Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

**Отчёт по лабораторным работам предмета «Математическое программирование»**

Студент: Кальчевский Д.А

ФИТ 2 курс 3 группа

Вариант 3

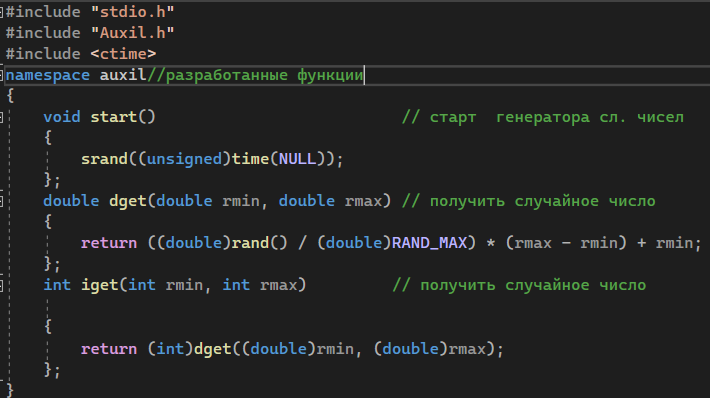
Минск 2023

**Лабораторная работа № 1**

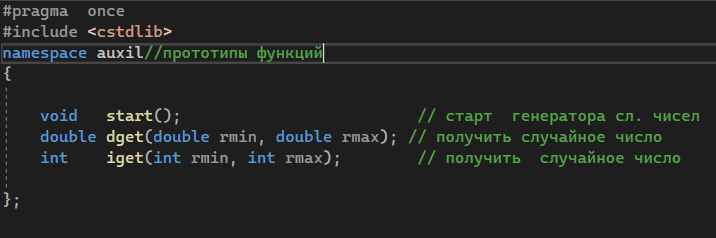
**ЦЕЛЬ РАБОТЫ:** приобретение навыков составления и отладки программ с использованием пользовательских функций для замера продолжительности процесса вычисления.

Задание №1

**Исходный файл Auxil.cpp:**



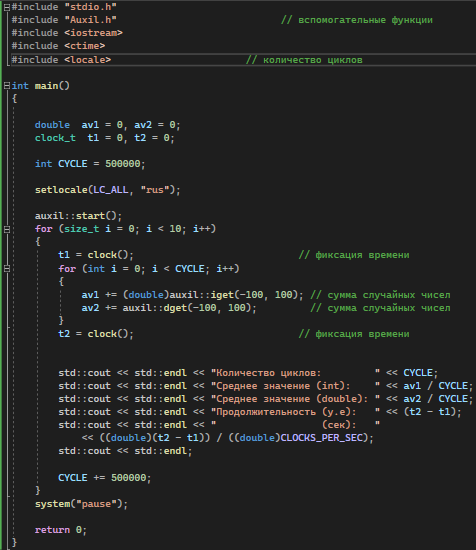
**Заголовочный файл Auhil.h:**



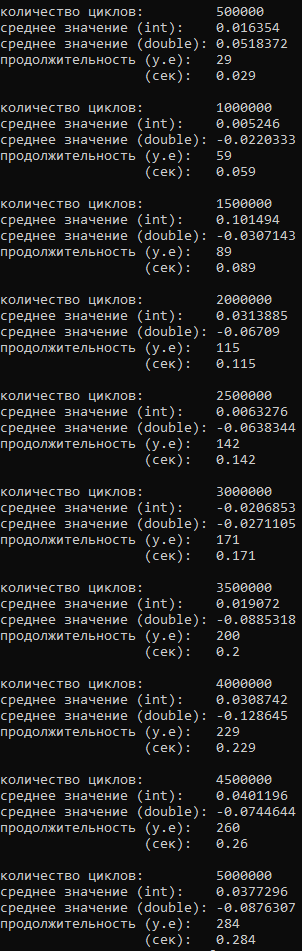
Задание №2

**Lab2.cpp:**

**Замер продолжительности процесса вычисления:**

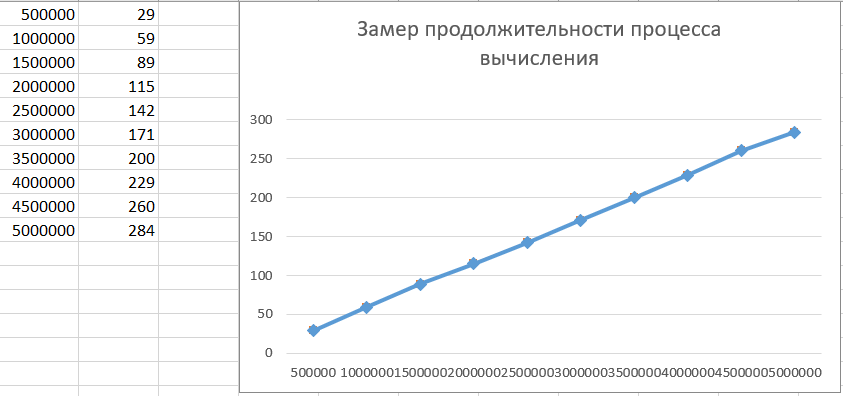


**Консоль программы:**

****

В результате 10 итераций мы получили статистические данные о количестве циклов и соответствующее им время вычисления в (с.).

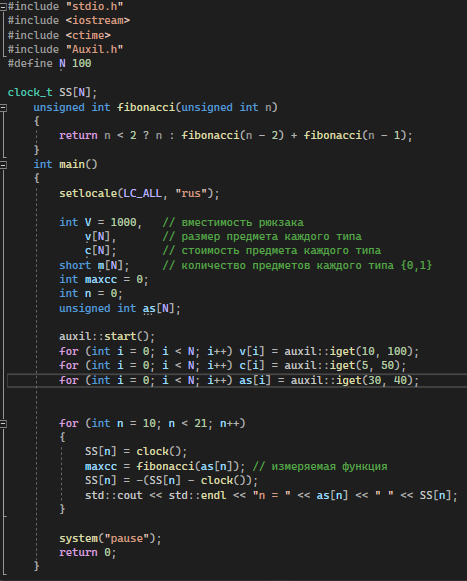
**График:**

****

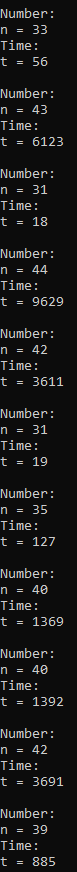
На основе результирующего графика можно определить, что с увеличением количества циклов, время вычисления изменяется, образуя почти линейную зависимость.

Задание №3

**Рекурсивный алгоритм числа ряда Фибоначчи:**



**Консоль алгоритма числа ряда Фибоначчи:**

****

**График зависимости продолжительности процесса:**

****

На основе результирующего графика мы можем определить, что данный алгоритм при увеличении числа пропорционально гиперболической форме увеличивает количество затраченного времени.

**Заключение:**

На основе проделанной лабораторной работы мы можем судить о том, что алгоритм из задания 2 является примером хорошего алгоритма с высокой производительностью.

Алгоритм задания 3 не является оптимизированным и при n-больших числах, время, затраченное на его выполнения, будет неоценимо велико.

**Лабораторная работа № 2**

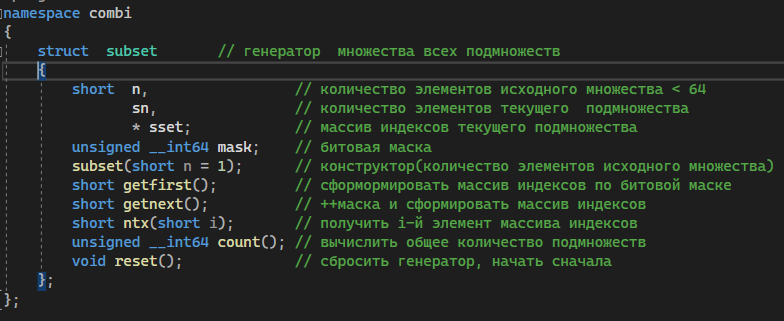
**Комбинаторные алгоритмы решения оптимизационных задач**

**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

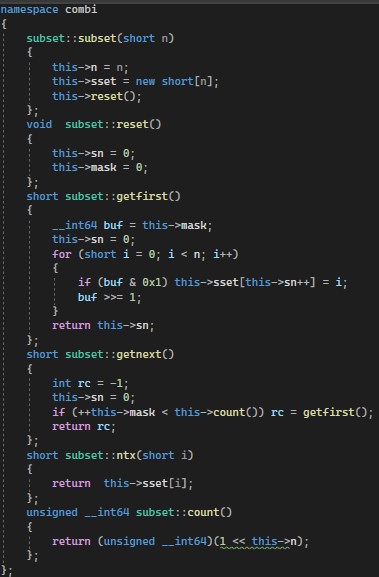
Задание №1

В терминологии теории множеств говорят, что множество B является подмножеством множества A, если каждый элемент B является в то же время и элементом множества A. Обозначается это знаком включения: B ⊂ A.

**Заголовочный файл Combi.h:**



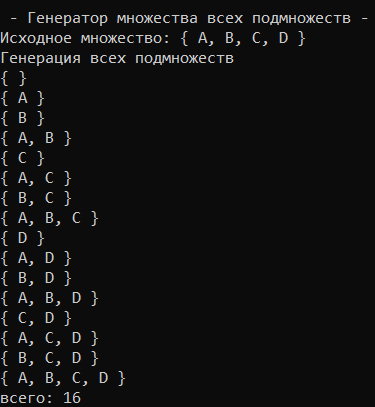
**Исходный файл Combi.cpp:**



**Код Main:**

****

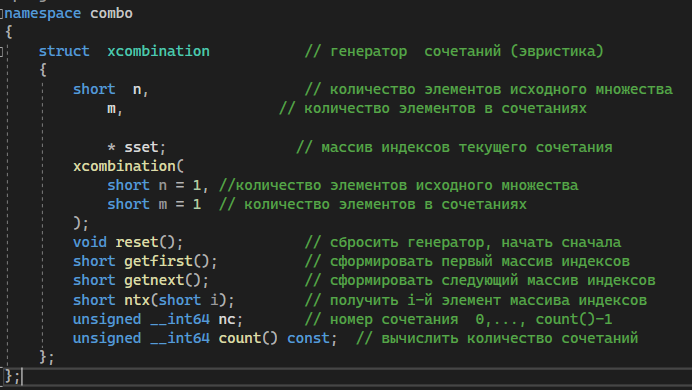
**Консоль программы:**

****

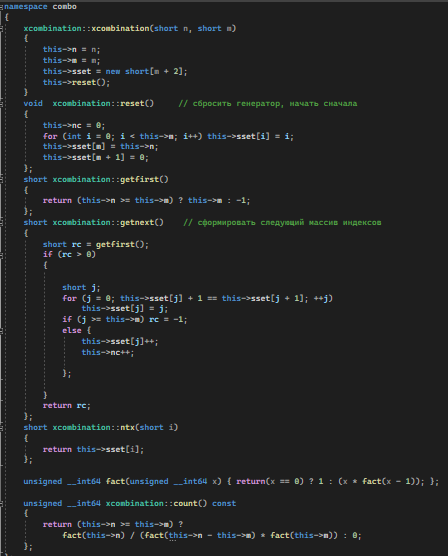
Задание №2

Сочетаниями из n элементов по k называются соединения, которые можно образовать из n элементов, собирая в каждое соединение k элементов; при этом соединения отличаются друг от друга только самими элементами (различие порядка их расположения во внимание не принимается).

