Федеральное государственное автономное образовательное учреждение  
высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт математики и фундаментальной информатики

Базовая кафедра вычислительных и информационных технологий

УТВЕРЖДАЮ Заведующий кафедрой

\_\_\_\_\_ В.В. Шайдуров

подпись

« \_\_\_\_\_» \_\_\_\_\_\_\_ 2023г.

**ОТЧЕТ ОБ УЧЕБНОЙ ПРАКТИКЕ**

Институт математики и фундаментальной информатики СФУ

Моделирование очереди в магазине

Вариант №16

Руководитель \_\_\_\_\_\_\_\_\_\_ канд. физ-мат. наук Е.В. Кучунова

подпись, дата

Студент ИМ22-01б \_\_\_\_\_\_\_\_\_\_ Д.В. Семенов

номер группы номер зачетной книжки подпись, дата

**Красноярск 2023**

# **Содержание**

|  |  |
| --- | --- |
| 1. Постановка задачи | 3 |
| 2. Описание программы | 4 |
| 2.1. Среда разработки программы | 4 |
| 2.2. Алгоритм решения задачи | 4 |
| 2.3. Описание основных функций программы | 8 |
| 3. Примеры результатов работы программы | 9 |
| Список использованных источников | 11 |
| Код программы | 12 |

**1. Постановка задачи**

Целью курсовой работы является создание программы на языке объектно-ориентированного программирования С++, моделирующей работу магазина. Магазин работает с 8:00 до 18:30. В магазине работает один продавец. Покупатели приходят в магазин через некоторые промежутки времени (от 5до 30 минут). Если продавец свободен, он тут же начинает обслуживать покупателя. В этом случае время ожидания равна нулю, время начала обслуживания равняется времени прихода в магазин. Время обслуживание одного покупателя колеблется с 5 до 10 минут. Если в момент прихода покупателя, продавец занят, то покупатель становится в очередь. Тогда его время начала обслуживания равно времени окончания обслуживания предыдущего покупателя. Если покупатель приходит в магазин до закрытия, но не успевает дождаться своей очереди (магазин уже закрылся), то он уходит домой необслуженный. Требуется:

* вывести таблицу покупателей: номер, время прихода, время начала обслуживания, время окончания обслуживания, время нахождения в очереди.
* Провести мониторинг очереди: через равные промежутки времени выводить длину очереди.
* Определить время, когда очередь была максимальной, и сколько человек в ней было.
* Вычислить среднее время ожидания в очереди.
* Найти покупателя, который простоял в очереди максимальное количество минут. Вывести его номер и время ожидания.
* Определить, сколько покупателей было обслужено в магазине и сколько покупателей остались необслуженными.

**2. Описание программы**

**2.1 Среда разработки программы**

Программа реализована в среде разработки Microsoft Visual Studio 2022 на языке объектно-ориентированного программирования С++ с применением библиотек:

“iostream” - объекты, управляющие чтением из стандартных потоков и записью в них

“string” - класс с методами и переменными для организации работы со строками в языке программирования C++. Он включён в стандартную библиотеку C++.

“cmath” - Включает заголовок <math.h> стандартной библиотеки C и добавляет связанные имена в std пространство имен.

“iomanip” - Включите стандартный заголовок <iomanip> объекта iostreams для определения нужного числа манипуляторов, каждый из которых принимает один аргумент.

**2.2 Алгоритм решения задачи**

**2.2.1** **Блок-схема алгоритма**

В данном разделе приводится графическая блок-схема общего алгоритма решения задачи.

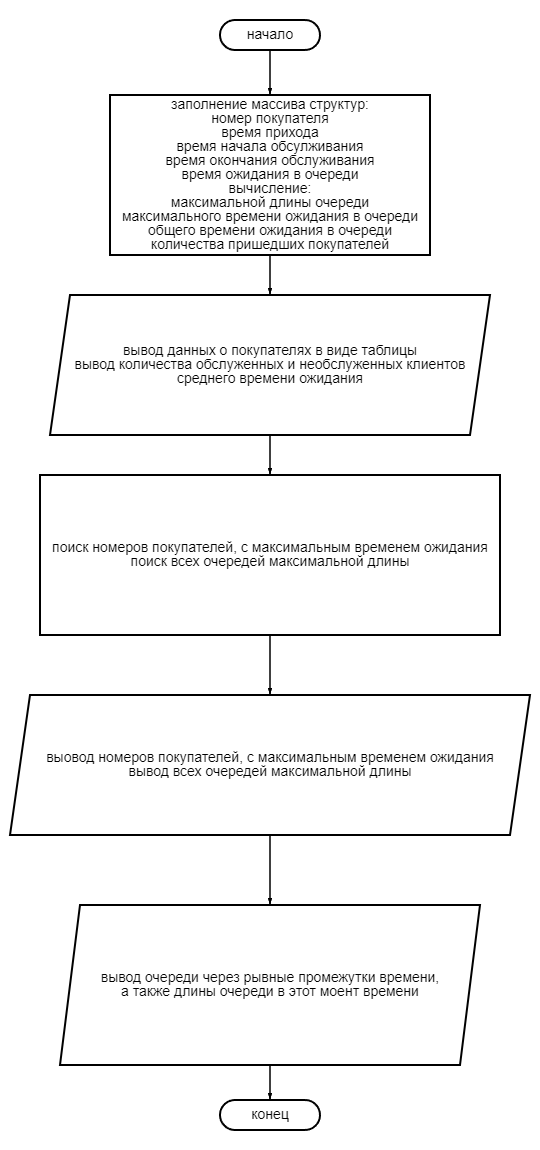


Рисунок 1 – Графическая блок-схема общего алгоритма решения задачи

**2.2.2** **Описание алгоритма решения задачи**

Работа программы начинается с объявления структуры, в которой содержатся данные о покупателе (его номер, время прихода, время начала и окончания обслуживания, время ожидания в очереди). Также объявляется динамический массив структур и глобальная переменная, которая отвечает за количество обслуженных клиентов.

Основной блок программы:

1. Прописывается функция srand().Для того, чтобы в функции srand() получить разные начальные значения используется функция time() с параметром NULL.
2. Инициализируются переменные, выделяется память для массива структур с помощью функции malloc(). Поскольку время прихода каждого покупателя колеблется от 5 до 30 минут, то конечное число покупателей неизвестно. Поэтому изначально выделяется память для максимального числа клиентов, которые могут прийти. Магазин работает 10 часов (600 минут), минимальный интервал прихода клиентов 5 минут, следовательно, максимальное число покупателей = 600/5 = 126.
3. Инициализируем данные для первого покупателя. Время прихода и обслуживания генерируются случайно с помощью функции rand().
4. С помощью цикла while заполняем данные о покупателях в массив:
   1. время прихода покупателя генерируется случайно в промежутке от 5 до 30 минут, с помощью функции rand().
   2. Номер покупателя.
   3. Переменная kol1(изначально равна максимальному количеству возможных покупателей, 126) уменьшается на 1, эта переменная понадобится для подсчета количества покупателей.
   4. Ставится условие, если время прихода покупателя меньше чем время окончания обслуживания предыдущего, то он встает в очередь. Время начала его обслуживания равно времени окончания обслуживания предыдущего, а время ожидания в очереди равно разнице между временем его прихода и началом времени его обслуживания.

* Если клиент встает в очередь, то переменная отвечающая за длину очереди увеличивается и запускается проверка: если время прихода покупателя меньше чем время окончания обслуживания покупателя, который стоит через одного перед ним, то переменная, отвечающая за длину очереди увеличивается на 1, затем сравниваем с временем окончания обслуживания покупателя, который пришел еще раньше. Когда длина очереди будет получена, проверяем ее на максимальность.
  1. Если условие не выполняется, то время начала обслуживания равняется времени прихода, время в очереди равно 0, а длина очереди обнуляется.
  2. время окончания обслуживания равно сумме времени начала обсл. и случаному числу минут от 5 до 10 (время обслуживания).
  3. Проверяем время ожидания на максимальное.
  4. Сумируем время ожидания с общим временем ожидания.

Цикл завершается.

1. Инициализируются переменные для нового цикла. Вычисляется количество обслуженных клиентов (от максимального числа клиентов отнимается значение переменной kol1). Освобождаем ненужную память с помощью функции realloc.
2. Выводим данные о покупателях в виде таблицы с помощью цикла for{}, а также дополнительный столбец, который показывает обслужен клиент или нет.
3. Выводим количество обслуженных и необслуженных клиентов (значения подсчитаны в цикле выше).
4. Вычисляем и выводим среднее время ожидания.
5. Инициализируем переменные
6. Запускается новый цикл
   1. Запоминаем номера покупателей, которые стояли в очереди дольше всего.
   2. С помощью условий и еще одного цикла while проверяем длину очереди (если она есть).
   3. Если очередь имеет максимальную длину: выводим время которое очередь существовала (с момента прихода первого покупателя в очереди, до момента начала обслуживания последнего).

