**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»

# ОТЧЕТ

По лабораторной работе №5

По курсу: «Моделирование»

Тема: «Моделирование информационного центра»

Студент: Пронин А. С.

Группа: ИУ7-72Б

Преподаватель: Рудаков И. В.

Оценка: _____

Москва

2022

# Задание

В информационный центр приходят клиенты через интервалы времени $10 \pm 2$ минуты. Если все три, имеющихся оператора, заняты, клиенту отказывают в обслуживании. Операторы имеют разную производительность. И могут обеспечивать обслуживание среднего запроса за $20 \pm 5$, $40 \pm 10$, $40 \pm 20$ минут. Клиенты стремятся занять свободного оператора с максимальной производительностью. Полученные запросы сдаются в приемные накопители, откуда они выбираются для обработки. На 1-ый компьютер запросы от 2-ого и 1-ого оператора. На 2-ой компьютер от 3-его оператора. Время обработки на 1-ом и 2-ом компьютере равны соответственно 15 и 30 минутам.

Смоделировать процесс обработки 300 запросов.

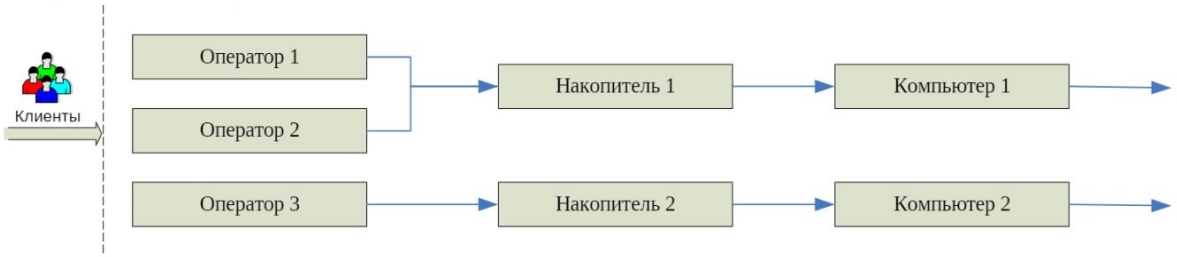


Рис. 1: Схема модели

# 1 Теория

## 1.1 Структурная схема модели

В процессе взаимодействия клиентов и центра возможны:

- режим нормального обслуживания (когда клиент выбирает одного из свободных операторов, но предпочитает того, у которого меньше номер);
- режим отказа.

Структурная схема модели представленна на рисунке 1.1.

