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 №11.2**

Дисциплина: «Основы алгоритмизации и программирования»

Тема: Информационные и динамические структуры. Очереди.

Вариант 23

Выполнил:

Студент группы РИС-20-1б

Кузнецов Михаил Сергеевич

Проверила:

Доцент кафедры ИТАС

Полякова О. А.

**Пермь, 2021**

**Цель работы**

Получить практические навыки работы с очередями.

**Постановка задачи**

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

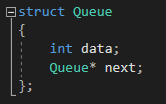
2. Распечатать полученную структуру.

3. Выполнить обработку структуры в соответствии с заданием.

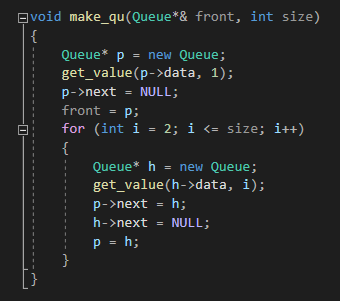
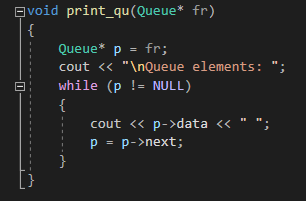
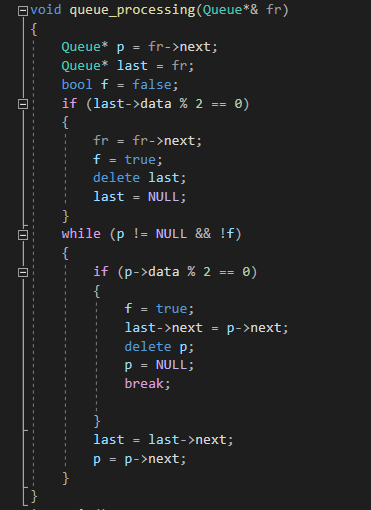
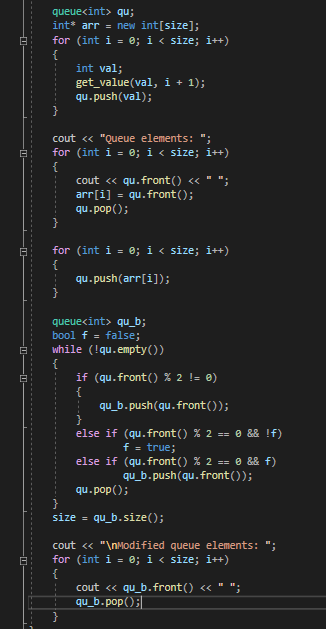
4. Распечатать полученный результат.

5. Удалить соответствующую структуру из памяти.

**Анализ задачи**

1. Определить какие действия необходимо выполнить для решения задачи.
   1. Реализация очереди производится через структуру Queue. 
   2. Создать очередь.



* 1. Заполнить очередь целыми числами. 
  2. Вывести все элементы очереди в консоль. 
  3. Поиск первого чётного элемента очереди. 
  4. Реализация через STL. 

1. Для решения задачи использовались следующие типы данных.

2.1 Данные элементов списка хранятся в формате int.



2.2 Указатель на следующий элемент является экземпляром структуры.



**Код**

#include <iostream>

#include <queue>

using namespace std;

struct Queue

{

int data;

Queue\* next;

};

template <typename T>

void get\_value(T& value, int i)

{

bool is\_correct = false;

do

{

cout << "Enter " << i << " element\n";

cin >> value;

if (cin.fail())

{

cout << "\nIncorrect value" << endl << endl;

cin.clear();

cin.ignore(32767, '\n');

}

else

{

is\_correct = true;

cin.ignore(32767, '\n');

}

} while (!is\_correct);

}

void make\_qu(Queue\*& front, int size)

{

Queue\* p = new Queue;

get\_value(p->data, 1);

p->next = NULL;

front = p;

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

{

Queue\* h = new Queue;

get\_value(h->data, i);

p->next = h;

h->next = NULL;

p = h;

}

}

void print\_qu(Queue\* fr)

{

Queue\* p = fr;

cout << "\nQueue elements: ";

while (p != NULL)

{

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

p = p->next;

}

}

void queue\_processing(Queue\*& fr)

{

Queue\* p = fr->next;

Queue\* last = fr;

bool f = false;

if (last->data % 2 == 0)

{

fr = fr->next;

f = true;

delete last;

last = NULL;

}

while (p != NULL && !f)

{

if (p->data % 2 == 0)

{

f = true;

last->next = p->next;

delete p;

p = NULL;

break;

}

last = last->next;

p = p->next;

}

}

int main()

{

int size;

do

{

cout << "Enter size of queue\n";

cin >> size;

if (cin.fail())

{

cin.clear();

cin.ignore(32767, '\n');

size = -1;

}

cin.ignore(32767, '\n');

} while (size <= 0);

cout << "\nManual queue implementation:\n\n";

Queue\* fr;

make\_qu(fr, size);

print\_qu(fr);

queue\_processing(fr);

print\_qu(fr);

cout << "\nSTL queue implementation:\n\n";

queue<int> qu;

int\* arr = new int[size];

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

{

int val;

get\_value(val, i + 1);

qu.push(val);

}

cout << "Queue elements: ";

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

{

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

arr[i] = qu.front();

qu.pop();

}

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

{

qu.push(arr[i]);

}

queue<int> qu\_b;

bool f = false;

while (!qu.empty())

{

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

{

qu\_b.push(qu.front());

}

else if (qu.front() % 2 == 0 && !f)

f = true;

else if (qu.front() % 2 == 0 && f)

qu\_b.push(qu.front());

qu.pop();

}

size = qu\_b.size();

cout << "\nModified queue elements: ";

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

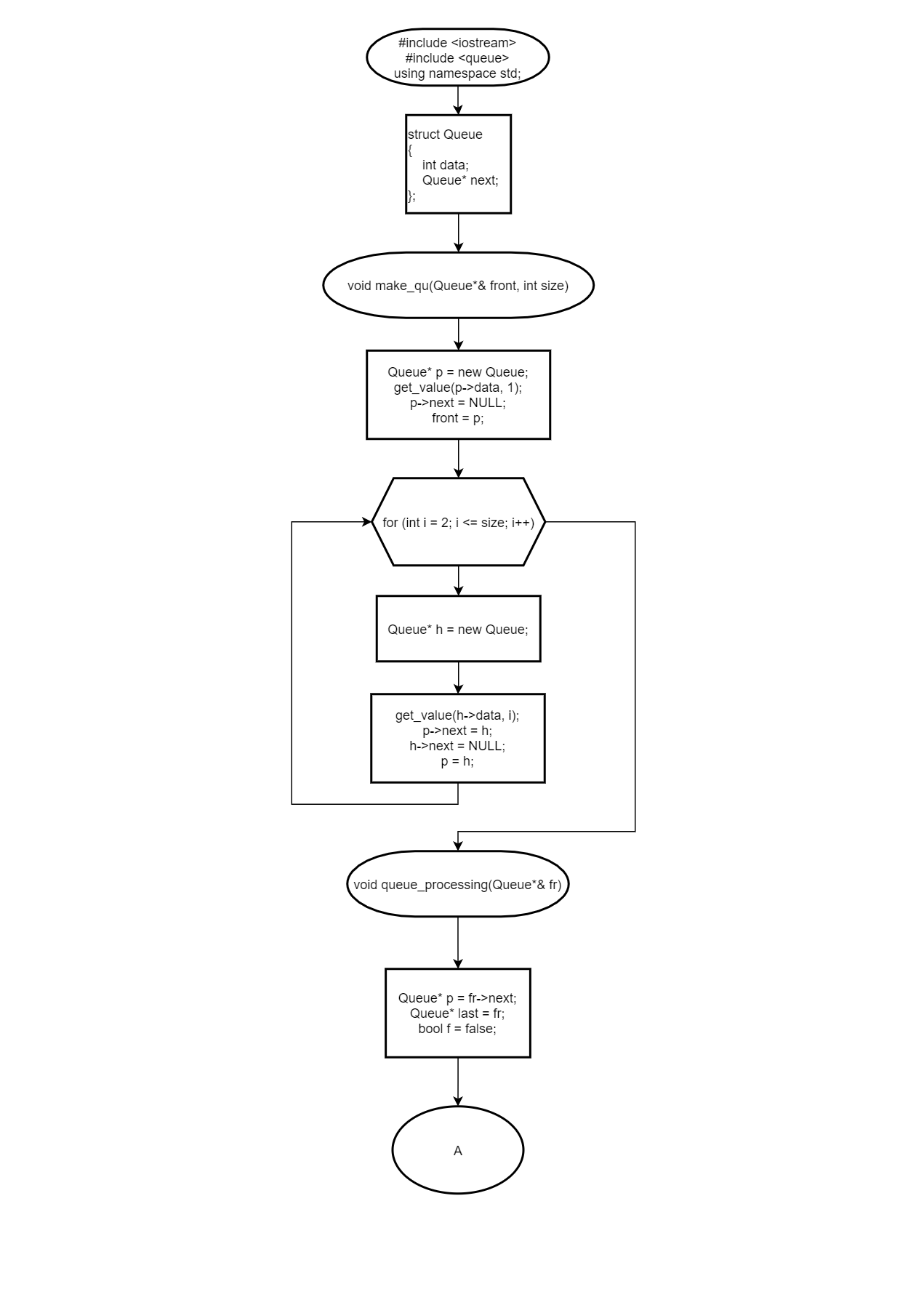
{

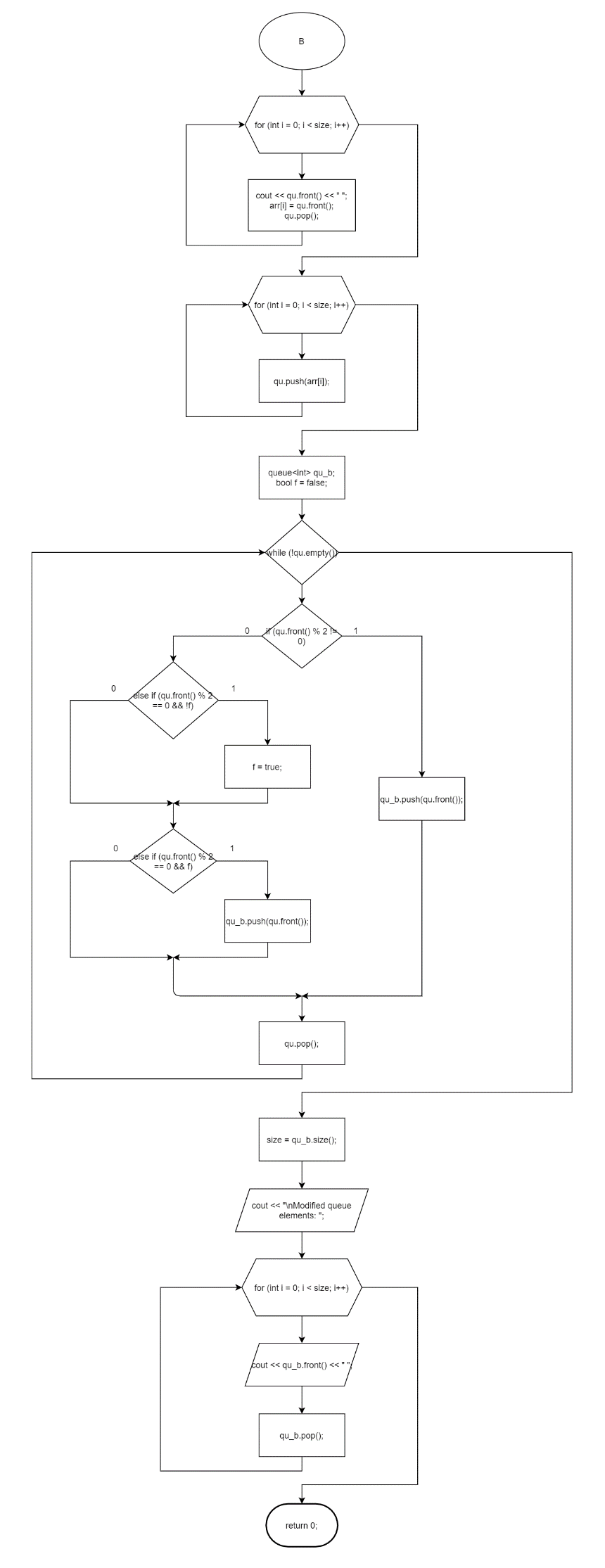