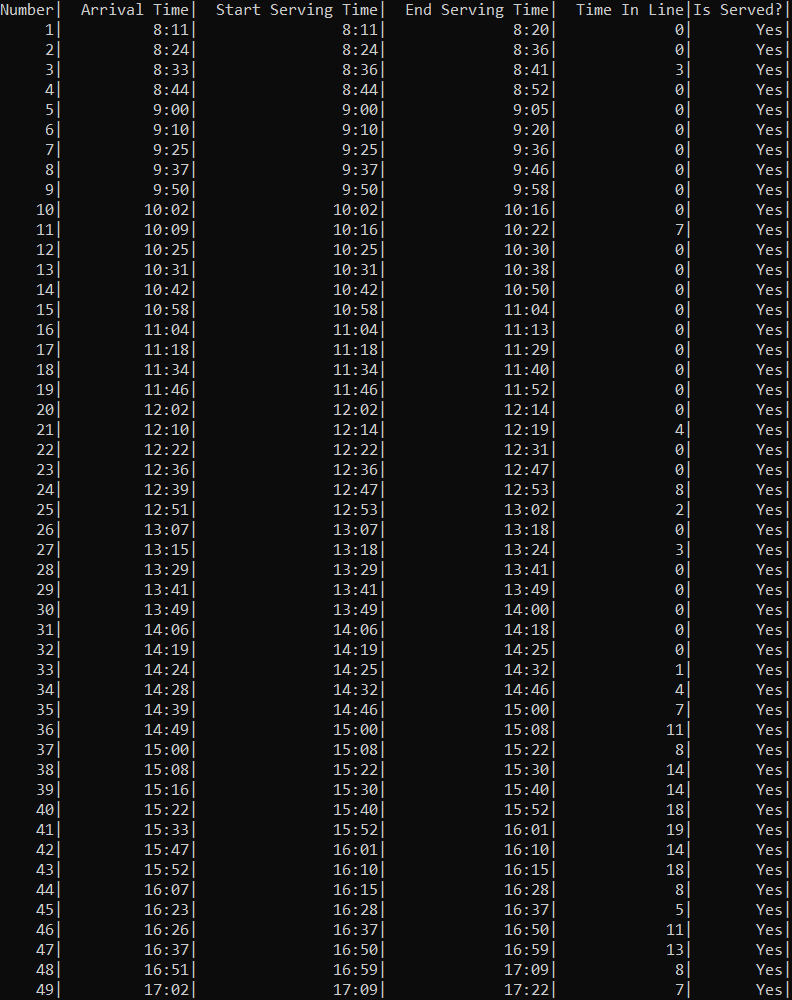
Цикл завершается.

1. Выводим номера клиентов, с самым долгим ожиданием
2. Запускается новый цикл for, в котором с помощью функции clientCounInLine() определяем количество людей в очереди в заданный момент времени. Выводим время, в которое проверяется и длину очереди в этот момент.
3. Освобождаем память, выделенную под массив.

**2.2.3** **Описание основных функций программы**

clientCountInLine() – функция с помощью цикла for проверяет находится ли клиент в очереди в данный момент, и возвращает длину очереди.

**3. Примеры результатов работы программы**

****

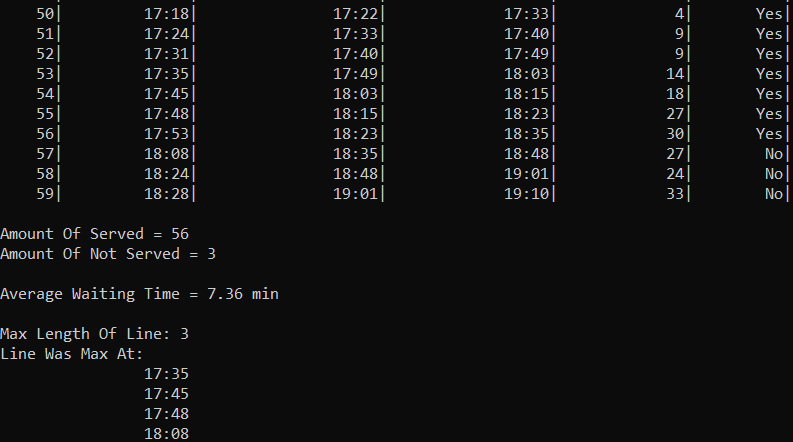
****

Рисунок 1 – таблица с данными клиентов, количество обслуженных и необслуженных клиентов, время когда очередь была максимальна.

