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

Кулик Сергій Вадимович

Прийняла: доц. Нежуміра О.І.

Дніпро, 2020

Тема. Модульне програмування. Розробка програми методом низхідного програмування.

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

**1. 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:

14. Знайти суму додатних елементів лівої половини матриці і суму від’ємних правої половини. Визначити яка з них більша за модулем.

**2. структурна схема розв’язання завдання**

****

**3. Текст программи**

Файл Source.cpp

//Программа считает суму положительных елементов левой части матрици

//и суму отрицательных правой части. Определяет наибольшую из сум по модулю.

//Вход: количество строк и столбцов матрицы.

//Выход: Наибольшая сума по модулю.

#include <iostream>

#include "func.h"

using namespace std;

int main()

{

setlocale(LC\_ALL, "ru");

Matrix array;

create(array);

cout << "Выберете способ заполнения матрицы:" << endl;

cout << "1. Ручное заполнение." << endl;

cout << "2. Заполнение случайными числами." << endl;

int choise;

cin >> choise;

switch (choise)

{

case 1:

manualFilling(array);

break;

case 2:

randomFilling(array);

break;

default:

cout << "Вы ввели неправильное значение для заполнения матрицы!" << endl;

}

show(array);

cout <<"Сума положительных елементов левой части: матрицы: "<<leftMatrix(array) << endl;

cout << "Сума отрицательных елементов правой части матрицы: " << rightMatrix(array) << endl;

cout << "Наибольшая сума по модулю: " << processing(array) << endl;

delete\_matrix(array);

}

Файл func.h

#pragma once

struct Matrix

{

int row;

int column;

int\*\* arr;

};

void create(Matrix &array);

void delete\_matrix(Matrix& array);

void manualFilling(Matrix& array);

void randomFilling(Matrix& array);

void show(Matrix& array);

int leftMatrix(Matrix& array);

int rightMatrix(Matrix& array);

int processing(Matrix& array);

Файл func.cpp

#include "func.h"

#include <iostream>

#include <iomanip>

using namespace std;

/// <summary>

/// Создание матрицы

/// </summary>

/// <param name="array">Масив в виде струкстуры</param>

void create(Matrix &array)

{

cout << "\nВведите количество строк: ";

cin >> array.row;

cout << "\nВведите количество столбцов: ";

cin >> array.column;

if (array.column < 2)

{

cout << "Матрицу с такими размерами нельзя разделить на 2 части"<<endl;

exit(0);

}

else

{

cout << endl;

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

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

{

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

}

}

}

/// <summary>

/// Отчистка памяти

/// </summary>

/// <param name="array">Масив виде структуры</param>

void delete\_matrix(Matrix& array)

{

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

{

delete[] array.arr[i];

}

delete[] array.arr;

}

/// <summary>

/// Ручное заполнение матрицы

/// </summary>

/// <param name="array">Масив виде структуры</param>

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];

getc;

cout << endl;

}

}

}

/// <summary>

/// Заполнение матрицы случайными числами

/// </summary>

/// <param name="array">Масив виде структуры</param>

void randomFilling(Matrix& array)

{

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

{

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

{

array.arr[i][j] = rand() % 100 + (-50);

}

}

}

/// <summary>

/// Вывод матрицы на экран

/// </summary>

/// <param name="array">Масив виде структуры</param>

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;

}

}

/// <summary>

/// Сума положительных елементов левой части матрицы

/// </summary>

/// <param name="array">Масив виде структуры</param>

/// <returns>Возвращает суму положительных чисел левой части матрицы</returns>

int leftMatrix(Matrix& array)

{

int sum\_left = 0;

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

{

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

{

if (array.arr[i][j] > 0)

sum\_left += array.arr[i][j];

}

}

return sum\_left;

}

/// <summary>

/// Сума отрицательных елементов првой части матрицы

/// </summary>

/// <param name="array">Масив виде структуры</param>

/// <returns>Возвращает суму отрицательных чисел правой части матрицы</returns>

int rightMatrix(Matrix& array)

{

int sum\_right = 0;

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

{

if (array.column % 2 != 0)

{

for (int j = array.column - 1; j > array.column / 2; j--)

{

if (array.arr[i][j] < 0)

sum\_right += array.arr[i][j];

}

}

else

{

for (int j = array.column - 1; j >= array.column / 2; j--)

{

if (array.arr[i][j] < 0)

sum\_right += array.arr[i][j];

}

}

}

return sum\_right;

}

/// <summary>

/// Сравнение сум по модулю

/// </summary>

/// <param name="array">Масив виде структуры</param>

/// <returns>Возвращает наибольшую суму по модулю</returns>

int processing(Matrix& array)

{

int sum1 = leftMatrix(array);

int sum2 = rightMatrix(array);

if (sum1 > abs(sum2))

{

return sum1;

}

else

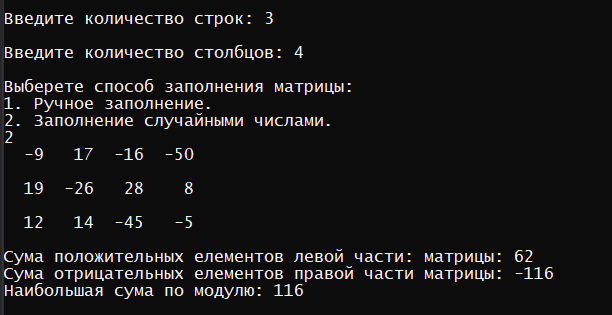
{

return abs(sum2);

}

}

**4. Результат роботи програми**



На скріншоты видно результат роботи програми. Програма вірно рахує суми з лівої та правої частини матриці та виводить більшу з них на екран.

**5. Висновок**

Під час виконання лабораторної роботи розробляв програму методом покрокової деталізації. Його суть полягає в розбиванні складної задачі на більш прості підзадачі(модулі). Завдяки цьому я можу коли завгодно вносити правки до коду не порушуючи правильності роботи програми. Для цього зв'зок між модулями повинен бути мінімальним.