Laborator 10

Diferențe între C și C++. Biblioteci C++. STL in C++

În acest capitol sunt prezentate consideraţii teoretice și probleme rezolvate privind diferențele semnificative între limbajele C și C++, definirea și utilizarea noțiunilor specifice programării orientate pe obiecte. Este de asemenea ilustrat modul în care se definesc clasele și componentele lor, operatorul de rezoluție, constructorii și destructorii, funcțiile prietene, conceptul de moștenire, etc.

CONSIDERAŢII TEORETICE

I. Diferente C/C++

În general, nu există diferențe între fișierele sursă din C si C++. Ambele limbaje acceptă directivele de compilator (Ex. #include și #define). Fișierele program scrise în C++ vor fi salvate cu extensia .cpp pentru a le diferenția de cele scrise în limbajul C.

În cele ce urmează sunt prezentate foarte pe scurt câteva dintre diferențele semnificative dintre cele două limbaje **C** și respectiv **C++**.

1. În C++ sunt posibile **conversiile de tip** (type casting)

În C și C++ conversiile de tip se realizează diferit.

Ex.:

in C:
```
int n;
float x;
x = (float) n;
```

in C++:
```
int n;
float x;
x = float(n);
```

2. În C++ se utilizează **cin** >> și respectiv **cout** << în locul funcțiilor scanf() și printf() (detalii cap.10).

3. În C++ **declarațiile variabile** se pot face oriunde în program

În C, declarațiile de tip ale variabilelor se realizează de regulă imediat după acolada deschisă a unei funcții (main sau altă funcție utilizator).
În C++ declarațiile de tip se pot realiza oriunde în program.
Ex.: variabila i este declarată în corpul funcției for în C++.

în C
```
#include <stdio.h>
void main()
{int i;
for (i=0;i<10;i++)
printf("%d\n", i);}
```

în C++
```
#include <iostream.h>
void main()
{for (int i=0;i<10;i++)
cout << i << '\n';}
```

Ex: program în C++
```
int a = 10;
cout << a << endl; // afiseaza 10
```

```
{     double a, b = 2.0;
a = 3.14 * b;
cout << a << endl; // afiseaza 6.28
}
a *= 3;
cout << a << endl; // afiseaza 18.84
```

4. În C++ este posibil **apelul prin referința (&)**

Există două posibilități de transmitere a parametrilor actuali către o funcție în C++:

- prin valoare
- prin referință

Transferul prin referință este util și atunci când parametrul are dimensiune mare (struct, class) și crearea în stiva a unei copii a valorii lui reduce viteza de execuție .

Ex.: interschimbarea a valorilor a două variabile prin referință

```
void schimba(int &a, int &b)
{int  aux=a;
a=b; b=aux;}
```

5. În C++ se pot defini **funcții care returnează variabile**

Ex.:

```
float &var_max(float &x, float &y)
{if (x > y) return x;
else return y;}
```

6. În C++ se pot utiliza **funcțiile inline**

În C++ se utilizează funcții inline pentru a crea programe mai rapide .

Ex. :

```
inline double ipot(double a, double b)
{     return sqrt(a*a + b*b);};
```

7. În C++ se pot utiliza **parametri impliciți**

La apelarea unei funcții cu parametri impliciți se poate omite specificarea parametrilor efectivi pentru acei parametri formali care au declarate valori implicite și se transferă automat valorile respective. Se pot specifica mai multe argumente cu valori implicite pentru o funcție.

Ex.:

```
int fct(int x=7, int y=9)
{      return x+y; }
void main()
{cout << fct() << endl; // 16
cout << fct(2) << endl; // 11
cout << fct(2, 3) << endl; // 5}
```

8. În C++ este posibilă **supraîncărcarea funcțiilor**

Există posibilitatea de a atribui unui simbol mai multe semnificații, care pot fi deduse în funcție de context. În particular, majoritatea operatorilor C++ pot fi priviți ca nume de funcții și deci pot fi supraîncărcați.

Ex.:
```
void fct(int a)
{  cout << "functia 1" << a;}
void fct(char *a)
{    cout << "functia 2" << a;}
```

9. În C++ **alocarea dinamică a memoriei** se realizează utilizând operatorii new și delete
Operatorii **new** și **delete** sunt similari funcțiilor din grupul malloc() și free(), dar constituie o metoda nouă, superioară acestora și adaptată programării orientate pe obiecte. Operatorul delete este complementarul lui new și înlocuiește funcția free() de dezalocare a memoriei dinamice alocate.

