Ministerul Educaţiei şi Cercetării al Republicii Moldova

Universitatea Tehnică a Moldovei

Facultatea Calculatoare, Informatică şi Microelectronică

Raport

Lucrarea de laborator nr. 1

Disciplina: **Analiza si proiectarea algortimilor**

Tema: **Algoritmul lui Fibonacci**

A efectuat:	Student grupa TI-231 FR	Apareci Aurica
A verificat:	Asistent universitar	Andrievschi-Bagrin Veronica

Chişinău 2025

Cuprins

1. Cadru teoretic

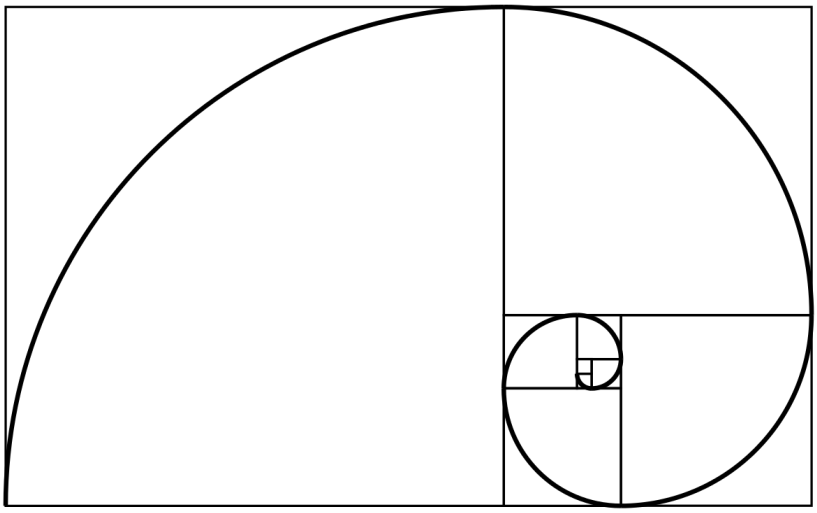
Tema: Numerele lui Fibonacci

Sarcina (conform variantei): Efectuati analiza empirica a algoritmilor de calculare a numerelor fibonacci. Determinati relatia ce determina complexitatea temporala pentru acesti algortmi.

Relatia lui Fibonacci constituie un caz des intalnit in practica. Ea este o relatie de recurenta omogena de ordinul doi cu coeficienti constanti.

$$F_0 = 0, F_1 = 1, F_i = F_{i-1} + F_{i-2} \text{ pentru } i \geq 2.$$

$$F_n = \frac{1}{\sqrt{5}}(((1+\sqrt{5})/2)^n - ((1-\sqrt{5})/2)^n)$$



2. Listingul programului

```
//se utilizeaza urmatoarea biblioteca: GitHub - dotnet/BenchmarkDot-
Net: Powerful .NET library for benchmarking versiunea: 0.13.2
using BenchmarkDotNet.Attributes;
using BenchmarkDotNet.Running;

namespace lab1v2
{
    internal class Program
    {
      static void Main(string[] args)
      {
         BenchmarkRunner.Run<Benchy>();
      }
    }

     [MemoryDiagnoser]
     public class Benchy
     {
        int value = 13;
        [Benchmark]
        public void fib1()
        {
          fibRec(value);
        }
        long fibRec(int a) //recursiva
          {
             if (a == 0)
             return 0;
             if (a == 1)
             return 1;
             return fibRec(a - 1) + fibRec(a - 2);
          }
       [Benchmark]
       public void fib2()
          {
             fibIt(value);
          }
       long fibIt(int a) //iterativa
       {
          long f1 = 0;
          long f2 = 1;
          long f3 = 1;
          for (int i = 0; i < a; i++)
          {
             f3 = f1 + f2;
             f1 = f2;
             f2 = f3;
          }
          return f1;
       }
```

```
    [Benchmark]
    public void fib3()
    {
       fibDir(value);
    }
    double fibDir(int a)
    {
        return (1 / Math.Sqrt(5)) * (Math.Pow(((1 +
        Math.Sqrt(5)) / 2), a) - (Math.Pow(((1 -
        Math.Sqrt(5)) / 2), a)));
    }
  }
}
```

3. Cazuri de testare

Drept cazuri de testare se vor calcula termenii sirului Fibonnaci cu pozitiile: 5,7,10,10,13,15,20,25,50,60,70,80,90,100. Definim urmatoarele functii ce calcue-laza numerele fibonaci utilizand metode diferite:

Fib1() – algoritm recursiv

Fib2() – algoritm iterativ

Fib3() – calcul prin formula definita.

N = 5

Method	Mean	Error	StdDev	Allocated
fib1	27.840 ns	0.5734 ns	0.7655 ns	-
fib2	3.945 ns	0.0980 ns	0.0917 ns	-
fib3	46.580 ns	0.9536 ns	2.1330 ns	-

N = 7

Method	Mean	Error	StdDev	Allocated
fib1	70.109 ns	1.4299 ns	1.6467 ns	-
fib2	6.329 ns	0.1750 ns	0.5134 ns	-
fib3	52.262 ns	1.0661 ns	2.4919 ns	-

N = 10

Method	Mean	Error	StdDev	Allocated
fib1	333.809 ns	6.4376 ns	6.3226 ns	-
fib2	6.715 ns	0.1555 ns	0.1910 ns	-
fib3	43.323 ns	0.4221 ns	0.3742 ns	-

