**Лабораторная работа №4**

**«Метод анализа иерархий»**

**1.Цель работы**

Углубление теоретических знаний в области системного анализа, приобретение навыков создания и описания иерархических структур, а также изучение понятий цель, критерий, альтернатива.

**2. Вариант заданий**

**Вариант 1:**

Предприятие для проектируемых изделий должно выбирать операционную систему по заданным критериям указанных в баллах.

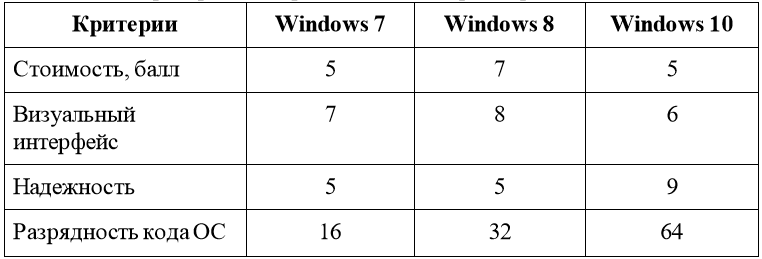


Рисунок 1 — Вариант задания

**3. Порядок выполнения работы**

1. Решить задачу методом анализа иерархий.

2. Написать программу, которая решает МАИ задачу любой размерности.

**4. Ход работы**

Для решения задачи методом анализа иерархий ручным способом используем программу Excel (рисунок 2):

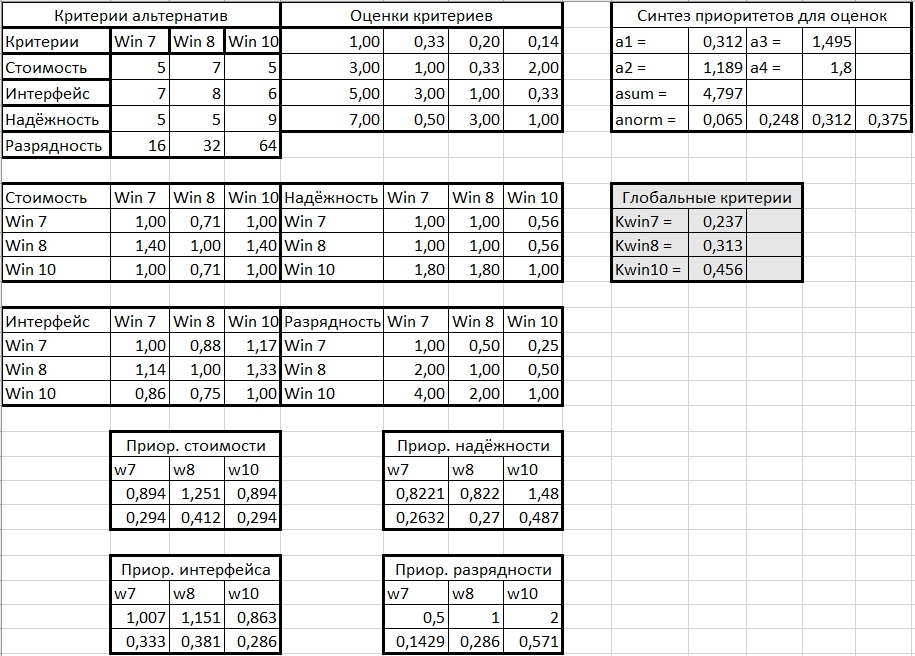


Рисунок 2 — Метод МАИ в Excel

Также для решения задачи данным методом была написана программа на языке программирования C/C++ (рисунок 3):

#include <iostream>

#include <math.h>

using namespace std;

/\* Фукция создания матрицы \*/

float \*\*createMatrix(int row, int col)

{

float\*\* m = new float\* [row];

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

{

m[i] = new float[col];

}

cout << "Enter " << row \* col << " matrix element: " << endl;

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

{

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

{

cin >> m[i][j];

}

}

return m;

}

/\* Функция создания матрицы сравнения \*/

float \*\*priorityCriteria(float \*\*matr, int quant\_alt, int curr\_crit)

{

float\*\* m = new float\* [quant\_alt];

for (int i = 0; i < quant\_alt; i++)

{

m[i] = new float[quant\_alt];

}

for (int i = 0; i < quant\_alt; i++)

{

for (int j = 0; j < quant\_alt; j++)

{

m[i][j] = matr[curr\_crit][i] / matr[curr\_crit][j];

}

}

return m;

}

/\* Функция подсчёта приоритетов \*/

float \*calcultPriority(float \*\*m, int quant\_alt)

{

float sum = 0; //сумма приоритетов

float\* vec = new float[quant\_alt]; //вектор приоритетов

for (int i = 0; i < quant\_alt; i++)

{

float proizv = 1; //произведение строки

for (int j = 0; j < quant\_alt; j++)

{

proizv = proizv \* m[i][j];

}

vec[i] = pow(proizv, (1.0 / quant\_alt));

sum += vec[i];

}

for (int i = 0; i < quant\_alt; i++)

{

vec[i] = vec[i] / sum; //нормализация

}

return vec;

}

/\* Функция расчёта глобальных приоритетов \*/

float\* calcultGlobalPriority(float\*\* m, float\* v, int quant\_crit, int quant\_alt)

{

float\* global\_prior = new float[quant\_alt];

for (int i = 0; i < quant\_alt; i++)

{

global\_prior[i] = 0;

}

for (int i = 0; i < quant\_alt; i++)

{

for (int j = 0; j < quant\_crit; j++)

{

global\_prior[i] += m[j][i] \* v[j];

}

}

return global\_prior;

}

int main()

{

int quant\_alt; //количество альтернатив

int quant\_crit; //количество критериев

cout << "Enter quantity criteria: "; cin >> quant\_crit;

cout << "Enter quantity alternative: "; cin >> quant\_alt;

float\*\* crit\_alt; //Матрица критериев альтернатив

float\*\* eval\_crit; //Матрица оценок критериев

float\*\* cost, \*\* reliab,

\*\* interf, \*\* capacity; //Матрицы сравнения критериев

float\*\* priority; //Матрица приоритетов

float\* prioir\_eval; //Вектор приоритетов оценок

float\* global\_crit; //Вектор глобальных приоритетов

crit\_alt = createMatrix(quant\_crit, quant\_alt);

eval\_crit = createMatrix(quant\_crit, quant\_crit);

cost = priorityCriteria(crit\_alt, quant\_alt, 0);

interf = priorityCriteria(crit\_alt, quant\_alt, 1);

reliab = priorityCriteria(crit\_alt, quant\_alt, 2);

capacity = priorityCriteria(crit\_alt, quant\_alt, 3);

prioir\_eval = calcultPriority(eval\_crit, quant\_crit);

priority = new float\* [quant\_crit];

priority[0] = calcultPriority(cost, quant\_alt);

priority[1] = calcultPriority(interf, quant\_alt);

priority[2] = calcultPriority(reliab, quant\_alt);

priority[3] = calcultPriority(capacity, quant\_alt);

global\_crit = calcultGlobalPriority(priority, prioir\_eval, quant\_crit, quant\_alt);

for (int i = 0; i < quant\_crit; i++)

{

delete[] crit\_alt[i];

}

delete[] crit\_alt;

for (int i = 0; i < quant\_crit; i++)

{

delete[] eval\_crit[i];

}

delete[] eval\_crit;

for (int i = 0; i < quant\_alt; i++)

{

delete[] cost[i];

}

delete[] cost;

for (int i = 0; i < quant\_alt; i++)

{

delete[] reliab[i];

}

delete[] reliab;

for (int i = 0; i < quant\_alt; i++)

{

delete[] interf[i];

}

delete[] interf;

for (int i = 0; i < quant\_alt; i++)

{

delete[] capacity[i];

}

delete[] capacity;

delete[] prioir\_eval;

for (int i = 0; i < quant\_crit; i++)

{

delete[] priority[i];

}

delete[] priority;

delete[] global\_crit;

return 0;

}

Запустим данную программу и введём данные (рисунок 3):

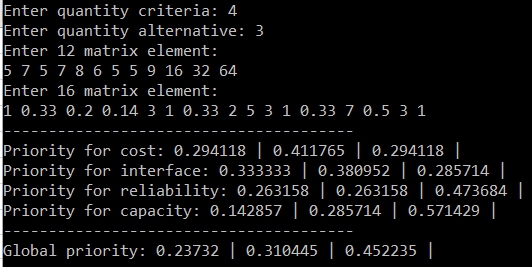


Рисунок 3 — Вывод программы

Таким образом задача была решена двумя способами с помощью МАИ. При решении ручным и программным способами получился один и тот же ответ, что говорит о правильности решения. Из полученных глобальных приоритетов видно, что Windows 10 набрала наибольший вес, а значит она подходит по заданным критериям больше остальных.

**Вывод**

В ходе выполнения лабораторной работы были углубленны теоретические знания в области системного анализа, приобретены навыки создания и описания иерархических структур, а также изучены такие понятия, как цель, критерий и альтернатива. В результате была с помощью метода анализа иерархий решена задача по выбору операционной системы для предприятия.