**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

Федеральное государственное автономное образовательное учреждение высшего профессионального образования

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Инженерная школа информационных технологий и робототехники

Направление 15.03.04 «Автоматизация технологических процессов и производств»

Отделение автоматизации и робототехники

Работа с очередью в C++

Наименование лабораторной работы

Отчет по лабораторной работе № 5

по дисциплине «Основы программирования и алгоритмизации»

Наименование учебной дисциплины

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Выполнил студент гр. 8Т32 |  | C:\Users\pavlu\OneDrive\Рабочий стол\Подпись.jpg |  |  |  | П. П. Гаврильев |
|  |  | Подпись |  | Дата |  | И.О. Фамилия |
|  |  |  |  |  |  |  |
| Проверил ассистент ОАР |  |  |  |  |  | А. В. Павловский |
| Должность |  | Подпись |  | Дата |  | И.О. Фамилия |

Томск – 2024 г.

**Цель работы:**

Ознакомиться и реализовать решение для очереди в магазине на практике на языке программирования С++.

**Ход работы:**

Таблица 1 – Формулировка задания

|  |
| --- |
| Очередь в магазине В магазине есть три кассы, на которых могут одновременно работать от 1 до 3 кассиров. Люди приходят и становятся в очередь. При появлении нового кассира, очередь должна перераспределиться. Реализовать структуру данных, представляющую очередь на кассы. Алгоритм должен выполнять операции по добавлению покупателей в очередь (добавление осуществляется естественным образом – где меньше людей, туда и становимся), по уходу расплатившихся покупателей (считать, что человек у кассы обслуживается за одну итерацию работы алгоритма), по перераспределению очереди, в случае появления еще одного/двух кассиров. |

Задание.

В данном задании поставлена задача реализовать очередь в магазине. Для реализации этой задачи были спроектированы блок схемы, представленные на рисунках 1.1, 1.2 и 1.3. По данным блок схемам были написаны программа, которая представлены в листингах 1.1 и 1.2. Результат выполнения программы представлен на рисунке 1.4.

Программа создает очередь и потом заполняет эту очередь покупателями. По числу покупателей открываются кассы. После этого покупатели перераспределяются по открывшимся кассам и обслуживаются на них. Для реализации поставленной задачи была реализована функция work,. После вызова функции work на консоль пользователя показывается обслуживание очереди.

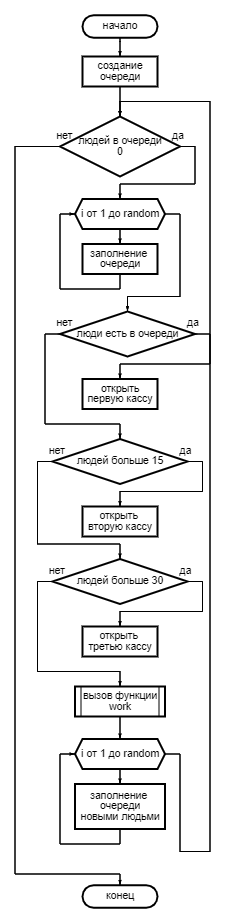


Рисунок 1.1 – Блок схема программы

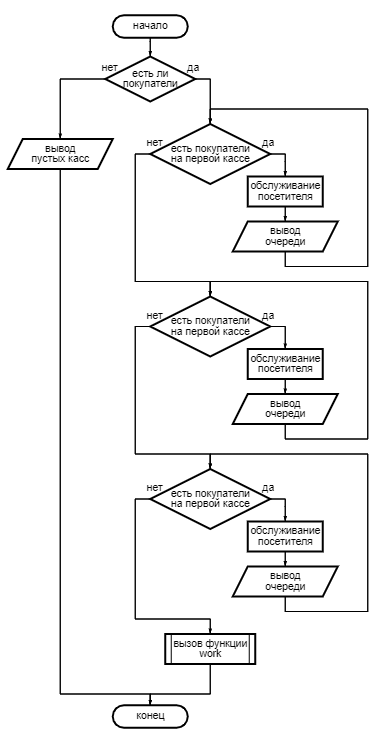


Рисунок 1.2 – Блок схема функции work (queue1, queue2, queue3, k1, k2, k3)

