Divide et Impera

Alexandru Nicoi - Andrei Duță

24 Martie 2021

1 Recursivitate

- Pentru a folosi tehnica Divide et Impera, trebuie sa cunoastem bine principiul recursivitatii. O problema se poate rezolva reutilizand succesiv aceleasi instructiuni (spre exemplu determinarea factorialului unui numar, determinarea pozitiei K in sirul Fibonacci).Deci putem defini recursivitatea ca proprietarea prin care se reapeleaza functia cu scopul rezolvarii cerintei date.

```
    /// Sa se scrie o functie C++ recursiva care sa determine cifra
        maxima si cifra minima a unui numar natural transmis ca
        parametru. Functia va intoarce rezultatele prin
        intermediul unor parametri de iesire.

void cifmaxmin(long n , int &maxi , int &mini)
{
    if(n < 10){
        maxi = n;
        mini = n;}
    else{
        cifmaxmin(n/10,maxi,mini);
        if(n % 10 > maxi)
            maxi = n % 10;
        if(n % 10 < mini)
            mini = n % 10;
    }
}
    /// sa se scrie o functie C++ care sa returneze rezultatul
        functiei Manna-Pnueli, definita mai jos:
    ///f(x) = {    x-1   ;   pentru x >= 12
    ///           f(f(x + 2)) ; pentru x < 12
    ///       }

int mp(int x)
{
    if(x >= 12)
        return x - 1;
```

```
    else return mp(mp(x + 2));
}

    /// suma elementelor dintr-un vector folosind recursivitatea

int sum(int v[], int n)
{
    if(n - 1 == 0)
        return v[n-1];
    else return v[n-1] + sum(v, n-1);
}
```

---

2 Tehnica Divide et Impera

- Divide et Impera este o metodă de programare bazată pe un principiu simplu:
  - problema dată se descompune în două (sau mai multe) subprobleme (de același tip ca problema inițială, dar de dimensiuni mai mici);
  - se rezolvă independent fiecare subproblemă;
  - se combină rezultatele obținute pentru subprobleme, obținând rezultatul problemei inițiale.

  Subproblemele trebuie să fie de același tip cu problema inițială, ele urmând a fi rezolvate prin aceeași tehnică.

  Subproblemele în care se descompun problema dată trebuie să fie:

  - de același tip cu problema dată;
  - de dimensiuni mai mici (mai "ușoare");
  - independente (să nu se suprapună, prelucrează seturi de date distincte).

  SURSA: PBINFO (https://www.pbinfo.ro/articole/7651/divide-et-impera), Autor Candale Silviu

---

```
    /// Se considera un sir cu n elemente, numere naturale.
        Folosind metoda Divide et Impera, determinati suma
        elementelor pare din acest sir.

int sum(int p , int u)
{
    int m , nr1 , nr2;
    if(p == u)
        return v[p];
    else{
        m = (p + u) / 2;
```

```
        nr1 = sum(p,m);
        nr2 = sum(m+1,u);
        if(nr1 % 2 == 0 && nr2 % 2 == 0)
            return nr1 + nr2;
        else if(nr1 % 2 == 0 && nr2 % 2 != 0)
            return nr1;
        else if(nr2 % 2 == 0)
            return nr2;
    }
}
    /// Se considera un sir cu n elemente, numere naturale.
        Folosind metoda Divide et Impera, determinati cate
        elemente impare sunt in acest sir.

int divide(int p , int u)
{
    int m , nr1 , nr2;
    if(p == u){
    if(v[p] % 2 != 0)
        return 1;
    return 0;
    }
    else{
        m = (p + u) / 2;
        nr1 = divide(p,m);
        nr2 = divide(m+1,u);
        return nr1 + nr2;}
}
    /// CMMDC al unui vector
int CMMDC(int a , int b)
{
    if(!b)
        return a;
    else return CMMDC(b , a % b);
}

int divide(int p , int u)
{
    int m , nr1 , nr2;
    if(p == u)
        return v[p];
    else{
        m = (p+u)/2;
        nr1 = divide(p,m);
        nr2 = divide(m+1,u);
        return CMMDC(nr1 , nr2);
    }
}
```