**Ministerul Educaţiei și Cercetării al Republicii Moldova**

**Universitatea Tehnică a Moldovei**

**Facultatea Calculatoare, Informatică și Microelectronică**

**RAPORT**

Lucrarea de laborator nr.1

*la Analiza și Proiectarea Algoritmilor*

A efectuat:

st. gr. TI-216 Roșca Dorin

A verificat: Lisnic Inga

Chişinău - 2022

**Lucrare de laborator nr 1**

**Tema:** Analiza algoritmilor

**Scopul lucrării**:

1. Analiza empirică a algoritmilor

2. Analiza teoretică a algoritmilor

3. Determinarea complexitații temporale și asimptotice a algoritmilor

**Sarcina**:

1. De efectuat analiza empirică a algoritmilor propuși

2. De determinat relația ce reprezintă complexitatea temporală pentru acești algoritmi

3. De determinat complexitatea asimptotică a algoritmilor

**Rezumat succint la tema lucrării de laborator:**

Şirul lui Fibonacci este definit prin următoarea recurenţa:



Acest celebru şir a fost descoperit în 1202 de către Leonardo Pisano (Leonardo din Pisa), cunoscut sub numele de Leonardo Fibonacci. Cel de-al *n-*lea termen al şirului se poate obtine direct din definiţie:

**1. function** *fib*1(*n*)  
     **if** *n* < 2   **then   return** *n* **else   return** *fib*1(*n*-1) + *fib*1(*n*-2)

Această metodă este foarte ineficienta, deoarece recalculează de mai multe ori aceleaşi valori iar folosind functia recursiva numarul de iteratii poate ajunge la un numar foarte mare foarte repede. Urmează o altă metodă, mai performantă, care rezolvă aceeaşi problemă.

**2. function** *fib*2(*n*)  
     *i*  1; *j*  0  
     **for** *k*  1 **to** *n* **do** *j*  *i* + *j*                                 *i*  *j* - *i* **return** *j*

Mai există un algoritm :

**3. function** *fib*3(*n*)  
     *i*  1; *j*  0; *k*  0; *h*  1,t  
     **while** *n* > 0 **do**          **if** *n* este impar **then** *t*  *jh  
                                           j*  *ih*+*jk*+*t  
                                           i*  *ik*+*t*          *t*   *h*2  
          *h*  2*kh*+*t*          *k*  *k*2+*t*          *n*  *n* **div** 2  
     **return** *j*

**Codul programului in C**

**#include** <iostream>

int count1 = 0;

int count2 = 0;

int count3 = 0;

long long **fib1**(long long n)//*Functia recursiva*

{

**if**(n **<** 2)

    {

        count1++;

**return** n;

    }

**else**

    {

    count1++;

**return** **fib1**(n-1)+**fib1**(n-2);

    }

}

long long  **fib2**(long long numar)//*Functia for*

{

    int i=1,j=0;

**for**(int k=1;k **<=**numar;k++)

    {

        count2++;

        j=i+j;

        i=j-i;

    }

**return** j;

}

long long  **fib3**(long long  numar)//*Functia While*

{

    int i = 1, j = 0, k = 0, h = 1, t;

**while** (numar **>** 0)

    {

**if** (numar % 2 **!=** 0)

        {

            t = j \* h;

            j = i \* h + j \* k + t;

            i = i \* k + t;

        }

        t = h \* h;

        h = 2 \* k \* h + t;

        k = k \* k + t;

        numar = numar / 2;

        count3++;

    }

**return** j;

}

int **main**()

{

    long long numar;

    std :: cout **<<** "Introduceti numarul din list Fibonaccu: \n";

    std :: cin **>>** numar;

    std::cout **<<** "Metoda 1-Functia Recursiva: " **<<** std::**endl**;

    std:: cout **<<** **fib1**(numar) **<<** std::**endl**;

    std::cout **<<** "Numarul de Iteratii: " **<<** std::**endl**;

    std::cout **<<** count1 **<<** std::**endl**;

    std::cout **<<** "Metoda 2-Functia For: " **<<** std::**endl**;

    std::cout **<<** **fib2**(numar) **<<** std::**endl**;

    std::cout **<<** "Numarul de Iteratii: " **<<** std::**endl**;

    std::cout **<<** count2 **<<** std::**endl**;

    std::cout **<<** "Metoda 3-Functia While: " **<<** std::**endl**;

