Пермский Национальный Исследовательский Политехнический Университет

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

Лабораторная работа № 11.2

по дисциплине

«Основы алгоритмизации и программирования»

Тема: «Динамические структуры данных. Очередь»

Вариант № 17

Выполнил:

студент группы РИС-20-2б

Пономарев Егор Витальевич

Проверила:

доцент кафедры ИТАС

Полякова О. А.

Пермь, 2021

**Цель работы**

Получить практические навыки работы с очередями.

**Постановка задачи**

* Сформировать очередь с помощью структуры и библиотеки STL.
* Распечатать полученную структуру.
* Выполнить обработку структуры в соответствии с заданием.
* Распечатать полученный результат.
* Удалить соответствующую структуру из памяти.

Задание варианта:

1. Тип информационного поля int.
2. Удалить из списка все элементы с четными информационными полями.

**Анализ задачи**

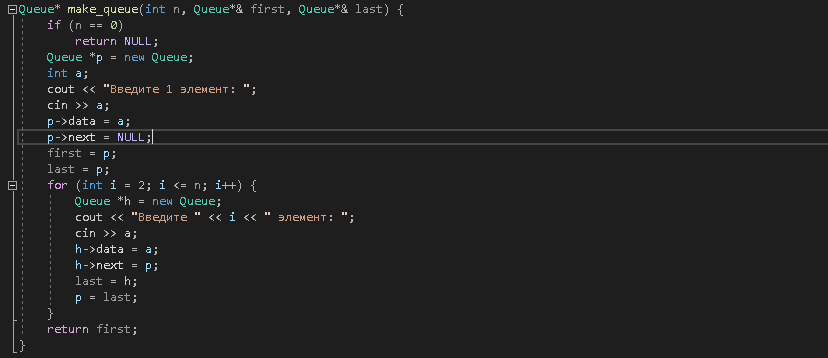
**Реализация через структуру:**

Для решения задачи необходимо:

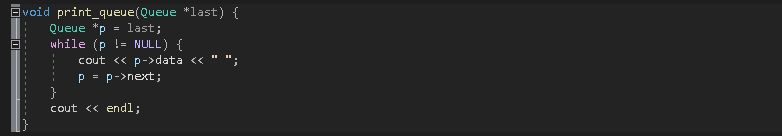
1. Объявляем структуру Queue с информационным полем int data и указателем на следующий элемент Queue\* next.



1. Для создания очереди создадим указатели на начало и конец очереди Queue\* first, \*last. А также выделим динамическую память для первого элемента Queue\* p = new Queue. Теперь first и last должны указывать на p. В цикле добавляем остальные элементы, не забывая работать с указателями.



1. В функцию печати передаем указатель на последний элемент и поочередно печатаем элемент и переходим к следующему, пока он не будет равен NULL.



1. Для возвращения и последующего удаления последнего элемента описываем функцию pop.
2. Добавление элемента в очередь создаем функцию push, которая возвращает указатель на последний элемент после добавления.



**Реализация через STL:**

1. В первую очередь необходимо подключить библиотеку <queue> для работы с очередями.



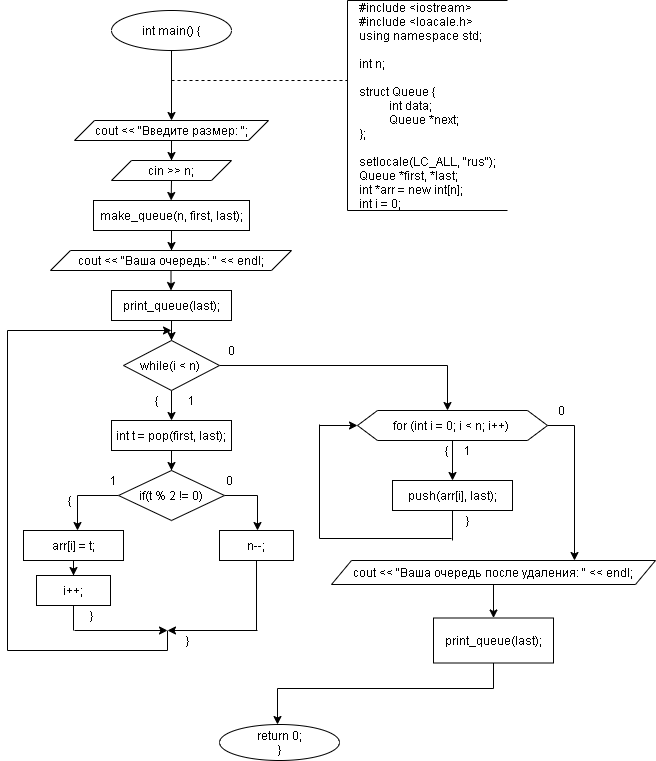
1. Объявляем очередь по шаблону queue <тип данных> <название очереди>;

