МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА»
(БГТУ им. В.Г. Шухова)

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Лабораторная работа №4

по дисциплине: Исследование операций

тема: «Закрытая транспортная задача»

Выполнил: ст. группы ПВ-233

Ситников Алексей Павлович

Проверил:

Вирченко Юрий Петрович

Белгород 2025 г.

Цель работы: изучить математическую модель транспортной задачи, овладеть методами решения этой задачи.

Постановка задачи

1. Изучить содержательную и математическую постановки закрытой транспортной задачи, методы нахождения первого опорного решения ее системы ограничений. Изучить понятие цикла пересчета в матрице перевозок. Овладеть распределительным методом и методом потенциалов, а также их алгоритмами.

2. Составить и отладить программы решения транспортной задачи распределительным методом и методом потенциалов.

3. Для подготовки тестовых данных решить вручную следующую задачу:

Вариант 13

13.

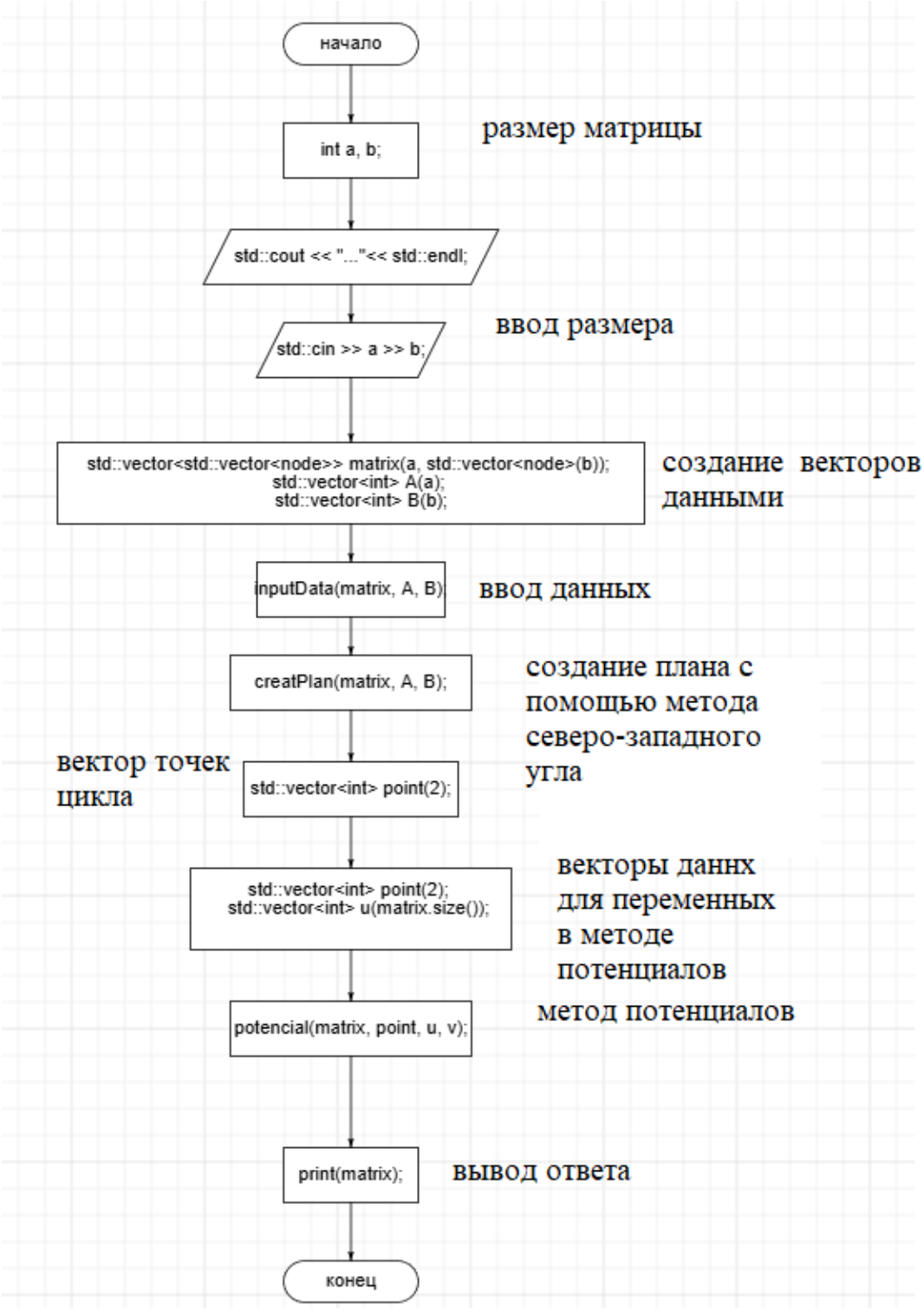
$$\vec{a} = (28, 15, 17, 14);$$

$$\vec{b} = (14, 15, 15, 15, 15);$$

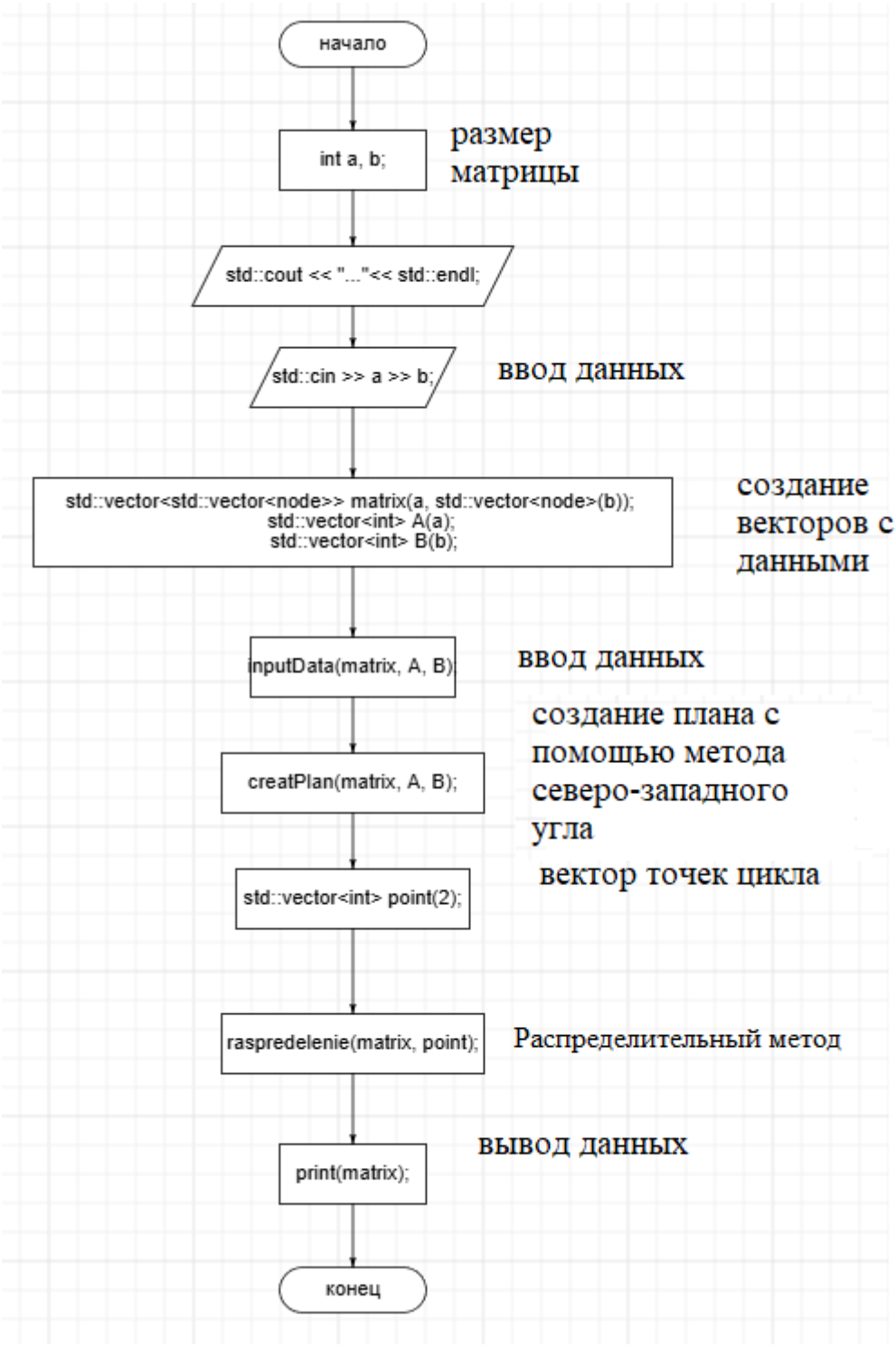
$$C = \begin{pmatrix} 27 & 6 & 8 & 12 & 23 \\ 1 & 25 & 19 & 11 & 12 \\ 28 & 19 & 15 & 17 & 29 \\ 16 & 22 & 18 & 5 & 13 \end{pmatrix}$$

