Учреждение образования

«Белорусский государственный технологический университет»

**Кафедра информационных систем и технологий**

**Лабораторная работа №1**

**Тема «Вспомогательные функции»**

Выполнила:

Студентка 2 курса 7 группы ФИТ

Курносенко Софья Андреевна

Проверил:

Барковский Евгений Валерьевич

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

**ВЫПОЛНЕНИЕ РАБОТЫ:** составить и реализовать программы.

**Задание 1**

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

//-- установка начального числа для генератора псевдослучайных чисел

// # include "Auxil.h"

// namespace auxil

**void start();**

// функция устанавливает в качестве начального числа для // генератора псевдослучайных чисел текущее значение // системного времени в фомате функции time()

//-- генерация действительного псевдослучайного числа в заданном

// # include "Auxil.h"

// namespace auxil

**double dget(**

**double rmin,** //[in]минимальное значение

**double rmax** //[in]максимальное значение

**);**

//-- функция возвращает действительное псевдослучайное число в // диапазоне оn **rmin** до **rmax**

//-- генерация целого псевдослучайного числа в заданном диапазоне

// # include "Auxil.h"

// namespace auxil

**int iget(**

**int rmin,** //[in]минимальное значение

**int rmax** //[in]максимальное значение

**);**

//-- функция возвращает целое псевдослучайное число в // диапазоне оn **rmin** до **rmax**

**Примечание**: разработанные функции должны располагаться в файле **Auxil.cpp**,  а в файле **Auxil.h –** прототипы функций (см. пример 1).

**Выполненные задания:**

**Auxil.h**

#pragma once

#include <iostream>

#include <cstdlib> // для функций rand() и srand()

#include <ctime> // для функции time()

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

};

**Auxil.cpp**

#include "Auxil.h"

namespace auxil {

// старт генератора случайных чисел

void start() {

srand((unsigned)time(0)); // устанавливаем значение системных часов в качестве стартового числа

};

// получить случайное число типа double в диапазоне [rmin, rmax]

double dget(double rmin, double rmax) {

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

};

// получить случайное число типа int в диапазоне [rmin, rmax]

int iget(int rmin, int rmax) {

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

};

}

**Задание 2**

Реализация программы из примера для проверки работоспособности разработанных функций и приобретения навыков замера продолжительности процесса вычисления.

#include "Auxil.h"

#define CYCLE 1000000

using namespace std;

int main() {

double av1 = 0,

av2 = 0;

clock\_t start = 0,

end = 0;

setlocale(LC\_ALL, "rus");

auxil::start();

start = clock(); // начало подсчета времени, необходимого для выполнения цикла

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

av1 += auxil::iget(-100, 100);

av2 += auxil::dget(-100, 100);

}

end = clock(); // завершение подсчета

cout << "количество циклов:\t" << CYCLE;

cout << "\nсреднее значение (int):\t" << av1 / CYCLE;

cout << "\nсреднее значение (double):\t" << av2 / CYCLE;

// Возвращает количество временных тактов, прошедших с начала запуска программы.

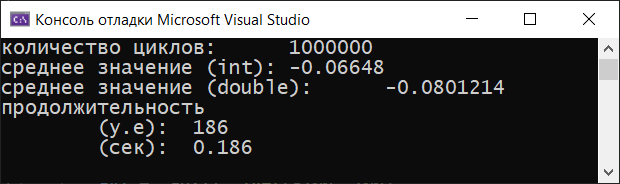
cout << "\nпродолжительность \n\t(у.е):\t" << (end - start);

// С помощью макроса CLOCKS\_PER\_SEC функция получает количество пройденных тактов за 1 секунду.

cout << "\n\t(сек):\t" << ((double)(end - start)) / ((double)CLOCKS\_PER\_SEC);

cout << endl;

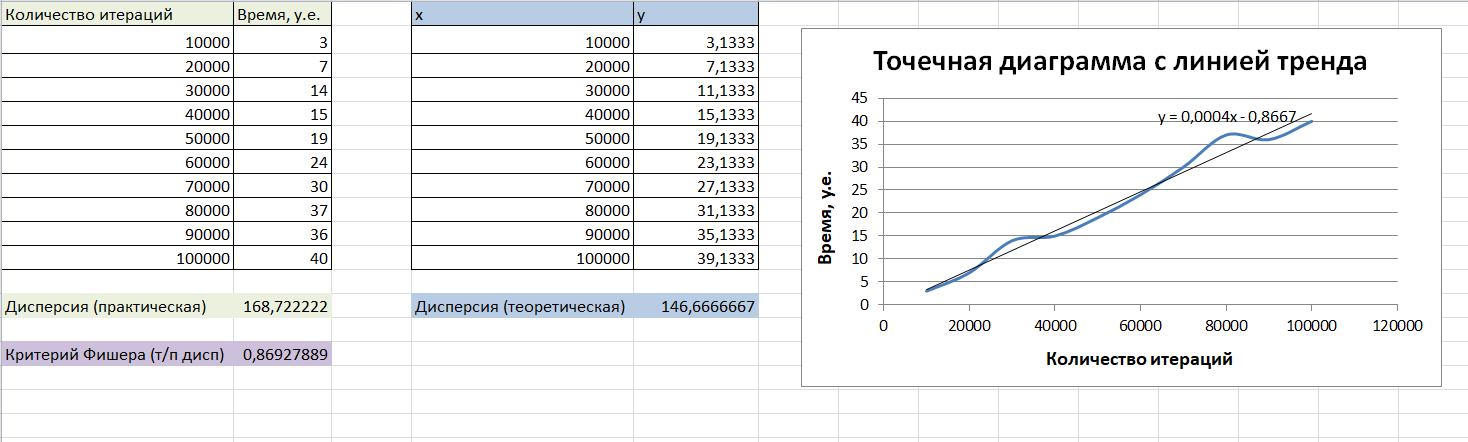
}



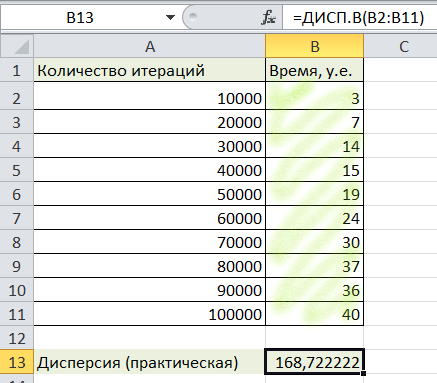
**Задание 3**

Проведите необходимые эксперименты (менять значения CYCLE и смотреть на то, как это влияет на количество тактов) и постройте график зависимости (Excel) продолжительности процесса вычисления от количества циклов в примере 2.

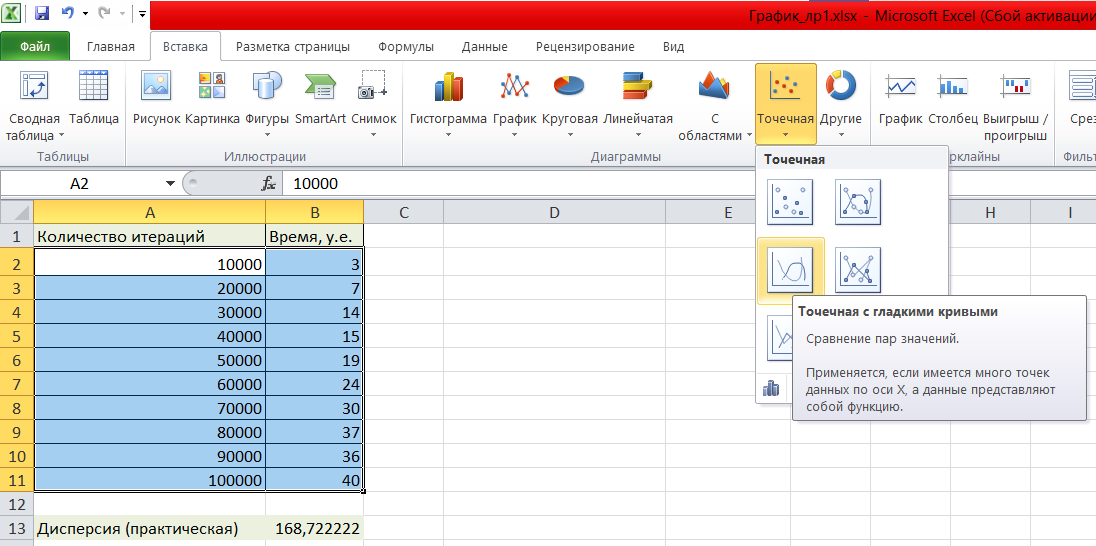
**Примечание**: продолжительность вычисления измерять в условных единицах процессорного времени (функция **clock**).



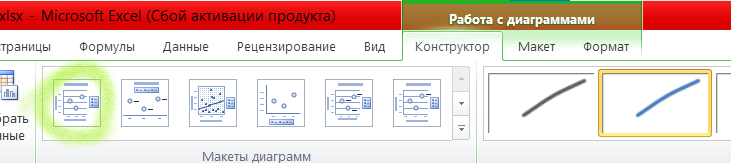
Значения первой таблицы со столбцами «Количество итераций» и «Время, у.е.» получаем запуская код с соответствующим количеством итераций. Практическая дисперсия рассчитывается при помощи функции ДИСП.В(), где в скобках записываем массив зависимых величин, игреки, то есть время:



Вставляем график. Выделяем столбцы, Вставка → Точечная → Точечная с гладкими кривыми:



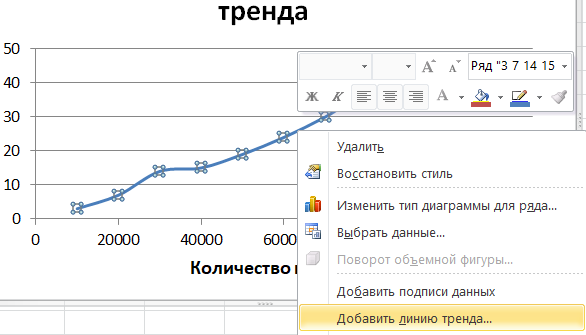
Выбираем первый макет и вводим названия графика и осей:



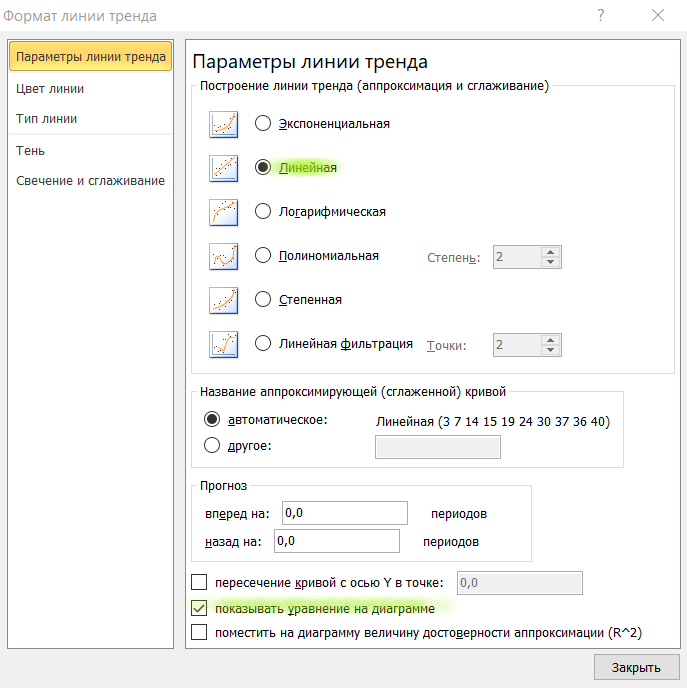
Получаем график:



Добавляем линию тренда. ПКМ по любой точке на кривой → Добавить линию тренда:



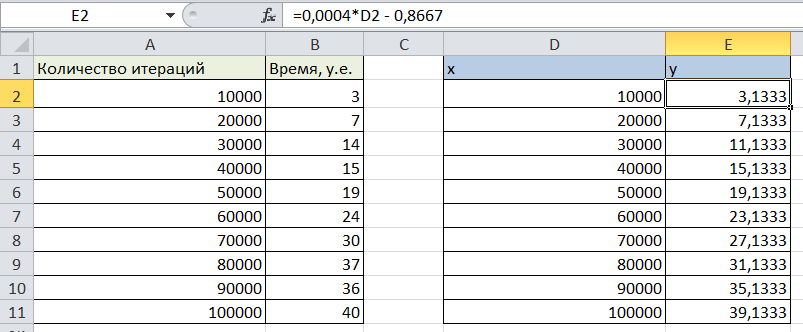
В параметрах выбираем линейную линию тренда и ставим флажок в пункте «Показывать уравнение на диаграмме»:



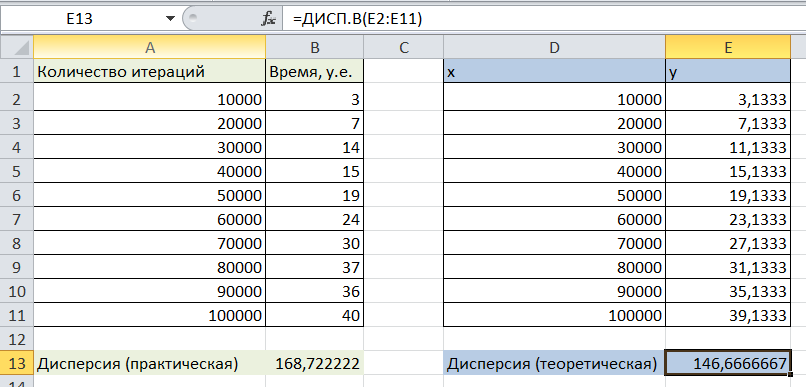
Получаем диаграмму:



Используем данное уравнение линии тренда для построения таблицы с теоретическими значениями x и y. Для этого значение x принимаем равным соответствующим значениям из столбца «Количество итераций» первой таблицы и подставляем в уравнение для расчета y:



Рассчитываем теоретическую дисперсию:



Критерий Фишера = теоретическая дисперсия, поделенная на практическую:

