**Ministerul Educaţiei și Cercetării al Republicii Moldova**

**Universitatea Tehnică a Moldovei**

**Facultatea Calculatoare, Informatică și Microelectronică**

**RAPORT**

Analiza și proiectarea algoritmilor

|  |  |
| --- | --- |
| A efectuat: | st. Gr. TI-214  Cojocari Dragos |
| A verificat: | asis. univ.  Andrievschi-Bagrin Veronica |

Chișinău 2022

**Lucrarea de laborator nr.1**

**Tema:** *Numerele Fibonacci*

**Scopul lucrării:** *Studierea și formarea algoritmilor de găsire a numerelor Fibonacci*

**Sarcina lucrării:**  Să se creeze un program care va calcula al n-lea număr Fibonacci.

**Codul Programului:**

|  |
| --- |
| #include <iostream>  #include <conio.h>  #define getch() \_getch();  using namespace std;  long long it;  long long fib1(long long x)  {  if (x == 0) { it++; return 0; };  if (x == 1) { it++; return 1; }  else  {  it++; return fib2(x - 1) + fib2(x - 2);  }  }  long long fib2(long long x)  {  long long fib, f1 = 0, f2 = 1, i;  if (x == 1) { fib = 1; it++; };  for (int i = 2; i <= x; i++)  {  fib = f1 + f2;  f1 = f2;  f2 = fib;  it++;  }    return fib;  }  long long fib3(long long x)  {  long long i = 1, j = 0, k = 0, h = 1,t;  while (x>0)  {  if (x % 2 == 1)  {  it++;  t = j\*h;  j = i\*h + j\*k + t;  i = i\*k + t;  }  it++;  t = h\*h;  h = 2 \* k\*h + t;  k = k\*k + t;  x = x / 2;  }  return j;  }  int main()  {  int r;  long long n;    cout << "n= "; cin >> n;  it = 0;  cout << "Metoda 1: " << endl;  cout << fib1(n) <<endl;  cout << "Iteratii: " << it << endl << endl;  it = 0;  cout << "Metoda 2: " << endl;  cout << fib2(n) <<endl;  cout << "Iteratii: " << it << endl << endl;  it = 0;  cout << "Metoda 3: " << endl;  cout << fib3(n) <<endl;  cout << "Iteratii: " << it << endl << endl;  getch();  return 0;  } |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | iteratii | 2 | 5 | 10 | 15 | 20 |
| recursiv | f1(n) | 3 | 9 | 117 | 1973 | 21891 |
| iterativ | f2(n) | 3 | 9 | 27 | 42 | 57 |
| while | f3(n) | 11 | 15 | 22 | 28 | 26 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | iteratii | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| recursiv | f1(n) | 3 | 5 | 9 | 15 | 25 | 41 | 67 |
| iterativ | f2(n) | 3 | 6 | 9 | 12 | 15 | 18 | 21 |
| while | f3(n) | 11 | 14 | 15 | 18 | 18 | 21 | 19 |

**Concluzie:** În urma efectuării acestei lucrări de laborator am studiat complexitatea algoritmilor și necesitatea de a studia complexitatea lor pentru obținerea unui răspuns cît mai rapid dar și corect. Studiind numerele Fibonacci am pus accentul pe 3 algoritmi de bază în căutarea unui număr Fibonacci și am observat că timpul de execuție nu poate fi luat în considerare ca parametru de măsură deoarece el diferă de la calculator la calculator, de resursele calculatorului dar și cît de utilizat este în momentul execuției programului, de aceea am luat numărul iterațiilor care se execută pe parcursul programului. Studiind fiecare algoritm aparte, am observat ca algoritmul recursiv are o creștere a iterațiilor exponențială ceea ce ne spune că la numele mai mari de 10 numărul de iterații crește semnificativ astfel crescînd și timpul de execuție exponențial, din asta putem concluziona ca metoda recursivă nu este printre cele mai bune în cazul dat pentru a afla un număr Fibonacci astfel încât pentru numere mai mari este nevoie de un timp îndelungat de așteptare. Celelalte metode ca for au un număr de iterații ce crește liniar față de numărul de intrare, astfel metodele date sunt eficiente, dar la numere mari timpul crește și el liniar. Folosind alt treilea algoritm am observat ca e cel mai eficient pentru numerele mari, observăm că crește logaritmic.

În concluzie pot spune că analiza algoritmilor este foarte necesară în construcția unui program eficient, verificînd mai mulți algoritmi și analizînd rapiditatea lor ne permite să creem un program rapid și ce utilizează puține resurse față de alți algoritmi mai complecși.