Министерство образования и науки Российской Федерации  
Федеральное‌ ‌государственное‌ ‌бюджетное‌ ‌образовательное‌ ‌учреждение‌

высшего‌ ‌образования‌

**«Пермский национальный исследовательский  
политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**ОТЧЁТ**

**по лабораторной работе 11.4**

Дисциплина: Основы алгоритмизации и программирования

Выполнил:

Студентка группы ИВТ-22-2б

Мифтахов Марат Ринатович

Проверил:

Доцент кафедры ИТАС

Полякова О. А.

**Пермь, 2023**

**Постановка задачи**

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

**Анализ задачи**

1. Создаём структуру очереди

struct Queue {

int data;

Queue\* next;

};

1. Создаём функцию для заполнения очереди

Queue\* Create(int n, Queue\*& first, Queue\*& last)

{

if (n == 0)

return NULL;

Queue\* p = new Queue;

int a;

cout << "Введите элементы очереди: ";

cin >> a;

p->data = a;

p->next = NULL;

first = p;

last = p;

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

{

Queue\* h = new Queue;

cin >> a;

h->data = a;

h->next = p;

last = h;

p = last;

}

return first;

}

1. Создаём функции Рop и Рush для удаления последнего элемента с чётным информационным полем из очереди.

Queue\* Push(int n, Queue\*& last)

{

Queue\* p = new Queue;

p->data = n;

p->next = last;

last = p;

return last;

}

int Pop(Queue\*& first, Queue\*& last)

{

Queue\* 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;

}

}

1. Для вывода создаём функцию Print

void Print(Queue\* last)

{

Queue\* p = last;

while (p != NULL)

{

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

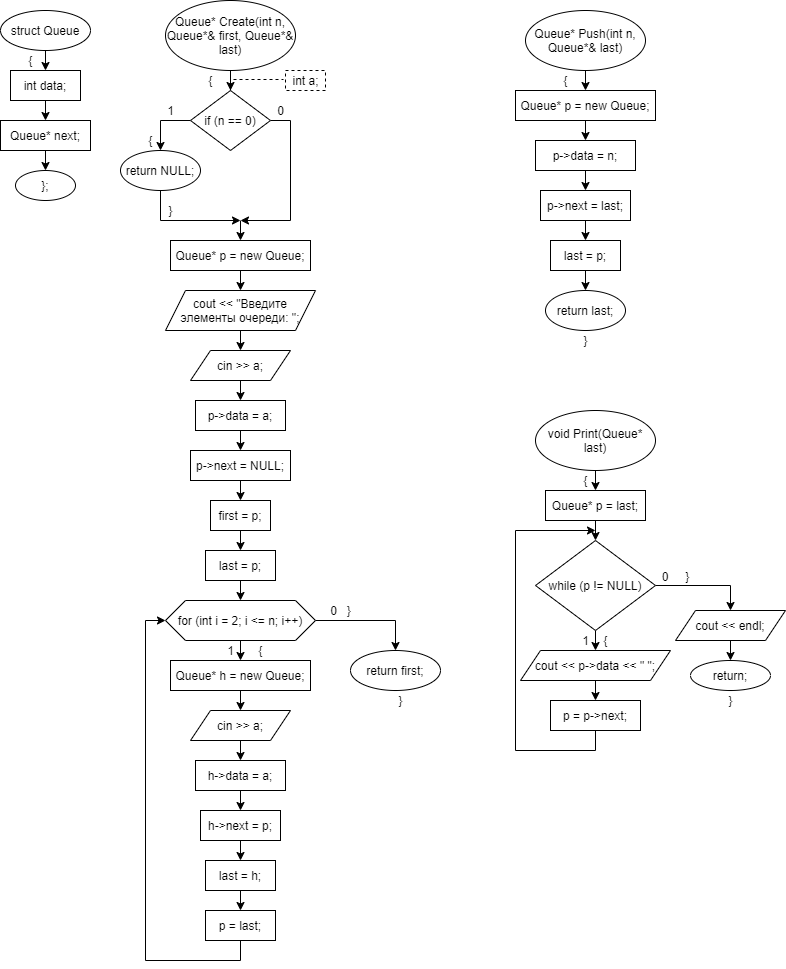
p = p->next;