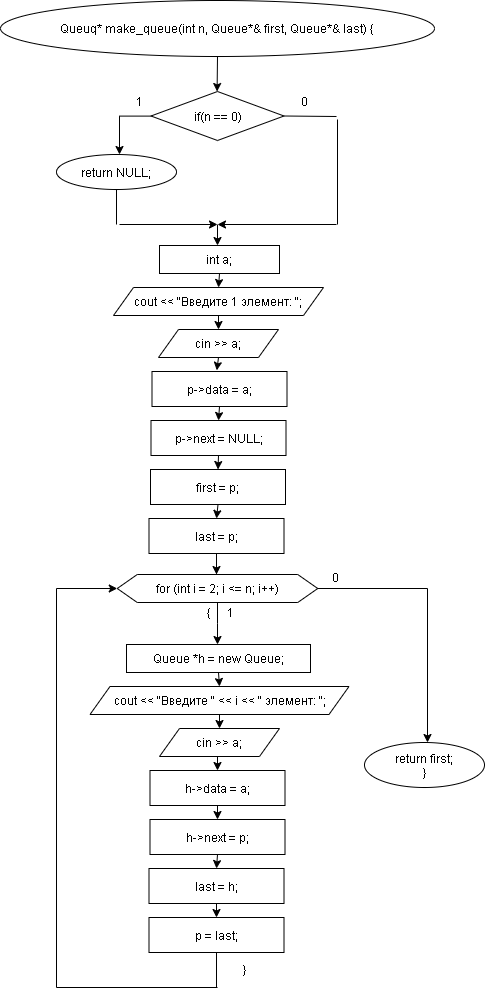


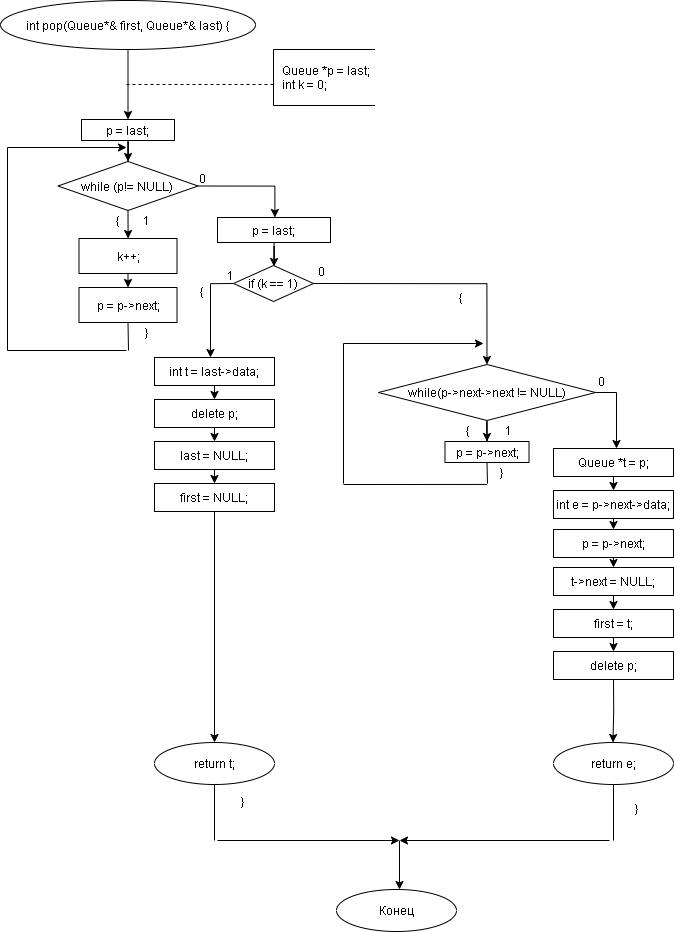
1. Для добавления элементов в очередь воспользуемся методом push ().
2. Размер очереди будем получать с помощью size ().
3. Метод front () возвращает значение последнего элемента, а pop () удаляет его.

