SDA - Recursivitate

Daniel Chiș - 2021, UPB, ACS, An I, Seria AC

29,30,1 Mai evaluare laborator

Recursivitate

Concept

Unele limbaje de programare permit ca unele module sau funcții să se auto-apeleze, acest proces fiind numit recursivitate. În recursivitate, o funcție *f* se poate apela pe ea direct sau poate apela o funcție *g* care la rândul ei apelează funcția originală *f*. Funcția *f* se numește funcție recursivă.

```c
//functie care se auto-apeleaza
int function(int value) {
   if(value < 1)
      return;
   function(value - 1);

   printf("%d ",value);
}
```

```c
//functie care este apelata de o alta
//care la randul ei o apeleaza pe prima
int function1(int value1) {
   if(value1 < 1)
      return;
   function2(value1 - 1);
   printf("%d ",value1);
}
int function2(int value2) {
   function1(value2);
}
```

Recursivitate

Proprietăți

Pentru a nu intra într-o buclă infinită o funcție recursivă trebuie să aibă două atribute:

1. Criteriu de bază: o condiție care atunci când este întâlnită funcția se oprește din a se auto-apela
2. Apropiere progresivă: apelările recursive trebuie să progreseze astfel încât să ne apropiem de condiția de bază

Implementare

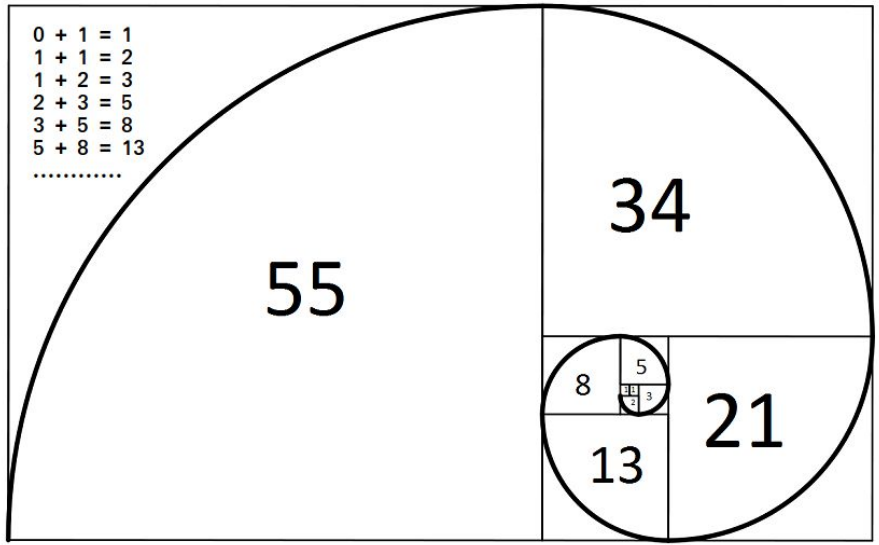
De fiecare dată când apelantul apelează o funcție, acesta predă și controlul la runtime. Astfel, execuția primei funcții se oprește până ce cea apelată este rezolvată și primește înapoi parametrii de execuție. Astfel o să avem mereu un “activation record” în care o să păstrăm parametrii primei funcții.

Serii fibonacci

Serii fibonacci

Seriile fibonacci generează următorul termen din secvența lor prin adunarea celor doi termeni anteriori. Seriile fibonacci încep mereu cu numere F0 și F1 care au fie valorile 0,1 sau 1,1.

$$Fn = Fn\text{-}1 + Fn\text{-}2$$



Fibonacci - Algoritmi

Algoritm iterativ

```
1  Fibonacci(n)
2     declare f0, f1, fib, loop
3
4     set f0 to 0
5     set f1 to 1
6
7     display f0, f1
8
9     for loop ← 1 to n
10
11       fib ← f0 + f1
12       f0 ← f1
13       f1 ← fib
14
15       display fib
16    end for
17
18 end
```