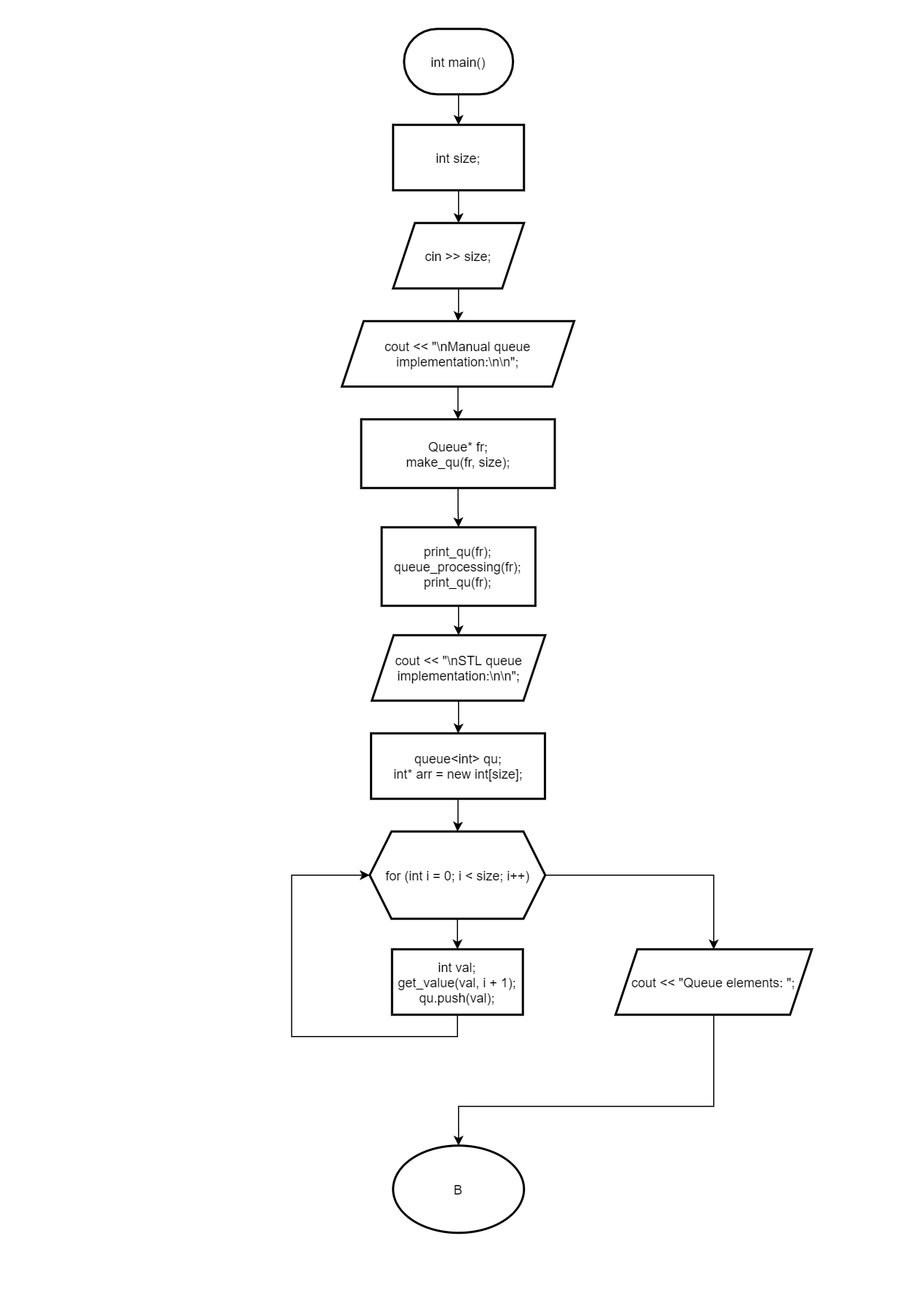
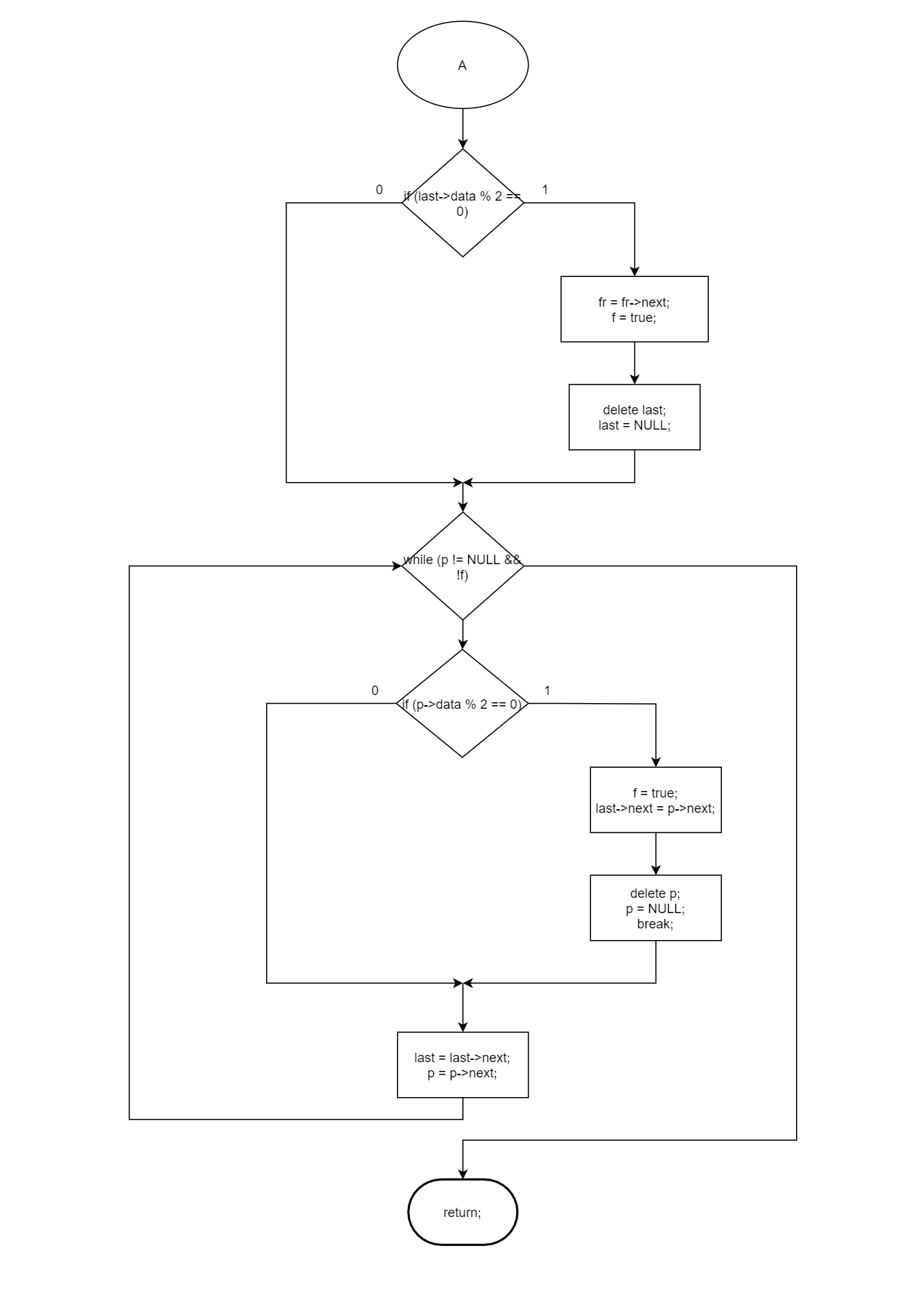
cout << qu\_b.front() << " ";

qu\_b.pop();

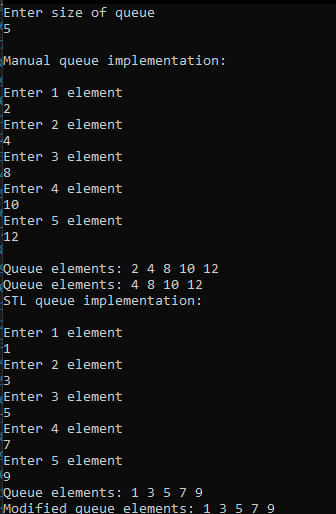
}

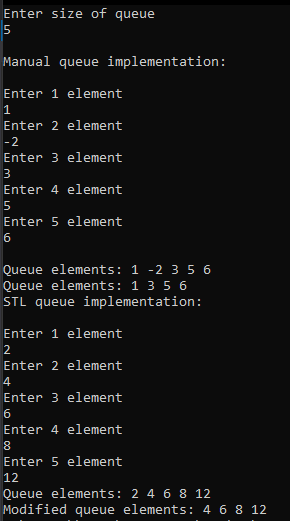
}

**Блок схема **

****

**Тесты №4**

****

****