LAB06

Task 1:

#include <iostream>

#include <chrono>

#include <time.h>

using namespace std;

using namespace std::chrono;

int fibonachiI(int number){

int first ;

int second;

int previousSum ;

int fib ;

for (int i = 1; i <= number; i++){

if (i == 1){

first = 0;

//cout << " "<< first << " ";

continue;

}

if (i == 2){

second = 1;

//cout << second << " ";

}

previousSum = first + second;

first = second;

second = previousSum;

//cout << previousSum << " ";

return previousSum;

}

}

int fibonaccRrc(int numb){

if (numb == 0){

return 0;

}

else if (numb == 1){

return 1;

}

else{

return fibonaccRrc(numb - 1) + fibonaccRrc(numb - 2);

}

}

int main(){

cout << "Enter the value of N: ";

int N;

cin >> N;

//cout << fibonaccRrc(N);

//clock\_t t;

//int func;

//t = clock();

//func = fibonaccRrc(N);

//t = (clock() - t) \* 1000000000; // ok

high\_resolution\_clock::time\_point t1 = high\_resolution\_clock::now();

fibonachiI(N);

high\_resolution\_clock::time\_point t2 = high\_resolution\_clock::now();

auto duration = duration\_cast<microseconds>(t2 - t1).count();

cout << endl;

cout << duration;

system("pause");

//return 0;

}

Task 2:

code

#include <iostream>

#include <chrono>

#include <time.h>

using namespace std;

using namespace std::chrono;

int long long factorial(int number){

int fact = 1;

for (int i = 1; i <= number; i++){

fact \*= i;

//cout << fact << " ";

}

return 0;

}

int long long factorialRec(int long numb){

if (numb == 0){

return 1;

}

else if (numb == 1){

return 1;

}

else{

return numb \*factorialRec(numb - 1);

}

}

int main(){

cout << "Enter the value of N: ";

int N;

cin >> N;

//cout << factorialRec(N);

high\_resolution\_clock::time\_point t1 = high\_resolution\_clock::now();

factorial(N);

high\_resolution\_clock::time\_point t2 = high\_resolution\_clock::now();

auto duration = duration\_cast<nanoseconds>(t2 - t1).count();

cout << endl;

cout << duration;

system("pause");

high\_resolution\_clock::time\_point t3 = high\_resolution\_clock::now();

factorialRec(N);

high\_resolution\_clock::time\_point t4 = high\_resolution\_clock::now();

auto duration01 = duration\_cast<nanoseconds>(t4 - t3).count();

cout << endl;

cout << duration01;

system("pause");

return 0;

}

Screenshots: