Curs 9. Pointeri pe functii

9.1. Pointeri pe funcţii

In C funcția în sine nu este o variabilă, dar există posibilitatea de a defini un pointer pe o funcție, care poate fi manipulat, transmis funcțiilor, plasat în tablouri etc. realizand o procedura de sortare a unor linii de caractere furnizate la intrare, si in care procedurile de comparare si de interschimbare a liniilor intre ele sunt transmise in mod dinamic, ca si functii, procedurii de sortate.

De obicei, un sort constă în trei părți, o comparație care realizează ordonarea oricarei perechi de obiecte, o schimbare, prin care se inverseaza ordinea obiectelor si un algoritm de sortare care face comparatii si schimbari pina cind obiectele sint definitiv ordonate. Algoritmul de sortare este independent de operatiile de comparare si schimbare, astfel incit prin transmiterea functiei de comparare si schimbare catre el, se va putea realiza sortarea pe diferite criterii. Acest lucru ni-l propunem în noul sort. In noul sort, functiile de comparare si schimbare vor fi transmise prin pointeri la functii.

Writelines scrie liniile sortate la iesire
Readlines citeste randuri si le salveaza in lineptr pana cand se intalneste EOF. Readlines aloca memoria necesara stocarii liniilor citite.

```
#define LINES 1000 //maximum de linii de sortat
#define MAXLEN 1000 // numarul de caractere maxim pe o linie
#include <stdio.h>
#include <stdlib.h>
#include <string.h>


void writelines(char *lineptr[], int nlines) //scrie linii la iesire
{
      while (--nlines >= 0)
            printf("%s\n", *lineptr++);
}


int readlines(char *lineptr[], int maxlines) // citeste linii de intrare
{
       int len, nlines;
       char *p, line[MAXLEN];
       nlines = 0;
       while ( fgets(line, MAXLEN, stdin) != 0)
       {
             len = strlen(line);
             if (nlines >= maxlines)
                  return(-1);
             else if ((p = malloc(len)) == NULL)
                  return(-1);
             else
             {
                   line[len-1] = '\0'; // newline
                   strcpy(p, line);
                   lineptr[nlines++] = p;
             }
       }
      return(nlines);
}
```

```
int numcmp(char *s1, char *s2) //comparã numeric s1 cu s2
{
      double v1, v2;
    v1 = atof(s1);
    v2 = atof(s2);
      if (v1 < v2)
            return(-1);
      else if (v1 > v2)
            return(1);
      else
            return(0);
}


swap(char *px[], char *py[]) /* interschimba *px si *py **/
{
      char *temp;
      temp = *px;
    *px = *py;
    *py = temp;
}


void sort(char* s[], int n, int (*comp)(), int (*exch)()) //sorteaza sirurile s[0]... s[n-1]
{
      int gap, i, j;
      for (gap = n/2; gap > 0; gap /= 2)
            for (i = gap; i < n; i++)
                  for (j = i - gap; j >= 0; j -= gap)
            {
                        if ((*comp)(s[j], s[j+gap]) <= 0)
                              break;
                        (*exch)(&s[j], &s[j+gap]);
            }
}


int main(int argc, char *argv[]) //sortarea linii input
{
    char *lineptr[LINES]; //pointeri pe liniile de text
      int nlines; //nr linii-input citire
      int numeric = 0; //1 daca este sort numeric
      if (argc > 1 && argv[1][0] == '-' && argv[1][1] == 'n')
            numeric = 1;
      if ((nlines = readlines(lineptr, LINES)) >= 0)
    {
            if (numeric)
                  sort(lineptr, nlines, numcmp, swap);
            else
                  sort(lineptr, nlines, strcmp, swap);
            writelines(lineptr, nlines);
    } else
      printf("intrare prea mare pentru sort\n");
}
```

`numecmp` și `swap` sunt adrese de funcţii; din moment ce ele sunt cunoscute ca fiind functii si ele se transmit in sortCompilatorul este cel care rezolvă transmiterea adresei funcției.
Al doilea pas este modificarea lui sort:

Constatăm faptul că se poate declara un pointer la o functie in felul următor :

*Tipreturn (*pfunctie)(lista argumente)*

Astfel se declară pfunctie ca si un pointer către o funcție care returnează o valoare de tipreturn si primeste la intrare o lista de argumente precum cea specificată.

Datorită faptului ca numele unei funcții din program reprezintă un pointer către zona de cod care conține instrucțiunile functiei respective in program, se va putea face atribuirea :

pfunctie = numefunctie;

astfel, apelul functiei numefunctie cu o lista de parametrii de apel se poate face fie

numefunctie(listaparamdeapel)

sau prin intermediul pointerului la functie

(*pfunctie)(listaparamdeapel)

In acest sens, o prin intermediul unor pointeri la functie se poate realiza o functionalitate dinamică a unui program. In exemplul de mai sus, functionalitatea dinamică este obtinută prin apelul explicit al functiei de sortare sort cu metoda numcmp care compara sirurile folosind atof, in locul functiei clasice strcmp.

Se poate defini un array de pointeri la functie:

Tipreturn (*sirpointerifunctie[DIM])(lista argumente)

Astfel, sirpointerifunctie este un sir de DIM pointeri la functii. De exemplu, prin utilizarea sirurilor de pointeri la functii se pot implementa functionalități similare meniurilor fără ca să fie nevoie utilizarea constructiilor gen switch.

9.2. Argumentele functiei main

Sintaxa functiei main este

int main(int argc, char *argv[])

cu urmatoarea semnificatie:

- Argc reprezintă numărul argumentelor transmise in cadrul liniei de comandă. Dacă linia de comandă contine doar numele executabilului, atunci argc este 1 iar argv va fi un sir de lungime 1 de siruri de caractere
- Argv reprezinta un sir de caractere care va contine argumentele transmise in linia de comandă executabilului. argv[0] reprezintă intotdeauna un sir care contine numele executabilului care se execută.

Sirul argv va avea exact atatea elemenete câte sunt indicate în variabila argc.

Astfel, prin argumentele liniei de comanda (sau argumentele functiei main) programul poate citi argumente care sunt transmise din sistemul de operare către program.