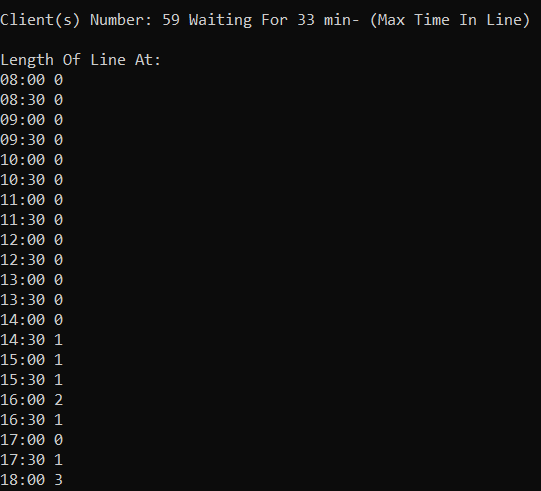


Рисунок 2 – номера клиентов, которые дольше всех стояли в очереди, мониторинг очереди через равные промежутки времени.

**Список использованных источников**

1. Баранов С.Н. Программирование на языке С++: учеб. пособие / С.Н. Баранов, И.В. Баранова. – Красноярск: ИПК СФУ, 2010. – 112 с.
2. Глушаков, С. В. Язык программирования С++ / С. В. Глушаков, А. В. Коваль, С. В. Смирнов. – М.: АСТ, 2004. – 500 с.
3. Лафоре Р. Объектно-ориентированное программирование в С++ / Р. Лафоре. – Спб.: Питер, 2004. – 923 с.
4. Павловская, Т. А. C/C++. Программирование на языке высокого уровня / Т. А. Павловская. – СПб.: Питер, 2010. – 461 с.
5. URL: <https://learn.microsoft.com/ru-ru/>

Код программы:

#include <iostream>

#include <cmath>

#include <string>

#include <iomanip>

using namespace std;

struct shop //Объявляем структуру

{

int number;

int arrivalTime;

int strtSrvngTime;

int endSrvngTime;

int lineTime;

};

shop\* massive;//Динамический массив структур

int kolobs;

int clientCountInLine(int tm) {//Функция подсчета очереди в данный момент времени

int res = 0;

for (int i = 0; i < kolobs; i++) {

if (massive[i].arrivalTime <= tm and tm < massive[i].strtSrvngTime)

res++;

}

return res;

}

int main()//основная программа

{

srand(time(NULL));

//инициализация переменных и выделение памяти для массива

int n = 480 + 3 + rand() % 15, nomer = 0, vremyaPrihoda = 0, i = 1, kol = 126, kol1 = 126, ochered = 0;

int sredvremyaofwaiting = 0, maxlinetime = 0, maxochered = 0, kolnotobs = 0, kolobsl = 0;

massive = (shop\*)malloc(kol \* sizeof(shop));

//инициализация данных для первого клиента

massive[0].number = 1;

massive[0].arrivalTime = n;

massive[0].strtSrvngTime = massive[0].arrivalTime;

massive[0].endSrvngTime = massive[0].strtSrvngTime + 5 + rand() % 10;

massive[0].lineTime = 0;

int pokaz = 1, y = 1;

//цикл заполнения массива

while (n < 1110) {

vremyaPrihoda = 3 + rand() % 15;

n += vremyaPrihoda;

kol1 -= 1;

massive[i].number = i + 1;

massive[i].arrivalTime = n;

if (massive[i].arrivalTime < massive[i - 1].endSrvngTime) {

massive[i].strtSrvngTime = massive[i - 1].endSrvngTime;

massive[i].lineTime = massive[i - 1].endSrvngTime - massive[i].arrivalTime;

//нахождение максимальной очереди

if (massive[i].arrivalTime < massive[i - y].endSrvngTime) {

y++;

if (y > 0) {

while (pokaz == 1) {

if (massive[i].arrivalTime < massive[i - y].endSrvngTime)

y++;

else

pokaz = 0;

