МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Розрахункова робота

З дисципліни

“Математичні методи дослідження операцій”

Виконав:

Студент групи КН-314

Ляшеник Остап

Прийняв

Шиманський Володимир Михайлович

Львів - 2023

Варіант 4

Лабораторна робота №1

ГРАФІЧНИЙ МЕТОД РОЗВ'ЯЗУВАННЯ

ЗАДАЧ ЛІНІЙНОГО ПРОГРАМУВАННЯ

Мета заняття: вивчення графічного методу розв'язування задач лінійного

програмування, що містять дві незалежні змінні.

Завдання:

Оптовий склад загальною площею S = 1000м^2 надає послуги зі зберігання вантажів двох типів А і Б. Для забезпечення належного зберігання склад має в наявності N = 500 одиниць складської тари. Зберігання тонни вантажу А потребує a1 (м^2) складських площ та b1 одиниць тари, зберігання тонни вантажу Б потребує a2 (м^2) складських площ та b2 одиниць тари. Прибуток складу на місяць від зберігання тонни вантажу А складає α грн., вантажу Б — β грн. Визначити, яку кількість кожного вантажу необхідно зберігати на складі, щоб отримати найбільший прибуток при виконанні додаткових умов:

варіанти 1-8: кількість вантажу А на складі повинна перевищувати кількість вантажу Б щонайменше на с тонн

Вар.	a_1	a_2	b_1	b_2	α	β	c
1	3	7	4	2	25	40	50
2	5	6	3	7	10	25	30
3	4	7	3	4	15	25	40
4	2	5	3	2	25	18	45

Складемо функцію цілі: $25x_1 + 18x_2 \rightarrow max$ (прибуток за місяць)

Обмеження:

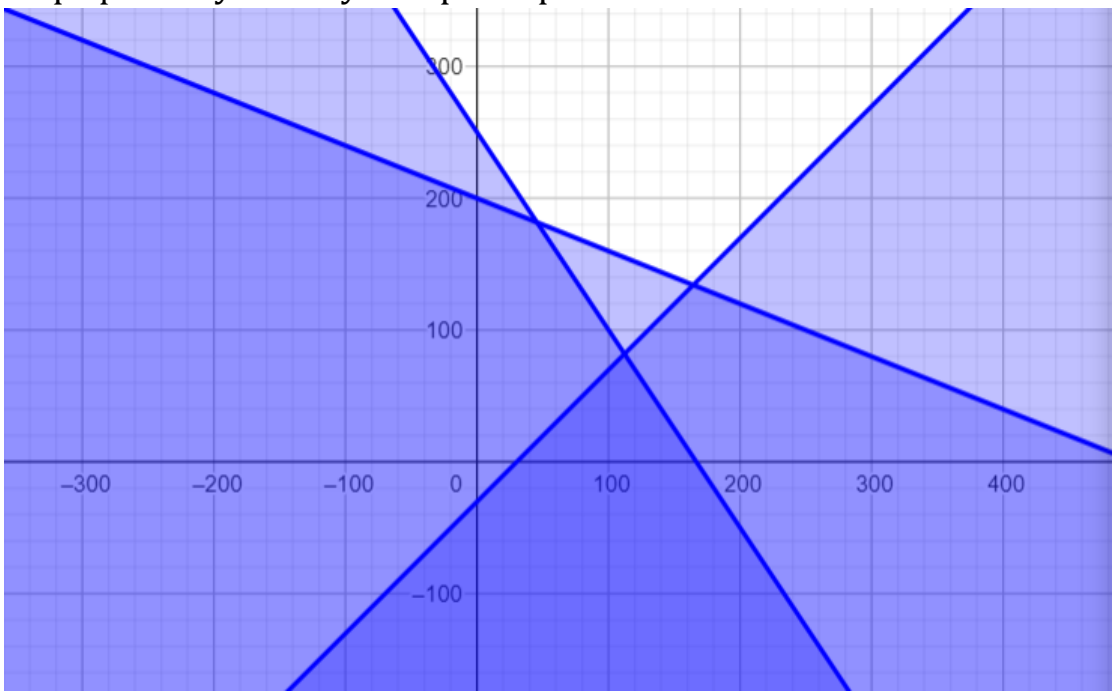
$2x_1 + 5x_2 \leq 1000$ (на площу на складі)

