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.ПЗ1911

Сафонов Д.Є.

Прийняла:

Куроп'ятник О. С.

Дніпро, 2020

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

**Мета роботи.** Отримати практичні навички розробки програм за методом низхідного програмування.2) постановка задачі згідно з загальним та індивідуальним завданнями;

# **Схема модульної структури програми.**



Рисунок 1

# **Протокол розробки програми за методом низхідного програмування.**

Рівень 0:

Завдання: 17. Відсортуйте в матриці стовпці, що містять елементи, в запису яких є однакові цифри. Інші стовпці переверніть.

Вхідні данні:

− кількість рядків матриці;

− кількість стовпчиків матриці;

− елементи матриці.

Вихідні данні: матриця з відсортованими та/або перевернутими стовпчиками .

Рівень 1:

* Створення матриці(модуль 1);
* Обробка матриці(модуль 2);
* Виведення матриці(модуль 3);
* Видалення матриці(модуль 4);

Рівень 2:

Модуль 1:

* Введення розмірності матриці(модуль 1.1);
* В залежності від вибору користувача заповнення матриці ним, або програмою(модуль 1.2);

Модуль 2:

* Перевірка кожного рядка на наявність елементів, в запису яких є однакові цифри(модуль 2.1);
* Сортування стовпчика(модуль 2.2);
* Перевертання стовпчика(модуль 2.3);

Модулі 3, 4 деталізації не потребують.

Рівень 3:

Модуль 1.1 деталізації не потребує.

Модуль 1.2:

* Введення матриці користувачем(модуль 1.2.1);
* Заповнення матриці випадковими значеннями(модуль 1.2.2);

Модулі 2.1, 2.2, 2.3деталізації не потребують

# **Лістинги модулів програми.**

“main.cpp”

#include <iostream>

#include "matrix.h"

#include <stdexcept>

#include <windows.h>

#undef max

//#include <cmath>

#include <climits>

#include <sstream>

#include <string>

int main() {

std::cout << "This application lets user create/generate matrix of any size he wants, change values of elements in the matrix." << std::endl;

std::cout << "And executes next acts: checks every column if it has any elements with duplicate digits." << std::endl;

std::cout << "If its true - sorts the column, else flips it." << std::endl << std::endl;

int rowsAmount = inputNum(1, 32767, "Enter amount of rows: ");

int columnsAmount = inputNum(1, 32767, "Enter amount of columns: ");

char choice;

std::cout << "Do you want to enter matrix yourself(if not - app will genreate it itself)?(Y/N)";

std::cin >> choice;

while ((choice != 'Y' && choice != 'y' && choice != 'N' && choice != 'n') || (std::cin.peek() != '\n')) {//until correct input

while (std::cin.get() != '\n');//clear istream

std::cout << "wrong input, try again." << std::endl;

std::cout << "Do you want to enter matrix yourself(if not - app will genreate it itself)?(Y/N)";

std::cin >> choice;

}

double\*\* matrix2d = createMatrix<double>(rowsAmount, columnsAmount);

if (choice == 'N' || choice == 'n') {

genMatrix(matrix2d, rowsAmount, columnsAmount);

printMatrix(matrix2d, rowsAmount, columnsAmount);

}

else {

setMatrix0(matrix2d, rowsAmount, columnsAmount);

setMatrix(matrix2d, rowsAmount, columnsAmount);

}

std::cout << std::endl;

exercise(matrix2d, rowsAmount, columnsAmount);

printMatrix(matrix2d, rowsAmount, columnsAmount);

deleteMatrix(matrix2d, rowsAmount, columnsAmount);

system("pause");

return 0;

}

“matrix.h”

#ifndef \_\_MATRIX\_H\_\_

#define \_\_MATRIX\_H\_\_

#include <iostream>

#include <iomanip>

#include <random>

#include <chrono>

#include <bitset>

#include <string>

//allocate memory for matrix

template<typename T>

T\*\* createMatrix(int rowsAmount, int columnsAmount) {//allocate memory for matrix

T\*\* matrix2d = new T \* [rowsAmount];

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

matrix2d[i] = new T[columnsAmount];

return matrix2d;

