Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

Механіко-технологічний факультет

Кафедра кібербезпеки та програмного забезпечення

# ЗВІТ

ПРО ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ № 12

з навчальної дисципліни “Базові методології та технології програмування”

ПРОГРАМНА РЕАЛІЗАЦІЯ АБСТРАКТНИХ ТИПІВ ДАНИХ

ВИКОНАВ

студент академічної групи

КІ-21-1 Стрюк В. Є.

ПЕРЕВІРИВ

викладач кафедри кібербезпеки

та програмного забезпечення

\_\_\_\_\_\_\_\_\_  П.С. Усік

Кропивницький – 2022

# ЛАБОРАТОРНА РОБОТА №12

**Тема:** Програмна реалізація абстрактних типів даних

**Мета роботи:** полягає у набутті ґрунтовних вмінь і практичних навичок об’єктного аналізу й проектування, створення класів С++ та тестування їх екземплярів, використання препроцесорних директив, макросів і макрооператорів під час реалізації програмних засобів у кросплатформовому середовищі Code::Blocks.

## Варіант 16

**Завдання:**

1. Як складову заголовкового файлу ModulesПрізвище.h розробити клас ClassLab12\_Прізвище –– формальне представлення абстракції сутності предметної області (об’єкта) за варіантом, ― поведінка об’єкта якого реалізовує розв’язування задачі 7.1.
2. Реалізувати додаток Teacher, який видає 100 звукових сигналів і в текстовий файл TestResults.txt записує рядок “Встановлені вимоги порядку виконання лабораторної роботи порушено!”, якщо файл проекта main.срр під час його компіляції знаходився не в \Lab12\prj, інакше –– створює об’єкт класу ClassLab12\_Прізвище із заголовкового файлу ModulesПрізвище.h та виконує його unit-тестування за тест-сьютом(ами) із \Lab12\TestSuite\, проколюючи результати тестування в текстовий файл \Lab12\TestSuite\TestResults.txt

# ХІД РОБОТИ

### **Завдання 12.1**

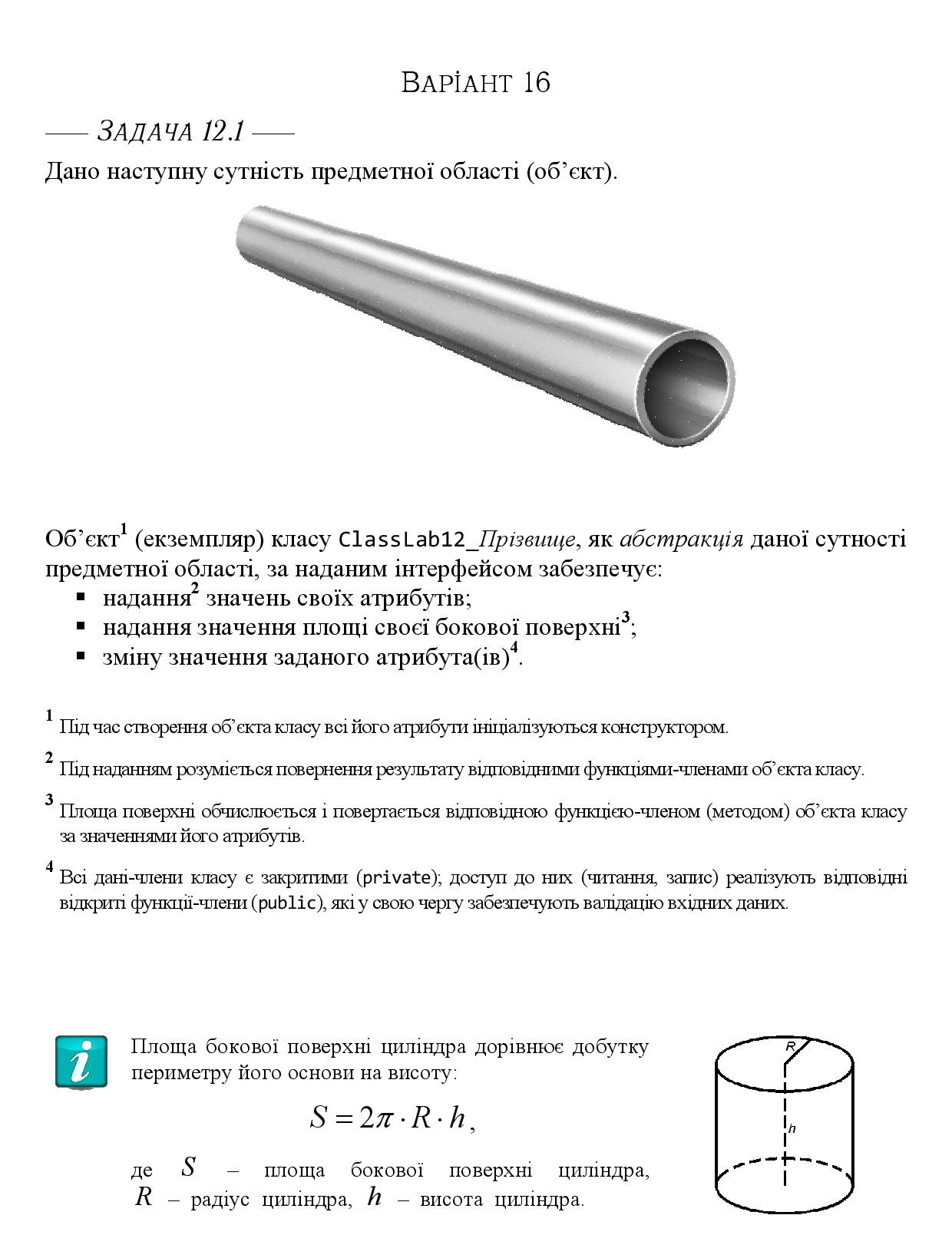


Рисунок 1 - Умова задачі 12.1

### **Концептуалізація предметної області:**

За умовою завдання, дано візуалізацію об’єкта, структуру якого можна інтерпретувати як “металева труба”. Однак, спираючись на вимоги щодо значень множини вхідних та вихідних даних, достатньо обмежитись концепцією більш абстрактного об’єкта - геометричної фігури циліндр, без верхньої та нижньої основи.

Тому, визначення всіх властивостей сутності описується лише значенням радіуса, висоти та площі бокової поверхні, залежної від попередніх двох значень. Обмеження та допущення щодо вхідних даних зводяться до пошуку найбільшої та найменшої в світі структури, схожою з обраною концепцією, адже маніпулювати довжинами у нескінченних та від’ємних межах - неможливо.

### **Вимоги до програмного модуля:**

Властивості об’єкта (радіус, висота - в якості полів, площа - в якості метода) оголошуються на рівні private. Методи отримання значень з рівня private (функції мають префікс get в ідентифікаторі) та методи ініціалізації полів класу (функції, що мають префікс set в ідентифікаторі) оголошуються на рівні public.

Поля класу за замовчуванням ініціалізуються *0.01* (в метричній системі), що надалі використовуються як обмеження або виявлення некоректної ініціалізації.

Процедури користувацького визначення радіуса та висоти мають коректно зчитувати як десяткову крапку так і кому, а також відкидати присвоєння нового значення, якщо аргумент значення не приведено до форми стандартного типу float з точністю до 6 цифр після коми.

### 

### **Артефакти детального проектування програмного модуля:**

Приватні поля класу:

* pipeHeight - висота об’єктної сутності (надалі - цилідндр);
* pipeRadius - радіус циліндра;

Приватні методи класу:

* pipeSquare() - функція, в якості аргументів отримує висоту та радіус циліндра та повертає значення площі бокової поверхні;

Публічні методи класу:

* getPipeHeight(): повертає значення приватної змінної pipeHeight;
* getPipeRadius(): повертає значення приватної змінної pipeRadius;
* getPipeSquare(): видає значення приватного методу pipeSquare();
* setPipeHeight(): функція приймає два аргументи: рядок, та адресу змінної типу float. В рядку всі коми замінюються на крапки. За допомогою бібліотеки <sstream>, виконується перевірка отриманого рядка на відповідність числовому значенню типу float. У випадку істинності виразу, приватне поле pipeHeight ініціалізується даним рядком, приведеним до типу float методом stof(), інакше - ініціалізується значенням за замовчуванням.
* setPipeRadius(): ідентично до setPipeHeight(), однак ініціалізується приватне поле pipeRadius;

### **Тестові артефакти:**

Вхідні значення та очікувані записані в текстовому файлі й приведені до спільної структури, що надалі буде використовуватись додатком Teacher.exe. Протоколювання читання здійснюється 2-ма умовами: у випадку, якщо зчитування неможливе, до текстового файлу TestResults.txt записується “Встановлені вимоги порядку виконання лабораторної роботи порушено!” та завершується робота застосунку; якщо зчитування вдале - відбувається Unit-тестування. Вміст файлу TS.txt:

pipeHeight: 2.7e3

pipeRadius: 1.5e400

pipeSquare: 0.000628

pipeHeight: word

pipeRadius: word

pipeSquare: 0.000628

pipeHeight: 12,90

pipeRadius: 23,78

pipeSquare: 1927.442505

pipeHeight: -5

pipeRadius: -8

pipeSquare: 0.000628

pipeHeight: 421

pipeRadius: 45

pipeSquare: 0.000628

pipeHeight: 420.0

pipeRadius: 44.0

pipeSquare: 116113.265625

pipeHeight: 45 12.2

pipeRadius: 3.1 23

pipeSquare: 0.000628

pipeHeight: 5.128

pipeRadius: 34.41

pipeSquare: 1108.696167

pipeHeight: 250.78

pipeRadius: 34.1

pipeSquare: 53731.273438

pipeHeight: 0,9

pipeRadius: 0,03

pipeSquare: 0.169646

### **Аналіз та архітектура завдання 12.2:**

На початку роботи застосунку, виконується перевірка, зазначена у вимогах завдання 12.2. Робота застосунку припиняється, якщо файл main.cpp не знайдено в теці ../prj.

Створюється об’єкт класу ClassLab12\_Striuk, файл TS.txt відкривається для зчитування, файл TestResults.txt відкривається для дозапису (попередній вміст очищається, при відкритті файлу). Відбувається перевірка на наявність даних текстових файлів в директорії та чи відкриті вони на момент роботи даної функції. Якщо хоча б одна умова хибна - робота застосунку припиняться.

Цикл на 10 ітерацій, призначений для декларування процесу тестування, виконує наступний порядок дій:

* оголошення рядків, призначених для збереження радіусу, висоти та площі циліндра. Буферний рядок, що використовуються в проміжних операціях. Змінна типу float, що не має бути ініціалізовано.
* зчитування з TS.txt в рядок bufLine методом getline(), починаючи з 13 символа в рядку. Відбуваєтьс окремо для висоти, радіуса та площі циліндра.
* радіус та висота ініціалізуються публічними методами, а площа що отримана публічним методом та приведена до типу string записується в змінну bufLine.
* виконується форматоване виведення значень та отриманого результату Тест Кейсів у файл TestResults.txt.

# ВИСНОВКИ

Під час процесу виконання завдань, виникли наступні спостереження щодо потреб лабораторної роботи: не дивлячись на кінцевий результат реалізації кожного методу та процедури, що не виділяються достатньою складністю, метою лабораторної роботи стала демонстрація навичок коректної абстракції сутності предметної. Розглядаючи точніше - абстрагування мало бути відповідним до вимог задачі, тобто мати обмеження щодо рівня абстрагування (деталізація об’єкта була мінімальною).

Згідно з виданого завдання, абстракція сутності металевої труби (власний висновок на основі зображення об’єкта) було зведено до простої геометричної фігури - прямий циліндр без основ. Змінних довжини, радіуса та метода отримання площі було достатньо для представлення властивостей циліндра.

Для кожного поля (приватної змінної класу) відведено два метода, визначені порядком лабораторної: функція отримання значення з приватного поля (метод повертає вміст змінної) та функція ініціалізації даних полів (містить перевірку на відповідність типу та порівняння з визначеними обмеженнями. Присвоює приватному полю коректне значення).

Була ідея, оголосити площу об’єкта як поле класу, проте зручнішим виявилось оголосити площу як закритий метод з поверненням результату розрахунків. Дана процедура є прив’язаною функція отримання площі, що також єдиним способом викликати закриту процедуру розрахунку площі.

Концепція роботи метода ініціалізації радіуса та довжини циліндра: в якості аргументів, функція отримує ініціалізований рядок та адресу оголошеного числа типу float. В тілі оголошується змінна типу char, рядку присвоюються цей же рядок, в якому коми заміщені крапками і відбувається визначення змінної типу stringstream, залежної від рядка. Якщо виконується умова побітової перевірки вхідного рядка з типом змінної float і приведена до цього типу змінна string не порушує обмежень - приватне поле ініціалізується приведеним до типу float рядка.

Обраною структурою TestSuite став текстовий файл, в кожному рядку якого, стала кількість перших символів відведена для дескрипції значення, записаного далі. Таким чином, при зчитуванні даних з файлу, з рядка відкидаються перші n символів.

Для зручної навігації та аналізу структури коду, додаткові локальні функції описано в файлі AdditionFunctions.h.

В цьому файлі присутні три функції:

* fileInDirectory(): відповідає за перевірку наявності файлу main.cpp в теці ../prj під час компіляції проєкту. Містить реалізацію вимог завдання №2.
* filesIsOpen(): виконує повну перевірку можливості до редагування та наявність вікритих текстових файлі TS.txt та TestResult.txt. Якщо хоча б одна умова хибна - в TestResult.txt записується повідомлення про порушення вимог лабораторної роботи.
* declareTestSuiteResults(): отримує всі необхідні аргументи, в якості значення, що використовуються для декларування результату модульного тестування за описаним тест кейсом.

Пропозиція на огляд та можливу зміну, щодо вимоги лабораторної роботи про спосіб декларування классу:

Доцільніше виконувати опис об’ємних функцій в \*.cpp файлі, що звя’язується з \*.h файлом з описом класу, та його методів. Заголовкові файли простіше асоціювати саме з набором заголовків та поверхневим описом користувацьких типів, а файли вихідного коду з розгорнутим описом кожного метода. Це буде гарною звичкою, котрою можна послуговуватися у майбутньому.

**ДОДАТОК А**

(набір тестових результатів TestResults.txt)

----------------------------------------------------

| TEST CASE 1 |

| (input) Pipe height: 0.01 м |

| (input) Pipe radius: 0.01 м |

| (output) Pipe square: 0.000628 м^2 |

| (expected) Pipe square: 0.000628 м^2 |

| TEST CASE RESULT: true |

----------------------------------------------------

----------------------------------------------------

| TEST CASE 2 |

| (input) Pipe height: 0.01 м |

| (input) Pipe radius: 0.01 м |

| (output) Pipe square: 0.000628 м^2 |

| (expected) Pipe square: 0.000628 м^2 |

| TEST CASE RESULT: true |

----------------------------------------------------

----------------------------------------------------

| TEST CASE 3 |

| (input) Pipe height: 12.9 м |

| (input) Pipe radius: 23.78 м |

| (output) Pipe square: 1927.442505 м^2 |

| (expected) Pipe square: 1927.442505 м^2 |

| TEST CASE RESULT: true |

----------------------------------------------------

----------------------------------------------------

| TEST CASE 4 |

| (input) Pipe height: 0.01 м |

| (input) Pipe radius: 0.01 м |

| (output) Pipe square: 0.000628 м^2 |

| (expected) Pipe square: 0.000628 м^2 |

| TEST CASE RESULT: true |

----------------------------------------------------

----------------------------------------------------

| TEST CASE 5 |

| (input) Pipe height: 0.01 м |

| (input) Pipe radius: 0.01 м |

| (output) Pipe square: 0.000628 м^2 |

| (expected) Pipe square: 0.000628 м^2 |

| TEST CASE RESULT: true |

----------------------------------------------------

----------------------------------------------------

| TEST CASE 6 |

| (input) Pipe height: 420 м |

| (input) Pipe radius: 44 м |

| (output) Pipe square: 116113.265625 м^2 |

| (expected) Pipe square: 116113.265625 м^2 |

| TEST CASE RESULT: true |

----------------------------------------------------

----------------------------------------------------

| TEST CASE 7 |

| (input) Pipe height: 0.01 м |

| (input) Pipe radius: 0.01 м |

| (output) Pipe square: 0.000628 м^2 |

| (expected) Pipe square: 0.000628 м^2 |

| TEST CASE RESULT: true |

----------------------------------------------------

----------------------------------------------------

| TEST CASE 8 |

| (input) Pipe height: 5.128 м |

| (input) Pipe radius: 34.41 м |

| (output) Pipe square: 1108.696167 м^2 |

| (expected) Pipe square: 1108.696167 м^2 |

| TEST CASE RESULT: true |

----------------------------------------------------

----------------------------------------------------

| TEST CASE 9 |

| (input) Pipe height: 250.78 м |

| (input) Pipe radius: 34.1 м |

| (output) Pipe square: 53731.273438 м^2 |

| (expected) Pipe square: 53731.273438 м^2 |

| TEST CASE RESULT: true |

----------------------------------------------------

----------------------------------------------------

| TEST CASE 10 |

| (input) Pipe height: 0.9 м |

| (input) Pipe radius: 0.03 м |

| (output) Pipe square: 0.169646 м^2 |

| (expected) Pipe square: 0.169646 м^2 |

| TEST CASE RESULT: true |

----------------------------------------------------

# ДОДАТОК Б

(Вихідний код ModulesStriuk.h)

#ifndef MODULESSTRIUK\_H\_INCLUDED

#define MODULESSTRIUK\_H\_INCLUDED

#include <string>

#include <cstring>

#include <sstream>

#include <regex>

#define PI 3.14159265359

using namespace std;

class ClassLab12\_Striuk

{

public:

ClassLab12\_Striuk();

float getPipeHeight() { return pipeHeight; }

float getPipeRadius() { return pipeRadius; }

float getPipeSquare() { return pipeSquare(pipeHeight, pipeRadius); }

void setPipeHeight(string height, float&f);

void setPipeRadius(string radius, float&f);

private:

float pipeHeight;

float pipeRadius;

float pipeSquare(float height, float radius) { return 2 \* PI \* radius \* height; };

};

ClassLab12\_Striuk::ClassLab12\_Striuk()

{

pipeHeight = 0.01;

pipeRadius = 0.01;

}

void ClassLab12\_Striuk::setPipeHeight(string height, float&f)

{

pipeHeight = 0.01;

char junk;

height = regex\_replace(height, regex(","), ".");

stringstream ss(height);

if (ss >> f && !(ss >> junk)) {

if (stof(height) >= 0.01 && stof(height) <= 420) {

pipeHeight = stof(height);

}

}

}

void ClassLab12\_Striuk::setPipeRadius(string radius, float&f)

{

pipeRadius = 0.01;

char junk;

radius = regex\_replace(radius, regex(","), ".");

stringstream ss(radius);

if (ss >> f && !(ss >> junk)) {

if (stof(radius) >= 0.01 && stof(radius) <= 44) {

pipeRadius = stof(radius);

}

}

}

#endif // MODULESSTRIUK\_H\_INCLUDED

# ДОДАТОК В

(Вихідний код додатку Teacher)

**… /Teacher / AdditionFunctions.h**

#ifndef ADDITIONFUNCTIONS\_H\_INCLUDED

#define ADDITIONFUNCTIONS\_H\_INCLUDED

#include "ModulesStriuk.h"

#include <iostream>

#include <fstream>

#include <iomanip>

using namespace std;

bool fileInDirectory()

{

string cpp = \_\_FILE\_\_;

size\_t found = cpp.find("\\lab12\\prj");

if (found == string::npos) {

ofstream resFile("../TestSuite/TestResults.txt");

resFile << "Встановлені вимоги порядку виконання лабораторної роботи порушено!" << endl;

for (int i = 0; i < 100; i++) {

cout << "\a";

}

resFile.close();

return false;

}

return true;

}

bool filesIsOpen(ofstream &wfile, ifstream &rfile)

{

if (!rfile.is\_open() || !wfile.is\_open() || !rfile || !wfile) {

ofstream resFile("../TestSuite/TestResults.txt");

resFile << "Встановлені вимоги порядку виконання лабораторної роботи порушено!" << endl;

resFile.close();

return false;

}

return true;

}

void declareTestSuiteResults(ofstream &resFile, ClassLab12\_Striuk pipe, string outArea, string expArea, int index)

{

resFile << "----------------------------------------------------" << endl;

resFile << "| TEST CASE " << setw(2) << index << setw(38) << "|" << endl;

resFile << "| (input) Pipe height: " << setw(20) << pipe.getPipeHeight() << " м |" << endl;

resFile << "| (input) Pipe radius: " << setw(20) << pipe.getPipeRadius() << " м |" << endl;

resFile << "| (output) Pipe square: " << setw(20) << outArea << " м^2 |" << endl;

resFile << "| (expected) Pipe square: " << setw(20) << expArea << " м^2 |" << endl;

resFile << "| TEST CASE RESULT: " << setw(20) << boolalpha << (expArea.compare(outArea) == 0) << " |" << endl;

resFile << "----------------------------------------------------" << endl;

}

#endif // ADDITIONFUNCTIONS\_H\_INCLUDED

**… /Teacher / main.cpp**

#include "AdditionFunctions.h"

using namespace std;

int main()

{

if (fileInDirectory() == false) { return 0; }

ClassLab12\_Striuk pipe;

ifstream caseFile("../TestSuite/TS.txt");

ofstream resultFile("../TestSuite/TestResults.txt");

if (filesIsOpen(resultFile, caseFile) == false) { return 0; }

for (int i = 1; i < 11; i++) {

string pipeHeight;

string pipeRadius;

string pipeSquare;

string bufLine;

float bufFloat;

getline(caseFile, bufLine);

pipeHeight = bufLine.erase(0, 12);

getline(caseFile, bufLine);

pipeRadius = bufLine.erase(0, 12);

getline(caseFile, bufLine);

pipeSquare = bufLine.erase(0, 12);

pipe.setPipeHeight(pipeHeight, bufFloat);

pipe.setPipeRadius(pipeRadius, bufFloat);

bufLine = to\_string(pipe.getPipeSquare());

declareTestSuiteResults(resultFile, pipe, bufLine, pipeSquare, i);

}

caseFile.close();

resultFile.close();

system("pause");

return 0;

}