Учреждение образования

«Белорусский государственный технологический университет»

**Лабораторная работа №1**

Вспомогательные функции

Выполнил:

Студент 2 курса 4 группы ФИТ

Кивлинас Олег Леонидович

2024 г.

**Задание 1**

Разработайте три функции (start, dget и iget), используя следующие спецификации (рисунок 1):

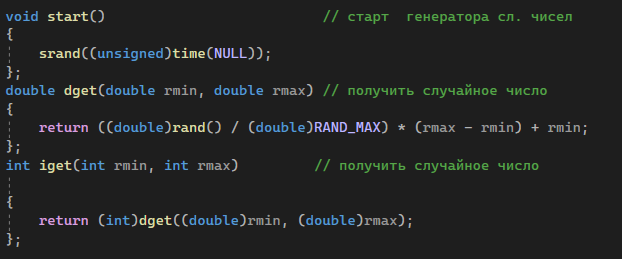


Рисунок 1 – Реализация функций

**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.

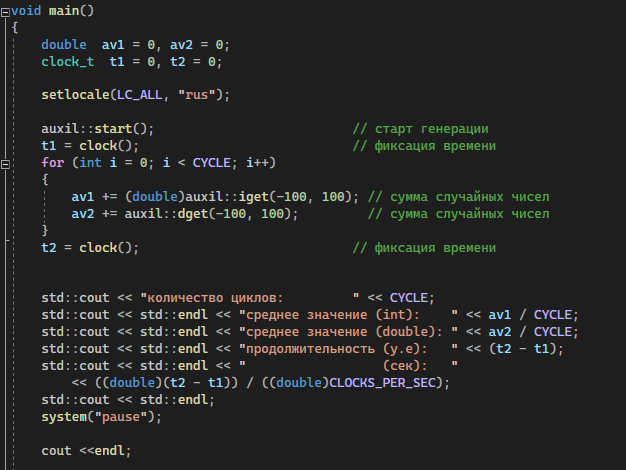


Рисунок 2 – Вызов функции и измерение времени выполнения

**Задание 3**

Проведите необходимые эксперименты (разработать кодом) и постройте график зависимости (Excel) продолжительности процесса вычисления от количества циклов в примере 2.

Примечание: продолжительность вычисления измерять в условных единицах процессорного времени (функция steady\_clock). График представлен на рисунке 3

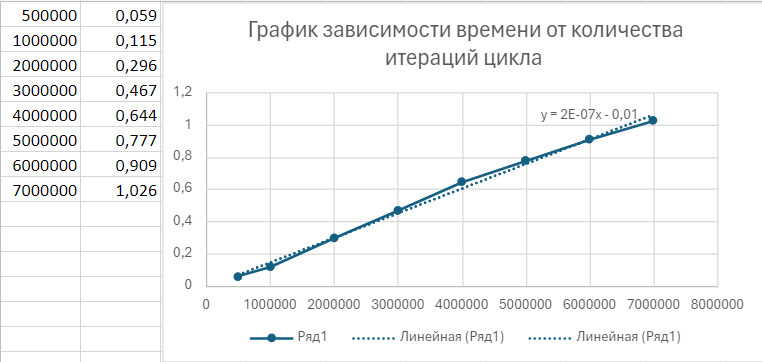


Рисунок 3 – график зависимости

Реализация Факториала в коде демонстрируется на рис. 4

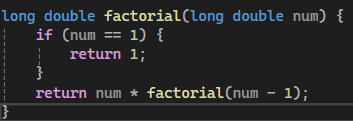


Рисунок 4 – Функция для нахождения факториала

Вызов и подсчет времени для реализации факториала представлен на рисунке 5:

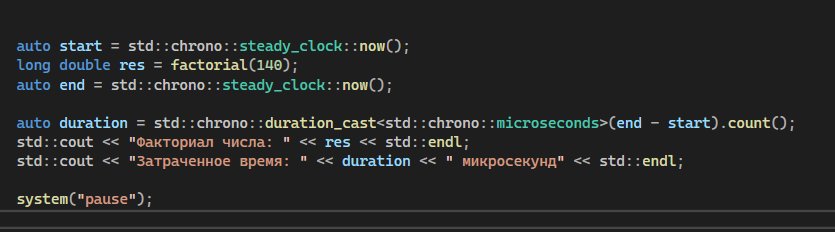


Рисунок 5 – Вызов функции для подсчёта факториала

График зависимости:

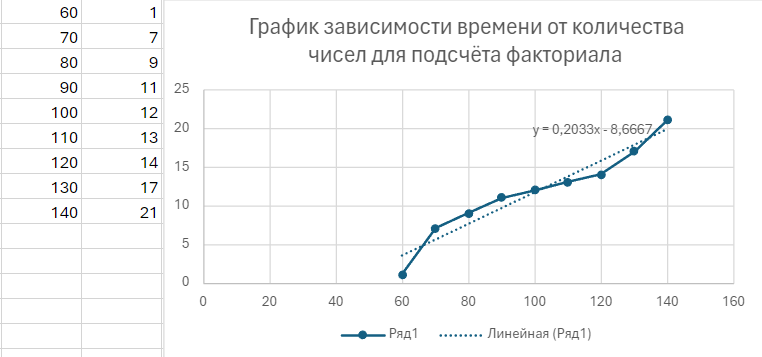


Рисунок 6 – График зависимости

**Вывод**: приобрел навыки составления и отладки программ с использованием пользовательских функций для замера продолжительности процесса вычисления по полученным измерениям, а так же выяснил, что время выполнения программы линейно зависит от количества циклов и значения факториала.

# 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.

**Задание 1:** Разработать генератор подмножеств заданного множества. Генератор подмножеств заданного множества будет на листинге 2.1.

|  |
| --- |
| #include <iostream>  #include "Comb.h"  #include <tchar.h>  using std::cout;  using std::endl;  int \_tmain(int argc, \_TCHAR\* argv[])  {  setlocale(LC\_ALL, "rus");  char AA[][2] = {"A", "B", "C", "D", "E"};  cout << "Генератор множества всех подмножеств\n";  cout << endl << "Исходное множество: ";  cout << "{ ";  for (int i = 0; i < sizeof(AA) / 2; i++)  cout << AA[i] << ((i < sizeof(AA) / 2 - 1) ? ", " : " ");  cout << "}\n\nИтоговая генерация всех подмножеств исходного множества:";  comb::subset s1(sizeof(AA) / 2); // создание генератора  int n = s1.getfirst(); // первое (пустое) подмножество  while (n >= 0) // пока есть подмножества  {  cout << endl << "{ ";  for (int i = 0; i < n; i++)  cout << AA[s1.ntx(i)] << ((i < n - 1) ? ", " : " ");  cout << "}";  n = s1.getnext(); // cледующее подмножество  };  cout << endl << "Всего множеств: " << s1.count() << endl;  system("pause");  return 0;  } |

Листинг 2.1 – Генератор подмножеств

Результат выполнения генерации представлен на рисунке 2.1.



Рисунок 2.1 – Генератор подмножеств заданного множества

Алгоритм представлен на рисунке 2.2.



Рисунок 2.2 – Генератор подмножеств заданного множества

**Задание 2:** Разработать генератор сочетаний. Код будет представлен на листинге 2.2.

|  |
| --- |
| int \_tmain(int argc, \_TCHAR\* argv[])  {  setlocale(LC\_ALL, "rus");  char AA[][2] = { "A", "B", "C", "D", "E"};  cout << "Генератор сочетаний\n";  cout << endl << "Исходное множество: ";  cout << "{ ";  for (int i = 0; i < sizeof(AA) / 2; i++)  cout << AA[i] << ((i < sizeof(AA) / 2 - 1) ? ", " : " ");  cout << "}\n";  cout << endl << "Генерация сочетаний ";  comb::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 << endl << "Всего сочетаний: " << xc.count() << endl;  system("pause");  return 0;  } |

