**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: Гребенков Дмитрий Игоревич

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.

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.
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:

* Вывод информации о фигурах в буфере на экран;
* Вывод информации о фигурах в буфере в файл. Для каждого буфера должен создаваться файл с уникальным именем.

1. Оба обработчика должны обрабатывать каждый введенный буфер. Т.е. после каждого заполнения буфера его содержимое должно выводиться как на экран, так и в файл.
2. Обработчики должны быть реализованы в виде лямбда-функций и должны хранится в специальном массиве обработчиков. Откуда и должны последовательно вызываться в потоке – обработчике.
3. В программе должно быть ровно два потока (thread). Один основной (main) и второй для обработчиков;
4. В программе должен явно прослеживаться шаблон Publish-Subscribe. Каждый обработчик должен быть реализован как отдельный подписчик.
5. 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.
6. Описание программы

Создан класс Figure, содержащий только виртуальные методы для вывода типа фигуры, вывода данных фигуры и записи этих данных в файл.

Созданы шаблонные классы для хранения трапеций, ромбов и пятиугольников. Они наследуются от класса Figure и содержат методы, раскрывающие методы класса Figure для каждой из заданных фигур.

Реализован класс Factory. Он содержит метод для создания фигур разных классов.

Создан класс Publisher. Он хранит очередь различных фигур и mutex, необходимый для работы с критической областью памяти. Методы класса Publisher позволяют вводить фигуры во внутренний буфер и “публиковать” сообщения, очищая этот буфер.

Также реализована функция для запуска в дополнительном потоке. В ней находятся обработчики сообщений в виде лямбда-функций, массив этих лямбда-функций, цикл для его применения и mutex для работы с критической областью памяти.

Взаимодействие с пользователем осуществляется при помощи меню.

В меню пользователь может выбрать, какую фигуру ввести в буфер. Также предусмотрена опция выхода из программы без публикации.

Примечание: Трапеция равнобедренная. Для ускорения вычислений и упрощения хранения на вводимые фигуры установлены некоторые ограничения. Во-первых, основания вводимых трапеций должны быть параллельны оси Ox. Во-вторых, большее основание трапеции должно быть снизу. В-третьих, диагонали вводимых ромбов должны быть параллельны осям Ox и Oy.

Трапеция задается координатами трех вершин.

Ромб задается координатами двух вершин.

Пятиугольник задается центром описанной окружности и радиусом.

1. Наборы тестов и результаты их выполнения

Демонстрация работы основных функций программы (вводятся фигуры разных типов, программа запускается с разными размерами буфера):

.\oop\_exercise\_08.exe 2

1. Enter trapezoid.

2. Enter rhombus.

3. Enter pentagon.

4. Quit the program without publishing.

Enter option: 1

Enter coordinates of left-top, left-bottom and right-bottom vertices:

1 2 0 0 4 0

Enter option: 2

Enter coordinates of left and bottom vertices:

0 1 2 0

Trapezoid: (1, 2)(0, 0)(4, 0)(3, 2)

Rhombus: (0, 1)(2, 0)(4, 1)(2, 2)

Enter option: 3

Enter coordinates of center and radius:

3 3 4

Enter option: 1

Enter coordinates of left-top, left-bottom and right-bottom vertices:

1 3 0 0 3 0

Pentagon: (-0.804225, 4.23607)(3, 7)(6.80422, 4.23607)(5.35114, -0.236069)(0.648861, -0.236069)

Trapezoid: (1, 3)(0, 0)(3, 0)(2, 3)

Enter option:

2

Enter coordinates of left and bottom vertices:

0 1 1 0

Enter option: 3

Enter coordinates of center and radius:

0 0 1

Rhombus: (0, 1)(1, 0)(2, 1)(1, 2)

Pentagon: (-0.951056, 0.309018)(0, 1)(0.951056, 0.309018)(0.587785, -0.809017)(-0.587785, -0.809017)

Enter option: 1

Enter coordinates of left-top, left-bottom and right-bottom vertices:

1 2 0 0 4 0

Enter option: 4

В результате работы программы были сгенерированы следующие файлы:

**0.txt:**

Trapezoid: (1, 2)(0, 0)(4, 0)(3, 2)

Rhombus: (0, 1)(2, 0)(4, 1)(2, 2)

**1.txt:**

Pentagon: (-0.804225, 4.23607)(3, 7)(6.80422, 4.23607)(5.35114, -0.236069)(0.648861, -0.236069)

Trapezoid: (1, 3)(0, 0)(3, 0)(2, 3)

**2.txt:**

Rhombus: (0, 1)(1, 0)(2, 1)(1, 2)

Pentagon: (-0.951056, 0.309018)(0, 1)(0.951056, 0.309018)(0.587785, -0.809017)(-0.587785, -0.809017)

Запуск с другим размером буфера:

.\oop\_exercise\_08.exe 1

0. Show menu.

1. Enter trapezoid.

2. Enter rhombus.

3. Enter pentagon.

4. Quit the program without publishing.

Enter option: 3

Enter coordinates of center and radius:

4 4 2

Pentagon: (2.09789, 4.61804)(4, 6)(5.90211, 4.61804)(5.17557, 2.38197)(2.82443, 2.38197)

Enter option: 2

Enter coordinates of left and bottom vertices:

-1 0 0 -1

Rhombus: (-1, 0)(0, -1)(1, 0)(0, 1)

Enter option: 4

В результате работы программы были сгенерированы следующие файлы:

**0.txt:**

Pentagon: (2.09789, 4.61804)(4, 6)(5.90211, 4.61804)(5.17557, 2.38197)(2.82443, 2.38197)

**1.txt:**

Rhombus: (-1, 0)(0, -1)(1, 0)(0, 1)

1. Листинг программы

/\*

Гребенков Д.И.

Группа М8О-207Б-19

Вариант: 12 (трапеция, ромб, пятиугольник).

Создать приложение, которое будет считывать из стандартного ввода данные фигур, согласно варианту задания, выводить их характеристики на экран и записывать в файл.

Программа должна:

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

Программа должна создавать классы, соответствующие введенным данных фигур;

Программа должна содержать внутренний буфер, в который помещаются фигуры. Для создания буфера допускается использовать стандартные контейнеры STL. Размер буфера задается параметром командной строки.

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

Обработка должна производиться в отдельном потоке;

Реализовать два обработчика, которые должны обрабатывать данные буфера:

Вывод информации о фигурах в буфере на экран;

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

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

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

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

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

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.

\*/

#include <iostream>

#include <cmath>

#include <memory>

#include <fstream>

#include <queue>

#include <thread>

#include <mutex>

#include <string>

#include <functional>

#include <vector>

const double pi = 3.14159;

std::ofstream wf;

template <typename T>

class Figure {

public:

virtual int type () = 0;

virtual void vertices () = 0;

virtual void writeInFile () = 0;

virtual ~Figure () {};

};

template <typename T>

class Trapezoid: public Figure <T> {

public:

std::pair <T, T> left\_top, left\_bottom, right\_bottom;

Trapezoid () {}

Trapezoid (T ltx, T lty, T lbx, T lby, T rbx, T rby) {

left\_top.first = ltx;

left\_top.second = lty;

left\_bottom.first = lbx;

left\_bottom.second = lby;

right\_bottom.first = rbx;

right\_bottom.second = rby;

}

virtual int type () {

return 1;

}

virtual void vertices () {

std::cout << "Trapezoid: (" << left\_top.first << ", " << left\_top.second << ")"

<< "(" << left\_bottom.first << ", " << left\_bottom.second << ")"

<< "(" << right\_bottom.first << ", " << right\_bottom.second << ")"

<< "(" << right\_bottom.first - left\_top.first + left\_bottom.first

<< ", " << left\_top.second << ")" << std::endl;

}

virtual void writeInFile () {

wf << "Trapezoid: (" << left\_top.first << ", " << left\_top.second << ")"

<< "(" << left\_bottom.first << ", " << left\_bottom.second << ")"