}

}

//cout << y << endl;

ochered = y - 1;

maxochered = max(ochered, maxochered);

y = 1;

pokaz = 1;

}

}

else {

massive[i].strtSrvngTime = massive[i].arrivalTime;

massive[i].lineTime = 0;

ochered = 0;

}

massive[i].endSrvngTime = massive[i].strtSrvngTime + 5 + rand() % 10;

sredvremyaofwaiting += massive[i].lineTime;

maxlinetime = max(maxlinetime, massive[i].lineTime);

i++;

}

int s = 1, ochrd = 0;

kolobs = kol - kol1;

//освобождение лишней памяти

massive = (shop\*)realloc(massive, kolobs \* sizeof(shop));

//вывод таблицы

printf("%s\n", "Number| Arrival Time| Start Serving Time| End Serving Time| Time In Line|Is Served?|");

for (int h = 0; h < kol - kol1; h++) {

cout << setw(6) << massive[h].number << "|";

if (massive[h].arrivalTime % 60 < 10) {

cout << setw(11) << massive[h].arrivalTime / 60 << ":" << "0" + to\_string(massive[h].arrivalTime % 60) << "|";

}

else {

cout << setw(11) << massive[h].arrivalTime / 60 << ":" << massive[h].arrivalTime % 60 << "|";

}

if (massive[h].strtSrvngTime % 60 < 10) {

cout << setw(17) << massive[h].strtSrvngTime / 60 << ":" << "0" + to\_string(massive[h].strtSrvngTime % 60) << "|";

}

else {

cout << setw(17) << massive[h].strtSrvngTime / 60 << ":" << massive[h].strtSrvngTime % 60 << "|";

}

if (massive[h].endSrvngTime % 60 < 10) {

cout << setw(15) << massive[h].endSrvngTime / 60 << ":" << "0" + to\_string(massive[h].endSrvngTime % 60) << "|";

}

else {

cout << setw(15) << massive[h].endSrvngTime / 60 << ":" << massive[h].endSrvngTime % 60 << "|";

}

cout << setw(14) << massive[h].lineTime << "|";

if (massive[h].strtSrvngTime > 1110) {

cout << setw(10) << "No" << "|" << endl;

kolnotobs++;

}

else {

cout << setw(10) << "Yes" << "|" << endl;

kolobsl++;

}

}

//вывод требуемых значений

cout << endl;

cout << "Amount Of Served = " << kolobsl << endl;

cout << "Amount Of Not Served = " << kolnotobs << endl;

cout << endl;

if (sredvremyaofwaiting == 0)

cout << "Average Waiting Time = " << 0 << endl;

else

cout << "Average Waiting Time = " << round(double(sredvremyaofwaiting) / kolobs \* 100) / 100 << " min" << endl;

cout << endl;

printf("Max Length Of Line: %d\n", maxochered);

cout << "Line Was Max At:" << endl;

//цикл для вывода времени,когда очередь была максимальна

int l = 1, pokazatel = 1;

string nomera;

while (s < kolobs) {

if (massive[s].lineTime == maxlinetime)

nomera += to\_string(massive[s].number) + ",";

if (massive[s].arrivalTime < massive[s - l].endSrvngTime) {

l++;

if (l > 0) {

while (pokazatel == 1) {

if (massive[s].arrivalTime < massive[s - l].endSrvngTime)

l++;

else

pokazatel = 0;

}

}

ochrd = l - 1;

l = 1;

pokazatel = 1;

if (ochrd == maxochered) {

printf(" %02d:%02d\n", massive[s - maxochered + 1].arrivalTime / 60, massive[s - maxochered + 1].arrivalTime % 60);

}

}

s++;

}

if (maxochered == 0)

cout << setw(23) << 0 << endl;

cout << endl;

//номера клиентов, с максимальным временем ожидания

if (maxlinetime == 0)

cout << "Noone was waitining" << endl;

else {

nomera.pop\_back();

cout << "Client(s) Number: " << nomera << " Waiting For " << maxlinetime << " min- (Max Time In Line)" << endl;

}

cout << endl;

//мониторинг очереди

int addingtime = 30;

cout << "Length Of Line At: " << endl;

for (int currentTime = 480; currentTime < 1110; currentTime += addingtime) {

printf("%02d:%02d %d\n", currentTime / 60, currentTime % 60, clientCountInLine(currentTime));

}

//осовбождение памяти массива

free(massive);

return 0;

}