Programarea calculatoarelor
Limbajul C

CURS 10

Pointeri la funcții

Fişiere text

Pointeri la funcții

- Problemă: funcție care să poată apela o funcție cu nume necunoscut, dar cu prototip şi efect cunoscut.
- Exemple:
  - Funcție care să sorteze un vector ştiind funcția de comparare a două elemente ale unui vector.
  - Funcție care să determine o rădăcină reală a oricărei ecuații (neliniare).
  - Funcție "listf" care poate afişa (lista) valorile unei alte funcții cu un singur argument, într-un interval dat şi cu un pas dat.

```
int main () {
      listf (sin, 0., 2*M_PI, M_PI/10.);
      listf (exp,1., 20., 1.);
      return 0;
}
```

Observații

- **Prin convenție, în limbajul C, numele unei funcții neînsoțit de o listă de argumente (chiar vidă) este interpretat ca un pointer către funcția respectivă (fără a se folosi operatorul de adresare '&')!**
- "sin" este adresa funcției "sin(x)" în apelul funcției "listf".
- O eroare de programare care trece de compilare şi se manifestă la execuție este apelarea unei funcții fără paranteze; compilatorul nu apelează funcția şi consideră că programatorul vrea să folosească adresa funcției!
- Exemplu:

```
if ( test ) break;          // gresit, echiv. cu if (1) break;
if ( test() ) break;
// iesire din ciclu daca funcția test intoarce true
```

Declarare pointeri la funcții

- Declararea unui argument formal (sau unei variabile) de tip pointer la o funcție are forma următoare:

  tip (*pf) (lista_arg_formale)

unde:

  - pf este numele argumentului (variabilei) pointer la funcție
  - tip este tipul rezultatului funcției

- Definirea funcției "listf":

```
void listf (double (*fp)(double), double min, double max, double
    pas) {
   double x, y;
   for (x=min; x<=max; x=x+pas) {
     y=fp(x);       // sau: y=(*fp)(x);
     printf ("\n%20.10lf %20.10lf", x, y);
   }
 }
```

Observații

- Parantezele sunt importante, deoarece absența lor modifică interpretarea declarației:

  Declarație *funcție cu rezultat pointer, nu pointer la funcție!!*

  tip * f (lista_arg_formale)
- Pentru a face programele mai explicite se pot defini nume de tipuri pentru tipuri pointeri la funcții, folosind declarația ***typedef***.

```
typedef double (* ftype) (double);
void listf (ftype fp, double min, double max, double pas) {
    double x, y;
    for (x=min; x<=max; x=x+pas) {
      y = fp(x);
      printf ("\n%20.10lf %20.10lf", x,y);
    }
  }
```

Funcții callback

- O funcție C transmisă, printr-un pointer, ca argument unei alte funcții F se numeşte şi funcție “callback”, pentru că ea va fi apelată “înapoi” de funcția F.
- De obicei, funcția F este o funcție de bibliotecă, iar funcția C este parte din aplicație.
- Funcția F poate apela o diversitate de funcții, dar toate cu acelaşi prototip, al funcției C.

Exemplu

- program cu meniu de opțiuni; operatorul alege una din funcțiile realizate de programul respectiv:

```
#include<stdio.h>
#include<stdio.h>
typedef void (*funPtr) ();

// funcții ptr. operatii realizate de program
void  unu () {
  printf ("unu\n");
}
void  doi () {
  printf ("doi\n");
}
void  trei () {
  printf ("trei\n");
}
```

Exemplu - continuare

```
// selectare şi apel funcție
int main () {
  funPtr tp[ ]= {unu,doi,trei};          // vector de pointeri la funcții
  short option=0;
  do {
     printf(“Optiune (1/2/3):“);
     scanf ("%hd", &option);
     if (option >=1 && option <=3)
            tp[option-1]();   // apel funcție (unu/doi/trei)
     else break;
  } while (1);
  return 0;
}
```

Exemplu - observație

Secvența echivalentă (cu *switch*) este :

```
do {
   printf(“Optiune (1/2/3):“);
   scanf ("%hd", &option);
   switch (option) {
     case 1: unu(); break;
     case 2: doi(); break;
     case 3: trei(); break;
   }
} while (1);
```

