Московский Авиационный Институт
(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

Лабораторная работа
по курсу «Объектно-ориентированное программирование»
III Семестр

Задание 8
Вариант 24
Ассинхронное программирование

Студент:	Синявский А.В
Группа:	М80-208Б-18
Преподаватель:	Журавлёв А.А
Оценка:	
Дата:	

Москва 2019

1. Код программы на языке C++

1.1 subscribers.h

```
#ifndef OOP_LAB8_SUBSCRIBERS_H
#define OOP_LAB8_SUBSCRIBERS_H

class SubscriberInterface {
public:
    virtual void output(std::vector<std::shared_ptr<figures::Figure>>) = 0;
    virtual ~SubscriberInterface() = default;
};

class ConsolePrint : public SubscriberInterface {
public:
    void output(std::vector<std::shared_ptr<figures::Figure>> buffer) override {
        for (auto& figure : buffer) {
            figure->PrintOut(std::cout);
        }
    }
};

class DocumentPrint : public SubscriberInterface {
public:
    DocumentPrint() : a(1) {}
    void output(std::vector<std::shared_ptr<figures::Figure>> buffer) override {
        std::string name = std::to_string(a);
        name += ".txt";
        std::ofstream file;
        file.open(name);
        if(!file.is_open())
        {
            file.clear();
            file.open(name, std::ios::out);
            file.close();
            file.open(name);
        }
        for (auto &figure : buffer) {
            figure->PrintOut(file);
        }
        ++a;
    }
private:
    int a;
};

#endif //OOP_LAB8_SUBSCRIBERS_H
```

1.2 main.cpp

```
#include <iostream>
#include <vector>
#include <memory>
#include <string>
#include <thread>
#include <mutex>
#include <condition_variable>
#include "figures/triangle.h"
#include "figures/square.h"
#include "figures/octagon.h"
#include "subscribers.h"

class Factory {
public:
    std::map<std::string, std::shared_ptr<FactoryInterface>> plants;
    Factory() {
        plants.emplace("triangle", std::make_shared<Triangle_factory>());
        plants.emplace("square", std::make_shared<Square_factory>());
        plants.emplace("octagon", std::make_shared<Octagon_factory>());
    }
};

int main(int args, char* argv[]) {
    if (args < 2) {
        std::cout << "USAGE: ./oop_exercise_08 [size of buffer]\n";
        return -1;
    }
    long buf_size = strtol(argv[1], nullptr, 10);
    std::vector<std::shared_ptr<figures::Figure>> buffer;
    buffer.reserve(buf_size);
    Factory factory;
    std::string command;

    std::condition_variable cv;
    std::condition_variable cv2;
    std::mutex mutex;
    bool done = false;
    int a = 1;

    std::vector<std::shared_ptr<SubscriberInterface>> subscribers;
    subscribers.push_back(std::make_shared<ConsolePrint>());
    subscribers.push_back(std::make_shared<DocumentPrint>());

    std::thread subscriber([&]() {
        std::unique_lock<std::mutex> subscriber_lock(mutex);
        while(!done) {
            cv.wait(subscriber_lock);
            if (done) {
```

```
            cv2.notify_all();
            break;
        }
        for (int i = 0; i < 2; ++i) {
            subscribers[i]->output(buffer);
        }
        buffer.resize(0);
        ++a;
        cv2.notify_all();
    }
});

while(command != "quit") {
    std::cin >> command;
    if (command == "quit") {
        done = true;
        cv.notify_all();
        break;
    } else if (command == "triangle" || command == "square" || command == "octagon") {
        auto tmp = factory.plants[command]->Figure_create(std::cin);
        std::unique_lock<std::mutex> main_lock(mutex);
        buffer.push_back(tmp);
        if (buffer.size() == buffer.capacity()) {
            cv.notify_all();
            cv2.wait(main_lock);
        }
    } else {
        std::cout << "no such figure\n";
    }
}
subscriber.join();
return 0;
}
```

2. Ссылка на репозиторий на GitHub

https://github.com/Siegmeyer1/oop_exercise_08

3. Набор тестов

1)
square 0 0 3 3
square 0 0 3 3
square 0 0 3 3
quit

2)
square 0 0 3 3
triangle 0 0 0 2 2 0
triangle 2 2 0 2 2 0
square 0 0 5 5
quit

4. Результаты тестов

1)
anri@andrew-HP-250-G6:~/Documents/Github_repositories/OOP_lab8/build$ ./oop_exercise_08 3
square 0 0 3 3
square 0 0 3 3
square 0 0 3 3
(3; 0), (0; 0), (0; 3), (3; 3)
(3; 0), (0; 0), (0; 3), (3; 3)
(3; 0), (0; 0), (0; 3), (3; 3)
quit
anri@andrew-HP-250-G6:~/Documents/Github_repositories/OOP_lab8/build$ ls
1.txt CMakeCache.txt CMakeFiles cmake_install.cmake Makefile oop_exercise_08
anri@andrew-HP-250-G6:~/Documents/Github_repositories/OOP_lab8/build$ cat 1.txt
(3; 0), (0; 0), (0; 3), (3; 3)
(3; 0), (0; 0), (0; 3), (3; 3)
(3; 0), (0; 0), (0; 3), (3; 3)
anri@andrew-HP-250-G6:~/Documents/Github_repositories/OOP_lab8/build$

2)
anri@andrew-HP-250-G6:~/Documents/Github_repositories/OOP_lab8/build$ ./oop_exercise_08 2
square 0 0 3 3
triangle 0 0 0 2 2 0

(3; 0), (0; 0), (0; 3), (3; 3)
(0; 0), (0; 2), (2; 0)
triangle 2 2 0 2 2 0
square 0 0 5 5

(2; 2), (0; 2), (2; 0)
(5; 0), (0; 0), (0; 5), (5; 5)
quit
anri@andrew-HP-250-G6:~/Documents/Github_repositories/OOP_lab8/build$ cat 1.txt
(3; 0), (0; 0), (0; 3), (3; 3)
(0; 0), (0; 2), (2; 0)
anri@andrew-HP-250-G6:~/Documents/Github_repositories/OOP_lab8/build$ cat 2.txt
(2; 2), (0; 2), (2; 0)
(5; 0), (0; 0), (0; 5), (5; 5)
anri@andrew-HP-250-G6:~/Documents/Github_repositories/OOP_lab8/build$

5. Объяснение работы программы

Программа состоит из 2-х потоков. Основной считывает команды и добавляет фигуры в буфер. Когда буфер заполняется, основной поток уведомляет второй, и начинает ждать уведомления от него. Второй поток вызывает у двух подписчиков метод их работы. Один подписчик выводит содержимое буфера в консоль, второй – создаёт файл, имя которого зависит от того, в какой раз вызывается функция, и записывает содержимое буфера туда. Затем буфер очищается и второй поток уведомляет основной, после чего цикл повторяется снова до ввода команды quit.

Вывод

Проделав работу, я изучил основы ассинхронной работы программы, узнал о том, что такое unique_lock и condition_variable и для чего они нужны. Узнал о принципе publish-subscribe.