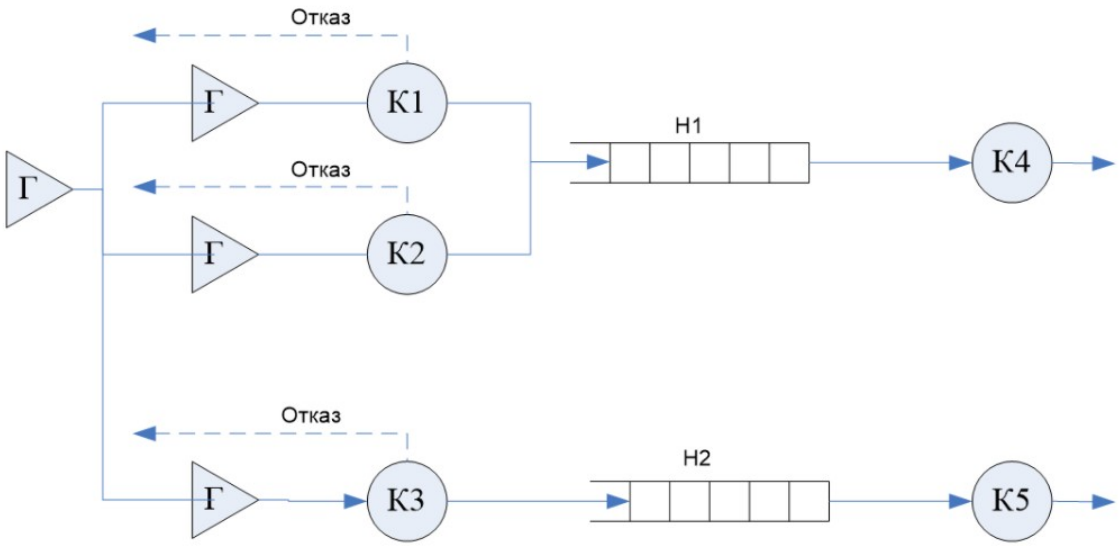


Рис. 1.1: Структурная схема модели

Согласно условию время обработки заявки оператором подчиняется закону равномерного распределения, а компьютер выполняет каждую обработку за фиксированное время.

Эндогенные переменные – время обработки заданий $i$-ым оператором ($i = \overline{0;2}$) и время решения задания на $j$-ом компьютере ($j = \overline{0;1}$).

Экзогенные переменные – $n_0$ = числу обслуженных клиентов, $n_1$ = числу клиентов получивших отказ.

Уравнения модели: вероятность отказа = $n_1/(n_0 + n_1)$

За единицу дискретного времени выбрана 0.01 минуты.

# 2 Текст программы

В листингах 2.1–2.6 представлен код программы, отвечающий за моделирование.

Листинг 2.1: Код генератора

```
class Generator
{
  public Generator(double _a, double _b)
  {
    a = _a;
    b = _b;
  }

  public bool isReady(double t)
  {
    return (t >= gen_time);
  }

  public Request genRequest()
  {
    Request new_r = new Request(gen_time);
    updateGenTime();
    return new_r;
  }

  private void updateGenTime()
  {
    double t_i = a + (b - a) * rnd.NextDouble();
    gen_time += Math.Round(t_i, 2);
  }

  public double a;
  public double b;
  public double gen_time = 0;

  private Random rnd = new Random();
}
```

Листинг 2.2: Код заявки

```
class Request
{
  public Request(double t)
  {
    create_time = t;
  }

  public double create_time = 0;
  public double serve_time = -1;
}
```

Листинг 2.3: Код очереди заявок

```
class ReqQue : Queue<Request>
{
  public bool push(Request r)
  {
    Enqueue(r);
    if (Count > max_size)
    {
      max_size = Count;
      return true;
    }
    return false;
  }
  public Request pop()
  {
    return Dequeue();
  }

  public int max_size = 0;
}
```

Листинг 2.4: Код обслуживающего аппарата

```
class Service
{
  public Service(double _a, double _b)
  {
    a = _a;
    b = _b;
  }

  public bool isFree(double t)
  {
    return (t >= free_time);
  }

  public void serve(Request r, double t)
  {
    updateFreeTime(t);
    r.serve_time = free_time;
  }

  public void updateFreeTime(double t)
  {
    double t_i = a + (b - a) * rnd.NextDouble();
    free_time = t + Math.Round(t_i, 2);
  }

  public double a;
  public double b;
  public double free_time = 0;

  private Random rnd = new Random();
}
```

Листинг 2.5: Код событий

```
abstract class BaseEvent : IComparable<BaseEvent>
{
  public BaseEvent(double _time)
  {
    time = _time;
  }

  public int CompareTo(BaseEvent other)
  {
    if (this.time > other.time)
      return 1;
    else
      return -1;
  }
  abstract public void Handle(EventModel model);

  public double time;
}

class EClientCame : BaseEvent
{
  public EClientCame(Request _client) : base(_client.create_time
   )
  {
    client = _client;
  }

  public override void Handle(EventModel model)
  {
    model.clients_n++;

    if (model.clients_n < model.max_clients_n)
    {
      Request next_client = model.clients.genRequest();
      model.addEvent(new EClientCame(next_client));
    }


