МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ



**Дніпровський національний університет  
залізничного транспорту імені академіка В. Лазаряна**

Кафедра «Комп’ютерні інформаційні технології»

**Лабораторна робота №1**

**з дисципліни «Основи програмної інженерії»**

**на тему: *«*Модульне програмування. Розробка програми методом низхідного програмування.»**

Виконав: студент гр. ПЗ2011

Горбань М.О.

Прийняла: к.т.н., доцент кафедри КІТ: Горбова О.В.

Дніпро, 2021

**Тема.** Модульне програмування. Розробка програми методом низхідного програмування.

**Мета роботи.** Отримати практичні навички розробки програм за методом низхідного програмування.

**Завдання.**

Розробіть структурну схему розв’язання задачі з обробки матриці методом покрокової деталізації (див. приклад на рис. 2) згідно індивідуального завдання.

За схемою розробіть програму мовою С++.

Вимоги до написання програми:

* матрицю представити у вигляді структури (struct);
* кожну принципово нову дію представити у вигляді функції. Обов’язковим є такий набір функцій:
  + main – головна функція програми;
  + create – створення матриці (зазначення розмірності, виділення пам’яті);
  + delete – очищення пам’яті, яку займає матриця;
  + processing – обробка матриці, не повинна містити операторів введення та/або виведення;
  + manualFilling, randomFilling – заповнення матриці з клавіатури та генератором випадкових чисел відповідно;
  + show – виведення матриці на екран;

– передбачити можливість введення розмірності матриці та вибору способу її заповнення (з клавіатури, за допомогою генератора випадкових чисел);

* функції обробки матриці прокоментувати, виходячи з методу покрокової деталізації;
* забороняється використовувати допоміжні масиви та матриці.

Індивідуальне завдання:

6. В рядках, в яких сума першого і останнього елементів перевищує значення діагонального елемента, поміняти місцями пари елементів (перший з другим, третій з четвертим і так далі).

**Структурна схема розв’язання завдання**

****

**Текст програми**

Файл Source.cpp

#include "Func.h"

using namespace std;

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

Matrix array;

input(array);

create(array);

cout << "Виберіть спосіб заповнення матриці:" << endl;

cout << "1. Заповнення з клавіатури." << endl;

cout << "2. Заповненян генератором випадкових чисел." << endl;

int choise;

cin >> choise;

switch (choise)

{

case 1:

manualFilling(array);

break;

case 2:

randomFilling(array);

break;

default:

cout << "Такого пункту немає!" << endl;

}

show(array);

processing(array);

cout << endl << endl;

show(array);

delete\_matrix(array);

}

Файл func.h

#pragma once

#include <windows.h>

#include <iostream>

#include <iomanip>

#include <ctime>

struct Matrix

{

int row;

int column;

int\*\* arr;

};

void create(Matrix& array);

void input(Matrix& array);

void delete\_matrix(Matrix& array);

void manualFilling(Matrix& array);

void randomFilling(Matrix& array);

void show(Matrix& array);

void processing(Matrix& array);

Файл func.cpp

#include "func.h"

using namespace std;

// Введення рядків та стовпців матриці

void input(Matrix& array)

{

int rows, columns;

cout << "Введіть кількість рядків: "; cin >> array.row;

cout << "Введіть кількість стовпців: "; cin >> array.column;

}

// Створення матриці

void create(Matrix& array)

{

array.arr = new int\* [array.row];

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

{

array.arr[i] = new int[array.column];

}

}

// звільнення пам'яті

void delete\_matrix(Matrix& array)

{

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

{

delete[] array.arr[i];

}

delete[] array.arr;

}

//ручне заповнення матриці

void manualFilling(Matrix& array)

{

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

{

for (int j = 0; j < array.column; j++)

{

cout << "array" << '[' << i << ']' << '[' << j << "]= ";

cin >> array.arr[i][j];

cout << endl;

}

}

}

//заповнення генератором

void randomFilling(Matrix& array)

{

srand(time(NULL));

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

{

for (int j = 0; j < array.column; j++)

{

array.arr[i][j] = rand() % 2 + 0;

}

}

}

//Друк матриці

void show(Matrix& array)

{

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

{

for (int j = 0; j < array.column; j++)

{

cout << setw(4) << array.arr[i][j] << ' ';

}

cout << endl << endl;

}

}

//порівняння сум першого і останнього елемента з діагональним

bool calculateSum(Matrix& array, int row)

{

int sum = array.arr[row][0] + array.arr[row][array.column - 1];

if (sum > array.arr[row][row])

return 1;

else return 0;

}

//Обробка матриці

void processing(Matrix& array)

{

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

{

bool swaping = calculateSum(array, i);

if (swaping)

{

for (int j = 0; j < array.column - 1; j+=2)

{

swap(array.arr[i][j], array.arr[i][j + 1]);

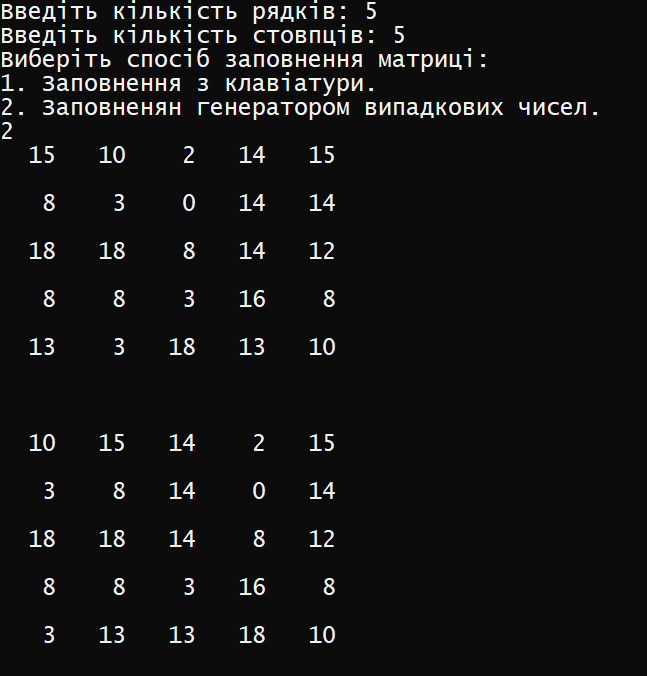
}

}

}

}

**Результат роботи програми**

****

На скріншоты видно результат роботи програми. Програма порівнює суму першого і останнього елемента матриці з діагональним елементом та в разі, якщо сума більша ніж діагональний елемент, міняє парні елементи у рядку місцями.

**Висновок**

Під час виконання лабораторної роботи познайомився з методом низхідного програмування, суть якого заключається в розбитті одної великої задачі на менші, більш прості задачі. Ці задачі називаються модулями. У хорошому коді зв'язок між модулями повинен бути мінімальним, а в середині самого модуля – максимальним.