}

//free up the memory

template<typename T>

void deleteMatrix(T\*\* matrix2d, int rowsAmount, int columnsAmount) {

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

delete[]matrix2d[i];

delete[]matrix2d;

}

//set random values for matrix

template<typename T>

void genMatrix(T\*\* matrix2d, int rowsAmount, int columnsAmount) {

typedef std::chrono::high\_resolution\_clock myclock;

myclock::time\_point beginning = myclock::now();

std::uniform\_int\_distribution<int> valueDistribution(0, 100);

myclock::duration d = myclock::now() - beginning;

unsigned seed = d.count();

std::default\_random\_engine generator(seed);

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

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

matrix2d[i][j] = valueDistribution(generator);

}

template<typename T>

void printMatrix(T\*\* matrix2d, int rowsAmount, int columnsAmount) {

std::cout << std::left;

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

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

std::cout << std::setw(12) << matrix2d[i][j] << " ";

std::cout << std::endl;

}

std::cout << std::right;

}

//sorts column using insertion sort algorithm

template <typename T>

void sortColumn(T\*\* matrix2d, int column, int rowsAmount) {

for (int i = 1; i < rowsAmount; i++)

if (matrix2d[i][column] < matrix2d[i - 1][column]) {//insertionSort

T temp = matrix2d[i][column];

int j = i - 1;

for (j; (j >= 0) && (temp < matrix2d[j][column]); j--)

matrix2d[j + 1][column] = matrix2d[j][column];//find position and shift to right elements between new and old position

matrix2d[j + 1][column] = temp;

}

}

template <typename T>

void flipColumn(T\*\* matrix2d, int column, int rowsAmount) {

for (int i = 0, j = rowsAmount - 1; i <= j; i++, j--) {

T temp = matrix2d[i][column];

matrix2d[i][column] = matrix2d[j][column];

matrix2d[j][column] = temp;

}

}

bool hasDuplicateDigits(int a) {

bool retValue = false;

a = abs(a);

if (a > 999999999)//>10digits, at least one is duplicate

retValue = true;

else if (a > 10) {//if >= 2digits

std::bitset<10> digitsUsed;

int currentDigit;

do {

currentDigit = a % 10;

a = (a - currentDigit) / 10;

if (digitsUsed[currentDigit])

retValue = true;

else

digitsUsed[currentDigit] = true;

} while (a > 0 && !retValue);

}

return retValue;

}

//checks if column has any elemnts with duplicate digits

template <typename T>

bool columnElementsDuplicateDigits(T\*\* matrix2d, int column, int rowsAmount) {

bool has = false;

T currentElement;

for (int i = 0; i < rowsAmount && !has; i++)//has = if matrix2d[i][column] has >= 2 same numerals

has = hasDuplicateDigits(matrix2d[i][column]);

return has;

}

//performs a check on every column and sorts it if it has any elements with duplicate digits, flips if not

template <typename T>

void exercise(T\*\* matrix2d, int rowsAmount, int columnsAmount) {

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

if (columnElementsDuplicateDigits(matrix2d, i, rowsAmount))

sortColumn(matrix2d, i, rowsAmount);

else

flipColumn(matrix2d, i, rowsAmount);

}

int inputNum(int leftLim, int rightLim, std::string invitation) {

std::string cinStr;//////////////////////////////////////////

int num = leftLim - 1;

size\_t\* pos = new size\_t{ 0 };

do {//input validation

std::cout << invitation;

std::cin >> cinStr;

if (std::cin.peek() != '\n') {//if istream isn't empty

while (std::cin.get() != '\n');//clear istream

std::cout << "incorrect format" << std::endl;

}

else {

\*pos = 0;//clear old

try {

num = std::stoi(cinStr, pos);//convert input to INT format, set \*pos value to length of used substring

}

catch (std::invalid\_argument) {

std::cout << "incorrect format" << std::endl;

num = leftLim - 1;

}

catch (std::out\_of\_range) {

std::cout << "out of range" << std::endl;

num = leftLim - 1;

}

if (\*pos != 0)

if (\*pos < cinStr.size()) {

std::cout << "incorrect format" << std::endl;

num = leftLim - 1;

}

else if (num < leftLim || num >= rightLim) {

std::cout << "out of range" << std::endl;

num = leftLim - 1;

}

}

} while (num < leftLim || num >= rightLim);

return num;

}

//lets user to chnge any elements by indexes

template <typename T>

void setMatrix(T\*\* matrix2d, int rowsAmount, int columnsAmount) {

system("cls");

printMatrix(matrix2d, rowsAmount, columnsAmount);

char choice;

std::cout << "Do you want to enter any elements?(Y/N)";

std::cin >> choice;

while ((choice != 'Y' && choice != 'y' && choice != 'N' && choice != 'n') || (std::cin.peek() != '\n')) {

while (std::cin.get() != '\n');//clear istream

std::cout << "wrong input, try again." << std::endl;

std::cout << "Do you want to enter any elements?(Y/N)";

std::cin >> choice;

}

while (choice != 'N' && choice != 'n') {

int i = inputNum(0, rowsAmount, "enter row of element you want to change: ");

int j = inputNum(0, columnsAmount, "enter column of element you want to change: ");

std::string elementName = "A[" + std::to\_string(i) + "][" + std::to\_string(j) + "] = ";

matrix2d[i][j] = inputNum(-32768, 32767, elementName);

system("cls");

printMatrix(matrix2d, rowsAmount, columnsAmount);

std::cout << "Do you want to enter any elements?(Y/N)";

std::cin >> choice;

while ((choice != 'Y' && choice != 'y' && choice != 'N' && choice != 'n') || (std::cin.peek() != '\n')) {

while (std::cin.get() != '\n');//clear istream

std::cout << "wrong input, try again." << std::endl;

std::cout << "Do you want to enter any more elements?(Y/N)";

std::cin >> choice;

}

}

}

//set all elements to 0

template <typename T>

void setMatrix0(T\*\* matrix2d, int rowsAmount, int columnsAmount) {

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

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

matrix2d[i][j] = 0;

}

#endif

# **Набори тестів для перевірки працездатності програми.**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Назва | Вхідні дані | Очікуваний результат |
| 1 | Перевірка сортування стовпчика. | Кількість стовпців: 5  Кількість рядків: 5  0; 17; 34; 67; 32  14; 34; 54; 0; 25  44; 7; 66; 7; 11  98; 88; 89; 33; 13  45; 84; 52; 18; 1 | 0; 7; 34; 0; 1  14; 17; 52; 7; 11  44; 34; 54; 18; 13  45; 84; 66; 33; 25  98; 88; 89; 67; 32  (усі стовпці відсортовані) |
| 2 | Перевірка перевертання стовпчика. | Кількість стовпців: 5  Кількість рядків: 5  0; 17; 34; 67; 32  14; 34; 54; 0; 25  4; 7; 6; 7; 12  98; 8; 89; 3; 13  45; 84; 52; 18; 1 | 45; 84; 52; 18; 1  98; 8; 89; 3; 13  4; 7; 6; 7; 12  14; 34; 54; 0; 25  0; 17; 34; 67; 32  (усі стовпці перевернуті) |
| 3 | Перевірка програми з обома типами стовпчиків. | Кількість стовпців: 5  Кількість рядків: 5  0; 17; 34; 67; 32  14; 34; 54; 0; 25  44; 7; 66; 7; 12  98; 8; 89; 3; 13  45; 84; 52; 18; 11 | 0; 84; 34; 18; 11  14; 8; 52; 3; 12  44; 7; 54; 7; 13  45; 34; 66; 0; 25  98; 17; 89; 67; 32  (стопці 0, 2, 4 – відсортовані; 1, 3 – перевернуті) |

# **Результати виконання програми та їх аналіз.**

Перевірка вводу коректності вводу(рис.1, рис.2, рис.3, рис.4, рис.5, рис.6):

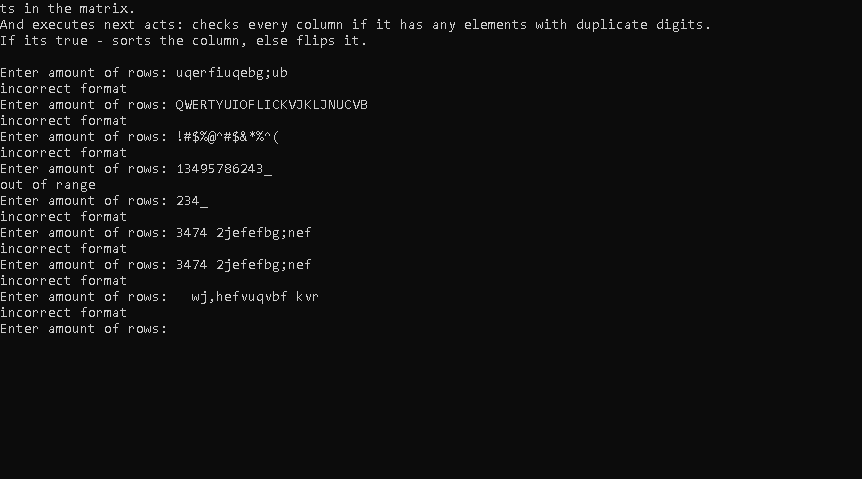


Рисунок 2

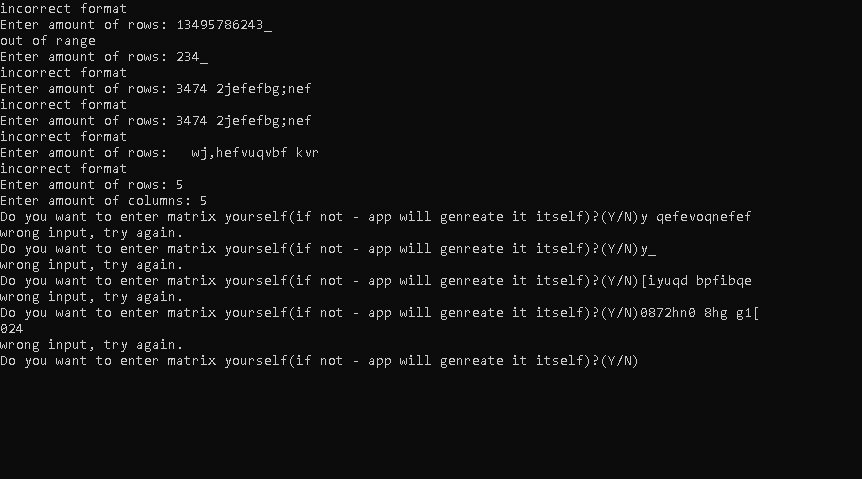


Рисунок 3

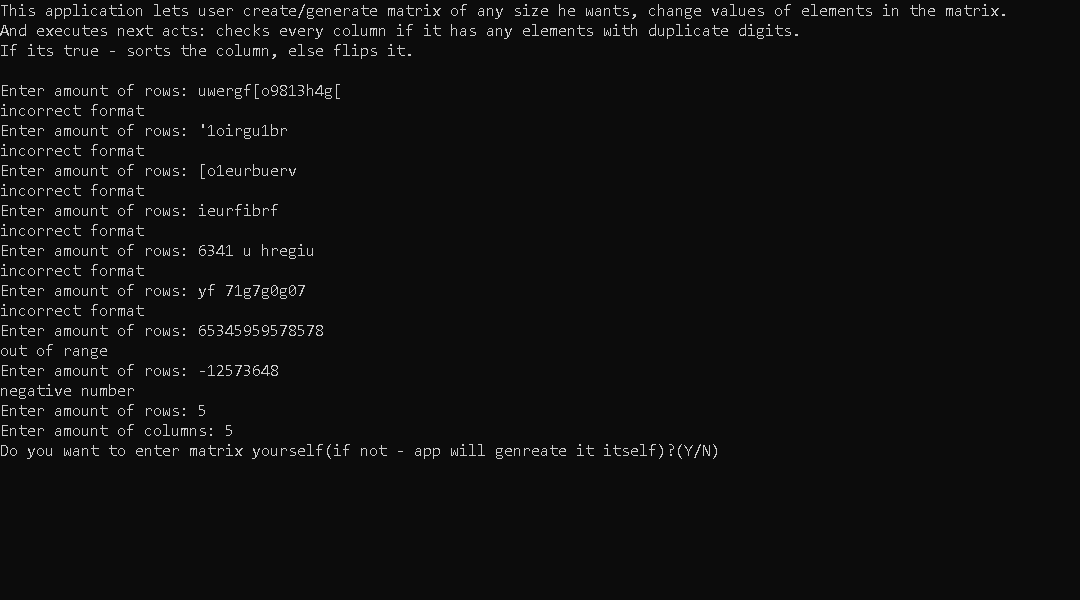


Рисунок 4

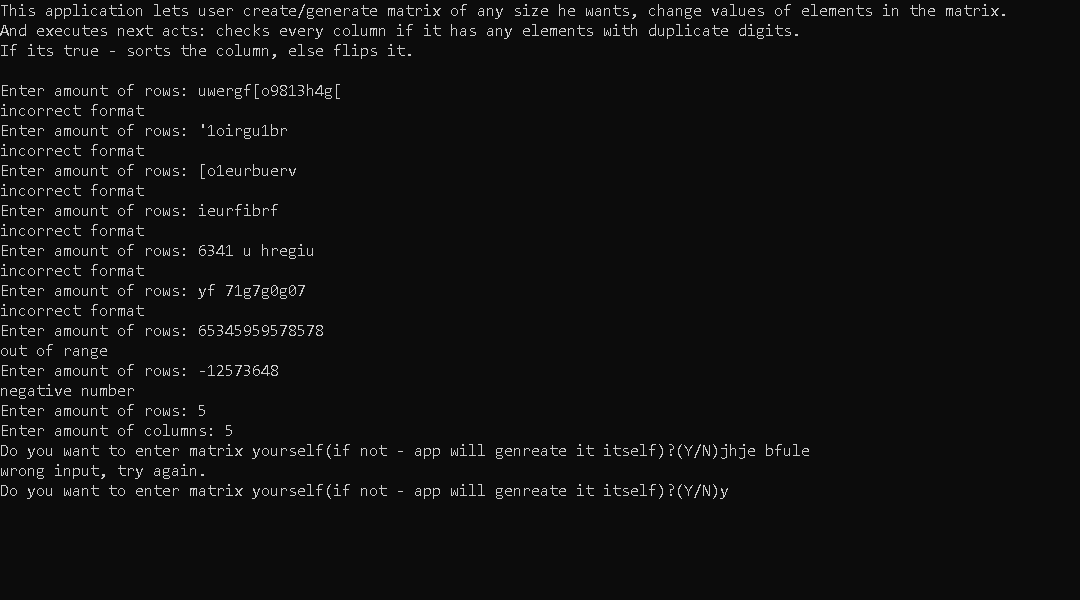


Рисунок 5

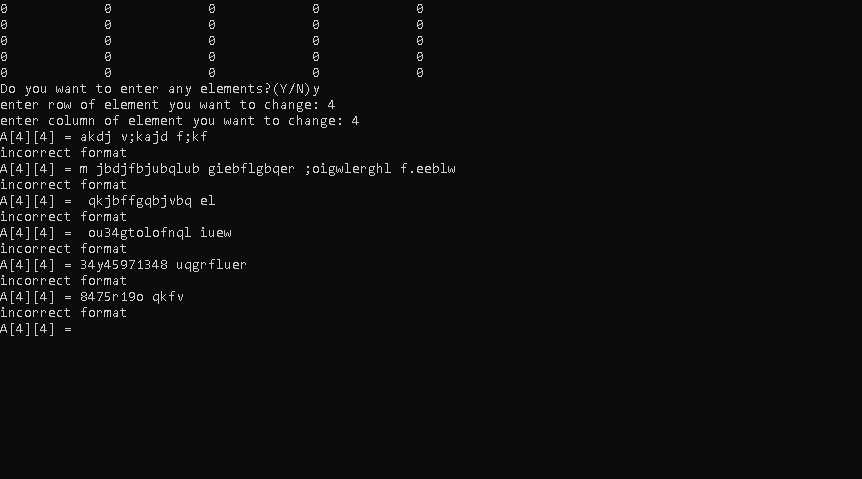
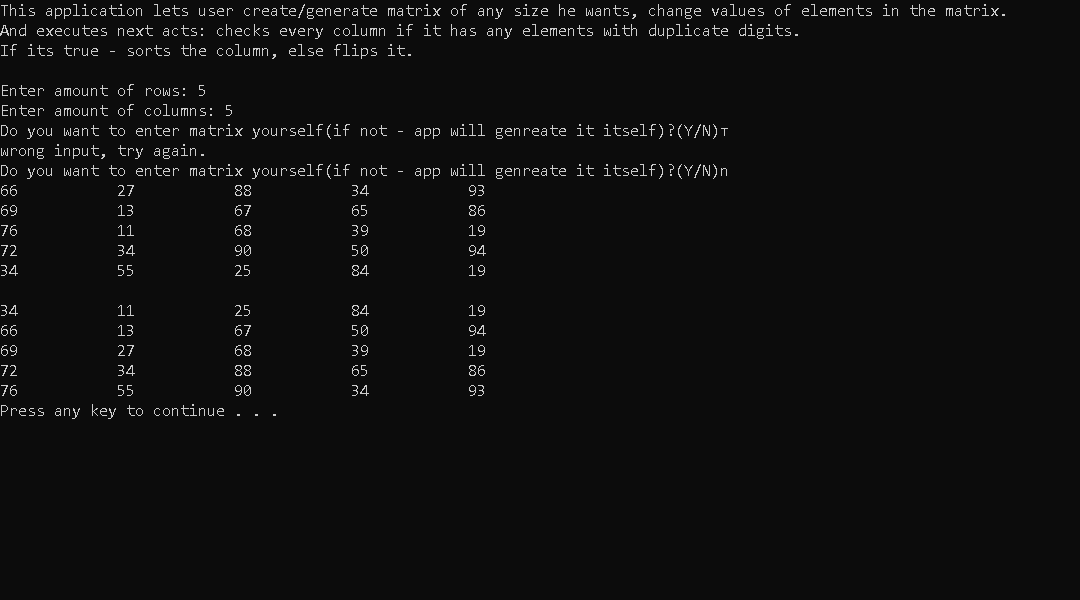


Рисунок 6

