Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«Пермский национальный исследовательский**

**политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**О Т Ч Ё Т**

**по лабораторной работе №11**

Дисциплина: основы алгоритмизации и программирования

Тема: “ Динамические структуры данных.

Очереди”

Вариант 16

Выполнил:

студент группы ИВТ-20-2Б Тедеев А.З.

Проверила: доцент кафедры ИТАС

Полякова О.А.

Пермь, 2021

**Цель задачи**

Получить практические навыки работы с очередями.

**Постановка задачи**

Тип информационного поля int. Удалить из очереди первый элемент с четным информационным полем.

2

**Анализ задачи**

**1.** Для решения задачи необходимо:

**1.1.** Организовать структуру Queue, поля которой будут использоваться для ввода данных типа int, для хранения адреса следующего элемента списка.

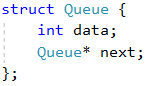
**1.2.** Организовать функцию Make() для формирования очереди.

**1.3.** Организовать функцию Deletion() для удаления элемента.

**1.4.** Организовать функцию GetLine() для вывода очереди на консоль.

**2.** В ходе работы были использованы следующие типы данных:

**2.1.** Структура Queue для организации очереди.



**2.2.** Функция Make(), в качестве параметров передаётся размер очереди size, указатель на ссылку на первый элемент очереди, указатель на ссылку на последний элемент очереди last. В переменную x записать первый элемент очереди и записать его в очереди через метод next.

Queue\* Make(int size, Queue\*& first, Queue\*& last){

...

Queue\* p = new Queue;

...

int x;

...

Queue\* nex = new Queue;

...

}

**2.3.** Функция Deletion(), в качестве параметров передаётся указатель на ссылку на первый элемент очереди – first, указатель на ссылку на последний элемент – last. Bool flag – флажок, Queue\* prev.

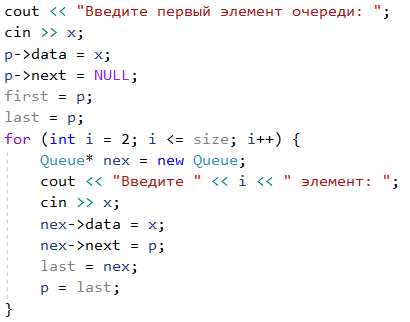
3

**3.** Для решения задачи данные были представлены в следующем виде:

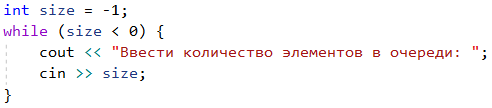
**3.1.** Данные вводятся через консоль в виде очереди через структуру Queue с полем типа int и указателем на следующий элемент списка.

**4.** Для операций ввода и вывода использовались следующие операторы и функции:

**4.1.** Ввод данных на консоль реализован функцией cin для первого элемента очереди и в цикле for (также через cin) для следующих.



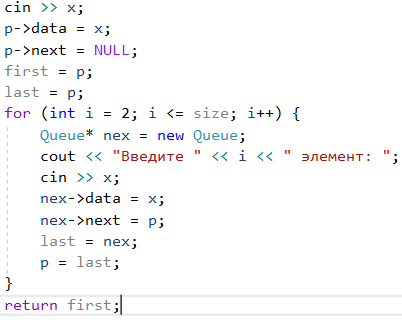
**4.2.** В функции main() вводится количество элементов в списке в переменной size через цикл while, чтобы пользователь не ввёл отрицательное число.



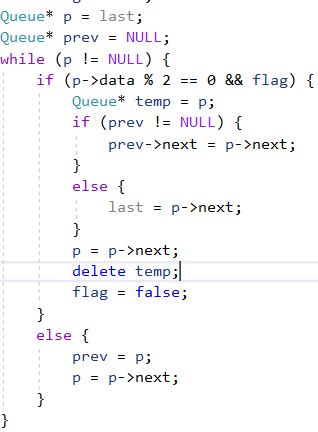
**5.** Поставленные задачи будут решены следующими действиями:

**5.1.** В функции make() осуществляется ввод всех элементов в очереди. Первый вводится вне цикла for и используется в качестве указателя на очередь. В цикле for заполняются элементы от 2 до size: резервируется память в переменной nex, в которую мы вводим следующий элемент после первого.

4

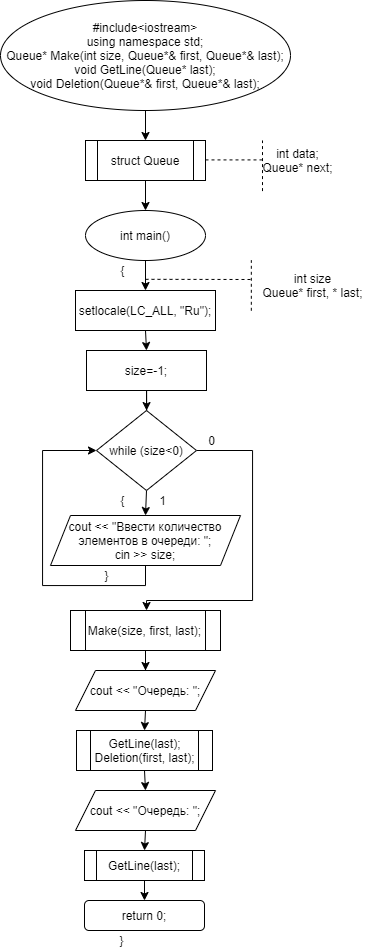


**5.2.** В функции Deletion() организовать удаление первого чётного элемента через указатель на последний элемент в очереди, так как вывод на консоль происходит с конца очереди. Цикл while действует пока в указатель не указывает на NULL, то есть пока элементы присутствуют в очереди. В ветвлении присутствует флажок, который меняет своё значение при нахождении чётного элемента. Внутри данного ветвления есть ещё одно ветвление для случая, когда чётный элемент оказывается без наличия предыдущего (то есть самый первый при проверке, если он чётный).