Directive preprocesor

- sunt interpretate într-o etapă preliminară compilării (traducerii) textului C, de un preprocesor
- nu se foloseste caracterul ';' pentru terminarea unei directive!
- #define ident text

  inlocuieşte toate aparițiile identificatorului "ident" prin şirul "text"
- #define ident (a1,a2,...) text

  defineşte o macroinstructiune cu argumente
- #include "fişier"

  include în compilare conținutul fişierului sursa "fişier"
- #if expr

  compilare condiționată de valoarea expresiei "expr"
- #if defined ident

  compilare condiționată de definirea unui identificator (cu #define)
- #endif

  terminarea unui bloc introdus prin directiva #if

Fişier

- Colecție de date memorate pe un suport extern (floppy, harddisk, etc) identificată printr-un nume.
- Entități ale sistemului de operare: numele lor respectă convențiile sistemului, fără legătură cu un limbaj de programare anume
- Conținutul:
  - texte (ex. programe sursă)
  - numere
  - alte informații binare: programe executabile, numere în format binar, imagini sau sunete codificate numeric s.a.
- Numărul de elemente ale unui fişier este variabil (poate fi nul).
- Se folosesc pentru
  - date inițiale sau rezultate mai numeroase
  - păstrarea permanentă a unor date de interes pentru anumite aplicații

Operarea cu fişiere

- De obicei "fişier" = fişier disc (pe suport magnetic sau optic)
- Noțiunea de fişier este mai generală şi include orice flux de date (stream) din exterior spre memorie sau dinspre memoria internă spre exterior.
- “Stream” (flux de date, canal) sinonim cu “file” (fişier): pune accent pe aspectul dinamic al transferului de date
- Programatorul se referă la un fişier printr-o variabilă; tipul acestei variabile depinde de limbajul folosit şi chiar de funcțiile utilizate (în C).
- Asocierea dintre numele extern (un şir de caractere) şi variabila din program se face la deschiderea unui fişier, printr-o funcție standard.

Tipuri de fişiere în C

- Fişiere text
  - conțin o succesiune de linii, separate prin NewLine
  - fiecare linie are 0 sau mai multe caractere tipăribile şi/sau tab
- Fişiere binare
  - conțin o succesiune de octeți

Fişiere text

- Caracter terminator de linie:
  - fişierele Unix/Linux: un singur caracter terminator de linie ‘\n’
  - fişierele Windows şi MS-DOS: caracterele ‘\r’ şi ’\n’ (CR,LF) ca terminator de linie
- Un fişier text poate fi terminat printr-un caracter terminator de fişier (CtrI-Z = EOF = -1)
  - nu este obligatoriu acest terminator
- Sfârşitul unui fişier disc poate fi detectat şi pe baza lungimii fişierului (număr de octeți), memorată pe disc.
- Se realizează conversia automată din/în format extern (şir de caractere) în/din format intern (binar virgulă fixă sau virgulă mobilă)

Fişiere binare

- Pot conține
  - numere în reprezentare internă (binară)
  - articole (structuri de date)
  - fişiere cu imagini grafice, în diverse formate, etc
- Citirea şi scrierea se fac fără conversie de format.
- Pentru fiecare tip de fişier binar este necesar un program care să cunoască şi să interpreteze corect datele din fişier (structura articolelor).
- Este posibil ca un fişier binar să conțină numai caractere, dar funcțiile de citire şi de scriere pentru aceste fişiere nu cunosc noțiunea de linie; ele specifică un număr de octeți care se citesc sau se scriu la un apel al funcției “fread” sau “fwrite”.

Operarea cu fişiere

1. *se defineşte o variabilă* de tip FILE * pentru accesarea fişierului;
   - FILE * un tip structură definită în stdio.h
   - conține informații referitoare la fişier şi la tamponul de transfer de date între memoria centrală şi fişier (adresa, lungimea tamponului, modul de utilizare a fişierului, indicator de sfârşit, de poziție în fişier)
2. *se deschide fişierul* pentru un anumit mod de acces, folosind funcția de biblioteca fopen, care realizează şi asocierea între variabila fişier şi numele extern al fişierului
3. se prelucrează fişierul - citire/scriere cu funcțiile specifice
4. se închide fişierul folosind funcția de biblioteca fclose.

Deschiderea unui fişier

FILE *fopen(const char *numefişier, const char *mod);

- Deschide fişierul cu numele dat pentru acces de tip mod
- Returnează pointer la fişier sau NULL dacă fişierul nu poate fi deschis
- Valoarea returnată este memorată în variabila fişier, care a fost declarată (FILE *) pentru accesarea lui.

- numefis: numele fişierului
- mod: şir de caractere (între 1 şi 3 caractere):
  - "r" - readonly , este permisă doar citirea dintr-un fişier existent
  - "w" - write, crează un nou fişier, sau dacă există deja, distruge vechiul conținut
  - "a" - append, deschide pentru scriere un fişier existent (scrierea se va face în continuarea informației deja existente în fişier, deci pointerul de acces se plasează la sfârşitul fişierului)
  - “+” = permite scrierea şi citirea din acelasi fişier - actualizare (ex: "r+", "w+", "a+").
  - "t" sau "b" = tip fişier ("text", "binary"), implicit este "t“

Nume fişier extern

- Poate include următoarele:
  - Numele unității de disc sau partiției disc (ex: A:, C:, D:, E:)
  - "Calea" spre fişier: succesiune de nume de fişiere catalog (director), separate printr-un caracter ('\' în MS-DOS şi MS-Windows, sau '/' în Unix şi Linux)
  - Numele propriu-zis al fişierului
  - Extensia, care indică tipul fişierului şi care poate avea între 0 şi 3 caractere în MS-DOS.
- Sistemele MS-DOS şi MS-Windows nu fac deosebire între litere mari şi litere mici, în cadrul numelor de fişiere
- Atenție! pentru separarea numelor de cataloage dintr-o cale se vor folosi:
  - "\\", pentru a nu se considera o secvență de caractere "Escape"
  - sau caracterul ‘/’.
  - char *numef = "C:\\WORK\\T.TXT";
  - char *numef = “c:/work/t.txt”;

Închiderea unui fişier

- **int fclose(FILE *fp);**
  - închide fişierul şi eliberează zona tampon
  - în caz de succes întoarce 0.
  - altfel, întoarce **EOF**.

- Atenție!
  - Închiderea unui fişier disc este absolut necesară pentru fişierele în care s-a scris ceva!
  - Poate lipsi dacă s-au făcut doar citiri din fişier!

Exemplu

```
#include <stdio.h>
int main ( ) {
  FILE * f;       // pentru referire la fişier
     // deschide un fişier text ptr citire
  f = fopen ( “c:\\t.txt", "rt“ );
  printf ( f == NULL ? "Fişier negasit" : " Fişier gasit");
  if (f)              // dacă fişier existent
    fclose(f);    // închide fişier
  return 0;
}
```

Prelucrarea fişierelor text

- se poate face fie la nivel de linie, fie la nivel de caracter:
  - **int fgetc (FILE *fp)**

    întoarce următorul caracter din fp ca un unsigned char convertit la int, sau EOF dacă s-a întâlnit sfârşitul de fişier sau în caz de eroare.
  - **char *fgets (char *s, int n, FILE *fp)**

    citeşte maxim n-1 caractere sau până la '\n' inclusiv, şi le depune în s, adaugă la sfârşit '\0' şi returnează adresa şirului. La eroare întoarce valoarea NULL.
  - **int fputc(int c,FILE *fp)**

    scrie caracterul cu codul ascii c în fişier
  - **int fputs(char *s,FILE *fp)**

    scrie şirul în fişier, fara caracterul '\0'. La eroare întoarce EOF.

Exemplu: citire şi afişare linii dintr-un fişier

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
  FILE *fp;
  char s[80];
  if ( (fp=fopen("c:\\test.c","r")) == NULL ) {
     printf ( "Nu se poate deschide la citire fişierul!\n“ );
      exit (1);
  }
  while ( fgets(s,80,fp) != NULL )
     printf ( "%s", s);
   fclose (fp);
   return 0;
}
```

Exemplu: scriere sub formă de litere mici caracterele dintr-un fişier în alt fişier

- numele sursei şi destinației transmise în linia de comandă.

```
#include<stdio.h>
#include<ctype.h>
int main(int argc, char** argv){
   FILE * f1, * f2;  int ch;
   f1= fopen (argv[1], "r");
   f2= fopen (argv[2], "w");
   if ( f1==0 || f2==0) {
      puts (" Eroare la deschidere fişiere \n");
     return 1;}
   while ( (ch=fgetc(f1)) != EOF)       // citeste  din f1
      fputc ( tolower(ch),f2);          // scrie în f2
   fclose(f1);
   fclose(f2);
   return 0;
}
```

- Lansarea în execuție a programului:

  copiere fişier_sursa.dat fişier_dest.dat

Intrări/ieşiri cu conversie de format

- Datele numerice pot fi scrise în fişiere disc
  - în format intern, binar (mai compact)
  - transformate în şiruri de caractere (cifre zecimale, semn ş.a): fişier text ocupă mai mult spațiu
- Formatul şir de caractere necesită şi caractere separator între numere, dar poate fi citit cu programe scrise în orice limbaj sau cu orice editor de texte sau cu alt program utilitar de vizualizare fişiere!

- int fprintf (FILE *fp, const char *format,...)

identică cu printf cu deosebirea că scrie într-un fişier.

- int fscanf (FILE *fp, const char *format,...)

realizează citirea cu format dintr-un fişier; analog scanf

- Exemplu:
  - int a; float b;
  - fscanf (fp, ”%d%f ”,&a, &b);
  - fprintf (fp, ”a= %d \t b=%f \n”, a, b);

Testare sfârşit de fişier

- int feof(FILE *fp)
  - testează dacă s-a ajuns la *end-of-file* al fişierului referit de fp
  - returnează 0 dacă nu s-a detectat sfârşit de fişier la ultima operație de citire, respectiv o valoare nenulă (adevarată) pentru sfârşit de fişier.
- Atenție! Se va scrie în fişierul de ieşire şi –1 - rezultatul ultimului apel al funcției “fgetc”:

```
while ( ! feof(f1))
                fputc(fgetc(f1),f2);
```

- Soluția preferabilă pentru ciclul de citire-scriere caractere este testarea rezultatului funcției de citire:

```
while ( (ch=fgetc(f1)) != EOF)
    fputc ( ch, f2);
```

Exemplu

- Într-un fişier de tip text sunt păstrate valorile reale ale unei măsuratori sub forma:

  nr_măsuratori

  val1

  val2

  val3 ...

A) Să se scrie programul care afişează numărul de măsurători şi valorile respective.

B) Se vor adăuga la fişier noi măsuratori până la introducerea valorii 0.

Rezolvare

```
# include <math.h>
# include <stdio.h>
# include <stdlib.h>
# include <ctype.h>
# define MAX  100

void afiseaza(double *mas,int nrm){
  int i;
  for (i=0; i<nrm; i++)
    printf ("Masuratoarea %d = %6.2e\n", i+1, mas[i]);
}

void loadmas (FILE *fp, int *nrm, double *mas){
  int i=0;
  fscanf (fp,"%d", nrm);
  if (*nrm>MAX) *nrm = MAX;
  for ( ; i<*nrm; i++)
    fscanf (fp, "%lf", &mas[i]);
}
```

Rezolvare

```
int main(){
  FILE *fp;
  double masur[MAX], mas_noua=1;
  char nume_fis[12];
  int nr_mas;
  printf ("nume fisier:");
  gets (nume_fis);
  if ( (fp = fopen(nume_fis, "r+") ) == 0 ){
     printf ("nu exista fisierul\n");
     exit (1);
  }
loadmas (fp,&nr_mas,masur);
afiseaza (masur,nr_mas);
fclose(fp);
if ( (fp = fopen(nume_fis, “a") ) == 0 ){
     printf ("nu exista fisierul\n");
     exit (1);
  }
```

Rezolvare

```
  printf ("\nmasuratori noi:\n”);
  while( nr_mas++<MAX-1){
    scanf(“%lf”,&mas_noua);
     if(mas_noua) fprintf (fp, "%lf\n", mas_noua);
     else break;
  }
  fclose(fp);
   if ( (fp = fopen(nume_fis, "r+") ) == 0 ){
     printf ("nu exista fisierul\n");
     exit (1);
  }
  fprintf (fp, "%d", --nr_mas);
  fclose (fp);
  return 0;
}
```

Exerciții

- Program pentru numărarea liniilor şi cuvintelor dintr-un fişier text printr-o singura parcurgere. Cuvintele sunt şiruri de orice caractere separate între ele prin (oricâte) spații albe.

Hint: Se poate folosi funcția de biblioteca "strtok".

Rezolvare tema

```
//converteste de la sir de caractere la nr real dubla precizie
double convf (char *s) {
  double val=0.0, putere;
  int i=0,semn;
  while ( isspace(s[i]) )
        i++;
  semn = (s[i]=='-')?-1:1;
  if (s[i]=='+‘ || s[i]=='-‘ )
        i++;
  for ( val=0.0; isdigit(s[i]); i++)
        val = 10*val + s[i]-'0‘;
```

Rezolvare tema

```
if (s[i]=='.') {
        i++;
        for (putere=1.0; isdigit(s[i]); i++) {
            val = val + (s[i]-'0‘)/putere;
            putere *= 10;
        }
  } /*sfirsit parte zecimala*/
  val *= semn;
  return val;
}
```