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: Лагуткина Мария Сергеевна
Группа: М8О-206Б-19
Преподаватель: Чернышов Л. Н.
Дата: 20.12.2020
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.
4. При накоплении буфера они должны запускаться на асинхронную обработку, после чего буфер должен очищаться; Обработка должна производиться в отдельном потоке;
5. 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.
6. Оба обработчика должны обрабатывать каждый введенный буфер. Т.е. после каждого заполнения буфера его содержимое должно выводиться как на экран, так и в файл.
7. Обработчики должны быть реализованы в виде лямбда-функций и должны хранится в специальном массиве обработчиков. Откуда и должны последовательно вызываться в потоке – обработчике.
8. В программе должно быть ровно два потока (thread). Один основной (main) и второй для обработчиков;
9. В программе должен явно прослеживаться шаблон Publish-Subscribe. Каждый обработчик должен быть реализован как отдельный подписчик;
10. 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 задания: 10.
Типы фигур: квадрат, прямоугольник, трапеция.

2 Описание программы

Для решения задачи был реализован класс Figure и классы-наследники для работы с квадратом, прямоугольником и трапецией. Нужные фигуры создаются при вводе их типа и координат вершин. Далее проверяются введенные вершины на корректность для указанного типа фигур. Вершины хранятся в векторе, как и указатели на фигуры.
Перед началом ввода фигур, вводится число - размер буфера, при накоплении которого запускаются обработчики, а буфер очищается. Обработка производится в классе Document, в конструкторе которого создается дополнительный поток для записи в файл и в консоль. Если буфер переполнился, то основной поток замораживается, пока не произойдет запись. Также из предыдущий лабораторной взяты классы фигур и обработчик тестов.

3 Набор тестов

Программа получает на ввод команду и, если это нужно, аргументы к ней. Ознакомится со списком команд можно через команду 7:

```
Enter choice:
0 -create document
1 -add
2 -remove
3 -undo
4 -save in file
5 -load out file
6 -print all figures
7 -print usage

Types of figures:
0 -square
1 -rectangle
2 -trapezoid
```

Тест №1.
Вначале теста вводится размер буфера, затем вводятся фигуры: название, затем

координаты вершин. Замечание: более подробно корректность ввода этих фигур тестировалась в лабораторной работе №3.

```
5
Trapezoid 0 0 3 4 6 4 9 0
Rectangle 1 1 1 3 3 3 3 1
Square 0 0 0 1 1 1 1 0
Square -1 0 0 1 1 0 0 -1
Rectangle 1 2 1 5 4 5 4 2
Trapezoid 0 0 3 4 6 4 9 0
Rectangle 1 1 1 3 3 3 3 1
Square 0 0 0 1 1 1 1 0
Square -1 0 0 1 1 0 0 -1
Rectangle 1 2 1 5 4 5 4 2
Trapezoid 0 0 3 4 6 4 9 0
Rectangle 1 1 1 3 3 3 3 1
Square 0 0 0 1 1 1 1 0
Square -1 0 0 1 1 0 0 -1
Rectangle 1 2 1 5 4 5 4 2
Trapezoid 0 0 3 4 6 4 9 0
Rectangle 1 1 1 3 3 3 3 1
Square 0 0 0 1 1 1 1 0
Square -1 0 0 1 1 0 0 -1
Rectangle 1 2 1 5 4 5 4 2
```

4 Результаты выполнения тестов

Тест №1.

```
5
Trapezoid 0 0 3 4 6 4 9 0
Rectangle 1 1 1 3 3 3 3 1
Square 0 0 0 1 1 1 1 0
Square -1 0 0 1 1 0 0 -1
Rectangle 1 2 1 5 4 5 4 2
Trapezoid 0 0 3 4 6 4 9 0
1:
Trapezoid:
(0,0),(3,4),(6,4),(9,0)
Rectangle:
```

```
(1,1),(1,3),(3,3),(3,1)
Square:
(0,0),(0,1),(1,1),(1,0)
Square:
(-1,0),(0,1),(1,0),(0,-1)
Rectangle:
(1,2),(1,5),(4,5),(4,2)

Rectangle 1 1 1 3 3 3 3 1
Square 0 0 0 1 1 1 1 0
Square -1 0 0 1 1 0 0 -1
Rectangle 1 2 1 5 4 5 4 2
2Trapezoid 0 0 3 4 6 4 9 0
:
Trapezoid:
(0,0),(3,4),(6,4),(9,0)
Rectangle:
(1,1),(1,3),(3,3),(3,1)
Square:
(0,0),(0,1),(1,1),(1,0)
Square:
(-1,0),(0,1),(1,0),(0,-1)
Rectangle:
(1,2),(1,5),(4,5),(4,2)

Rectangle 1 1 1 3 3 3 3 1
Square 0 0 0 1 1 1 1 0
Square -1 0 0 1 1 0 0 -1
Rectangle 1 2 1 5 4 5 4 2
3Trapezoid 0 0 3 4 6 4 9 0
:
Trapezoid:
(0,0),(3,4),(6,4),(9,0)
Rectangle:
(1,1),(1,3),(3,3),(3,1)
Square:
(0,0),(0,1),(1,1),(1,0)
Square:
(-1,0),(0,1),(1,0),(0,-1)
Rectangle:
(1,2),(1,5),(4,5),(4,2)
```

```
Rectangle 1 1 1 3 3 3 3 1
Square 0 0 0 1 1 1 1 0
Square -1 0 0 1 1 0 0 -1
Rectangle 1 2 1 5 4 5 4 2
4:
Trapezoid:
(0,0),(3,4),(6,4),(9,0)
Rectangle:
(1,1),(1,3),(3,3),(3,1)
Square:
(0,0),(0,1),(1,1),(1,0)
Square:
(-1,0),(0,1),(1,0),(0,-1)
Rectangle:
(1,2),(1,5),(4,5),(4,2)
```

5 Листинг программы

```
1  // Лагуткина Мария
2  //вариант 10: квадрат, прямоугольник, трапеция
3  #include <iostream>
4  #include <fstream>
5  #include <exception>
6  #include <list>
7  #include <vector>
8  #include <string>
9  #include <utility>
10 #include <memory>
11 #include <future>
12 #include <condition_variable>
13
14 using namespace std;
15
16 struct Vertex {
17     double x = 0;
18     double y = 0;
19     void Print() const {
20         cout << "(" << x << ", " << y << ")";
21     }
22 };
23
24 istream& operator>>(istream& input, Vertex& vertex) {
25     input >> vertex.x >> vertex.y;
```

```
26      return input;
27  }
28
29  ostream& operator<<(ostream& output, const Vertex& vertex) {
30      output << vertex.x << " " << vertex.y;
31      return output;
32  }
33
34  enum class FigureType {
35      Square,
36      Rectangle,
37      Trapezoid
38  };
39
40  string TypeToString(FigureType type) {
41      switch (type) {
42      case FigureType::Trapezoid:
43          return "Trapezoid";
44      case FigureType::Square:
45          return "Square";
46      case FigureType::Rectangle:
47          return "Rectangle";
48      default:
49          throw  runtime_error("Undefined figure type");
50      }
51  }
52
53  class Figure {
54  public:
55      Figure() = default;
56      Figure(vector<Vertex> vec) : stats_(vec) {}
57      void Print() const {
58          cout << TypeToString(GetType()) << ":\n ";
59          bool is_first = true;
60          for (const Vertex& v : stats_) {
61              if (!is_first) {
62                  cout << ", ";
63              }
64              is_first = false;
65              v.Print();
66          }
67          cout << "\n";
68      }
69      virtual FigureType GetType() const = 0;
70      virtual ~Figure() {}
71      friend  istream& operator>>(istream& input, Figure& rb);
72      friend  ostream& operator<<(ostream& output, const Figure& rb);
73
74
```

```
protected:
    vector<Vertex> stats_;
};

istream& operator>>(istream& input, Figure& figure) {
    for (Vertex& v : figure.stats_) {
        input >> v;
    }
    return input;
}

ostream& operator<<(ostream& output, const Figure& figure) {
    output << static_cast<int>(figure.GetType()) << " ";
    for (const Vertex& v : figure.stats_) {
        output << v << " ";
    }
    return output;
}

class Square : public Figure {
public:
    Square() : Figure(vector<Vertex>(4)) {}
    Square(Vertex a, Vertex  b, Vertex  c, Vertex  d) : Figure({ a, b, c ,d}) {}
    virtual FigureType GetType() const override { return FigureType::Square; }
};

class Rectangle : public Figure {
public:
    Rectangle() : Figure(vector<Vertex>(4)) {}
    Rectangle(Vertex a, Vertex  b, Vertex  c, Vertex d) : Figure({ a, b, c, d }) {}
    virtual FigureType GetType() const override { return FigureType::Rectangle; }
};

class Trapezoid : public Figure {
public:
    Trapezoid() : Figure(vector<Vertex>(4)) {}
    Trapezoid(Vertex a, Vertex  b, Vertex  c, Vertex  d) : Figure({ a, b, c, d}) {}
    virtual FigureType GetType() const override { return FigureType::Trapezoid; }
};

class Document {
public:
    Document() {
        fut_con = async([&]() {Logger(); });
    }
    void Set_size(size_t s) { critical_size = s; }
    ~Document() {
        production_stopped = true;
        cv_consumption.notify_all();
```

```
124         fut_con.get();
125     }
126     void Export() {
127         static int i = 0;
128         ++i;
129         ofstream out("log" + to_string(i) + ".txt");
130         for (const auto& ptr : data) {
131             out << *ptr << "\n";
132         }
133         out << "\n";
134     }
135     void Add(shared_ptr<Figure> figure_ptr) {
136         std::unique_lock<std::mutex> lock(m);
137         cv_production.wait(lock, [&] { return data.size() < critical_size; });
138         data.push_back(move(figure_ptr));
139         cv_consumption.notify_all();
140     }
141 
142     void Print() const {
143         static int i = 0;
144         ++i;
145         cout << i << ":\n";
146         for (const auto& ptr : data) {
147             ptr->Print();
148         }
149         cout << "\n";
150     }
151 private:
152     list<shared_ptr<Figure>> data;
153     size_t critical_size = 3;
154     mutex m;
155     condition_variable cv_production;
156     condition_variable cv_consumption;
157     future<void> fut_con;
158     bool production_stopped = false;
159     void Logger() {
160         while (!production_stopped) {
161             std::unique_lock<std::mutex> lock(m);
162             cv_consumption.wait(lock, [&] { return data.size() >= critical_size ||
    production_stopped; });
163             if (!data.empty()) {
164                 Print();
165                 Export();
166                 data.clear();
167             }
168             cv_production.notify_all();
169         }
170     }
171 };
```

```
172 double lenght(Vertex a, Vertex b) {
173     return sqrt((b.x - a.x)*(b.x - a.x) + ((b.y - a.y)*(b.y - a.y)));
174 }
175 int CorrectInput(Vertex& a, Vertex& b, Vertex& c, Vertex& d, int fig) {
176     if (((abs(a.x - c.x) < numeric_limits<double>::epsilon()) &&
177         (abs(a.y - c.y) < numeric_limits<double>::epsilon())) ||
178         ((abs(b.x - d.x) < numeric_limits<double>::epsilon()) &&
179         (abs(b.y - d.y) < numeric_limits<double>::epsilon()))) {
180         throw runtime_error("ERROR INPUT\n");
181     }
182     double len_ab = lenght(a, b);
183     double len_bc = lenght(b, c);
184     double len_cd = lenght(c, d);
185     double len_da = lenght(d, a);
186     double len_ac = lenght(a, c);
187     double len_bd = lenght(b, d);
188     if (fig == 1) {
189         if (abs(len_cd - len_ab) < numeric_limits<double>::epsilon() &&
190             abs(len_cd - len_bc) < numeric_limits<double>::epsilon() &&
191             abs(len_cd - len_da) < numeric_limits<double>::epsilon() &&
192             abs(len_bd - len_ac) < numeric_limits<double>::epsilon()) {
193             return 0;
194         }
195         throw runtime_error("ERROR INPUT\n");
196     }
197     //проверка коректности прямоугольника: равенсво противоположных сторон и
    диагоналей
198     if (fig == 2) {
199         if (abs(len_ab - len_cd) < numeric_limits<double>::epsilon() && abs(len_bc -
    len_da) < numeric_limits<double>::epsilon() &&
200             abs(len_ac - len_bd) < numeric_limits<double>::epsilon()) {
201             return 0;
202         }
203         throw runtime_error("ERROR INPUT\n");
204     }
205     //проверка коректности равнобедренной трапеции: равенсво боковых строн и
    диагоналей
206     if (fig == 3) {
207         if (abs(len_ab - len_cd) < numeric_limits<double>::epsilon() &&
208             abs(len_ac - len_bd) < numeric_limits<double>::epsilon()) {
209             return 0;
210         }
211         throw runtime_error("ERROR INPUT\n");
212     }
213     return 0;
214 }
215
216 void InputError() {
217     throw runtime_error("ERROR INPUT\n");
```

```
}

int main() {
        try {
            Document doc;
            string cmd;
            size_t size = 3;
            cin >> size;
            doc.Set_size(size);
            while (cin >> cmd) {

                if (cmd == "Square") {
                    Vertex a, b, c, d;
                    if (!(cin >> a >> b >> c >> d)) {
                        InputError();
                    }
                    CorrectInput(a, b, c, d, 1);
                    doc.Add(make_shared<Square>(a, b, c, d));
                }
                else if (cmd == "Rectangle") {
                    Vertex a, b, c, d;
                    if (!(cin >> a >> b >> c >> d)) {
                        InputError();
                    }
                    CorrectInput(a, b, c, d, 2);
                    doc.Add(make_shared<Rectangle>(a, b, c, d));
                }
                else if (cmd == "Trapezoid") {
                    Vertex a, b, c, d;
                    if (!(cin >> a >> b >> c >> d)) {
                        InputError();
                    }
                    CorrectInput(a, b, c, d, 3);
                    doc.Add(make_shared<Trapezoid>(a, b, c, d));
                }
                else {
                    cout << cmd << " is not a command\n";
                }
            }
        }
        catch (exception& ex) {
            cerr << ex.what();
        }
    return 0;
}
```

6 GitHub

https://github.com/marianelia/MAI/tree/main/OOP/oop_exercise_08

7 Вывод

Выполняя лабораторную работу, я узнала как работать с потоками в C++, а также как распараллеливать обработку данных.

Список литературы

[1] *Курс «Основы разработки на C++: белый пояс».* [Электронный ресурс] URL: https://www.coursera.org/learn/c-plus-plus-white (дата обращения 05.12.2020).

[2] *Документация Microsoft по C++.* [Электронный ресурс] URL: https://docs.microsoft.com/ru-ru/?view=msvc-16 (дата обращения 05.12.2020).