$3x_1 + 2x_2 \leq 500$ (на упаковку)

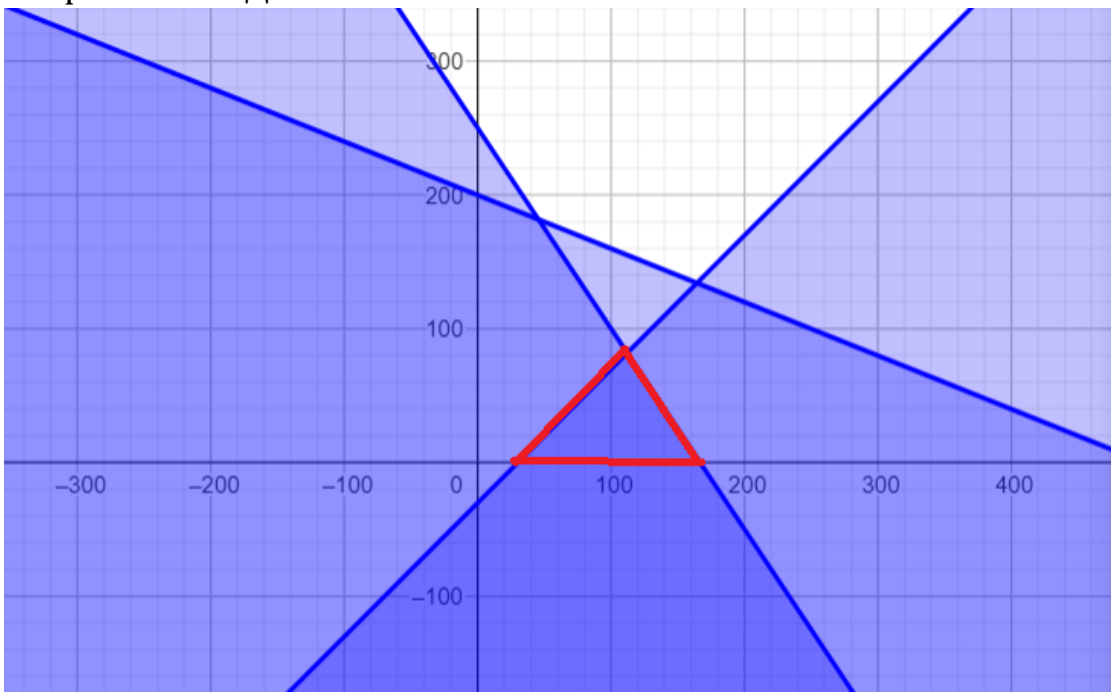
$x_1 - x_2 \geq 30$

$x_1 \geq 0, x_2 \geq 0$

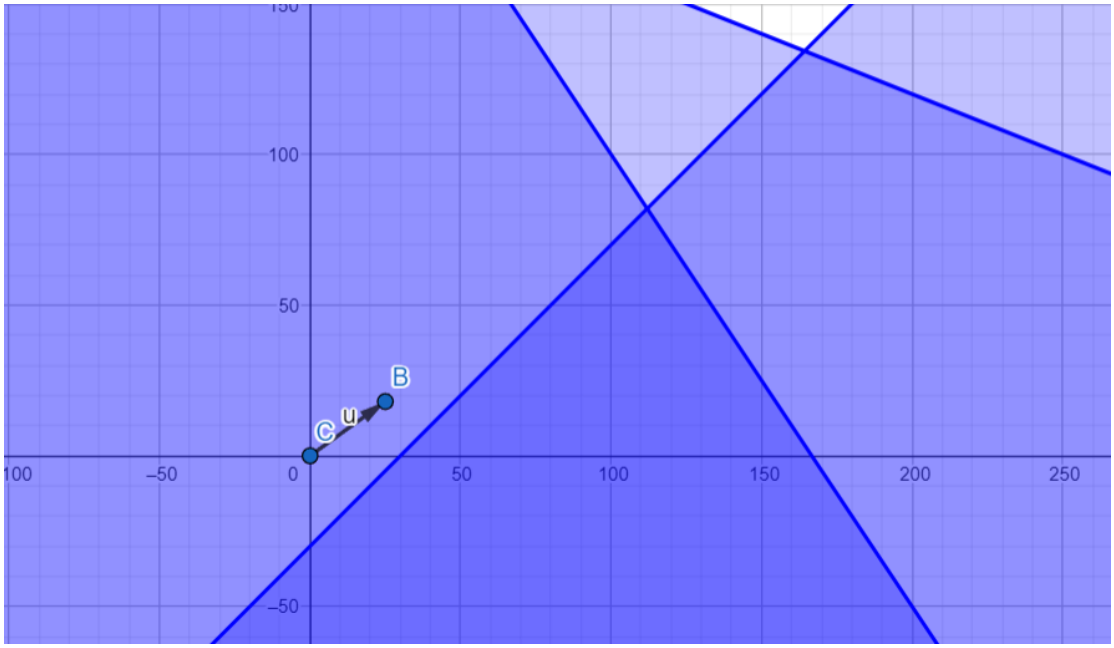