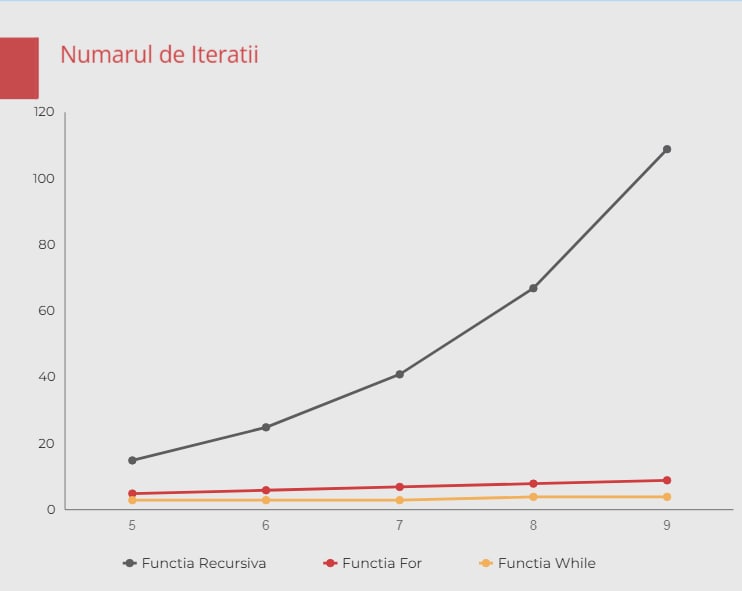
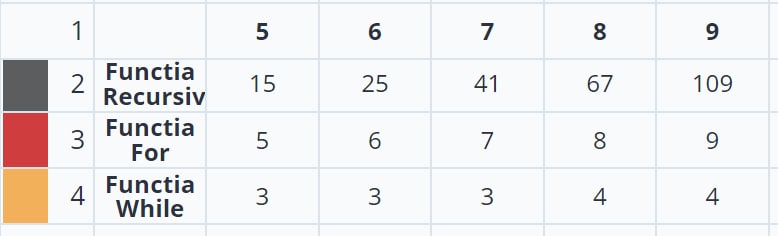
    std::cout **<<** **fib3**(numar) **<<** std::**endl**;

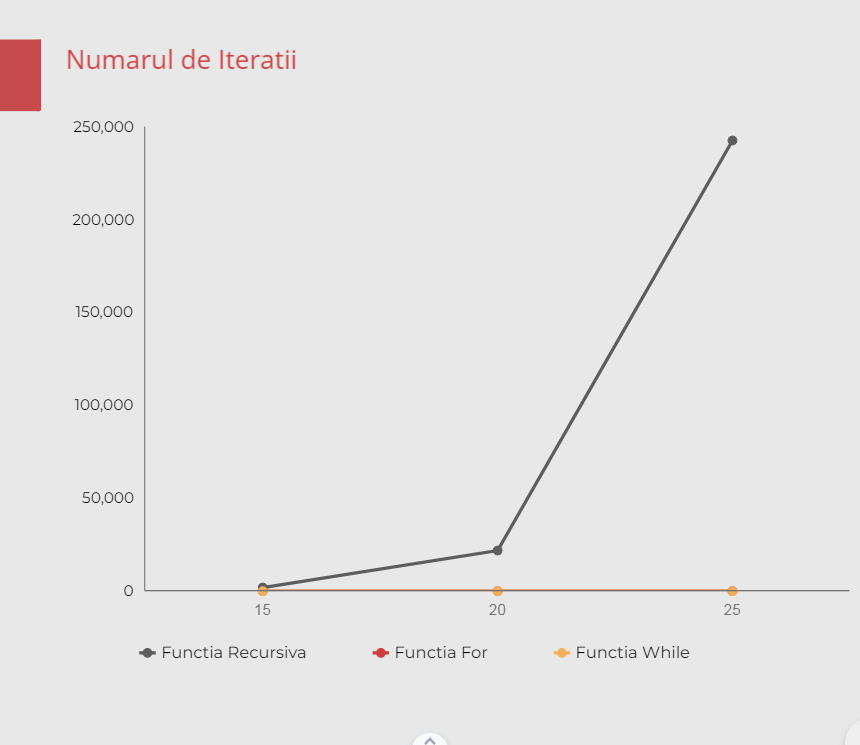
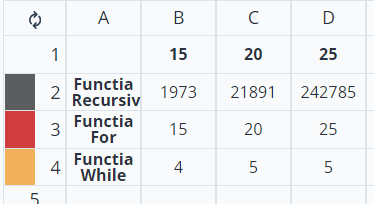
    std::cout **<<** "Numarul de Iteratii: " **<<** std::**endl**;

    std::cout **<<** count3 **<<** std::**endl**;

}

**Analiza algoritmilor**





**Concluzie**

Scopul lucrării propuse este de a analiza empiric și teoretic algoritmul pentru determinarea celui de-al n-lea număr Fibonacci, care a fost realizat cu succes. Sunt folosiți trei algoritmi diferiți: recursiunea, iterația și formula pentru a determina complexitățile și asimptotele lor de timp și pentru a evidenția cel mai eficient algoritm, care ne va arăta rezultatul dorit folosind cât mai puține iterații. Compararea algoritmilor se face cu ajutorul unor tabele în care colectăm date empirice asupra iterațiilor primite în timpul lucrului și grafice construite în Excel care ne arată diferențele clare dintre complexitatea algoritmică. De aici observăm că metoda recursivă este cea mai puțin eficientă deoarece recalculează aceeași valoare de mai multe ori, în timp ce metoda formulei ne oferă rezultate mai favorabile. Algoritmul iterativ este, de asemenea, un algoritm suficient de bun cu complexitate liniară. Prin urmare, pentru a determina al n-lea număr Fibonacci, se folosesc metode iterative și formule.