**Лабораторная работа 1. Вспомогательные функции**

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

Для достижения поставленной цели были выполнены следующие шаги:

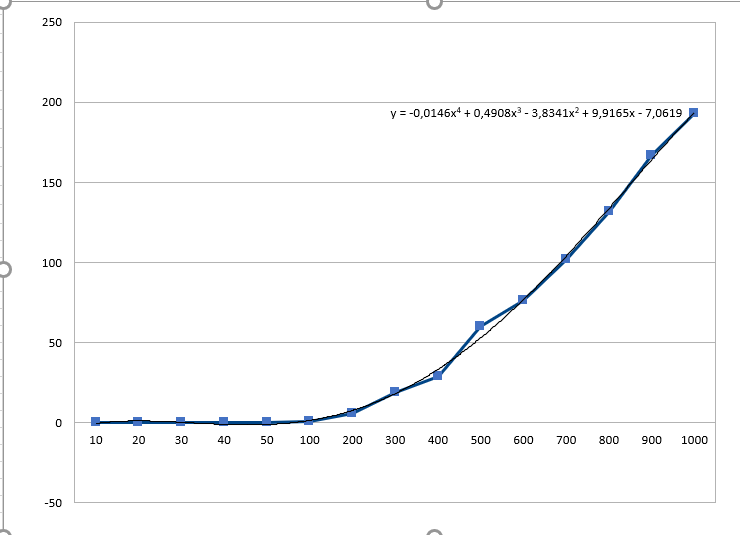
1. Написана программа на C++, реализующая вычисление факториала числа с использованием рекурсии.

#include <iostream>  
#include <cstdint>  
#include <ctime>  
#include <string>  
  
using namespace std;  
*//task 1*void start()  
{  
 srand(time(NULL));  
}  
  
double dget(double rmin, double rmax)  
{  
 return ((double) rand() / (double) RAND\_MAX) \* (rmax - rmin) + rmin;  
}  
  
int iget(int rmin, int rmax)  
{  
 return (int)dget((double)rmin, (double)rmax);   
}  
*//task 2*#define CYCLE 10000  
string multiplyStrings(string num1, string num2)  
{  
 int len1 = num1.length();  
 int len2 = num2.length();  
 string result(len1 + len2, '0');  
  
 for (int i = len1 - 1; i >= 0; --i)  
 {  
 int carry = 0;  
 for (int j = len2 - 1; j >= 0; --j)  
 {  
 int digit1 = num1[i] - '0';  
 int digit2 = num2[j] - '0';  
  
 int product = digit1 \* digit2 + carry + (result[i + j + 1] - '0');  
  
 carry = product / 10;  
 result[i + j + 1] = '0' + (product % 10);  
 }  
  
 result[i] += carry;  
 }  
  
 size\_t startpos = result.find\_first\_not\_of('0');  
 if (startpos != string::npos)  
 {  
 return result.substr(startpos);  
 }  
 return "0";  
}  
  
string strFactorial(const string& n)  
{  
 if (n == "0" || n == "1")  
 {  
 return "1";  
 }  
  
 string prevFactorial = strFactorial(to\_string(stoll(n) - 1));  
 return multiplyStrings(n, prevFactorial);  
}  
  
int main(int argc, char\*\* argv)  
{   
 int n = 10;  
 constexpr int n2 = 5;  
 double av1 = 0, av2 = 0;  
 clock\_t t1 = 0, t2 = 0;  
  
 start();  
  
 t1 = clock();  
 for(int i = 0; i < CYCLE; ++i)  
 {  
 av1 += (double)iget(-100, 100);  
 av2 += dget(-100, 100);  
 }  
 t2 = clock();  
  
 cout << endl << "number of cycles: " << CYCLE; *//количество циклов* cout << endl << "the average value (int): " << av1/CYCLE; *//среднее значение int* cout << endl << "the average value (double): " << av2/CYCLE; *//среднее значение double* cout << endl << "duration (y.e): " << (t2-t1);  
 cout << endl << " (sek): " << ((double)(t2-t1))/((double)CLOCKS\_PER\_SEC); *//сек* cout << endl << endl;   
  
 t1 = clock();  
 cout << strFactorial("1000") << endl;  
 t2 = clock();  
  
 cout << endl << " (sek): " << ((double)(t2-t1))/((double)CLOCKS\_PER\_SEC); *//сек* cout << endl << "duration (y.e): " << (t2-t1);  
 cout << endl << endl;   
  
 return 0;  
}

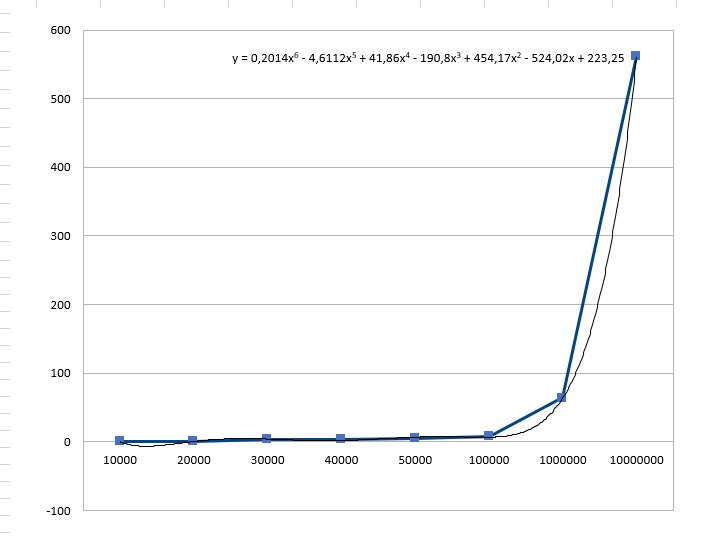
2. Для проведения эксперимента были выбраны различные значения параметра (номер факториала) - 10, 20, 30, 40, 50, 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000.

3. Запущена программа для каждого значения параметра, измерена продолжительность вычислений с помощью функции `clock()` и записаны полученные значения в таблицу Excel.

|  |  |
| --- | --- |
| 10 | 0 |
| 20 | 0 |
| 30 | 0 |
| 40 | 0 |
| 50 | 0 |
| 100 | 1 |
| 200 | 6 |
| 300 | 19 |
| 400 | 29 |
| 500 | 60 |
| 600 | 76 |
| 700 | 102 |
| 800 | 132 |
| 900 | 167 |
| 1000 | 201 |



|  |  |
| --- | --- |
| 1000 | 0 |
| 10000 | 0 |
| 20000 | 1 |
| 30000 | 3 |
| 40000 | 4 |
| 50000 | 5 |
| 100000 | 8 |
| 1000000 | 64 |
| 10000000 | 561 |



4. Построен график зависимости продолжительности процесса вычисления от номера факториала, используя полученные значения из таблицы Excel.

**ВЫВОД**. В заключении можно отметить, что алгоритм вычисления факториала с использованием рекурсии имеет экспоненциальную сложность, что означает, что время выполнения растет очень быстро с увеличением размера входных данных. Для больших значений параметра вычисления факториала следует использовать более эффективные алгоритмы, такие как итеративный подход или использование таблицы значений, чтобы сократить время выполнения.