Листинг 2.2 – Код генератора сочетаний

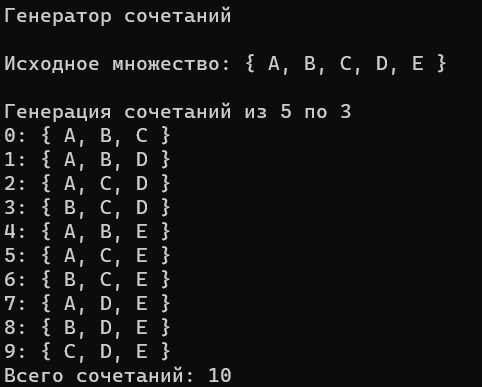


Рисунок 2.3 –Результат генератора сочетаний

Алгоритм будет представлен на рисунке 2.4

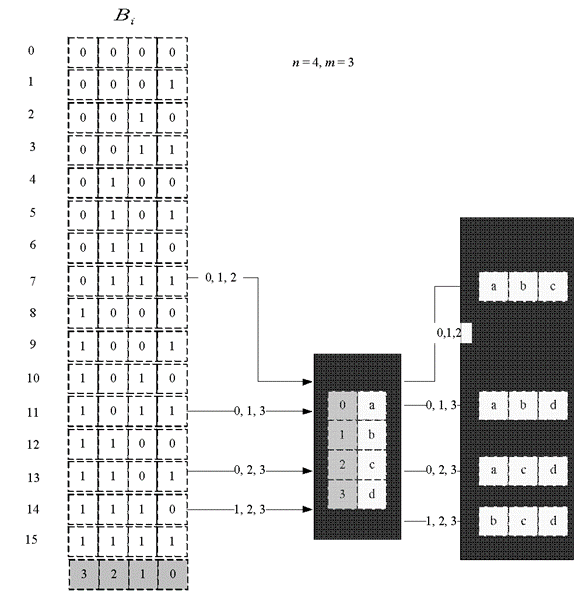


Рисунок 2.4 – Алгоритм генератора сочетаний

**Задание 3:** Разработать генератор перестановок. Код будет представлен на листинге 2.3.

|  |
| --- |
| int \_tmain(int argc, \_TCHAR\* argv[])  {  setlocale(LC\_ALL, "rus");  char AA[][2] = { "A", "B", "C", "D" };  cout << "Генератор перестановок\n";  cout << endl << "Исходное множество: {";  for (int i = 0; i < sizeof(AA) / 2; i++)  cout << AA[i] << ((i < sizeof(AA) / 2 - 1) ? ", " : " ");  cout << "}";  cout << endl << "\nГенерация перестановок ";  comb::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 << endl << "Всего перестановок: " << p.count() << endl;  system("pause");  return 0;} |

Листинг 2.3 – Код генератора перестановок

Результат выполнения представлен на рисунке 2.5.

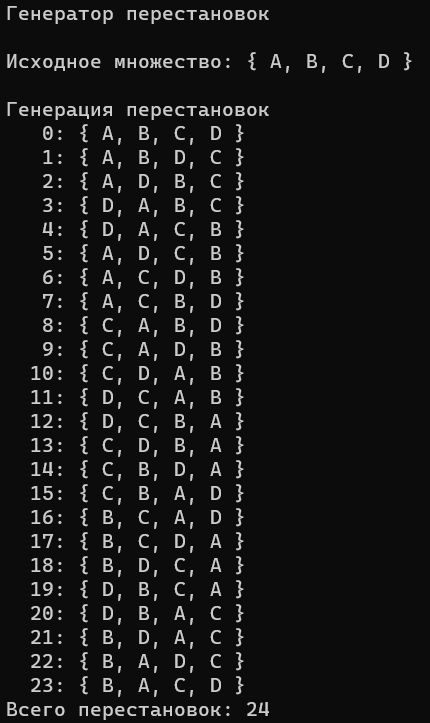


Рисунок 2.5 –Результат генератора перестановок



Рисунок 2.6 – Генератор перестановок

**Задание 4:** Разработать генератор размещений. Код будет представлен в листинге 2.4.

|  |
| --- |
| int \_tmain(int argc, \_TCHAR\* argv[])  {  setlocale(LC\_ALL, "rus");  char AA[][2] = { "A", "B", "C", "D" };  cout << "Генератор размещений\n";  cout << endl << "Исходное множество: ";  cout << "{ ";  for (int i = 0; i < N; i++)  cout << AA[i] << ((i < N - 1) ? ", " : " ");  cout << "}";  cout << endl << "\nГенерация размещений из " << N << " по " << M;  comb::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 << endl << "Всего размещений: " << s.count() << endl;  system("pause");  return 0;  } |

Листинг 2.4 – Реализация генератора размещений

Результат выполнения будет представлен на рисунке 2.7.

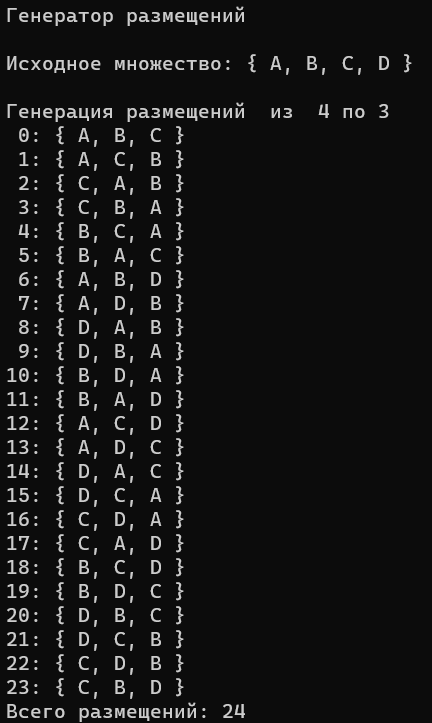


Рисунок 2.7 –Результат генератора размещений

Алгоритм будет представлен на рисунке 2.8.



Рисунок 2.8– Генератор размещений

**Задание 5:** (2, 6, 10, 14) упрощенную о рюкзаке (веса предметов и их стоимость сгенерировать случайным образом: вместимость рюкзака 300 кг, веса предметов 10 – 300 кг, стоимость предметов 5 – 55 у.е.; количество предметов – 18 шт.). Код этого задания будет представлен на листинге 2.5.

|  |
| --- |
| int \_tmain(int argc, \_TCHAR\* argv[])  {  setlocale(LC\_ALL, "rus");  int V = 300, // вместимость рюкзака  v[NN], // размер предмета каждого типа  c[NN]; // стоимость предмета каждого типа  short m[NN]; // количество предметов каждого типа {0,1}  RandomFilling(v, NN, 10, 300); // заполнение массива с размерами каждого предмета  RandomFilling(c, NN, 5, 55); // заполнение массива со стоимостями каждого предмета  int maxcc = knapsack\_s(  V, // [in] вместимость рюкзака  NN, // [in] количество типов предметов  v, // [in] размер предмета каждого типа  c, // [in] стоимость предмета каждого типа  m // [out] количество предметов каждого типа  );  std::cout << std::endl << "-------- Задача о рюкзаке ---------";  std::cout << std::endl << "- количество предметов : " << NN;  std::cout << std::endl << "- вместимость рюкзака : " << V;  std::cout << std::endl << "- номер : вес : стоимость : стоимость предмета";  for (int i = 0; i < NN; i++) {  std::cout << std::endl << "- " << i + 1 << " : " << v[i] << " : " << c[i] << " : " << v[i] \* c[i];  }  std::cout << std::endl << "- оптимальная стоимость рюкзака: " << maxcc;  std::cout << std::endl << "- вес рюкзака: ";  int s = 0; for (int i = 0; i < NN; i++) s += m[i] \* v[i];  std::cout << s;  std::cout << std::endl << "- выбраны предметы: ";  for (int i = 0; i < NN; i++) std::cout << " " << m[i];  std::cout << std::endl << std::endl;  return 0;  } |

Листинг 2.5 – Реализация 5-го задания

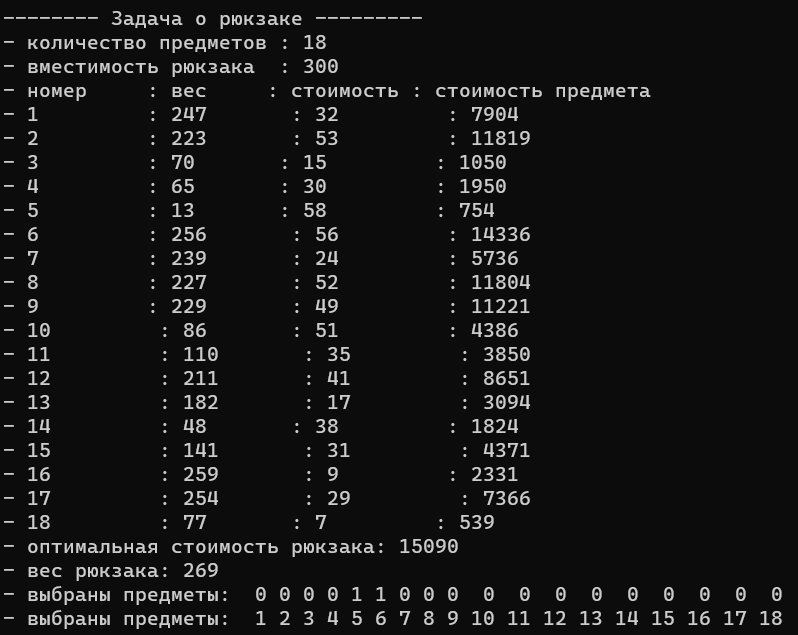


Рисунок 2.9 –Результат выполнения 5-го задания

**Задание 6: (**2, 6, 10, 14) упрощенную о рюкзаке (количество предметов 12 – 20 шт.). Код 6-го задания представлен в листинге 2.6.

