Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Математическое программирование

Студент: Ворович К. С.

ФИТ 2 курс 5 группа

Лаборант:

Минск 2024

# Лабораторная работа №1. Вспомогательные функции

Разработайте три функции (start, dget и iget), используя следующие спецификации

|  |
| --- |
| #include "stdafx.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

|  |
| --- |
| #pragma once  #include <cstdlib>  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

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

|  |
| --- |
| #include "stdafx.h"  #include "Auxil.h" // вспомогательные функции  #include "Fibonachi.h"  #include <iostream>  #include <ctime>  #include <locale>  #define CYCLE 1000000 // количество циклов  int main(int argc, char\* argv[])  {  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

Проведите необходимые эксперименты и постройте график зависимости (Excel) продолжительности процесса вычисления от количества циклов в примере 2. Проанализируйте характер зависимости. Проведите исследование любого другого рекурсивного алгоритма, например, вычисления факториала или генератора чисел Фибоначчи и включите в отчет график.



График зависимости продолжительности процесса вычисления от количества циклов

**Числа Фибоначчи:**

Последовательность чисел Фибоначчи определяется формулой Fn = Fn-1 + Fn-2. То есть, следующее число получается как сумма двух предыдущих.

Первые два числа равны 1, затем 2(1+1), затем 3(1+2), 5(2+3) и так далее: 1, 1, 2, 3, 5, 8, 13, 21....

Реализацию данного алгоритма и график зависимости времени выполнения от заданного числа можно посмотреть в листинге 1.4 и на рисунке 1.3., соответственно.

|  |
| --- |
| //-- Fibonachi.cpp  #include "Fibonachi.h"  #include <iostream>  // Посчитать N-ое число Фибоначчи  long double fibonachi(int n)  {  if (n == 0)  return 0;  if (n == 1)  return 1;  return fibonachi(n - 1) + fibonachi(n - 2);  }  //-- Lab1.cpp  std::cout << "=============\n\nФункция чисел Фиббоначи";  clock\_t t3 = 0, t4 = 0; int a;  std::cout << std::endl << "Введите N-ое число: ";  std::cin >> a;  t3 = clock();  long double result = fibonachi(a);  std::cout << std::endl << a << " число Фиббоначи = " << result;  t4 = clock();  std::cout << std::endl << "продолжительность (у.е): " << (t4 - t3);  std::cout << std::endl << " (сек): "  << ((double)(t4 - t3)) / ((double)CLOCKS\_PER\_SEC);  std::cout << std::endl; |

Реализация алгоритма чисел Фибоначчи и тестирование

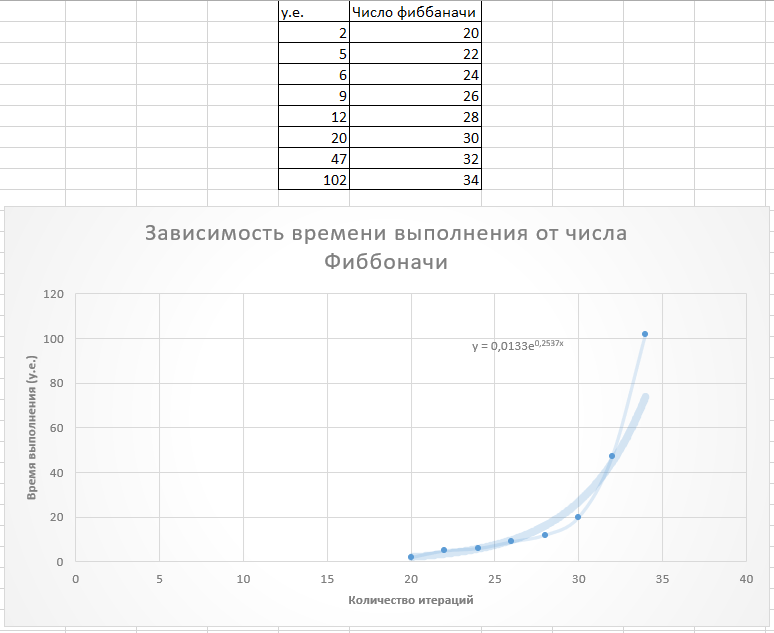


График зависимости продолжительности процесса вычисления от количества циклов

Вывод: скорость выполнения программы линейно зависит от количества итераций цикла. Скорость нахождения N-го числа Фибоначчи имеет вид экспонитиальной зависимости.