**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ**

**Московский государственный технический университет**

**им. Н.Э. Баумана**

**(МГТУ им. Н.Э. Баумана)**

**Кафедра «Информационная безопасность» (ИУ8)**

Лабораторная работа № 4

По дисциплине: «Методы оптимизации»

# Тема: «Решение задачи многокритериальной оптимизации»

# Вариант 11

Преподаватель**:**   
Коннова Н.С.

Студент**:**   
Новак А.Д.

Группа**:**   
ИУ8-35

г. Москва 2019 г.

**Цель работы**

Изучить постановку задачи многокритериальной оптимизации (МКО); овладеть навыками решения задач МКО с помощью различных методов, осуществить сравнительный анализ результатов, полученных при помощи разных методов.

**Постановка задачи МКО**

Выбрать лучшую из альтернатив решения предложенной задачи по варианту с точки зрения указанных критериев.

**Условие задачи**

Покупка автомобиля:

Альтернативы:  
А. Suzuki;

В. Mitsubishi;  
С. Honda;

D. Toyota.

Критерии:  
1.Стоимость;  
2.Расходы на обслуживание;  
3.Расход бензина;  
4.Комфорт.

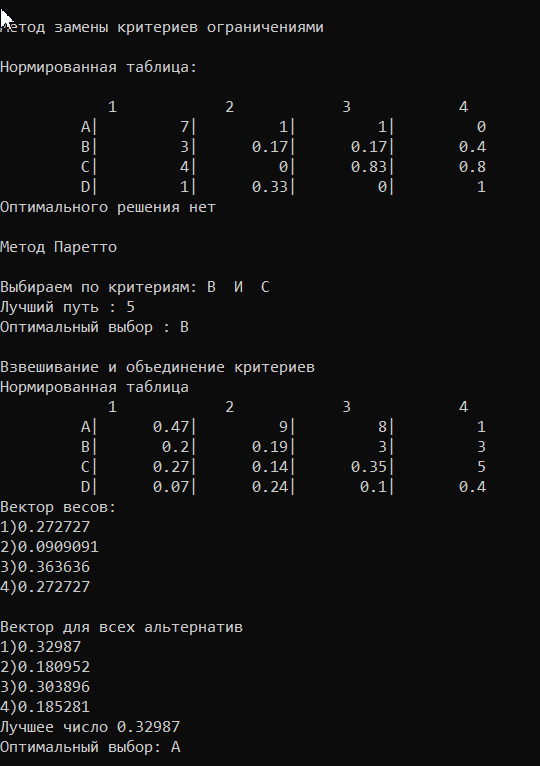
Описание предпочтений:  
Стоимость: Suzuki существенно дороже всех, Honda немного дороже Mitsubishi, Toyota существенно дешевле всех. Расходы на обслуживание: Mitsubishi дороже всех, Toyota и Suzuki примерно равны, Honda дешевле всех. Расход бензина: самый высокий у Suzuki, немного меньше у Honda, существенно меньше у Mitsubishi, самый низкий – у Toyota. Комфорт: самая комфортная –Toyota, чуть менее – Mitsubishi, существенно хуже – Honda, самая некомфортная – Suzuki.