В графічному калькуляторі зображаємо обмеження



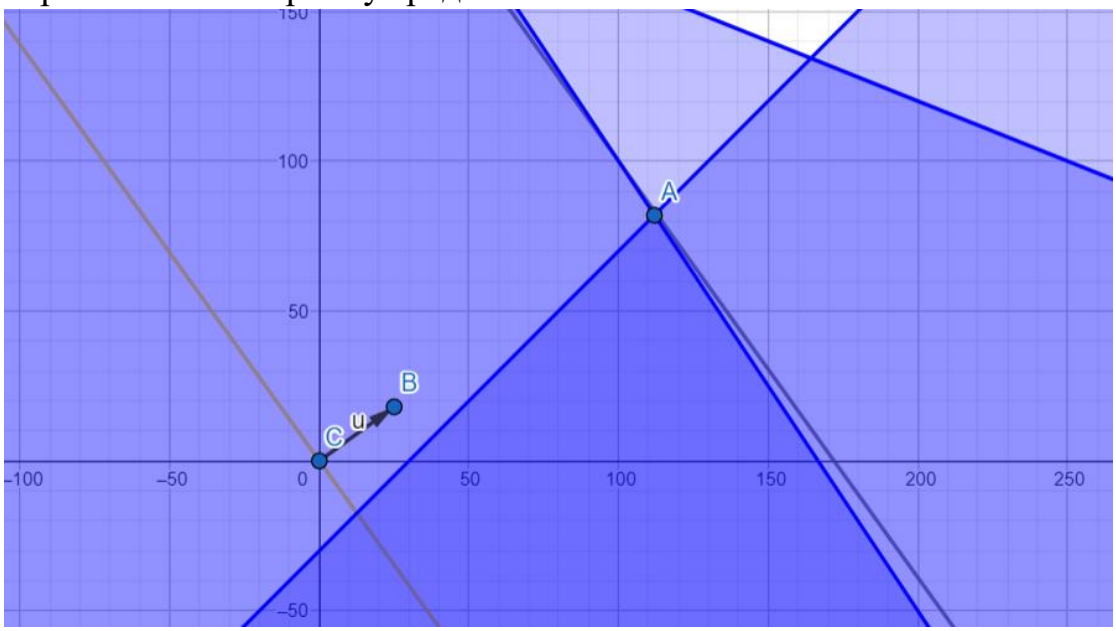
Зображаємо ОДЗ



Вектор градієнту (знаходимо як часткові похідні функції цілі)
A(25;18)



