**Московский авиационный институт**

**(Национальный исследовательский университет)**

Институт: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

Дисциплина: «Объектно-ориентированное программирование»

**Лабораторная работа № 6**

Тема: Асинхронное программирование

Студент: Калугин Кирилл Алексеевич

Группа: 80-207

Преподаватель: Чернышов Л.Н.

Дата:

Оценка:

Москва, 2020

1. Постановка задачи

Создать приложение, которое будет считывать из стандартного ввода данные фигур, согласно варианту задания, выводить их характеристики на экран и записывать в файл. Фигуры могут задаваться как своими вершинами, так и другими характеристиками (например, координата центра, количество точек и радиус).

Программа должна:

1. Осуществлять ввод из стандартного ввода данных фигур, согласно варианту задания;

2. Программа должна создавать классы, соответствующие введенным данных фигур;

3. Программа должна содержать внутренний буфер, в который помещаются фигуры. Для создания буфера допускается использовать стандартные контейнеры STL. Размер буфера задается параметром командной строки. Например, для буфера размером 10 фигур: **oop\_exercise\_08 10**

4. При накоплении буфера они должны запускаться на асинхронную обработку, после чего буфер должен очищаться;

5. Обработка должна производиться в отдельном потоке;

6. Реализовать два обработчика, которые должны обрабатывать данные буфера:

a. Вывод информации о фигурах в буфере на экран;

b. Вывод информации о фигурах в буфере в файл. Для каждого буфера должен создаваться файл с уникальным именем.

7. Оба обработчика должны обрабатывать каждый введенный буфер. Т.е. после каждого заполнения буфера его содержимое должно выводиться как на экран, так и в файл.

8. Обработчики должны быть реализованы в виде лямбда-функций и должны хранится в специальном массиве обработчиков. Откуда и должны последовательно вызываться в потоке – обработчике.

9. В программе должно быть ровно два потока (thread). Один основной (main) и второй для обработчиков;

10. В программе должен явно прослеживаться шаблон Publish-Subscribe. Каждый обработчик должен быть реализован как отдельный подписчик.

11. Реализовать в основном потоке (main) ожидание обработки буфера в потоке-обработчике. Т.е. после отправки буфера на обработку основной поток должен ждать, пока поток обработчик выведет данные на экран и запишет в файл.

1. Описание программы

Программа создает буфер на основе стандартного вектора. После этого пользователь вводит фигуры до тех пор, пока буфер не заполнится. Когда это происходит, вектор обрабатывается вторым потоком, выводится в стандартный вывод и записывается в файл. Файлы имеют расширение “.txt”. Их имена генерируются на основании счетчика. Таким образом, все имена при запуске будут уникальными. Продолжение или прерывание работы осуществляется вводом цифр 1 или 0 соответственно. Все фигуры являются правильными и строго сориентированными относительно координатной сетки. Безопасность всех взаимодействий потоков защищается мьютексами.

1. Тесты

PS C:\Users\Кирилл\Desktop\VSC\ООП\_8> .\a.exe 5

Choose figure type: 1. Pentagon.

2. Hexagon.

3. Octogon

1

Enter center coordinates and radius: 1

1

1

Choose figure type: 1. Pentagon.

2. Hexagon.

3. Octogon

2

Enter center coordinates and radius: 2

2

2

Choose figure type: 1. Pentagon.

2. Hexagon.

3. Octogon

3

Enter center coordinates and radius: 3

3

3

Choose figure type: 1. Pentagon.

2. Hexagon.

3. Octogon

1

Enter center coordinates and radius: 4

4

4

Choose figure type: 1. Pentagon.

2. Hexagon.

3. Octogon

2

Enter center coordinates and radius: 5

5

5

----------

A = (1,2)

B = (1.88295,1.46947)

C = (1.71934,0.305342)

D = (0.561629,0.101206)

E = (0.00973193,1.13917)

----------

----------

A = (2,4)

B = (3.73205,3)

C = (3.73205,1)

D = (2,0)

E = (0.267949,1)

F = (0.267949,3)

----------

----------

A = (3,6)

B = (5.12132,5.12132)

C = (6,3)

D = (5.12132,0.87868)

E = (3,0)

F = (0.87868,0.87868)

G = (0,3)

H = (0.87868,5.12132)

----------

----------

A = (4,8)

B = (7.53179,5.87789)

C = (6.87736,1.22137)

D = (2.24652,0.404824)

E = (0.0389277,4.55669)

----------

----------

A = (5,10)

B = (9.33013,7.5)

C = (9.33013,2.5)

D = (5,0)

E = (0.669873,2.5)

F = (0.669873,7.5)

----------

1

Choose figure type: 1. Pentagon.

2. Hexagon.

3. Octogon

1

Enter center coordinates and radius: 1

1

1

Choose figure type: 1. Pentagon.

2. Hexagon.

3. Octogon

1

Enter center coordinates and radius: 1

1

11

Choose figure type: 1. Pentagon.

2. Hexagon.

3. Octogon

1

Enter center coordinates and radius: 1

11

1

Choose figure type: 1. Pentagon.

2. Hexagon.

3. Octogon

1

Enter center coordinates and radius: 1

1

1

Choose figure type: 1. Pentagon.

2. Hexagon.

3. Octogon

1

Enter center coordinates and radius: 1

1

1

----------

A = (1,2)

B = (1.88295,1.46947)

C = (1.71934,0.305342)

D = (0.561629,0.101206)

E = (0.00973193,1.13917)

----------

----------

A = (1,12)

B = (10.7124,6.16419)

C = (8.91274,-6.64124)

D = (-3.82208,-8.88673)

E = (-9.89295,2.5309)

----------

----------

A = (1,12)

B = (1.88295,11.4695)

C = (1.71934,10.3053)

D = (0.561629,10.1012)

E = (0.00973193,11.1392)

----------

----------

A = (1,2)

B = (1.88295,1.46947)

C = (1.71934,0.305342)

D = (0.561629,0.101206)

E = (0.00973193,1.13917)

----------

----------

A = (1,2)

B = (1.88295,1.46947)

C = (1.71934,0.305342)

D = (0.561629,0.101206)

E = (0.00973193,1.13917)

----------

1

Choose figure type: 1. Pentagon.

2. Hexagon.

3. Octogon

1

Enter center coordinates and radius: 1

2

2

Choose figure type: 1. Pentagon.

2. Hexagon.

3. Octogon

2

Enter center coordinates and radius: 2

2

2

Choose figure type: 1. Pentagon.

2. Hexagon.

3. Octogon

3

Enter center coordinates and radius: 3

3

3

Choose figure type: 1. Pentagon.

2. Hexagon.

3. Octogon

3

Enter center coordinates and radius: 3

3

3

Choose figure type: 1. Pentagon.

2. Hexagon.

3. Octogon

3

Enter center coordinates and radius: 3

3

3

----------

A = (1,4)

B = (2.7659,2.93894)

C = (2.43868,0.610683)

D = (0.123258,0.202412)

E = (-0.980536,2.27835)

----------

----------

A = (2,4)

B = (3.73205,3)

C = (3.73205,1)

D = (2,0)

E = (0.267949,1)

F = (0.267949,3)

----------

----------

A = (3,6)

B = (5.12132,5.12132)

C = (6,3)

D = (5.12132,0.87868)

E = (3,0)

F = (0.87868,0.87868)

G = (0,3)

H = (0.87868,5.12132)

----------

----------

A = (3,6)

B = (5.12132,5.12132)

C = (6,3)

D = (5.12132,0.87868)

E = (3,0)

F = (0.87868,0.87868)

G = (0,3)

H = (0.87868,5.12132)

----------

----------

A = (3,6)

B = (5.12132,5.12132)

C = (6,3)

D = (5.12132,0.87868)

E = (3,0)

F = (0.87868,0.87868)

G = (0,3)

H = (0.87868,5.12132)

----------

0

1. Листинг программы

/\*

Калугин К. А. М8О-207Б-19

Создать приложение, которое будет считывать из стандартного ввода данные фигур, согласно варианту задания, выводить их характеристики на экран и записывать в файл. Фигуры могут задаваться как своими вершинами, так и другими характеристиками (например, координата центра, количество точек и радиус).

Программа должна:

1. Осуществлять ввод из стандартного ввода данных фигур, согласно варианту задания;

2. Программа должна создавать классы, соответствующие введенным данных фигур;

3. Программа должна содержать внутренний буфер, в который помещаются фигуры. Для создания буфера допускается использовать стандартные контейнеры STL. Размер буфера задается параметром командной строки. Например, для буфера размером 10 фигур: oop\_exercise\_08 10

4. При накоплении буфера они должны запускаться на асинхронную обработку, после чего буфер должен очищаться;

5. Обработка должна производиться в отдельном потоке;

6. Реализовать два обработчика, которые должны обрабатывать данные буфера:

a. Вывод информации о фигурах в буфере на экран;

b. Вывод информации о фигурах в буфере в файл. Для каждого буфера должен создаваться файл с уникальным именем.

7. Оба обработчика должны обрабатывать каждый введенный буфер. Т.е. после каждого заполнения буфера его содержимое должно выводиться как на экран, так и в файл.

8. Обработчики должны быть реализованы в виде лямбда-функций и должны хранится в специальном массиве обработчиков. Откуда и должны последовательно вызываться в потоке – обработчике.

9. В программе должно быть ровно два потока (thread). Один основной (main) и второй для обработчиков;

10. В программе должен явно прослеживаться шаблон Publish-Subscribe. Каждый обработчик должен быть реализован как отдельный подписчик.

11. Реализовать в основном потоке (main) ожидание обработки буфера в потоке-обработчике. Т.е. после отправки буфера на обработку основной поток должен ждать, пока поток обработчик выведет данные на экран и запишет в файл.

\*/

// Вариант - пятиугольник, шестиугольник, восьмиугольник

#include <iostream>

#include <memory>

#include <windows.h>

#include <algorithm>

#include <vector>

#include <math.h>

#include <string>

#include <iterator>

#include <mutex>

#include <queue>

#include <thread>

#include <functional>

#include <fstream>

using namespace std;

const double PI = 3.141592653589793238462643383279502884197169399375105820974944;

const int Five\_Id = 1;

const int Six\_Id = 2;

const int Eight\_Id = 3;

ofstream file;

int nname = 0;

template<class T>

class Q {//класс, выполняющий функцию очереди

public:

queue<T> messages;

mutex m;

void Add(T & mes) {

m.lock();

messages.push(mes);

m.unlock();

}

bool Empty() {

m.lock();

bool a = messages.empty();

m.unlock();

return a;

}

T Front() {

m.lock();

T a = messages.front();

m.unlock();

return a;

}

void Pop() {

m.lock();

messages.pop();

m.unlock();

}

};

class Figure {//создание класса фигур

public:

virtual void points () {}

virtual void write() {}

};

template <typename T>

class Five: public Figure {//создание класса-наследника для пятиугольников

public:

using type = T;

pair<T, T> O;

T R;

virtual void points () {//функция вывода вершин фигуры

cout << "----------\n"<< "A = (" << O.first << "," << O.second + R << ")\n" << "B = ("

<< O.first + R \* cos (28.0 / 180.0 \* PI) << "," << O.second + R \* sin (28.0 / 180.0 \* PI)

<< ")\n" << "C = (" << O.first + R \* cos (- 44.0 / 180.0 \* PI) << ","

<< O.second + R \* sin (- 44.0 / 180.0 \* PI) << ")\n" << "D = ("

<< O.first + R \* cos (- 116.0 / 180.0 \* PI) << "," << O.second + R \* sin (- 116.0 / 180.0 \* PI)

<< ")\n" << "E = (" << O.first + R \* cos (- 188.0 / 180.0 \* PI)

<< "," << O.second + R \* sin (- 188.0 / 180.0 \* PI) << ")\n" << "----------\n";

}

virtual void write () {//функция записи в файл

file <<"Pentagon\n" << "----------\n"<< "A = (" << O.first << "," << O.second + R << ")\n" << "B = ("

<< O.first + R \* cos (28.0 / 180.0 \* PI) << "," << O.second + R \* sin (28.0 / 180.0 \* PI)

<< ")\n" << "C = (" << O.first + R \* cos (- 44.0 / 180.0 \* PI) << ","

<< O.second + R \* sin (- 44.0 / 180.0 \* PI) << ")\n" << "D = ("

<< O.first + R \* cos (- 116.0 / 180.0 \* PI) << "," << O.second + R \* sin (- 116.0 / 180.0 \* PI)

<< ")\n" << "E = (" << O.first + R \* cos (- 188.0 / 180.0 \* PI)

<< "," << O.second + R \* sin (- 188.0 / 180.0 \* PI) << ")\n" << "----------\n";

}

Five () {};//конструктор класса

Five (T x, T y, T r): O(x, y), R(r) {};//конструктор класса

};

template <typename T>

class Six: public Figure {//создание класса-наследника для шестиугольников

public:

using type = T;

pair<T, T> O;

T R;

virtual void points () {//функция вывода вершин фигуры

cout << "----------\n" << "A = (" << O.first << "," << O.second + R << ")\n" << "B = ("

<< O.first + R \* cos (PI / 6.0) << "," << O.second + R \* sin (PI / 6.0)

<< ")\n" << "C = (" << O.first + R \* cos (- PI / 6.0) << ","

<< O.second + R \* sin (- PI / 6.0) << ")\n" << "D = (" << O.first << ","

<< O.second - R << ")\n" << "E = (" << O.first + R \* cos (- 5.0 \* PI / 6.0)

<< "," << O.second + R \* sin (- 5.0 \* PI / 6.0) << ")\n" << "F = ("

<< O.first + R \* cos (- 7.0 \* PI / 6.0) << ","

<< O.second + R \* sin (- 7.0 \* PI / 6.0) << ")\n" << "----------\n";

}

virtual void write () {//функция записи в файл

file << "Gexagon\n" << "----------\n" << "A = (" << O.first << "," << O.second + R << ")\n" << "B = ("

<< O.first + R \* cos (PI / 6.0) << "," << O.second + R \* sin (PI / 6.0)

<< ")\n" << "C = (" << O.first + R \* cos (- PI / 6.0) << ","

<< O.second + R \* sin (- PI / 6.0) << ")\n" << "D = (" << O.first << ","

<< O.second - R << ")\n" << "E = (" << O.first + R \* cos (- 5.0 \* PI / 6.0)

<< "," << O.second + R \* sin (- 5.0 \* PI / 6.0) << ")\n" << "F = ("

<< O.first + R \* cos (- 7.0 \* PI / 6.0) << ","

<< O.second + R \* sin (- 7.0 \* PI / 6.0) << ")\n" << "----------\n";

}

Six () {};//конструктор класса

Six (T x, T y, T r): O(x, y), R(r) {};//конструктор класса

};

template <typename T>

class Eight: public Figure {//создание класса-наследника для восьмиугольник

public:

using type = T;

pair<T, T> O;

T R;

virtual void points () {//функция вывода вершин фигуры

cout << "----------\n" << "A = (" << O.first << "," << O.second + R << ")\n" << "B = ("

<< O.first + R \* cos (PI / 4) << "," << O.second + R \* sin (PI / 4)

<< ")\n" << "C = (" << O.first + R << "," << O.second << ")\n"

<< "D = (" << O.first + R \* cos (- PI / 4) << "," << O.second + R \* sin (- PI / 4)

<< ")\n" << "E = (" << O.first << "," << O.second - R << ")\n"

<< "F = (" << O.first + R \* cos (- 3 \* PI / 4) << "," << O.second + R \* sin (- 3 \* PI / 4)

<< ")\n" << "G = (" << O.first - R << "," << O.second << ")\n" << "H = ("

<< O.first + R \* cos (- 5 \* PI / 4) << "," << O.second + R \* sin (- 5 \* PI / 4) << ") \n" << "----------\n";

}

virtual void write () {//функция записи в файл

file <<"Octogon\n" << "----------\n" << "A = (" << O.first << "," << O.second + R << ")\n" << "B = ("

<< O.first + R \* cos (PI / 4) << "," << O.second + R \* sin (PI / 4)

<< ")\n" << "C = (" << O.first + R << "," << O.second << ")\n"

<< "D = (" << O.first + R \* cos (- PI / 4) << "," << O.second + R \* sin (- PI / 4)

<< ")\n" << "E = (" << O.first << "," << O.second - R << ")\n"

<< "F = (" << O.first + R \* cos (- 3 \* PI / 4) << "," << O.second + R \* sin (- 3 \* PI / 4)

<< ")\n" << "G = (" << O.first - R << "," << O.second << ")\n" << "H = ("

<< O.first + R \* cos (- 5 \* PI / 4) << "," << O.second + R \* sin (- 5 \* PI / 4) << ") \n" << "----------\n";

}

Eight () {};//конструктор класса

Eight (T x, T y, T r): O(x, y), R(r) {};//конструктор класса

};

template<class T, class FIGURE>

class Factory;

template <class T>

class Factory <T, Five <T>> {//класс Factory для создания пятиугольников

public:

static shared\_ptr <Figure> Create () {

T x;

T y;

T r;

cout << "Enter center coordinates and radius: ";

cin >> x;

cin >> y;

cin >> r;

Five <T>\* ret = new Five <T> (x, y, r);

return shared\_ptr <Figure> (ret);

}

};

template <class T>

class Factory <T, Six <T>> {//класс Factory для создания шестиугольников

public:

static shared\_ptr <Figure> Create () {

T x;

T y;

T r;

cout << "Enter center coordinates and radius: ";

cin >> x;

cin >> y;

cin >> r;

Six <T>\* ret = new Six <T> (x, y, r);

return shared\_ptr <Figure> (ret);

}

};

template <class T>

class Factory <T, Eight <T>> {//класс Factory для создания восьмиугольников

public:

static shared\_ptr <Figure> Create () {

T x;

T y;

T r;

cout << "Enter center coordinates and radius: ";

cin >> x;

cin >> y;

cin >> r;

Eight <T>\* ret = new Eight <T> (x, y, r);

return shared\_ptr <Figure> (ret);

}

};

Q <shared\_ptr <Figure>> eueu;//объявление объекта класса очереди Q

void threadFunction() {//функция для потока №2

function <void(shared\_ptr<Figure> fig)> Print\_fig = [](shared\_ptr<Figure> fig) {

fig->points();

};

function<void(shared\_ptr<Figure> fig)> Write\_fig = [](shared\_ptr<Figure> fig) {

fig->write();

};

vector<function<void(shared\_ptr<Figure> fig)>> functions({Print\_fig, Write\_fig});

while(1) {

if (!eueu.Empty()) {

if (eueu.Front() == NULL) {

break;

}

nname ++;

string name = (to\_string (nname)).append (".txt");

file.open(name.c\_str());

while (!eueu.Empty()) {

shared\_ptr<Figure> el = eueu.Front();

eueu.Pop();

for (auto func : functions) {

func(el);

}

}

file.close ();

}

}

}

int main(int argc, char\*\* argv) {

thread thread\_2(threadFunction);//запуск второго потока

int size = stoi(string(argv[1]));

int vsize = 0;

if (size <= 0) {

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(hConsole, (WORD) ((0 << 4) | 4));

cout << "ERROR" << endl << "Incorrect buffer length." << endl;

SetConsoleTextAttribute(hConsole, (WORD) ((0 << 4) | 7));

return -1;

}

vector < shared\_ptr <Figure> > vault;

int id;

while (1) {//Ввод фигур

while (vsize < size) {

cout << "Choose figure type: 1. Pentagon.\n";

cout << " 2. Hexagon.\n";

cout << " 3. Octogon\n";

cin >> id;

if ((id != 1) && (id != 2) && (id != 3)) {

cout << "ERROR\nIncorrect number.\n";

break;

}

shared\_ptr <Figure> nf;

if (id == Five\_Id) {

nf = Factory<double, Five <double>>::Create ();

} else if (id == Six\_Id) {

nf = Factory<double, Six <double>>::Create ();

} else if (id == Eight\_Id) {

nf = Factory<double, Eight <double>>::Create ();

}

vault.push\_back (nf);

vsize ++;

}

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

eueu.Add(vault[i]);

}

while (!eueu.Empty()) {}

vault.clear();

vsize = 0;

int end;

cin >> end;

if (!end) {

break;

}

}

shared\_ptr<Figure> null = NULL;//завершение дополнительного потока

eueu.Add(null);

thread\_2.join();

return 0;

}

1. Выводы

Выполнив эту работу, я получил практические навыки работы с потоками. Также, я получил базовые понятия о асинхронном программировании и использовании мьютексов.

Литература

1. Издатель-подписчик (шаблон проектирования) - Википедия [Электронный ресурс]

URL: [https://ru.wikipedia.org/wiki/Издатель-подписчик\_(шаб..](https://vk.com/away.php?utf=1&to=https%3A%2F%2Fru.wikipedia.org%2Fwiki%2F%D0%98%D0%B7%D0%B4%D0%B0%D1%82%D0%B5%D0%BB%D1%8C-%D0%BF%D0%BE%D0%B4%D0%BF%D0%B8%D1%81%D1%87%D0%B8%D0%BA_(%D1%88%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)) (дата обращения 27.12.2020).

1. C++ | Чтение и запись текстовых файлов [Электронный ресурс]

URL: [https://metanit.com/cpp/tutorial/8.3.php](https://vk.com/away.php?utf=1&to=https%3A%2F%2Fmetanit.com%2Fcpp%2Ftutorial%2F8.3.php) (дата обращения 27.12.2020).

1. Потоки, блокировки и условные переменные в C++11 [Часть 1] / Хабр [Электронный ресурс]

URL: [https://habr.com/ru/post/182610/](https://vk.com/away.php?utf=1&to=https%3A%2F%2Fhabr.com%2Fru%2Fpost%2F182610%2F) (дата обращения 27.12.2020).

1. std::queue-[cppreference.com](https://vk.com/away.php?utf=1&to=http%3A%2F%2Fcppreference.com) [Электронный ресурс]

URL: [https://en.cppreference.com/w/cpp/container/queue](https://vk.com/away.php?utf=1&to=https%3A%2F%2Fen.cppreference.com%2Fw%2Fcpp%2Fcontainer%2Fqueue) (дата обращения 27.12.2020).

1. std::mutex - [cppreference.com](https://vk.com/away.php?utf=1&to=http%3A%2F%2Fcppreference.com) [Электронный ресурс]

URL: [https://en.cppreference.com/w/cpp/thread/mutex](https://vk.com/away.php?utf=1&to=https%3A%2F%2Fen.cppreference.com%2Fw%2Fcpp%2Fthread%2Fmutex) (дата обращения 27.12.2020).