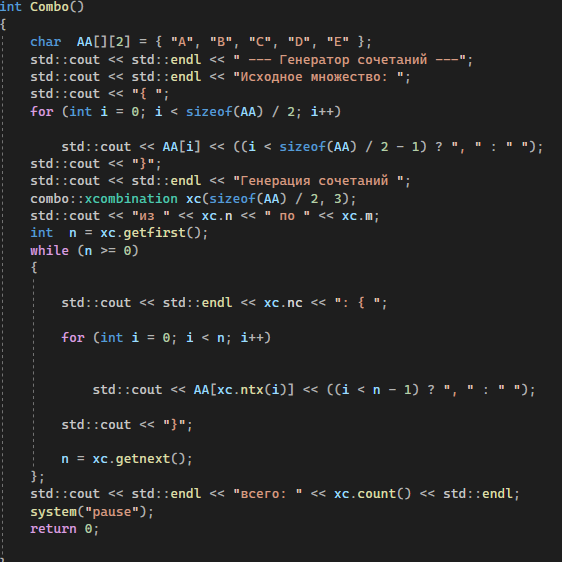
**Заголовочный файл Combo.h:**

****

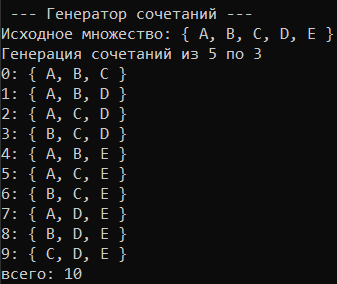
**Исходный файл Combo.cpp:**

****

**Код Main:**

****

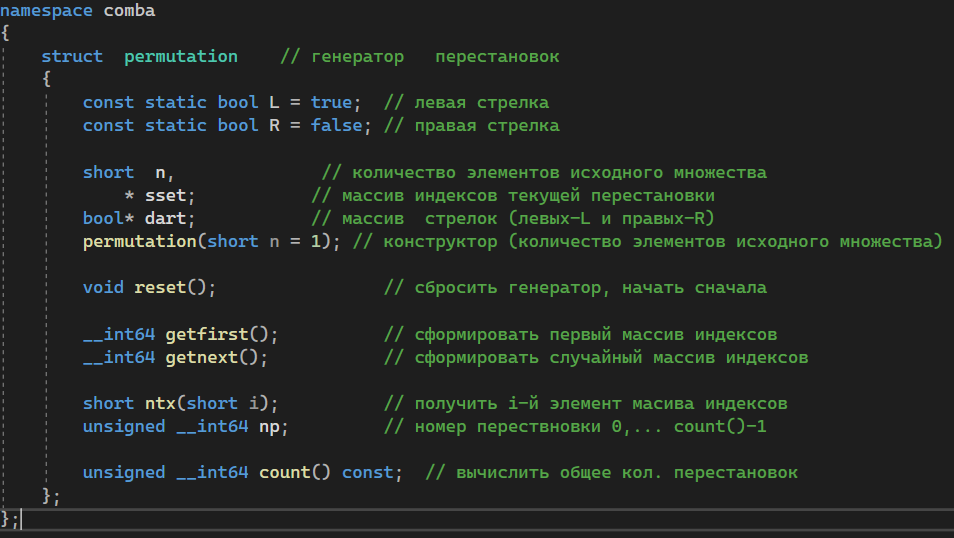
**Консоль программы:**

****

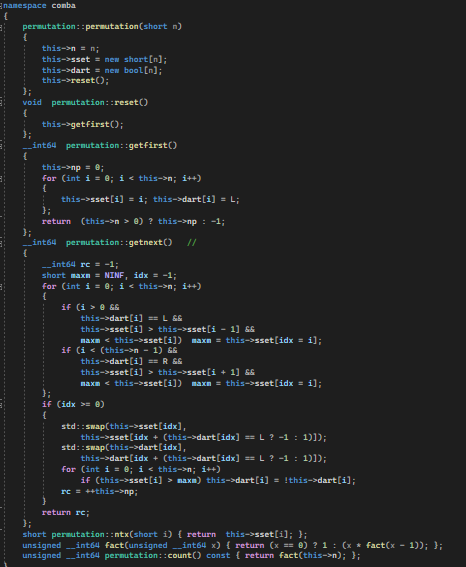
Задание №3

Перестановками называются такие выборки элементов, которые отличаются только порядком расположения элементов, но не самими элементами.

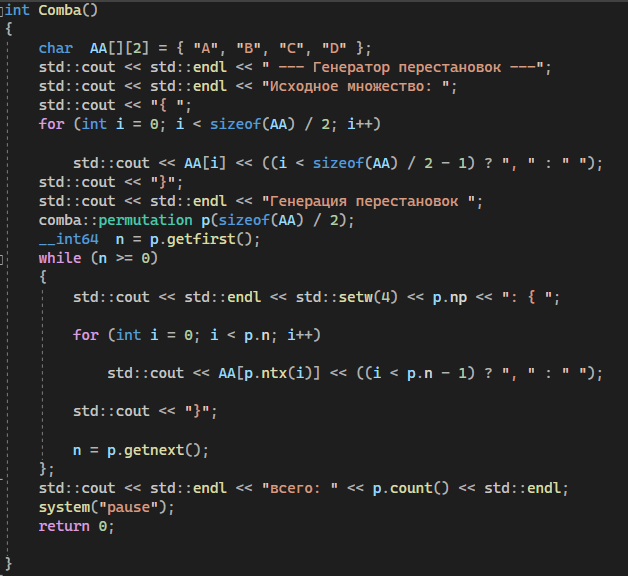
**Заголовочный файл Comba.h:**



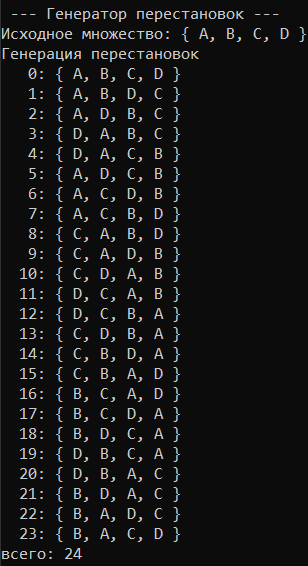
**Исходный файл Comba.cpp:**

****

**Код Main:**

****

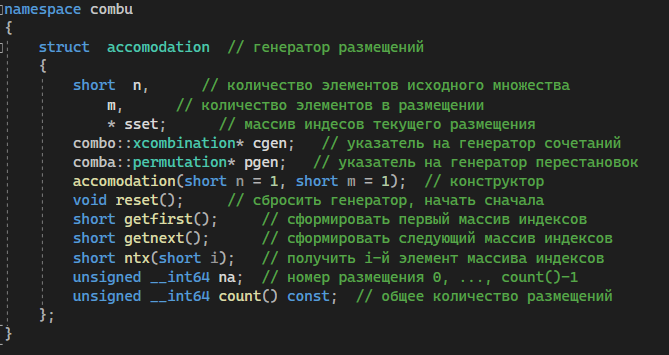
**Консоль программы:**

****

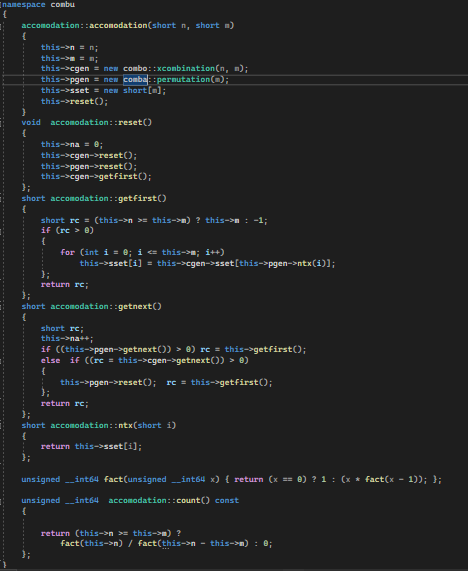
Задание №4

В комбинаторике размещением (из n по k) называется упорядоченный набор из k различных элементов из некоторого множества различных n элементов. (то есть совпадают как сочетания).

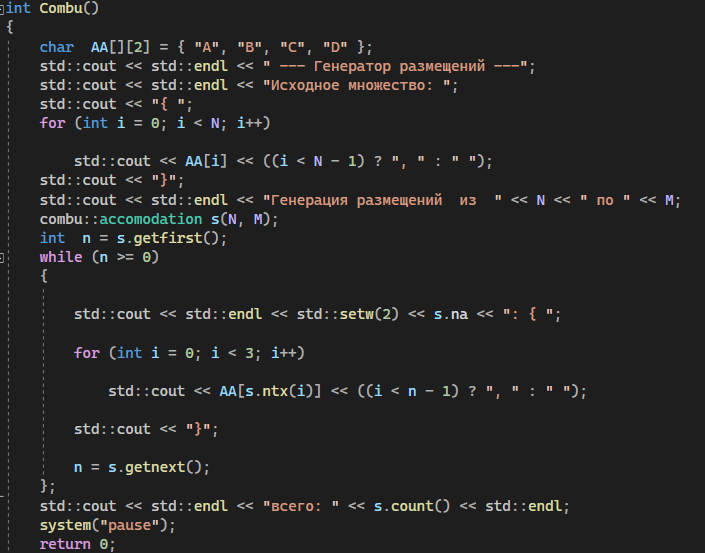
**Заголовочный файл Combu.h:**

****

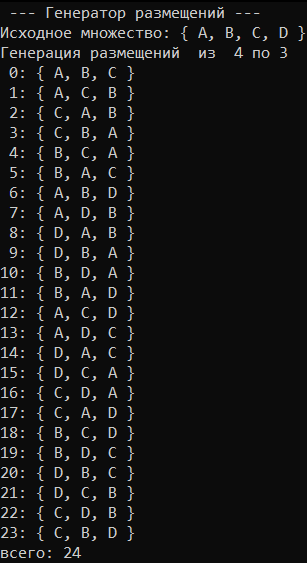
**Исходный файл Combu.cpp:**

****

**Код Main:**

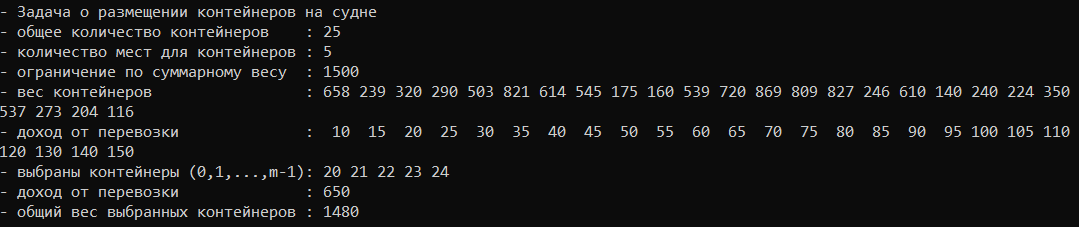
****

**Консоль программы:**

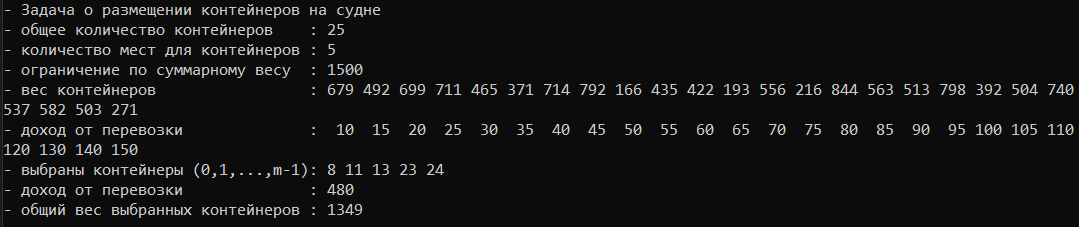
****

Задание №5

Задача об оптимальной загрузке судна (веса контейнеров сгенерировать случайным образом: ограничение по общему весу – 1500 кг., количество мест на судне для контейнеров – 5, количество контейнеров 25, веса контейнеров 100 – 900 кг., доход от перевозки 10 – 150 у.е.);



Дополнительные вычисления для подтверждения корректности работы программы.



Оценка продолжительности решения задачи о загрузке судна при разном количестве контейнеров. В программе фиксируется значение параметра **m** (количество мест для контейнеров) и вычисляется продолжительность работы функции boat в зависимости от параметра n (общее количество контейнеров).

Задание №6

Задача об оптимальной загрузке судна (количество мест на судне для контейнеров – 6, количество контейнеров 25 – 40.

Консоль:

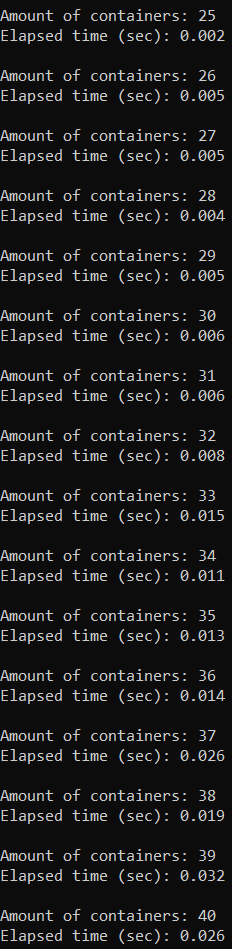


График:



Статистические данные, полученные при выполнении данного задания, говорят о степенной зависимости количества контейнеров к времени выполнения поставленной задачи.