Ex:
```
int * ip1, *ip2, *ip3;
ip1=new int;      // variabila întreaga neinitializata
ip2=new int(2);   // variabila întreaga initializata cu 2
ip3=new int[100]; // tablou de 100 de întregi
```

II. Biblioteci C++

Biblioteca Standard C++ (Standard C++ Library) cuprinde toate bibliotecile standard C precum si biblioteci dedicate C++.
Biblioteci functii I/O: <iostream>, <fstream>, etc: pentru procesarea fisierelor de intrare/iesire si operatii consola
Biblioteci functii siruri: <cstring> si <string> : functii procesare siruri de caractere/ text
Biblioteca functii matematice <cmath>: functii matematice
Biblioteca functii caractere <cctype>: pentru functii identificare/conversie caractere

III. Biblioteca STL (Standard Template Library)

Biblioteca STL: colecție de algoritmi, structuri de date și alte componente care pot fi utilizate pentru a simplifica si optimiza codul C++.
Componente de baza:
Algoritmi: pentru sortare, selectie și căutare binară a datelor stocate în containere.
Containere: vector, listă, map, set și stivă (stack), utile pentru a stoca și manipula date.
Iteratoare: obiecte care oferă o modalitate de a parcurge elementele unui container. Ex: forward_iterator, bidirectional_iterator și random_access_iterator.
Functors-Obiecte funcție: sunt obiecte care pot fi folosite ca argumente de funcție pentru algoritmi. Ele oferă o modalitate de a transmite o funcție unui algoritm
Adaptoare: sunt componente care modifică comportamentul altor componente din STL. Ex. adaptorul reverse_iterator poate fi folosit pentru a inversa ordinea elementelor dintr-un container.

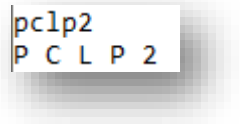
PROBLEME REZOLVATE

Ex.1: Programul în C++ citeste un sir de caractere (string) si-l afiseaza cu majuscule.

Varianta in C++

```
#include <iostream>
#include <string>
#include <cctype>
using namespace std;
int main() {
   string s1;char c;
   cin>>s1; //daca introduc de la tastatura “PCLP”
   for (int i=0;i<s1.length();i++)
   {c= toupper(s1[i]);cout<<c<<" ";} //afiseaza sirul “PCLP”
   return 0;}
```

Rezultate:



Aplicație:

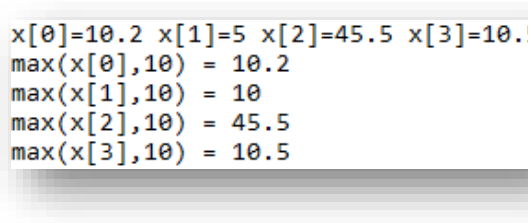
Să se modifice programul astfel încât să se afiseze toate caracterele din string cu litere mici.

Ex.2: Programul în C++ afiseaza maximul dintre fiecare element al unui tablou de nr reale x[i] utilizand fmax() care are sintaxa: double fmax (double x, double y); sau float fmax (float x, float y);

Varianta in C++

```
#include <iostream>
#include <cmath>
using namespace std;
int main() {
double x[100]={10.2,5.,45.5,10.5};
for (int i=0;i<4;i++)
   cout<<"x["<<i<<"]="<<x[i]<<" ";
cout<<endl;
for (int i=0;i<4;i++)
   cout <<"max(x["<<i<<"],10) = "<< fmax(x[i],10)<<endl;
return 0;}
```

Rezultate:



Aplicație:

Să se modifice programul astfel încât să se afiseze minimul dintre valorile tabloului si valoarea 5.5

Ex.3: Programul în C++ afiseaza un sir de numere intregi dupa ordonare crescatoare si cauta o valoare in sir utilizand STL algorithm, sort(), begin(), end() si binary_search(startaddress, endaddress, valuetofind)

