Programarea calculatoarelor

FMI

Secția Calculatoare și tehnologia informației, anul I

Cursul 13 / 08.01.2024

Chestiuni organizatorice

- Examen scris:

30 ianuarie 2024, ora 9.00-11.00

(-se intră în sală cu cel mult 30 minute înainte, cel puțin 10 minute;

-se prezintă CI sau Carnetul de student cu fotografie;

-se respectă așezarea în bănci indicată de profesor;

-se folosește pix sau stilou cu cerneală albastră;

-copiatul sau tentativa de copiat => restanță)

- Regulament de evaluare și notare:

$$Nota = min\,(\mathbf{10}, \mathit{Examen} + \mathit{Laborator} + \mathit{Seminar})$$

6p **4p** **1p**

Chestiuni organizatorice

- Laborator= maximum 4 puncte
- Seminar = maximum 1 punct (prezență*0.1p +activitate)
- Examen = notă de la 0 la 10. Nu există punct din oficiu! Nota 10=6puncte

- NU intrați în examen (/restanță) dacă:
  - Nu aveți cel puțin jumătate din punctajul de la laborator (2p din 4p)
  - Nu ați participat la cel puțin 3 seminarii (0.3p din 1p)

- Aveți restanță dacă:
  - Nu intrați în examen
  - Nu vă prezentați la examen
  - Nu luați cel puțin nota 5 la examenul scris

- Restanța: se păstrează punctele obținute pe parcursul semestrului la seminar și laborator.

Programa cursului

❑ **Introducere**
- Algoritmi
- Limbaje de programare.

❑ **Fundamentele limbajului C**
- Introducere în limbajul C. Structura unui program C.
- Tipuri de date fundamentale. Variabile. Constante. Operatori. Expresii. Conversii.
- Tipuri derivate de date: pointeri, tablouri, șiruri de caractere, structuri, uniuni, câmpuri de biți, enumerări
- Instrucțiuni de control
- Directive de preprocesare. Macrodefiniții.
- Funcții de citire/scriere.
- Etapele realizării unui program C.

❑ **Fișiere text**
- Funcții specifice de manipulare.

❑ **Funcții (1)**
- Declarare și definire. Apel. Metode de trasmitere a paramerilor. Pointeri la funcții.

❑ **Tablouri și pointeri**
- Legătura dintre tablouri și pointeri
- Aritmetica pointerilor
- Alocarea dinamică a memoriei
- Clase de memorare

❑ **Șiruri de caractere**
- Funcții specifice de manipulare

❑ **Fișiere binare**
- Funcții specifice de manipulare

❑ **Structuri de date complexe și autoreferite**
- Definire și utilizare

❑ **Funcții (2)**
- Funcții cu număr variabil de argumente
- Preluarea argumentelor funcției main din linia de comandă
- Programare generică
- **Recursivitate**

Cuprinsul cursului de azi

1. **Funcții recursive**

2. ***Exemple de subiecte***

Recursivitate

- Procesul recursiv este procesul care, în timpul execuţiei, generează apariţia unor procese identice, aflate în legătură directă cu procesul ce le generează. Un proces poate fi descris printr-un subprogram.
- Într-o funcție recursivă recurența este realizată prin autoapelul funcției.
- În algoritmul recursiv sunt necesare două elemente:
  - formula de recurență (autoapelul)
  - o valoare inițială cunoscută (condiția de terminare a apelului).
- Fiecare autoapel al funcției duce spre soluția finală care corespunde valorii inițiale cunoscute.

Funcții recursive

Exemplul 1.

Factorial

```
#include <stdio.h>
 long int f (int n)
{
   long int f1=1;
   for (int i=1; i<=n;i++)  f1=f1*i;
   return f1;
}
int main()
{
   int n;
   scanf("%d", &n);

   if(!n) printf("0!=1\n");
   else
      printf("%d!=%ld\n", n, f(n));
   return 0;
}
```

```
#include<stdio.h>
long int f (int n)
{
  if (n==1) return 1;
  else return n*f(n-1);
}

int main()
{
   int n;
   scanf("%d", &n);

   if(!n) printf("0!=1\n");
   else
         printf("%d!=%ld\n", n, f(n));
  return 0;
}
```

Funcții recursive

- Pentru memorarea parametrilor, subprogramele folosesc zona de memorie numită stivă.
- La apelul subprogramului se vor depune în stivă, în ordine, parametrii transmişi și se rezervă spaţiu pentru variabilele locale.
- În cazul în care subprogramul se autoapelează pe un al doilea nivel, se depun din nou parametrii transmişi şi se rezervă un nou spaţiu pentru variabilele locale.
- În acest exemplu, calculele se efectuează la revenirea din apel. Valorile au fost reţinute în stivă, iar la revenire au fost folosite pentru înmulţire.

