МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»**

МОДЕЛИРОВАНИЕ СИСТЕМ. ВАРИАНТ 2

ОТЧЕТ О ПРАКТИКЕ

студента 4 курса 411 группы
направления 02.03.02 — Фундаментальная информатика и информационные технологии
факультета КНиИТ
Аношкина Андрея Алексеевича

Проверил
к. ф.-м. н., доцент _____ И. Е. Тананко

Саратов 2025

СОДЕРЖАНИЕ

1 ИМИТАЦИОННОЕ МОДЕЛИРОВАНИЕ СИСТЕМ И СЕТЕЙ МАССОВОГО ОБСЛУЖИВАНИЯ

1.1 Системы массового обслуживания

1.1.1 Постановка задачи

Дана СМО типа M | M | 2. Построить имитационную модель системы. На основании 1000 выборочных значений оценить $\overline{u}$ и $\overline{n}$.

1.1.2 Код

Model.h

```
#pragma once
#include <queue>
#include <vector>
#include <time.h>

using namespace std;

class Model {
private:
	queue<double> q;
	double timeOfTaskDoneFirst;
	double timeOfTaskDoneSecond;
	double mu;
	bool active_1;
	bool active_2;
	vector<double> timeInQueue;

	double generateTime();

public:
	Model(double mu);

	void addToQueue(double time);
	size_t getQueueLength();

	double getTimeOfTaskDoneFirst();
	double getTimeOfTaskDoneSecond();

	void analyze(double time);

	vector<double>& getTimeInQueue();
};
```

Model.cpp

```
#include "Model.h"

Model::Model(double mu) {
	srand(time(0));
	this->mu = mu;
	this->active_1 = false;
	this->active_2 = false;
	this->timeOfTaskDoneFirst = 100000;
	this->timeOfTaskDoneSecond = 100000;
}

void Model::addToQueue(double time) {
	this->q.push(time);
}

size_t Model::getQueueLength() {
	return this->q.size();
}

double Model::getTimeOfTaskDoneFirst() {
	return this->timeOfTaskDoneFirst;
}

double Model::getTimeOfTaskDoneSecond() {
	return this->timeOfTaskDoneSecond;
}

double Model::generateTime() {
	int r = rand();
	return -1.0 / this->mu * log(min(max(r, 1), RAND_MAX - 1) * 1.0 / RAND_MAX);
}

void Model::analyze(double time) {
	if (this->active_1)
		if (time >= this->timeOfTaskDoneFirst && time <=
		 ↪  this->timeOfTaskDoneFirst) {
			this->active_1 = false;
			this->timeOfTaskDoneFirst = 100000;
		}

	if (!this->active_1)
		if (!this->q.empty()) {
```

```
42                      this->active_1 = true;
43                      this->timeInQueue.push_back(time - this->q.front());
44                      this->q.pop();
45                      this->timeOfTaskDoneFirst = time + this->generateTime();
46              }
47
48
49      if (this->active_2)
50              if (time >= this->timeOfTaskDoneSecond && time <=
                ↪  this->timeOfTaskDoneSecond) {
51                      this->active_2 = false;
52                      this->timeOfTaskDoneSecond = 100000;
53              }
54
55      if (!this->active_2)
56              if (!this->q.empty()) {
57                      this->active_2 = true;
58                      this->timeInQueue.push_back(time - this->q.front());
59                      this->q.pop();
60                      this->timeOfTaskDoneSecond = time + this->generateTime();
61              }
62  }
63
64  vector<double>& Model::getTimeInQueue() {
65      return this->timeInQueue;
66  }
```

main.cpp

```
1   #include <iostream>
2   #include <vector>
3   #include <time.h>
4   #include <cmath>
5   #include "Model.h"
6
7   using namespace std;
8
9   double lambda = 20;
10  double mu = 15;
11
12  double generateTask() {
13      int r = rand();
14      return -1.0 / lambda * log(min(max(r, 1), RAND_MAX - 1) * 1.0 / RAND_MAX);
15  }
```

```
16
17  double nextMoment(Model& model, double time_of_task) {
18          double min_time =
19                  (model.getTimeOfTaskDoneFirst() <= model.getTimeOfTaskDoneSecond()) ?
20                  model.getTimeOfTaskDoneFirst() : model.getTimeOfTaskDoneSecond();
21
22          double next_moment = (min_time <= time_of_task) ? min_time : time_of_task;
23
24          return next_moment;
25  }
26
27  double m(Model& model) {
28          float M = 0;
29          vector<double>::iterator begin = model.getTimeInQueue().begin();
30          vector<double>::iterator end = model.getTimeInQueue().end();
31          for (vector<double>::iterator it = begin; it != end; M += *it, ++it);
32
33          return M / model.getTimeInQueue().size();
34  }
35
36  double n(vector<double>& v, double T) {
37          float n = 0;
38          for (int i = 0; i < v.size(); n += i * v[i] / T, ++i);
39
40          return n;
41  }
42
43  int main() {
44          srand(time(0));
45
46          double current_time = 0;
47          int current_count = 0;
48          int total_count = 1000;
49
50          cout << "Model M|M|2\n\n";
51
52          cout << "lambda = "; cin >> lambda;
53          cout << "mu = "; cin >> mu;
54          cout << "\n";
55
56          Model model(mu);
57
58          double time_of_task = 0;
```

```
59          vector<double> countInQueue;
60
61          while ((current_count < total_count) || (model.getTimeInQueue().size() <
            ↪  total_count)) {
62                  if (current_time == time_of_task) {
63                          current_count++;
64
65                          model.addToQueue(time_of_task);
66
67                          time_of_task += generateTask();
68                  }
69
70                  model.analyze(current_time);
71                  for (int i = countInQueue.size(); i < model.getQueueLength() + 1;
                    ↪  countInQueue.push_back(0), ++i);
72
73                  countInQueue[model.getQueueLength()] -= current_time;
74                  current_time = nextMoment(model, time_of_task);
75                  countInQueue[model.getQueueLength()] += current_time;
76          }
77
78          cout << "m = " << m(model) << "\n";
79          cout << "n = " << n(countInQueue, current_time) << "\n";
80
81          system("pause");
82
83          return 0;
84  }
```

1.1.3 Результат

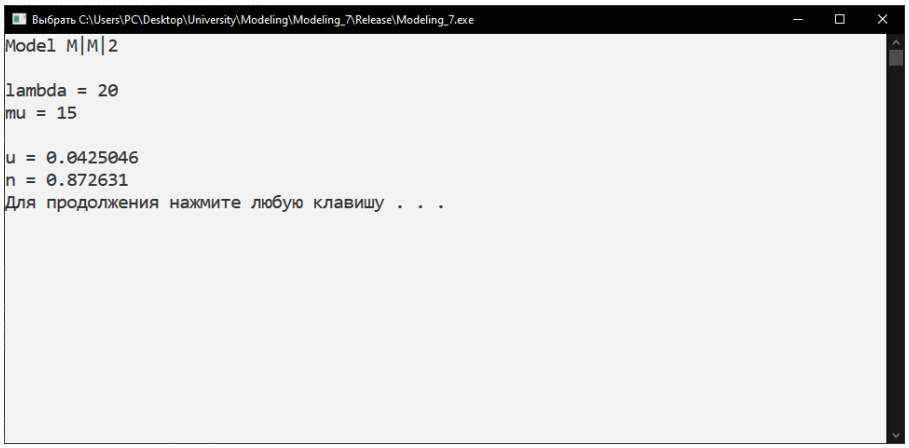

Рисунок 1 – $\lambda = 20, \mu = 15$