Учреждение образования

«Белорусский государственный технологический университет»

**Лабораторная работа №1**

**Вспомогательные функции**

Выполнил:

Студент 2 курса 7 группы ФИТ

Кореневский Кирилл Русланович

**ЦЕЛЬ РАБОТЫ: приобретение навыков составления и отладки программ с использованием пользовательских функций для замера продолжительности процесса вычисления.**

***Задание 1***. Разработайте три функции (start, dget и iget)

namespace auxil

{

void start(); // старт генератора сл. чисел

double dget(double rmin, double rmax); // получить случайное число

int iget(int rmin, int rmax); // получить случайное число

};

Файл Auxil.h

#include "Auxil.h"

#include <ctime>

namespace auxil

{

void start() // старт генератора сл. чисел

{

srand((unsigned)time(NULL));

};

double dget(double rmin, double rmax) // получить случайное число

{

return ((double)rand() / (double)RAND\_MAX) \* (rmax - rmin) + rmin;

};

int iget(int rmin, int rmax) // получить случайное число

{

return (int)dget((double)rmin, (double)rmax);

};

}

Файл Auxil.cpp

Разработаны функции для генерации псевдослучайных вещественных и целых чисел; функция, которая устанавливает в качестве начального числа для генератора псевдослучайных чисел текущее значение системного времени.

***Задание 2***. Для проверки работоспособности разработанных функций и приобретения навыков замера продолжительности процесса вычисления реализуйте программу, приведенную в примере 2.

#include <iostream>

#include "Auxil.h"

#define CYCLE 1000000

int main()

{

double av1 = 0, av2 = 0;

clock\_t t1 = 0, t2 = 0;

setlocale(LC\_ALL, "rus");

auxil::start(); // старт генерации

t1 = clock(); // фиксация времени

for (int i = 0; i < CYCLE; i++)

{

av1 += (double)auxil::iget(-100, 100); // сумма случайных чисел

av2 += auxil::dget(-100, 100); // сумма случайных чисел

}

t2 = clock(); // фиксация времени

std::cout << std::endl << "количество циклов: " << CYCLE;

std::cout << std::endl << "среднее значение (int): " << av1 / CYCLE;

std::cout << std::endl << "среднее значение (double): " << av2 / CYCLE;

std::cout << std::endl << "продолжительность (у.е): " << (t2 - t1);

std::cout << std::endl << " (сек): "

<< ((double)(t2 - t1)) / ((double)CLOCKS\_PER\_SEC);

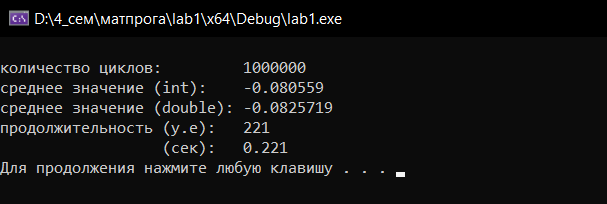
std::cout << std::endl;

system("pause");

return 0;

}

Пример 2

****

****

Для данного алгоритма зависимость линейная

Код программы был изменен для вычисления чисел Фибоначчи:

#include "Auxil.h"

#include <iostream>

#include <fstream>

#include <ctime>

#define N 30

clock\_t SS[N];

int fibonacci(int n)

{

if (n <= 1)

return n;

else

return fibonacci(n - 1) + fibonacci(n - 2);

}

int main()

{

setlocale(LC\_ALL, "rus");

std::ofstream file("fibonacci\_results.csv");

file << "n, Fibonacci, Время выполнения (мс)" << std::endl;

for (int n = 0; n < N; n++)

{

SS[n] = clock();

int result = fibonacci(n);

SS[n] = -(SS[n] - clock());

std::cout << std::endl << "n = " << n << ", Fibonacci = " << result << ", Время выполнения: " << SS[n] << " мс";

file << n << ", " << result << ", " << SS[n] << std::endl;

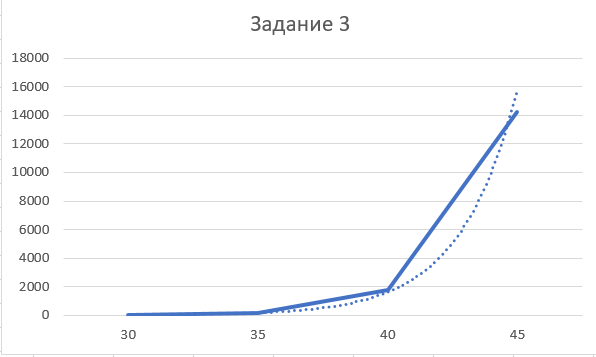
}

file.close();

system("pause");

return 0;

}



Для данного алгоритма зависимость нелинейная

**ВЫВОД: В ходе выполнения лабораторной работы была исследована зависимость продолжительности процесса вычисления от количества циклов. Были приобретены навыки составления и отладки программ с использованием пользовательских функций для замера продолжительности процесса вычисления.**