Reguli pentru construirea unei funcții recursive

- Corpul unei funcții recursive cuprinde două elemente :
  - **cazul general al soluției:** conține operații necesare pentru reducerea dimensiunii problemei, cea mai importantă operație fiind autoapelul.
  - **cazul de bază al soluției:** conține o operație care rezolvă un caz particular al problemei.
- este obligatoriu să existe o **condiție de oprire a repetiției** exprimată printr-o expresie logică. Aceasta expresie logică este folosită pentru a face trecerea de la cazul general la cazul de bază al soluției.
- O funcție recursivă trebuie să asigure condiția de consistență, modificarea parametrilor și sau a variabilelor locale, de la o activare la alta trebuie să asigure condiția de ieșire din recursivitate.

Funcții recursive. Observații

- în elaborarea algoritmilor recursivi se aplică raţionamentul că ce se întâmplă la un nivel, se întâmplă la orice alt nivel;
- pentru orice algoritm recursiv există unul iterativ care rezolvă aceeaşi problemă;
- nu întotdeauna alegerea unui algoritm recursiv reprezintă un avantaj.

Funcții recursive

Exemplul 2.

```
/*suma primelor n numere naturale nenule*/
#include<stdio.h>
 int s (int n)
{
        if (n==0) return 0;
        else return (n + s (n-1));
}

int main()
{
    int n;
    scanf("%d", &n);
    printf("suma este %d\n", s (n));
    return 0;
}
```

Funcții recursive

Exemplul 3.

```
/*calculul recursiv al sumei elementelor unui vector, de n<=10, de nr intregi.*/
#include <stdio.h>
int a[10], n;
int S (int n, int a[10])
{ if(n==0) return 0; else return(a[n]+S(n-1,a)); }
 int main()
{
    scanf("%d", &n);
    for (int i=1; i<=n; i++)
          scanf("%d", &a[i]);
    printf("%d", S(n,a));
    return 0;
}
```

Funcții recursive

Exemplul 4.

/*cmmdc-ului a două numere întregi fără semn. Algoritmul lui Euclid prin scaderi*/

```
#include <stdio.h>
unsigned int c (unsigned int a, unsigned int b)
{   while(a!=b)  { if(a>b) a=a-b; else b=b-a; } return a;}
 unsigned int c_r (unsigned int a, unsigned int b)
{   if(a==b) return a;
    else if(a>b) return c_r (a-b,b); else return c_r(a,b-a);
}
int main()
{
   unsigned int x,y;
   scanf("%u %u",&x, &y);
   if(!x || !y) printf("c = 1\n");
   else printf("c= %u\n",c_r(x,y)); //c(x,y)
   return 0;
}
```

Funcții recursive

Exemplul 5.

```
/*cmmdc-ului a n numere întregi fără semn. Algoritmul lui Euclid prin scaderi*/
#include <stdio.h>
unsigned int c_2(unsigned int a, unsigned int b)
{ if(a==b) return a;
  if(a>b) return c_2(a-b,b);   else return c_2(a,b-a);}

unsigned int c_n(unsigned int x[], int n)
{ if (n==2) return c_2(x[0],x[1]);
      else return c_2(c_n(x,n-1),x[n-1]);}

int main(){
  unsigned int x[20];    int n, i;    scanf("%d",&n);
  for(i=0;i<n;i++){ printf("elementul %d= ",i+1); scanf("%u",&x[i]);}
  if (n==1) printf("\n%u",x[0]);
     else printf("\n%u",c_n(x,n));
  return 0;
}
```

Funcții recursive

Exemplul 6.

```
/*sumei cifrelor unui numar natural*/
#include <stdio.h>
 int f(int n)
{ if (!n) return 0;
    else
  return n%10+f(n/10);}

int main()
{
        int n;
        scanf("%d", &n);

        printf("%d", f(n));

        return 0;
}
```

Funcții recursive

Exemplul 7.