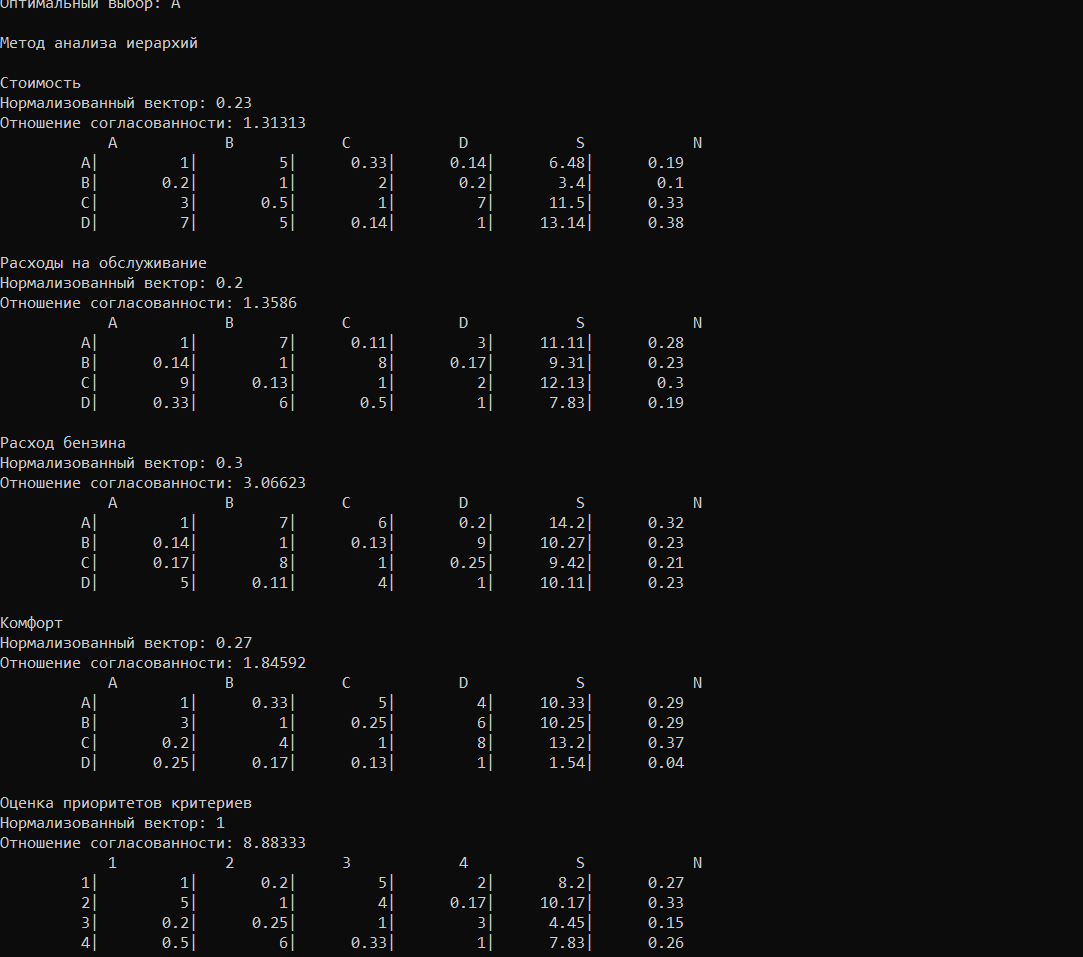
**Ход решения:**

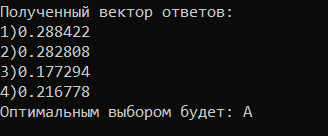
Для решения поставленной задачи были созданы 5 функций , которые находили решение отдельным методом.

**Вывод программы:**

При запуске программы выводится следущее:







**Вывод:**

При выполнении лабораторной работы были полученные разные оптимальные решения. Это свидетельствует о том , что важность критериев в каждом методе отличалась от других.

**Приложение**

#include "pch.h"

#include <iostream>

#include <vector>

#include <algorithm>

#include <cmath>

#include <iomanip>

using namespace std;

void change\_meth(vector<vector<double>> prim, int num, vector<char> vibor)

{

cout << "\nМетод замены критериев ограничениями" << endl;

vector<double> minn(prim.size());

vector<double> maxn(prim.size());

for (int i = 0; i < prim.size(); i++)

{

minn[i] = 100;

maxn[i] = -100;

if (i == num) { continue; }

for (int j = 0; j < prim[i].size(); j++)

{

minn[i] = min(minn[i], prim[j][i]);

maxn[i] = max(maxn[i], prim[j][i]);

}

}

for (int i = 0; i < prim.size(); i++)

{

for (int j = 0; j < prim[i].size(); j++)

{

if (j == num) { continue; }

prim[i][j] = (prim[i][j] - minn[j]) / (maxn[j] - minn[j]);

}

}

cout << "\nНормированная таблица: " << endl;

cout << endl;

vector<char> vibor1 = { '1','2','3','4' };

for (int i = 0; i < 4; i++)

{

cout << setw(13) << vibor1[i];

}

cout << endl;

for (int i = 0; i < prim.size(); i++)

{

cout << setw(10) << vibor[i];

for (int j = 0; j < prim[i].size(); j++)

{

cout << "|" << setw(10) << round(prim[i][j] \* 100) / 100;

}

cout << "\n";

}

int check = 0;

for (int i = 0; i < prim.size(); i++)

{

for (int j = 0; j < prim[i].size(); j++)

{

if (prim[j][i] == 0) {

check++;

if (check == 3)

{

cout << "Оптимального решения нет" << endl;

break;

}

}

}

}

for (int i = 0; i < prim.size(); i++)

{

for (int j = 0; j < prim[i].size(); j++)

{

if (i == num) { continue; }

maxn[i] = max(maxn[i], prim[j][i]);

}

}

maxn[1] = maxn[1] \* 0.2;

maxn[2] = maxn[2] \* 0.5;

maxn[2] = maxn[3] \* 0.6;

for (int i = 0; i < 4; i++)

{

int index1 = 0;

for (int j = 0; j < prim.size(); j++)

{

if (i == num) { continue; }

if (prim[i][j] > maxn[i])

{

index1++;

if (index1 == 3)

{

cout << "Оптимальное решение: " << vibor[i] << endl;

break;

}

}

}

}

}

void meth\_Pareto(vector<vector<double>> prim, int num1, int num2, vector<char> vibor)

{

cout << "\nМетод Паретто" << endl;

int best = 100000000;

int perem = 0;

cout << "\nВыбираем по критериям: " << vibor[num1] << " И " << vibor[num2] << endl;

for (int i = 0; i < prim.size(); i++)

{

int f = sqrt(pow((0 - prim[i][num1]), 2) + pow((0 - prim[i][num2]), 2));

if (f < best)

{

best = f;

perem = i;

}

}

cout << "Лучший путь : " << best << endl;

cout << "Оптимальный выбор : " << vibor[perem] << endl;

}

void weight\_meth(vector<vector<double>> prim, vector<char> vibor) {

cout << "\nВзвешивание и объединение критериев" << endl;

vector<double> summ(prim.size());

for (int i = 0; i < prim.size(); i++)

{

for (int j = 0; j < prim[i].size(); j++)

{

summ[i] = summ[i] + prim[j][i];

}

}

cout << "Нормированная таблица" << endl;

vector<char> vibor1 = { '1','2','3','4' };

for (int i = 0; i < 4; i++)

{

cout << setw(13) << vibor1[i];

}

cout << endl;

for (int i = 0; i < prim.size(); i++)

{

cout << setw(10) << vibor[i];

for (int j = 0; j < prim[i].size(); j++)

{

prim[j][i] = prim[j][i] / summ[i];

cout << "|" << setw(10) << round(prim[i][j] \* 100) / 100;

}

cout << "\n";

}

double sumv = 0;

vector<double> vec{ 0.5 + 1,0.5,1 + 1,1 + 0.5 };

for (int i = 0; i < vec.size(); i++)

{

sumv = sumv + vec[i];

}

cout << "Вектор весов: " << endl;

for (int i = 0; i < vec.size(); i++)

{

vec[i] = vec[i] / sumv;

cout << i + 1 << ")" << vec[i] << endl;

}

vector<double> otv(vec.size());

for (int i = 0; i < prim.size(); i++)

{

for (int j = 0; j < prim[i].size(); j++)

{

otv[i] = otv[i] + vec[j] \* prim[i][j];

}

}

sumv = 0;

int index = 0;

cout << "\nВектор для всех альтернатив " << endl;

for (int i = 0; i < otv.size(); i++)

{

cout << i + 1 << ")" << otv[i] << endl;

if (otv[i] > sumv) {

index = i;

sumv = otv[i];

}

}

cout << "Лучшее число " << sumv << "\n" << "Оптимальный выбор: " << vibor[index] << endl;

}

void normalize\_tab(vector<vector<double>> & prim, double sum, double k, double koef, vector<char> vibor)

{

double sums = 0;

for (int i = 0; i < 4; i++)

{

sum = 0;

for (int j = 0; j < 4; j++)

{

if (i == j) {

prim[i][j] = 1;

}

if (prim[i][j] == 0 && prim[j][i] != 0)

{

prim[i][j] = 1 / prim[j][i];

}

sum = sum + prim[i][j];

}

prim[i][4] = sum;

sums = sums + sum;

}

for (int i = 0; i < 4; i++)

{

prim[i][5] = prim[i][4] \* 1 / sums;

}

sum = 0;

for (int i = 0; i < 4; i++)

{

for (int j = 0; j < 4; j++)

{

sum = sum + prim[i][j];

}

}

sum = ((koef \* sum) - 4) / 3;

cout << "\nНормализованный вектор: " << koef << "\nОтношение согласованности: " << sum << endl;

for (int i = 0; i < 6; i++)

