ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет компьютерных наук
Образовательная программа бакалавриата «Программная инженерия»

Многопоточное программирование. Задача о картинной галерее.

Пояснительная записка

Исполнитель:

студент группы БПИ191 (2 подгруппа),

Кирюхин Андрей Алексеевич

Москва 2020

Москва 2020

СОДЕРЖАНИЕ

1. ВВЕДЕНИЕ

1.1. Текст задания

Задача о картинной галерее. Вахтер следит за тем, чтобы в картинной галерее было не более 50 посетителей. Для обозрения представлены 5 картин. Посетитель ходит от картины к картине, и если на картину любуются более чем десять посетителей, он стоит в стороне и ждет, пока число желающих увидеть картину не станет меньше. Посетитель может покинуть галерею. Создать многопоточное приложение, моделирующее работу картинной галереи.

1.2. Алгоритм программы

В программе реализованы три класса: посетитель картинной галереи (Visitor), вахтер (WatchMan), картинная галерея (Gallery).

Рассмотрим сценарий взаимодействия субъектов.

После создания посетителя он добавляется в очередь в картинную галерею через вахтера. Если в галерее на данный момент меньше 50 человек, то вахтер пропускает этого посетителя в галерею, где он начинает обход картин. Если же в галерее 50 человек и более, то вахтер не пропускает посетителя, а ждет пока какой-либо посетитель не покинет галерею.

Итак, посетитель галереи начинает обход картин. При создании посетителя, у него генерируется очередность обхода картин. Узнав, на какую картину он собирался смотреть на данный момент, он идет к ней. Если этой картиной любуются больше, чем 10 посетителей, то он ждет пока не освободится место. При освобождении места у картины (галерея уведомляет об этом посетителя), посетитель идет любоваться картиной. Затем он переходит к следующей картине и описанный сценарий повторяется.

После обхода всех картин посетитель покидает галерею, о чем уведомляет вахтера, что позволяет ему впустить следующего посетителя из очереди.

Программа продолжает работать, пока вахтер наблюдает людей в очереди или в картинной галерее, после того как людей не осталось, вахтер ждет какое-то время и закрывает галерею (программа заканчивает свою работу).

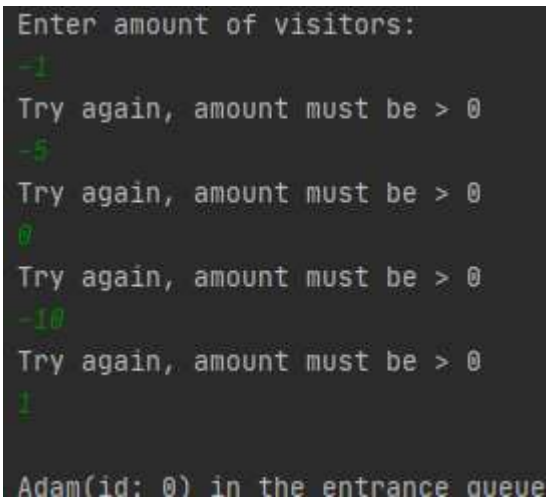
Каждый описанный выше шаг сопровождается выводом состояния в консоль.

1.3. Ограничение входных данных

На вход программе подается количество посетителей картинной галереи, это число должно быть больше 0. Программа будет повторять ввод числа посетителей, пока пользователь не введет корректное значение.

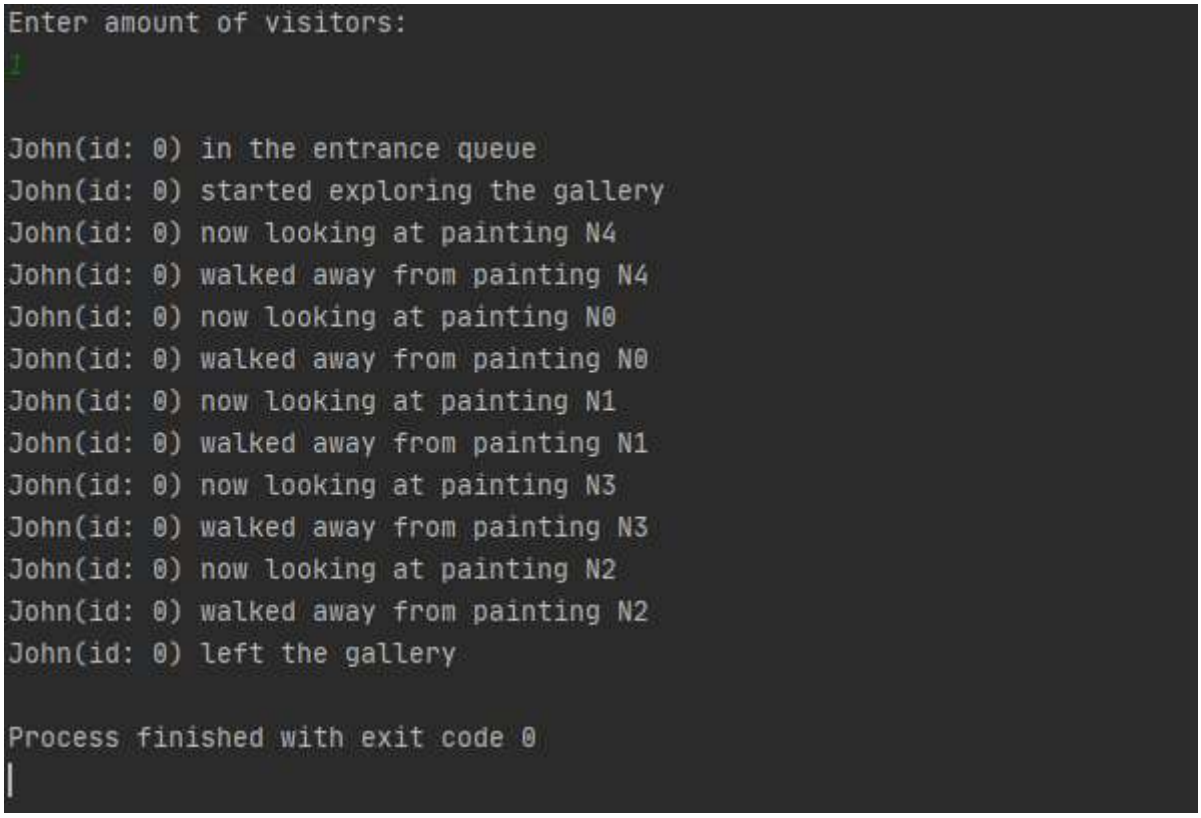
2. ТЕСТИРОВАНИЕ ПРОГРАММЫ

2.1. Неправильный ввод количества посетителей



Как можно заметить из скриншота, программа на начнет свою работу, пока не будет введено корректное число посетителей галереи.

2.2. Проверка работоспособности программы при одном посетителе.



Программа корректно отработала и завершила свою работу, а из скриншота можно увидеть каждый шаг данного посетителя. Как можно заметить, он обходил картины в следующем порядке: 4-0-1-3-2, после чего покинул галерею.

2.3. Проверка работоспособности при добавлении более 50 посетителей.

```
Enter amount of visitors:
60

John(id: 0) in the entrance queue
John(id: 1) in the entrance queue
John(id: 0) started exploring the gallery
John(id: 0) now looking at painting N3
Max(id: 2) in the entrance queue
John(id: 1) started exploring the gallery
John(id: 1) now looking at painting N0
John(id: 3) in the entrance queue
Max(id: 2) started exploring the gallery
```

Полный отчет программы содержит более 600 строк, поэтому проследим за некоторыми конкретными посетителями, для проверки корректности программы.

1. John(id: 0)

```
John(id: 0) in the entrance queue
John(id: 1) in the entrance queue
John(id: 0) started exploring the gallery
John(id: 0) now looking at painting N3
John(id: 0) walked away from painting N3
John(id: 0) now looking at painting N1
Max(id: 38) now looking at painting N1
John(id: 0) walked away from painting N1
John(id: 0) now looking at painting N2
John(id: 3) walked away from painting N1
John(id: 41) now looking at painting N1
John(id: 0) walked away from painting N2
John(id: 0) now looking at painting N0
Max(id: 4) walked away from painting N3
John(id: 0) walked away from painting N0
John(id: 0) now looking at painting N4
Max(id: 4) walked away from painting N1
John(id: 0) walked away from painting N4
John(id: 0) left the gallery
```

Как можно заметить, данный посетитель корректно обошел все картины.

2. Max(id: 53)

```
Max(id: 53) in the entrance queue
```

```
Max(id: 53) started exploring the gallery
Max(id: 53) in the queue to painting N3
```

```
Max(id: 53) now looking at painting N3
```

```
Max(id: 53) walked away from painting N3
Max(id: 53) now looking at painting N2
```

```
Max(id: 53) walked away from painting N2
Max(id: 53) now looking at painting N4
```

