Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени
Н. Э. Баумана (национальный исследовательский университет)»

Факультет «Робототехника и комплексная автоматизация»
Кафедра «Системы автоматизированного проектирования»

Отчет по вычислительному практикуму

Выполнил: Студент: Степанов Н. Н.
Группа: РК6-35Б

Проверил: Берчун Ю. В.
Дата: 09.01.21
Подпись: _____

Москва, 2020 г

Отчет по вычислительному практикуму № 3

Задание:

Требуется разработать программу, реализующую дискретно-событийное моделирование системы, рассмотренной в задании 2 домашнего задания №4.

Результатом работы программы должен быть лог-файл, содержащий записи типа: «В момент времени 12.345 транзакт с идентификатором 1 вошёл в модель», «В момент времени 123.456 транзакт с идентификатором 123 встал в очередь 1», «В момент времени 234.567 транзакт с идентификатором 234 занял устройство 2», «В момент времени 345.678 транзакт с идентификатором 345 освободил устройство 1», «В момент времени 456.789 транзакт с идентификатором 456 вышел из модели».

Описание входных данных:

Через директиву препроцессора #define определяются значения R1, G1, B1.

Описание алгоритма:

Для реализации поставленной задачи используется механизм многопоточного программирования языка C++.

Изначально описывается класс *Cashier*, то есть класс, симулирующий работу реальной кассы. В публичном поле данного класса присутствуют формально описанный конструктор по умолчанию и конструктор инициализации. Кроме того в данном поле присутствуют методы класса *job()* и *waiting()*, задача которых, согласно их названиям – обработка транзактов и ожидание транзакта в очереди. Так же в публичной области находятся данные о транзактах: целочисленная переменная *id* – номер текущего обслуживаемого на устройстве транзакта и объект типа *std::queue <int>,* необходимый для формирования очереди к устройствам. В приватной части класса содержатся данные, отвечающие за реализацию механизма многопоточного программирования – объекты класса *mutex()*, обеспечивающие возможность остановки работы вспомогательных потоков. Помимо этого, в приватной области класса содержатся данные, характерные для каждой из касс – границы распределения времени обработки транзактов и номер устройства обработки.

В основном потоке программы создаются два объекта класса *Cashier*, то есть класс кассы, затем создаются 4 вспомогательных потока, каждый из которых связан с методом обработки/ожидания каждой из касс. В момент создания с помощью механизма условных переменных *condition_variable*, которые являются глобальными, каждый из потоков находится в “спящем” режиме. Затем создается файл для записи лога программы – *log.txt*.

В основном цикле главного потока реализуется рабочий процесс поступления заявки в модель, а именно: информирование об этом и перенаправление транзакта на поступление в очередь – то есть в метод waiting*()*. При этом в зависимости от того, в какой из касс меньшая очередь – происходит пробуждение потока ожидания меньшей очереди. В самом методе waiting*()* реализуется информирование о том, что транзакт встал в очередь и добавление id транзакта в очередь библиотеки стандартных шаблонов, как только соответствующие информирование и добавление произошло, активируется поток обслуживания транзакта, т. е. метод *job()*. При этом поток ожидания заявок вновь становится неактивным до следующего добавления транзакта в очередь. В потоке обработки транзактов происходит информирование о поступлении транзакта на обслуживание, затем поток “засыпает” на время обслуживания, после пробуждения происходит удаление элемента из очереди и информирование о том, что соответствующий транзакт освободил устройство и покинул модель. Поток засыпает тогда и

только тогда, когда длина в очереди становится равной 0, то есть тогда, когда нет никаких транзактов в очереди на обслуживание. Весь описанный алгоритм работает для двух касс параллельно, выполняясь циклически.

Основной поток программы работает до тех пор, пока модельное время не превысит значения ***3600***, при этом генерация всех случайных величин, равномерно распределенных на заданных промежутках, происходит с помощью функции *uniform()*, которая с помощью функции *rand()* генерирует случайные величины. Чтобы генерация случайных величин была действительно случайной, а не псевдослучайной, в начале работы программы с помощью комбинации функций srand() и time (), а именно - *srand(time(NULL)* устанавливается семя генератора случайных чисел на текущее календарное время. Так же основной поток программы засыпает на время прихода заявки, тем самым симулируя интервальность приходов заявок в модель.

Так как потоки работают практически параллельно, то вывод о текущем состоянии того или иного транзакта теоретически может накладываться на вывод о состоянии другого транзакта. Тем не менее, целостная картина моделирования так или иначе достаточна ясна.

Так как потенциально возможна ситуация, когда основной поток отрабатывает раньше других, необходимо учесть этот момент и избежать преждевременного прерывания работы других потоков. Поэтому после работы основного цикла, в главном потоке программы с помощью метода *join ()* объекта потока – *thread(),* происходит ожидание основным потоком других потоков.

Затем происходит закрытие файла на запись и возврат программой нуля в качестве идентификатора корректной работы программы.

Описание выходных данных:

Файл log.txt, содержащий в себе информацию о продвижении транзактов в модели.