Блок-схемы программ:

Метод потенциалов



Распределительный метод



Код программ:

Метод потенциалов

```
#include <iostream>
#include <windows.h>
#include <vector>

typedef struct node{
    int C = 0;//цена
    int x = 0;//сколько отправили
    int g = 0;//ля вычисление дельты
```

```
    int flag = 1;//для выяснения свободен ли элемент
}node;

void inputData(std::vector<std::vector<node>> &matrix, std::vector<int> &A,
std::vector<int> &B){
    std::cout << "Введите матрицу стоймости размером " << matrix.size() <<
"x" << matrix[0].size() << ", где ij элемент это стоимость доставки из i-той
базы к j-тому потребителю" << std::endl;
    for(int i = 0; i < matrix.size(); i++){
        for(int j = 0; j < matrix[0].size(); j++){
            std::cin >> matrix[i][j].C;
        }
    }
    std::cout << "Введите запасы начиная с первого склада\n";
    for(int &i : A){
        std::cin >> i;
    }

    std::cout << "Введите запросы начиная с первого потребителя\n";
    for(int &i : B){
        std::cin >> i;
    }

}


int finduniq(std::vector<std::vector<int>> &rout, int i, int j){
    for(auto x : rout){
        if(x[0] == i && x[1] == j){
            return 0;
        }
    }
    return 1;
}
int findRout(std::vector<std::vector<int>> &rout,
std::vector<std::vector<node>> &matrix, int *flag,
             int start_i, int start_j, int current_i, int current_j,
             int prev_i, int prev_j, int direction) {
    // direction: 0 - начальное, 1 - горизонтальное, 2 - вертикальное

    // Если вернулись в начало и цикл не пустой
    if (current_i == start_i && current_j == start_j && rout.size() > 1) {
        *flag = 1;
        return 1;
    }

    // Проверяем, были ли уже в этой точке (кроме стартовой)
    if (!(current_i == start_i && current_j == start_j)) {
        if (!finduniq(rout, current_i, current_j)) {
            return 0;
        }
    }




    // Чередуем направления (горизонтальное/вертикальное)
    if (direction != 1) {  // Проверяем по вертикали
        for (int i = 0; i < matrix.size(); i++) {
            if (i != current_i && i != prev_i && matrix[i][current_j].x != 0)
{
                rout.push_back({current_i, current_j});

                if (findRout(rout, matrix, flag, start_i, start_j,
```

```
                            i, current_j, current_i, current_j, 2)) {
                        return 1;
                    }
                }
            }
        }

        if (direction != 2) {  // Проверяем по горизонтали
            for (int j = 0; j < matrix[0].size(); j++) {
                if (j != current_j && j != prev_j && matrix[current_i][j].x != 0)
{
                    rout.push_back({current_i, current_j});

                    if (findRout(rout, matrix, flag, start_i, start_j,
                                 current_i, j, current_i, current_j, 1)) {
                        return 1;
                    }
                }
            }
        }

        // Если путь не найден - откатываемся

        return 0;
}

// Функция-обертка для запуска поиска
void findCycle(std::vector<std::vector<int>> &rout,
std::vector<std::vector<node>> &matrix, int *flag,
               int start_i, int start_j) {
    *flag = 0;
    findRout(rout, matrix, flag, start_i, start_j, start_i, start_j, -1, -1,
0);
}

void creatPlan(std::vector<std::vector<node>> &m, std::vector<int> A,
std::vector<int> B){
    //северо-западный угол
    for(int i = 0; i < m.size(); i++){
        for(int j = 0; j < m[0].size(); j++){
            if(m[i][j].flag) {//если элемент доступен
                if (A[i] > B[j]) {
                    m[i][j].x = B[j];
                    A[i] -= B[j];
                    B[j] = 0;
                    for (auto &row: m) {
                        row[j].flag = 0;
                    }
                } else if (A[i] < B[j]) {
                    m[i][j].x = A[i];
                    B[j] -= A[i];
                    A[i] = 0;
                    for(auto &col : m[i]){
                        col.flag = 0;
                    }
                }