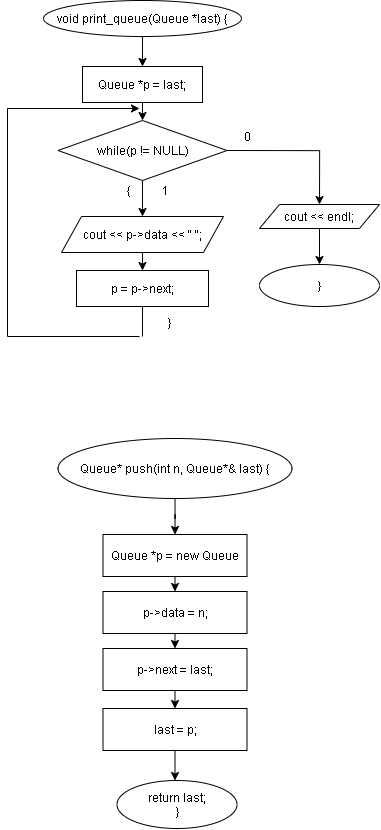
**Блок-схема**

**Структура:**

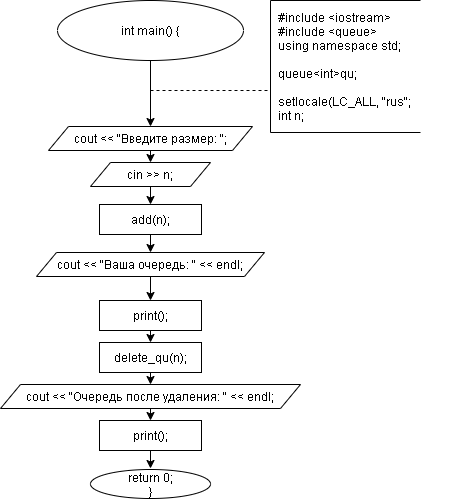
****

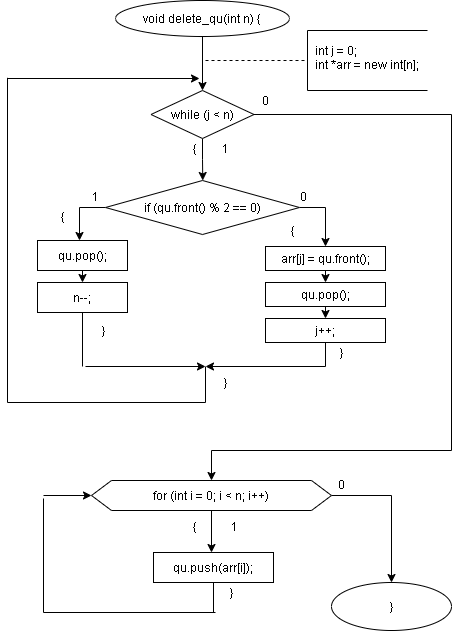
****

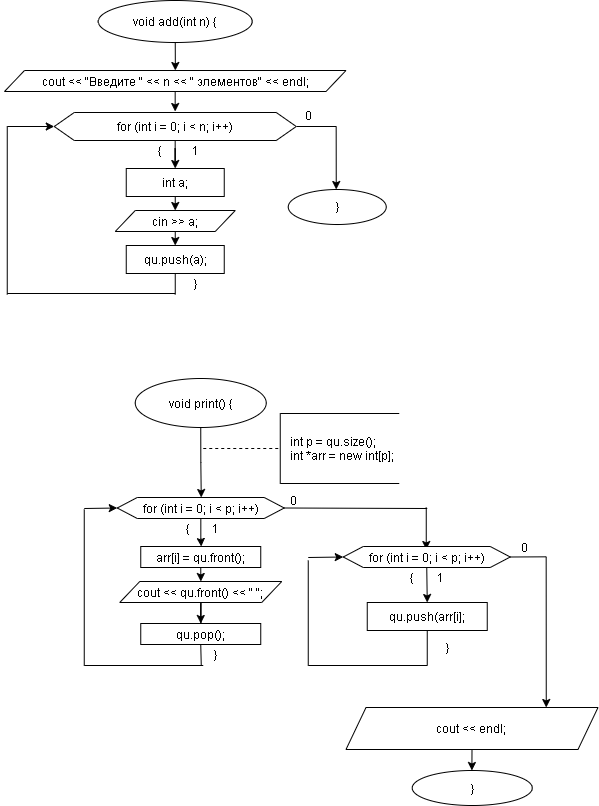
****

****

**STL:**

****

****

****

**Код С++**

**Структура:**

#include <locale.h>

#include <iostream>

using namespace std;

int n;

struct Queue {

int data;

