Учреждение образования

«Белорусский государственный технологический университет»

**Кафедра информационных систем и технологий**

**Лабораторная работа №2**

**Тема «Комбинаторные алгоритмы решения оптимизационных задач»**

Выполнила:

Студентка 2 курса 7 группы ФИТ

Курносенко Софья Андреевна

Проверил:

Барковский Евгений Валерьевич

**ЦЕЛЬ РАБОТЫ:** приобрести навыки разработки генераторов подмножеств, перестановок, сочетаний и размещений на С++; научиться применять разработанные генераторы для решения задач о рюкзаке (упрощенную, коммивояжера, об оптимальной загрузке судна и об оптимальной загрузке судна с центровкой.

**ЗАДАНИЕ ДЛЯ ВЫПОЛНЕНИЯ:**

**Задание 1.** Разработать генератор подмножеств заданного множества.

Шаблон структуры генератора множества всех подмножеств:

// Combi1.h

// Шаблон структуры генератора множества всех подмножеств

#pragma once

namespace combi1

{

struct subset // генератор множества всех подмножеств

{

short n, // количество элементов исходного множества < 64

sn, // количество элементов текущего подмножества

\* sset; // массив индексов текущего подмножества

unsigned \_\_int64 mask; // битовая маска

subset(short n = 1); // конструктор(кол-во эл-ов исх. мн-ва)

short getfirst(); // сформ. массив индексов по битовой маске

short getnext(); // ++маска и сформировать массив индексов

short ntx(short i); // получить i-й элемент массива индексов

unsigned \_\_int64 count(); // вычислить общее кол-во подмножеств

void reset(); // сбросить генератор, начать сначала

};

};

Реализация методов структуры subset:

// Combi1.cpp

// Реализация методов структуры subset

#include "Combi1.h"

namespace combi1

{

subset::subset(short n) {

this->n = n;

this->sset = new short[n];

this->reset();

};

void subset::reset() {

this->sn = 0;

this->mask = 0;

};

short subset::getfirst() {

\_\_int64 buf = this->mask;

this->sn = 0;

for (short i = 0; i < n; i++) {

if (buf & 0x1) this->sset[this->sn++] = i;

buf >>= 1;

}

return this->sn;

};

short subset::getnext() {

int rc = -1;

this->sn = 0;

if (++this->mask < this->count()) rc = getfirst();

return rc;

};

short subset::ntx(short i) {

return this->sset[i];

};

unsigned \_\_int64 subset::count() {

return (unsigned \_\_int64)(1 << this->n);

};

};

Пример применения генератора множества всех подмножеств:

// Main

#include <iostream>

#include "Combi1.h"

#include <tchar.h>

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

{

setlocale(LC\_ALL, "rus");

char AA[][2] = { "A", "B", "C", "D" };

std::cout << std::endl << " - Генератор множества всех подмножеств -";

std::cout << std::endl << "Исходное множество: ";

std::cout << "{ ";

for (int i = 0; i < sizeof(AA) / 2; i++)

std::cout << AA[i] << ((i < sizeof(AA) / 2 - 1) ? ", " : " ");

std::cout << "}";

std::cout << std::endl << "Генерация всех подмножеств ";

combi1::subset s1(sizeof(AA) / 2); // создание генератора

int n = s1.getfirst(); // первое (пустое) подмножество

while (n >= 0) // пока есть подмножества

{

std::cout << std::endl << "{ ";

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

std::cout << AA[s1.ntx(i)] << ((i < n - 1) ? ", " : " ");

std::cout << "}";

n = s1.getnext(); // cледующее подмножество

};

std::cout << std::endl << "всего: " << s1.count() << std::endl;

system("pause");

return 0;

}

**Задание 2.** Разработать генератор сочетаний

Шаблон структуры генератора сочетаний:

// Combi2.h

// Шаблон структуры генератора сочетаний

#pragma once

namespace combi2

{

struct xcombination // генератор сочетаний (эвристика)