**Заключение:**

При выполнении данной лабораторной работы были закреплены навыки разработки и создания алгоритмов комбинаторики, таких как: сочетание, размещение, перестановки и нахождение всех подмножества исходного множества. Были решены задачи об оптимальной загрузке судна с прилежащими параметрами.

Сформированы статистические данные о времени выполнения задачи с разным количеством контейнеров, степенная зависимость.

**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.

Задание №1

Для простаты города пронумеруем и будем их обозначать с помощью номера. Таблица содержит расстояние между каждой парой из пяти городов.

Например, расстояние из города 2 в город 3 равно 15 + n единицам, а из города 4 в город 5 – 3 \* n единицам. По диагонали, таблицы установлены символы, обозначающие бесконечность, что свидетельствует о невозможности такого передвижения коммивояжера.

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

Задание №2

Метод ветвей и границ – это общий алгоритмический метод решение задач комбинаторной оптимизации.

Метод ветвей и границ был предложен для решения общей задачи целочисленного линейного программирования.

Метод является вариацией полного перебора с отсевом подмножеств допустимых решений, заведомо не содержащих оптимальных решений.

Параметр n = 3, матрица расстояний будет выглядеть следующим образом:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Город | 1 | 2 | 3 | 4 | 5 |
| 1 | 1NF | 6 | 24 | 1NF | 3 |
| 2 | 3 | 1NF | 18 | 65 | 81 |
| 3 | 5 | 9 | 1NF | 86 | 52 |
| 4 | 20 | 55 | 12 | 1NF | 9 |
| 5 | 90 | 69 | 52 | 16 | 1NF |

**В соответствии с утверждением 1:** если все элементы первой строки таблицы уменьшить на 3 (наименьшее значение в строке), то это не повлияет на порядок городов в кратчайшем кольцевом маршруте, проходящем через все города по одному разу, а лишь сократит его дину на 3. Операция называется приведением таблицы по строке, а число 3 – константой привидения.

Аналогично можно поступить со всеми строками таблицы. Столбец снизу содержит константы привидения для каждого столбца, а под столбцом сумма этих констант (число 36).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Город | 1 | 2 | 3 | 4 | 5 |
| 1 | 1NF | 3 | 21 | 1NF | 0 |
| 2 | 0 | 1NF | 15 | 62 | 78 |
| 3 | 0 | 4 | 1NF | 81 | 47 |
| 4 | 11 | 46 | 3 | 1NF | 0 |
| 5 | 74 | 53 | 36 | 0 | 1NF |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3 | 3 | 5 | 9 | 16 |

|  |
| --- |
| 36 |

**В соответствии с утверждением 2:** если все элементы второго столбца таблицы уменьшить на 3 (наименьшее значение в столбце), то это не повлияет на порядок городов в кратчайшем кольцевом маршруте, проходящем через все города по одному разу, а лишь сократит его длину на 3. Будем называть эту операцию приведением таблицы по столбцу, а число 3 – константой привидения.

Аналогично можно поступить со всеми столбцами таблицы. Столбец снизу содержит константы привидения для каждого столбца, а под столбцом сумма этих констант.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Город | 1 | 2 | 3 | 4 | 5 |
| 1 | 1NF | 0 | 18 | 1NF | 0 |
| 2 | 0 | 1NF | 12 | 62 | 78 |
| 3 | 0 | 1 | 1NF | 81 | 47 |
| 4 | 11 | 43 | 0 | 1NF | 0 |
| 5 | 74 | 50 | 33 | 0 | 1NF |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 | 3 | 3 | 0 | 0 |

|  |
| --- |
| 6 |

**Утверждение 3. Приведение матрицы расстояний не влияет на выбор оптимального маршрута коммивояжера.**

Y = ∑а + ∑b = 36 + 6 = 42 – сумма констант приведения

R- множество всех допустимых решений(маршрутов);

**Утверждение 4. f(фи)(R) = y(гамма)(R) – нижняя граница длины маршрута.**

Если бы после приведения матрицы расстояний все её элементы были бы равны 0, то длина кратчайшего маршрута была бы равна y(гамма).

f(фи) = 36 + 6 = 42

Эту величину можно принять в качестве нижней границы длины кратчайшего кольцевого маршрута.

Прежде чем будет применена процедура ветвления BR, оценим, как будет изменяться ветвления BR, оценим, как будет изменяться нижняя граница подмножеств, полученных после применения процедуры BR в зависимости от выбора дуги, которая используется для разбиения.

Наибольший интерес представляет та дуга, которая наиболее сильно может повлиять на нижнюю границу длины допустимых кольцевых маршрутов.

Анализировать, прежде всего, следует дуги, имеющие нулевой вес, так как только их удаления может повлиять на нижнюю границу.

Таблица, которая представлена чуть ниже является таблицей, приведенной по строкам и по столбцам. Далее будем называть такие таблицы **полностью приведенными таблицами.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Город | 1 | 2 | 3 | 4 | 5 |
| 1 | 1NF | 0 | 18 | 1NF | 0 |
| 2 | 0 | 1NF | 12 | 62 | 78 |
| 3 | 0 | 1 | 1NF | 81 | 47 |
| 4 | 11 | 43 | 0 | 1NF | 0 |
| 5 | 74 | 50 | 33 | 0 | 1NF |

Определяем ребро ветвления и разобьём всё множество маршрутов относительного этого ребра на два подмножества (a, b) и (a’, b’).

С этой целью для всех клеток матрицы с нулевыми элементами заменяем поочерёдно нули на 1NF и определяем для них сумму образовавшихся констант привидения.

d (1, 2) = 0 + 1 = 1; d (1, 5) = 0 + 0 = 0; d (2, 1) = 12 + 0 = 12; d (3, 1) = 1 + 0 = 1;

d (4, 3) = 0 + 12 = 12; d (4, 5) = 0 + 0 = 0; d (5, 4) = 33 + 62 = 95

Наибольшая сумма констант приведения равна (33 + 62) = 95 для ребра (5,4), следовательно, множество разбивается на два подмножества (5,4) и (5’,4’).

**Исключение ребра** (5, 4) проводим путем замены элемента d5,4 = 0 на 1NF, после чего осуществляем очередное приведение матрицы расстояний для образовавшегося подмножества (5’,4’), в результате получим редуцированную матрицу.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Город | 1 | 2 | 3 | 4 | 5 |
| 1 | 1NF | 0 | 18 | 1NF | 0 |
| 2 | 0 | 1NF | 12 | 62 | 78 |
| 3 | 0 | 1 | 1NF | 81 | 47 |
| 4 | 11 | 43 | 0 | 1NF | 0 |
| 5 | 74 | 50 | 33 | 1NF | 1NF |

Нижняя граница гамильтоновых циклов этого подмножества:  
R (5’,4’) = 42 + 95 = 137

**Включение ребра** (5,4) проводится путем исключения всех элементов 5-ой строки и 4-го столбца, в которой элемент d4,5 заменяем на 1NF, для исключения образования негамильтонова цикла.

В результате получим другую сокращенную матрицу (4 x 4), которая подлежит операции приведения.  
После операции приведения сокращенная матрица будет иметь вид:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Город | 1 | 2 | 3 | 5 |
| 1 | 1NF | 0 | 18 | 0 |
| 2 | 0 | 1NF | 12 | 78 |
| 3 | 0 | 1 | 1NF | 47 |
| 4 | 11 | 43 | 0 | 0 |

Сумма констант приведения сокращенной матрицы:

Y = ∑а + ∑b = 0 + 0 = 0

Нижняя граница подмножества (5,4) равна:  
R (5,4) = 42 + 0 = 42 ≤ 137

Поскольку нижняя граница этого подмножества (5, 4) меньше, чем подмножества (5’, 4’), то ребро (5, 4) включаем в маршрут с новой границей H = 42

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Город | 1 | 2 | 3 | 5 |
| 1 | 1NF | 0 | 18 | 0 |
| 2 | 0 | 1NF | 12 | 78 |
| 3 | 0 | 1 | 1NF | 47 |
| 4 | 11 | 43 | 0 | 0 |

d (1, 2) = 0 +1 = 0; d (1, 5) = 0 +47 = 47; d (2, 1) = 12 + 0 = 12; d (3, 1) = 1 + 0 = 1;

d (4, 3) = 11 + 12 = 23;

Наибольшая сумма констант приведения равна (0 + 47) = 47 для ребра (1,5).

**Исключение ребра** (1,5):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Город | 1 | 2 | 3 | 5 |
| 1 | 1NF | 0 | 18 | 1NF |
| 2 | 0 | 1NF | 12 | 78 |
| 3 | 0 | 1 | 1NF | 47 |
| 4 | 11 | 43 | 0 | 0 |

Нижняя граница гамильтоновых циклов этого подмножества:  
R (1’, 5’) = 42 + 47 = 89;

**Включение ребра** (1,5) проводится путем исключения всех элементов 1-ой строки и 5-го столбца, в которой элемент d51 заменяем на М, для исключения образования негамильтонова цикла.

В результате получим другую сокращенную матрицу (3 x 3), которая подлежит операции приведения.  
После операции приведения сокращенная матрица будет иметь вид:

|  |  |  |  |
| --- | --- | --- | --- |
| Город | 1 | 2 | 3 |
| 2 | 0 | 1NF | 12 |
| 3 | 0 | 1 | 1NF |
| 4 | 11 | 43 | 0 |

Сумма констант приведения сокращенной матрицы:

Y = ∑а + ∑b = 0 + 1 = 1

Нижняя граница подмножества (1, 5) равна:

R (1, 5) = 42 + 1 = 43 ≤ 89

Чтобы исключить подциклы, запретим следующие переходы: (4,1).

Поскольку нижняя граница этого подмножества (1,5) меньше, чем подмножества (1’,5’), то ребро (1,5) включаем в маршрут с новой границей R = 43

|  |  |  |  |
| --- | --- | --- | --- |
| Город | 1 | 2 | 3 |
| 2 | 0 | 1NF | 12 |
| 3 | 0 | 0 | 1NF |
| 4 | 11 | 42 | 0 |

d (2, 1) = 12 + 0 = 12; d (3, 1) = 0 + 0 = 0; d (3, 2) = 0 + 42 = 42; d (4, 3) = 42 + 12 = 54;

Исключение ребра (4, 3):

|  |  |  |  |
| --- | --- | --- | --- |
| Город | 1 | 2 | 3 |
| 2 | 0 | 1NF | 12 |
| 3 | 0 | 0 | 1NF |
| 4 | 11 | 42 | 1NF |

Нижняя граница подмножества (1, 5) равна:

R (4’, 3’) = 43 + 54 = 97

**Включение ребра** (4,3) проводится путем исключения всех элементов 4-ой строки и 3-го столбца, в которой элемент d34 заменяем на М, для исключения образования негамильтонова цикла.

В результате получим другую сокращенную матрицу (2 x 2), которая подлежит операции приведения.  
После операции приведения сокращенная матрица будет иметь вид:

|  |  |  |
| --- | --- | --- |
| Город | 1 | 2 |
| 2 | 0 | 1NF |
| 3 | 0 | 0 |

Сумма констант приведения сокращенной матрицы:

Y = ∑а + ∑b = 0 + 0 = 0

Нижняя граница подмножества (1, 5) равна:

R (4, 3) = 43 + 0 = 43 ≤ 97

Поскольку нижняя граница этого подмножества (4, 3) меньше, чем подмножества (4’, 3’), то ребро (4, 3) включается в маршрут с новой границей R = 43

В соответствии с этой матрицей включаем в гамильтонов маршрут ребра (2, 1) и (3, 2).

В результате по дереву ветвей гамильтонов цикл образует ребра:

(5, 4), (4, 3), (3, 2), (2, 1) (1, 5)

Длина маршрута равна 43.

Задание №3

Проверка полученного решения при помощи генератора перестановок.

Консоль программы:



**Заключение:**

Выполняя поставленную задачу через метод ветвей и границ я не только сэкономил ресурсы для выполнения, но и время, о чём соответствует существенно малое время работы.

Также освоил общие принципы решения задач метод ветвей и границ, решил задачу о коммивояжере данным методом и сравнил полученное решение задачи с комбинаторным методом перестановок.