|  |
| --- |
| int \_tmain(int argc, \_TCHAR\* argv[])  {  setlocale(LC\_ALL, "rus");  int V = 300, // вместимость рюкзака  v[NN], // размер предмета каждого типа  c[NN]; // стоимость предмета каждого типа  short m[NN]; // количество предметов каждого типа {0,1}  clock\_t t1, t2; // таймер  for (int i = 12; i <= NN; i++)  {  t1 = clock();  RandomFilling(v, i, 10, 300); // заполнение массива с размерами каждого предмета  RandomFilling(c, i, 5, 55); // заполнение массива со стоимостями каждого предмета  int maxcc = knapsack\_s(  V, // [in] вместимость рюкзака  i, // [in] количество типов предметов  v, // [in] размер предмета каждого типа  c, // [in] стоимость предмета каждого типа  m // [out] количество предметов каждого типа  );  t2 = clock();  std::cout << t2 - t1 << std::endl;  }  std::cout << std::endl << "-------- Задача о рюкзаке ---------";  std::cout << std::endl << "- количество предметов : " << NN;  std::cout << std::endl << "- вместимость рюкзака : " << V;  std::cout << std::endl << "- размеры предметов : ";  for (int i = 0; i < NN; i++) std::cout << v[i] << " ";  std::cout << std::endl << "- стоимости предмета : ";  for (int i = 0; i < NN; i++) std::cout << c[i] << " ";  std::cout << std::endl << "- стоимости предметов : ";  for (int i = 0; i < NN; i++) std::cout << v[i] \* c[i] << " ";  std::cout << std::endl << "- вес рюкзака: ";  int s = 0; for (int i = 0; i < NN; i++) s += m[i] \* v[i];  std::cout << s;  std::cout << std::endl << "- выбраны предметы: ";  for (int i = 0; i < NN; i++) std::cout << " " << m[i];  std::cout << std::endl << std::endl;  return 0;  } |