/*transforma un numar natural n, din baza 10 în baza k (1<k<=10). Numarul se împarte la k, se retine restul, câtul se împarte la k, se retine restul... pâna când câtul este mai mic decât împartitorul. Rezultatul se obtine prin scrierea în ordine inversa a resturilor obtinute. Tiparirea restului se face dupa autoapel.*/

```
#include <stdio.h>
 void transform(int n,int b)
{ int r=n%b; if (n>=b) transform(n/b,b); printf("%d",r);}

int main()
{
        int n,b;
        scanf("%d",&n);
        scanf("%d",&b);
        transform(n,b);
        return 0;
}
```

Funcții recursive

Exemplul 8.

```
/*functia Manna-Pnueli*/

#include <stdio.h>
 int F(int x)
{
        if (x>=12) return x-1;
        return F(F(x+2));
}

int main()
{
        int x;
        printf("x="); scanf("%d",&x);
        printf("Valoarea functiei este: %d",F(x));
        return 0;
}
```

Funcții recursive

Exemplul 9.

```
/*șirul Fibonacci*/
 #include <stdio.h>
 int U (int n)
{
   if (!n) return 0;
   else if (n==1) return 1;
   else return U(n-1)+U(n-2);
}
 int main()
{
   int n;
   scanf("%d",&n);

   printf(„pt n: %d",U(n));

return 0;
}
```

```
 #include <stdio.h>
  int main()
{
   int n,U0=0,U1=1,U2;
   printf("n="); scanf("%d",&n);
   if(!n) printf("%d",U0);
   else
     if (n==1) printf("%d",U1);
          else
          {
          for (int i=2;i<=n;i++)
             { U2=U0+U1; U0=U1; U1=U2; }
          printf("%d",U2);
          }

   return 0;
}
```

Funcții recursive

Exemplul 10.

Se considera urmatoarele declaratii şi conventii:

typedef int vector[20];

x este un vector

n este lungimea lui x (n>=1)

Se cere sa se scrie functii recursive pentru a determina, pentru un vector x de lungime n, urmatoarele:

a. citirea componentelor sirului
b. afisarea elementelor din sir
c. suma componentelor
d. produsul componentelor
e. numarul componentelor negative
f. produsul componentelor pozitive
g. media aritmetica a elementelor

Funcții recursive

Exemplul 10.

```
#include <stdio.h>
/* un tip propriu definit pentru memorarea sirurilor de elemente intregi,
elemente intregi, cu o dimensiune maxima de 20 de componente */
typedef int vector[20];
void citire(vector x, int n) //functia de citire; n este dimensiunea lui x
{   //citim ultimul element din sir
    printf("\telementul %d: ",n); scanf("%d",&x[n-1]);
    if(n>=2) citire(x, n-1); //apelul recursiv al functiei
}
void afisare(vector x, int n) //functia de afisare
{ //afisam ultimul element
   printf("%d ",x[n-1]);
   if(n>=2)  afisare(x, n-1); //apelul recursiv al functiei
}
```

Funcții recursive

Exemplul 10.

```
 //adunarea componentelor unui sir
int suma(vector x, int n) //n in acest caz este indice element din sir
{
   if(n == -1) return 0; /* nu mai sunt elemente, pozitia n=-1  nefiind in sir*/
   else return x[n] + suma(x, n-1);
}

//produsul componentelor
int produs(vector x, int n)
{ if(n == -1) return 1; else return x[n]*produs(x, n-1); }

//numarul de componente negative
int numar_negative(vector x, int n)
{/*ne pozitionam pe primul element din sir, verificam daca acesta este<0*/
   if(n==0) return (x[n]<0); /*expresia va returna 1 in cazul True, 0 altfel*/
   else return (x[n]<0)+numar_negative(x,n-1);
}
```

Funcții recursive

Exemplul 10.

```
 //produsul componentelor pozitive
int produs_pozitive(vector x, int n)
{ if(n==0) return (x[n]>0? x[n]:1);
   /* Daca expresia este True, se va lua in calcul x[n], altfel, valoarea 1 */
   else
   return (x[n]>0? x[n]:1)*produs_pozitive(x, n-1);
}

//media aritmetica a componentelor sirului
float media(vector x, int m, int n)
//m=indicele elementelor,
//n=dimensiunea reala a sirului
{  return (float)x[m]/n + ((m!=0)? media(x, m-1, n):0);
    // expresia (float) => conversie explicita a rezultatului spre un tip real;
    //x[m] este un element (in prima faza, acesta fiind ultimul element din sir)
}
```

