Programarea calculatoarelor
Limbajul C

CURS 5

Continuare Funcții

Domeniu de vizibilitate (scope)

- Un nume (variabilă, funcție) poate fi utilizat numai după ce a fost declarat.
- Domeniul de vizibilitate (scope) al unui nume este mulțimea instrucțiunilor (liniilor de cod) în care poate fi utilizat acel nume (numele este vizibil).
- *Regula de bază: identificatorii sunt accesibili doar în blocul în care au fost declarați; ei sunt necunoscuti în afara acestor blocuri.*

Domeniu de vizibilitate (scope)

- variabile globale –variabile ce sunt declarate în afara oricărui bloc

- variabile locale – variabilele ce sunt declarate: în funcții, în blocuri, ca parametri.

Variabile locale şi variabile globale

Exemplu:

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

int fact=1;
void factorial(int n)
{
  int i;
  fact=1;
  for(i=2;i<=n;i++) fact=fact*i;
}
```

Variabile locale şi variabile globale

```
int main(void) {
        int v;
        factorial(3);
        printf("3!=%d\n",fact);
        printf("Introd o valoare:");
        scanf("%d",&v);
        factorial(v);
        printf("%d!=%d\n",v,fact);
        return 0;
      }
```

Observații

- Definiția unui identificator maschează pe cea a aceluiaşi identificator declarat într-un suprabloc sau în fişier (global)!
- Apariția unui identificator face referință la declararea sa în cel mai mic bloc care conține această apariție!

```
#include<stdio.h>
#include<stdlib.h>
int fact=1;
void factorial(int n)
{
  int i;
  int fact=1;
  for(i=2;i<=n;i++) fact=fact*i;
}
```

Observații

```
int main(void)
{
  int v;
  factorial(3);
  printf("3!=%d\n",fact);
  printf("Introd o valoare:");
  scanf("%d",&v);
  factorial(v);
  printf("%d!=%d\n",v,fact);
  return 0;
}
```

Atenție! Va afişa 3!=1. Rezultatul este 1 oricare ar fi valoarea lui v !!!

Probleme rezolvate

1. Să se calculeze şi să se afişeze valoarea expresiei $x^m+y^n+(xy)^{m\hat{}n}$, x,y,m,n fiind citiți de la tastatură, astfel încât întregii m,n să fie pozitivi. Ridicarea la putere se va realiza printr-o funcție *putere* care primeşte baza şi exponentul ca parametri şi returnează rezultatul.

- Observație: apelul funcției putere (analog pow) apare într-o expresie şi de asemenea ca parametru actual într-un apel.

- Folosind funcția putere scrieți programul care citeste N întregi (N variabil, între 2 şi Max) şi calculează $i_1\hat{}i_2\hat{}i_3\hat{}...\hat{}i_N$. Atenție la depăşiri! Ce modificări trebuie operate asupra tipurilor pentru ca rezultatul să fie corect?
  Observație: întregii vor fi citiți într-un tablou!

Rezolvare

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

double putere (double baza, int exp); // calculează baza^exp

int main(void){
    int m,n;
    double x,y;
    while( printf( “m(>=0):"), scanf("%d",&m), m<0);
    while( printf( “n(>=0):"), scanf("%d",&n), n<0);
    if ( m==0 && n==0 ) { //semnalare eroare 0^0
        puts("0^0 imposibil");
        return; // din main
    }
```

Rezolvare

```
    printf("valorile lui x,y:");
    scanf("%lf%lf",&x,&y);
    printf("%lf^%d+%lf^%d+(%lf*%lf)^%d^%d (cu putere ):%lf\n",
           x,m,y,n,x,y,m,n,  putere(x,m)+putere(y,n) +
           putere(x*y,putere(m,n)));
    printf("rezultat cu pow: %lf\n", pow(x,m) + pow(y,n) +
           pow(x*y,pow(m,n)));
    return 0;
} // main

double putere (double baza, int exp){
      int i; float rez;
      for(i=rez=1;i<=exp;i++)rez*=baza;
      return rez;
} // putere
```

Observație

- while(printf("m(>=0): "),scanf("%d",&m),m<0);

- Echivalent cu:

```
printf("m(>=0): ");
scanf("%d",&m);
while(m<0){
        printf("m(>=0): ");
        scanf("%d",&m);
}
```

Programarea calculatoarelor
Limbajul C

Pointeri

Introducere

- *Definiție:*

Un pointer este o variabilă care păstrează *adresa unei date* (nu valoarea datei).

