Учреждение образования

«Белорусский государственный технологический университет»

**Лабораторная работа №2**

Комбинаторные алгоритмы решения оптимизационных задач

Выполнил:

Студент 2 курса 4 группы ФИТ

Сосновец Мария Игоревна

2024 г.

**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**

**Задание 1.** Разобрать и разработать генератор подмножеств заданного множества.

Был разобран генератор подмножества заданного множества из методического пособия и представлен код файла Lab\_02.cpp, Combi.cpp представлены в листингах 1.1, 1.2 соответственно.

Листинг 1.1 – Код файла Lab\_02.cpp

|  |
| --- |
| #include <iostream>  #include "Combi.h"  int main()  {  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 ";  combi::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;  } |

Листинг 1.2 – Код файла Combi.cpp

|  |
| --- |
| #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);  };  }; |

Так же был реализован заголовочный файл для генератора подмножества заданного множества, который представлен в листинге 1.3.

Листинг 1.3 – Код заголовочного файла Combi.h

|  |
| --- |
| #pragma once  namespace combi  {  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  };  }; |

На рисунке 1 представлен вывод в консоль генератора подмножества заданного множества.

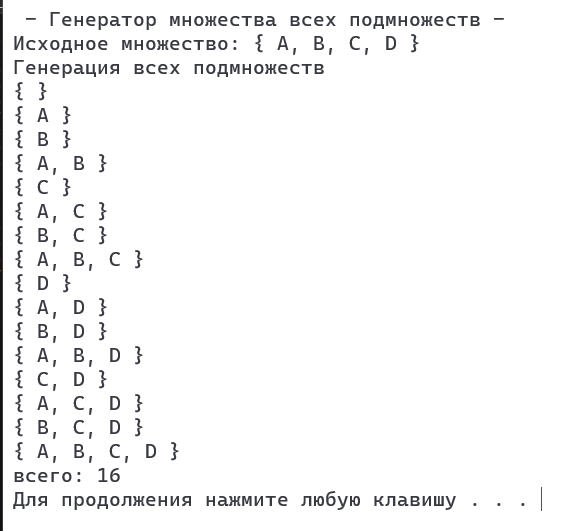


Рисунок 1- Результат работы программы

**Задание 2.** Разобрать и разработать генератор сочетаний.

Был разобран генератор сочетаний из методического пособия и представлен код файла Main2.cpp, Combi.cpp представлены в листингах 2.1, 2.2 соответственно.

Листинг 2.1 – Код файла Main2.cpp

|  |
| --- |
| #include <iostream>  #include "Combi.h"  int main()  {  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;  } |

Листинг 2.2 – Код файла Combi.cpp

|  |
| --- |
| #include "Combi.h"  #include <algorithm>  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;  };  }; |

Так же был реализован заголовочный файл для генератора сочетаний, который представлен в листинге 2.3.

Листинг 2.3 – Код файла Combi.h

|  |
| --- |
| #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  };  } |

На рисунке 2 представлен вывод в консоль генератора сочетаний.

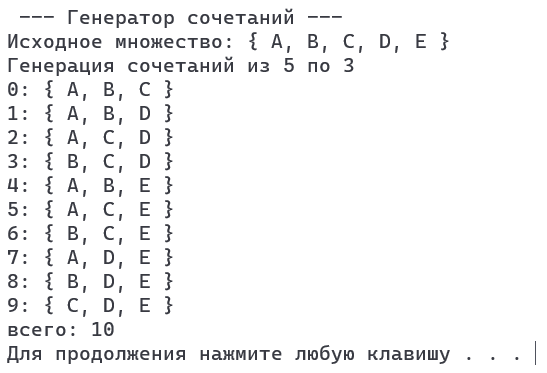


Рисунок 2- Результат работы программы

**Задание 3.** Разобрать и разработать генератор перестановок.

Был разобран генератор перестановок из методического пособия и представлен код реализации заголовочного файла Combi.h представлен в листинге 3.1.

Листинг 3.1 – Код файла Combi.h

|  |
| --- |
| #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  };  }; |

В листинге 3.2 представлен шаблон структуры генератора перестановок, а в листинге 3.3 реализация генератора.

Листинг 3.2 – Код файла Combi.cpp, который представляет собой шаблон

|  |
| --- |
| #include "Combi.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); };  } |

Листинг 3.3 – Код файла Main3.cpp

|  |
| --- |
| #include <iostream>  #include "Combi.h"  #include <iomanip>  int main()  {  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;  } |

На рисунке 3 представлен вывод в консоль генератора перестановок.

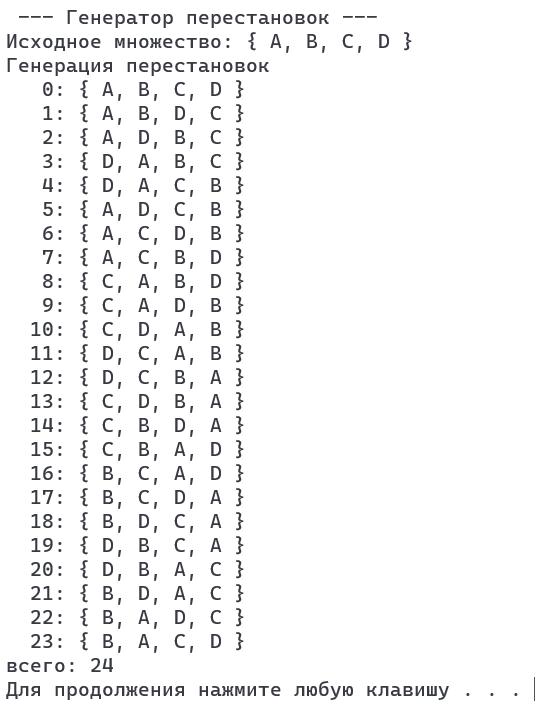


Рисунок 3- Результат работы программы

**Задание 4.** Разобрать и разработать генератор размещений.

Был разобран генератор размещений из методического пособия и представлен код реализации заголовочного файла Combi.h представлен в листинге 4.1.

Листинг 4.1 – Код файла Combi.h

|  |
| --- |
| // Combi4.h  #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  };  } |

В листинге 4.2 представлен шаблон структуры генератора размещений, а в листинге 4.3 реализация генератора.

Листинг 4.2 – Код файла Combi.cpp

|  |
| --- |
| #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() // 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;  };  } |

Листинг 4.3 – Код файла Main.cpp

|  |
| --- |
| #include <iostream>  #include <iomanip>  #include "Combi.h"  #define N (sizeof(AA)/2)  #define M 3  int main()  {  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;  } |

На рисунке 4 представлен вывод в консоль генератора размещений.

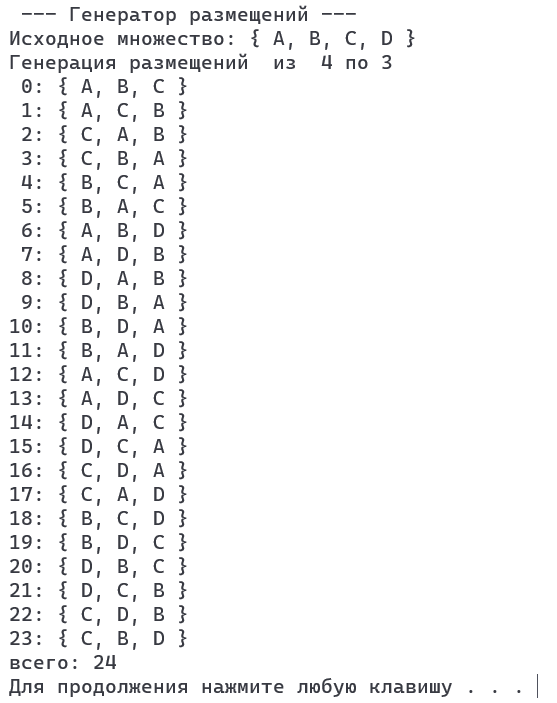


Рисунок 4 - Результат работы программы

**Задание 5.** Решить в соответствии с вариантом задачу и результат занести в отчет (Вариант распределяется по списку).

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);

Задача была разобрана и решена с помощью генератора размещений, реализация представлена в листинге 5.1.



