**Лабораторная работа 1. Вспомогательные функции**

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

Важно:

**stdafx.h** - это стандартное название предварительно скомпилированного заголовочного файла в Visual Studio 2017 и более ранних версиях. Начиная с VS2019, вместо этого используется **pch.h**. Его не нужно ниоткуда скачивать, в него вписываются часто включаемые в коде заголовочные файлы для ускорения компиляции. Сам файл должен создаваться автоматически при создании проекта; но если его нет, можно добавить его вручную. Содержимое должно может выглядеть, например, так:

#pragma once

#include <stdio.h>

//другие директивы include...

Совместно с ним в проекте должен находится **pch.cpp**, состоящий из единственной строки:

#include "pch.h"

Для использования предварительно скомпилированных заголовков их необходимо включить параметром /Y или через страницу свойств проекта **Configuration Properties > C/C++ > Precompiled Headers**. После этого в каждом cpp-файле в самом начале (до любых директив препроцессора или строк кода) нужно добавить #include "pch.h".

**ТЕОРЕТИЧЕСКОЕ ВВЕДЕНИЕ:**

**Генерация случайных чисел**

//-- установка начального числа для генератора псевдослучайных чисел

// # include <cstdlib>

**void srand (**

**unsigned int s** // [in] стартовое число генератора

**)**

//-- генератор псевдослучайного целого числа

// # include <cstdlib>

**int rand ()**

//-- функция возвращает псевдослучайное целоче число от **0** до // **RAND\_MAX**

**Функции времени**

//-- получение текущей даты и времени

// # include <ctime>

**time\_t time (**

**time\_t\* t** //[out] указатель на буфер (8 байт)

**)**

//-- в качестве параметра может быть указан **NULL**

/\*-- функция возвращает количество секунд прошедшие с **00:00:00** **01.01.1970** к точке вызова или/и в буфер, если параметр **t** не **NULL** \*/

//-- получение процессорного времени

// # include <ctime>

**clock\_t clock()**

//-- функция возвращает количество единиц процессорного

// времени (1 сек = **CLOCKS\_PER\_SEC** единиц) прошедших с

// момента старта приложения

**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 // заданом

// # 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 заданом // 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).

**Пример 1**

//-- Auxil.h

#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

};

//-- Auxil.cpp

#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);

};

}

***Задание 2***

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

**Пример 2.**

#include "stdafx.h"

#include "Auxil.h" // вспомогательные функции

#include <iostream>

#include <ctime>

#include <locale>

#define CYCLE 1000000 // количество циклов

int \_tmain(int argc, \_TCHAR\* 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): "

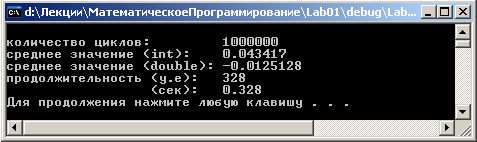
<<((double)(t2-t1))/((double)CLOCKS\_PER\_SEC);

std::cout<<std::endl;

system("pause");

return 0;

}



**Задание 3**

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 (прим. – например вычислите каким будет 100-е, 200-е, 300-е и т.д число), и включите в отчет график.

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

**Пример применения**

#include "stdafx.h"

#include <iostream>

#include <ctime>

#include "Knapsack.h"

#include "Auxil.h"

x

clock\_t SS[N];

int \_tmain(int argc, \_TCHAR\* argv[])

{

setlocale(LC\_ALL, "rus");

int V = 1000, // вместимость рюкзака

v[N], // размер предмета каждого типа

c[N]; // стоимость предмета каждого типа

short m[N]; // количество предметов каждого типа {0,1}

int maxcc = 0;

int n = 0;

auxil::start();

for(int i = 0; i < N; i++) v[i] = auxil::iget(10, 100);

for(int i = 0; i < N; i++) c[i] = auxil::iget(5, 50);

for (int n = 10; n < 21; n++)

{

SS[n] = clock();

maxcc = knapsack\_s(V,n, v, c, m ); // измеряемая функция

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

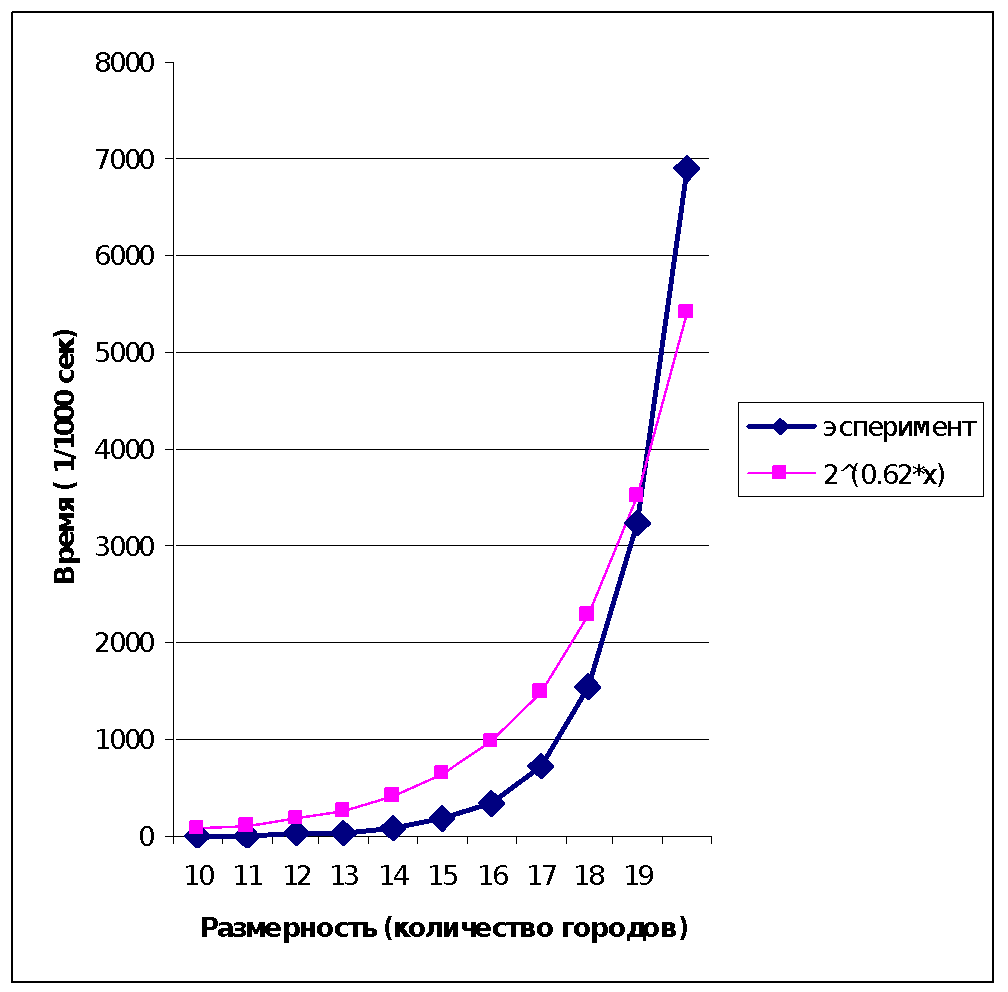
std::cout<< std::endl<<"n = "<<n << " " << SS[n];

}

system("pause");

return 0;

}



**Отчет по лабораторным работам оформите в единый документ. Он понадобится вам перед экзаменом.**