МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет: Информационных технологий

Кафедра: Программной инженерии

Выполнила: студентка 2 курса 5 группы

специальности ПОИТ Вовна Я. Р.

**Отчёт**

По дисциплине “Математическое программирование”

На тему “Комбинаторные алгоритмы решения оптимизационных задач”

Минск

2024

**Лабораторная работа 2. Комбинаторные алгоритмы решения оптимизационных задач**

**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.

**Ход Работы**

1. **Разработка генератора подмножеств заданного множества**

#pragma once

namespace combi

{

struct subset

{

short n,

sn,

\* sset;

unsigned \_\_int64 mask;

subset(short n = 1);

short getfirst();

short getnext();

short ntx(short i);

unsigned \_\_int64 count();

void reset();

};

};

Листинг 1 – Содержание файла Combi.h

В файле записаны прототипы функций getfirst(), getnext(), ntx(short i), count() и reset(), которые нужны для работы с генератором подмножеств множества.

#include "Combi.h"

#include <algorithm>

namespace combi

{

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);};

};

Листинг 2 ­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­– Содержание файла Combi.cpp

В файле были реализованы нужны функции, которые были определены в Compi.h.

#include <iostream>

#include "Combi.h"

#include <tchar.h>

using namespace std;

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

{

setlocale(LC\_ALL, "rus");

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

cout << "\n - Генератор множества всех подмножеств -";

cout << "\nИсходное множество: ";

cout << "{ ";

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

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

cout << "}\nГенерация всех подмножеств ";

combi::subset s1(sizeof(AA) / 2);

int n = s1.getfirst();

while (n >= 0)

{

cout << "\n{ ";

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

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

cout << "}";

n = s1.getnext();

};

cout << "\nвсего: " << s1.count() << endl;

system("pause");

return 0;

}

Листинг 3 ­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­– Содержание файла main.cpp

В файле main.cpp применены реализованные функции для получения всех подмножеств заданного множества. Результат приведён на рисунке 1.

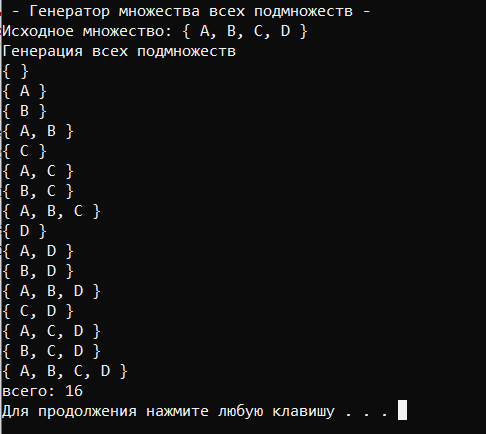


Рис 1 – Результат выполнения программы

1. **Разработка генератора сочетаний**

#pragma once

namespace combi

{

struct xcombination

{

short n,

m,

\* sset;

xcombination(

short n = 1,

short m = 1

);

void reset();

short getfirst();

short getnext();

short ntx(short i);

unsigned \_\_int64 nc;

unsigned \_\_int64 count() const;

};

};

Листинг 1 – Содержание файла Combi.h

В файле записаны прототипы функций getfirst(), getnext(), ntx(short i), count() и reset(), которые нужны для работы с генератором сочетаний.

#include "Combi.h"

#include <algorithm>

namespace combi

{

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;

};

}

Листинг 2 ­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­– Содержание файла Combi.cpp

В файле были реализованы нужны функции, которые были определены в Compi.h.

#include <iostream>

#include "Combi.h"

#include <tchar.h>

using namespace std;

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

{

setlocale(LC\_ALL, "rus");

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

cout << "\n --- Генератор сочетаний ---";

cout << "\nИсходное множество: ";

cout << "{ ";

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

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

cout << "}";

cout << "\nГенерация сочетаний ";

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

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

int n = xc.getfirst();

while (n >= 0)

{

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

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

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

cout << "}";

n = xc.getnext();

};

cout << "\nвсего: " << xc.count() << endl;

system("pause");

return 0;

}

Листинг 3 ­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­– Содержание файла main.cpp

В файле main.cpp применены реализованные функции для получения всех сочетаний элементов множества. Результат приведён на рисунке 1.

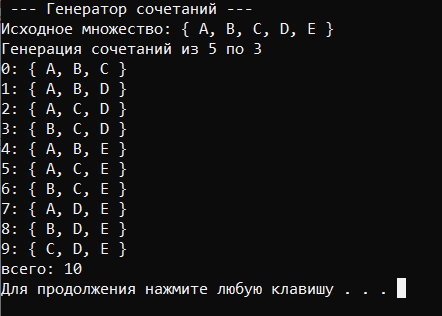


Рис 1 – Результат выполнения программы

1. **Разработка генератора перестановок**

#pragma once

namespace combi

{

struct permutation

{

const static bool L = true;

const static bool R = false;

short n,

\* sset;

bool\* dart;

permutation(short n = 1);

void reset();

\_\_int64 getfirst();

\_\_int64 getnext();

short ntx(short i);

unsigned \_\_int64 np;

unsigned \_\_int64 count() const; };

};

Листинг 1 – Содержание файла Combi.h

В файле записаны прототипы функций getfirst(), getnext(), ntx(short i), count() и reset(), которые нужны для работы с генератором перестановок.

#include "Combi.h"

#include <algorithm>

#define NINF ((short)0x8000)

using std::swap;

namespace combi

{

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) {

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

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); };

};

Листинг 2 ­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­– Содержание файла Combi.cpp

В файле были реализованы нужны функции, которые были определены в Compi.h.

#include <iostream>

#include "Combi.h"

#include <iomanip>

#include <tchar.h>

using namespace std;

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

{

setlocale(LC\_ALL, "rus");

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

cout << "\n --- Генератор перестановок ---";

cout << "\nИсходное множество: ";

cout << "{ ";

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

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

cout << "}";

cout << "\nГенерация перестановок ";

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

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

while (n >= 0)

{

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

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

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

cout << "}";

n = p.getnext();

};

cout << "\nвсего: " << p.count() << endl;

system("pause");

return 0;

}

Листинг 3 ­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­– Содержание файла main.cpp

В файле main.cpp применены реализованные функции для получения всех перестановок элементов множества. Результат приведён на рисунке 1.

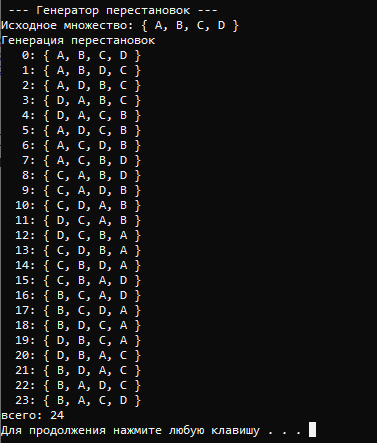


Рис 1 – Результат выполнения программы

1. **Разработка генератора размещений**

#pragma once

namespace combi4

{

struct xcombination

{

short n,

m,

\* sset;

xcombination(

short n = 1,

short m = 1

);

void reset();

short getfirst();

short getnext();

short ntx(short i);

unsigned \_\_int64 nc;

unsigned \_\_int64 count() const;

};

struct permutation

{

const static bool L = true;

const static bool R = false;

short n,

\* sset;

bool\* dart;

permutation(short n = 1);

void reset();

\_\_int64 getfirst();

\_\_int64 getnext();

short ntx(short i);

unsigned \_\_int64 np;

unsigned \_\_int64 count() const;

};

struct accomodation

{

short n,

m,

\* sset;

xcombination\* cgen;

permutation\* pgen;

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

void reset();

short getfirst();

short getnext();

short ntx(short i);

unsigned \_\_int64 na;

unsigned \_\_int64 count() const;

};

}

Листинг 1 – Содержание файла Combi.h

В файле кроме стандартных функций getfirst, getnext и т.д. для генератора размещений сочетаются все функции для работы с генератором сочетаний и перестановок, а так же указатели на вышеуказанные генераторы.

#include "Combi.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()

{

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()

{

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;

};

}

Листинг 2 ­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­– Содержание файла Combi.cpp

В файле были реализованы нужны функции, которые были определены в Compi.h.

#include <iostream>

#include <iomanip>

#include <tchar.h>

#include "Combi.h"

#define N (sizeof(AA)/2)

#define M 3

using namespace std;

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

{

setlocale(LC\_ALL, "rus");

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

cout << "\n --- Генератор размещений ---";

cout << "\nИсходное множество: ";

cout << "{ ";

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

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

cout << "}";

cout << "\nГенерация размещений из " << N << " по " << M;

combi4::accomodation s(N, M);

int n = s.getfirst();

while (n >= 0) {

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

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

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

cout << "}";

n = s.getnext();

};

cout << "\nвсего: " << s.count() << endl;

system("pause");

return 0;

}

Листинг 3 ­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­– Содержание файла main.cpp

В файле main.cpp применены реализованные функции для получения всех размещений элементов множества. Результат приведён на рисунке 1.

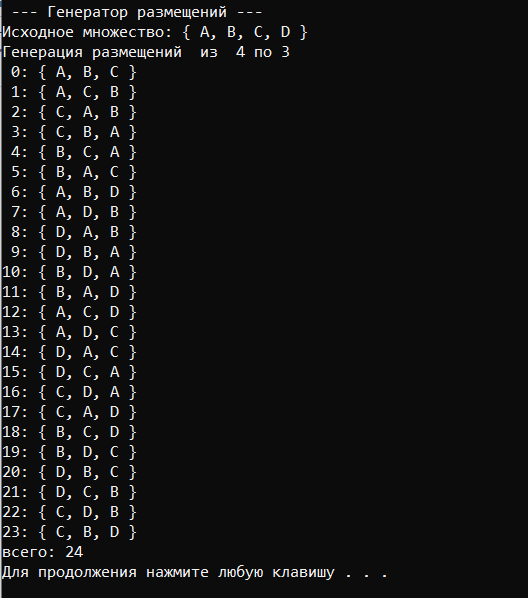


Рис 1 – Результат выполнения программы

1. **Решение задачи о оптимальном размещении контейнеров на судне с**

**помощью генератора размещений**

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

);

Листинг 1 – Содержание файла Combi.h

В файле кроме определения функций для генераторов описана функция boat\_c, которая возвращает максимальных доход от перевозки.

bool compv(combi::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(combi::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(

short m,

int minv[],

int maxv[],

short n,

const int v[],

const int c[],

short r[]

)

{

combi::accomodation s(n, m);

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

while (i > 0)

{

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

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

{

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

}

i = s.getnext();

};

return rc;

}

Листинг 2 ­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­– Содержание файла Combi.cpp

В файле кроме функций для генераторов были реализованы так же дополнительные, которые нужны для решения задачи.

#include <iostream>

#include <iomanip>

#include "Combi.h"

#include <tchar.h>

#include <ctime>

#define MM 3

using namespace std;

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

{

setlocale(LC\_ALL, "rus");

int v[] = { 100, 200, 300, 400 };

int c[] = { 10, 15, 20, 25 };

int minv[] = { 350, 250, 0 };

int maxv[] = { 750, 350, 750 };

short r[MM];

short NN = (sizeof(v) / sizeof(int));

clock\_t t1 = 0, t2 = 0;

t1 = clock();

int cc = combi::boat\_с(MM, minv, maxv, NN, v, c, r);

t2 = clock();

cout << "\n- Задача о размещении контейнеров на судне -";

cout << "\n- общее количество контейнеров : " << NN;

cout << "\n- количество мест для контейнеров : " << MM;

cout << "\n- минимальный вес контейнера : ";

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

cout << "\n- максимальный вес контейнера : ";

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

cout << "\n- вес контейнеров : ";

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

cout << "\n- доход от перевозки : ";

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

cout << "\n- выбраны контейнеры (0,1,...,m-1) : ";

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

cout << "\n- доход от перевозки : " << cc << endl;

cout << "Время" << (t2 - t1) << endl;

system("pause");

return 0;

}

Листинг 3 ­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­– Содержание файла main.cpp

В файле main.cpp применены реализованные функции для получения максимального дохода согласно условию задания. Результат приведён на рисунке.

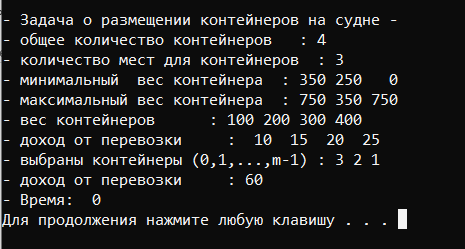


Рис 1 – Результат выполнения программы

1. **Исследование зависимости времени вычисления от количества**

**контейнеров в задаче про оптимальную загрузку судна**

Линия тренда графика испытаний показала, что зависимость количества испытаний от времени выполнения экспоненциальная. Результаты измерений приведены на рисунке 1.

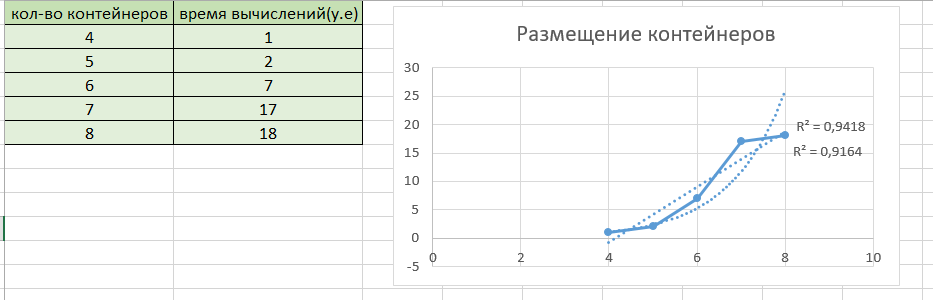


Рис 1 – Результаты измерений и график

**Вывод:** были разработаны генераторы подмножеств, перестановок, сочетаний и размещений на С++. Применены разработанные генераторы для решения задачи об оптимальной загрузке судна с центровкой. Исследована зависимость количества контейнеров от времени выполнения программы.