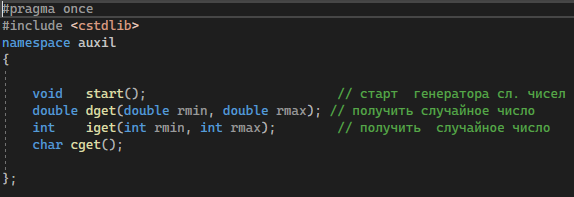
**Лабораторная работа № 4**

**Динамическое программирование**

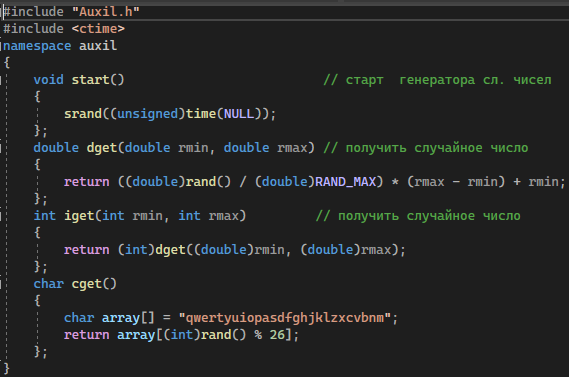
**ЦЕЛЬ РАБОТЫ:** освоить общие принципы решения задач методом динамического программирования, сравнить полученные решения задач с рекурсивным методом.

Задание № 1

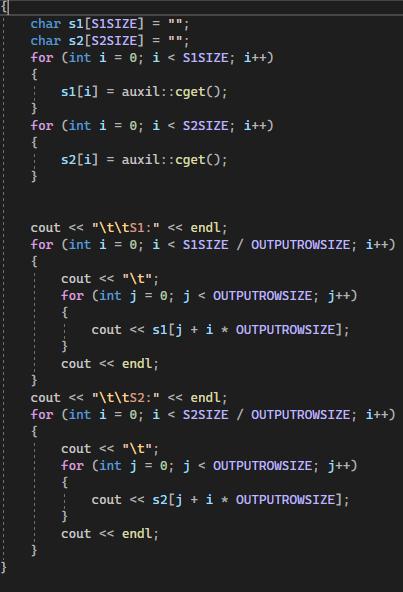
**Заголовочный файл Auxil.h:**



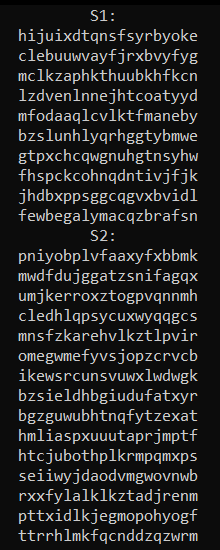
**Исходный файл Auhil.cpp:**

****

**Исходный код функции Main:**

****

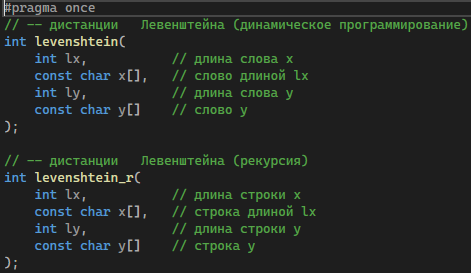
**Консоль программы:**

****

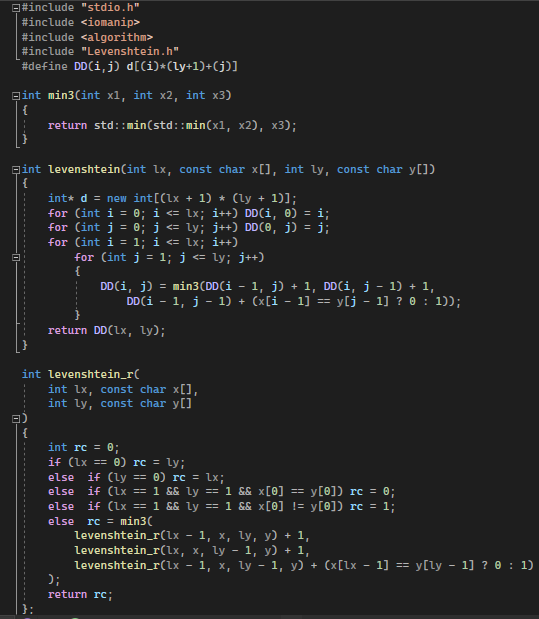
Задание № 2

Ниже приведены варианты реализации нахождения дистанции Левенштейна при помощи динамического программирования и рекурсивного алгоритма.

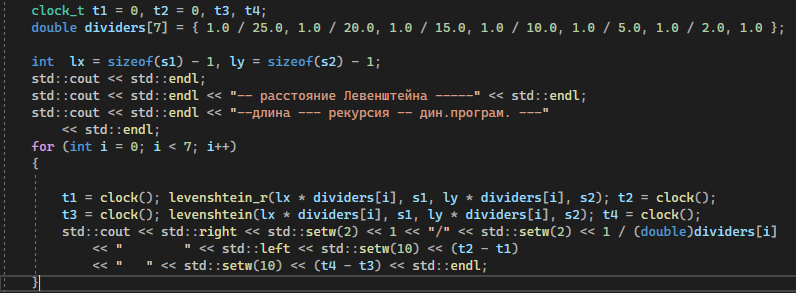
**Заголовочный файл Levenshtein.h:**



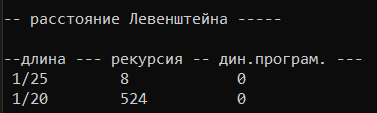
**Исходный файл Levenshtein.cpp:**

****

**Исходный код функции Main:**

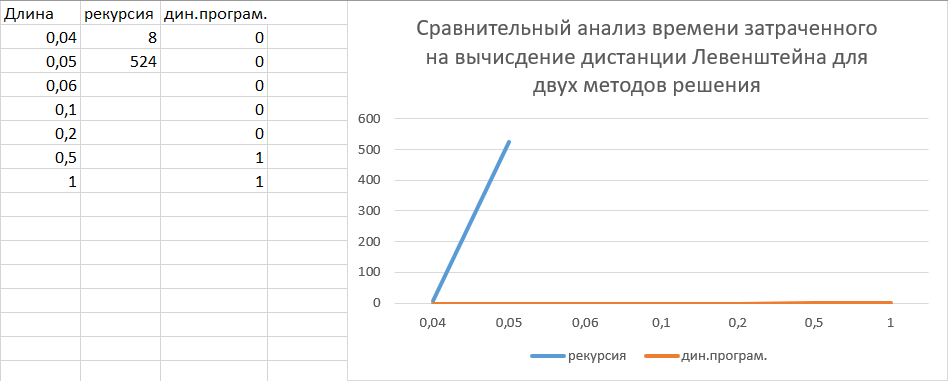
****

**Консоль программы:**

****

Задание № 3

График зависимости времени от количества символов(k):



На основе полученного графика можно сделать вывод, что вычисления, выполненные с помощью метода динамического программирования, вычисляются во много раз быстрее, нежели через метод рекурсии.

Задание № 4

Пример вычисления дистанции Левенштейна при помощи рекурсивного алгоритма.

Пусть необходимо вычислить L (“Лом”, “Гомон”). Тогда имеем следующую последовательность шагов:



= 5.

= 4.



= 4.

= 3.



= 3

= 2.



= 3

= 2.



= 0



= 2

= 0

= 1



= 0

= 2

= 1

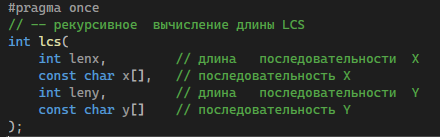
8. 3

Задание № 5

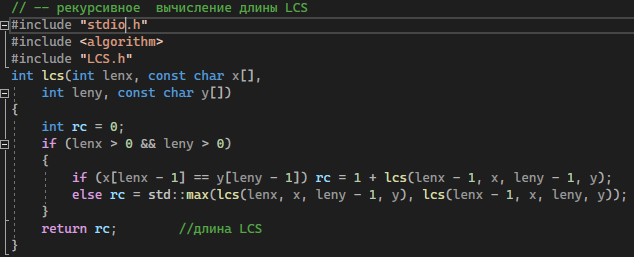
Cравнительный анализ времени затраченного на решение задачи о наибольшей общей подпоследовательности для двух методов решения (рекурсивное решение, динамическое программирование).

**Рекурсивное решение:**

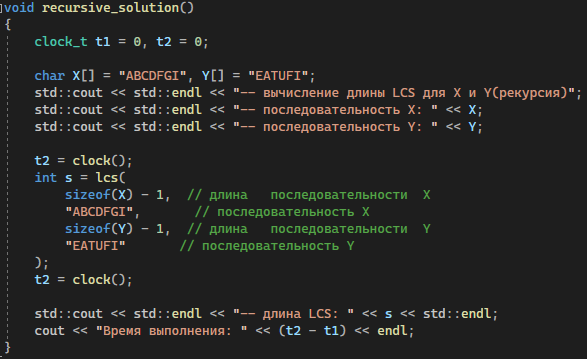
**Заголовочный файл LCS.H:**



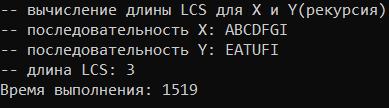
**Исходный файл LCS.CPP:**



**Код функции Main:**

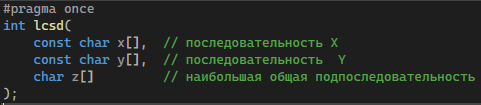


**Консоль программы:**

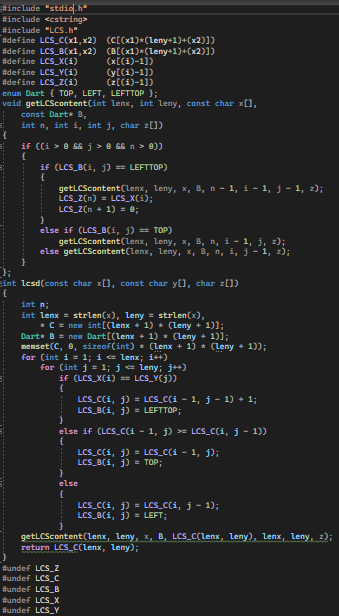


**Динамическое программирование:**

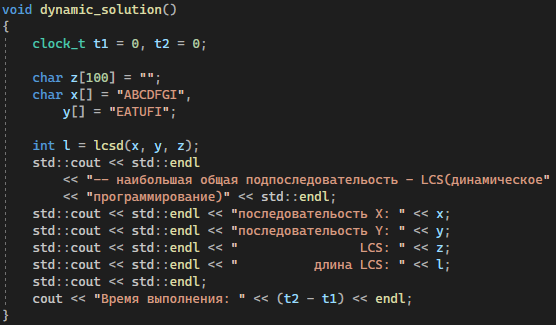
**Заголовочный файл LCH.h:**



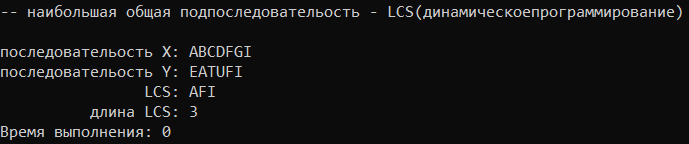
**Исходный файл LCS.cpp:**

****

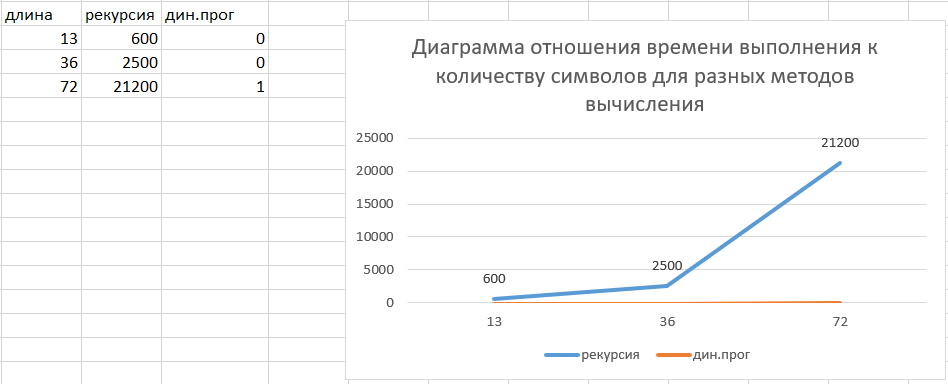
**Код функции Main:**

****

**Консоль программы:**

****

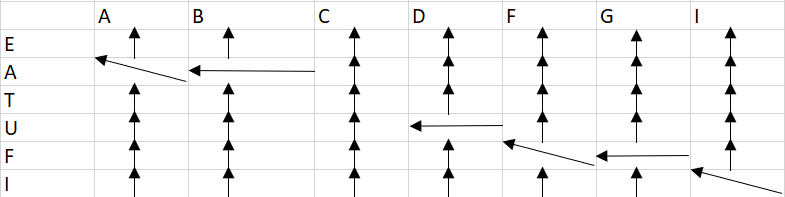
**График:**

****

На следующем рисунке представлена наглядная реализация динамического программирования, с помощью визуализации её на таблице последовательностей. Путём проверки сходимости значений, получаем итоговую цепочку для получения желаемого результата.