{

cout << setw(13) << vibor[i];

}

cout << endl;

for (int i = 0; i < 4; i++)

{

if (koef == 1) { cout << setw(10) << vibor[i]; }

else { cout << setw(10) << vibor[i]; }

for (int j = 0; j < 6; j++)

{

cout << "|" << setw(10) << round(prim[i][j] \* 100) / 100;

}

cout << "\n";

}

for (int i = 0; i < 4; i++)

{

for (int j = 0; j < 4; j++)

{

prim[i][j] = 0;

}

}

}

void metHierarchy(vector<double> & vect, vector<char> vibor)

{

cout << "\nМетод анализа иерархий" << endl;

double k = 0;

double sum = 0;

int index = 0;

for (int i = 0; i < vect.size(); i++)

{

sum = sum + vect[i];

}

for (int i = 0; i < vect.size(); i++)

{

vect[i] = round(vect[i] \* (1 / sum) \* 100) / 100;

}

vector<vector<double>> matr(4);

vector<vector<double>> otv(4);

for (int i = 0; i < matr.size(); i++)

{

otv[i].resize(4);

matr[i].resize(6);

}

cout << "\nСтоимость";

matr[0][1] = 5.;

matr[0][2] = 1. / 3.;

matr[0][3] = 1. / 7.;

matr[1][2] = 2.;

matr[1][3] = 1. / 5.;

matr[2][3] = 7.;

normalize\_tab(matr, sum, k, vect[k], vibor);

for (int i = 0; i < 4; i++)

{

otv[i][k] = matr[i][5];

}

k++;

cout << "\nРасходы на обслуживание ";

matr[0][1] = 7;

matr[0][2] = 1. / 9.;

matr[0][3] = 3;

matr[1][2] = 8;

matr[1][3] = 1. / 6.;

matr[2][3] = 2;

normalize\_tab(matr, sum, k, vect[k], vibor);

for (int i = 0; i < 4; i++)

{

otv[i][k] = matr[i][5];

}

k++;

cout << "\nРасход бензина";

matr[0][1] = 7.;

matr[0][2] = 6;

matr[0][3] = 1. / 5.;

matr[1][2] = 1. / 8.;

matr[1][3] = 9.;

matr[2][3] = 1. / 4.;

normalize\_tab(matr, sum, k, vect[k], vibor);

for (int i = 0; i < 4; i++)

{

otv[i][k] = matr[i][5];

}

k++;

cout << "\nКомфорт";

matr[0][1] = 1. / 3.;

matr[0][2] = 5;

matr[0][3] = 4;

matr[1][2] = 1. / 4.;

matr[1][3] = 6.;

matr[2][3] = 8.;

normalize\_tab(matr, sum, k, vect[k], vibor);

for (int i = 0; i < 4; i++)

{

otv[i][k] = matr[i][5];

}

cout << "\nОценка приоритетов критериев";

matr[0][1] = 1. / 5.;

matr[0][2] = 5;

matr[0][3] = 2;

matr[1][2] = 4;

matr[1][3] = 1. / 6.;

matr[2][3] = 3.;

vector<char> vibor1 = { '1','2','3','4','S','N' };

normalize\_tab(matr, sum, k, 1, vibor1);

vector<double> ends(4);

for (int i = 0; i < otv.size(); i++)

{

for (int j = 0; j < otv[i].size(); j++)

{

ends[i] = ends[i] + otv[i][j] \* matr[i][5];

}

}

cout << "Полученный вектор ответов: " << endl;

for (int i = 0; i < 4; i++)

{

cout << i + 1 << ")" << ends[i] << endl;

}

sum = 0;

for (int i = 0; i < ends.size(); i++)

{

if (ends[i] > sum) {

index = i;

sum = ends[i];

}

}

cout << "Оптимальным выбором будет: " << vibor[index] << endl;

}

int main()

{

setlocale(LC\_ALL, "Russian");

vector<double> koef = { 1,2,3,4 };

vector<char> vibor = { 'A','B','C','D','S','N' };

vector<vector<double>> tab = { {7, 9, 8, 1},

{3, 4, 3, 3 },

{4, 3, 7, 5},

{1, 5, 2, 6 } };

vector<double> vecto = { 7,6,9,8 };

change\_meth(tab, 0, vibor);

meth\_Pareto(tab, 1, 2, vibor);

weight\_meth(tab, vibor);

metHierarchy(vecto, vibor);

}