Varianta in C++

```
#include <algorithm>
#include <iostream>
```

```
using namespace std;
int main()
{ int x;
int arr[] = {3, 5, 1, 2, 4};
cout<<"Sirul initial:";
for (int i : arr)     {cout << i << " ";}
cout<<endl;
sort(begin(arr), end(arr));
cout<<"Sirul sortat crescator:";
for (int i : arr)     {cout << i << " ";}
cout<<"\nx=";cin>>x;
cout<<"Este "<<x<<" in sir?";
if (binary_search(arr, arr+5, x))
cout <<endl<<x<<" este in sir";
else cout <<endl<<x<<" nu este in sir";
return 0;}
```

Rezultate:

```
Sirul initial:3 5 1 2 4
Sirul sortat crescator:1 2 3 4 5
x=2
Este 2 in sir?
2 este in sir
```

Aplicație:

Să se modifice programul astfel încât să se afiseze sirul ordonat descrescator. Ex. sort(begin(arr), end(arr), greater<int>());

Ex.4: Programul în C++ creaza un sir de numere intregi (utilizand vector) citite de la tastatura , ultimul fiind 0, apoi sterge acest element s afiseaza vectorul, utilizand tipul vector si STL

Varianta in C++

```
#include <iostream>
#include <vector>
using namespace std;
int main ()
{vector<int> a;
int myint;
cout << "introduceti nr intregi (0 la sfarsit):\n";
do { cin >> myint;
   a.push_back (myint);
  } while (myint);
for (const int i : a)
   cout << i << ' ';
cout << "Vector: " << int(a.size()) << " numere.\n";
cout<<"sterg ultimul element= 0";
a.pop_back();
cout<<"vectorul este:";
for (const int i : a)
   cout << i << ' ';
return 0;}
```

Rezultate:

Aplicație:

```
introduceti nr intregi (0 la sfarsit):
12 4 5 3 8 2 0
12 4 5 3 8 2 0 Vector: 7 numere.
sterg ultimul element= 0vectorul este:12 4 5 3 8 2
```

Să se modifice programul astfel încât să se afiseze elementele pare din vector. Ex: se adauga in cod if (i%2==0) cout << i << ' ';

Ex.5: Programul în C++ initializeaza o lista de nr. intregi, afisand elementele pare, primul si ultimul element, si lista sortata crescator si descrescator utilizand list, si metodele de sortare din STL.

Varianta in C++

```
#include <iostream>
#include <list>
using namespace std;
void afis(list<int> g)
{   list<int>::iterator it;
for (it = g.begin(); it != g.end(); ++it)       cout << ' ' << *it;}
void afispar(list<int> g)
{   list<int>::iterator it;
   for (it = g.begin(); it != g.end(); ++it)   if (*it%2==0)cout << ' ' << *it;}
int main()
{   list<int> list{12,45,8,6};
   cout<<"lista initiala:"; afis(list);
   cout<<"\nElemente pare:" ; afispar(list);
   cout<<"\nPrimul element:"<<list.front();
   cout<<"\nUltimul element:"<<list.back();
   list.sort();         cout<<"\nLista sortata:"; afis(list);
   list.reverse();    cout<<"\nLista sortata invers:"; afis(list);
   return 0;}
```

Rezultate:

```
lista initiala: 12 45 8 6
Elemente pare: 12 8 6
Primul element:12
Ultimul element:6
Lista sortata: 6 8 12 45
Lista sortata invers: 45 12 8 6
```

Aplicație:

Să se modifice programul astfel încât să se afiseze elementele impare din lista.

Ex.6: Programul în C++ initializeaza un sir de nr. intregi, utilizand array si STL si afiseaza suma lor si elementele pare.

Varianta in C++

```
#include<iostream>
#include<array>
using namespace std;
int main()
{ int sum=0;
   array<int,6> ar = {1, 2, 3, 4, 5, 6};
cout << "Elemente array : ";
for (int i : ar)  {sum+=i;cout << i << ' ';}
cout <<"Suma="<<sum<< endl;
cout <<"Elemente pare array: ";
for (int i : ar)  if (i%2==0) cout<< i << ' ';
cout << endl; return 0;}
```

Rezultate:

```
Elemente array : 1 2 3 4 5 6 Suma=21
Elemente pare array: 2 4 6
```

Aplicație:

Să se modifice programul astfel încât să se afiseze si produsul elementelor impare din array.