По итогу составленной таблицы легко вывести результат – количество действий, необходимых для получения из множества А множество В.

Ответ: 3.



**Заключение:**

На основе проведённых вычислений методом динамического программирования, а также с помощью рекурсивного алгоритма можно судить о крайне маленькой скорости выполнения, присущей рекурсивному алгоритму.

**Лабораторная работа № 5**

**Транспортная задача**

**Цель работы:** Приобретение навыков решения открытой транспортной задачи.

**Метод наименьшей стоимости:**

Пусть m = 5 поставщиков продукции, n = 6 потребителей продукции.

I – индекс для поставщиков.

J – индекс для потребителей.

Решение открытой задачи сводится к решению закрытой.

Запасы A = (171, 116, 153, 162, 103),

Потребность B = (146, 110, 134, 196, 98, 166).

Затраты на перевозку продукции C=(c(ij)) =

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Потребители  Поставщики | 1 | 2 | 3 | 4 | 5 | 6 | Запасы |
| 1 | 15 | 5 | 9 | 6 | 14 | 4 | 171 |
| 2 | 13 | 3 | 11 | 8 | 10 | 16 | 116 |
| 3 | 4 | 8 | 14 | 11 | 5 | 14 | 153 |
| 4 | 7 | 13 | 13 | 6 | 16 | 5 | 162 |
| 5 | 6 | 14 | 12 | 3 | 13 | 7 | 103 |
| Потребности | 146 | 110 | 134 | 196 | 98 | 166 |  |

Проверим необходимое и достаточное условие разрешимости задачи:

Для этого нам нужно узнать сумму мощностей поставщиков и сумму мощностей потребителей:

∑a(i) = 171 + 116 + 153 + 162 + 103 = 705;

∑b(j) = 146 + 110 + 134 + 196 + 98 + 166 = 850;

Как видно, суммарная потребность превышает суммарные запасы.

Следовательно, модель исходной транспортной задачи является открытой.

Чтобы получить закрытую модель, введем дополнительного(фиктивного) поставщика. Его мощность будет равна разности между суммами мощностей поставщиков и суммой мощностей потребителей: 850 – 705 = 145. Тарифы будут равны нулю.

Составление опорного плана:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Потребители  Поставщики | 1 | 2 | 3 | 4 | 5 | 6 | Запасы |
| 1 | 15 | 5 | 9 | 6 | 14 | 4 | 171 |
| 2 | 13 | 3 | 11 | 8 | 10 | 16 | 116 |
| 3 | 4 | 8 | 14 | 11 | 5 | 14 | 153 |
| 4 | 7 | 13 | 13 | 6 | 16 | 5 | 162 |
| 5 | 6 | 14 | 12 | 3 | 13 | 7 | 103 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 145 |
| Потребности | 146 | 110 | 134 | 196 | 98 | 166 |  |

Используя метод наименьшей стоимости, построим первый опорный план транспортной задачи.

Суть метода заключается в том, что из всей таблицы стоимостей выбирают наименьшую, и в клетку, которая ей соответствует, помещают меньшее из чисел a(i), или b (j).

Затем, из рассмотрения исключают либо строку, соответствующую поставщику, запасы которого полностью израсходованы, либо столбец, соответствующий потребителю, потребности которого полностью удовлетворены, либо и строку и столбец, если израсходованы запасы поставщика и удовлетворены потребности потребителя.

Из оставшейся части таблицы стоимостей снова выбирают наименьшую стоимость, и процесс распределения запасов продолжают, пока все запасы не будут распределены, а потребности удовлетворены.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Потребители  Поставщики | 1 | 2 | 3 | 4 | 5 | 6 | Запасы |
| 1 | 15 | 5 | 9 | 6 | 14 | 4 | 171 |
| 2 | 13 | 3 | 11 | 8 | 10 | 16 | 116 |
| 3 | 4 | 8 | 14 | 11 | 5 | 14 | 153 |
| 4 | 7 | 13 | 13 | 6 | 16 | 5 | 162 |
| 5 | 6 | 14 | 12 | 3 | 13 | 7 | 103 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 145 |
| Потребности | 146 | 110 | 134 | 196 | 98 | 166 |  |

Выбор ячейки с наименьшим значением с (2, 2) = min (116, 110) = 110.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Потребители  Поставщики | 1 | 2 | 3 | 4 | 5 | 6 | Запасы |
| 1 | 15 | 5 | 9 | 6 | 14 | 4 | 171 |
| 2 | 13 | 3|110 | 11 | 8 | 10 | 16 | 6 |
| 3 | 4 | 8 | 14 | 11 | 5 | 14 | 153 |
| 4 | 7 | 13 | 13 | 6 | 16 | 5 | 162 |
| 5 | 6 | 14 | 12 | 3|103 | 13 | 7 | 103 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 145 |
| Потребности | 146 | 0 | 134 | 196 | 98 | 166 |  |

Выбор ячейки с наименьшим значением с (5, 4) = min (103, 196) = 103.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Потребители  Поставщики | 1 | 2 | 3 | 4 | 5 | 6 | Запасы |
| 1 | 15 | 5 | 9 | 6 | 14 | 4|166 | 171 |
| 2 | 13 | 3|110 | 11 | 8 | 10 | 16 | 6 |
| 3 | 4 | 8 | 14 | 11 | 5 | 14 | 153 |
| 4 | 7 | 13 | 13 | 6 | 16 | 5 | 162 |
| 5 | 6 | 14 | 12 | 3|103 | 13 | 7 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 145 |
| Потребности | 146 | 0 | 134 | 93 | 98 | 166 |  |

Выбор ячейки с наименьшим значением с (1, 6) = min (171, 166) = 166.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Потребители  Поставщики | 1 | 2 | 3 | 4 | 5 | 6 | Запасы |
| 1 | 15 | 5 | 9 | 6 | 14 | 4|166 | 5 |
| 2 | 13 | 3|110 | 11 | 8 | 10 | 16 | 6 |
| 3 | 4|146 | 8 | 14 | 11 | 5 | 14 | 153 |
| 4 | 7 | 13 | 13 | 6 | 16 | 5 | 162 |
| 5 | 6 | 14 | 12 | 3|103 | 13 | 7 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 145 |
| Потребности | 146 | 0 | 134 | 93 | 98 | 0 |  |

Выбор ячейки с наименьшим значением с (3, 1) = min (171, 166) = 166.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Потребители  Поставщики | 1 | 2 | 3 | 4 | 5 | 6 | Запасы |
| 1 | 15 | 5 | 9 | 6 | 14 | 4|166 | 5 |
| 2 | 13 | 3|110 | 11 | 8 | 10 | 16 | 6 |
| 3 | 4|146 | 8 | 14 | 11 | 5|7 | 14 | 7 |
| 4 | 7 | 13 | 13 | 6 | 16 | 5 | 162 |
| 5 | 6 | 14 | 12 | 3|103 | 13 | 7 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 145 |
| Потребности | 0 | 0 | 134 | 93 | 98 | 0 |  |

Выбор ячейки с наименьшим значением с (3, 5) = min (7, 98) = 7.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Потребители  Поставщики | 1 | 2 | 3 | 4 | 5 | 6 | Запасы |
| 1 | 15 | 5 | 9 | 6|5 | 14 | 4|166 | 5 |
| 2 | 13 | 3|110 | 11 | 8 | 10 | 16 | 6 |
| 3 | 4|146 | 8 | 14 | 11 | 5|7 | 14 | 0 |
| 4 | 7 | 13 | 13 | 6 | 16 | 5 | 162 |
| 5 | 6 | 14 | 12 | 3|103 | 13 | 7 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 145 |
| Потребности | 0 | 0 | 134 | 93 | 91 | 0 |  |

Выбор ячейки с наименьшим значением с (1, 4) = min (5, 93) = 5.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Потребители  Поставщики | 1 | 2 | 3 | 4 | 5 | 6 | Запасы |
| 1 | 15 | 5 | 9 | 6|5 | 14 | 4|166 | 0 |
| 2 | 13 | 3|110 | 11 | 8 | 10 | 16 | 6 |
| 3 | 4|146 | 8 | 14 | 11 | 5|7 | 14 | 0 |
| 4 | 7 | 13 | 13 | 6|88 | 16 | 5 | 162 |
| 5 | 6 | 14 | 12 | 3|103 | 13 | 7 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 145 |
| Потребности | 0 | 0 | 134 | 88 | 91 | 0 |  |

Выбор ячейки с наименьшим значением с (4, 4) = min (162, 88) = 88.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Потребители  Поставщики | 1 | 2 | 3 | 4 | 5 | 6 | Запасы |
| 1 | 15 | 5 | 9 | 6|5 | 14 | 4|166 | 0 |
| 2 | 13 | 3|110 | 11 | 8 | 10|16 | 16 | 6 |
| 3 | 4|146 | 8 | 14 | 11 | 5|7 | 14 | 0 |
| 4 | 7 | 13 | 13 | 6|88 | 16 | 5 | 74 |
| 5 | 6 | 14 | 12 | 3|103 | 13 | 7 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 145 |
| Потребности | 0 | 0 | 134 | 0 | 91 | 0 |  |

Выбор ячейки с наименьшим значением с (2, 5) = min (6, 91) = 6.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Потребители  Поставщики | 1 | 2 | 3 | 4 | 5 | 6 | Запасы |
| 1 | 15 | 5 | 9 | 6|5 | 14 | 4|166 | 0 |
| 2 | 13 | 3|110 | 11 | 8 | 10|6 | 16 | 0 |
| 3 | 4|146 | 8 | 14 | 11 | 5|7 | 14 | 0 |
| 4 | 7 | 13 | 13|74 | 6|88 | 16 | 5 | 74 |
| 5 | 6 | 14 | 12 | 3|103 | 13 | 7 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 145 |
| Потребности | 0 | 0 | 134 | 0 | 85 | 0 |  |

Выбор ячейки с наименьшим значением с (4, 3) = min (74, 134) = 74.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Потребители  Поставщики | 1 | 2 | 3 | 4 | 5 | 6 | Запасы |
| 1 | 15 | 5 | 9 | 6|5 | 14 | 4|166 | 0 |
| 2 | 13 | 3|110 | 11 | 8 | 10|6 | 16 | 0 |
| 3 | 4|146 | 8 | 14 | 11 | 5|7 | 14 | 0 |
| 4 | 7 | 13 | 13|74 | 6|88 | 16 | 5 | 0 |
| 5 | 6 | 14 | 12 | 3|103 | 13 | 7 | 0 |
| 6 | 0 | 0 | 0|60 | 0 | 0 | 0 | 145 |
| Потребности | 0 | 0 | 60 | 0 | 85 | 0 |  |

Выбор ячейки с наименьшим значением с (6, 3) = min (145, 60) = 60.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Потребители  Поставщики | 1 | 2 | 3 | 4 | 5 | 6 | Запасы |
| 1 | 15 | 5 | 9 | 6|5 | 14 | 4|166 | 0 |
| 2 | 13 | 3|110 | 11 | 8 | 10|6 | 16 | 0 |
| 3 | 4|146 | 8 | 14 | 11 | 5|7 | 14 | 0 |
| 4 | 7 | 13 | 13|74 | 6|88 | 16 | 5 | 0 |
| 5 | 6 | 14 | 12 | 3|103 | 13 | 7 | 0 |
| 6 | 0 | 0 | 0|60 | 0 | 0|85 | 0 | 85 |
| Потребности | 0 | 0 | 0 | 0 | 85 | 0 |  |

Выбор ячейки с наименьшим значением с (6, 5) = min (85, 85) = 85.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Потребители  Поставщики | 1 | 2 | 3 | 4 | 5 | 6 | Запасы |
| 1 | 15 | 5 | 9 | 6|5 | 14 | 4|166 | 0 |
| 2 | 13 | 3|110 | 11 | 8 | 10|6 | 16 | 0 |
| 3 | 4|146 | 8 | 14 | 11 | 5|7 | 14 | 0 |
| 4 | 7 | 13 | 13|74 | 6|88 | 16 | 5 | 0 |
| 5 | 6 | 14 | 12 | 3|103 | 13 | 7 | 0 |
| 6 | 0 | 0 | 0|60 | 0 | 0|85 | 0 | 0 |
| Потребности | 0 | 0 | 0 | 0 | 0 | 0 |  |

В результате получен первый опорный план, который является допустимым. План соответствует системе ограничений транспортной задачи.

