Laborator 8 - Tehnici de programare

8.1 Recursivitate

Un algoritm recursiv este caracterizat de:

- apeluri recursive prin care algoritmul se apelează pe el însuşi cel puţin o dată; valorile parametrilor trebuie să conveargă în urma apelurilor recursive spre valorile parametrilor dintr-o condiţie de oprire;
- condiţii de oprire care specifică situaţiile în care rezultatul poate fi calculat direct, fără a fi necesar un apel recursiv.

Mai jos este descris un exemplu de algoritm recursiv generic.

```
algoritm_ recursiv(n)
    if n == n0 then
        R0
    else
        algoritm_recursiv (h(n))
    end if
end algoritm_recursiv
```

Funcţia h are proprietatea că există $k \geq 0$ astfel încât $h^{(k)}(n) = h \circ h \circ \ldots h(n) = n_0$

Integrati in programe C complete, independente, fiecare din exemplele de mai jos, utilizand functiile recursive corespunzator.

Exemplul 8.1. Funcţia de mai jos afişează toate numerele mai mici decât o valoare dată în ordine descrescătoare.

```
void afisare( int n){
    if (n == 0 )
        printf("%d ",n);
    else {
        printf("%d ",n);
        afisare(n - 1 );
    }
}
```

Putem rescrie funcţia astfel încât condiţia de oprire să fie înlocuită cu o condiţie de continuare.

```
void afisare( int n){
    if (n >= 0 )
    {
        printf("%d ",n);
        afisare(n - 1 );
    }
}
```

Dacă dorim afişarea numerelor în ordine crescătoare, vom face următoarea modificare.

```
void afisare( int n){
    if (n >= 0 )
    {
        afisare(n - 1 );
        printf("%d ",n);
    }
}
```

Exemplul 8.2. Să se scrie o funcţie recursivă care să afişeze divizorii unui număr citit de la tastatură.

```
void divizori(int n,int d){
if( d<=n/2)
{
    if (n%d==0)
        printf("%d ",d);
    divizori(n,d+1);

}
}
```

Exemplul 8.3. Scrieţi o funcţie recursivă care calculează *inversul* (oglinditul) unui număr n citit de la tastatură.

```
int invers(int n, int inv){
 if ( n == 0 )
    return inv;
 else
    return invers(n/10,inv*10+n%10);
}
```

Exemplul 8.4. Să se calculeze suma $\sum_{k=1}^{n} \frac{k}{(k+1)*(k+2)}$

```
// suma n/((n+1)*(n+2))
float expresie(int k)
{
return (float)k/((k+1)*(k+2));
}
float suma(int k)
{
if (k==1)
    return (float)1/(2*3);
else
    return expresie(k)+suma(k-1);
}
```

8.5. Scrieti un program recursiv care calculeaza 3^n-2^n.

8.6. Scrieti un program recursiv care calculeaza factorialul unui numar natural n.

8.2 Divide et Impera

Această tehnică de programare se bazează pe următoarea strategie:

- divide: problema inițială este împărțită în subprobleme; fiecare subproblemă este rezolvată recursiv în aceeaşi manieră; atunci când dimensiunea problemei curente se reduce la o valoare suficient de mică, atunci aceasta este rezolvată direct;
- impera: rezultatele obținute prin rezolvarea subproblemelor sunt combinate în aşa fel încât să se calculeze soluția problemei de dimensiune mai mare; astfel că, la revenirea din recursivitate, problema inițială este rezolvată.

Studiu de caz. Pentru a calcula suma elementelor unui şir format din n elemente, se poate aplica divide et impera astfel: suma tuturor elementelor se poate scrie ca suma dintre suma elementelor din prima jumătate şi suma elementelor din a doua jumătate a şirului; fiecare subşir este impărțit în mod recursiv în două subşiruri; atunci când un subşir este suficient de mic (de exemplu, când este format dintr-un singur element) rezultatul poate fi calculat direct.

Fişierul $x_1, x_2, x_3, \ldots, x_n$. Se poate construi o funcție suma(p; u) = $x_p + x_{p+1} + \ldots + x_u$ după principiul divide et impera astfel:

```
int suma_dei(int v[100], int st, int dr)
{
    int m;
     if(st==dr)
        return v[st];
    else
        {
         m= (st+dr)/2;
         return (suma_dei(v,st, m)+ suma_dei(v,m+1, dr));
         }
}
```

Exemplul 8.5. Scrieți un algoritm recursiv care să implementeze căutarea binară a unei valori în elementele unui vector.

```
int caut_binar_di (int v[], int x,int s, int d)
{

    if(s>d)
        return -1;
    else
        {
            if (x==v[(s+d)/2])
                return (s+d)/2;
            if (x<v[(s+d)/2])
                return caut_binar_di(v,x,s,(s+d)/2-1);
            else
                return caut_binar_di(v,x,(s+d)/2+1,d);
        }
}
```

Get the mobile app