Листинг 2.6 – Реализация 6-го задания

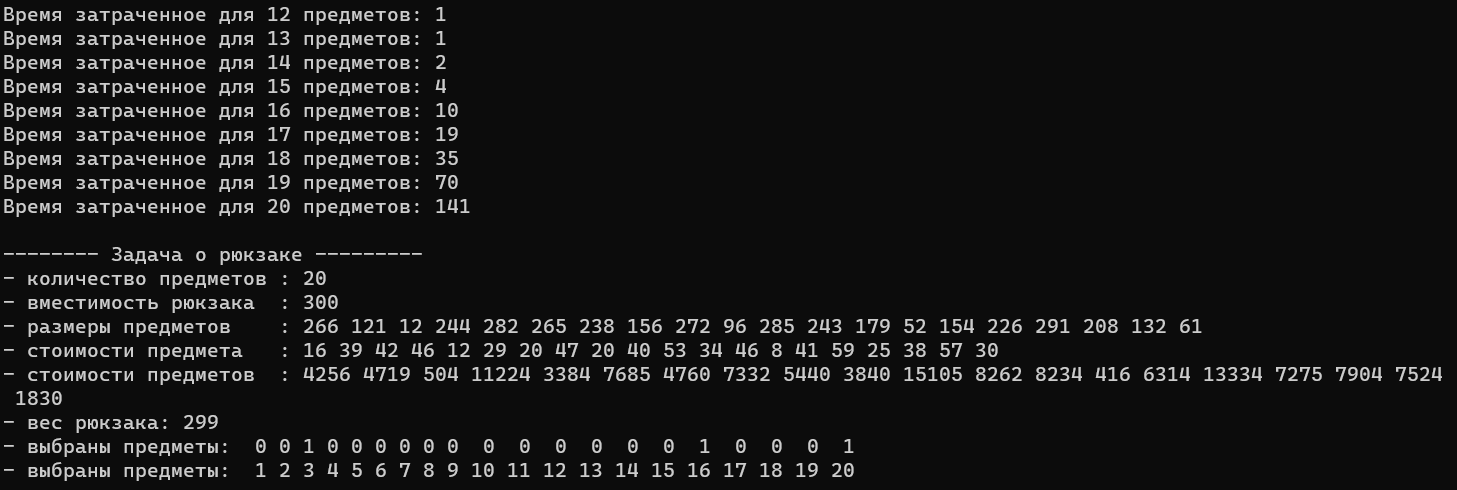


Рисунок 2.11 – Результат выполнения 6-го задания



Рисунок 2.12 – График зависимости времени вычисления от количества предметов



Рисунок 2.18– Схема решения задачи о рюкзаке

**Вывод:** научился разрабатывать генераторы подмножеств, перестановок, сочетаний и размещений на ЯП С++; разработано решение упрощённой задачи о рюкзаке; на основе данных, полученных с графика видно, что график возрастает по экспоненциальной функции

# Лабораторная работа 3

**Метод ветвей и границ. Задача коммивояжера и методы её решения.**

**Цель работы:** освоить общие принципы решения задач методом ветвей и границ, решить задачу о коммивояжере данным методом, сравнить полученное решение задачи с комбинаторным методом перестановок.

**Примечание:** Задания и вопросы со знаком (\*), выполняются в необязательном порядке, но их выполнение поощряется.

**Задание 1:** Сформулировать условие задачи коммивояжера с параметром. Для этого:

* принять элементы матрицы расстояний равными:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Город** | **1** | **2** | **3** | **4** | **5** |
| **1** |  | 2 \* n | 21 + n |  | n |
| **2** | n |  | 15 + n | 68 - n | 84 - n |
| **3** | 2 + n | 3 \* n |  | 86 | 49 + n |
| **4** | 17 + n | 58 - n | 4 \* n |  | 3 \* n |
| **5** | 93 - n | 66 + n | 52 | 13 + n |  |

где *n* – номер варианта или номер по журналу (вариант 6, соответственно n=6)

Задачу следует решить с использованием метода ветвей и границ. Исходная матрица после подстановки n, представлена на рисунке 3.1

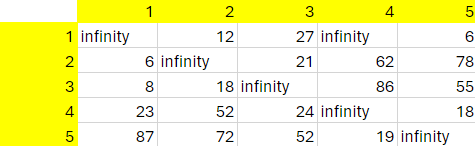


Рисунок 3.1 – Исходная матрица, после подстановки n

Найдем минимальные значения в каждой из строк, и проведем редукцию строк, результат представлен на рисунке 3.2.

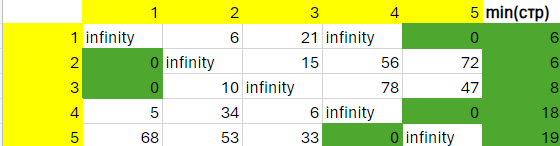


Рисунок 3.2 – Результат

Проделаем поиск минимальных значений, для столбцов, и проведём редукцию столбцов, результат выполнения представлен на рисунке 3.3.

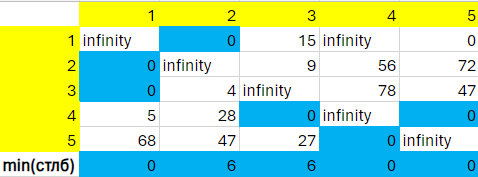


Рисунок 3.3 – Результат

Проделаем поиск локальной нижней границы , где di и dj, это константы приведения, т.е. суммы максимумов и минимумов.

H0 = (6+6+8+18+19)+(0+6+6+0+0) = 69

После, произведём вычисление оценок нулевых клеток. Т.е. для каждой клетки с нулевым значением почитаем сумму минимумов в строке и в столбце, и найдем максимальную оценку. Результат представлен на рисунке 3.4.