Проверка плана невырожденности плана перевозок:

Подсчитаем число занятых клеток таблицы, их 11, а должно быть m + n - 1 = 11 (5+6-1=10). Следовательно, опорный план является *невырожденным*.  
Первое допустимое решение: x (1, 4) = 5, x (1, 6) = 166, x (2,2) = 110, x (2, 5) = 6, x (3, 1) = 146, x (3, 5) = 7, x (4, 3) = 74, x (4, 4) = 88, x (5, 4) = 103, x (6, 3) = 60, x (6, 5) = 85.

Значение функции цели:

Z = 5\*6 + 166\*4 + 110\*3 + 6\*10 + 146\*4 + 7\*5 + 74\*13 + 88\*6 + 103\*3 + 60\*0 + 85\*0 = 3502.

**Метод потенциалов**

В методе потенциалов каждой строке i и каждого столбца j транспортной таблицы ставятся в соответствии числа (потенциалы) u(i)(поставщики) и v(j)(потребители). Для каждой базисной переменной x (i, j) потенциалы u(i) и v(j) удовлетворяют уравнению: u(i) + v(j) = c(ij)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Потребители  Поставщики | 1 | 2 | 3 | 4 | 5 | 6 | Запасы |
| 1 | 15 | 5 | 9 | 6|5 | 14 | 4|166 | u1 = 0 |
| 2 | 13 | 3|110 | 11 | 8 | 10|6 | 16 | u2 = -3 |
| 3 | 4|146 | 8 | 14 | 11 | 5|7 | 14 | u3 = -8 |
| 4 | 7 | 13 | 13|74 | 6|88 | 16 | 5 | u4 = 0 |
| 5 | 6 | 14 | 12 | 3|103 | 13 | 7 | u5 =-3 |
| 6 | 0 | 0 | 0|60 | 0 | 0|85 | 0 | u6 =-13 |
| Потребности | v1=12 | v2=6 | v3 =13 | v4 = 6 | v5=13 | v6 = 4 |  |

Определяем потенциалы для всех базисных переменных:

u1 + v4 = 6

u1 + v6 = 4

u2 + v2 = 3

u2 + v5 = 10

u3 + v1 = 4

u3 + v5 = 5

u4 + v3 = 13

u4 + v4 = 6

u5 + v4 = 3

u6 + v3 = 0

u6 + v5 = 0

Уравнений 11 неизвестных 12:

Присваиваем одному из них произвольное значение (обычно u1 = 0).

Для свободных клеток:

|  |  |
| --- | --- |
| Небазисная переменная |  |
| x (1,1) | u1 + v1 – c11 = -3 |
| x (1,2) | u1 + v2 - c12 = 1 |
| x (1,3) | u1 + v3 – c13 = 4 |
| x (1,5) | u1 + v5 – c15 = -1 |
| x (2,1) | u2 + v1 – c21 = -4 |
| x (2,3) | u2 + v3 – c23 = -1 |
| x (2,4) | u2 + v4 – c24 = -5 |
| x (2,6) | u2 + v6 – c26 = -15 |
| x (3,2) | u3 + v2 – c32 = -10 |
| x (3,3) | u3 + v3 – c33 = -9 |
| x (3,4) | u3 +v4 – c34 = -13 |
| x (3,6) | u3 + v6 – c36 = -18 |
| x (4,1) | u4 + v1 – c41 = 5 |
| x (4,2) | u4 + v2 – c42 = -7 |
| x (4,5) | u4 + v5 – c45 = -3 |
| x (4,6) | u4 + v6 – c46 = -1 |
| x (5,1) | u5 + v1 – c51 = 3 |
| x (5,2) | u5 + v2 – c52 = -11 |
| x (5,3) | u5 + v3 – c53 = -2 |
| x (5,5) | u5 + v5 – c55 = -3 |
| x (5,6) | u5 + v6 – c56 = -6 |
| x (6,1) | u6 + v1 – c61 = -1 |
| x (6,2) | u6 + v2 – c62 = -7 |
| x (6,4) | u6 + v4 – c64 = -7 |
| x (6,6) | u6 + v6 - c66 = -9 |

Если все u(i) + v(j) <= c (i, j), то план оптимальный.

Если есть хоть одно u(i) + v(j) >= c (i, j), то план не оптимальный, требуется перераспределение перевозок по циклу, и проверка нового плана.

x (1,2) >= 0,

x (1,3) >= 0,

x (4,1) >= 0,

x (5,1) >= 0.

max (1, 4, 5, 3) = 5

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Потребители  Поставщики | 1 | 2 | 3 | 4 | 5 | 6 | Запасы |
| 1 | 15 | 5 | 9 | 6|5 | 14 | 4|166 | u1 = 0 |
| 2 | 13 | 3|110 | 11 | 8 | 10|6 | 16 | u2 = -3 |
| 3 | 4|146 | 8 | 14 | 11 | 5|7 | 14 | u3 = -8 |
| 4 | 7 | 13 | 13|74 | 6|88 | 16 | 5 | u4 = 0 |
| 5 | 6 | 14 | 12 | 3|103 | 13 | 7 | u5 =-3 |
| 6 | 0 | 0 | 0|60 | 0 | 0|85 | 0 | u6 =-13 |
| Потребности | v1=12 | v2=6 | v3 =13 | v4 = 6 | v5=13 | v6 = 4 |  |

Выбираем максимальную оценку свободной клетки (4;1): 7  
Для этого в перспективную клетку (4;1) поставим знак «+», а в остальных вершинах многоугольника чередующиеся знаки «-», «+», «-».

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Потребители  Поставщики | 1 | 2 | 3 | 4 | 5 | 6 | Запасы |
| 1 | 15 | 5 | 9 | 6|5 | 14 | 4|166 | u1 = 0 |
| 2 | 13 | 3|110 | 11 | 8 | 10|6 | 16 | u2 = -3 |
| 3 | [-]  4|146 | 8 | 14 | 11 | [+]5|7 | 14 | u3 = -8 |
| 4 | [+]7 | 13 | [-] 13|74 | 6|88 | 16 | 5 | u4 = 0 |
| 5 | 6 | 14 | 12 | 3|103 | 13 | 7 | u5 =-3 |
| 6 | 0 | 0 | [+]0|60 | 0 | [-]0|85 | 0 | u6 =-13 |
| Потребности | v1=12 | v2=6 | v3 =13 | v4 = 6 | v5=13 | v6 = 4 |  |

Цикл приведен в таблице (4,1 → 4,3 → 6,3 → 6,5 → 3,5 → 3,1).  
Из грузов x (i, j) стоящих в минусовых клетках, выбираем наименьшее, т.е. у = min (4, 3) = 74. Прибавляем 74 к объемам грузов, стоящих в плюсовых клетках и вычитаем 74 из x (i, j), стоящих в минусовых клетках. В результате получим новый опорный план.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Потребители  Поставщики | 1 | 2 | 3 | 4 | 5 | 6 | Запасы |
| 1 | 15 | 5 | 9 | 6|5 | 14 | 4|166 | u1 = 0 |
| 2 | 13 | 3|110 | 11 | 8 | 10|6 | 16 | u2 = -3 |
| 3 | 4|72 | 8 | 14 | 11 | 5|81 | 14 | u3 = -8 |
| 4 | 7|74 | 13 | 13 | 6|88 | 16 | 5 | u4 = 0 |
| 5 | 6 | 14 | 12 | 3|103 | 13 | 7 | u5 =-3 |
| 6 | 0 | 0 | 0|134 | 0 | 0|11 | 0 | u6 =-13 |
| Потребности | v1=12 | v2=6 | v3 =13 | v4 = 6 | v5=13 | v6 = 4 |  |

Определяем потенциалы для всех базисных переменных:

u1 + v4 = 6

u1 + v6 = 4

u2 + v2 = 3

u2 + v5 = 10

u3 + v1 = 4

u3 + v5 = 5

u4 + v1 = 7

u4 + v4 = 6

u5 + v4 = 3

u6 + v3 = 0

u6 +v5 = 0

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Потребители  Поставщики | 1 | 2 | 3 | 4 | 5 | 6 | Запасы |
| 1 | 15 | 5 | 9 | 6|5 | 14 | 4|166 | u1 = 0 |
| 2 | 13 | 3|110 | 11 | 8 | 10|6 | 16 | u2 = 2 |
| 3 | 4|72 | 8 | 14 | 11 | 5|81 | 14 | u3 = -3 |
| 4 | 7|74 | 13 | 13 | 6|88 | 16 | 5 | u4 = 0 |
| 5 | 6 | 14 | 12 | 3|103 | 13 | 7 | u5 =-3 |
| 6 | 0 | 0 | 0|134 | 0 | 0|11 | 0 | u6 =-8 |
| Потребности | v1=7 | v2=1 | v3 =8 | v4 = 6 | v5=8 | v6 = 4 |  |

Опорный план является оптимальным, так все оценки свободных клеток удовлетворяют условию u(i) + v(j) <= c (i, j).

Минимальные затраты составят: F(x) = 6\*5 + 4\*166 + 3\*110 + 10\*6 + 4\*72 + 5\*81 + 7\*74 + 6\*88 + 3\*103 + 0\*134 + 0\*11 = 3192.

**Заключение:**

По окончанию сделанной лабораторной задачи о транспортной задаче были приобретены навыки составления транспортной задачи и её оптимизации. Был изучен алгоритм составления опорного плана и заполнения данных для подсчёта. Был изучен метод потенциалов.

**Лабораторная работа 6. АЛГОРИТМЫ НА ГРАФАХ**

**ЦЕЛЬ РАБОТЫ:** освоить сущность и программную реализацию: а) способов представления графов; б) алгоритмов поиска в ширину и глубину; в) алгоритма топологической сортировки графов. Разобрать алгоритм Прима и алгоритм Крускала.

Задание № 1

**Ориентированный граф G взять в соответствии с вариантом. Представить его в отчете в виде матрицы смежности, матрицы инцидентности, списка смежных вершин.**

Граф:



**Матрица смежности для ориентированного графа**

Элемент матрицы смежности *s*ij ориентированного графа определяется следующим образом:

- равен единице, если из вершины *v*i в вершину *v*j входит дуга;

- равен нулю, если из вершины *v*i в вершину *v*j дуга не входит.

**Матрица смежности**:

**Матрица инцидентности для ориентированного графа**

Элемент матрицы инцидентности для ориентированного графа hij определяется следующим образом:

0 – вершина i не инцидентна дуге j;

1 – дуга j выходит из вершины i;

-1 – дуга j входит в вершину i.

**Матрица инцидентности:**

**Список смежных вершин** — представляют собой набор из m множеств, соответствующих m вершинам графа. Множество, соответствующее вершине i, i = 1, m либо пустое, либо содержит номера вершин, в которые входит дуга, выходящая из вершины i.

**Список смежности вершин:**

S0 = {1, 3};

S1 = {0, 3, 4};

S2 = {3, 5};

S3 = {0, 1, 2, 4, 6};

S4 = {1, 3, 6};

S5 = {2, 6};

S6 = {3, 4, 5};

Задание № 2

**Осуществить алгоритмы поиска в ширину и глубину, а также алгоритма топологической сортировки аналогично примерам, рассмотренным на лекциях. Оформить отчет, включив в него каждый шаг выполнения алгоритмов.**

Алгоритм графа в ширину (BFS) используется для обхода графа и нахождения кратчайшего пути от начальной вершины до всех остальных вершин графа. Он работает следующим образом:

Создать очередь и добавить в нее начальную вершину.