Ex.7: Programul în C++ initializeaza un sir de nr. intregi, utilizand queue si STL si afiseaza dimensiunea, primul si ultimul element, si sterge primul element.

Varianta in C++

```
#include <iostream>
#include <queue>
using namespace std;
void showq(queue<int> q1)
{  queue<int> g = q1;
   while (!g.empty()) { cout << ' ' << g.front();  g.pop();    }
    cout << '\n';}
int main()
{queue<int> c; int x;
do{ cout<<"x="; cin>>x; c.push(x); }
    while (x);
cout << "queue  : "; showq(c);
cout << "\nqueue size() : " << c.size();
cout << "\nqueue.front() : " << c.front();
cout << "\nqueue.back() : " << c.back();
cout << "\nqueue.pop() : ";
cout<<"Sterg primul element\n";
c.pop();cout<<"queue dupa stergere prim element:";
showq(c);return 0;}
```

Rezultate:

```
x=4
x=5
x=6
x=2
x=9
x=0
queue  :  4 5 6 2 9 0

queue size() : 6
queue.front() : 4
queue.back() : 0
queue.pop() : Sterg primul element
queue dupa stergere prim element: 5 6 2 9 0
```

Aplicație:

Să se modifice programul astfel încât să se afiseze si suma elementelor.

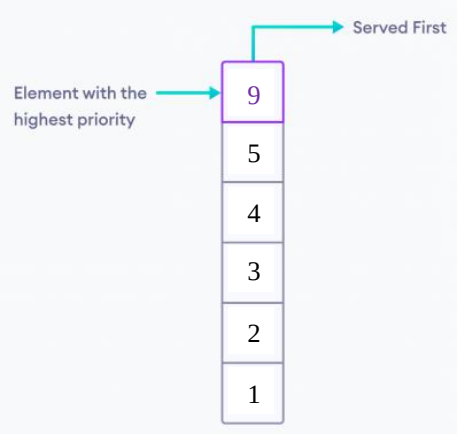
Ex.8: Programul în C++ citeste un sir de nr. intregi, si il stocheaza intr-o priority_queue (STL) si afiseaza primul element =maximul din priority_queue, si toate elementele.

Varianta in C++

```
#include <iostream>
#include <queue>
using namespace std;
int main() {
int arr[5], x;
cout<<"Array elements: "<<endl;
for (x = 0; x < 5; x++) {
   cout << "a[" << x << "]=";
   cin >> arr[x];}
priority_queue<int> pq;
cout << "Array: ";
for (auto i : arr) cout << i << ' ';
cout << endl;
for (int i = 0; i < 5; i++) {
   pq.push(arr[i]);    }
if (!pq.empty()) {
   int top = pq.top();
   cout << "Top element priority_queue=maxim: " << top;}
   else {cout << "\nPriority Queue is empty.";}
cout << "\nPriority Queue: ";
while (!pq.empty()) {
   cout << pq.top() << ' '; pq.pop();}
return 0;}
```

Rezultate:

```
a[0]=1
a[1]=3
a[2]=4
a[3]=5
a[4]=9
Array: 1 3 4 5 9
Top element priority_queue=maxim: 9
Priority Queue: 9 5 4 3 1
```



Aplicație:

Să se modifice programul astfel încât să se afiseze si suma elementelor.

PROBLEME PROPUSE

1. ***Să se scrie un program în C++ care utilizeaza un vector initializat astfel: vector<int> a={1,2,50,4,11,7,22,10,3,44,32}; si scrie intr-un fisier vectorul sortat utilizand metoda sort() din biblioteca STL <algorithm>***

2. ***Să se scrie un program în C++ care utilizeaza un vector care este initializat cu 10 valori numerice intregi si scrie intr-un fisier valorile pare, si suma lor.***

3. ***Să se scrie un program în C++ care utilizeaza un vector declarat astfel: vector<int> a = {1, 2, 3, 4, 5, 8, 10, 12}; si afiseaza cate valori sunt impare.***

4. ***Să se scrie un program în C++ care utilizand containere (vector, list, etc) citeste elementele unui sir de nr reale dintr-un fisier si le rotunjeste la partea intreaga , apoi le scrie intr-un alt fisier .***