Перевірка генерації матриці(рис.7)

Рисунок 7

Тести:

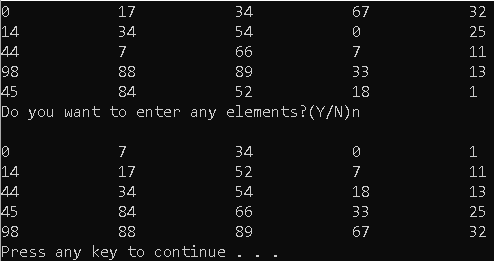


Рисунок 8(Тест 1)

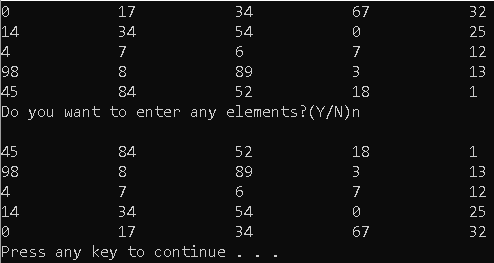


Рисунок 9(Тест 2)

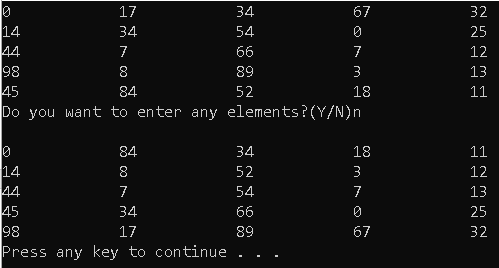


Рисунок 10(Тест 3)

Результати віповідають очікуванням

# **6. Висновки щодо розробки програми за методом низхідного**

# **програмування.**

Метод низхідного програмування полягає в покроковій деталізації алгоритма до елементарних дій. Одна з головних переваг цього методу полягає в тому, що при реалізації алгоритму вирішуються окремі задачі. Це вигідно через те, що можна використовувати ці «підзадачі» декілька разів навіть в одній програмі.