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

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Лабораторная работа № 5

Дисциплина: Моделирование

Тема: Моделирование работы информационного центра

Студент: Платонова О. С.

Группа: ИУ7-75Б

Оценка (баллы): _____

Преподаватель: Рудаков И. В.

Москва, 2021 г.

Цель работы: моделирование работы системного центра. В информационный центр приходят клиенты через интервал времени 10 +- 2 минуты. Если все три имеющихся оператора заняты, клиенту отказывают в обслуживании. Операторы имеют разную производительность и могут обеспечивать обслуживание среднего запроса пользователя за 20 +- 5; 40 +- 10; 40 +- 20. Клиенты стремятся занять свободного оператора с максимальной производительностью. Полученные запросы сдаются в накопитель. Откуда выбираются на обработку. На первый компьютер запросы от 1 и 2-ого операторов, на второй – запросы от 3-его. Время обработки запросов первым и 2-м компьютером равны соответственно 15 и 30 мин. Промоделировать процесс обработки 300 запросов.

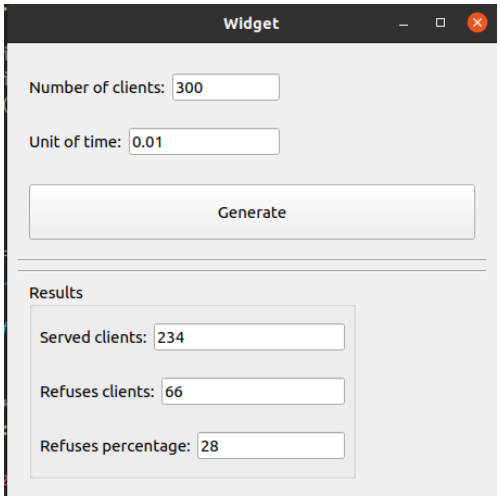
Процесс обработки

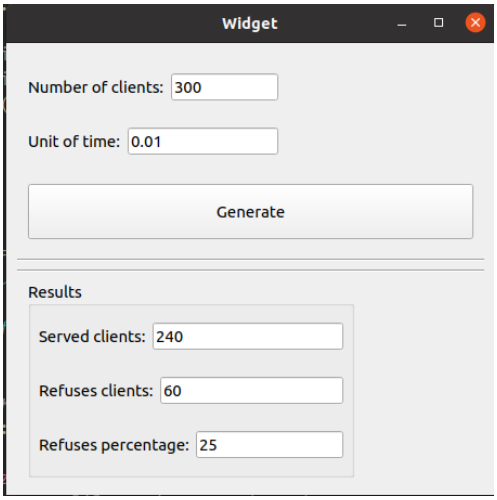
В цикле (пока количество текущих клиентов не превосходит заданного количества клиентов) генератором выполняется создание клиента. В случае успешного создания, клиент распределяется между операторами. Для этого необходимо в цикле для каждого существующего оператора выполнить проверку на его готовность принять клиента, и, в случае успеха, отправить клиента оператору. В случае неудачи, следует рассматривать клиента как отказ.

Далее каждому оператору следует перенести своего клиента в соответствующий накопитель. Оттуда клиент будет отправлен на соответствующий компьютер и обслужен им.

После выполнения цикла, следует проверить каждого оператора на наличие необслуженных клиентов. И в случае наличия, отправить их на накопитель, а затем на обслуживание компьютером.

Результаты





Листинг

Файл generator

```
7  class Generator
8  {
9  public:
10     Generator();
11
12     void setRnd(std::default_random_engine *re);
13     void setEvenDistribution(float _a, float _b);
14
15     double even();
16     bool produceClient(double dt);
17
18 private:
19     float a, b;
20     double endTime;
21     std::default_random_engine *gnt;
22 };
```

```
34 bool Generator::produceClient(double dt)
35 {
36     endTime -= dt;
37
38     if (endTime <= 0) {
39         endTime = even();
40         return true;
41     }
42     return false;
43 }
```

Файл operator

```
7  class Operator
8  {
9  public:
10     Operator(std::default_random_engine *re, std::vector<int> *_queue);
11
12     void setEvenDistribution(float _a, float _b);
13     double even();
14
15     bool isFree();
16     void acceptRequest();
17     void sendRequest(double dt);
18
19 private:
20     std::default_random_engine *gnt;
21     std::vector<int> *queue;
22     double endTime;
23     float a, b;
24     bool busy;
25 };
```

```
33 void Operator::acceptRequest()
34 {
35     busy = true;
36     endTime = even();
37 }
38
39 void Operator::sendRequest(double dt)
40 {
41     endTime -= dt;
42
43     if (busy && endTime <= 0) {
44         busy = false;
45         (*queue).push_back(1);
46     }
47 }
```

Файл computer

```cpp
class Computer
{
public:
    Computer(std::default_random_engine *re, std::vector<int> *_queue);

    void setTime(const double _time);
    bool serveClient(double dt);

private:
    std::default_random_engine *gnt;
    std::vector<int> *queue;
    double time, endTime;
    bool busy;
};
```

```cpp
bool Computer::serveClient(double dt)
{
    endTime -= dt;

    if (busy && endTime <= 0) {
        busy = false;
        return true;
    }

    if (!busy && (*queue).size() != 0) {
        (*queue).erase((*queue).begin());
        endTime = time;
        busy = true;
    }

    return false;
}
```

Файл model

```cpp
class Model
{
public:
    Model();
    Result generate(const int numRequests, double step);

private:
    std::vector<int> storage1, storage2;
    Generator generator;
    std::vector<Operator> operators;
    std::vector<Computer> computers;

    std::random_device rd;
    std::default_random_engine gnt;

    bool distributeClient();
    int serveClients(const double step);
    void storageTransfer(const double step);
};
```

```cpp
Result Model::generate(const int numClients, double step)
{
    int curClients = 0, numServed = 0, numRefusals = 0;

    while (curClients < numClients) {
        bool client = generator.produceClient(step);
        if (client) {
            curClients++;
            numRefusals += distributeClient();
        }

        storageTransfer(step);
        numServed += serveClients(step);
    }

    while (numRefusals + numServed < numClients) {
        storageTransfer(step);
        numServed += serveClients(step);
    }

    Result res;
    res.Service = numServed;
    res.Refusals = numRefusals;
    res.PerRefusals = (numServed) ? numRefusals * 100 / numServed : 0;

    return res;
}
```

```cpp
//Распределение клиента по операторам
bool Model::distributeClient()
{
    size_t i = 0;
    for (; i < operators.size(); i++) {
        if (operators[i].isFree()) {
            operators[i].acceptRequest();
            break;
        }
    }

    if (i == operators.size()) {
        return true;
    }
    return false;
}

//Отправка клиента в накопитель
void Model::storageTransfer(const double step)
{
    for (size_t i = 0; i < operators.size(); i++) {
        operators[i].sendRequest(step);
    }
}

//Обслуживание клиента компьютерами
int Model::serveClients(const double step)
{
    int numProceesed = 0;
    for (size_t i = 0; i < computers.size(); i++) {
        numProceesed += computers[i].serveClient(step);
    }

    return numProceesed;
}
```