Будуємо пряму, перпендикулярну до вектора і паралельно переносимо в напрямку градієнта



Точка мінімуму: А - перетин 3 x+2 y=500 з x-y=30. A(112; 82)
Значення функції = 28*112+18*82 = 4612 грн прибутку

Лабораторна робота №2

СИМПЛЕКС-МЕТОД РОЗВ’ЯЗУВАННЯ ЗАДАЧІ ЛІНІЙНОГО

ПРОГРАМУВАННЯ ЗА НАЯВНОСТІ ПОЧАТКОВОГО ДОПУСТИМОГО

БАЗИСНОГО РІШЕННЯ

Мета заняття: вивчення алгоритму симплекс-методу розв’язування задачі

лінійного програмування за наявності початкового допустимого базисного рішення.

Вар.	a_1	a_2	a_3	b_1	b_2	b_3	c_1	c_2	c_3	α	β	γ
1	3	5	2	6	5	1	2	8	3	4	5	10
2	7	3	1	3	2	2	4	4	3	6	7	10
3	4	5	1	6	5	3	2	6	1	7	9	6
4	9	5	0	5	3	2	4	6	3	15	8	9

Код програми

```
import numpy as np


n, m = 3, 3


a = np.array([
    [9, 5, 0],
```

```
    [5, 3, 3],
    [2, 4, 6]
], dtype=float)

plan = np.array([200, 400, 90], dtype=float)
coefs = np.array([15, 8, 9], dtype=float)
basis_id = np.array([0, 1, 2], dtype=int)

max_iterations = 100
tolerance = 1e-5
for iteration_counter in range(max_iterations):
    print(f"Iteration {iteration_counter + 1}")
    reduced_costs = coefs - np.dot(coefs[basis_id], a)

    if np.all(reduced_costs <= tolerance):
        print("Optimal solution found.")
        break

    entering = np.argmin(reduced_costs)

    if all(a[:, entering] <= 0):
        print("Problem is unbounded.")
        break

    ratios = np.divide(plan, a[:, entering], out=np.full(n, np.inf), where=a[:, entering] > tolerance)
    leaving = np.argmin(ratios)

    pivot = a[leaving, entering]
    a[leaving] /= pivot
    plan[leaving] /= pivot
    for i in range(n):
        if i != leaving:
            factor = a[i, entering]
            a[i] -= factor * a[leaving]
            plan[i] -= factor * plan[leaving]
```

```
    basis_id[leaving] = entering

else:
    print("Maximum iterations reached. The solution may not be optimal.")

final_solution = np.zeros(m)
final_solution[basis_id] = plan
RESULT = np.dot(coefs, final_solution)
print("Final solution:")
print(final_solution)
print(f"Maximum of objective function: {RESULT:.2f}")
```

Результат обрахунку

```
Iteration 1
Optimal solution found.
Final solution:
[200. 400.  90.]
Maximum of objective function: 7010.00
```

Лабораторна робота №6

ТРАНСПОРТНА ЗАДАЧА ЛІНІЙНОГО ПРОГРАМУВАНЯ

ЗА КРИТЕРІЄМ ВАРТОСТІ ПЕРЕВЕЗЕНЬ

Мета заняття: вивчення методу рішення транспортної задачі лінійного програмування за критерієм вартості перевезень методом потенціалів.

Варіант	Потреба у піску підприємств, *m*				
	$\mathbf{B}_1$	$\mathbf{B}_2$	$\mathbf{B}_3$	$\mathbf{B}_4$	$\mathbf{B}_5$
1	90	100	70	130	110
2	160	200	100	185	-
3	180	140	190	120	170
4	200	100	145	185	150

Вар.	Варіант матриці відстаней	Запаси піску на складах, *m*				
		A_1	A_2	A_3	A_4	A_5
1	1	200	150	150	–	–
2	2	120	95	180	150	100
3	3	300	280	220	–	–
4	4	180	250	105	150	95

Таблиця 6.7 — Матриця відстаней, км (варіант 4)

Кар'єри	Підприємства				
	B_1	B_2	B_3	B_4	B_5
A_1	12	16	21	19	32
A_2	4	4	9	5	24
A_3	3	8	14	10	26
A_4	24	33	36	34	49
A_5	9	25	30	20	31

Код програми:

```
#include <iostream>

using namespace std;

const int n(5), m(5);

int** cycle_runner(bool mask[n][m], int start[2], int last[2], int
current[2], int l, int** way, int& size){

    int** local_way = new int*[l+1];

    for(int i = 0; i < l; i++){

        local_way[i] = new int[2];

        local_way[i][0] = way[i][0];
```

```
    local_way[i][1] = way[i][1];

}

local_way[l] = new int[2];

local_way[l][0] = current[0];

local_way[l][1] = current[1];

if(last[0] == current[0] || l == 0){

    for(int i = 0; i < n; i++){

        if(i == start[0] && current[1] == start[1] && l > 1) {

            size = l+1;

            return local_way;

        }

        if(mask[i][current[1]] == 1 && i != current[0]){

            int next_cords[2] {i, current[1]};
```

```
            int** tmp_arr = cycle_runner(mask, start, current, next_cords,
l+1, local_way, size);

            if(tmp_arr != nullptr)

                return tmp_arr;

        }

    }

    return nullptr;

}

if(last[1] == current[1] || l == 0){

    for(int i = 0; i < m; i++){

        if(i == start[1] && current[0] == start[0] && l > 1) {

            size = l+1;

            return local_way;

        }

        if(mask[current[0]][i] == 1 && i != current[1]){
```

```
            int next_cords[2] {current[0], i};

            int** tmp_arr = cycle_runner(mask, start, current, next_cords,
l+1, local_way, size);

            if(tmp_arr != nullptr)

                return tmp_arr;

        }

    }

    return nullptr;

  }

}

int main(){

  int c[n][m] = {

        {12, 16, 21, 19, 32},

        {4, 4, 9, 5, 24},
```

```
        {3, 8, 14, 10, 26},

        {24, 33, 36, 34, 49},

        {9, 25, 30, 20, 31}

};



int a[n] = {180, 250, 105, 150, 95};

int b[m] = {200, 100, 145, 185, 150};



int summ = 0;

for(int i = 0; i < n; i++){

    summ += a[i];

}

for(int i = 0; i < m; i++){

    summ -= b[i];
```

```
}

if(summ) {

    cout << "problem is unsolvable" << endl;

    return 0;

}

int potentials[n][m]{0};

bool potentials_mask[n][m]{0};

int i_(0), j_(0);

while (true) {

    if (a[i_] == 0) ++i_;

    else if (b[j_] == 0) ++j_;

    if (i_ == n || j_ == m)

        break;
```

```
    if (a[i_] >= b[j_]) {
        a[i_] -= b[j_];
        potentials[i_][j_] = b[j_];
        b[j_] = 0;
    } else {
        b[j_] -= a[i_];
        potentials[i_][j_] = a[i_];
        a[i_] = 0;
    }
    potentials_mask[i_][j_] = 1;
}
int alpha[n]{0};
int betha[m]{0};
bool alpha_mask[n] {0};
```

```
bool betha_mask[n] {0};

while(true){

    for(int i = 0; i < n; i++){

        alpha_mask[i] = 0;

    }

    alpha_mask[0] = 1;

    for(int i = 0; i < m; i++){

        betha_mask[i] = 0;

    }