- Un pointer poate fi utilizat pentru referirea diferitelor date şi structuri de date. Schimbând adresa memorată în pointer, pot fi manipulate informații situate la diferite locații de memorie.
- Programul şi datele sale sunt păstrate în memoria RAM (*Random Access Memory*) a calculatorului.

Declarare şi operatori

- *Declarația* unei variabile pointer
  - se foloseşte operatorul de indirectare *:
  - tip_referit * var_pointer;

- *Inițializarea* cu adresa unei variabile:

  tip_referit var,* var_pointer;

  var_pointer=&var; // operator de referențiere

- *Valoarea* de la adresa indicată de pointer:
  - *var_pointer // este de tip_referit
  - *Operator de dereferențiere (indirectare)*

Operatori

- *Spațiul* ocupat de o variabilă pointer:
  - sizeof(var_pointer)
  - valoarea expresiei este 2 (în modelul small), oricare ar fi tip_referit
  - expresile de mai jos conduc la aceeaşi valoare 2:
    - sizeof(&var)
    - sizeof(tip_referit *)
- *Tipărirea* valorii unui pointer: se foloseşte prototipul %p, valoarea apărând sub forma a patru cifre hexa

Operatori

- *Adresa* unei variabile pointer este un pointer, la fel ca adresa unei variabile de orice alt tip:
  - &var_pointer

- *Observație:* Un pointer poate fi utilizat doar după inițializare, inițializare care se poate face:
  - prin atribuirea adresei unei variabile sau
  - prin alocare dinamică.

Exemplu

```
int *p, n=5, m;
p=&n;
m=*p;
m=*p+1;
```

```
int *a,**b, c=1, d;
a=&c;
b=&a;
d=**b;          //d=1
```

```
int *p;
float x=1.23, y;
p=&x;
y=*p; //valoare eronata pentru y
```

Operatii cu pointeri

- Adunarea (scăderea) unei constante la un pointer

```
tip*p;
p++;        ↔ p=p+sizeof(tip);
p--;        ↔ p=p-sizeof(tip);
p=p+c;      ↔ p=p+c*sizeof(tip);
p=p-c;      ↔ p=p-c*sizeof(tip);
```

- Scăderea a doi pointeri de acelaşi tip

- Compararea a doi pointeri (operații relaționale cu pointeri) = =, !=, <, >, <=, >=

Accesul la memorie

- Pointerul 0 este predefinit ca NULL şi semnifică faptul că nu indică nimic.
- void * înseamna un pointer de tip neprecizat
- Nu putem face operații aritmetice asupra acestor pointeri.
- Pointerii void ne permit păstrarea gradului de generalitate al unui program la maximum.

Pointeri şi vectori

- *Numele unui tablou* este un pointer constant spre primul său element.
- Expresiile de mai jos sunt echivalente:
  - nume_tablou
  - &nume_tablou
  - &nume_tablou[0]
- Şi de asemenea:
  - *nume_tablou
  - nume_tablou[0]

Pointeri şi vectori

- Declarații echivalente:
  - tip v[lim1][lim2]…[limn];
  - tip *…*v;
- Exemplu:
  - int v[10]; /*echivalent cu:*/
  - int *v;
    v=(int *)malloc(10*sizeof(int)); // cu alocare dinamică!!!
- Referire elemente:
  - v[i]
  - *(v+i)

Pointeri şi vectori

- Exemplu:

```
int i;
double v[100], x, *p;
p=&v[0];          ->corect, neelegant
p=v;
x=v[5];
x=*(v+5);
v++;              ->incorect
p++;              ->corect
```

Obs:

p[4]=2.5 ->corect sintactic, dar nu este alocată memorie pentru p!!!

Transmiterea vectorilor ca argumente funcțiilor

- void f(int *p, int n){
  …
}

- void f(int a[10] , int n){
  …
}

- void f(int a[] , int n) {
  …
}

Transmiterea vectorilor ca argumente funcțiilor

- Apel:

```
void main(void)
{
   int v[10], n;
   …
   f(v,n);
…
}
```

- Sau:

```
void main(void)
{
   int *v, n;
   …
   f(v,n);
…
}
```

Exemplu

```
#include <stdio.h>
#include <stdlib.h>
#define N 5
int citire1(int tab[]){
/*citeste elementele lui tab prin accesarea indexata a elementelor */
      int i=0;
      printf("Introduceti elementele tabloului:\n");
      while(scanf(“%d”,&tab[i]!=EOF) i++;
      return i;
}
void tiparire1(int *tab, int n){
/* tipareste elementele tabloului prin accesarea indexata a elementelor */
      int i;
      printf("Elementele tabloului:\n");
      for(i=0;i<n;i++)
          printf("%d ",tab[i]);
      printf(”\n”);
}
```