<< "(" << right\_bottom.first << ", " << right\_bottom.second << ")"

<< "(" << right\_bottom.first - left\_top.first + left\_bottom.first

<< ", " << left\_top.second << ")" << std::endl;

}

};

template <typename T>

class Rhombus: public Figure <T> {

public:

std::pair <T, T> left, bottom;

Rhombus () {}

Rhombus (T lx, T ly, T bx, T by) {

left.first = lx;

left.second = ly;

bottom.first = bx;

bottom.second = by;

}

virtual int type () {

return 2;

}

virtual void vertices () {

std::cout << "Rhombus: (" << left.first << ", " << left.second << ")";

std::cout << "(" << bottom.first << ", " << bottom.second << ")";

std::cout << "(" << 2 \* bottom.first - left.first << ", " << left.second << ")";

std::cout << "(" << bottom.first << ", " << 2 \* left.second - bottom.second << ")"

<< std::endl;

}

virtual void writeInFile () {

wf << "Rhombus: (" << left.first << ", " << left.second << ")"

<< "(" << bottom.first << ", " << bottom.second << ")"

<< "(" << 2 \* bottom.first - left.first << ", " << left.second << ")"

<< "(" << bottom.first << ", " << 2 \* left.second - bottom.second << ")"

<< std::endl;

}

};

template <typename T>

class Pentagon: public Figure <T> {

public:

std::pair <T, T> center;

T radius;

Pentagon () {}

Pentagon (T cx, T cy, T r) {

center.first = cx;

center.second = cy;

radius = r;

}

virtual int type () {

return 3;

}

virtual void vertices () {

std::cout << "Pentagon: (" << center.first - radius \* sin (72 \* pi / 180) << ", "

<< center.second + radius \* cos (72 \* pi / 180) << ")";

std::cout << "(" << center.first << ", " << center.second + radius << ")";

std::cout << "(" << center.first + radius \* sin (72 \* pi / 180) << ", " <<

center.second + radius \* cos (72 \* pi / 180) << ")";

std::cout << "(" << center.first + radius \* sin (36 \* pi / 180) << ", " <<

center.second - radius \* cos (36 \* pi / 180) << ")";

std::cout << "(" << center.first - radius \* sin (36 \* pi / 180) << ", " <<

center.second - radius \* cos (36 \* pi / 180) << ")" << std::endl;

}

virtual void writeInFile () {

wf << "Pentagon: (" << center.first - radius \* sin (72 \* pi / 180) << ", "

<< center.second + radius \* cos (72 \* pi / 180) << ")"

<< "(" << center.first << ", " << center.second + radius << ")"

<< "(" << center.first + radius \* sin (72 \* pi / 180) << ", "

<< center.second + radius \* cos (72 \* pi / 180) << ")"

<< "(" << center.first + radius \* sin (36 \* pi / 180) << ", "

<< center.second - radius \* cos (36 \* pi / 180) << ")"

<< "(" << center.first - radius \* sin (36 \* pi / 180) << ", "

<< center.second - radius \* cos (36 \* pi / 180) << ")" << std::endl;

}

};

template <typename T>

class Factory {

public:

static std::shared\_ptr <Figure <T> > create (int type) {

if ((type > 0) && (type < 4)) {

if (type == 1) {

T ltx, lty, lbx, lby, rbx, rby;

std::cout << "Enter coordinates of left-top, left-bottom and right-bottom vertices: "

<< std::endl;

std::cin >> ltx >> lty >> lbx >> lby >> rbx >> rby;

Trapezoid <T> \* temp (new Trapezoid <T> (ltx, lty, lbx, lby, rbx, rby));

return std::shared\_ptr <Figure <T> > (temp);

}

else if (type == 2) {

T lx, ly, bx, by;

std::cout << "Enter coordinates of left and bottom vertices: " <<

std::endl;

std::cin >> lx >> ly >> bx >> by;

Rhombus <T> \* temp (new Rhombus <T> (lx, ly, bx, by));

return std::shared\_ptr <Figure <T> > (temp);

}

else if (type == 3) {

T r, cx, cy;

std::cout << "Enter coordinates of center and radius: " <<

std::endl;

std::cin >> cx >> cy >> r;

Pentagon <T> \* temp (new Pentagon <T> (cx, cy, r));

return std::shared\_ptr <Figure <T> > (temp);

}

}

std::shared\_ptr <Figure <T> > temp;

return (temp);

}

};

int gen = 0;

std::queue <std::shared\_ptr <Figure <int> > > mes;

template <typename T>

class Publisher {

public:

std::queue <std::shared\_ptr <Figure <T> > > que;

std::mutex mtx;

void add (int type) {

std::shared\_ptr <Figure <T> > temp = Factory <T> :: create (type);

que.push (temp);

}

void publish () {

mtx.lock ();

while (!que.empty ()) {

mes.push (que.front ());

que.pop ();

}

mtx.unlock ();

}

};

template <typename T>

void threadF () {

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

name.append (".txt");

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

std::function <void (std::shared\_ptr <Figure <T> > figure)> print =

[] (std::shared\_ptr <Figure <T> > figure) {

figure->vertices ();

};

std::function <void (std::shared\_ptr <Figure <T> > figure)> write =

[] (std::shared\_ptr <Figure <T> > figure) {

figure->writeInFile ();

};

std::vector <std::function <void (std::shared\_ptr <Figure <T> >)> > arr = {print, write};

std::mutex mtx;

std::shared\_ptr <Figure <T> > ptr;

if (!mes.empty ()) {

mtx.lock ();

while (!mes.empty ()) {

ptr = mes.front ();

for (auto func: arr) {

func (ptr);

}

mes.pop ();

}

mtx.unlock ();

gen ++;

}

wf.close ();

}

void showMenu () {

std::cout << "0. Show menu." << std::endl;

std::cout << "1. Enter trapezoid." << std::endl;

std::cout << "2. Enter rhombus." << std::endl;

std::cout << "3. Enter pentagon." << std::endl;

std::cout << "4. Quit the program without publishing." << std::endl;

}

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

unsigned long long int bufferSize = std::stoi (std::string (argv [1]));

Publisher <int> pub;

int option = 0;

showMenu ();

while (option != 4) {

std::cout << "Enter option: ";

std::cin >> option;

switch (option) {

case 0: {

showMenu ();

break;

}

case 1: {

pub.add (1);

if (pub.que.size () >= bufferSize) {

pub.publish ();

std::thread thr (threadF <int>);

thr.join ();

}

break;

}

case 2: {

pub.add (2);

if (pub.que.size () >= bufferSize) {

pub.publish ();

std::thread thr (threadF <int>);

thr.join ();

}

break;

}

case 3: {

pub.add (3);

if (pub.que.size () >= bufferSize) {

pub.publish ();

std::thread thr (threadF <int>);

thr.join ();

}

break;

}

case 4: {

break;

}

default: {

std::cerr << "ERROR: incorrect option." << std::endl;

break;

}

}

}

return 0;

}

1. Вывод

Приобретены практические навыки параллельной обработки данных и синхронизации потоков.

Список литературы

1. Издатель-подписчик (шаблон проектирования) - Википедия [Электронный ресурс] URL: [https://ru.wikipedia.org/wiki/Издатель-подписчик\_(шаблон\_проектирования)](https://ru.wikipedia.org/wiki/%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).
2. C++ | Чтение и запись текстовых файлов [Электронный ресурс] URL: <https://metanit.com/cpp/tutorial/8.3.php> (дата обращения 27.12.2020).
3. Потоки, блокировки и условные переменные в C++11 [Часть 1] / Хабр [Электронный ресурс] URL: <https://habr.com/ru/post/182610/> (дата обращения 27.12.2020).
4. std::queue - cppreference.com [Электронный ресурс] URL: <https://en.cppreference.com/w/cpp/container/queue> (дата обращения 27.12.2020).
5. std::mutex - cppreference.com [Электронный ресурс] URL: <https://en.cppreference.com/w/cpp/thread/mutex> (дата обращения 27.12.2020).