    while(true) {

        for (int i = 0; i < n; i++) {

            for (int j = 0; j < m; j++) {

                if (potentials_mask[i][j] == 1) {
```

```
            if (betha_mask[j] == 0 && alpha_mask[i]) {
                betha_mask[j] = 1;
                betha[j] = c[i][j] - alpha[i];
            } else if (alpha_mask[i] == 0 && betha_mask[j]) {
                alpha_mask[i] = 1;
                alpha[i] = c[i][j] - betha[j];
            }
        }
    }
}
bool braker = 1;
for(int i = 0; i < n; i++){
    if(!alpha_mask[i])
        braker = 0;
```

```
    }

    for(int i = 0; i < m; i++){

        if(!betha_mask[i])

            braker = 0;

    }

    if(braker)

        break;

}

int highest_empty_potential = 0;

int h_i, h_j;

for(int i = 0; i < n; i++){

    for(int j = 0; j < m; j++){

        if(potentials_mask[i][j] == 0){
```

```
            potentials[i][j] = alpha[i]+betha[j]-c[i][j];

            if(potentials[i][j] > highest_empty_potential){

                highest_empty_potential = potentials[i][j];

                h_i = i;

                h_j = j;

            }

        }

    }

}

for(int i = 0; i < n; i++){

    cout << alpha[i] << '\t';

}

cout << endl;

for(int i = 0; i < n; i++){
```

```
    cout << betha[i] << '\t';

}

cout << endl;

for(int i = 0; i < n; i++){

    for(int j = 0; j < m; j++){

        if(potentials_mask[i][j])

            cout << "[" << potentials[i][j] << "]\t";

        else

            cout << potentials[i][j] << '\t';

    }

    cout << endl;

}

cout << endl;
```

```
        if(highest_empty_potential <= 0){

            break;

        }

        int start_cords[2] {h_i, h_j};

        int len;

        int** cycle_cords = cycle_runner(potentials_mask, start_cords,
start_cords, start_cords, 0, nullptr, len);

        if(cycle_cords == nullptr){

            cout << "Unable to solve" << endl;

            break;

        }

        for(int i = 0; i < len; i++){

            cout << cycle_cords[i][0]+1 << " " << cycle_cords[i][1]+1 <<
endl;

        }
```

```
        cout << endl;

        cout << "----------------------" << endl;

        int min_potential = 1000000;

        int l_i, l_j;

        for(int i = 1; i < len; i+=2){

            if(potentials[cycle_cords[i][0]][cycle_cords[i][1]] <=
min_potential){

                min_potential =
potentials[cycle_cords[i][0]][cycle_cords[i][1]];

                l_i = cycle_cords[i][0];

                l_j = cycle_cords[i][1];

            }

        }

        potentials_mask[h_i][h_j] = 1;

        potentials_mask[l_i][l_j] = 0;
```

```
        potentials[h_i][h_j] = min_potential;

        for(int i = 1; i < len; i++){

            if(cycle_cords[i][0] != l_i || cycle_cords[i][1] != l_j){

                if(i % 2){

                    potentials[cycle_cords[i][0]][cycle_cords[i][1]] -=
min_potential;

                }else{

                    potentials[cycle_cords[i][0]][cycle_cords[i][1]] +=
min_potential;

                }

            }

        }

    }

    int sum = 0;

    for(int i = 0; i < n; i++){
```

```
        for(int j = 0; j < m; j++){

            if(potentials_mask[i][j])

                sum += potentials[i][j]*c[i][j];

        }

    }

    cout << "RESULT: " << sum;

}
```