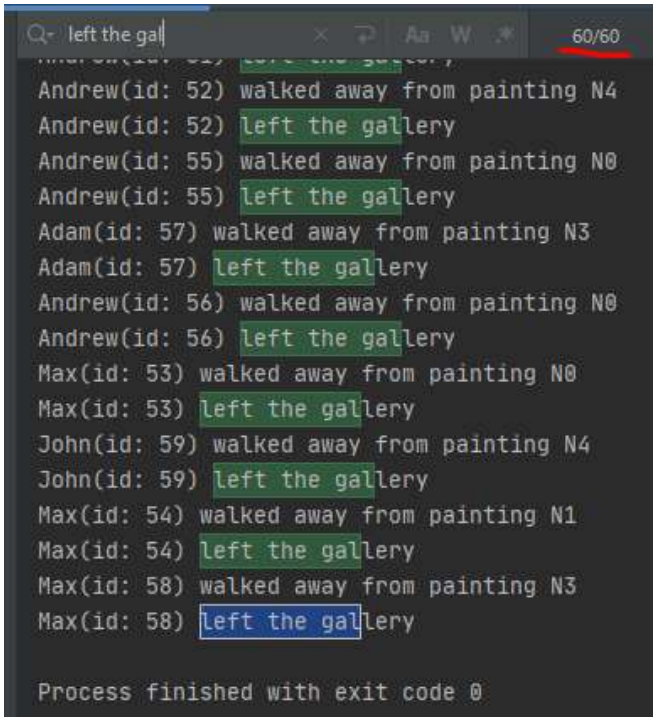
```
Max(id: 53) walked away from painting N4
Max(id: 53) now looking at painting N1
```

```
Max(id: 53) walked away from painting N1
Max(id: 53) now looking at painting N0
```

```
Max(id: 53) walked away from painting N0
Max(id: 53) left the gallery
```

Данный посетитель также корректно обошел все картины в галерее, так же здесь продемонстрирован сценарий ожидания, пока место у картины не освободится.

Также по количеству людей, который покинули галерею, можно сказать, что ни один поток не попал в dead lock и все они корректно закончили свою работу:



3. СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1) StackOverflow [Электронный ресурс] // URL: https://stackoverflow.com/ (Режим доступа: свободный).
2) SoftCraft Архитектура вычислительных систем [Электронный ресурс] // URL: http://softcraft.ru/edu/comparch/ (Режим доступа: свободный).
3) CyberForum [Электронный ресурс] // URL: https://www.cyberforum.ru/ (Режим доступа: свободный).
4) Cplusplus [Электронный ресурс] // URL: http://www.cplusplus.com/ (Режим доступа: свободный)

