Lab 1
Algoritmi si structuri de date

Un pas spre C++

- Limbajul C++ este o extensie a limbajului C care a fost creat pentru programarea orientată pe obiecte şi abstractizarea datelor
- Deşi conceptele cheie în C++ sunt

clasa şi obiectele

C++ a venit cu unele îmbunătăţiri faţă de C care nu sunt legate strict de programarea orientată pe obiecte.

Declaratiile de variabile

- Declaratiile de variabile pot apare oriunde este nevoie nu numai la începutul funcţiilor.
- S = 0;
- for (int i = 1; i <= n; i ++)
- S += i;
- Atentie la variabilele
- locale!

```
int main()
{
    int i=3;
    for(int i=2; i>0; i--)
        cout << i << endl;
    cout << i << endl;
    return 0;
}
```

Tipul de date bool

- S-a definit un *tip de date* special pentru rezultatul unei expresii logice, tip de date denumit ***bool***, care are doar două valori *true* şi *false*.
- Orice valoare întreagă diferită de zero este convertită în valoarea true, iar 0 la false.

iostream si fstream

- Au fost create noi bibliotecile de intrare şi ieşire:
- **iostream** pentru I/O standard (input de la tastatură, output la monitor) şi
- ***fstream*** pentru I/O cu fişiere.
- Se pot folosi (obiecte ale clasei iostream)
  - **cin** pentru citire
  - **cout** pentru scriere

I/O standard

- cin >> v; asteaptă introducerea de la tastatură a unei valori care este convertită la tipul variabilei v şi atribuită variabilei v.
- Aceasta înlocuieşte apelarea funcţiei: scanf("%d", &v);
- cout << "Hello world! "; tipăreşte mesajul Hello world! pe ecran.
- cout << endl; tipăreşte caracterul linie nouă.
- cout << "v="<< v; tipăreşte v= urmat de valoarea lui v.
- cout << expresie; tipăreşte valoarea expresiei.

I/O fisiere

- ifstream input("fisierintrare.txt"); deschide fişierul *fisierintrare.txt* pentru intrare şi permite citirea datelor din fişier folosind input, aşa cum a fost folosit *cin* pentru citirea de la tastatură.

- input >> n; citeşte din fişier următoarea valoare şi o atribuie lui n.
- ofstream output("fisieriesire.txt"); deschide fişierul *fisieriesire.txt* pentru scriere şi permite scrierea datelor în fişier folosind output, aşa cum a fost folosit *cout* pentru citirea de la tastatură.