{

short n, // количество элементов исходного множества

m, // количество элементов в сочетаниях

\* sset; // массив индексов текущего сочетания

xcombination(

short n = 1, //количество элементов исходного множества

short m = 1 // количество элементов в сочетаниях

);

void reset(); // сбросить генератор, начать сначала

short getfirst(); // сформировать первый массив индексов

short getnext(); // сформировать следующий массив индексов

short ntx(short i); // получить i-й элемент массива индексов

unsigned \_\_int64 nc; // номер сочетания 0,..., count()-1

unsigned \_\_int64 count() const; // вычислить количество сочетаний

};

};

Реализация функций генератора сочетаний:

// Combi2.cpp

// Реализация функций генератора сочетаний

#include "Combi2.h"

namespace combi2

{

xcombination::xcombination(short n, short m) {

this->n = n;

this->m = m;

this->sset = new short[m + 2];

this->reset();

}

void xcombination::reset() // сбросить генератор, начать сначала

{

this->nc = 0;

for (int i = 0; i < this->m; i++) this->sset[i] = i;

this->sset[m] = this->n;

this->sset[m + 1] = 0;

};

short xcombination::getfirst() {

return (this->n >= this->m) ? this->m : -1;

};

short xcombination::getnext() // сформировать следующий массив индексов

{

short rc = getfirst();

if (rc > 0)

{

short j;

for (j = 0; this->sset[j] + 1 == this->sset[j + 1]; ++j) this->sset[j] = j;

if (j >= this->m) rc = -1;

else {

this->sset[j]++;

this->nc++;

};

}

return rc;

};

short xcombination::ntx(short i) { return this->sset[i]; };

unsigned \_\_int64 fact(unsigned \_\_int64 x) { return(x == 0) ? 1 : (x \* fact(x - 1)); };

unsigned \_\_int64 xcombination::count() const

{

return (this->n >= this->m) ?

fact(this->n) / (fact(this->n - this->m) \* fact(this->m)) : 0;

};

};

Пример применения генератора сочетаний:

// Main

#include <iostream>

#include <tchar.h>

#include "Combi2.h"

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

{

setlocale(LC\_ALL, "rus");

char AA[][2] = { "A", "B", "C", "D", "E" };

std::cout << std::endl << " --- Генератор сочетаний ---";

std::cout << std::endl << "Исходное множество: ";

std::cout << "{ ";

for (int i = 0; i < sizeof(AA) / 2; i++)

std::cout << AA[i] << ((i < sizeof(AA) / 2 - 1) ? ", " : " ");

std::cout << "}";

std::cout << std::endl << "Генерация сочетаний ";

combi2::xcombination xc(sizeof(AA) / 2, 3);

std::cout << "из " << xc.n << " по " << xc.m;

int n = xc.getfirst();

while (n >= 0)

{

std::cout << std::endl << xc.nc << ": { ";

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

std::cout << AA[xc.ntx(i)] << ((i < n - 1) ? ", " : " ");

std::cout << "}";

n = xc.getnext();

};

std::cout << std::endl << "всего: " << xc.count() << std::endl;

system("pause");

return 0;

}

**Задание 3.** Разработать генератор перестановок

Шаблон структуры генератора перестановок:

// Combi3.h

// Шаблон структуры генератора перестановок

#pragma once

namespace combi3

{

struct permutation // генератор перестановок

{

const static bool L = true; // левая стрелка

const static bool R = false; // правая стрелка

short n, // количество элементов исходного множества

\* sset; // массив индексов текущей перестановки

bool\* dart; // массив стрелок (левых-L и правых-R)

permutation(short n = 1); // конструктор (кол-во эл-ов исх. мн-ва)

void reset(); // сбросить генератор, начать сначала

\_\_int64 getfirst(); // сформировать первый массив индексов

\_\_int64 getnext(); // сформировать случайный массив индексов

short ntx(short i); // получить i-й элемент масива индексов

unsigned \_\_int64 np; // номер перествновки 0,... count()-1

unsigned \_\_int64 count() const; // вычислить общее кол. перестановок

};

};

Реализация функций генератора перестановок:

// Combi3.cpp

// Реализация функций генератора перестановок

#include "Combi3.h"

#include <algorithm>

#define NINF ((short)0x8000)

namespace combi3

{

permutation::permutation(short n) {

this->n = n;

this->sset = new short[n];

this->dart = new bool[n];

this->reset();

};

void permutation::reset() { this->getfirst(); };

\_\_int64 permutation::getfirst() {

this->np = 0;

for (int i = 0; i < this->n; i++) { this->sset[i] = i; this->dart[i] = L; };

return (this->n > 0) ? this->np : -1;

};

\_\_int64 permutation::getnext() {

\_\_int64 rc = -1;

short maxm = NINF, idx = -1;

for (int i = 0; i < this->n; i++) {

if (i > 0 &&

this->dart[i] == L &&

this->sset[i] > this->sset[i - 1] &&

maxm < this->sset[i]) maxm = this->sset[idx = i];

if (i < (this->n - 1) &&

this->dart[i] == R &&

this->sset[i] > this->sset[i + 1] &&

maxm < this->sset[i]) maxm = this->sset[idx = i];

};

if (idx >= 0) {

std::swap(this->sset[idx], this->sset[idx + (this->dart[idx] == L ? -1 : 1)]);

std::swap(this->dart[idx], this->dart[idx + (this->dart[idx] == L ? -1 : 1)]);

for (int i = 0; i < this->n; i++)

if (this->sset[i] > maxm) this->dart[i] = !this->dart[i];

rc = ++this->np;

}

return rc;

};

short permutation::ntx(short i) { return this->sset[i]; };

unsigned \_\_int64 fact(unsigned \_\_int64 x) { return (x == 0) ? 1 : (x \* fact(x - 1)); };

unsigned \_\_int64 permutation::count() const { return fact(this->n); };

};

Пример применения генератора перестановок:

// Main

#include <iostream>

#include "Combi3.h"

#include <iomanip>

#include <tchar.h>

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

{

setlocale(LC\_ALL, "rus");

char AA[][2] = { "A", "B", "C", "D" };

std::cout << std::endl << " --- Генератор перестановок ---";

std::cout << std::endl << "Исходное множество: ";

std::cout << "{ ";

for (int i = 0; i < sizeof(AA) / 2; i++)

std::cout << AA[i] << ((i < sizeof(AA) / 2 - 1) ? ", " : " ");

std::cout << "}";

std::cout << std::endl << "Генерация перестановок ";

combi3::permutation p(sizeof(AA) / 2);

\_\_int64 n = p.getfirst();

while (n >= 0)

{

std::cout << std::endl << std::setw(4) << p.np << ": { ";

for (int i = 0; i < p.n; i++)

std::cout << AA[p.ntx(i)] << ((i < p.n - 1) ? ", " : " ");

std::cout << "}";

n = p.getnext();

};

std::cout << std::endl << "всего: " << p.count() << std::endl;

system("pause");

return 0;

}

**Задание 4.** Разработать генератор размещений

Шаблон структуры генератора размещений:

// Combi4.h

// Шаблон структуры генератора размещений

#pragma once

namespace combi4

{

struct xcombination // генератор сочетаний (эвристика)

{

short n, // кол-во элементов исходного множества

m, // количество элементов в сочетаниях

\* sset; // массив индексов текущего сочетания

xcombination(

short n = 1, //количество элементов исходного множества

short m = 1 // количество элементов в сочетаниях

);

void reset(); // сбросить генератор, начать сначала

short getfirst(); // сформ. первый массив индексов

short getnext(); // сформ. следующий массив индексов

short ntx(short i); // получить i-й эл-т массива индексов

unsigned \_\_int64 nc; // номер сочетания 0,..., count()-1

unsigned \_\_int64 count() const; // вычислить кол-во сочетаний

};

struct permutation // генератор перестановок

{

const static bool L = true; // левая стрелка

const static bool R = false; // правая стрелка

short n, // количество элементов исходного множества

\* sset; // массив индексов текущей перестановки

bool\* dart; // массив стрелок (левых-L и правых-R)

permutation(short n = 1); // конструктор (кол-во эл-ов исх. мн-ва)

void reset(); // сбросить генератор, начать сначала

\_\_int64 getfirst(); // сформ. первый массив индексов

\_\_int64 getnext(); // сформ. случайный массив индексов

short ntx(short i); // получить i-й элемент масива индексов

unsigned \_\_int64 np; // номер перествновки 0,... count()-1

unsigned \_\_int64 count() const; // вычислить общее кол. перестановок

};

struct accomodation // генератор размещений

{

short n, // количество элементов исходного множества

m, // количество элементов в размещении

\* sset; // массив индесов текущего размещения

xcombination\* cgen; // указатель на генератор сочетаний

permutation\* pgen; // указатель на генератор перестановок

accomodation(short n = 1, short m = 1); // конструктор

void reset(); // сбросить генератор, начать сначала

short getfirst(); // сформировать первый массив индексов

short getnext(); // сформировать следующий массив индексов

short ntx(short i); // получить i-й элемент массива индексов

unsigned \_\_int64 na; // номер размещения 0, ..., count()-1

unsigned \_\_int64 count() const; // общее количество размещений

};

}

Реализация функций генератора размещений:

// Combi4.cpp

// Реализация функций генератора размещений

#include "Combi4.h"

#include <algorithm>

#define NINF ((short)0x8000)

namespace combi4

{

unsigned \_\_int64 fact(unsigned \_\_int64 x) { return(x == 0) ? 1 : (x \* fact(x - 1)); };

xcombination::xcombination(short n, short m) {

this->n = n;

this->m = m;

this->sset = new short[m + 2];

this->reset();

}

void xcombination::reset() // сбросить генератор, начать сначала

{

this->nc = 0;

for (int i = 0; i < this->m; i++) this->sset[i] = i;

this->sset[m] = this->n;

this->sset[m + 1] = 0;

};

short xcombination::getfirst() { return (this->n >= this->m) ? this->m : -1; };

short xcombination::getnext() // сформировать следующий массив индексов

{

short rc = getfirst();

if (rc > 0) {

short j;

for (j = 0; this->sset[j] + 1 == this->sset[j + 1]; ++j) this->sset[j] = j;

if (j >= this->m) rc = -1;

else {

this->sset[j]++;

this->nc++;

};

}

return rc;

};

short xcombination::ntx(short i) { return this->sset[i]; };

unsigned \_\_int64 xcombination::count() const {

return (this->n >= this->m) ?

fact(this->n) / (fact(this->n - this->m) \* fact(this->m)) : 0;

};

permutation::permutation(short n) {

this->n = n;

this->sset = new short[n];

this->dart = new bool[n];

this->reset();

};

void permutation::reset() { this->getfirst(); };

\_\_int64 permutation::getfirst() {

this->np = 0;

for (int i = 0; i < this->n; i++) { this->sset[i] = i; this->dart[i] = L; };

return (this->n > 0) ? this->np : -1;

};

\_\_int64 permutation::getnext() {

\_\_int64 rc = -1;

short maxm = NINF, idx = -1;

for (int i = 0; i < this->n; i++) {

if (i > 0 && this->dart[i] == L && this->sset[i] > this->sset[i - 1]

&& maxm < this->sset[i]) maxm = this->sset[idx = i];

if (i < (this->n - 1) && this->dart[i] == R

&& this->sset[i] > this->sset[i + 1]

&& maxm < this->sset[i]) maxm = this->sset[idx = i];

};

if (idx >= 0)

{

std::swap(this->sset[idx], this->sset[idx + (this->dart[idx] == L ? -1 : 1)]);

std::swap(this->dart[idx], this->dart[idx + (this->dart[idx] == L ? -1 : 1)]);

for (int i = 0; i < this->n; i++)

if (this->sset[i] > maxm) this->dart[i] = !this->dart[i];

rc = ++this->np;

}

return rc;

};

short permutation::ntx(short i) { return this->sset[i]; };

unsigned \_\_int64 permutation::count() const { return fact(this->n); };

accomodation::accomodation(short n, short m) {

this->n = n;

this->m = m;

this->cgen = new xcombination(n, m);

this->pgen = new permutation(m);

this->sset = new short[m];

this->reset();

}

void accomodation::reset() {

this->na = 0;

this->cgen->reset();

this->pgen->reset();

this->cgen->getfirst();

};

short accomodation::getfirst() {

short rc = (this->n >= this->m) ? this->m : -1;

if (rc > 0) {

for (int i = 0; i <= this->m; i++)

this->sset[i] = this->cgen->sset[this->pgen->ntx(i)];

};

return rc;

};

short accomodation::getnext() {

short rc;

this->na++;

if ((this->pgen->getnext()) > 0) rc = this->getfirst();

else if ((rc = this->cgen->getnext()) > 0)

{

this->pgen->reset(); rc = this->getfirst();

};

return rc;

};

short accomodation::ntx(short i) { return this->sset[i]; };

unsigned \_\_int64 accomodation::count() const {

return (this->n >= this->m) ? fact(this->n) / fact(this->n - this->m) : 0;

};

}

Пример использования генератора перестановок:

// main

#include "stdafx.h"

#include <iostream>

#include <iomanip>

#include <tchar.h>

#include "Combi4.h"

#define N (sizeof(AA)/2)

#define M 3

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

{

setlocale(LC\_ALL, "rus");

char AA[][2] = { "A", "B", "C", "D" };

std::cout << std::endl << " --- Генератор размещений ---";

std::cout << std::endl << "Исходное множество: ";

std::cout << "{ ";

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

std::cout << AA[i] << ((i < N - 1) ? ", " : " ");

std::cout << "}";

std::cout << std::endl << "Генерация размещений из " << N << " по " << M;

combi4::accomodation s(N, M);

int n = s.getfirst();

while (n >= 0) {

std::cout << std::endl << std::setw(2) << s.na << ": { ";

for (int i = 0; i < 3; i++)

std::cout << AA[s.ntx(i)] << ((i < n - 1) ? ", " : " ");

std::cout << "}";

n = s.getnext();

};

std::cout << std::endl << "всего: " << s.count() << std::endl;

system("pause");

return 0;

}

**Задание 5.**  Решить в соответствии с вариантом задачу:

(Вариант 4, 8, 12, 16) об оптимальной загрузке судна с условием центровки (веса контейнеров сгенерировать случайным образом: количество мест на судне для контейнеров – 5, количество контейнеров 8, веса контейнеров 100 – 200 кг., доход от перевозки 10 – 100 у.е.; минимальный вес контейнера для каждого места 50 – 120 кг, максимальный вес контейнера для каждого места 150 – 850 кг);

Генератор псевдослучайных чисел берем из 2 лабораторной:

Шаблон генератора:

#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

};

Реализация:

#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);

};

}

**Решение задачи.**

Функция boat\_c, решающая задачу об оптимальном размещении контейнеров на судне:

// ВoatС.h

// Функция boat\_c, решающая задачу об оптимальном размещении контейнеров на судне

#pragma once

#include "Combi4.h"

int boat\_с(

short m, // [in] количество мест для контейнеров

int minv[], // [in] минимальный вес контейнера на каждом месте

int maxv[], // [in] максимальный вес контейнера на каждом месте

short n, // [in] всего контейнеров

const int v[],// [in] вес каждого контейнера

const int c[],// [in] доход от перевозки каждого контейнера

short r[] // [out] номера выбранных контейнеров

);

Реализация функции boat\_c:

// ВoatС.cpp

// Реализация функции boat\_c

#include "BoatC.h"

#include "Combi4.h"

namespace boatfnc

{

bool compv(combi4::accomodation s, const int ming[],

const int maxg[], const int v[]) {

int i = 0;

while (i < s.m && v[s.ntx(i)] <= maxg[i] && v[s.ntx(i)] >= ming[i])i++;

return (i == s.m);

};

int calcc(combi4::accomodation s, const int c[]) {

int rc = 0;

for (int i = 0; i < s.m; i++) rc += c[s.ntx(i)];

return rc;

};

void copycomb(short m, short\* r1, const short\* r2) {

for (int i = 0; i < m; i++) r1[i] = r2[i];

};

}

int boat\_с( // функция возвращает доход от перевозки контейнеров

short m, // [in] количество мест для контейнеров

int minv[], // [in] минимальный вес контейнера на каждом месте

int maxv[], // [in] максимальный вес коннтейнера каждом месте

short n, // [in] всего контейнеров

const int v[],// [in] вес каждого контейнера

const int c[],// [in] доход от перевозки каждого контейнера

short r[] // [out] номера выбранных контейнеров

) {

combi4::accomodation s(n, m);

int rc = 0, i = s.getfirst(), cc = 0;

while (i > 0) {

if (boatfnc::compv(s, minv, maxv, v))

if ((cc = boatfnc::calcc(s, c)) > rc)

{

rc = cc; boatfnc::copycomb(m, r, s.sset);

}

i = s.getnext();

};

return rc;

};

Пример решения задачи об оптимальном размещении контейнеров на судне:

// main (решение задачи о размещении контейнеров)

#include <iostream>

#include <iomanip>

#include "BoatC.h"

#include <tchar.h>

#define NN 8

#define MM 5

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

{

//--------------- Пример решения задачи об оптимальном размещении контейнеров на судне

// вес (у нас 8 контейнеров с весом от 100 до 200 кг, по заданию нужно сгенерировать веса этом диапазоне)

// код для генерации псевдослучайных чисел из 1 лр

int v[NN];

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

v[i] = auxil::iget(100, 200);

// доход (доход для 8 контейнеров от 10 до 100 у.е.)

int c[NN];

// минимальный вес контейнеров для каждого места 50-120 кг

int minv[NN];

// максимальный вес контейнеров для каждого места 150-850 кг

int maxv[NN];

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

v[i] = auxil::iget(100, 200);

c[i] = auxil::iget(10, 100);

minv[i] = auxil::iget(50, 120);

maxv[i] = auxil::iget(150, 850);

}

short r[MM];

int cc = boat\_с(

MM, // [in] количество мест для контейнеров

minv, // [in] минимальный вес контейнера на каждом месте

maxv, // [in] максимальный вес контейнера на каждом месте

NN, // [in] всего контейнеров

v, // [in] вес каждого контейнера

c, // [in] доход от перевозки каждого контейнера

r // [out] номера выбранных контейнеров

);

std::cout << std::endl << "- Задача о размещении контейнеров на судне с условием центровки-";

std::cout << std::endl << "- общее количество контейнеров : " << NN;

std::cout << std::endl << "- количество мест для контейнеров : " << MM;

std::cout << std::endl << "- минимальный вес контейнера : ";

for (int i = 0; i < MM; i++) std::cout << std::setw(3) << minv[i] << " ";

std::cout << std::endl << "- максимальный вес контейнера : ";

for (int i = 0; i < MM; i++) std::cout << std::setw(3) << maxv[i] << " ";

std::cout << std::endl << "- вес контейнеров : ";

for (int i = 0; i < NN; i++) std::cout << std::setw(3) << v[i] << " ";

std::cout << std::endl << "- доход от перевозки : ";

for (int i = 0; i < NN; i++) std::cout << std::setw(3) << c[i] << " ";

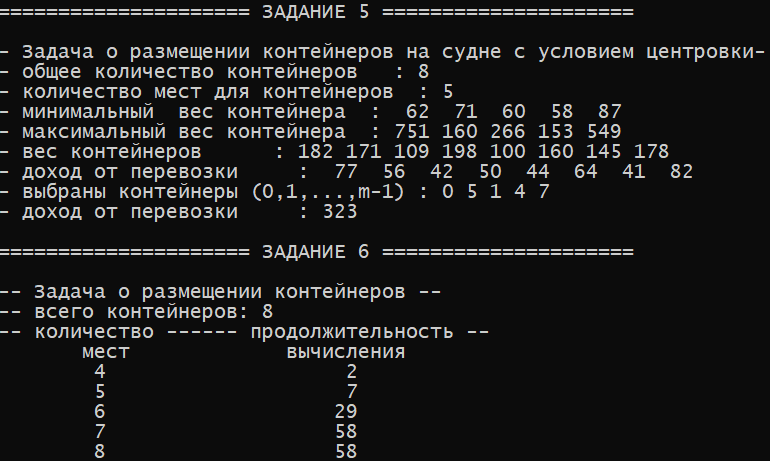
std::cout << std::endl << "- выбраны контейнеры (0,1,...,m-1) : ";

for (int i = 0; i < MM; i++) std::cout << r[i] << " ";

std::cout << std::endl << "- доход от перевозки : " << cc;

std::cout << std::endl << std::endl;

}

****

**Задание 6.** Исследовать зависимость времени вычисления необходимое для решения задачи (в соответствии с вариантом) от размерности задачи:

(Вариант 4, 8, 12, 16) об оптимальной загрузке судна с условием центровки (количество мест на судне для контейнеров 4 – 8);

Оценка продолжительности решения задачи о размещении контейнеров на судне:

// main (решение задачи о размещении контейнеров)

#include <iostream>

#include <iomanip>

#include <time.h>

#include "Auxil.h"

#include "tchar.h"

#include "BoatC.h"

#define SPACE(n) std::setw(n)<<" "

#define NN 8

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

{

//--------------- Оценка продолжительности решения задачи о размещении контейнеров на судне

int v[NN + 1], c[NN + 1], minv[NN + 1], maxv[NN + 1];

short r[NN];

auxil::start();

// генерация значений в массивах весов, доходов, мин. и макс. весов

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

v[i] = auxil::iget(100, 200);

c[i] = auxil::iget(10, 100);

minv[i] = auxil::iget(50, 120);

maxv[i] = auxil::iget(150, 850);

}

std::cout << std::endl << "-- Задача о размещении контейнеров -- ";

std::cout << std::endl << "-- всего контейнеров: " << NN;

std::cout << std::endl << "-- количество ------ продолжительность -- ";

std::cout << std::endl << " мест вычисления ";

clock\_t t1, t2;

// i++ - увеличиваем количество мест

for (int i = 4; i <= NN; i++) {

// замеряем продолжительность

t1 = clock();

boat\_с(i, minv, maxv, NN, v, c, r);

t2 = clock();

std::cout << std::endl << SPACE(7) << std::setw(2) << i

<< SPACE(15) << std::setw(6) << (t2 - t1);

}

}

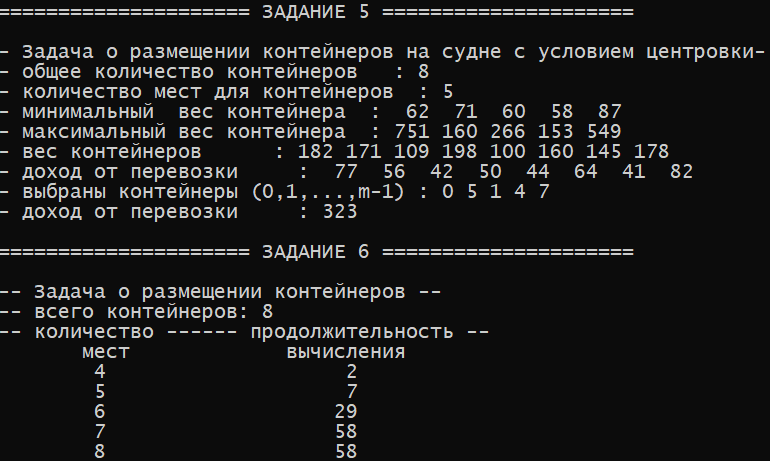


График Excel:

