Міністерство освіти і науки України

Національний університет “Львівська політехніка”

Кафедра інформаційних систем та мереж

ЗВІТ

про виконання лабораторної роботи № 3

з дисципліни Алгоритми та структури даних

Варіант 18

Виконав студент групи СА-32

Маленчак Валентин Богданович

Прийняв доцент Щербак C.C.

*Львів 2017*

**Тема:** інтегровані структури даних, запису.

**Мета роботи:** придбання і закріплення навиків в роботі із записами, в інтеграції даних, в модульному програмуванні.

**Завдання на роботу:** для заданої прикладної області розробити опис об'єктів цієї області. Розробити процедури, що реалізують базові операції над цими об'єктами, зокрема:

* текстове введення-виведення (консольний і файловий);
* присвоювання;
* видалення;
* редагування
* задання константних значень;
* порівняння (не менше 2-х типів);
* пошук за шаблоном.

Підготувати файл початкових даних, що містять не менше 10 значень конкретних об'єктів. В якості формату даних/файлів використовуйте CSV або XML.

Використовуючи процедури і описи модуля типу даних, розробити програму, що забезпечує введення та зчитування початкових даних з першого файлу даних в пам'ять і зберігання їх в масиві, сортування масиву по алфавітному і по числовому параметру.

**Індивідуальне завдання:**

|  |  |
| --- | --- |
| Опис зображення | тип фігури (квадрат, коло і т.п.), координати на площини, числові характеристики |

# Хід роботи

Текст програми на мові С++:

// Файл Figure.h

#pragma once

#include <iostream>

#include "UInterface.h"

const double PI = atan(1.0) \* 4;

using namespace std;

class Figure

{

public:

virtual void getParameters() {};

virtual void setParameters() {};

};

class Point : public Figure

{

protected:

double x;

double y;

public:

Point(double x1 = 0.0, double y1 = 0.0) : x(x1), y(y1) {}

virtual void getParameters()

{

UInterface::getID(1);

UInterface::getPoint(x, y);

}

virtual void setParameters()

{

UInterface::getID(1);

UInterface::setPoint(x, y);

}

double getX()

{

return x;

}

double getY()

{

return y;

}

};

class Circle : public Point

{

protected:

double radius;

public:

Circle(double xCenter = 0.0, double yCenter = 0.0, double radius = 1.0) : Point(xCenter, yCenter)

{

this -> radius = radius;

}

virtual void getParameters()

{

UInterface::getID(2);

UInterface::getPoint(x, y);

UInterface::getID(3);

UInterface::getRadius(radius);

}

virtual void setParameters()

{

UInterface::getID(2);

UInterface::setPoint(x, y);

UInterface::getID(3);

UInterface::setRadius(radius);

}

double getXCenter()

{

return x;

}

double getYCenter()

{

return y;

}

double getRadius()

{

return radius;

}

};

class Triangle : public Point

{

protected:

double a;

double b;

double fi;

public:

Triangle(double x1 = 0.0, double y1 = 0.0, double a1 = 1.0, double b1 = 1.0, double fi1 = PI/3) :

Point(x1, y1), a(a1), b(b1), fi(fi1) {}

virtual void getParameters()

{

UInterface::getID(1);

UInterface::getPoint(x, y);

UInterface::getID(4);

UInterface::getLineSegment(a);

UInterface::getID(5);

UInterface::getLineSegment(b);

UInterface::getID(6);

UInterface::getAngle(fi);

}

virtual void setParameters()

{

UInterface::getID(1);

UInterface::setPoint(x, y);

UInterface::getID(4);

UInterface::setLineSegment(a);

UInterface::getID(5);

UInterface::setLineSegment(b);

UInterface::getID(6);

UInterface::setAngle(fi);

}

double getX()

{

return x;

}

double getY()

{

return y;

}

double getFirstLineSegment()

{

return a;

}

double getSecondLineSegment()

{

return b;

}

double getAngle()

{

return fi;

}

};

class Quadrilateral : public Point

{

protected:

double diagonal1;

double diagonal2;

double psi;

public:

Quadrilateral(double x1 = 0.0, double y1 = 0.0, double diagonal11 = 1, double diagonal22 = 1, double psi1 = PI/2)

: Point(x1, y1) , diagonal1(diagonal11), diagonal2(diagonal22), psi(psi1) {}

virtual void getParameters()

{

UInterface::getID(4);

UInterface::getPoint(x, y);

UInterface::getID(7);

UInterface::getLineSegment(diagonal1);

UInterface::getID(8);

UInterface::getLineSegment(diagonal2);

UInterface::getID(9);

UInterface::getAngle(psi);

}

virtual void setParameters()

{

UInterface::getID(1);

UInterface::setPoint(x, y);

UInterface::getID(7);

UInterface::setLineSegment(diagonal1);

UInterface::getID(8);

UInterface::setLineSegment(diagonal2);

UInterface::getID(9);

UInterface::setAngle(psi);

}

double getX()

{

return x;

}

double getY()

{

return y;

}

double getFirstDiagonal()

{

return diagonal1;

}

double getSecondDiagonal()

{

return diagonal2;

}

double getAngle()

{

return psi;

}

};

// Файл CRUD.h

#pragma once

#include <iostream>

#include <string>

#include <fstream>

#include <vector>

#include <algorithm>

#include "UInterface.h"

#include "Figure.h"

using namespace std;

enum objectTypes

{

objectPoint,

objectCircle,

objectTriangle,

objectQuadrilateral

};

class CRUD

{

private:

Point point;

Circle circle;

Triangle triangle;

Quadrilateral quadrilateral;

bool Search(int index, string str)

{

string tmp;

double tmp1;

ifstream myFile;

myFile.open("MyFile.csv");

if (!myFile.is\_open())

{

UInterface::getID(17);

return false;

}

while (!myFile.eof())

{

getline(myFile, tmp, ';');

if (tmp == str)

{

try

{

switch (index)

{

case objectPoint:

getline(myFile, tmp, ';');

tmp1 = stod(tmp);

if (tmp1 != point.getX())

{

getline(myFile, tmp, '\n');

continue;

}

getline(myFile, tmp, '\n');

tmp1 = stod(tmp);

if (tmp1 != point.getY())

continue;

return true;

break;

case objectCircle:

getline(myFile, tmp, ';');

tmp1 = stod(tmp);

if (tmp1 != circle.getX())

{

getline(myFile, tmp, '\n');

continue;

}

getline(myFile, tmp, ';');

tmp1 = stod(tmp);

if (tmp1 != circle.getY())

{

getline(myFile, tmp, '\n');

continue;

}

getline(myFile, tmp, '\n');

tmp1 = stod(tmp);

if (tmp1 != circle.getRadius())

continue;

return true;

break;

case objectTriangle:

getline(myFile, tmp, ';');

tmp1 = stod(tmp);

if (tmp1 != triangle.getX())

{

getline(myFile, tmp, '\n');

continue;

}

getline(myFile, tmp, ';');

tmp1 = stod(tmp);

if (tmp1 != triangle.getY())

{

getline(myFile, tmp, '\n');

continue;

}

getline(myFile, tmp, ';');

tmp1 = stod(tmp);

if (tmp1 != triangle.getFirstLineSegment())

{

getline(myFile, tmp, '\n');

continue;

}

getline(myFile, tmp, ';');

tmp1 = stod(tmp);

if (tmp1 != triangle.getSecondLineSegment())

continue;

getline(myFile, tmp, '\n');

tmp1 = stod(tmp);

if (tmp1 != triangle.getAngle())

continue;

return true;

break;

case objectQuadrilateral:

getline(myFile, tmp, ';');

tmp1 = stod(tmp);

if (tmp1 != quadrilateral.getX())

{

getline(myFile, tmp, '\n');

continue;

}

getline(myFile, tmp, ';');

tmp1 = stod(tmp);

if (tmp1 != quadrilateral.getY())

{

getline(myFile, tmp, '\n');

continue;

}

getline(myFile, tmp, ';');

tmp1 = stod(tmp);

if (tmp1 != quadrilateral.getFirstDiagonal())

{

getline(myFile, tmp, '\n');

continue;

}

getline(myFile, tmp, ';');

tmp1 = stod(tmp);

if (tmp1 != quadrilateral.getSecondDiagonal())

{

getline(myFile, tmp, '\n');

continue;

}

getline(myFile, tmp, '\n');

tmp1 = stod(tmp);

if (tmp1 != quadrilateral.getAngle())

continue;

return true;

break;

default:

break;

}

}

catch (invalid\_argument)

{

UInterface::getID(13);

return false;

}

catch (out\_of\_range)

{

UInterface::getID(13);

return false;

}

}

else

{

getline(myFile, tmp, '\n');

continue;

}

}

myFile.close();

return false;

}

public:

void addObject()

{

int tmp;

ofstream myFile;

myFile.open("MyFile.csv", ios\_base::app);

UInterface::chooseSomeObject(tmp);

if (!myFile.is\_open())

{

UInterface::getID(17);

return;

}

switch (tmp)

{

case 1:

circle.setParameters();

myFile << "Circle" << ";";

myFile << circle.getX() << ";";

myFile << circle.getY() << ";";

myFile << circle.getRadius() << ";";

myFile << "\n";

break;

case 2:

triangle.setParameters();

myFile << "Triangle" << ";";

myFile << triangle.getX() << ";";

myFile << triangle.getY() << ";";

myFile << triangle.getFirstLineSegment() << ";";

myFile << triangle.getSecondLineSegment() << ";";

myFile << triangle.getAngle() << ";";

myFile << "\n";

break;

case 3:

quadrilateral.setParameters();

myFile << "Quadrilateral" << ";";

myFile << quadrilateral.getX() << ";";

myFile << quadrilateral.getY() << ";";

myFile << quadrilateral.getFirstDiagonal() << ";";

myFile << quadrilateral.getSecondDiagonal() << ";";

myFile << quadrilateral.getAngle() << ";";

myFile << "\n";

break;

default:

point.setParameters();

myFile << "Point" << ";";

myFile << point.getX() << ";";

myFile << point.getY() << ";";

myFile << "\n";

break;

}

myFile.close();

}

void outputData()

{

string tmp;

ifstream myFile;

myFile.open("MyFile.csv");

if (!myFile.is\_open())

{

UInterface::getID(17);

return;

}

while (!myFile.eof())

{

getline(myFile, tmp, '\n');

for (int i = 0; i < tmp.size(); i++)

if (tmp.at(i) == ';') tmp.at(i) = ' ';

UInterface::getLine(tmp);

}

myFile.close();

}

void searchObject()

{

int tmp;

UInterface::chooseSomeObject(tmp);

UInterface::getID(12);

string objectType;

switch (tmp)

{

case 1:

circle.setParameters();

if (Search(1, "Circle"))

UInterface::getID(14);

else

UInterface::getID(15);

break;

case 2:

triangle.setParameters();

if (Search(2, "Triangle"))

UInterface::getID(14);

else

UInterface::getID(15);

break;

case 3:

quadrilateral.setParameters();

if (Search(3, "Quadrilateral"))

UInterface::getID(14);

else

UInterface::getID(15);

break;

case 4:

point.setParameters();

if (Search(0, "Point"))

UInterface::getID(14);

else

UInterface::getID(15);

break;

default:

break;

}

}

void deleteObject()

{

int numberOfLine;

int iteration = 0;

int count = 0;

vector<string> vct;

string tmp;

ifstream myFile;

myFile.open("MyFile.csv");

if (!myFile.is\_open())

{

UInterface::getID(17);

return;

}

UInterface::getID(19);

UInterface::inputIntValue(numberOfLine);

if (numberOfLine < 1)

{

UInterface::getID(18);

return;

}

while (!myFile.eof())

{

iteration++;

if (iteration == numberOfLine)

{

getline(myFile, tmp, '\n');

continue;

}

count++;

vct.resize(count);

getline(myFile,vct[count-1], '\n');

}

myFile.close();

remove("MyFile.csv");

ofstream myNewFile;

myNewFile.open("MyFile.csv", ios\_base::trunc);

if (!myNewFile.is\_open())

{

UInterface::getID(17);

return;

}

for (int i = 0; i < vct.size(); i++)

if(vct[i]!= "")

myNewFile << vct[i] << '\n';

myNewFile.close();

}

void editObject()

{

int numberOfLine;

int iteration = 0;

vector<string> vct;

string tmp;

ifstream myFile;

myFile.open("MyFile.csv");

if (!myFile.is\_open())

{

UInterface::getID(17);

return;

}

UInterface::getID(20);

UInterface::inputIntValue(numberOfLine);

if (numberOfLine < 1)

{

UInterface::getID(18);

return;

}

while (!myFile.eof())

{

iteration++;

vct.resize(iteration);

if (iteration == numberOfLine)

{

vct[iteration - 1] = "";

UInterface::getID(21);

addObject();

getline(myFile, tmp, '\n');

continue;

}

getline(myFile, vct[iteration - 1], '\n');

}

myFile.close();

vct[numberOfLine - 1] = vct[iteration - 2];

remove("MyFile.csv");

ofstream myNewFile;

myNewFile.open("MyFile.csv", ios\_base::trunc);

if (!myNewFile.is\_open())

{

UInterface::getID(17);

return;

}

for (int i = 0; i < vct.size() - 2; i++)

myNewFile << vct[i] << "\n";

myNewFile.close();

}

void fileSort()

{

vector <string> vct;

int count = 0;

ifstream myFile;

myFile.open("MyFile.csv");

if (!myFile.is\_open())

{

UInterface::getID(17);

return;

}

while (!myFile.eof())

{

vct.resize(++count);

getline(myFile, vct[count - 1], '\n');

if (vct[count - 1] == "")

{

vct.resize(--count);

}

}

myFile.close();

sort(vct.begin(), vct.end());

remove("MyFile.csv");

ofstream myNewFile;

myNewFile.open("MyFile.csv", ios\_base::trunc);

if (!myNewFile.is\_open())

{

UInterface::getID(17);

return;

}

for (int i = 0; i < vct.size(); i++)

myNewFile << vct[i] << "\n";

myNewFile.close();

}

};

Результати комп'ютерної реалізації:

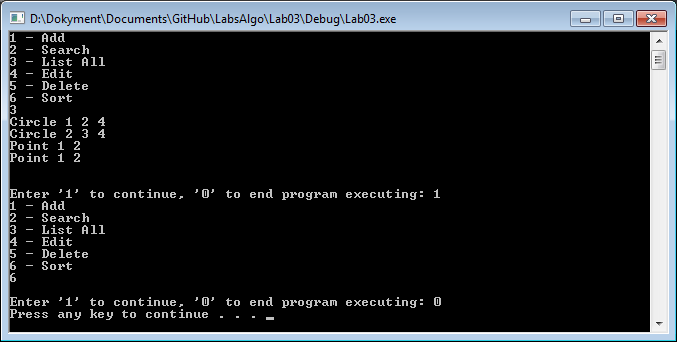


Рис.1. Виконання програми.

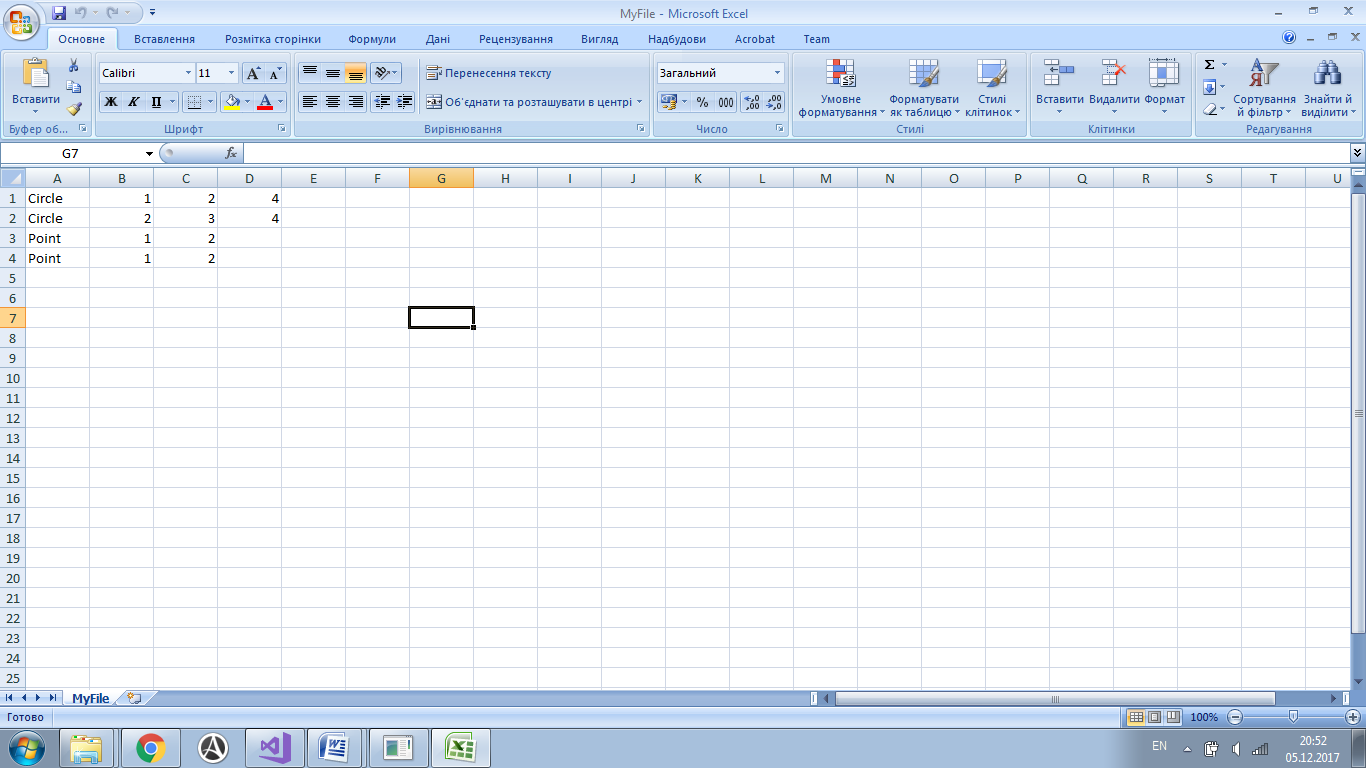


Рис.2. Вміст csv-файлу.

**Висновок:** під час виконання лабораторної роботи я набув практичних навичок роботи з інтегрованими структурами даних, запису. Була розроблена програма, яка опрацьовує csv файл. Можливим варіантом покращення програми є створення додаткового функціоналу для роботи з файлом.