Результат:

```
-6      -1      [65]    [185]   -4
[5]     [100]   0       0       -1
[150]   -4      -1      -3      -3
[45]    -11     -10     -4      [50]

RESULT: 12760
```

Лабораторна робота №8

ЗАДАЧА ПРО ЗАВАНТАЖЕННЯ ТРАНСПОРТНОГО ЗАСОБУ

Мета заняття: вивчення застосування методу динамічного програмуваннядля розв’язування задачі про оптимальне

завантаження транспортного засобу.

4. $\begin{bmatrix} 3,1 & 3,8 & 4,0 & 2,9 \\ 3,3 & 3,9 & 4,3 & 3,0 \\ 3,9 & 4,5 & 4,7 & 4,9 \\ 5,0 & 4,6 & 5,1 & 5,7 \\ 5,3 & 5,2 & 5,3 & 6,5 \end{bmatrix}$

Код програми:

```
import numpy as np


def f__(f_, matrix):
    for i in range(len(f_)):
        f_[i][0] = np.amax(matrix[i])
        index = np.where(matrix[i] == np.amax(matrix[i]))
        f_[i][1] = index[0][0]


def solve(table, invest):
    iter = len(table)
    inv = invest
    matrix = np.array([[0] * (invest + 1) for i in range(invest + 1)],
dtype=float)
    f_ = np.array([[[0, 0] for i in range(invest + 1)] for j in range(iter+1)],
dtype=float)
    res = []
    while iter > 0:
        for x in range(invest+1):
            for k in range(invest+1):
                if k <= x:
                    lst_ = [0]
```

```
                for i in table[iter - 1]:
                    if i[0] == k:
                        lst_.append(i[1])
                matrix[x][k] = max(lst_) + f_[iter][x-k][0]
        f__(f_[iter-1], matrix)
        iter -= 1

    f_ = np.delete(f_, -1, axis=0)
    for i in range(len(f_)):
        t = f_[i][inv][1]
        res.append((i + 1, t))
        inv -= int(t)
    return res, np.amax(matrix[-1])

invest = 20
table = [
    [[4, 3.1], [8, 3.8], [12, 4.5], [16, 4.9], [20, 5.3]],
    [[4, 3.3], [8, 3.9], [12, 4.7], [16, 5.1], [20, 5.7]],
    [[4, 4.0], [8, 4.3], [12, 4.9], [16, 5.5], [20, 6.5]],
    [[4, 3.0], [8, 3.5], [12, 4.0], [16, 4.5], [20, 5.0]]
]

res, incomes = solve(table, invest)

for i, j in res:
    print(f"Factory {i} - {j} cars")
```

print(f"Income {incomes:.2f} ")

```
Factory 1 - 8.0 cars
Factory 2 - 4.0 cars
Factory 3 - 4.0 cars
Factory 4 - 4.0 cars
Income 14.10
```

Лабораторна робота №9

СИСТЕМИ МАСОВОГО ОБСЛУГОВУВАННЯ

З ГРУПОВИМ НАДХОДЖЕННЯМ ВИМОГ

Мета заняття: засвоєння розрахунку основних характеристик функціонування розімкненої багатоканальної системи масового обслуговування з очікуванням та груповим надходженням вимог.

Вар.	λ	μ	n	m
1	3	3	6	4
2	2	3	5	3
3	4	2	7	3
4	3	4	6	4

Код програми:

```
import math


n = 6

m = 4

k = 0

l = 3

nu = 4

fk = [0] * 500

pk = [0] * 500

alfa = l / nu

sumfk = 1
```

```
p0 = 1
toc = 0

fk[k] = 1.0

while fk[k] >= 0.01:
    k += 1
    if k < m:
        fk[k] = ((l + (k - 1) * nu) * fk[k - 1]) / (k * nu)
    if k >= m and k <= n:
        fk[k] = ((l + (k - 1) * nu) * fk[k - 1]) / (k * nu) - (l * fk[k - m]) / (k * nu)
    if k > n:
        fk[k] = ((l + n * nu) * fk[k - 1]) / (n * nu) - (l * fk[k - m]) / (n * nu)
    sumfk += fk[k]

p0 = 1 / sumfk
Mc = 0
Moc = 0
No = 0

for i in range(k):
    pk[i] = fk[i] * p0
    Mc += i * pk[i]
    if i > n:
        Moc += (i - n) * pk[i]
    if i < n:
        No += (n - i) * pk[i]

print(f"Moc: {Moc:.3f}")
print(f"Mc: {Mc:.3f}")
print(f"No: {No:.3f}")

toc = (Moc / l) * 24
```

```
print(f"Toc: {toc:.3f} hours")
```

Результат: