**Московский авиационный институт**

**(Национальный исследовательский университет)**

Институт: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

Дисциплина: «Объектно-ориентированное программирование»

**Лабораторная работа № 8**

Тема: Асинхронное программирование

Студент: Тимофеев Алексей Владимирович

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**

В программе реализован класс Figure и его наследники (фигуры вращения) Pentagon, Hexagon, Octagon. Данные конструкции были взяты из 3 лабораторной работы и адаптированы под нужды 8 работы, а именно: был переписан конструктор и определение названия фигуры.Также реализован класс  Initiate, в котором инициализируются фигуры. По умолчанию размер буфера равен 1. Для хранения хранения лямбда функций я использую std::function. Реализован шаблонный класс server, который нужен для работы в Publish-Subscribe. В этом классе реализован шаблон Singleton, который позволяет использовать единственный экземпляр класса. Также в классе реализован метод добавления “подписчиков” класса - лямбда функций, которые добавляются в самом начале программы, вторая лямбда-функция захватывает переменную, которая определяет имя файла, куда будет записываться буффер, что обеспечивает уникальность файлов с буферами. В классе реализован метод добавления в очередь сообщений массива фигур. И метод исполнения лямбда-функций, в котором проверяется, пуста ли очередь, и, в случае, когда очередь не пуста,мьютекс захватывает процесс и выполняются лямбда-функции. Если очередь пуста, то управление передается основному потоку.

1. **Набор  тестов**

координаты центральной точки и длина радиуса 10 10 10

координаты центральной точки и длина радиуса 8  9 10

координаты центральной точки и длина радиуса 6 6 6

    координаты центральной точки и длина радиуса -100 -50 20

1. **Результаты выполнения тестов**

dude@DESKTOP-9IO9OQQ:/mnt/d/Ycheba2kurs/OOp/OOPlab/lab8$ ./lab8 4

Введите:

1 - добавить пятиугольник

2 - добавить шестиугольник

3 - добавить восьмиугольник

4 - распечатать буфер

5 - завершить программу

Введите команду:1

Введите координаты центра описанной окружности и длину радиуса описанной окружности через пробел: 10 10 10

Введите команду:2

Введите координаты центра описанной окружности и длину радиуса описанной окружности через пробел: 8 9 10

Введите команду:3

Введите координаты центра описанной окружности и длину радиуса описанной окружности через пробел: 6 6 6

Введите команду:4

Pentagon

(0.327494,12.5382) (18.7115,5.08978) (2.82015,16.9606) (15.178,1.44496) (7.16309,19.5892)

Hexagon

(-1.52413,5.95189) (16.1418,14.8061) (2.0154,0.988474) (11.2578,18.4545) (7.77903,-0.997558) (5.16309,18.5892)

Octagon

(9.15193,11.1054) (3.31156,11.364) (0.023473,6.53021) (2.40924,1.19308) (8.20392,0.419431) (11.9063,4.94372) (10.0015,10.4708) (4.29785,11.7535)

Введите команду:1

Введите координаты центра описанной окружности и длину радиуса описанной окружности через пробел: -100 -50 20

Pentagon

(0.327494,12.5382) (18.7115,5.08978) (2.82015,16.9606) (15.178,1.44496) (7.16309,19.5892)

Hexagon

(-1.52413,5.95189) (16.1418,14.8061) (2.0154,0.988474) (11.2578,18.4545) (7.77903,-0.997558) (5.16309,18.5892)

Octagon

(9.15193,11.1054) (3.31156,11.364) (0.023473,6.53021) (2.40924,1.19308) (8.20392,0.419431) (11.9063,4.94372) (10.0015,10.4708) (4.29785,11.7535)

Pentagon

(-119.345,-44.9235) (-82.5771,-59.8204) (-114.36,-36.0788) (-89.6441,-67.1101) (-105.674,-30.8217)

Буфер очищен и сохранен в файл 1.txt

Введите команду:4

Введите команду:1

Введите координаты центра описанной окружности и длину радиуса описанной окружности через пробел: 5 5 5

Введите команду:2

Введите координаты центра описанной окружности и длину радиуса описанной окружности через пробел: 10 4 5

Введите команду:3

Введите координаты центра описанной окружности и длину радиуса описанной окружности через пробел: 4 4 4

Введите команду:3

Введите координаты центра описанной окружности и длину радиуса описанной окружности через пробел: 4 5 7

Pentagon

(0.163747,6.26912) (9.35574,2.54489) (1.41007,8.48029) (7.58898,0.722478) (3.58154,9.79458)

Hexagon

(5.23794,2.47595) (14.0709,6.90306) (7.0077,-0.00576318) (11.6289,8.72723) (9.88952,-0.998779) (8.58154,8.79458)

Octagon

(6.10129,7.40361) (2.20771,7.57599) (0.0156487,4.35347) (1.60616,0.795389) (5.46928,0.27962) (7.93753,3.29582) (6.66766,6.98053) (2.86524,7.83566)

Octagon

(7.67725,10.9563) (0.863485,11.258) (-2.97261,5.61858) (-0.18922,-0.608068) (6.57124,-1.51066) (10.8907,3.76768) (8.66841,10.2159) (2.01416,11.7124)

Буфер очищен и сохранен в файл 2.txt

Введите команду:5

dude@DESKTOP-9IO9OQQ:/mnt/d/Ycheba2kurs/OOp/OOPlab/lab8$ cat 1.txt

Pentagon

(0.327494,12.5382) (18.7115,5.08978) (2.82015,16.9606) (15.178,1.44496) (7.16309,19.5892)

Hexagon

(-1.52413,5.95189) (16.1418,14.8061) (2.0154,0.988474) (11.2578,18.4545) (7.77903,-0.997558) (5.16309,18.5892)

Octagon

(9.15193,11.1054) (3.31156,11.364) (0.023473,6.53021) (2.40924,1.19308) (8.20392,0.419431) (11.9063,4.94372) (10.0015,10.4708) (4.29785,11.7535)

Pentagon

(-119.345,-44.9235) (-82.5771,-59.8204) (-114.36,-36.0788) (-89.6441,-67.1101) (-105.674,-30.8217)

dude@DESKTOP-9IO9OQQ:/mnt/d/Ycheba2kurs/OOp/OOPlab/lab8$ cat 2.txt

Pentagon

(0.163747,6.26912) (9.35574,2.54489) (1.41007,8.48029) (7.58898,0.722478) (3.58154,9.79458)

Hexagon

(5.23794,2.47595) (14.0709,6.90306) (7.0077,-0.00576318) (11.6289,8.72723) (9.88952,-0.998779) (8.58154,8.79458)

Octagon

(6.10129,7.40361) (2.20771,7.57599) (0.0156487,4.35347) (1.60616,0.795389) (5.46928,0.27962) (7.93753,3.29582) (6.66766,6.98053) (2.86524,7.83566)

Octagon

(7.67725,10.9563) (0.863485,11.258) (-2.97261,5.61858) (-0.18922,-0.608068) (6.57124,-1.51066) (10.8907,3.76768) (8.66841,10.2159) (2.01416,11.7124)

1. **Листинг программы**

server.hpp

#pragma once

#include <iostream>

#include <queue>

#include <vector>

#include <functional>

#include <mutex>

#include <thread>

template <class T>

class Server {

private:

Server(){};

std::vector<std::function<void(T &)>> subscribers;

std::queue<T> message\_queue;

std::mutex mutex;

public:

using subscriber\_type = std::function<void(T &)>;

static Server &get() {

static Server instance;

return instance;

}

void register\_subscriber(const subscriber\_type &sub) {

std::lock\_guard<std::mutex> lock(mutex);

subscribers.push\_back(sub);

}

void publish(const T &msg) {

std::lock\_guard<std::mutex> lock(mutex);

message\_queue.push(msg);

}

void run() {

while (!false) {

if (!message\_queue.empty()) {

std::lock\_guard<std::mutex> lock(mutex);

T fig = message\_queue.front();

if (fig.empty()) {

break;

}

message\_queue.pop();

for (auto sub : subscribers)

try {

sub(fig);

} catch (std::exception &ex) {

std::cout << "Exception in subscriber:" << ex.what() << std::endl;

}

} else {

std::this\_thread::yield();

}

}

}

};

pentagon.hpp

#pragma once

#include <cmath>

#include "figure.hpp"

class Pentagon: public Figure {

private:

int length;

public:

Pentagon() : Figure() { length = 0; }

Pentagon(double x1, double y1, int R) {

double angle,outX,outY;

for(int i = 1; i <= 5; i++) {// расчет точек

angle = 360 / 5;

outX = R \* cos(i\*angle) + x1;

outY = R \* sin(i\*angle) + y1;

points.emplace\_back(Pair(outX, outY));

}

length = R;

}

std::string Name() override {

return "Pentagon";

}

int Get() override {

return length;

}

std::vector<Pair> Coord() override {

return points;

}

};

octagon.hpp

#pragma once

#include <cmath>

#include "figure.hpp"

class Octagon: public Figure {

private:

int length;

public:

Octagon() : Figure() { length = 0; }

Octagon(double x1, double y1, int R) {

double angle,outX,outY;

for(int i = 1; i <= 8; i++) {// расчет точек

angle = 360 / 8;

outX = R \* cos(i\*angle) + x1;

outY = R \* sin(i\*angle) + y1;

points.emplace\_back(Pair(outX, outY));

}

length = R;

}

std::string Name() override {

return "Octagon";

}

int Get() override {

return length;

}

std::vector<Pair> Coord() override {

return points;

}

};

main.cpp

#include <iostream>

#include <string>

#include <vector>

#include <functional>

#include <fstream>

#include <thread>

#include <unistd.h>

#include "figure.hpp"

#include "pentagon.hpp"

#include "hexagon.hpp"

#include "octagon.hpp"

#include "server.hpp"

using FigVec = std::vector<std::shared\_ptr<Figure>>;

template<class A, class B, class C>

class Initiate {

public:

Initiate() {}

~Initiate() {}

std::shared\_ptr<Figure> Pentagon(double x1, double y1, int side) {

return std::make\_shared<A>(x1, y1, side);

}

std::shared\_ptr<Figure> Hexagon(double x1, double y1, int side) {

return std::make\_shared<B>(x1, y1, side);

}

std::shared\_ptr<Figure> Octagon(double x1, double y1, int side) {

return std::make\_shared<C>(x1, y1, side);

}

};

void Print(){

std::cout << "Введите координаты центра описанной окружности и длину радиуса описанной окружности через пробел: ";

}

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

int file = 1;

Server<FigVec>::get().register\_subscriber([] (FigVec &fig) {

std::cout << fig << std::endl;

std::cout.flush();

});

Server<FigVec>::get().register\_subscriber([&file] (FigVec &fig) {

std::ofstream f;

std::string name = std::to\_string(file);

name.append(".txt");

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

f << fig;

f.close();

++file;

std::cout << "Буфер очищен и сохранен в файл " << (file - 1) << ".txt" << std::endl;

std::cout.flush();

});

std::thread thread([]() {

Server<FigVec>::get().run();

});

int size\_buf = 1;

if (argc == 2) size\_buf = atoi(argv[1]);

double x1, y1;

int side, m;

FigVec fig;

Initiate<Pentagon, Hexagon, Octagon> Init;

std::cout << "Введите:\n 1 - добавить пятиугольник\n 2 - добавить шестиугольник\n 3 - добавить восьмиугольник\n";

std::cout << " 4 - распечатать буфер \n 5 - завершить программу\n";

std::cout << "Введите команду:";

while (std::cin >> m && m < 5 && m > 0) {

switch (m) {

case 1: {

Print();

std::cin >> x1 >> y1 >> side;

fig.push\_back(Init.Pentagon(x1, y1, side));

break;

}

case 2: {

Print();

std::cin >> x1 >> y1 >> side;

fig.push\_back(Init.Hexagon(x1, y1, side));

break;

}

case 3: {

Print();

std::cin >> x1 >> y1 >> side;

fig.push\_back(Init.Octagon(x1, y1, side));

break;

}

case 4: {

std::cout << fig;

break;

}

default:

break;

}

if (fig.size() == size\_buf) {

Server<FigVec>::get().publish(fig);

fig.clear();

sleep(1);

}

std::cout << "Введите команду:";

}

fig.clear();

Server<FigVec>::get().publish(fig);

thread.join();

}

hexagon.hpp

#pragma once

#include <cmath>

#include "figure.hpp"

class Hexagon: public Figure {

private:

int length;

public:

Hexagon() : Figure() {length = 0;}

Hexagon(double x1, double y1, int R) {

double angle,outX,outY;

for(int i = 1; i <= 6; i++) {// расчет точек

angle = 360 / 6;

outX = R \* cos(i\*angle) + x1;

outY = R \* sin(i\*angle) + y1;

points.emplace\_back(Pair(outX, outY));

}

length = R;

}

std::string Name() override {

return "Hexagon";

}

int Get() override {

return length;

}

std::vector<Pair> Coord() override {

return points;

}

};

figure.hpp

#pragma once

#include <iostream>

#include <vector>

#include <string>

struct Pair {

double x;

double y;

Pair(): x(0), y(0) {}

Pair(double a, double b): x(a), y(b) {}

};

class Figure {

protected:

std::vector<Pair> points;

public:

Figure() {}

Figure(double x1, double y1, int c) {

points.emplace\_back(Pair(x1,y1));

}

virtual std::string Name() {

return "Point";

}

virtual int Get() {

return 0;

}

virtual std::vector<Pair> Coord() {

return points;

}

};

std::ostream& operator<<(std::ostream &os, Pair p) {

os << '(' << p.x << ',' << p.y << ')';

return os;

}

std::ostream& operator<<(std::ostream &os, std::vector<Pair> pair) {

for (auto p: pair) os << " " << p;

return os;

}

std::ostream& operator<<(std::ostream &os, std::vector<std::shared\_ptr<Figure>> fig){

for (auto f: fig) os << f->Name() << std::endl << f->Coord() << std::endl;

return os;

}

1. **Вывод**

При выполнении лабораторной работы я научился работе с потоками. Еще я повторил лямбда-функции и научился работать с std::function. Реализовал класс Server и научился работать с мьютексами в Си++ .

Список литературы

1. cppreference.com [Электронный ресурс]. URL: https://en.cppreference.com/w/cpp/thread/thread (дата обращения: 21.12.20)
2. Метапрограммирование в C++ [Электронный ресурс]. URL: https://habr.com/ru/post/38622/(дата обращения: 21.12.20)
3. cppreference.com [Электронный ресурс]. URL: https://en.cppreference.com/w/cpp/thread/mutex (дата обращения: 21.12.20)