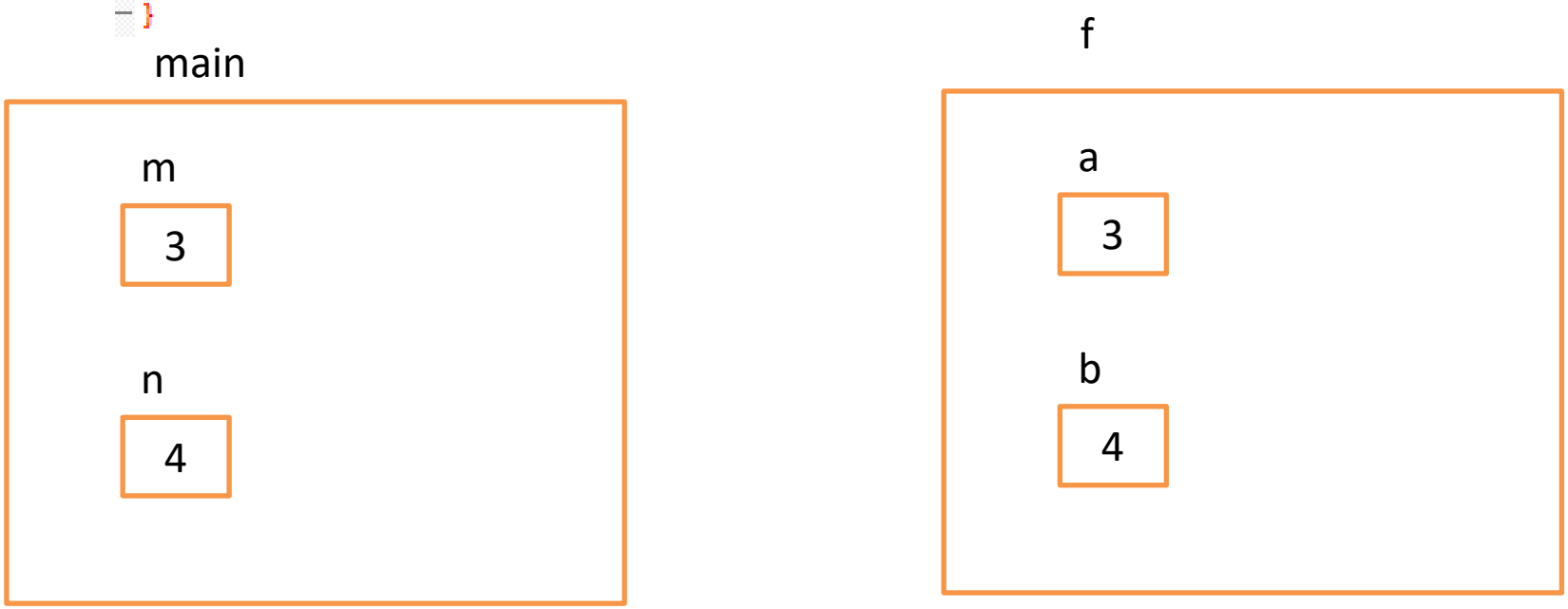
- output << "Hello world! "; scrie mesajul Hello world! în fişier.
- output << endl; scrie caracterul linie nouă.
- output << "v="<< v; scrie v= urmat de valoarea lui v tot în fişier..
- input.close(); închide fişierul de care este legată variabila input.

Apelarea prin referinta

- A fost introdusă apelarea prin referinţă.
- **nume_tip &nume_var;**
- La apelarea unei functii, parametrii pot fi transmisi
- prin valoare – o copie a param este creat in functie (*asa cum e in C)*
- sau
- prin referinta – parametrul va accesa aceeasi locatie de memorie ca si argumentul *(nou in C++)*

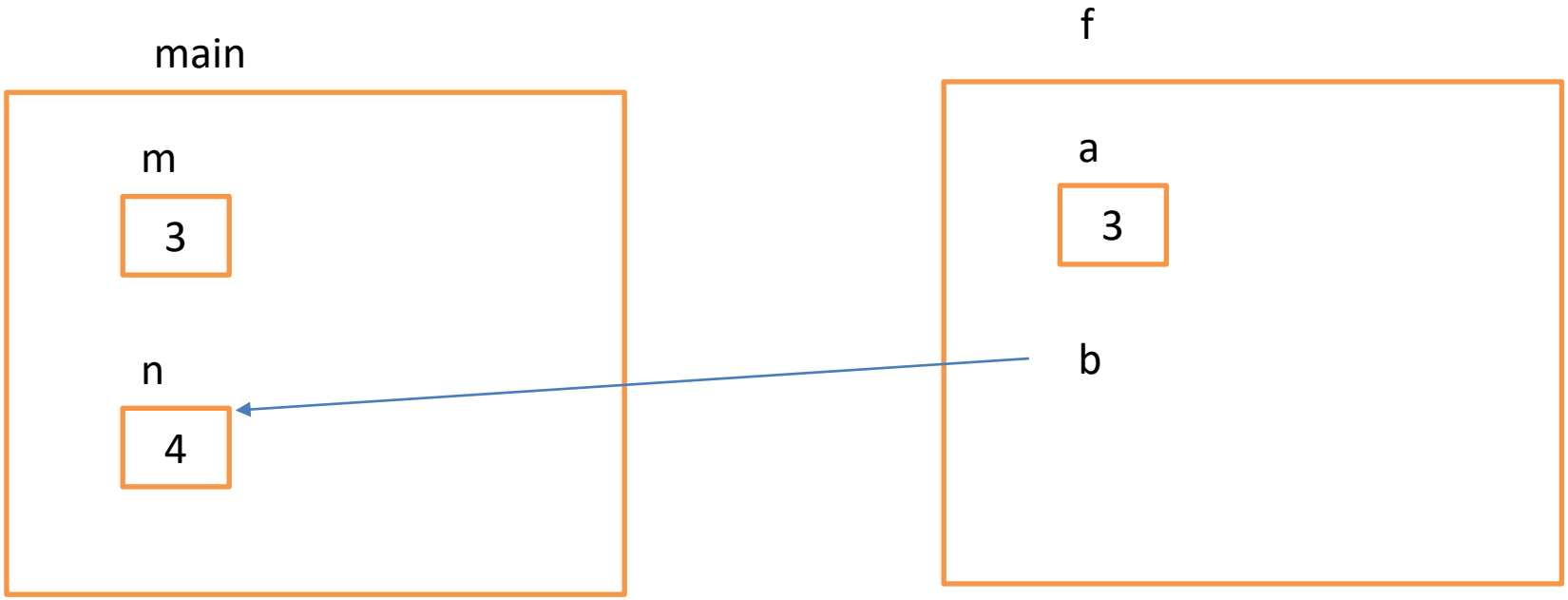
Apelare prin valoare

```
void f(int a, int b)
{
    ........
}
int main()
{
    int m=3, n=4;
    f(m,n); // m si n sunt transmisi ca valoare

}
```