Funcții recursive

Exemplul 10.

```
int main()
{  vector x; int n; //n=dimensiunea lui x(numarul de componente citite)
   // citirea sirului
   printf("Dati numarul de elemente: "); scanf("%d",&n);
   printf("Introduceti elementele sirului:\n"); citire(x,n);

   //afisarea sirului
   printf("Sirul de elemente este: "); afisare(x,n);

   printf("\nSuma elementelor: %d", suma(x, n-1));
   /* n-1=indicele ultimului element din sir */
   printf("\nProdusul elementelor:\n%d", produs(x,n-1));
   printf("Numarul elementelor negative: %d",numar_negative(x,n-1));
   printf(" Produsul elementelor pozitive: %d", produs_pozitive(x,n-1));
   printf("\nMedia componentelor din sir: %.2f", media(x,n-1,n));
   return 0;
}
```

Funcții recursive. Probleme propuse

- Determinați produsul numerelor pare mai mici sau egale cu n.
- Determinați media aritmetica a numerelor divizibile cu 3 mai mici sau egale cu n.
- Se citesc n numere. Determinati:
  - produsul numerelor cu 2 cifre
  - cate numere impare s-au citit
  - suma numerelor care nu sunt multiplii numarului 5
- Pentru un numar n dat. Determinati:
  - produsul cifrelor mai mari decat 2
  - cate cifre nule contine numarul n
  - cifra maxima din n
  - Inversul (/oglinditul/ rasturnatul) lui n

Cuprinsul cursului de azi

1. Funcții recursive

2. *Exemple de subiecte*

Exemple

1. Reprezentati pe 32 de biti numerele intregi 1234 si – 5678.
2. Analizati programul urmator. Daca este corect spuneti ce se va afisa, in caz contrar explicati de unde provine eroarea.

```
#include <stdio.h>
int v[10];
int main() {
int *pv, *pv1, i;
pv = v;
*(pv + 1) = 10; *(pv + 4) =4; *(pv - 3) = 1;
pv1 = &(*(pv++)); *(pv1+5) =11; *(pv1+8) = v[2] - 4;
for(i=0; i<9; i++) printf("%d ", i+v[i]);
}
```

Exemple

3. Ce valoare va avea variabila x in urma executarii secventei de cod de mai jos? Justificați.

```
int n = 100, x;
x = n | (40 & 10) && n>>(4<<2) + n | (10 ^ 8);
```

4. Functia sumaCifre de mai jos este scrisa pentru a returna suma cifrelor unui numar natural n. Functia este scrisa gresit.

```
int sumaCifre (int a) { if (a<=10) return a; return sumaCifre(a/10);}
```

a) dați exemplu de o intrare pentru care functia returneaza corect rezultatul. Justificati.

b) dați exemplu de o intrare pentru care functia returneaza incorect rezultatul. Justificati.

c) scrieți varianta corecta a functiei sumaCifre (puteți porni de la actuala varianta sau puteți rescrie în totalitate functia).

Exemple

5. Fișierul text $note.txt$ conține pe prima linie un număr natural $n \geq 1$, iar pe fiecare din următoarele n linii informații despre nota obținută de un student la un examen, astfel:

nume student, prenume student, grupa, disciplina, nota obținută

Definiți o structură $Student$ care să permită stocarea informațiilor referitoare la un student și, eventual, a altor informații pe care le considerați necesare pentru rezolvarea cerințelor de mai jos, în mod optim din punct de vedere al memoriei utilizate.

a) Afișați numărul studenților care au promovat examenul la o disciplină d, citită de la tastatură, precum și numărul studenților care au susținut examen la disciplina respectivă.

b) Folosind un tablou unidimensional cu elemente de tip $Student$ și funcția qsort din biblioteca $\mathrm{stdlib.h}$, calculați media fiecărui student și afișați studenții în ordinea descrescătoare a mediilor, iar în cazul unor medii egale în ordine alfabetică.

c) Scrieți într-un fișier binar informațiile despre studenții care au mediile maxime. Numele fișierului binar se va citi de la tastatură.

