LIMBAJE DE PROGRAMARE

LABORATOR 10

I. ASPECTE TEORETICE

Pointeri

Dupa cum se stie, un program impreuna cu datele sale este pastrat in memoria calculatorului.
Memoria RAM (Random Access Memory) este o memorie cu acces aleator si care, la nivelul cel mai inferior, este alcatuita din biti ce memoreaza o valoare din doua, interpretate de obicei ca 0 si 1, atat timp cat calculatorul este in functiune.
Opt biti formeaza un octet, doi octeti alcatuiesc un cuvant, iar patru octeti un cuvant lung.

Un pointer este o variabila care contine adresa altei variabile, sau altfel zis reprezinta o variabila care pastreaza adresa unei date, in loc de a memora data insasi.

Pointerii se utilizeaza pentru a face referire la date cunoscute prin adresele lor.

1. Declaratia de pointer si tipul pointer

Un pointer se declara ca orice variabila, cu precizarea ca numele este precedat de caracterul *.
In general, un pointer se declara prin:

```
tip *nume_p;
```

ceea ce inseamna ca nume_p este un pointer care pointeaza spre o zona de memorie ce contine o data de tipul tip.

Comparand declaratia de mai sus cu cea uzuala:

```
tip nume;
```

putem considera ca tip* dintr-o declaratie de pointeri reprezinta tip dintr-o declaratie obisnuita.
Asadar, putem spune ca tip* reprezinta un tip nou, tipul pointer. Acest tip se spune ca este tipul pointer spre tip.

Precizare
Amintim faptul ca:
* -> operatorul de indirectare. Expresia care-l urmeaza este un pointer iar rezultatul este o lvaloare.
& -> operatorul de obtinere a unui pointer. Operandul este lvaloare iar rezultatul este pointer.

Daca avem declaratiile:

```
int x;
int *p;
float y;
```

atunci atribuirea:

```
p = &x;
```

este corecta, in timp ce:

```
p = &y;
```

nu este corecta, deoarece p poate contine numai adrese de zone de memorie in care se pastreaza date de tip int.
Daca exista declaratia:

```
float *q;
```

atunci se poate folosi atribuirea:

```
q = &y;
```

Exista cazuri in care se doreste ca un pointer sa fie utilizat cu mai multe tipuri de date.
In acest caz, la declararea lui nu putem specifica un tip. Aceasta se realizeaza folosind cuvantul void:

```
void *nume_p;
```

Exemplu:

```
int x;
float y;
char c;
void *p;
...
p = &x;
...
p = &y;
...
p = &c;
...
```

Folosindu-ne de cuvantul cheie void, lui p i s-au putut atribui adrese de zone de memorie care contin date de tipuri diferite.

Cand se folosesc pointeri de tip void, este necesar sa se faca conversii explicite prin expresii de tip cast (valoarea unui operand se converteste catre tipul tip folosind operatorul unar (tip)),
pentru a preciza tipul datei spre care pointeaza un astfel de pointer:

```
(tip) operand
```

Astfel, daca se utilizeaza p declarat ca in exemplul de mai sus, atunci o atribuire de forma:

```
*p = 100;
```

nu este corecta, fiindca nu este definit tipul datei spre care pointeaza p.
In cazul de fata, valoarea lui p trebuie convertita spre tipul int* folosind expresia cast:

```
(int*)p
```

In felul acesta atribuirea de mai sus devine:

```
*(int*)p = 100;
```

2. Functii necesare in programe ce opereaza cu pointeri

2.1. Functia "malloc"

Este declarata in fisierul header alloc.h .
Functia:

```
malloc(val)
```

"aduna" val octeti consecutivi din memoria disponibila, returnand adresa lor de inceput.

2.2. Expresia "sizeof"

```
sizeof(tip)
```

retuneaza numarul de octeti necesari unei variabile de tipul tip.
De exemplu sizeof(int) returneaza valoarea 2.

2.3. Functia "calloc"

Este declarata in fisierul header alloc.h .

Functia are doi parametri:

```
calloc(nr, dimens)
```

Primul parametru indica pentru cate obiecte se va aloca spatiu, iar al doilea dimensiunea fiecarui obiect in octeti.

Ex.:

```
int *sir;
...
sir = (int*)calloc(5, sizeof(int));
```

"sir" pointeaza la o zona de memorie suficienta pentru a contine 5 intregi (5 x 2 = 10 octeti).

Expresia (int*) indica faptul ca adresa va fi un pointer de tipul int, reprezentand o conversie de tip (cast).
Expresia nu e necesara in Turbo C, ea fiind scrisa pentru portabilitatea programului.

II. DESFASURAREA LUCRARII

1. Scrieti un program care utilizeaza doua variabile intregi v si p, dintre care ultima este de tipul pointer.
Lui v i se atribuie valoarea 200, iar lui p adresa variabilei v. Sa se afiseze valorile, locatia de memorie si valoarea pointata de cele doua variabile.

2. Sa se scrie un program care utilizeaza o singura variabila de tip pointer intreg p pentru care este alocata o zona de memorie cu functia malloc.
Pentru o anumita marime aleasa afisati valoarea lui p si valoarea pointata de p.

3. Fie doua variabile intregi a si b initializate cu doua valori arbitrar alese. Prin intermediul altor variabile de tip pointer realizati un program care sa inverseze valorile celor doua variabile.

4. Conform principiilor programarii procedurale extrageti functia:

```
void invers(int *x, int *y)
```

din corpul functiei main definita la punctul anterior.

5. Realizati un program care cu ajutorul functiei calloc aloca spatiu pentru o variabila pointer de dimensiunea a 3 variabile de tip int.
Initializati, apoi afisati atat adresele cat si continutul locatiilor de memorie.

6. Scrieti un program in care functia main apeleaza o functie de citire

```
void citire(float *p)
```

a unei valori reale de la tastatura, care apoi va fi si afisata.