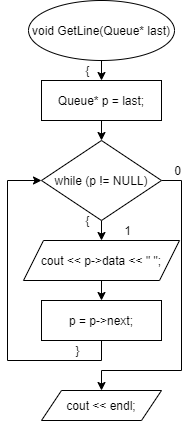


5

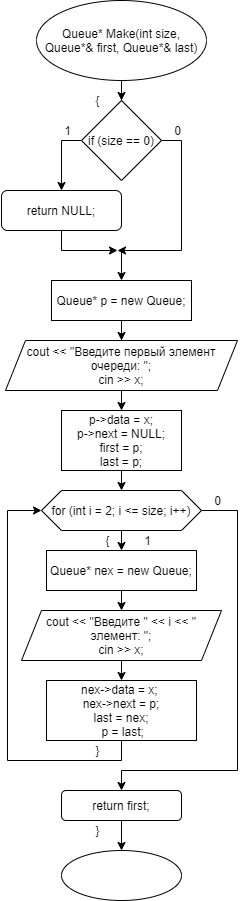
**Блок-схема**



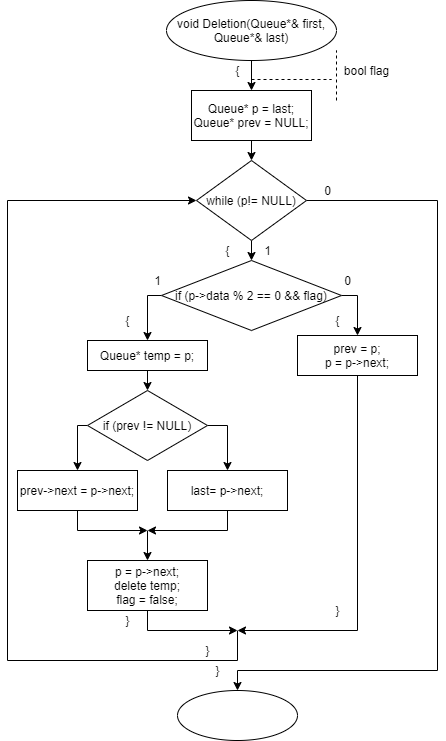
6



7



8



9

**Код**

**Реализация через структуру**

#include<iostream>

using namespace std;

struct Queue {

int data;

Queue\* next;

};

Queue\* Make(int size, Queue\*& first, Queue\*& last);

void GetLine(Queue\* last);

void Deletion(Queue\*& first, Queue\*& last);

int main()

{

setlocale(LC\_ALL, "Ru");

int size = -1;

while (size < 0) {

cout << "Ввести количество элементов в очереди: ";

cin >> size;

}

Queue\* first, \* last;

Make(size, first, last);

cout << "Очередь: ";

GetLine(last);

Deletion(first, last);

GetLine(last);

}

Queue\* Make(int size, Queue\*& first, Queue\*& last)

{

if (size == 0) return NULL;

Queue\* p = new Queue;

int x;

cout << "Введите первый элемент очереди: ";

cin >> x;

p->data = x;

p->next = NULL;

first = p;

last = p;

for (int i = 2; i <= size; i++) {

Queue\* nex = new Queue;

cout << "Введите " << i << " элемент: ";

cin >> x;

nex->data = x;

nex->next = p;

last = nex;

p = last;

}

return first;

}

void Deletion(Queue\*& first, Queue\*& last)

{

bool flag = true;

Queue\* p = last;

Queue\* prev = NULL;

while (p != NULL) {

if (p->data % 2 == 0 && flag) {

Queue\* temp = p;

if (prev != NULL) {

prev->next = p->next;

}

10

else {

last = p->next;

}

p = p->next;

delete temp;

flag = false;

}

else {

prev = p;

p = p->next;

}

}

}

void GetLine(Queue\* last)

{

Queue\* p = last;

while (p != NULL) {

cout << p->data << " ";

p = p->next;

}

cout << endl;

}

11

**Реализация через STL библиотеки**

#include<iostream>

#include<queue>

#include<vector>

using namespace std;

void GetLine(queue<int>line);

void Deletion(queue<int>&line);

int main()

{

setlocale(LC\_ALL, "ru");

int size = 0;

while (size < 1) {

cout << "Введите размерность очереди: ";

cin >> size;

}

vector<int> arr(size);

queue<int> line;

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

cout << "Введите " << i << " элемент: ";

cin >> arr[i];

line.push(arr[i]);

}

cout << "Введённая очередь - ";

GetLine(line);

Deletion(line);

cout << "Изменённая очередь - ";

GetLine(line);

}

void GetLine(queue<int>line)

{

int size = line.size();

for (int i = 1; i <= size; i++) {

cout << line.front() << " ";

line.pop();

}

cout << endl;

}

void Deletion(queue<int>&line)

{

bool flag = true;

int j = 0;

int size = line.size();

vector<int> tmp(size);

while (j < size) {

if (line.front() % 2 == 0&&flag) {

line.pop();

size--;

flag = false;

}

else {

tmp[j] = line.front();

line.pop();

j++;

}

}

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

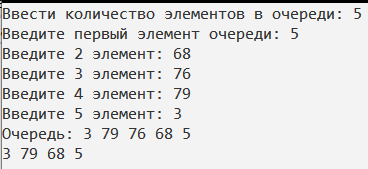
line.push(tmp[i]);

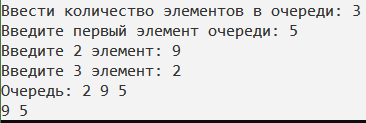
}

}

12

**Скриншоты**





13