**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.

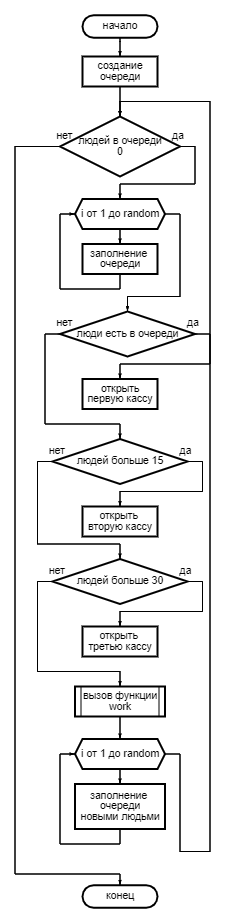


Рисунок 1.1 – Блок схема программы

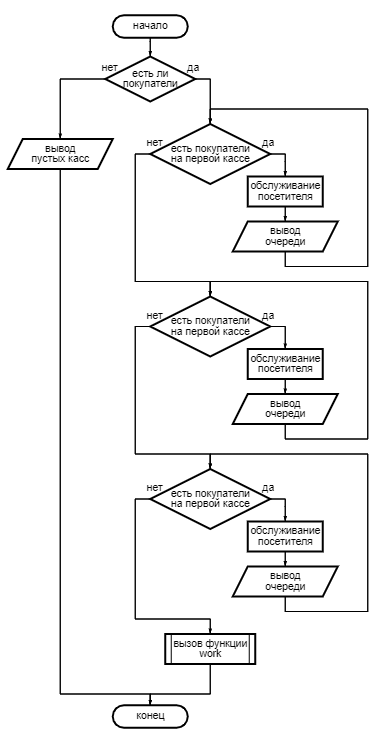


Рисунок 1.2 – Блок схема функции work (mass1, mass2, mass3, size1, size2, size3)

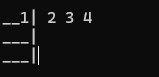


Рисунок 1.4 – Итог запуска программы

Листинг 1.1 – Код программы

|  |
| --- |
| #include <iostream>  #include <string>  #include <random>  #include "work.h"  using namespace std;  int main()  {  int front=0, rear=0;  bool k1 = false;  bool k2 = false;  bool k3 = false;  int size1 = 0;  int size2 = 0;  int size3 = 0;  setlocale(LC\_ALL, "ru");  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  int size = digitdist(gen);  rear = size;  if (size > 0 and size < 16) {  k1 = true;  size1 = size;  size2 = 0;  size3 = 0;  }  if (size > 15 and size < 31) {  k1 = true;  k2 = true;  size2 = size/2;  if (size%2==0)  {  size1 = size / 2;  }  else {  size1 = size / 2+1;  }  size3 = 0;  }  if (size > 30) {  k1 = true;  k2 = true;  k3 = true;  size2 = size / 3;  size3 = size / 3;  if (size % 3 != 0)  {  size1 = size / 3+1;  }  else {  size1 = size / 3;  }  } |

Продолжение листинга 1.1

|  |
| --- |
| int\* mass = new int[size];  int\* mass1 = new int[size1];  int\* mass2 = new int[size2];  int\* mass3 = new int[size3];  for (int i=0; i <size; i++) {  mass[i] = i + 1;  }  while (size!=0) {  cout << "\x1B[2J\x1B[H";    cout << "\_\_\_|\n\_\_\_|\n\_\_\_|";  if (k1 == true and k2 == false and k3 == false) {  for (int j = 0; j < size; j++) {  mass1[j] = mass[j];  }  }  if (k1 == true and k2 == true and k3 == false) {  int l = 0;  for (int j = 0; j < size; j++) {  mass1[j] = mass[l];  l++;  if (l == size) {  break;  }  mass2[j] = mass[l];  l++;  if (l == size) {  break;  }  }  }  if (k1 == true and k2 == true and k3 == true) {  int l = 0;  for (int j = 0; j < size; j++) {  mass1[j] = mass[l];  l++;  if (l == size) {  break;  }  mass2[j] = mass[l];  l++;  if (l == size) {  break;  }  mass3[j] = mass[l];  l++;  if (l == size) {  break;  }  }  }  work(mass1, mass2, mass3, size1,size2, size3);  size = 0;  }  delete[] mass;  delete[] mass1;  delete[] mass2;  delete[] mass3;  } |

Листинг 1.3 - Библиотека work (mass1, mass2, mass3, size1, size2, size3)

|  |
| --- |
| #pragma once  #include <queue>  #include <iostream>  #include <chrono>  #include <thread>  using namespace std;  #include "work.h"  void work(int\* mass1, int\* mass2, int\* mass3, int size1, int size2, int size3 )  {  std::this\_thread::sleep\_for(std::chrono::seconds(1));  std::cout << "\x1B[2J\x1B[H";  if ((size1+size2+size3)!=0 )  {  cout << "\_\_" << mass1[0] << "| ";  for (int j = 1; j < size1; j++) {  cout << mass1[j]<<" ";  }  if (size2!=0){  cout << "\n\_\_" << mass2[0] << "| ";  for (int j = 1; j < size2; j++) {  cout << mass2[j] << " ";  }  }  else { cout << "\n\_\_\_|"; }  if (size3 != 0) {  cout << "\n\_\_" << mass3[0] << "| ";  for (int j = 1; j < size3; j++) {  cout << mass3[j] << " ";  }  }  else { cout << "\n\_\_\_|"; }  int size11 = 0;  int size21 = 0;  int size31 = 0;  if (size1!=0)  {  size11 = size1-1;  }  if (size2 != 0)  {  size21 = size2 - 1;  }  if (size3 != 0)  {  size31 = size3 - 1;  }  int\* mass11 = new int[size11];  int\* mass21 = new int[size21];  int\* mass31 = new int[size31];    for (int j = 0; j < size11; j++) {  mass11[j] = mass1[j + 1];  }  for (int j = 0; j < size21; j++) {  mass21[j] = mass2[j + 1];  }  for (int j = 0; j < size31; j++) {  mass31[j] = mass3[j + 1];  }  work(mass11, mass21, mass31, size11, size21, size31);  delete[] mass11;  delete[] mass21;  delete[] mass31;  }  else { |

Продолжение листинга 1.2

|  |
| --- |
| 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.