                else{
                    m[i][j].x = A[i];
                    B[j] = 0;
                    A[i] = 0;
                    for (auto &row: m) {
                        row[j].flag = 0;
                    }
                }
```

```
            }
        }
    }
}
//метод потенциалов
void potencial(std::vector<std::vector<node>> &matrix, std::vector<int>
&point, std::vector<int> &u, std::vector<int> &v){
    while (true) {//пока не будет решение
        for (int i = 0; i < matrix.size(); i++) {//находим u и v
            for (int j = 0; j < matrix[0].size(); j++) {
                if (matrix[i][j].x != 0) {
                    if (i == 0) {
                        v[j] = matrix[i][j].C;
                    } else if (u[i] != 0) {
                        v[j] = matrix[i][j].C - u[i];
                    } else {
                        u[i] = matrix[i][j].C - v[j];
                    }
                }
            }
        }
        int min = INT_MAX;
        //вычисляем дельты
        for (int i = 0; i < matrix.size(); i++) {
            for (int j = 0; j < matrix[0].size(); j++) {
                matrix[i][j].g = matrix[i][j].C - u[i] - v[j];
                if (min > matrix[i][j].g) {
                    min = matrix[i][j].g;
                }
            }
        }
        if (min >= 0) {
            return;
        }
        std::vector<std::vector<int>> rout(1);
        //делаем перераспределение
        for (int i = 0; i < matrix.size(); i++) {
            int isEnd = 0;
            for (int j = 0; j < matrix[0].size(); j++) {
                if (min == matrix[i][j].g) {
                    rout[0].push_back(i);
                    rout[0].push_back(j);
                    int flag = 0;
                    findCycle(rout, matrix, &flag, i, j);//находим цикл
                    isEnd = 1;
                    break;
                }
            }
            if(isEnd){
                break;
            }
        }
        //находим насколько надо перебрать
        int minC = matrix[rout[0][0]][rout[0][1]].C <
matrix[rout[rout.size()-1][0]][rout[rout.size()-1][1]].C ?
matrix[rout[0][0]][rout[0][1]].C : matrix[rout[rout.size()-
1][0]][rout[rout.size()-1][1]].C;

        //перераспределение
        for(int i = 0; i < rout.size(); i++){
            if(i%2==0){
                matrix[rout[i][0]][rout[i][1]].x += minC;
            }
            else{
```

```
                matrix[rout[i][0]][rout[i][1]].x -= minC;
            }
        }
        //опустошаем векторы для повторной работы
        std::fill(u.begin(), u.end(), 0);
        std::fill(v.begin(), v.end(), 0);
    }
}

void print(std::vector<std::vector<node>> &matrix){
    //выводим решение
    for(int i = 0; i < matrix.size(); i++){
        for(int j = 0; j < matrix[0].size(); j++){
            if(matrix[i][j].x!=0){
                std::cout << matrix[i][j].x << " ";
            }
            else{
                std::cout << "  ";
            }
        }
        std::cout << "\n";
    }
    int sum = 0;
    int f = 0;
    for(int i = 0; i < matrix.size(); i++){
        for(int j = 0; j < matrix[0].size(); j++){
            if(matrix[i][j].x!=0){
                if(f){
                    std::cout << "* ";
                }
                std::cout << matrix[i][j].x << "*" << matrix[i][j].C <<" ";
                f = 1;
                sum+=matrix[i][j].x * matrix[i][j].C;
            }
        }
    }
    std::cout << "= " << sum;
}




int main(){
    SetConsoleOutputCP(CP_UTF8);
    int a, b;
    std::cout << "Введите количество баз и количество потребителей"<<
std::endl;
    std::cin >> a >> b;

    std::vector<std::vector<node>> matrix(a, std::vector<node>(b));
    std::vector<int> A(a);
    std::vector<int> B(b);
    inputData(matrix, A, B);
    creatPlan(matrix, A, B);
    std::vector<int> point(2);
    std::vector<int> u(matrix.size());
    std::vector<int> v(matrix[0].size());
    potencial(matrix, point, u, v);
    print(matrix);
```

```
    return 0;
}
```

