Laborator 8 – Limbaje de Programare

1 Recapitulare şi exemple

Să ne amintim câteva exemple de la curs:

1.1 Argumentele liniei de comandă

1. Tipărirea argumentelor liniei de comandă:

```
#include <stdio.h>

int main(int argc, char *argv[]) {
  int i;
  printf("Numele programului: %s\n", argv[0]);

  if (argc==1)
    printf("Nu are parametri\n");
  else
    for (i = 1; i < argc; i++)
      printf("Parametrul %d: %s\n", i, argv[i]);

  return 0; /* codul returnat de program */
}
```

1.2 Pointeri

1. Pointeri ca argumente la funcţii:

```
// schimba valorile de la 2 adrese
void swap (int *pa, int *pb) {
  int tmp; // variabila temporara

  tmp = *pa;
  *pa = *pb;
  *pb = tmp;
  // trei atribuiri de intregi
 }
  ... //in main
  int x = 3, y = 5;
  swap(&x, &y);  //apelul functiei
  // acum x = 5 si y = 3
```

2. Parcurgerea unui tablou folosind pointeri şi aritmetica pointerilor, returnarea unei adrese:

```
char *endptr(char *s) {
  //returneaza pointer la sfarsitul sirului s

   char *p = s;
   // echivalent cu: char *p; p = s;

   while (*p)
       p++;

   //adica pana ajunge la pozitia
```

```
    //din sir marcata cu '\0' (*p == 0)
    //adresa memorata in p creste cu 1

    return p;
 }
```

3. O variantă de implementare pentru funcţia standard *strchr* (din *string.h*), folosind aritmetica pointerilor pentru parcugerea şirului de caractere:

```
char *strchr_p(char *s, int c) {
// cauta un caracter in sir
// scrisa folosind pointeri

  for ( ;*s; ++s)
  // folosim chiar parametrul pentru parcurgere
  // conditia ciclului *s!=0, adica caracterul *s nu e '\0'
     if (*s == c)
       return s;
       // s indica elem. curent

   return NULL;
   // nu s-a gasit caracterul cautat
   // returneaza pointerul invalid
}
```

1.3 Alocare dinamică de memorie

1. Alocarea dinamică a unui tablou de întregi, de dimensiune aflată la rulare (citită de la tastatură):

```
int i, n, *t;
printf("Nr. de elemente ?");
scanf("%d", &n);

t = malloc(n * sizeof(int));
//alocam memorie pentru n elemente de tip int

if (t != NULL)
//daca alocarea memoriei nu a dat eroare
//citim elementele tabloului
      for (i = 0; i < n; i ++)
           scanf("%d", &t[i]);
//...
free(t);
//la sfarsit, eliberam intotdeauna memoria alocata
```

2. Crearea unei copii pentru un şir de caractere primit ca parametru:

```
char *strdup(const char *s) {
// creeaza o copie a lui s

   char *d = malloc(strlen(s) + 1);
   // loc pentru sir si '\0'

   if (d!=NULL){ //alocare ok
      // fa copia, returneaza d
       strcpy(d, s);
       return d;

   }else //eroare la alocare
      return NULL;
}
```

```
//in main
char *sirInitial = "ceva";
char *sir = strdup(sirInitial);
...
free(sir);
//la sfarsit, eliberam memoria alocata
```

3. Citirea de la tastatură a unei linii de dimensiune nelimitată:

```
#include <stdio.h>
#include <stdlib.h>

#define BLOCK 16

char *getline(void) {

    char *p, *s = NULL;
    // s initializat cu NULL pentru realloc
    // deoarece realloc(NULL, size) e echivalent cu malloc(size)

    int c, lim = -1, size = 0;
    // pastram o pozitie suplimentara pentru '\0'

   while ((c = getchar()) != EOF) {

     if (size >= lim) { // s-a umplut zona alocata
        lim += BLOCK;   //dimensiunea maxima creste cu BLOCK octeti

        if ((p = realloc(s, lim + 1)) == NULL) {
         // aloca o noua zona de memorie, mai mare cu 16 octeti

          ungetc(c, stdin);
          break;

         // daca realocarea de memorie a esuat pune inapoi
         // ultimul caracter citit, apoi iese din ciclu

       } else
           s = p; // tine minte noua adresa alocata

       s[size++] = c; // adauga ultimul caracter
       if (c == '\n')
          break; // iese din ciclu la linie noua
       }
    }

    // termina cu \0, realoca doar cat e nevoie
    if (s != NULL) {
       s[size++] = '\0';
       s = realloc(s, size);
    }

   return s;
}

//in main
char *linie = getline();
...
free(linie)
//la sfarsit, eliberam intotdeauna memoria alocata
```

4. Alocarea dinamică a unei matrici

```
int **a = NULL;
// a va fi o matrice de intregi
// matricea va fi de tipul int **

int i, nl, nc;

printf("nr. linii = ");
scanf("%d", &nl);
printf("nr. coloane = ");
scanf("%d", &nc);

a = malloc(nl * sizeof(int *));
// se aloca intai memorie pentru
// un tablou de pointeri la intreg
// care va contine adresele de inceput ale liniilor matricii

if (a!=NULL){
// verificam daca s-a alocat zona de memorie

    for (i = 0; i < n; i++){
       a[i] = calloc(nc, sizeof(int));
       // se aloca memorie pentru fiecare linie a matricii
       // in plus, calloc initializeaza cu zero
       // zona de memorie alocata

       if ( a[i] == NULL)  //daca eroare la alocare
             break;
          }
      }
....
// la sfarsit, se elibereaza memoria alocata

  for (i = 0; i < nl; i++)
     free(a[i]);
      //intai se elibereaza fiecare linie in parte

  free(a);
  // apoi se elibereaza memoria alocata pentru
  // tabloul de pointeri
```

2 Lucrarea 8

Să se implementeze următoarele funcţii, să se apeleze în **main** şi să se tipărească la ieşire rezultatele obţinute.

- Scrieţi o funcţie care primeşte ca parametru un şir de caractere şi returnează adresa primei cifre întâlnite în şir, sau NULL dacă nu există nicio cifră. Parcurgerea şirului se va face folosind aritmetica pointerilor.
- Scrieţi o funcţie care primeşte ca parametru un tablou de numere întregi şi returnează adresa la care a memorat suma elementelor din tablou. Obs.: NU încercaţi să returnaţi adresa unei variabile locale, memoria pentru o variabilă locală se alocă pe stivă şi se pierde la revenirea din apel! Alocaţi memorie dinamic şi reţineţi valoarea calculată în zona de memorie alocată dinamic.
- Scrieţi o funcţie care primeşte ca parametru un şir de caractere, creează şi returnează un şir nou, alocat dinamic, care conţine doar literele mici din şirul primit ca parametru.
- Scrieţi o funcţie care primeşte ca parametru un tablou de şiruri de caractere şi returnează un şir nou, alocat dinamic şi de dimensiune potrivită, care conţine toate şirurile din tablou, concatenate. Puteti testa funcţia cu tabloul conţinând argumentele liniei de comandă (parametrul char* argv[ ] al funcţiei main fiind un tablou de şiruri de caractere).