N = 13

Method	Mean	Error	StdDev	Allocated
fib1	1,322.56 ns	25.195 ns	29.014 ns	-
fib2	10.41 ns	0.113 ns	0.088 ns	-
fib3	41.05 ns	0.809 ns	0.757 ns	-

N = 15

Method	Mean	Error	StdDev	Allocated
fib1	4,031.75 ns	80.405 ns	86.033 ns	-
fib2	14.12 ns	0.186 ns	0.174 ns	-
fib3	48.73 ns	1.000 ns	1.401 ns	-

N = 20

Method	Mean	Error	StdDev	Allocated
fib1	48,701.23 ns	918.419 ns	1,976.995 ns	-
fib2	23.39 ns	0.471 ns	0.441 ns	-
fib3	53.22 ns	1.089 ns	1.377 ns	-

N = 25

Method	Mean	Error	StdDev	Allocated
fib1	488,966.91 ns	13,781.197 ns	39,761.903 ns	-
fib2	25.50 ns	0.680 ns	2.005 ns	-
fib3	52.44 ns	1.072 ns	2.090 ns	-

N = 50

Method	Mean	Error	StdDev	Median	Allocated
fib2	33.92 ns	0.690 ns	0.794 ns	33.64 ns	-
fib3	44.26 ns	0.917 ns	1.766 ns	43.56 ns	-

N = 60

Method	Mean	Error	StdDev	Allocated
fib2	41.13 ns	0.689 ns	0.575 ns	-
fib3	49.54 ns	1.224 ns	3.550 ns	-

N = 70

Method	Mean	Error	StdDev	Median	Allocated
fib2	53.11 ns	1.200 ns	3.519 ns	52.70 ns	-
fib3	46.41 ns	0.964 ns	2.555 ns	45.24 ns	-

N = 80

Method	Mean	Error	StdDev	Allocated
fib2	65.89 ns	1.349 ns	3.871 ns	-
fib3	53.38 ns	1.099 ns	2.913 ns	-

N = 90

Method	Mean	Error	StdDev	Allocated
fib2	78.12 ns	2.097 ns	6.151 ns	-
fib3	59.85 ns	1.274 ns	3.675 ns	-

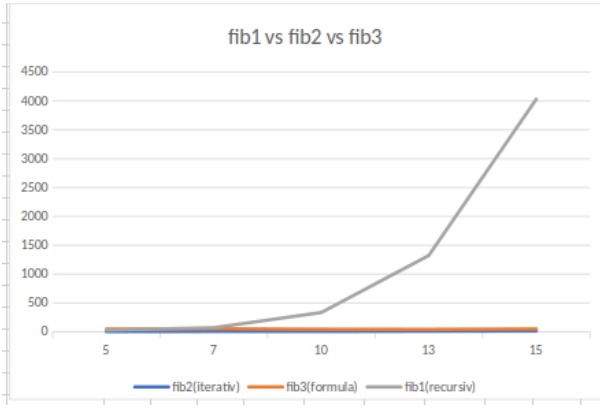
N = 100

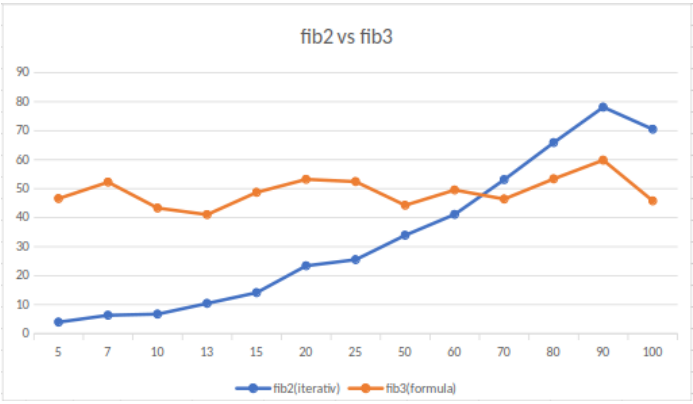
Method	Mean	Error	StdDev	Median	Allocated
fib2	70.50 ns	1.441 ns	3.534 ns	69.21 ns	-
fib3	45.79 ns	0.945 ns	2.539 ns	45.53 ns	-

Tabelul valorilor obtinute

	5	7	10	13	15	20	25	50	60	70	80	90	100
fib2 (iterativ)	3.95	6.33	6.72	10.41	14.12	23.39	25.5	33.92	41.13	53.11	65.89	78.12	70.5
fib3 (formula)	46.6	52.3	43.3	41.05	48.73	53.22	52.44	44.26	49.54	46.41	53.38	59.85	45.79
fib1 (recursiv)	27.8	70.1	334	1322.6	4031.8	48701	488967						

Analiza valorilor obtinute





4. Concluzii

Din cauza neeficientei sale, metoda recursiva a fost exclusa din cercetare dupa calculul celui de al 25-lea numar Fib. In cazul metodei de calcul prin aplicarea algoritmului iterativ, observam o continua crestere a timpului necesar pentru efentuarea operatiilor. Metoda de calcul prin formula poate fi considerata optimala din cauza timpului de executie relativ constant. Potrivit datelor obţinute, la calculul celui de al 68-lea număr Fibonacci, metoda iterativa va avea nevoie de ~ aceiaşi timp ca metoda de calcul prin aplicarea formulei, însă aceste date pot să difere de la maşină de calcul la alta.