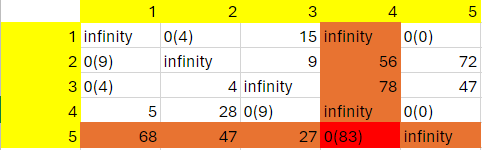


Рисунок 3.4 – Результат

Далее произведем вычисление нижней границы первой ветви (рис 3.5)

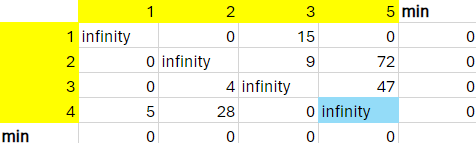
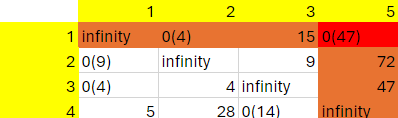


Рисунок 3.5 – Результат

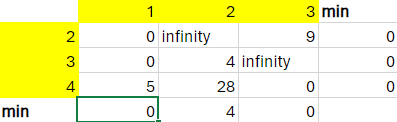
Значение локальной нижней границы: H1=69+(0+0+0+0)+(0+0+0+0)

Далее произведем вычисление нижней границы второй ветви, то есть сумма предыдущей локальной нижней границы и максимальной оценки. Вычислим: H1\*=69+83=152

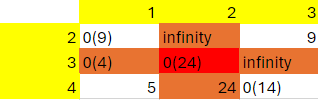
А далее идем по точно такому же алгоритму, пока не найдём полный маршрут



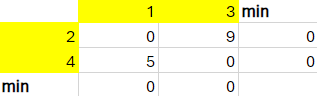
H2\*=69+47=117



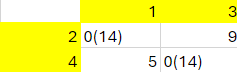
H2=69+4=73



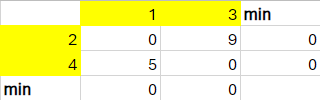
H3\*=73+24=97



H3=73+0=73



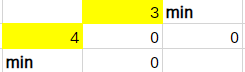
H4\*=73+14=87



H4=73+0=73



H5\*=73+inf=inf



H5=73+0=73

Длина итогового оптимального маршрута равна 73.

Сам маршрут: 1->5->4->3->2->1

Ниже представлен граф обхода исходной матрицы, с нахождением оптимального пути.

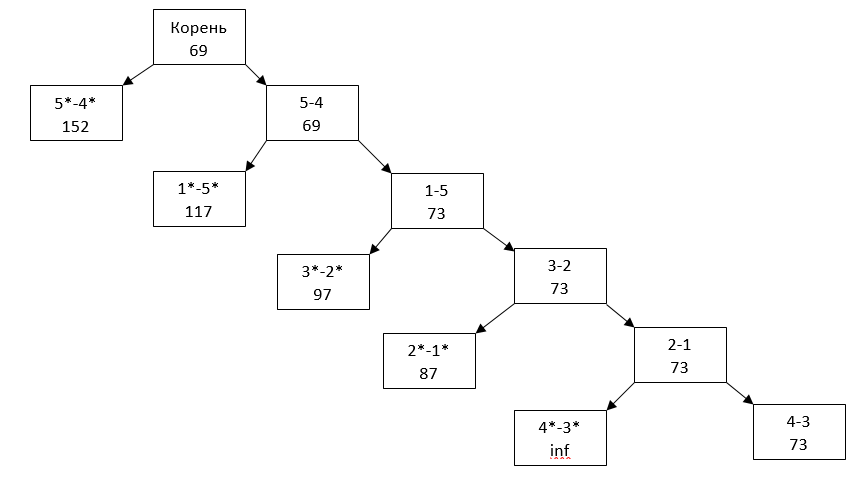


Рисунок 3.6 – Граф обхода

**Задание 3.** Проверить полученное решение при помощи генератора перестановок и включить копию экрана с решением в отчет.

Результат выполнения программы представлен на рисунке 3.7.

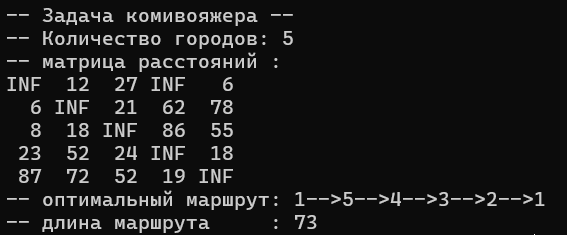


Рисунок 3.7 – Результат

**Вывод**: После проверки пошагового решения из задания 1 и результата генерации кода из задания 3. Выявлено абсолютное совпадение, как в длине, так и в самом маршруте.