Вывод программы:

```
C:\Users\admin\CLionProjects\Io\cmake-build-debug\Io.exe
Введите количество баз и количество потребителей
4 5
Введите матрицу стоймости размером 4x5, где ij элемент это стоимость доставки из i-той базы к j-тому потребителю
27 6 8 12 23
1 25 19 11 12
28 19 15 17 29
16 22 18 5 13
Введите запасы начиная с первого склада
28 15 17 14
Введите запросы начиная с первого потребителя
14 15 15 15 15
   15 13
14          1
       2 15
           14
15*6 + 13*8 + 14*1 + 1*12 + 2*15 + 15*17+14*13 = 687
Process finished with exit code 0
```

Распределительный метод

```
#include <iostream>
#include <windows.h>
#include <vector>

typedef struct node{
    int C = 0;//цена
    int x = 0;//сколько отправили
    int flag = 1;//для выяснения свободен ли элемент
}node;

void inputData(std::vector<std::vector<node>> &matrix, std::vector<int> &A,
std::vector<int> &B){
    std::cout << "Введите матрицу стоймости размером " << matrix.size() <<
"x" << matrix[0].size() << ", где ij элемент это стоимость доставки из i-той
базы к j-тому потребителю" << std::endl;
    for(int i = 0; i < matrix.size(); i++){
        for(int j = 0; j < matrix[0].size(); j++){
            std::cin >> matrix[i][j].C;
        }
    }
    std::cout << "Введите запасы начиная с первого склада\n";
    for(int &i : A){
        std::cin >> i;
    }

    std::cout << "Введите запросы начиная с первого потребителя\n";
    for(int &i : B){
        std::cin >> i;
    }

}


int finduniq(std::vector<std::vector<int>> &rout, int i, int j){
    for(auto x : rout){
        if(x[0] == i && x[1] == j){
            return 0;
```

```
        }
    }
    return 1;
}
int findRout(std::vector<std::vector<int>> &rout,
std::vector<std::vector<node>> &matrix, int *flag,
             int start_i, int start_j, int current_i, int current_j,
             int prev_i, int prev_j, int direction) {
    // direction: 0 - начальное, 1 - горизонтальное, 2 - вертикальное

    // Если вернулись в начало и цикл не пустой
    if (current_i == start_i && current_j == start_j && rout.size() > 1) {
        *flag = 1;
        return 1;
    }

    // Проверяем, были ли уже в этой точке (кроме стартовой)
    if (!(current_i == start_i && current_j == start_j)) {
        if (!finduniq(rout, current_i, current_j)) {
            return 0;
        }
    }




    // Чередуем направления (горизонтальное/вертикальное)
    if (direction != 1) {  // Проверяем по вертикали
        for (int i = 0; i < matrix.size(); i++) {
            if (i != current_i && i != prev_i && matrix[i][current_j].x != 0)
{
                rout.push_back({current_i, current_j});

                if (findRout(rout, matrix, flag, start_i, start_j,
                             i, current_j, current_i, current_j, 2)) {
                    return 1;
                }
            }
        }
    }

    if (direction != 2) {  // Проверяем по горизонтали
        for (int j = 0; j < matrix[0].size(); j++) {
            if (j != current_j && j != prev_j && matrix[current_i][j].x != 0)
{
                rout.push_back({current_i, current_j});

                if (findRout(rout, matrix, flag, start_i, start_j,
                             current_i, j, current_i, current_j, 1)) {
                    return 1;
                }
            }
        }
    }

    // Если путь не найден - откатываемся

    return 0;
}

// Функция-обертка для запуска поиска
int  findCycle(std::vector<std::vector<int>> &rout,
std::vector<std::vector<node>> &matrix, int *flag,
               int start_i, int start_j) {
```

```
    *flag = 0;
    return findRout(rout, matrix, flag, start_i, start_j, start_i, start_j, -
1, -1, 0);
}

void creatPlan(std::vector<std::vector<node>> &m, std::vector<int> A,
std::vector<int> B){
    //северо-западный угол
    for(int i = 0; i < m.size(); i++){
        for(int j = 0; j < m[0].size(); j++){
            if(m[i][j].flag) {//если элемент доступен
                if (A[i] > B[j]) {
                    m[i][j].x = B[j];
                    A[i] -= B[j];
                    B[j] = 0;
                    for (auto &row: m) {
                        row[j].flag = 0;
                    }
                } else if (A[i] < B[j]) {
                    m[i][j].x = A[i];
                    B[j] -= A[i];
                    A[i] = 0;
                    for(auto &col : m[i]){
                        col.flag = 0;
                    }
                }
                else{
                    m[i][j].x = A[i];
                    B[j] = 0;
                    A[i] = 0;
                    for (auto &row: m) {
                        row[j].flag = 0;
                    }
                }
            }
        }
    }
}
//распределительный метод
void raspredelenie(std::vector<std::vector<node>> &matrix, std::vector<int>
&point){
    while (true) {//пока не будет решение
        std::vector<std::vector<int>> rout(1);
        //подсчитываем g
        int min = INT_MAX;
        int indX = 0;//запоминаем индексы минимального элемента по сумме
        int indY = 0;
        for (int i = 0; i < matrix.size(); i++) {

            int flag = 0;
            for (int j = 0; j < matrix[0].size(); j++) {//находим гаммы
                if (findCycle(rout, matrix, &flag, i, j)) {//находим цикл
                    int sum = 0;
                    for(int k = 0; k < rout.size(); k++){
                        if(k%2==0){//находим гаммы
                            sum+=matrix[rout[i][0]][rout[i][1]].C;
                        }
                        else{
                            sum-=matrix[rout[i][0]][rout[i][1]].C;
                        }
                    }
                    if(min > sum){//запоминаем минимальные
                        min = sum;
                        indX = i;
```

```
                        indY = j;
                    }
                }
            }
        }
        int flag = 0;//минимальный перераспределяем
        findCycle(rout, matrix, &flag, indX, indY);
        //находим насколько надо перебрать
        int minC = matrix[rout[0][0]][rout[0][1]].C <
matrix[rout[rout.size()-1][0]][rout[rout.size()-1][1]].C ?
matrix[rout[0][0]][rout[0][1]].C : matrix[rout[rout.size()-
1][0]][rout[rout.size()-1][1]].C;

        //перераспределение
        for(int i = 0; i < rout.size(); i++){
            if(i%2==0){
                matrix[rout[i][0]][rout[i][1]].x += minC;
            }
            else{
                matrix[rout[i][0]][rout[i][1]].x -= minC;
            }
        }
        if(min >= 0){//если минимальный не отрицательный, то оптимальное
решение найдено
            break;
        }
    }
}

void print(std::vector<std::vector<node>> &matrix){
    //выводим решение
    for(int i = 0; i < matrix.size(); i++){
        for(int j = 0; j < matrix[0].size(); j++){
            if(matrix[i][j].x!=0){
                std::cout << matrix[i][j].x << " ";
            }
            else{
                std::cout << "  ";
            }
        }
        std::cout << "\n";
    }
    int sum = 0;
    int f = 0;
    for(int i = 0; i < matrix.size(); i++){
        for(int j = 0; j < matrix[0].size(); j++){
            if(matrix[i][j].x!=0){
                if(f){
                    std::cout << "* ";
                }
                std::cout << matrix[i][j].x << "*" << matrix[i][j].C <<" ";
                f = 1;
                sum+=matrix[i][j].x * matrix[i][j].C;
            }
        }
    }
    std::cout << "= " << sum;
}

int main(){
    SetConsoleOutputCP(CP_UTF8);
    int a, b;
    std::cout << "Введите количество баз и количество потребителей"<<
std::endl;
```

```
    std::cin >> a >> b;

    std::vector<std::vector<node>> matrix(a, std::vector<node>(b));
    std::vector<int> A(a);
    std::vector<int> B(b);
    inputData(matrix, A, B);
    creatPlan(matrix, A, B);
    std::vector<int> point(2);
    raspredelenie(matrix, point);
    print(matrix);
    return 0;
}
```