Код программы:

```
#include <iostream>
#include <cstdlib>
#include <ctime>
#include <thread>
#include <chrono>
#include <queue>
#include <iomanip>
#include <condition_variable>
#include <fstream>
#define R1 9
#define G1 7
#define B1 8
#include <mutex>
std::condition_variable cv1;
std::condition_variable cv2;
std::condition_variable cv3;
std::condition_variable cv4;

bool state1 = false;
bool state2 = false;
bool state3 = false;
bool state4 = false;
bool first_cashier_use () {return state1;}
bool second_cashier_use() {return state2;}
bool first_queue_is_exist() {return state3;}
bool second_queue_is_exist() {return state4;}
```

```
double globaltime = 0;


double uniform (double min, double max)
{
    return (double)(rand())/RAND_MAX*(max-min)+min;
}


class Cashier
{
public:
    Cashier ();
    Cashier (int, int, int);
    void waiting (std::ofstream&);
    void job(std::ofstream&);
    std::queue <int> q;
    int id;

private:
    int min, max, q_num;
    std::mutex mtx, mtx2;
    double worktime;
    double arrivetime;

};


void Cashier::job(std::ofstream& f)
{

    while (globaltime < 3600) {
        std::unique_lock<std::mutex> ulm1(mtx);
        if (q_num == 1)
            cv1.wait(ulm1, first_cashier_use);
        else
            cv2.wait(ulm1, second_cashier_use);
        arrivetime = globaltime;
        id = q.front();
        worktime = uniform(min, max);
        f<< "В момент времени " << std::fixed << std::setprecision(3) <<
arrivetime
                 << " транзакт с идентификатором "
                 << id << " занял устройство " << q_num << std::endl;
        std::this_thread::sleep_for(std::chrono::duration<double,
std::nano>(worktime));
        f << "В момент времени " << std::fixed << std::setprecision(3) <<
arrivetime + worktime
                 << " транзакт с идентификатором "
                 << id << " освободил устройство " << q_num <<" и покинул
модель" << std::endl;
        q.pop();
        if (q.size() == 0 and q_num == 1)
            state1 = false;
        else if (q.size() == 0 and q_num == 2)
            state2 = false;
    }


}

void Cashier::waiting(std::ofstream &f)
{
    while (globaltime < 3600) {
```

```
        std::unique_lock<std::mutex> ulm1(mtx2);
        if (q_num == 1)
            cv3.wait(ulm1, first_queue_is_exist);
        else
            cv4.wait(ulm1, second_queue_is_exist);
        arrivetime = globaltime;
        f << "В момент времени " << std::fixed << std::setprecision(3) <<
arrivetime
                  << " транзакт с идентификатором "
                  << q.front() << " занял очередь " << q_num << std::endl;

        if (q_num == 1)
        {state3 = false;
            state1 = true;
            cv1.notify_one();}
        else if (q_num == 2)
        {state4 = false;
            state2 = true;
            cv2.notify_one();}

    }
}

Cashier::Cashier (int n, int x,int num)
{
    min = n;
    max = x;
    q_num = num;
}

int main (int argc, char* argv[])
{
    int id = 1;
    system("chcp 65001");
    srand(time(NULL));

    Cashier q1 (R1, R1 + G1 + B1, 1);
    Cashier q2 (G1, R1 + G1 + B1, 2);

    std::ofstream fout;

    fout.open("log.txt", std::ios::trunc);



    std::thread th1 ([&]()
                     {
                         q1.job(fout);
                     });

    std::thread th2 ([&]()
                     {
                         q1.waiting(fout);
                     });

    std::thread th3 ([&]()
                     {
                         q2.job(fout);
                     });

    std::thread th4 ([&]()
                     {
                         q2.waiting(fout);
                     });
```

```
    while (globaltime < 3600)
    {
        double time = uniform(0, 24);
        globaltime += time;

        fout << "В момент времени " << std::fixed << std::setprecision(3) <<
globaltime << " транзакт с идентификатором "
                 << id << " вошел в модель" << std::endl;


        if (q1.q.size() <= q2.q.size())
        {
            q1.q.push(id);
            state3 = true;
            cv3.notify_all();

        }
        else
        {
            q2.q.push(id);
            state4 = true;
            cv4.notify_all();
        }

        id += 1;
        std::this_thread::sleep_for(std::chrono::duration<double,
std::nano>(time));

    }

    th1.join();
    th2.join();
    th3.join();
    th4.join();

    fout.close();
    return 0;
}
```

Список используемой литературы:

1. Липпман, Стенли Б., Лажойе, Жози, Му, Барбара Э. Язык программирования C++. Базовый курс. 5-е изд.: пер. с англ. - М.:ООО «И.Д.Вильямс», 2014 – 1120 с. [Stanley B. Lippman, Josee Lajoie,Barbara E. Moo. C++ Primer. – Addison-Wesley, 2013]

2. Страуструп Б. Язык программирования C++. 4-е изд.: пер. с англ. – М:Бином, 2015 – 1136 с. [Stroustrup B. The C++ Programming Language - Addison-Wesley, 2000.]

3. Волосатова Т.М., Родионов С.В. «Объектно-ориентировнное программирование на C++» // Режим доступа: http://bigor.bmstu.ru/?cnt/?doc=VU/base.cou (дата обращения 02.01.2021).