**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 Vlașițchi Ștefan

A verificat:

asist. univ. Andrievschi-Bagrin Veronica

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. 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  
     **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>

#include <math.h>

using namespace std;

unsigned long long i,j,k,h,t;

unsigned long long it1=0,it2=0,it3=0;

 unsigned long long fib1(unsigned long long n)

{

    if (n<2)

    {

        return n;

    }

    else

    {

        it1 = it1 + 4;

        return(fib1(n-1)+fib1(n-2));

    }

}

 unsigned long long fib2(unsigned long long n)

{

    long a=0;

    long b=1;

    long fib2    ;

   if (n == 0) {

        fib2 = 0;

    };

    if (n == 1) {

        fib2 = 1;

        it2 = it2 + 3;

    };

    if (n >= 2) {

        it2 = it2 + 1;

        for (int i = 2; i <= n; i++) {

            it2 = it2 + 7;

            fib2 = a + b;

            a = b;

            b = fib2;

        }

    };

    return a;

}

 unsigned long long fib3(unsigned long long n)

{

    i=1;

    j=0;

    k=0;

    h=1;

    it3=5;

    while(n>0)

    {

        if (n%2!=0)

        {

            t=j\*h;

            j=i\*h+j\*k+t;

            i=i\*k+t;

            it3 = it3 + 12;

        }

        t=h\*h;

        h=2\*k\*h+t;

        k=k\*k+t;

        n=n/2;

        it3 = it3 + 11;

    }

    return j;

}

int main()

{

    long a,b,c;

    int x=0,y=0;

    cin>>x;

    y=x+1;

    a=fib1(x);

    b=fib2(y);

    c=fib3(x);

    //cout<<a<<" "<<it1<<endl<<b<<" "<<it2<<endl<<c<<" "<<it3;

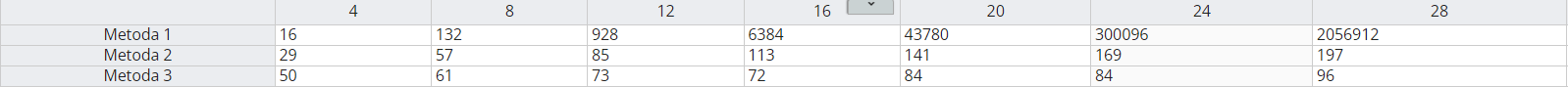
    cout<<"Prin recursie nr. fib "<<a<<" are "<<it1<<"iteratii"<<endl;

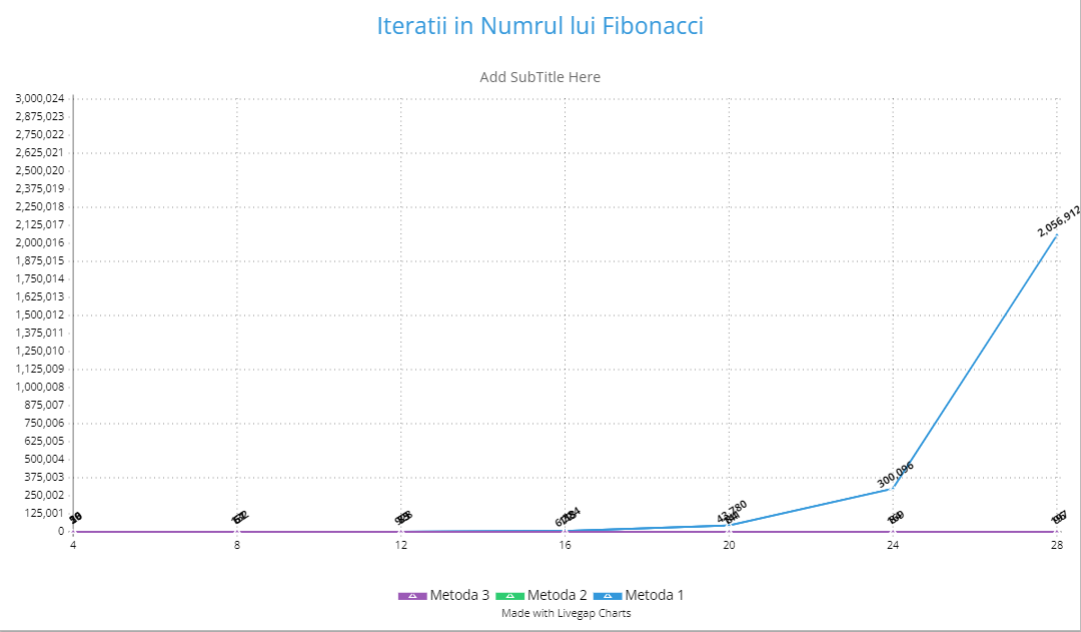
    cout<<"Prin ciclu nr. fib "<<b<<" are "<<it2<<"iteratii"<<endl;