Queue \*next;

};

Queue\* make\_queue(int n, Queue\*& first, Queue\*& last) {

if (n == 0)

return NULL;

Queue \*p = new Queue;

int a;

cout << "Введите 1 элемент: ";

cin >> a;

p->data = a;

p->next = NULL;

first = p;

last = p;

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

Queue \*h = new Queue;

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

cin >> a;

h->data = a;

h->next = p;

last = h;

p = last;

}

return first;

}

void print\_queue(Queue \*last) {

Queue \*p = last;

while (p != NULL) {

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

p = p->next;

}

cout << endl;

}

int pop(Queue\*& first, Queue\*& last) {

Queue \*p = last;

p = last;

int k = 0;

while (p != NULL) {

k++;

p = p->next;

}

p = last;

if (k == 1) {

int t = last->data;

delete p;

last = NULL;

first = NULL;

return t;

}

else {

while (p->next->next != NULL) {

p = p->next;

}

Queue \*t = p;

int e = p->next->data;

p = p->next;

t->next = NULL;

first = t;

delete p;

return e;

}

}

Queue\* push(int n, Queue\*& last) {

Queue \*p = new Queue;

p->data = n;

p->next = last;

last = p;

return last;

}

int main()

{

setlocale(LC\_ALL, "Rus");

cout << "Введите количество элементов: ";

cin >> n;

while (n < 1) {

cout << "Очередь не может быть создана! Введите размер ещё раз: ";

cin >> n;

}

Queue \*first, \*last;

make\_queue(n, first, last);

cout << "Ваша очередь: " << endl;

print\_queue(last);

int \*arr = new int[n];

int i = 0;

while (i < n) {

int t = pop(first, last);

if (t % 2 != 0) {

arr[i] = t;

i++;

}

else {

n--;

}

}

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

push(arr[i], last);

}

cout << "Ваша очередь после удаления: " << endl;

print\_queue(last);

return 0;

}

**STL:**

#include <locale.h>

#include <iostream>

#include <queue>

using namespace std;

queue<int>qu;

void add(int n) {

cout << "Введите " << n << " элементов" << endl;

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

int a;

cin >> a;

qu.push(a);

}

}

void print() {

int p = qu.size();

int \*arr = new int[p];

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

arr[i] = qu.front();

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

qu.pop();

}

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

qu.push(arr[i]);

}

cout << endl;

}

void delete\_qu(int n) {

int j = 0;

int \*arr = new int[n];

while (j < n) {

if (qu.front() % 2 == 0) {

qu.pop();

n--;

}

else {

arr[j] = qu.front();

qu.pop();

j++;

}

}

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

qu.push(arr[i]);

}

}

int main()

{

setlocale(LC\_ALL, "Rus");

int n;

cout << "Введите размер: ";

cin >> n;

while (n <= 0) {

cout << "Очередь не может быть создана! Введите размер ещё раз: ";

cin >> n;

}

add(n);

cout << "Ваша очередь: " << endl;

print();

delete\_qu(n);

cout << "Очередь после удаления: " << endl;

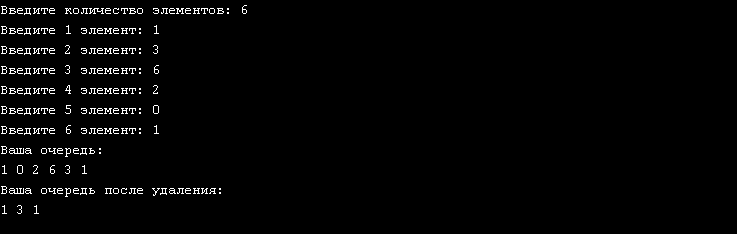
print();

return 0;

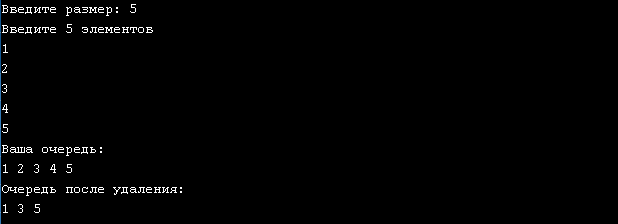
}

**Результаты выполнения**

**Структура:**

****

**STL:**

****