Apelare prin referinta

```
void f(int a, int &b)
{
     ........
}
int main()
{
     int m=3, n=4;
     f(m,n); // m este transmis ca valoare, iar n ca referinta

}
```



Apelarea prin referinta

- void interschimba(int &a, int &b)
- {
- int c = a;
- a = b;
- b = c;
- }

- interschimba(x, y); // schimba valorile intre ele

Vectori - alocare statica

```
int v[5];
    for (int i=0;i<5;i++)
        cin >> v[i];
    for (int i=0;i<5;i++)
        cout << v[i]<<" ";
```

Vectori - alocare statica

```
int v[10];  // sau const int MAX=10; int v[MAX];
int n;
cout <<"nr elem vector=";
cin >>n;
for (int i=0;i<n;i++)
cin >> v[i];
for (int i=0;i<n;i++)
cout << v[i]<<" ";
```

Vectori - alocare statica

```
int n;
cin >>n;
int v[n]; //  merge pt unele compilatoare (de ex
// gcc dar nu si pt Visual C++)
// desi se cere ca dim vectorului sa fie cunoscuta la
// compile-time
for (int i=0;i<n;i++)
cin >> v[i] ;
for (int i=0;i<n;i++)
cout << v[i]<<" ";
```

Pointeri

Vectori - alocare dinamica

```
int n;
cin >>n;
int v[n]; // se inlocuieste cu
for (int i=0;i<n;i++)
cin >> v[i] ;
for (int i=0;i<n;i++)
cout << v[i]<<" ";
```

```
int n;
cin >>n;
int *v=new int [n];
for (int i=0;i<n;i++)
cin >> v[i] ;
for (int i=0;i<n;i++)
cout << v[i]<<" ";
```

Operatorii new si delete

- Au fost introduşi operatorii pentru managementul memoriei: ***new, delete.***
- Operatorul ***new*** creează un obiect de un tip specificat şi returnează un pointer la acel obiect sau NULL dacă obiectul nu a putut fi creat.
- Operatorul ***delete*** eliberează locaţia de memorie punctată de un pointer creat prin folosirea operatorului new.
- Aceştia se folosesc în locul funcţiilor de alocare dinamica (malloc, calloc, realloc, free).

- new int aloca spatiu pentru un int şi returneaza un pointer la locaţia de memorie şi se poate folosi în locul apelării: malloc(sizeof(int)),
- new int [n] aloca spaţiu pentru un şir de n întregi şi returneaza un pointer la primul element al şirului şi se poate folosi în locul apelării: malloc(n*sizeof(int)),
- delete p eliberează locaţia de memorie punctată de p şi se poate folosi în locul apelării: free(p),
- delete [] p pentru p ce punctează un şir, eliberează locaţia de memorie ocupată de întregul şir către care punctează p şi se poate folosi în locul apelării: free(p).