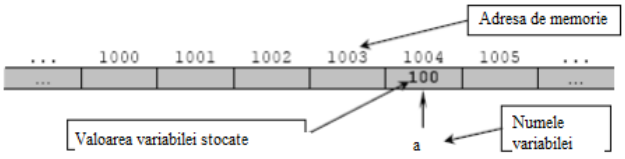
PROGRAMARE PROCEDURALĂ
- lab. 5 -

1. Pointeri

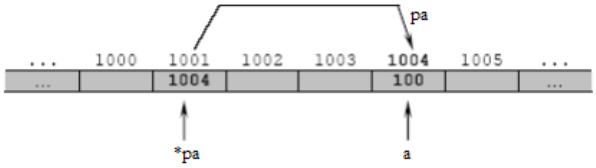
- la declararea unei variabile compilatorul alocă memorie cu o adresă unică; această adresă va fi asociată cu numele variabilei, iar valoarea va fi stocată la acea adresă unică de memorie.

int a;



- pentru a stoca într-o variabilă adresa de memorie la care se găsește o altă variabilă declarăm:

int *pa; pa este variabilă de tip pointer la int



- pentru ca *pa* să indice către *a* facem următoarea atribuire:

pa = &a; folosind simbolul & îi spunem compilatorului să întoarcă adresa variabilei precedată de &

- în orice expresie în care apare *a*, se poate înlocui cu **pa*.

Ex: int a =5, *pa = &a; (*pa)++; // se va modifica valoarea lui a în 6

- operaţiile permise pentru un pointer sunt: adunarea și scăderea;

Ex: int *p; *(p+1) este pointer-ul ce indică următoarea zonă de memorie după cea indicată de p; *(p-1) va fi anterioara

- singura operaţie permisă între doi pointeri este scăderea lor și reprezintă numărul de obiecte aflate între cei doi pointeri(practic, între cele două locaţii de memorie). De asemenea sunt permise operaţiile de comparaţie (==,!=,<,>,<=,>=)

2. Transmiterea parametrilor către funcţii prin referinţă

Parametrii pot fi transmiși prin valoare sau prin referinţă. La transmiterea prin valoare, valorile parametrilor actuali sunt depuse pe stivă, iar la apelul funcţiei vor fi consideraţi parametri formali. Astfel valorile lor nu vor fi modificate după execuţia funcţiei. Pentru a modifica valorile parametrilor, se transmit către funcţie pointeri la parametri.

Ex:
```
void add(int a){
        a+=10;
        }
void main(){
        int x = 5;
        printf(„%d”, add(x));
        }
```
Se va afișa 5.

```
void add(int *a){
        a+=10;
        }
void main(){
        int x = 5;
        printf(„%d”, add(x));
        }
```
Se va afișa 15.

3. Tablouri și pointeri

- numele unui tablou reprezintă adresa către primul element din tablou;

Ex: int a[]; a este adresa primului element din vector ⇔ &a[0]

- elementele tablourilor sunt stocate în locaţii de memorie consecutivă;
- ***(p+1)** ⇔ a[1], unde avem declarate int a[],*p=a;

```
int *p,a[]={1,2,4};
p=a;
printf("%d",*(p+1));
```
Se va afișa 2.

- pentru a declara un tablou de pointeri avem : *tip * nume_variabila[dimensiune]*

  Ex: char *a[20]; // fiecare element din vector va fi un pointer, deci va indica o locaţie de memorie ce conţine un element de tip char

- pentru a declara un tablou multidimensional, avem: *tip nume_variabilă[dim1][dim2][dim3]...*

  Ex: int a[10][5]; // tablou bidimensional cu 10 linii si 5 coloane

- a[i][j]⇔*(*(a+i)+j)

4. Probleme

1. Să se calculeze și să se afișeze valoarea expresiei $x^m+y^n+(xy)^{m\wedge n}$, unde x,y,m,n sunt citiţi de la tastatură, iar m și n sunt întregi pozitivi. Ridicarea la putere se va face de o funcţie care primește baza și exponentul ca parametri și returnează rezultatul.

2. Se citesc de la tastatură un număr *n* ce reprezintă dimensiunea unui vector și elementele vectorului. Să se declare un pointer la acest vector cu ajutorul căruia să se parcurgă și să se afișeze elementele vectorului.

3. a. Să se scrie o funcţie care interschimbă valorile a două numere întregi. Să se încerce mai întâi transmiterea parametrilor prin valoare, apoi prin referinţă și să se aleagă varianta corectă.

3. b. Se citesc 2 vectori de la tastatură, dimensiunile acestora și un număr k. Să se facă operaţii pe cei doi vectori astfel încât primul vector să conţină toate elementele din cei doi vectori mai mici decât k, iar cel de-al doilea vector elementele mai mari decât k.

4. Se citesc de la tastură un număr natural n ce reprezintă dimensiunea unei matrice pătratice și elementele unei matrice pătratice. Folosind pointeri, să se afișeze elementul de la intersecţia diagonalelor(pentru n impar), iar apoi elementele de pe cele două diagonale.