Создать массив посещенных вершин и пометить начальную вершину как посещенную.

Пока очередь не пуста, извлечь из очереди первую вершину и для каждой смежной с ней вершины, которая еще не была посещена, пометить ее как посещенную и добавить в очередь.

Повторять шаг 3 до тех пор, пока очередь не станет пустой.

В результате работы алгоритма, для каждой вершины графа будет найден кратчайший путь от начальной вершины до нее.

**Поиск в ширину:**



0: q = (0),

1: 0, q = (1, 3),

2: 1, q = (3, 4),

3: 3, q = (4, 2, 6),

4: 4, q = (2, 6),

5: 2, q = (6, 5),

6: 6, q = (5),

7: 5, q = ().

Очередь графа пуста, поэтому обход графа закончен.

Алгоритм обхода графа в глубину (Depth-First Search, DFS) — это алгоритм, который используется для обхода всех вершин графа. Он начинает с одной вершины и идет вглубь графа, пока не достигнет конца пути. Затем он возвращается на предыдущий уровень и продолжает обход до тех пор, пока не пройдет все вершины.

**Поиск в глубину:**

4, 6, 5, 2, 3, 1, 0.

Так как все вершины помечены как посещённые, то мы заканчиваем обход графа в глубину.

Топологическая сортировка (Topological sort) — один из основных алгоритмов на графах, который применяется для решения множества более сложных задач.  
Задача топологической сортировки графа состоит в следующем: указать такой линейный порядок на его вершинах, чтобы любое ребро вело от вершины с меньшим номером к вершине с большим номером. Очевидно, что если в графе есть циклы, то такого порядка не существует.

Задача топологической сортировки графа состоит в следующем: указать такой линейный порядок на его вершинах, чтобы любое ребро вело от вершины с меньшим номером к вершине с большим номером. Очевидно, что если в графе есть циклы, то такого порядка не существует.

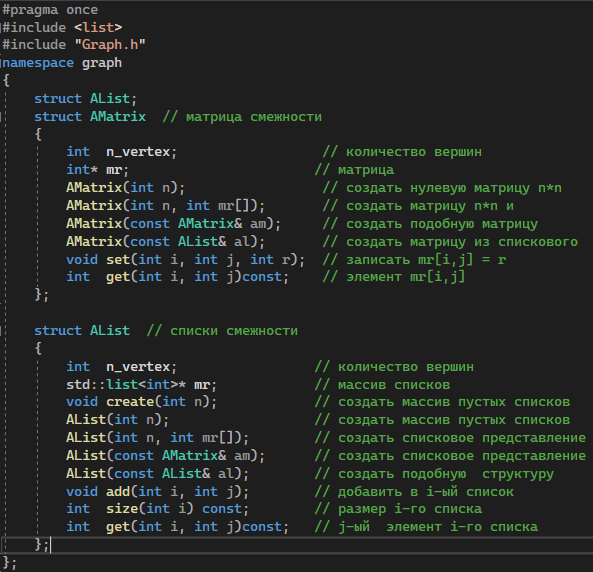
**Топологическая сортировка:**

4, 6, 5, 2, 3, 1, 0.

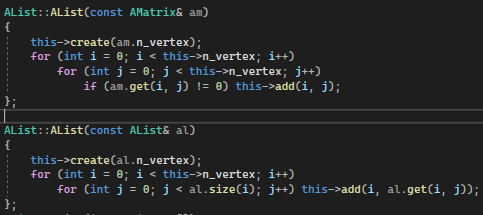
Задание № 3

**Осуществить программную реализацию алгоритмов на C++. Разработать структуры AMatrix и АList для представления ориентированного графа матричным и списковым способом. Разработать функции преобразования из одного способа представления в другой. Разработать функцию BFS обхода вершин графа, используя метод поиска в ширину. Продемонстрировать работу функции.**

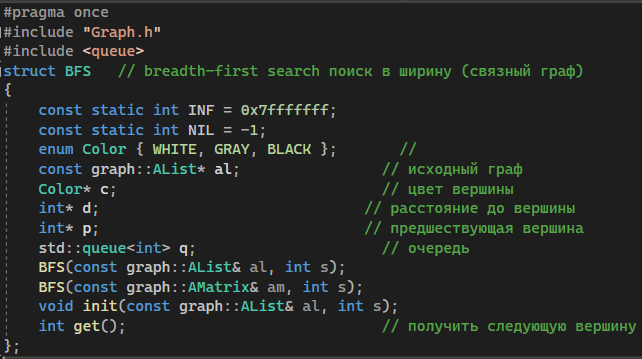
**Заголовочный файл Graph.h:**



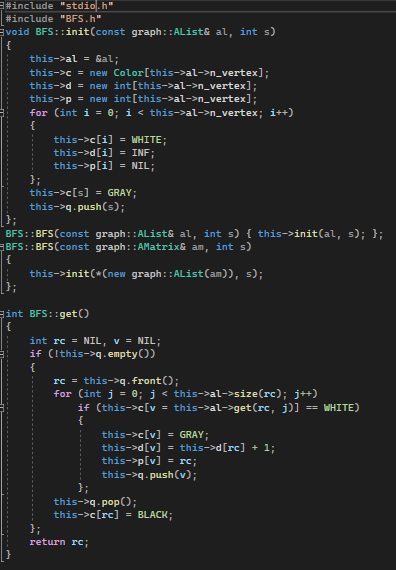
**Функции преобразования исходного файла Graph.cpp:**

****

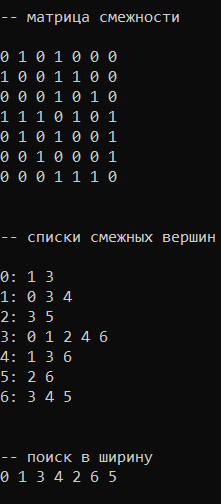
**Заголовочный файл BFS.h:**

****

**Исходный файл BFS.cpp:**

****

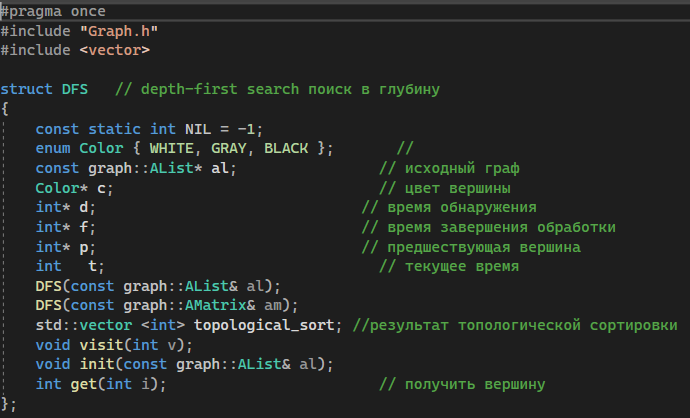
**Консоль программы:**

****

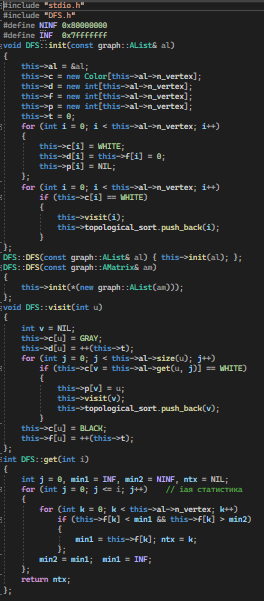
Задание № 4, 5

**Разработать функцию DFS обхода вершин графа, используя метод поиска глубину. Доработайте функцию DFS, для выполнения топологической сортировки графа.**

**Заголовочный файл DFS.h:**



**Исходный файл: DFS.cpp:**

****

**Консоль программы:**

****

Задание № 6

**По графу, составить минимальное остовное дерево по алгоритму Прима.**

**Алгоритм Прима** — это алгоритм поиска минимального остовного дерева в связном взвешенном графе. Он начинает с одной вершины и постепенно добавляет к ней ближайшие вершины, пока не будет построено остовное дерево, содержащее все вершины графа.

Выбрать любую вершину графа и поместить ее в множество X.

Найти все ребра, соединяющие вершины из множества X с вершинами, не принадлежащими X.

Выбрать ребро с минимальным весом из найденных и добавить его в остовное дерево.

Добавить к множеству X вершину, соединенную выбранным ребром.

Повторять шаги 2-4, пока все вершины не будут включены в остовное дерево.



Веса ребер:

W (e0, 1) = 8, W (e0, 3) = 2; W (e3, 1) = 4, W (e3, 4) = 1; W (e4, 1) = 3; W (e3, 2) = 9, W(e2, 5) = 11, W (e5, 6) = 2, W (e4, 6) = 10, W (e3,6) = 9.

1. 3 следует в 4 = 1;
2. 0 следует в 3 = 2;
3. 4 следует в 1 = 3;
4. 3 следует в 6 = 9;
5. 5 следует в 6 = 2;
6. 3 следует в 2 = 9;

Вес минимального остовного дерева = 1 + 2 + 3 + 9 + 2 + 9 = 26.

Задание № 7

**По графу, составить минимальное остовное дерево по алгоритму Крускала.**

**Алгоритм Крускала** — это алгоритм нахождения минимального остовного дерева в связном взвешенном графе. Он работает следующим образом:

Сортируем все ребра графа по весу в порядке возрастания.

Создаем пустой список ребер, который будет содержать ребра минимального остовного дерева.

Для каждого ребра в отсортированном списке:

Если добавление этого ребра не приведет к циклу в списке ребер, то добавляем его в список. Если добавление этого ребра приведет к циклу, то пропускаем его и переходим к следующему ребру.



Когда все ребра обработаны, список ребер будет содержать минимальное остовное дерево.

1. 3 следует в 4 = 1;
2. 0 следует в 3 = 2;
3. 5 следует в 6 = 2;
4. 4 следует в 1 = 3;
5. 3 следует в 6 = 9;
6. 3 следует в 2 = 9;

Вес минимального остовного дерева = 1 + 2 + 2 + 3 + 9 + 9 = 26.

**Заключение:**

По окончанию сделанной лабораторной работы были приобретены навыки освоения сущности и программной реализации: способов представления графов; алгоритмов поиска в ширину и глубину; алгоритма топологической сортировки графов. Разобрали алгоритм Прима и алгоритм Крускала.

**Лабораторная работа №7. Сетевые модели**

**Цель работы:** Приобретение навыков сетевого планирования и составления сетевых графиков, приобретение опыта нахождения критического пути.

Задание №1

**Задание 1.** Структурное планирование. Подумайте и выделите в проекте, согласно вашему варианту не менее 4 этапов работ. Также разбейте полученные этапы на задачи, их количество в совокупности по этапам должно быть не менее 12. Пример оформления задания смотрите в приложении ниже и в лекционном материале по теме.

**Задание 2. Календарное планирование.**

Распределите время, отпущенное на ваш проект согласно вариантам, на выделенные вами этапы. Скорректируйте сформулированные вами задачи, если это необходимо.