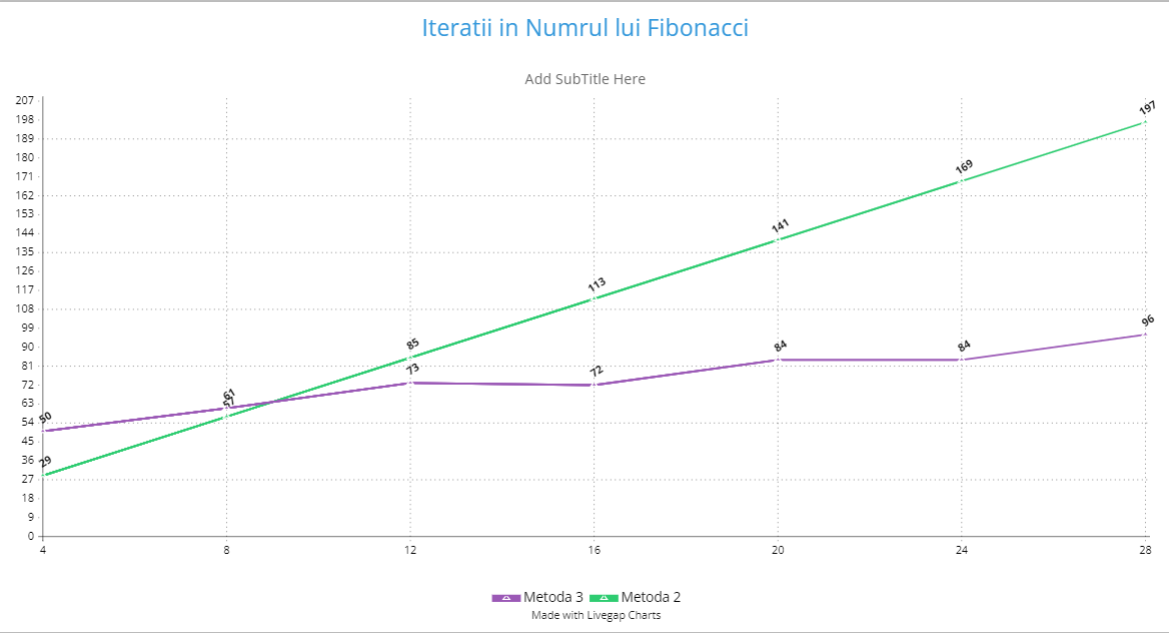
    cout<<"Prin formula nr. fib "<<c<<" are "<<it3<<"iteratii"<<endl;

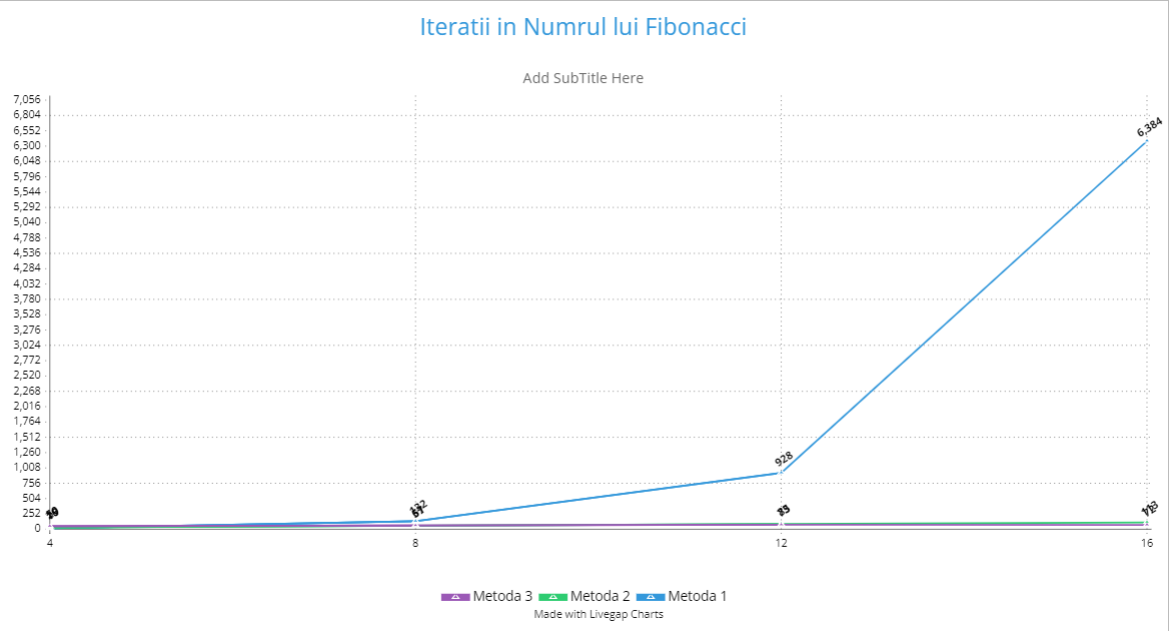
}

**Analiza grafica a algoritmilor:**







****

**Concluzie**

Scopul final al lucrări a fost în analiză și perceperea teoretică a algoritilor de a stabili al n-lea număr din șirul Fibonacci.Pentru această am utilizat trei algoritme diferite pentru a identifica cel mai eficient algoritm care va folosi un număr minim de iterații printand același rezultat.Folosind un simplu tablou am putut identifica dar și analiză rezultatele fiecărui algoritm.În final am format unele concluzii precum că,metodă recursiva este cea mai puțin eficientă, căci această calculează de prea multe ori pe când metodă prin formulă să dovedit a fi cea mai eficientă.Pe de altă parte algoritmul iterativ este unul mediu acesta nu a depășit cu mult numarulde iterații.Întrun final putem spune că cele două metode iterativă și prin formulă sau dovedit a fi cel mai eficiente pentru a află al n-lea număr a lui Fibonacci.