlib`. Programul următor generează un număr cuprins între 1 şi 52, pentru a alege la întâmplare una dintre cele 52 de cărţi dintr-un pachet:

```
#include <iostream>
#include <cstdlib>
using namespace std;

int main()
{
     srand(time(0));
     cout << 1 + rand() % 52 << endl;
     return 0;
}
```

Funcţia `rand()` generează o secvenţă *pseudoaleatoare*, adică secvenţa se repetă de fiecare dată în mod identic. Funcţia `srand()` care primeşte un parametru întreg iniţializează generatorul de numere aleatoare, fiecare valoare a parametrului generând o altă secvenţă de numere. Funcţia time(0) întoarce o valoare care reprezintă în secunde timpul calculatorului, astfel că prin apelul `srand(time(0))` ne asigurăm că generatorul de numere aleatoare va fi iniţializat de fiecare dată cu o valoare nouă.

Scrieţi un program care simulează aruncarea cu zarul şi calculaţi probabilitatea de apariţie a valorii 2 după 100 de aruncări.