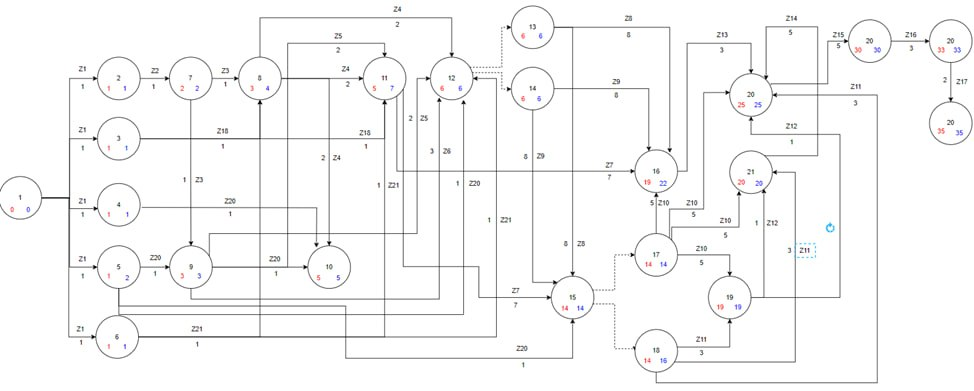
Структурное планирование десктопного приложения:

Тема «Десктопное приложение»

|  |  |  |  |
| --- | --- | --- | --- |
| Код  операции | Наименование операции | Предшествующие операции | t |
| I. АНАЛИЗ | | | |
| Z1 | Системный анализ |  | 1 |
| Z2 | Анализ рынка | Z1 | 1 |
| Z3 | Анализ требований | Z2 | 1 |
| II. ПРОЕКТИРОВАНИЕ | | | |
| Z4 | Проектирование базы данных | Z3, Z18, Z21 | 2 |
| Z5 | Проектирование FRONT-END части (пользовательский интерфейс) | Z3, Z20 | 2 |
| Z6 | Проектирование BACK-END части (функционал) | Z4, Z20 | 3 |
| III. КОДИРОВАНИЕ | | | |
| Z7 | Кодирование интерфейсов пользователей (FRONT-END) | Z4, Z5, Z18, Z21 | 7 |
| Z8 | Кодирование BACK-END части | Z4, Z5, Z6, Z20, Z21 | 8 |
| Z9 | Кодирование структуры и объектов базы данных | Z4, Z5, Z6, Z20, Z21 | 8 |
| IV. ТЕСТИРОВАНИЕ | | | |
| Z10 | Функциональное тестирование | Z7, Z8, Z9, Z20 | 5 |
| Z11 | Структурное тестирование | Z7, Z8, Z9, Z20 | 3 |
| Z12 | Тестирование производительности | Z10, Z11 | 1 |
| V. ВНЕДРЕНИЕ | | | |
| Z13 | Разработка документации | Z7, Z8, Z9, Z10 | 3 |
| Z14 | Обучение пользователей | Z10, Z11, Z12 | 5 |
| Z15 | Испытание | Z10, Z11, Z12, Z13, Z14 | 5 |
| Z16 | Рекламное продвижение | Z15 | 3 |
| Z17 | Завершение работ | Z16 | 2 |
| VI. ДОПОЛНИТЕЛЬНЫЕ РАБОТЫ | | | |
| Z18 | Установка СУБД | Z1 | 1 |
| Z19 | Установка web-сервера | Z1 | 1 |
| Z20 | Установка инструментария | Z1 | 1 |
| Z21 | Подготовка полигона | Z1 | 1 |
| Z22 | Поддержка и обновление приложения | Z17 | 1 |

**Задание 3. Сетевой график, нахождение критического пути.**

Согласно составленному перечню задач и распределённому времени составьте сетевой график вашего проекта.



Критический путь в сетевом графике — это последовательность связанных задач, которые определяют длительность проекта. Алгоритм нахождения критического пути на сетевом графике можно описать следующим образом:

1. Построить сетевой график, где вершины представляют задачи, а дуги – связи между задачами.

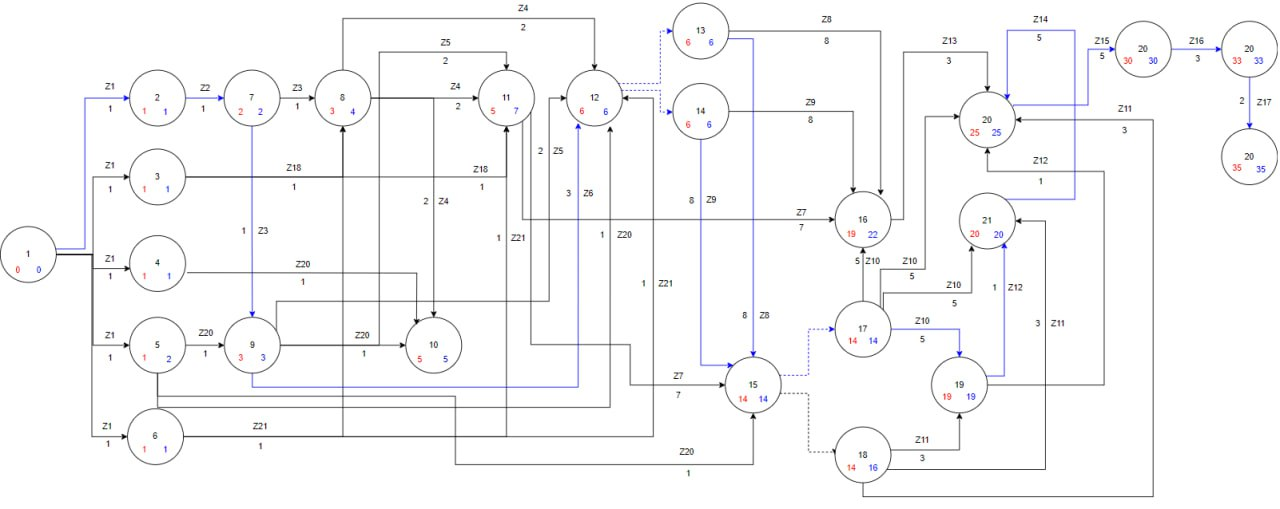
2. Для каждой задачи определить ее продолжительность.

3. Определить ранние и поздние сроки начала и окончания каждой задачи.

4. Рассчитать полный путь для каждой задачи, который представляет собой сумму продолжительностей всех задач, предшествующих данной задаче.

5. Найти критический путь, который представляет собой последовательность задач с наибольшим полным путем.

6. Определить общую длительность проекта, которая равна полному пути последней задачи в критическом пути.



**Задание 4. Оптимизация**

Предложите варианты оптимизации вашего проекта с привлечением денежных средств или человеческого ресурса.

Существует множество способов оптимизации десктопного приложения, которые могут потребовать как денежных средств, так и человеческого ресурса. Некоторые из них:

1. Оптимизация кода: найти и исправить узкие места в коде, улучшить алгоритмы, использовать более эффективные библиотеки и фреймворки. Для этого может потребоваться найм опытных разработчиков или использование специализированных инструментов для анализа кода.

2. Оптимизация интерфейса: упростить и улучшить интерфейс приложения, сделать его более интуитивно понятным и удобным для пользователей. Для этого может потребоваться найм дизайнеров и UX-специалистов.

3. Оптимизация производительности: улучшить производительность приложения, уменьшить время загрузки и отклика, уменьшить потребление ресурсов компьютера. Для этого может потребоваться найм специалистов по оптимизации производительности и использование специализированных инструментов для анализа производительности.

4. Добавление новых функций: добавить новые функции, которые могут привлечь больше пользователей и улучшить опыт использования приложения. Для этого может потребоваться найм разработчиков и дизайнеров.

5. Расширение платформ: расширить поддержку различных платформ, таких как мобильные устройства или веб-браузеры. Для этого может потребоваться найм разработчиков и тестировщиков.

6. Улучшение безопасности: улучшить безопасность приложения, защитить его от взлома и кражи данных пользователей. Для этого может потребоваться найм специалистов по безопасности и использование специализированных инструментов для анализа безопасности.

7. Улучшение качества: улучшить качество приложения, устранить ошибки и дефекты, улучшить стабильность и надежность. Для этого может потребоваться найм тестировщиков и использование специализированных инструментов для тестирования.

Конкретные варианты оптимизации зависят от конкретного приложения и его целей, а также от доступных ресурсов и бюджета.

**Заключение:**

По окончанию сделанной лабораторной работы были приобретены навыкы сетевого планирования и составления сетевых графиков, приобретение опыта нахождения критического пути.

**Лабораторная работа № 8**

**Цель работы:** освоить решение задач графическим методом.

Задание рассчитано на повторение пройденного материала.

Задание № 1

Строим область допустимых решений, т.е. решаем графически систему неравенств. Для этого строим каждую прямую и определяем полуплоскости, заданные неравенствами (полуплоскости обозначаем штрихом).

Начертим первую прямую 3 \* x1 - x2 <= 3.

Для этого найдем две точки на этой прямой, подставив значения x1 = 0 и x2 = 0 в уравнение и решив его. Получим точки (0, -3) и (1, 0). Проведем прямую через эти две точки и штрихуем полуплоскость ниже прямой 3 \* x1 - x2 <= 3.

Начертим вторую прямую x1 + x2 <= 5.

Для этого найдем две точки на этой прямой, подставив значения x1 = 0 и x2 = 0 в уравнение и решив его. Получим точки (0, 5) и (5, 0). Проведем прямую через эти две точки и штрихуем полуплоскость ниже прямой x1 + x2 <= 5.

Начертим третью прямую

Данная прямая проходит через точку (0, 0). Проведем прямую через эти две точки и штрихуем полуплоскость правее прямой x1 ≥ 0.

Начертим четвертую прямую x2 ≥ 1

Данная прямая проходит через точку (0, 1). Проведем прямую через эти две точки и штрихуем полуплоскость выше прямой x2 ≥ 1.

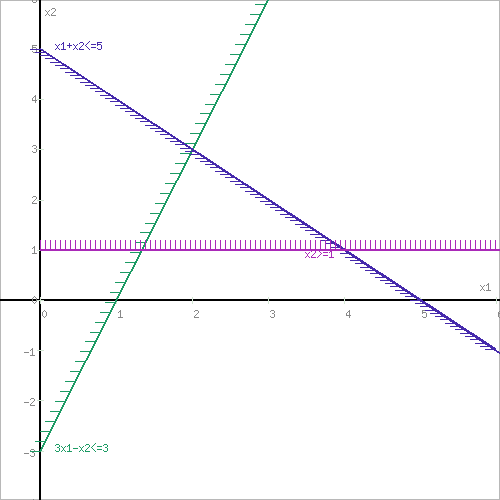


Рисунок 1 – График всех прямых

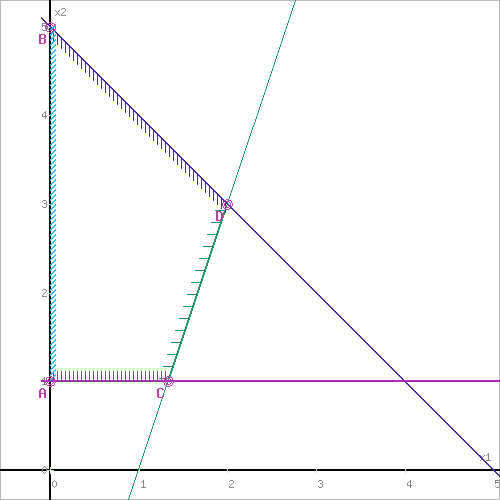


Рисунок 2 – Полуплоскость

Задание № 2

Строим прямую, соответствующую задаче, или целевой функции, приравненной к нулю. Область допустимых решений может представлять бесконечное множество. Поэтому ищем max и min в области ограничений, если это возможно.

Нарисуем прямую Z = 3\*x1 + 5\*x2 = 0. Для этого найдем две точки на этой прямой, подставив значения x1 = 0 и x2 = 1 в уравнение и решив его. Получим точки (0, 0) и (-5/3, 1). Проведем прямую через эти две точки.

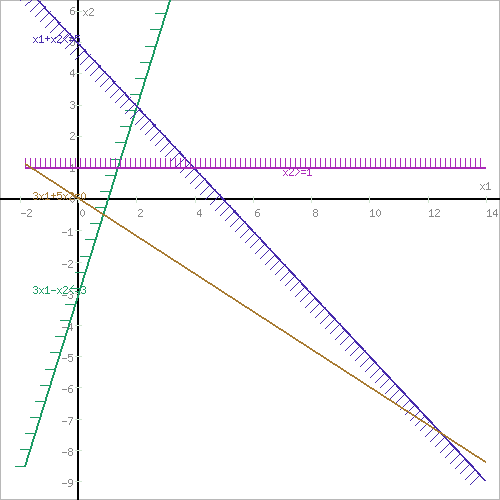


Рисунок 3 – Построение целевой функции

Теперь, чтобы найти максимальное и минимальное значение Z = 3\*x1 + 5\*x2 = 0 в области ограничений, мы проверяем каждую вершину этой области (точки пересечения прямых).

Вычислим значение Z для каждой из вершин:

Для точки (0, -3) => Z = 3\*0 + 5\*(-3) = -15

Для точки (1, 0) => Z = 3\*1 + 5\*0 = 3

Для точки (0, 5) => Z = 3\*0 + 5\*5 = 25

Для точки (5, 0) => Z = 3\*5 + 5\*0 = 15

Таким образом, максимальное значение функции Z равно 25, а минимальное значение Z равно 3.

**Заключение**

По окончанию сделанной лабораторной работы были освоены навыки решение задач графическим методом.