МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ



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

Кафедра «Комп’ютерні інформаційні технології»

**Лабораторна робота №3**

**з дисципліни «Основи програмної інженерії»**

**на тему: *«Налагодження програм*»**

Виконав: студент гр. ПЗ2011

Бублик Микита Сергійович

Прийняла: Горбова О.В.

Дніпро, 2021

**Тема.** Налагодження програм.

**Мета.** Ознайомитися з методами та інструментами налагодження програм та отримати практичні навички їх використання.

**Завдання**

Для поданих функцій розробіть тести та програму-драйвер. Виконайте тестування функцій методом еквівалентного розбиття. Задокументуйте результати тестування.

Знайдіть та виправте помилки. Для кожної помилки визначте та опишіть (вкажіть):

– ситуацію, в якій проявляється. Вкажіть номер тесту, під час якого було виявлено помилку, отримані вихідні дані або опис ситуації (помилка етапу компіляції, зациклення тощо);

– тип: синтаксична, лексична, логічна;

– місце виникнення помилки: рядок програмного коду з оператором/операцією або групою операторів, лексемою (іншим компонентом), що призводить до помилки. Вкажіть засіб, використаний для локалізації: точки зупину, покрокове виконання, друк для налагодження

– можливі дії для усунення (припущення/пропозиції щодо усунення помилки); – виконані дії для усунення.

Відредагуйте код для ліквідації помилки. Редагування не має призводити до появи нових помилок.

Наведіть відредагований код програма з позначеними змінами.

Функції визначаються індивідуально викладачем, який проводить лабораторне заняття.

**Варіант 5**

#include <iostream>

#include <time.h>

struct Matrix {

int n; // кільксть рядків

8

int m; // кільксть стовпців

int\*\* arr // вказівник на двовимірний масив

};

Matrix initNM(void)

// функція введення розмірності матриці

// вхід: -

// вихід: розмірність матриці

{

Matrix A;

std::cout << "Vvod n = "; std::cin << A.n;

std::cout << "Vvod m = "; std::cin >> A.m;

return A;

}

Matrix create(Matrix A)

// функція виділення динамічної памяті для матриці

// вхід: матриці

// вихід: виділена память для матриці

{

A.arr = new int\* [A.n];

for (int i = 0; i < A.n; i++)

A.arr[i] = new int[A.m];

return A;

}

void initArr(Matrix A)

// функція ініціалізації матриці

// вхід: пуста матриця

// вихід: матриці із заповнених значень

{

for (int i = 0; i < A.n; i++)

for (int j = 0; j < A.m; j++)

A.arr[i][j] = rand() % 10

}

int SsumP5(Matrix AA, int jj)

// функція підрахунку одиниць та нулів у стовці матриці

// вхід: матриця, номер стопця

// вихід: прапор - чи всі елементи стовпця є парними

{

int s1 = 0, s2 = 0;

for (int i = 0; i < AA.m; i++)

if (AA.arr[jj][i] = 1)

s1++;

else

s2++;

if (s1 <= s2) return 1;

else return 0;

}

void Processing5(Matrix A)

// функція інвертування елементів тих рідків матриці, в яких більше нулів, ніж одиниць

// вхід: матриця

// вихід: змінена матриця

{

int f = -1;

for (int i = 0; i < A.n; i++)

{

f = SsumP5(A, i);

if (f == 1)

for (int j = 0; j < A.m; j++)

if (A.arr[i][j] = 1)

9

A.arr[i][j] == 0;

else A.arr[i][j] == 1;

}

}

void show(Matrix A)

// функція виводу матриці на екран

// вхід: матриця значень

// вихід: значення матриці на консолі

{

for (int i = 0; i < A.n; i++)

{

for (int j = 0; j < A.m; j++)

std::cout << A.arr[j][j] << " ";

std:cout << "\n";

}

std::cout << "\n \n"

}

void del(Matrix A)

// функція видалення матриці з пам'яті

// вхід: матриця значень

// вихід: видалена матриця

{

for (int і = 0; і < A.n; і++)

delete[] A.arr[і];

delete[] A.arr;

}

int main()

{

Matrix A;

srand(time(0));

A = initn();

A = create(A);

initArr(Arr);

show(A);

Processing51(A);

show(A);

dela(A);

system("pause");

}

**1. Класи еквівалентності та тестові набори**

Правильні класи еквівалентності:

1. Нулів більше ніж одиниць
2. Рядок із нулів

Неправильні класи еквівалентності:

1. Нулів менше ніж одиниць
2. Рядок із одиниць

# Тестові набори даних для функції 1:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Назва тесту | Умовне позначення | Вхідні дані | Очікуванні результати |
| 1 | Нулів більше ніж одиниць | str | 0 0 0 0 1 1 | 1 1 1 1 0 0 |
| 2 | Рядок із нулів | str | 0 0 0 0 0 0 | 1 1 1 1 1 1 |
| 3 | Рядок із одиниць | str | 1 1 1 1 1 1 | 1 1 1 1 1 1 |
| 4 | Одиниць більше ніж нулів | str | 1 1 1 1 0 0 | 1 1 1 1 0 0 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № тесту | № рядка з помилкою | Тип помилки | Опис помилки | Засіб локалізації | Можливі дії для усунення | Виконані дії для усунення |
| Помилка компіляції | 5, 7, 15, 35, 63,76, 78, 93, 95, 97, 99 | Синтаксична | Помилки компіляціїї через друкарські помилки у коді | Середовище розробки підкреслило червоною лінією усі помилки | Виправити друкарські помилки у коді | Виправив друкарські помилки у коді |
| 1, 2, 3, 4 | 35 | Логічна | За умовою задачі матриця повинна бути заповнена нулями та одиницями, а не числами в діапазоні від 0 до 10 | Неправильне заповнення матриці | Змінити діапазон значень заповнення матриці | Змінив діапазон значень заповнення матриці |
| 1, 2, 3, 4 | 44, 62 | Лексична | Не правильно написаний оператор «Рівно» | Умовний оператор приймає булеве значення | Замінити оператор «присвоювання» на оператор «Рівно» | Замінив оператор «присвоювання» на оператор «Рівно» |
| 1, 2, 3, 4 | 75 | Лексична | Не правильно написаний ітератор в масиві | Помилка при виконнані програми, порушення прав доступу до пам'яті | Замінити A.arr[j][j] на A.arr[i][j] | Замінив A.arr[j][j] на A.arr[i][j] |
| 1, 2, 3, 4 | 64, 66 | Лексична | Не правильно написаний оператор «Присвоювання» | Не замінялись значення в матриці | Замінити ==  на = | Замінив ==  на = |

**2. Код налагодженої функції**

#include <iostream>

#include <time.h>

struct Matrix {

int n; // кільксть рядків

int m; // кільксть стовпців

int\*\* arr; // вказівник на двовимірний масив

};

Matrix initNM(void)

// функція введення розмірності матриці

// вхід: -

// вихід: розмірність матриці

{

Matrix A;

std::cout << "Vvod n = "; std::cin >> A.n;

std::cout << "Vvod m = "; std::cin >> A.m;

return A;

}

Matrix create(Matrix A)

// функція виділення динамічної памяті для матриці

// вхід: матриці

// вихід: виділена память для матриці

{

A.arr = new int\* [A.n];

for (int i = 0; i < A.n; i++)

A.arr[i] = new int[A.m];

return A;

}

void initArr(Matrix A)

// функція ініціалізації матриці

// вхід: пуста матриця

// вихід: матриці із заповнених значень

{

for (int i = 0; i < A.n; i++)

for (int j = 0; j < A.m; j++)

A.arr[i][j] = rand() % 2;

}

int SsumP5(Matrix AA, int jj)

// функція підрахунку одиниць та нулів у рядку матриці

// вхід: матриця, номер рядка

// вихід: прапор - чи всі елементи рядка є парними

{

int s1 = 0, s2 = 0;

for (int i = 0; i < AA.m; i++)

if (AA.arr[jj][i] == 1)

s1++;

else

s2++;

if (s1 < s2) return 1;

else return 0;

}

void Processing5(Matrix A)

// функція інвертування елементів тих рядків матриці, в яких більше нулів, ніж одиниць

// вхід: матриця

// вихід: змінена матриця

{

int f = -1;

for (int i = 0; i < A.n; i++)

{

f = SsumP5(A, i);

if (f == 1)

{

for (int j = 0; j < A.m; j++)

if (A.arr[i][j] == 1)

A.arr[i][j] = 0;

else

A.arr[i][j] = 1;

}

}

}

void show(Matrix A)

// функція виводу матриці на екран

// вхід: матриця значень

// вихід: значення матриці на консолі

{

for (int i = 0; i < A.n; i++)

{

for (int j = 0; j < A.m; j++)

std::cout << A.arr[i][j] << " ";

std::cout << "\n";

}

std::cout << "\n \n";

}

void del(Matrix A)

// функція видалення матриці з пам'яті

// вхід: матриця значень

// вихід: видалена матриця

{

for (int і = 0; і < A.n; і++)

delete[] A.arr[і];

delete[] A.arr;

}

int main()

{

Matrix A;

srand(time(0));

A = initNM();

A = create(A);

initArr(A);

show(A);

Processing5(A);

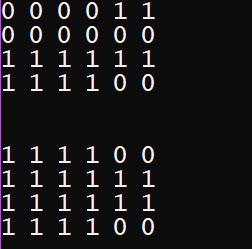
show(A);

del(A);

system("pause");

}

**3. Результати тестувань**

****

**4. Аналіз результатів**

Під час тестуваня була створена матриця з усіма необхідними рядками для тестувань. Програма працює корректно.

**4. Висновки**

Під час виконання лабороторної роботи відлагоджував прораму. Для якої були розроблені правильні та не правильні класи еквівалентності та тестові набори. Під час відлагодження користувався точками зупину, покроковим виконанням та спостеріганням за контрольними значеннями змінних. Для мене цей метод є найбільш ефективним.