LABORATOR 2 - STRUCTURI DE DATE - INFORMATICĂ anul I sem. II

POINTERI ŞI LISTE ÎNLĂNŢUITE

1. VARIABILE DE TIP POINTER

➔ Memoria internă poate fi privită ca o succesiune de octeţi. Pentru a-i distinge aceştia sunt numerotaţi. *Numărul de ordine al unui octet se numeşte adresa lui.*

◆ Orice variabilă ocupă un număr de octeţi succesivi. De exemplu, o variabilă de tip **int** ocupă doi octeţi.

Definiţie: Adresa primului octet al variabilei se numeşte adresa variabilei.

◆ Uneori, în loc de adresă a unei variabile se foloseşte termenul pointer.
◆ Memorarea adreselor variabilelor se face cu ajutorul variabilelor de tip pointer.

➔ *Variabilele de tip pointer se caracterizează prin faptul că valorile pe care le pot memora sunt adrese ale altor variabile.*

◆ Limbajul C++ face distincţie între natura adreselor care pot fi memorate. Astfel, există adrese ale variabilelor de tip **int**, adrese ale variabilelor de tip **float**, adrese ale variabilelor de tip **char**, etc. Din acest motiv şi tipul variabilelor de tip pointer este diferit.

➔ *Tipul unei variabile de tip pointer se declară prin:*

tip *nume;

Exemple:

1. Variabile de tip pointer către variabile de tip **int**. Variabilele **adr1**, **adr2** pot reţine adrese ale variabilelor de tip **int**:

```
int *adr1, *adr2;
```

2. Variabile de tip pointer către variabile de tip **elev**, care sunt de tip **struct**. Variabilele **a** şi **b** pot reţine adrese ale variabilelor de tipul **elev**.

```
struct elev
{
    char nume[20], prenume[20];
    float nota_mate, nota_info;
 }
```

elev *a, *b;

➔ *Adresa unei variabile se obţine cu ajutorul operatorului de referenţiere '&', care trebuie să preceadă numele variabilei:*

&nume_variabila;

Exemplu: **adr1 = &numar;** - variabilei **adr1** i se atribuie adresa variabilei **numar**.

➔ *Fiind dată o variabilă de tip pointer către variabile de un anume tip, care memorează o adresă a unei variabile de acel tip, pentru a obţine conţinutul variabilei a cărei adresă este memorată se utilizează operatorul unar '*', numit şi operator de dereferenţiere.*

***nume_variabila;**

Exemple:

1. Variabile **a** este iniţializată cu 7, iar variabila adra este iniţializată cu adresa lui **a**. Secvenţa afişează conţinutul variabilei **a** (7) pornind de la adresa ei reţinută de adra:

```
int a = 7, *adra = &a;
cout<<*adra;
```

2. Variabile **a**, de tip **elev**, este inţializată, iar variabila **adra**, de tip pointer către variabile de tip **elev** este iniţializată cu adresa variabilei **a**. Secvenţa tipăreşte conţinutul variabilei **a**.

```
struct elev
{
   char nume[20], prenume[20];
 }

elev a, *adra = &a;
strcpy(a.nume, "Bojian");
strcpy(a.prenume, "Andronache");
cout<<(*adra).nume<<" "<<(*adra).prenume<<endl;
```

◆ Operatorul '.' - numit operator de selecţie, are prioritatea 1, deci maximă.

◆ Operatorul '*' - unar, numit şi operator de dereferenţiere, are prioritatea 2. Prin urmare, în absenţa parantezelor rotunde, se încearcă mai întâi evaluarea expresiei **adra.nume**, expresie care nu are sens. Parantezele schimbă ordinea de evaluare, se evaluează mai întâi ***adra**.

➔ *Pentru o astfel de selecţie, în loc să folosim trei operatori, se poate utiliza unul singur, operatorul de selecţie indirectă: '->'.* Acesta accesează un câmp al unei structuri pornind de la un pointer (adresă) către acea structură. El are prioritatea maximă. Afişarea se poate face şi astfel:

cout<<adra->nume<<" "<<adra->prenume;

➔ *Între variabile de tip pointer sunt permise atribuiri doar în cazul în care au acelaşi tip pointer* (*reţin adrese către acelaşi tip de variabile*).

Exemplu:

int *adr1, *adr2;
float *adr3;

Atribuirea **adr1 = adr2** este corectă.
Atribuirea **adr3 = adr2** nu este corectă.
Atribuirea **adr3 = (float*) adr2** este corectă.

2. ALOCAREA DINAMICĂ A MEMORIEI

Să se studieze (recapituleze) în cadrul laboratorului noţiunele de bază despre pointeri (**secţiunea 1**), alocarea dinamică a memoriei (**secţiunea 3**) şi legatura dintre pointeri şi vectori (**secţiunea 4**). Celelalte secţiuni se vor studia acasă.

Sugestie bibliografie:
https://home.csulb.edu/~pnguyen/cecs282/lecnotes/Pointer.pdf

3. LISTE ÎNLĂNŢUITE

TEMĂ pentru laborator:

Să se studieze şi implementeze operaţiile de la pagina 250 (din Cormen) pentru liste simplu înlănţuite, pentru liste dublu înlănţuite şi pentru liste circulare.

Bibliografie:
https://dl.ebooksworld.ir/books/Introduction.to.Algorithms.4th.Leiserson.Stein.Rivest.Cormen.MIT.Press.9780262046305.EBooksWorld.ir.pdf