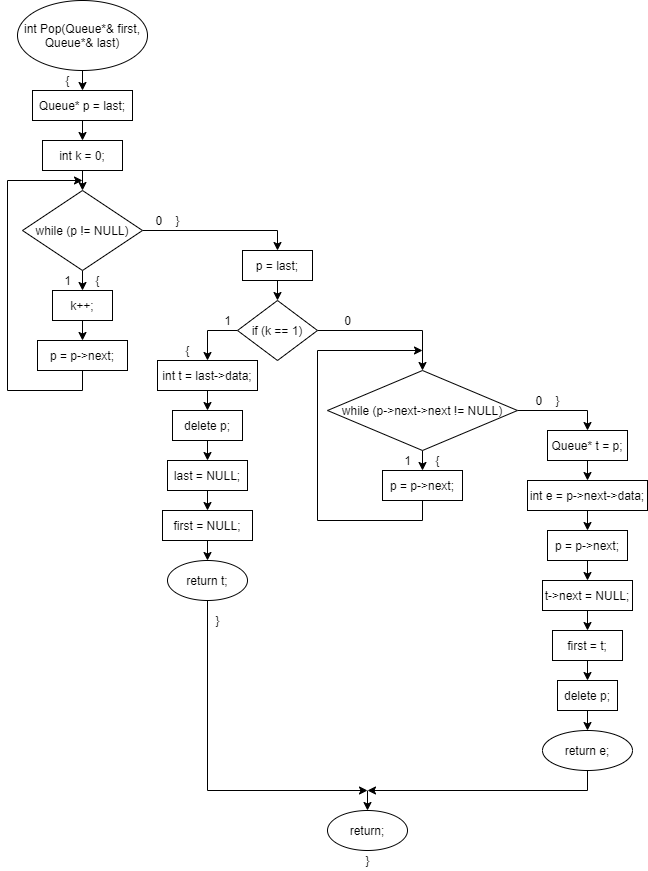
}

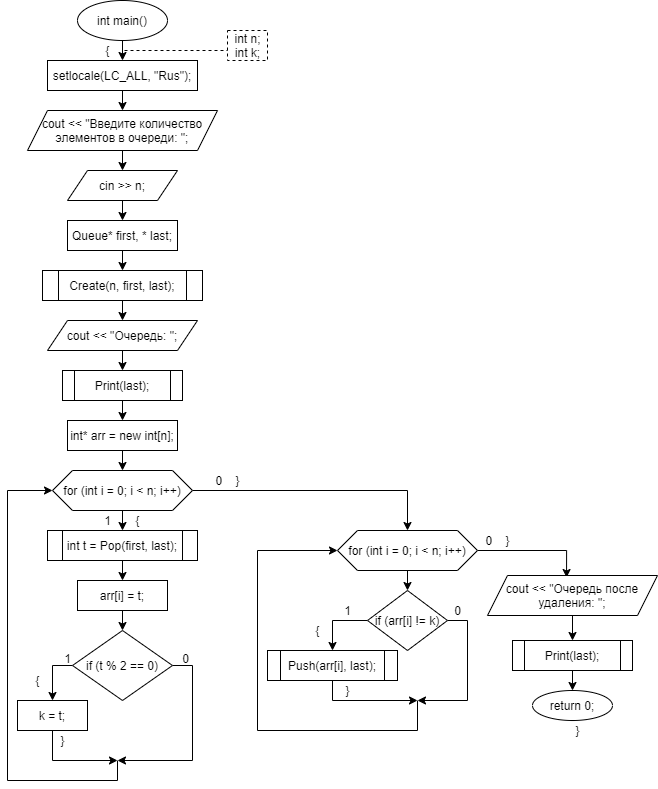
cout << endl;

}

**Блок-схема**





**Исходный код**

#include <iostream>

using namespace std;

struct Queue {

int data;

Queue\* next;

};

Queue\* Create(int n, Queue\*& first, Queue\*& last)

{

if (n == 0)

return NULL;

Queue\* p = new Queue;

int a;

cout << "Введите элементы очереди: ";

cin >> a;

p->data = a;

p->next = NULL;

first = p;

last = p;

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

{

Queue\* h = new Queue;

cin >> a;

h->data = a;

h->next = p;

last = h;

p = last;

}

return first;

}

Queue\* Push(int n, Queue\*& last)

{

Queue\* p = new Queue;

p->data = n;

p->next = last;

last = p;

return last;

}

void Print(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;

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;

}

}

int main()

{

setlocale(LC\_ALL, "Rus");

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

int n;

int k;

cin >> n;

Queue\* first, \* last;

Create(n, first, last);

cout << "Очередь: ";

Print(last);

int\* arr = new int[n];

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

{

int t = Pop(first, last);

arr[i] = t;

if (t % 2 == 0)

{

k = t;

}

}

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

{

if (arr[i] != k)

{

Push(arr[i], last);

}

}

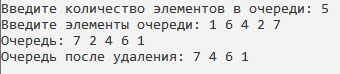
cout << "Очередь после удаления: ";

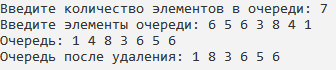
Print(last);

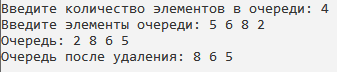
return 0;

}

**Скриншоты программы**

****

****

****

**Анализ (объяснение скриншотов)**

Программа запрашивает у пользователя количество элементов очереди и все элементы очереди, затем выводит очередь. Далее программа удаляет последний элемент с четным информационным полем и выводит получившуюся очередь.