Вывод программы:

```
C:\Users\admin\CLionProjects\Io\cmake-build-debug\Io.exe
Введите количество баз и количество потребителей
4 5
Введите матрицу стоймости размером 4x5, где ij элемент это стоимость доставки из i-той базы к j-тому потребителю
27 6 8 12 23
1 25 19 11 12
28 19 15 17 29
16 22 18 5 13
Введите запасы начиная с первого склада
28 15 17 14
Введите запросы начиная с первого потребителя
14 15 15 15 15
   15 13
14          1
      2 15
          14
15*6 + 13*8 + 14*1 + 1*12 + 2*15 + 15*17+14*13 = 687
Process finished with exit code 0
```

Решения совпали

Аналитическое решение методом потенциалов.

Поставщик\потребитель	14	15	15	15	15
28	14	14			
15		1	14		
17			1	15	1
14					14

Проверка оптимальности методом потенциалов

Вычисление потенциалов:

v1 = 27, v2 = 6, v3 = 0, v4 = 2, v5 = 14.

u1 = 0, u2 = 19, u3 = 15, u4 = -1

Ɣ13 = 8, Ɣ14 = 10, Ɣ15 = 9, Ɣ21 = -45, Ɣ24 = -10, Ɣ25 = -21, Ɣ31 = -14, Ɣ32 = -2, Ɣ41 = -10, Ɣ42 = 17, Ɣ43 = 19, Ɣ 44 = 4.

Есть отрицательные элементы, минимальный Ɣ21. Строим цикл и получаем:

Поставщик\потребитель	14	15	15	15	15
28	13	15			
15	1		14		
17			1	15	1
14					14

Проверка оптимальности методом потенциалов

Вычисление потенциалов:
v1 = 27, v2 = 6, v3 = 45, v4 = 47, v5 = 59.

u1 = 0, u2 = -26, u3 = -30, u4 = -46

Ɣ13 = -37, Ɣ14 = -35, Ɣ15 = -36, Ɣ22 = 45, Ɣ24 = -10, Ɣ25 = -21, Ɣ31 = 31, Ɣ32 = 45, Ɣ41 = 35, Ɣ42 = 62, Ɣ43 = 14, Ɣ 44 = 4.

Есть отрицательные элементы, значит план не оптимальный, минимальный Ɣ13, строим цикл

Поставщик\потребитель	14	15	15	15	15
28		15	13		
15	14		1		
17			1	15	1
14					14

v1 = 16, v2 = 6, v3 = 8, v4 = 10, v5 = 22.

u1 = 0, u2 = 11, u3 = 7, u4 = -9.

Ɣ11 = 11, Ɣ14 = 2, Ɣ15 = 1, Ɣ22 = 3, Ɣ24 = 0, Ɣ25 = -21, Ɣ31 = 5, Ɣ32 = 6, Ɣ41 = 9, Ɣ42 = 25, Ɣ43 = 19, Ɣ 44 = 4.

Есть отрицательные элементы, значит план не оптимальный, минимальный Ɣ25, строим цикл

Поставщик\потребитель	14	15	15	15	15
28		15	13		
15	14				1
17			2	15	
14					14

v1 = 10, v2 = 6, v3 = 8, v4 = 10, v5 = 21.

u1 = 0, u2 = -9, u3 = 7, u4 = -8.