ПРИЛОЖЕНИЕ 1

ТЕКСТ ПРОГРАММЫ

```
/*
    Кирюхин Андрей Алексеевич, БПИ191
    Вариант - 18
    Условие задачи:
    Задача о картинной галерее. Вахтер следит за тем, чтобы в
    картинной галерее было не более 50 посетителей. Для обозрения
    представлены 5 картин. Посетитель ходит от картины к картине, и если на
    картину любуются более чем десять посетителей, он стоит в стороне и ждет,
    пока число желающих увидеть картину не станет меньше. Посетитель может
    покинуть галерею. Создать многопоточное приложение, моделирующее
    работу картинной галереи
*/
#include <iostream>
#include <thread>
#include <mutex>
#include <queue>
#include <string>
#include <algorithm>
#include <random>
#include <chrono>
#include <time.h>
#include <condition_variable>


using namespace std;

unsigned seed = chrono::system_clock::now().time_since_epoch().count();
auto rng = default_random_engine{seed};
mutex locker;
std::condition_variable cv[5];


class WatchMan;

//Класс посетителя картинной галереи
class Visitor {
public:
    //При создании посетителя, у него в случайном порядке генерируется путь обхода
всех картин в галерее.
    Visitor(string _name, int _id) {
        name = _name;
        id = _id;
        shuffle(std::begin(path), std::end(path), rng);
    }

    string GetName() {
        return name + "(id: " + to_string(id) + ")";
    }

    void startExploring(WatchMan &watchMan);
    bool isWaiting = false;
private:
    string name;
    int id;
    int currentPainting;
    vector<int> path = {0, 1, 2, 3, 4};
    WatchMan *watchMan;

    //Получение следующей картины для посещения
```

```
    int getNextPainting() {
        if (path.empty()) return -1;
        int painting = path.back();
        path.pop_back();
        return painting;
    }

    void lookAtCurrentPainting();

    void moveToNextPainting();
};

//Класс картинной галереи
class Gallery {
public:
    //Инкремент количества посетителей смотрящих определенную картину
    void paintingAddVisitor(int index) {
        locker.lock();
        paintings[index]++;
        locker.unlock();
    }

    //Уход посетителя от конкретной картины
    void leavePaintingVisitor(int index) {
        locker.lock();
        if (paintings[index] != 0) {
            paintings[index]--;
        }
        if(!paintingsQueue[index].empty()){
            paintingsQueue[index].front()->isWaiting = false;
            paintingsQueue[index].pop();
        }
        locker.unlock();
        cv[index].notify_all();
    }

    //Получение количество посетителей, смотрящих определенную картину
    int getVisitorsNumAt(int index) {
        locker.lock();
        int result = paintings[index];
        locker.unlock();
        return result;
    }

    void addVisitorToPaintingQueue(Visitor &visitor, int index){
        locker.lock();
        paintingsQueue[index].push(&visitor);
        locker.unlock();
    }

private:
    int paintings[5] = {0};
    queue<Visitor*> paintingsQueue[5];
};

Gallery mainGallery;

class WatchMan {
public:
    //Запуск посетителя в галерею:
    //Если в галерее меньше 50 посетителей, то создается новый поток для посетителя и
он начинает обход галереи
    void getNextPersonFromQueue() {
        if (!visitorsQueue.empty() && amountInGallery < 50) {
```

```
            Visitor visitor = visitorsQueue.front();
            thread v(&Visitor::startExploring, visitor, ref(*this));
            v.detach();
            visitorsQueue.pop();
            amountInGallery++;
        }
    }

    //Добавление посетителя в очередь в галерею
    void addVisitorToQueue(Visitor &visitor) {
        locker.lock();
        visitorsQueue.push(visitor);
        cout << visitor.GetName() + " in the entrance queue" << endl;
        locker.unlock();
        getNextPersonFromQueue();
    }

    //Проверка не осталось ли посетителей в очереди и в самой галереи, установлено
пятисекундное окно
    //чтобы избежать ситуации, когда посетитель ушел из очереди и ещё не попал в
галерею, а программа уже завершилась
    void isActivePeople() {
        while (true) {
            locker.lock();
            if (visitorsQueue.empty() && amountInGallery == 0){
                this_thread::sleep_for(chrono::seconds(2));
                if (visitorsQueue.empty() && amountInGallery == 0) return;
            }
            locker.unlock();
            this_thread::sleep_for(chrono::seconds(5));
        }
    }

    //Уведомление вахтера, что посетитель покинул галерею
    void notifyLeft() {
        locker.lock();
        amountInGallery--;
        if (amountInGallery < 50) getNextPersonFromQueue();
        locker.unlock();
    }

private:
    queue<Visitor> visitorsQueue;
    int amountInGallery = 0;
};

//Начало прохода по картинной галерее
void Visitor::startExploring(WatchMan &_watchMan) {
    watchMan = &_watchMan;
    locker.lock();
    cout << GetName() << " started exploring the gallery" << endl;
    locker.unlock();

    moveToNextPainting();
}

//Метод просмотра текущей картины посетителем и переход к следующей картине
void Visitor::lookAtCurrentPainting() {
    mainGallery.paintingAddVisitor(currentPainting);

    locker.lock();
    cout << GetName() << " now looking at painting N" << to_string(currentPainting)
<< endl;
    locker.unlock();
```

```
    this_thread::sleep_for(chrono::seconds(3));

    mainGallery.leavePaintingVisitor(currentPainting);

    locker.lock();
    cout << GetName() << " walked away from painting N" << to_string(currentPainting)
<< endl;
    locker.unlock();
    moveToNextPainting();
}

//Переход посетителя к следующей картине, если на картину смотрят больше 10 человек,
//то он ждет в стороне пока количество смотрящих не будет < 10
void Visitor::moveToNextPainting() {
    if (path.empty()) {
        locker.lock();
        cout << GetName() << " left the gallery" << endl;
        locker.unlock();
        watchMan->notifyLeft();
        return;
    }
    currentPainting = getNextPainting();
    if (mainGallery.getVisitorsNumAt(currentPainting) > 10) {
        locker.lock();
        cout << GetName() << " in the queue to painting N" << currentPainting <<
endl;
        locker.unlock();

        isWaiting = true;
        mainGallery.addVisitorToPaintingQueue(*this, currentPainting);

        unique_lock<std::mutex> lck(locker);
        while(isWaiting) {
            cv[currentPainting].wait(lck);
        }
        lck.unlock();
        lookAtCurrentPainting();
    } else {
        lookAtCurrentPainting();
    }
}

int main() {
    cout<<"Enter amount of visitors:"<<endl;
    int amount;
    while(true){
        cin >> amount;
        if(amount <= 0) cout<<"Try again, amount must be > 0"<<endl;
        else break;
    }
    cout<<endl;
    srand(static_cast<unsigned int>(time(0)));
    string names[] = {"Adam", "John", "Max", "Andrew", "Alex", "Bill", "Mary", "Ann",
"Kate", "William", "Ella", "Lucy",
                      "Kinsley", "Kendall"};
    WatchMan watchMan;
    for (int i = 0; i < amount; ++i) {
        Visitor visitor(names[rand() % names->length()], i);
        watchMan.addVisitorToQueue(visitor);
    }

    //Создаем поток, проверяющий каждые 5 секунд, не остались ли ещё люди в очереди
или в галерее, при завершении
    //этого потока можно завершать программу.
```

```
    thread watchManThread(&WatchMan::isActivePeople, ref(watchMan));
    watchManThread.join();
}
```