Exemplu

```
int citire2(int tab[]){
/* citeste elementele lui tab - accesarea fiecarui element se face printr-un pointer la
      el */
      int *pi;
      pi=tab;
      printf("Introduceti elementele tabloului:\n");
      while(scanf(“%d”,pi)!=EOF) pi++;
      return pi-tab;
}

void tiparire2 (int tab[], int n){
/* tipareste elementele lui tab prin accesare prîn pointeri */
        int *pi;
        printf("Elementele tabloului:\n");
        for (pi=tab; pi<tab+n; pi++)
                printf("%d ",*pi);
        printf(“\n”);
}
```

Exemplu

```
int main(){
    int tab1[N], tab2[N], n, m;
    n=citire1(tab1);
    tiparire1(tab1,n);
    m=citire2(tab2);
    tiparire2(tab2,m);
    return 0;
}
```

Transmiterea matricilor ca parametri funcțiilor

- Declarare funcții:

```
int min( int t[][NMAX], int m, int n){
  ...
}

int min( int *t [NMAX], int m, int n){
  ...
}

int min( int **t, int m, int n){
  ...
}
```

Transmiterea matricilor ca parametri funcțiilor

- Apel funcții:

```
int a[NMAX][NMAX], m, n;
…..
int mimim=min( a, m, n);
    ...
}
```

Exerciții propuse

1. Să se scrie următoarele funcții pentru o matrice pătratică de numere întregi:
   - citeşte o matrice
   - afişează elementele unei matrici
   - returnează suma elementelor unei matrici
   - însumează două matrici într-o a treia
   - înmulțeşte două matrici într-o a treia.

2. Program pentru citirea unui vector de întregi şi extragerea elementelor distincte într-un al doilea vector, care se va afisa. Se vor utiliza funcții. Ce funcții trebuie definite?

Exerciții propuse

3. Să se scrie o funcție de desenare a unei bare orizontale compuse din n caractere ch. Să se utilizeze această funcție pentru afişarea unei histograme ale cărei valori sunt memorate într-un vector (se va crea o funcție). Program de testare a funcțiilor scrise anterior.

Rezolvare 1. Tiparire elemente matrice

```
#include <stdio.h>
void tip ( int t[][10], int n, char *nume){
        int i,j;
       printf("Elementele matricii %s:\n",nume);
       for (i=0; i<n; i++){
         for (j=0; j<n; j++)
         printf("%d ",t[i][j]);
       putchar('\n');
       } // for i
}

int main(){
       int t1[40][10]={{4,5},{3,4}};
       tip (t1, 2, "t1");
       getchar();
       return 0;
}
```

Rezolvare 2 Extragere elemente distincte dintr-un vector

```
#define MAX 30
#include <stdio.h>
/* cauta pe x în vectorul a*/
int gasit(int v[], int n, int x){
      int m=0,i;
      for (i=0;i<n; i++)
            if (v[i]==x)  return i;
      return -1;
}

void main () {
       int a[MAX];            /* un vector de intregi*/
       int b[MAX];             /* aici se pun elementele distincte dîn a*/
       int n,m,i,j; /* n=dimensiune vector a, m=dimensiune vector b*/
```

Rezolvari 2 Extragere elemente distincte dintr-un vector

```
        printf("Numar de elemente vector n="); scanf("%d",&n);
         printf ("Introducere %d numere intregi:\n",n);
         /* citire vector*/
         for (i=0;i<n;i++) scanf("%d",&a[i]);

        m=0;
        for (i=0;i<n;i++)
              if(gasit(b,m,a[i])==-1) b[m++]=a[i];

        /* scrie vector b*/
         printf("Elementele distincte sunt:");
         for (j=0;j<m;j++)
           printf ("%5d",b[j]);
}
```

Rezolvare 3 Desenarea unei bare orizontale compuse din n caractere ch

```
#define M 20
#include <stdio.h>
 /* Desenarea unei bare orizontale din n caractere ch*/
void bar1 (int n, char ch) {
        int i;
        for (i=0; i<n; i++)
                putchar(ch);
        putchar('\n');
}

/*funcție de tip "void" cu argument de tip vector
 Desenare histograma pe baza unui vector de intregi*/
void hist (int a[], int n) {
      int i;
```

Rezolvare 3 Desenarea unei bare orizontale compuse din n caractere ch

```
for (i=0;i<n;i++)
        bar1(a[i],'*');
}

void main () {
        int a[M],n,i;
        scanf("%d",&n);
        for(i=0;i<n;i++)
                scanf("%d",&a[i]);
        hist (a,n);
}
```