Ɣ11 = 11, Ɣ14 = 2, Ɣ15 = 2, Ɣ22 = 27, Ɣ23 = 20, Ɣ24 = 12, Ɣ31 = 11, Ɣ32 = 6, Ɣ35 = 1, Ɣ41 = 14, Ɣ42 = 24, Ɣ43 = 18, Ɣ 44 = 3.

Отрицательные элементы отсутствуют, оптимальный план получен

Подсчёт стоимости:

15*6 + 13*8 + 14*1 + 1*12 + 2*15 + 15*17+14*13 = **687**

Аналитическое решение распределительным методом.

Поставщик\потребитель	14	15	15	15	15
28	14	14			
15		1	14		
17			1	15	1
14					14

Ɣ13 = c13 - c12 + c22 - c23 = 8–6 + 25–19 = 8

Ɣ21 = c21 – c11 + c12 – c22 = 1–27 + 6–25 = -45 < 0

Ɣ24 = c24 – c34 + c33 - c23 = 11–17 + 15–19 = -10 < 0

Ɣ25 = c25 – c23 + c33 – c35 = 12–19 + 15–29 = -21 < 0

Ɣ32 = c32 - c42 + c43 – c33 = 19–22 + 18–15 = 0

Ɣ43 = c43 – c33 + c35 – c45 = 18–15 + 29–13 = 19

Ɣ44 = c44 – c34 + c35 – c45 = 5–17 + 29–13 = 4

Ɣ21 включаем в решение:

Поставщик\потребитель	14	15	15	15	15
28	13	15			
15	1		14		
17			1	15	1
14					14

Ų13 = c13 - c11 + c21 – c23 = 8–27 + 1–19 = -37 < 0

Ų22 = c22 – c21 + c11 – c12 = 25–1 + 27–6 = 45

Ų24 = c24 – c34 + c33 - c23 = 11–17 + 15–19 = -10 < 0

Ų43 = c43 – c33 + c35 – c45 = 18–15 + 29–13 = 19

Ų44 = c44 – c34 + c35 – c45 = 5–17 + 29–13 = 4

Ų13 включаем в решение:

Поставщик\потребитель	14	15	15	15	15
28		15	13		
15	14		1		
17			1	15	1
14					14

Ų11 = c11 – c21 + c23 – c13 = 27–1 + 19–8 = 37

Ų14 = c14 - c13 + c33 – c34 = 12–8 + 15–17 = 2

Ų15 = c15 - c13 + c33 – c35 = 23–8 + 15–29 = 1

Ų22 = c22 - c12 + c13 – c23 = 25–6 +8–19 = 8

Ų24 = c24 – c23 + c33 – c34 = 17–15 + 18–5 = 15

Ų25 = c25 – c23 + c33 – c35 = 12–19 + 15–29 = -21 < 0

Ų31 = c31 – c21 + c23 – c33 = 28-1+19-15 = 31

Ų32 = c32 – c12 + c13 – c33 = 19-6+8-15 = 6

Ų43 = c43 – c33 + c35 – c45 = 18-15+29-13 = 19

Ų44 = c44 – c34 + c35 – c45 = 5-17+29-13 = 4

Ų25 включаем в решение:

Поставщик\потребитель	14	15	15	15	15
28		15	13		
15	14				1
17			2	15	
14					14

$\gamma_{14} = c_{14} - c_{34} + c_{33} - c_{13} = 12 - 17 + 15 - 8 = 2$

$\gamma_{32} = c_{32} - c_{12} + c_{13} - c_{33} = 19 - 6 + 8 - 15 = 6$

$\gamma_{41} = c_{41} - c_{21} + c_{25} - c_{45} = 16 - 1 + 12 - 13 = 14$

Отрицательных элементов нет, оптимальный план получен.

Подсчёт стоимости:

15*6 + 13*8 + 14*1 + 1*12 + 2*15 + 15*17+14*13 = **687**

Решения совпали.

Вывод: после проделанной работы получили программы, которые решают транспортную задачу методом потенциалов и распределительным методом.