```

```
40      for (int i = 0; i < model.operators.Count; i++)
41        if (model.operators[i].isFree(this.time))
42        {
43          model.operators[i].serve(this.client, this.time);
44          Request client_req = new Request(this.client.serve_time)
    ;
45          model.addEvent(new EClientServed(client_req, i));
46
47          model.served_n++;
48          return;
49        }
50
51      model.refused_n++;
52    }
53
54    private Request client;
55  }
56
57  class EClientServed : BaseEvent
58  {
59    public EClientServed(Request _client_request, int _operator_id
     ) : base(_client_request.create_time)
60    {
61      client_req = _client_request;
62      operator_id = _operator_id;
63    }
64
65    public override void Handle(EventModel model)
66    {
67      int target_storage_id = (operator_id == 2) ? 1 : 0;
68      model.storages[target_storage_id].push(client_req);
69      if (model.computers[target_storage_id].isFree(this.time))
70        model.addEvent(new EClientReqServed(this.time,
     target_storage_id));
71    }
72
73    private Request client_req;
74    private int operator_id;
75  }
76
77
```

```
78 class EClientReqServed : BaseEvent
79 {
80   public EClientReqServed(double _time, int _storage_id) : base(
     _time)
81   {
82     id = _storage_id;
83   }
84
85   public override void Handle(EventModel model)
86   {
87     if (model.storages[id].Count != 0)
88     {
89       Request client_req = model.storages[id].pop();
90       model.computers[id].serve(client_req, this.time);
91       model.addEvent(new EClientReqServed(model.computers[id].
     free_time, id));
92     }
93   }
94
95   private int id;
96 }
```

Листинг 2.6: Код событийной модели

```
class EventModel
{
  public EventModel(int _max_clients_n = 300)
  {
    clients = new Generator(8, 12);
    operators = new List<Service> { new Service(15, 25), new
   Service(30, 50), new Service(20, 60) };
    storages = new List<ReqQue> { new ReqQue(), new ReqQue()};
    computers = new List<Service> { new Service(15, 15), new
   Service(30, 30) };
    max_clients_n = _max_clients_n;
  }

  public double getRefuseProb()
  {
    reset();

    while (events.Count > 0)
    {
      BaseEvent e = events[0];
      events.RemoveAt(0);

      end_time = e.time;
      e.Handle(this);
    }

    return (double)refused_n / (refused_n + served_n);
  }

  public void addEvent(BaseEvent e)
  {
    events.Add(e);
    events.Sort();
  }
```

```
39    public string getResultsStr()
40    {
41      string res = String.Format("Обслужено клиентов: {0:D} | ",
      this.served_n);
42      res += String.Format("Отказов: {0:D} | ", this.refused_n);
43      res += String.Format("Вероятность отказа: {0:F4} | ", Math.
      Round((double)refused_n / (refused_n + served_n), 4));
44      res += String.Format("Время моделирования: {0:F4} мин[] ",
      this.end_time);
45      return res;
46    }
47
48    private void reset(int _max_clients_n = 300)
49    {
50      clients.gen_time = 0;
51      for (int i = 0; i < operators.Count; i++)
52        operators[i].free_time = 0;
53      for (int i = 0; i < storages.Count; i++)
54        storages[i].max_size = 0;
55      for (int i = 0; i < computers.Count; i++)
56        computers[i].free_time = 0;
57      max_clients_n = _max_clients_n;
58      end_time = 0;
59      refused_n = 0;
60      served_n = 0;
61      clients_n = 0;
62      Request first_client = clients.genRequest();
63      events = new List<BaseEvent> { new EClientCame(first_client)
        };
64    }
65    public Generator clients;
66    public List<Service> operators;
67    public List<ReqQue> storages;
68    public List<Service> computers;
69    public int max_clients_n = 300;
70    public double end_time = 0;
71    public int refused_n = 0;
72    public int served_n = 0;
73    public int clients_n = 0;
74    private List<BaseEvent> events = new List<BaseEvent>();
75  }
```

# 3 Результаты

Пример работы программы приведен на рисунке 3.1.

```
Обслужено клиентов: 236 | Отказов: 64 | Вероятность отказа: 0,2133 | Время моделирования: 3069,91 [мин]
Обслужено клиентов: 231 | Отказов: 69 | Вероятность отказа: 0,2300 | Время моделирования: 3059,87 [мин]
Обслужено клиентов: 233 | Отказов: 67 | Вероятность отказа: 0,2233 | Время моделирования: 3014,72 [мин]
Обслужено клиентов: 236 | Отказов: 64 | Вероятность отказа: 0,2133 | Время моделирования: 3046,28 [мин]
Обслужено клиентов: 240 | Отказов: 60 | Вероятность отказа: 0,2000 | Время моделирования: 3080,82 [мин]
Обслужено клиентов: 238 | Отказов: 62 | Вероятность отказа: 0,2067 | Время моделирования: 3080,00 [мин]
Обслужено клиентов: 235 | Отказов: 65 | Вероятность отказа: 0,2167 | Время моделирования: 3035,91 [мин]
Обслужено клиентов: 235 | Отказов: 65 | Вероятность отказа: 0,2167 | Время моделирования: 3039,08 [мин]
Обслужено клиентов: 234 | Отказов: 66 | Вероятность отказа: 0,2200 | Время моделирования: 3011,32 [мин]
Обслужено клиентов: 239 | Отказов: 61 | Вероятность отказа: 0,2033 | Время моделирования: 3061,02 [мин]
Обслужено клиентов: 234 | Отказов: 66 | Вероятность отказа: 0,2200 | Время моделирования: 3026,54 [мин]
Обслужено клиентов: 234 | Отказов: 66 | Вероятность отказа: 0,2200 | Время моделирования: 3039,25 [мин]
Обслужено клиентов: 242 | Отказов: 58 | Вероятность отказа: 0,1933 | Время моделирования: 3045,46 [мин]
Обслужено клиентов: 235 | Отказов: 65 | Вероятность отказа: 0,2167 | Время моделирования: 3050,13 [мин]
Обслужено клиентов: 237 | Отказов: 63 | Вероятность отказа: 0,2100 | Время моделирования: 3036,76 [мин]
Обслужено клиентов: 239 | Отказов: 61 | Вероятность отказа: 0,2033 | Время моделирования: 3072,06 [мин]
Обслужено клиентов: 239 | Отказов: 61 | Вероятность отказа: 0,2033 | Время моделирования: 3032,26 [мин]
Обслужено клиентов: 236 | Отказов: 64 | Вероятность отказа: 0,2133 | Время моделирования: 3004,30 [мин]
Обслужено клиентов: 240 | Отказов: 60 | Вероятность отказа: 0,2000 | Время моделирования: 3032,65 [мин]
Обслужено клиентов: 235 | Отказов: 65 | Вероятность отказа: 0,2167 | Время моделирования: 2997,80 [мин]
Обслужено клиентов: 243 | Отказов: 57 | Вероятность отказа: 0,1900 | Время моделирования: 3031,97 [мин]
Обслужено клиентов: 230 | Отказов: 70 | Вероятность отказа: 0,2333 | Время моделирования: 3015,13 [мин]
Обслужено клиентов: 235 | Отказов: 65 | Вероятность отказа: 0,2167 | Время моделирования: 3033,12 [мин]
Обслужено клиентов: 237 | Отказов: 63 | Вероятность отказа: 0,2100 | Время моделирования: 3067,46 [мин]
Обслужено клиентов: 235 | Отказов: 65 | Вероятность отказа: 0,2167 | Время моделирования: 3057,33 [мин]
Нажмите enter для завершения...
```

Рис. 3.1: Пример работы программы