Laboratorul 2

Code::Blocks este o platformă open-source care permite dezvoltarea aplicațiilor C++ și poate fi folosită pe Windows, Linux sau macOS. Poate fi descărcat de la următoarea adresă:

`http://sourceforge.net/projects/codeblocks/files/Binaries/16.01/Windows/codeblocks-16.01-setup.exe`

Dev-Cpp este un mediu integrat de dezvoltare a aplicaţiilor C++. Foloseşte varianta Mingw a compilatorului GCC (GNU Compiler Collection). Poate fi descărcat de la următoarea adresă:

`http://prdownloads.sourceforge.net/dev-cpp/devcpp-4.9.9.2_setup.exe`

O listă mai cuprinzătoare de medii de dezvoltare a aplicațiilor C++ este disponibilă aici:

`https://en.wikipedia.org/wiki/Comparison_of_integrated_development_environments#C.2FC.2B.2B`

1. **Folosirea mediului de programare.** Deschideţi unul dintre editoarele de aplicații C++ şi scrieţi următorul program:

```
//Acesta este un program C++
#include <iostream>
using namespace std;

int main ()
{
  cout << "Bun venit la laboratorul de PCLP2 !";
  return 0;
}
```

În urma rulării acestui program, pe ecran va trebui să vă apară textul

```
Bun venit la laboratorul de PCLP2 !
```

Acesta este unul dintre cele mai simple programe care se pot scrie in C++, dar care conţine câteva elemente specifice acestui limbaj. Iată care este semnificaţia fiecărei linii de cod:

`//`**`Acesta este un program C++`**

Instrucţiunea de mai sus conţine un comentariu. Toate instrucţiunile care încep cu semnul // sunt considerate comentarii şi nu au niciun efect asupra comportamentului programului. Programatorul poate să le folosească pentru a include explicaţii scurte sau observaţii asupra codului sursă. In cazul nostru, linia este o scurtă descriere a programului.

`#include <iostream>`

Liniile care încep cu semnul # sunt directive pentru preprocesor. Ele nu sunt linii obişnuite de cod deoarece conţin indicaţii pentru preprocesorul compilatorului. În cazul nostru, directiva `#include <iostream>` spune preprocesorului să includă fişierul standard `iostream`. Acest fişier include elemente din biblioteca standard C++ de intrare/ieşire. Componente din această bibliotecă sunt necesare mai departe în program.

`using namespace std;`

Toate elementele bibliotecii standard C++ sunt declarate într-un spaţiu de nume (namspece), şi anume in namespace-ul cu titlul `std`. Pentru a accesa funcţionalităţile acestei biblioteci, trebuie să utilizăm instrucţiunea de mai sus care este foarte frecventă în programele C++ care folosesc biblioteca standard.

`int main ()`

Această linie de program corespunde începutului definiţiei principalei funcţii care alcătuieşte programul nostru. Funcţia se numeşte `main` şi este punctul din care începe rularea programului, indiferent de locul în care se găseşte ea în codul sursă. Nu contează dacă în program există funcţii cu alte nume definite înaintea sa. Instrucţiunile conţinute în funcţia `main` sunt executate primele. De altfel, orice program C++ trebuie să conţină o funcţie care se numeşte `main`.

Cuvântul `main` este urmat de o pereche de paranteze rotunde `()` pentru că este o declaraţie de funcţie. În C++ declaraţiile de funcţii sunt formate din numele funcţiei urmat de paranteze rotunde. Opţional, intre aceste paranteze se poate gasi o lista de parametri.

După parantezele () se găseşte corpul funcţiei `main` cuprins între acolade `{}`. Instrucţiunile care se găsesc între acolade reprezintă comenzile care se rulează atunci când se execută funcţia.

`cout << " Bun venit la laboratorul de PCLP2 !";`

Aceasta este o instrucţiune C++ care produce afişarea pe ecran a textului `Bun venit la laboratorul de PCLP2 !`

`cout` reprezintă un obiect din fişierul `iostream` din biblioteca standard C++ şi care face parte din namespace-ul `std`. Iată de ce a fost nevoie de cele două instrucţiuni de la începulul programului.

Orice instrucţiune C++ se termină prin caracterul `;`.

`return 0;`

Această instrucţiune produce încheierea execuţiei programului. Instrucţiunea `return` se foloseşte pentru a genera un cod, iar codul 0 înseamnă că programul s-a încheiat fără erori. Este cel mai obişnuit mod de a încheia un program C++.

2. **Funcţii cu parametri valoare sau cu parametri referinţă**. Parametrii funcţiilor pot fi de tip valoare dau de tip referinţă. Un apel de funcţie într-un program înseamnă execuţia corpului funcţiei apelate. Funcţiile sunt subprograme care primesc date prin intermediul parametrilor valoare sau al celor de tip referinţă. Funcţiile generează date prin intermediul parametrilor de tip referinţă sau prin valorile returnate. Parametrii valoare ai unei funcţii primesc copii ale parametrilor furnizate în apelul funcţiei

(parametrii actuali). Parametrii referinţă primesc adresele de memorie ale parametrilor actuali.

Programul de mai jos calculează în două moduri pătratul unui număr întreg. Prima funcţie primeşte numărul prin intermediul unui parametru de tip valoare, iar a doua printr-un parametru de tip referinţă. În primul caz, rezultat se recuperează prin citirea valorii returnate, iar in al doilea caz prin citirea valorii din parametrul referinţă.

```
#include <iostream>
using namespace std;
int PatratPrinValoare(int);
void PatratPrinReferinta(int&);

int main()
{
 int x = 2;
 int z = 4;
 cout << "x= " << x << " inainte de PatratPrinValoare\n"
      << "Valoarea returnata de PatratPrinValoare: "
      << PatratPrinValoare(x) << endl
      << "x= " << x << " dupa PatratPrinValoare\n"
      << endl;

 cout << "z= " << z << " inainte de PatratPrinReferinta"
      << endl;
 PatratPrinReferinta(z);
 cout << "z= " << z << " dupa de PatratPrinReferinta"
      << endl;
 return 0;
}

int PatratPrinValoare(int a)
{
 a = a * a;
 return a;
}

void PatratPrinReferinta(int& b)
{
 b = b * b;
}
```

3. **Funcţii recursive**. Programul de mai jos calculează factorialul unui număr natural. Rescrieţi funcţia `Factorial` astfel încât să calculeze factorialul într-o manieră iterativă (nerecursivă).

```
 #include <iostream>
#include <iomanip>
using namespace std;
int Factorial(int);
int main()
{
```

```
     int n;
     cout << "Introduceti numarul pentru care se calculeaza
factorialul: ";
     cin >> n;
     cout << n << "! = " << Factorial(n) << endl;
     return 0;
    }

    int Factorial(int val)
    {
     if(val <= 1) //cazul de baza
       return 1;
     else
       return val * Factorial(val - 1);
 }
```