Exemple

Subiectul I – 2 puncte

a) Scrieți o funcție care să returneze un tablou unidimensional alocat dinamic format din valorile strict pozitive aflate într-un tablou unidimensional v având n elemente de tip întreg, precum și numărul acestora. (1 punct)

b) Considerăm definite următoarele 3 funcții:

- $void\ multiply(int\ *\ v, int\ n, int\ x)$ – înmulțește cu x fiecare element al tabloului unidimensional v format din n numere întregi;
- $void\ sort(int\ *\ v, int\ n)$ – sortează crescător tabloul unidimensional v format din n numere întregi;
- $void\ display(int\ *\ v, int\ n)$ – afișează pe ecran tabloul unidimensional v format din n numere întregi.

Folosind doar apeluri utile ale funcției definite la punctul a) și ale celor 3 funcții precizate mai sus, scrieți o funcție care să afișează în ordine descrescătoare numerele strict negative dintr-un tabloul unidimensional v format din n numere întregi. (1 punct)

Observație: Nu este permisă utilizarea unor variabile globale!

Exemple

Subiectul II - 2 puncte

Scrieți o funcție care primește ca parametru un număr natural nenul $n \geq 2$ și returnează un tablou bidimensional triunghiular alocat dinamic și construit după următoarele reguli:

- prima coloană conține numerele de la 1 la n, în ordine descrescătoare;
- ultima linie conține numerele de la 1 la n, în ordine crescătoare;
- orice alt element este egal cu suma vecinilor săi de la vest și sud.

Exemplu: Pentru $n = 4$ funcția trebuie să returneze următorul tablou:

```
4
3 7
2 4 7
1 2 3 4
```

Exemple

Subiectul III – 2 puncte

Fișierul text *cuvinte.in* conține pe prima linie un cuvânt w format din $n \geq 1$ litere mici ale alfabetului englez, iar pe următoarele linii un text, format tot doar din litere mici ale alfabetului englez, în care cuvintele sunt despărțite prin spații și semnele de punctuație uzuale. Realizați un program care să scrie în fișierul text *cuvinte.out* toate cuvintele din fișierul *cuvinte.in* care au ca prefix cuvântul w sau mesajul "*Imposibil*" dacă în fișierul de intrare nu există nici un cuvânt cu proprietatea cerută.

Observație: Se vor utiliza funcții pentru manipularea șirurilor de caractere din biblioteca string.h!

Exemple

Subiectul IV - 3 puncte (**Observație:** Nu este permisă utilizarea unor variabile globale!)

a) Definiți o structură *Student* care să permită memorarea numelui, grupei, numărului total de credite obținute și situației școlare a unui student (Promovat/Nepromovat). Scrieți o funcție care să determine situațiile școlare a n studenți ale căror date sunt memorate într-un tablou unidimensional t cu elemente de tip *Student*. Un student este considerat promovat dacă a obținut un număr de credite cel puțin egal cu un număr natural nenul c. (1 punct)

b) Scrieţi o funcţie care să sorteze elementele unui tablou unidimensional t format din n elemente de tip *Student* în ordinea crescătoare a grupelor, iar în cadrul fiecărei grupe studenții se vor ordona descrescător în ordinea numărului total de credite obținute. (1 punct)

c) Realizați o funcție care să scrie într-un fișier binar informațiile despre studenții promovați dintr-o grupă. Funcția va avea ca parametrii un tablou unidimensional t cu n elemente de tip *Student* și un număr natural nenul g reprezentând numărul unei grupe. (1 punct)

Observație: Pentru sortarea tabloului, se va folosi funcția `qsort` din biblioteca `stdlib.h`!

Cursul 13/ 08.01.2024

1. Funcții recursive

2. Exemple de subiecte

Cursul 14/ 15.01.2024

Recapitulare

Examen/ 30.01.2024

Restanță ? (19-25.02.2024)

Cursul **Programarea calculatoarelor II**/ an I, CTI, sem. al II-lea

Capitole:

- Complexitatea computațională a algoritmilor
  Recursivitate
- Metoda Greedy
- Metoda backtracking
- Metoda Divide et Impera
- Metoda programării dinamice
- Aplicații ale metodelor de parcurgere a unui graf

Prof.: conf. univ. dr. Radu Boriga