Листинг 5.1 – Код файла Lab\_02(task 5)

|  |
| --- |
| #include "pch.h"  #include <iostream>  #include <ctime>  #include <locale>  #include <iomanip>  #include "Combi.h"  #include "Boat.h"  #include "Auxil.h"  #define N (sizeof(AA)/2)  #define M 3  #define NN 8  #define MM 5  int main()  {  setlocale(LC\_ALL, "rus");  clock\_t t1 = 0, t2 = 0;  int\* v = new int[NN]; // вес  int\* c = new int[NN]; // доход  int\* minv = new int[MM]; // минимальный вес  int\* maxv = new int[NN]; // максимальный вес  short r[MM];  for (int i = 0; i < NN; i++)  {  v[i] = 100 + rand() % 200;  c[i] = 10 + rand() % 100;  }  for (int j = 0; j < MM; j++)  {  //c  minv[j] = 50 + rand() % 120;  maxv[j] = 150 + rand() % 850;  }  auxil::start(); // старт генерации  t1 = clock();  int cc = boat\_с(  MM, // [in] количество мест для контейнеров  minv, // [in] максимальный вес контейнера на каждом месте  maxv, // [in] минимальный 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  );  t2 = clock();  std::cout << std::endl << "- Задача о размещении контейнеров на судне -";  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::endl;  std::cout << "время выполнения - " << ((double)(t2 - t1)) / ((double)CLOCKS\_PER\_SEC) << "сек" << std::endl;  system("pause");  return 0;  } |

На рисунке 5 представлен вывод в консоли задачи под номером 5.

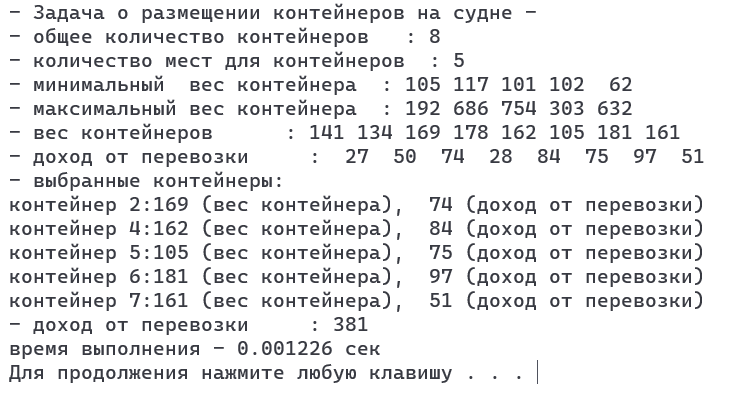


Рисунок 5 - Результат решения задачи

**Задание 6.** Исследовать зависимость времени вычисления необходимое для решения задачи (в соответствии с вариантом) от размерности задачи и результат в виде графика с небольшим пояснением занести в отчет:

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

Реализация задачи об оптимальной загрузке судна с условием центровки представлена в листинге 6.1.

Листинг 6.1 – Код файла Lab\_02(task 6)

|  |
| --- |
| #include <iostream>  #include <iomanip>  #include "BoatC.h"  #include <tchar.h>  #include "Lab\_02(task 6).h"  #define NN 8  #define MM 5  #define SPACE(n) setw(n)<<" "  using std::cout;  using std::endl;  using std::setw;  void fillArrayWithRandomValues(int\* weights, int sizeOfArray, int min, int max) {  std::srand(static\_cast<unsigned int>(std::time(nullptr)));  for (int i = 0; i < sizeOfArray; i++)  {  weights[i] = min + (rand() % (max - min + 1));  }  }  int \_tmain(int argc, \_TCHAR\* argv[])  {  setlocale(LC\_ALL, "rus");  int v[NN];// вес  int c[NN]; // доход  int minv[MM];  int maxv[MM];  fillArrayWithRandomValues(v, NN, 100, 200);  fillArrayWithRandomValues(c, NN, 10, 100);  fillArrayWithRandomValues(minv, MM, 50, 120);  fillArrayWithRandomValues(maxv, MM, 150, 850);  short r[MM];  int cc = boat\_с(  MM, // [in] количество мест для контейнеров  minv, // [in] максимальный вес контейнера на каждом месте  maxv, // [in] минимальный 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  );  cout << endl << "-- Задача о размещении контейнеров -- ";  cout << endl << "-- всего контейнеров: " << NN;  cout << endl << "-- количество ------ продолжительность -- ";  cout << endl << " мест вычисления ";  clock\_t t1, t2;  for (int i = 4; i <= NN; i++) {  t1 = clock();  boat\_с(i, minv, maxv, NN, v, c, r);  t2 = clock();  cout << endl << SPACE(7) << setw(2) << i  << SPACE(15) << setw(6) << (t2 - t1);  }  cout << endl; system("pause");  return 0;  } |

Результат работы программы представлен на рисунке 6. Исследование зависимости представлено на рисунке 7.

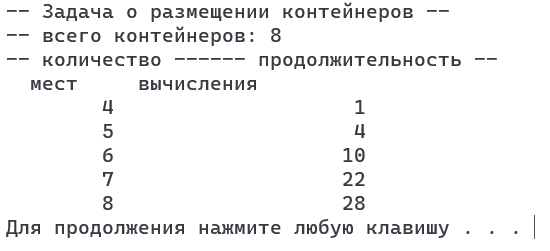


Рисунок 6 - Результат решения задачи

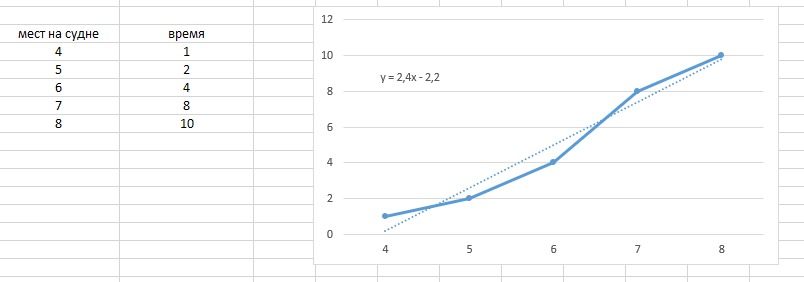


Рисунок 7 - Результат исследования зависимости

**Вывод:** в лабораторной работе №2 были приобретены навыки разработки генераторов подмножеств, перестановок, сочетаний и размещений на С++; так же научились применять разработанные генераторы для решения задач о рюкзаке (упрощенную, коммивояжера, об оптимальной загрузке судна и об оптимальной загрузке судна с центровкой.

Для каждой задачи был предложен алгоритм решения, основанный на генерации соответствующих комбинаций, и выполнена их реализация на языке C++. Для исследования зависимости времени выполнения от размерности задачи был проведен эксперимент с генерацией случайных данных различной размерности и измерением времени выполнения алгоритмов.

На основе полученных результатов были построены графики зависимости времени выполнения от размерности задачи для каждой задачи, и проанализированы полученные результаты. Результаты работы были представлены в отчете в виде описания алгоритмов, примеров работы программ, графиков зависимости времени выполнения от размерности задачи, а также выводов о возможностях и ограничениях применения генераторов комбинаций для решения задач оптимизации.