Algoritm recursiv

```
1 Fibonacci(n)
2 Begin
3   if n <= 1 then
4         Return n
5   else
6         Return Call Fibonacci(n-1) + Call Fibonacci(n-2)
7   endif
8 End
```

Turnul din Hanoi

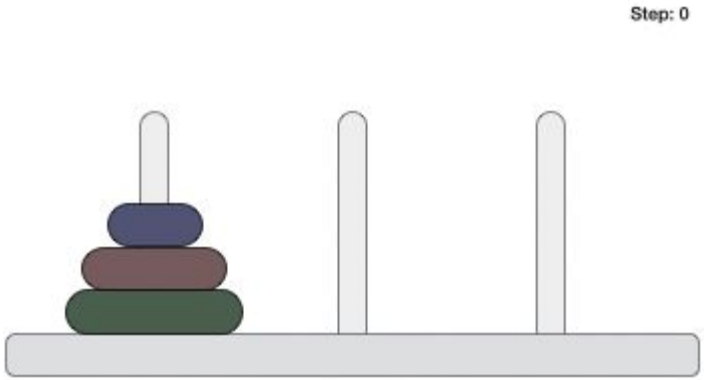
Turnul din Hanoi

Turnul din Hanoi reprezintă o problemă matematică care constă în trei turnuri și mai multe discuri aranjate pe unul din turnuri.

Reguli:

- Poți muta un singur disc între turnuri
- Poți muta doar discul de deasupra turnului
- Nu poți avea un disc mai mare deasupra unui mai mic

Turnul din Hanoi poate fi rezolvat în 2^n-1 pași (n reprezântând numătul de discuri).



Algoritm

Cele trei turnuri se denumesc: sursa, destinație și aux.

Pas 1 - mut n-1 discuri de la sursa la aux

Pas 2 - mut discul n la destinatie

Pas 3 - mut n-1 discuri de la aux la dest

```
1  Hanoi(disk, source, dest, aux)
2
3     IF disk == 1, THEN
4        move disk from source to dest
5     ELSE
6        Hanoi(disk - 1, source, aux, dest)
7        move disk from source to dest
8        Hanoi(disk - 1, aux, dest, source)
9     END IF
10
11 END
```

Programare dinamică

Programare dinamică

Programarea dinamică este similară ca și concept cu cel de divide and conquer, astfel problema principală se sparge în subprobleme cât mai atomic posibil. Spre deosebire de divide and conquer, subproblemele nu sunt rezolvate independent, rezultatele sunt ținute minte li folosite în rezolvarea unor subprobleme similare.

Algoritmii asociați programării dinamice sunt folosiți pentru optimizări. Spre deosebire de algoritmii greedy care caută optimizarea locală, aceștia caută optimizarea generală. Algoritmii dinamici se folosesc de memorea unui rezultat a unei subprobleme deja rezolvate.

Exemple: serii fibonacci, problema rucsacului, turnul din Hanoi.

Fibonacci programare dinamică

```c
//Fibonacci Series using Dynamic Programming
#include<stdio.h>

int fib(int n)
{
  /* Declare an array to store Fibonacci numbers. */
  int f[n+2];   // 1 extra to handle case, n = 0
  int i;

  /* 0th and 1st number of the series are 0 and 1*/
  f[0] = 0;
  f[1] = 1;

  for (i = 2; i <= n; i++)
  {
      /* Add the previous 2 numbers in the series
         and store it */
      f[i] = f[i-1] + f[i-2];
  }

  return f[n];
}

int main ()
{
  int n = 9;
  printf("%d", fib(n));
  getchar();
  return 0;
}
```

Exerciții

Exerciții

1. Realizați algoritmul fibonacci recursiv și iterativ (n=10). 2p
2. Realizați algortimul turnului din hanoi recursiv (4 discuri). 3p
3. Realizați un program de afișare a unui vector folosind un algoritm recursiv. 2p
4. Realizați un exemplu de recursivitate care poate fi explicat unui copil de 5 ani. 3p