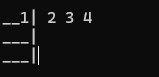


Рисунок 1.4 – Итог запуска программы

Листинг 1.1 – Код программы

|  |
| --- |
| #include <iostream>  #include <string>  #include <random>  #include <queue>  #include "work.h"  using namespace std;  int main()  {  bool k1 = false;  bool k2 = false;  bool k3 = false;  int i = 1;  setlocale(LC\_ALL, "ru");  int new\_pokup = 0;  std::random\_device rd;//получение началоьного значения от аппаратного обеспечения  std::mt19937 gen(rd());//создайте объект класса engine  std::uniform\_int\_distribution<> digitdist(1, 50);//диапазон для чисел  std::queue<int> queue; // пустая очередь  std::queue<int> queue1;//очередь кассы 1  std::queue<int> queue2;//очередь кассы 2  std::queue<int> queue3;//очередь кассы 3  for (i; i <=digitdist(gen); i++) {  queue.push(i);  }  while (queue.size() != 0) {  std::cout << "\x1B[2J\x1B[H";  if (queue.size() > 0 and queue.size() < 16) {  k1 = true;  }  if (queue.size() > 15 and queue.size() < 31) {  k1 = true;  k2 = true;  }  if (queue.size() > 30) {  k1 = true;  k2 = true;  k3 = true;  }  cout << "\_\_\_| ";  if (k1 == true and k2 == false and k3 == false) {  while (queue.size() != 0) {  queue1.push(queue.front());  queue.pop();  }  }  if (k1 == true and k2 == true and k3 == false) {  while (queue.size() != 0) {  queue1.push(queue.front());  queue.pop();  if (queue.size() != 0) {  queue2.push(queue.front());  queue.pop(); |

Продолжение листинга 1.1

|  |
| --- |
| }  }  }  if (k1 == true and k2 == true and k3 == true) {  while (queue.size() != 0) {  queue1.push(queue.front());  queue.pop();  if (queue.size() != 0) {  queue2.push(queue.front());  queue.pop();  }  if (queue.size() != 0) {  queue3.push(queue.front());  queue.pop();  }  }  }  std::queue<int> queue12 = queue1;  std::queue<int> queue22 = queue2;  std::queue<int> queue32 = queue3;  while (queue1.size() != 0) {  cout << queue1.front() << " ";  queue1.pop();  }  cout << "\n\_\_\_| ";  while (queue2.size() != 0 and k2 == true) {  cout << queue2.front() << " ";  queue2.pop();  }  cout << "\n\_\_\_| ";  while (queue3.size() != 0 and k3 == true) {  cout << queue3.front() << " ";  queue3.pop();  }  work(queue12, queue22, queue32, k1, k2, k3);  for ( i; i <= digitdist(gen); i++) {  queue.push(i);  }  }  } |

Листинг 1.3 - Библиотека work (queue1, queue2, queue3, k1, k2, k3)

|  |
| --- |
| #pragma once  #include <queue>  #include <iostream>  #include <chrono>  #include <thread>  using namespace std;  void work(std::queue<int> queue1, std::queue<int> queue2, std::queue<int> queue3, bool k1, bool k2, bool k3)  {  std::this\_thread::sleep\_for(std::chrono::seconds(1));  std::cout << "\x1B[2J\x1B[H";  std::queue<int> queue12=queue1;  std::queue<int> queue22=queue2;  std::queue<int> queue32=queue3;  if ((queue1.size()+ queue2.size() + queue3.size() )!=0 )  {  cout << "\_\_" << queue12.front() << "| ";  queue12.pop();  queue1.pop();  while (queue1.size() != 0 and k1==true) {  cout << queue1.front() << " "; |

Продолжение листинга 1.2

|  |
| --- |
| queue1.pop();    }  if (queue22.size()!=0){  cout << "\n\_\_" << queue22.front() << "| ";  queue22.pop();  queue2.pop();  while (queue2.size() != 0 and k2 == true) {  cout << queue2.front() << " ";  queue2.pop();  }  }  else { cout << "\n\_\_\_|"; }  if (queue32.size() != 0) {  cout << "\n\_\_" << queue32.front() << "| ";  queue32.pop();  queue3.pop();  while (queue3.size() != 0 and k3 == true) {  cout << queue3.front() << " ";  queue3.pop();  }  }  else { cout << "\n\_\_\_|"; }  work(queue12, queue22, queue32, k1, k2, k3);  }  else {  cout << "\_\_\_|";  cout << "\n\_\_\_|";  cout << "\n\_\_\_|";  }  } |

**Вывод**

В ходе лабораторной работы была изучены информация по работе с очередью в магазине. На основе этой информации была проделана работа по написанию кода, представленных на листингах (1.1, 1.2), для заданий. Также были сделаны блок схемы, которые показаны на рисунках (1.1, 1.2). В итоге благодаря этой информации можно реализовать различные задания с очередью в различных структурах или предприятих.