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 №11.1**

Дисциплина: «Основы алгоритмизации и программирования»

Тема: Информационные и динамические структуры. Одно- и Двунаправленные списки

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 однонаправленными списками.

**Постановка задачи**

1. Сформировать однонаправленный и двунаправленный списки. Тип информационного поля double. Добавить в список после каждого элемента с отрицательным информационным полем элемент с информационным полем равным 0.

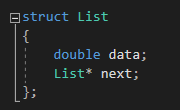
2. Распечатать полученную структуру.

3. Выполнить обработку структуры в соответствии с заданием.

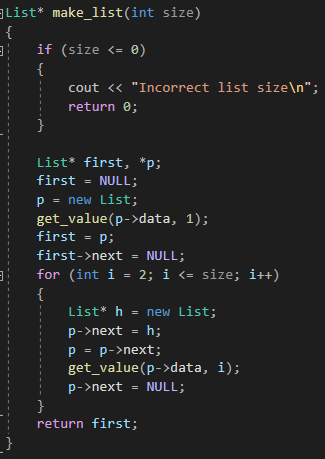
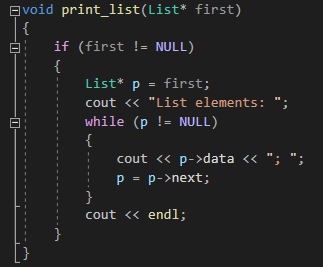
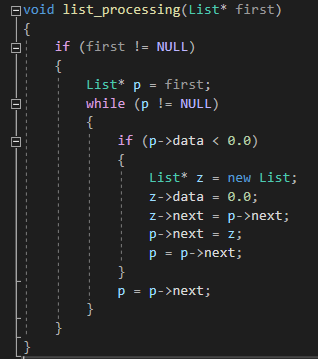
4. Распечатать полученный результат.

5. Удалить соответствующую структуру из памяти.

**Анализ задачи**

1. Определить какие действия необходимо выполнить для решения задачи.
   1. Реализация списка производится через структуру List. 
   2. Создать однонаправленный список.



* 1. Заполнить список вещественными числами. 
  2. Вывести все элементы списка в консоль. 
  3. Далее необходимо найти элементы с отрицательными значениями, и перед ними вставить элемент со значением равным 0. 
  4. В конце выводятся все элементы списка.

1. Для решения задачи использовались следующие типы данных.

2.1 Данные элементов списка хранятся в формате double.



2.2 Указатель на следующий элемент является экземпляром структуры. 

**Код**

#include <iostream>

using namespace std;

struct List

{

double data;

List\* 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);

}

List\* make\_list(int size)

{

if (size <= 0)

{

cout << "Incorrect list size\n";

return 0;

}

List\* first, \*p;

first = NULL;

p = new List;

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

first = p;

first->next = NULL;

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

{

List\* h = new List;

p->next = h;

p = p->next;

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

p->next = NULL;

}

return first;

}

void print\_list(List\* first)

{

if (first != NULL)

{

List\* p = first;

cout << "List elements: ";

while (p != NULL)

{

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

p = p->next;

}

cout << endl;

}

}

void list\_processing(List\* first)

{

if (first != NULL)

{

List\* p = first;

while (p != NULL)

{

if (p->data < 0.0)

{

List\* z = new List;

z->data = 0.0;

z->next = p->next;

p->next = z;

p = p->next;

}

p = p->next;

}

}

}

void delete\_list(List\* first)

{

if (first != NULL)

{

List\* p = first;

while (p != NULL)

{

List\* q = p;

p = p->next;

delete q;

}

}

}

int main()

{

int size;

do

{

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

cin >> size;

if (cin.fail())

{

cin.clear();

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

size = -1;

}

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

} while (size < 0);

List\* l = make\_list(size);

print\_list(l);

list\_processing(l);

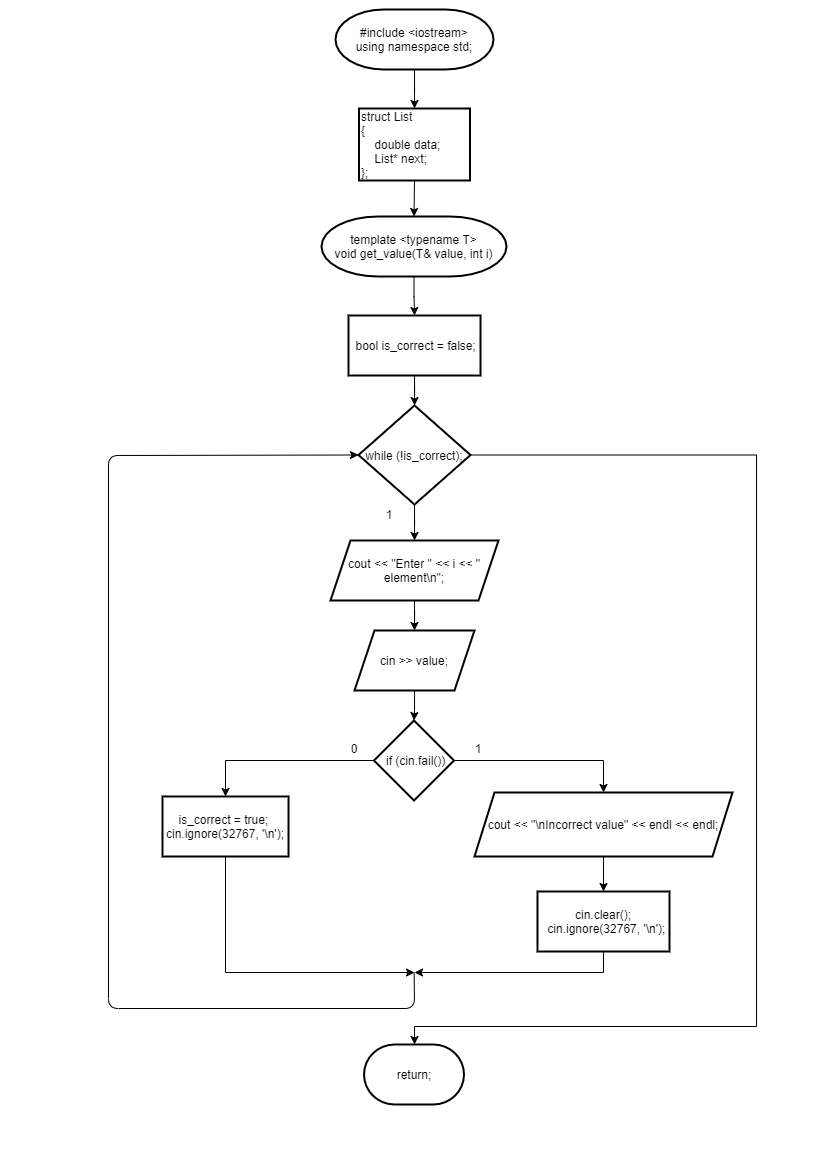
print\_list(l);

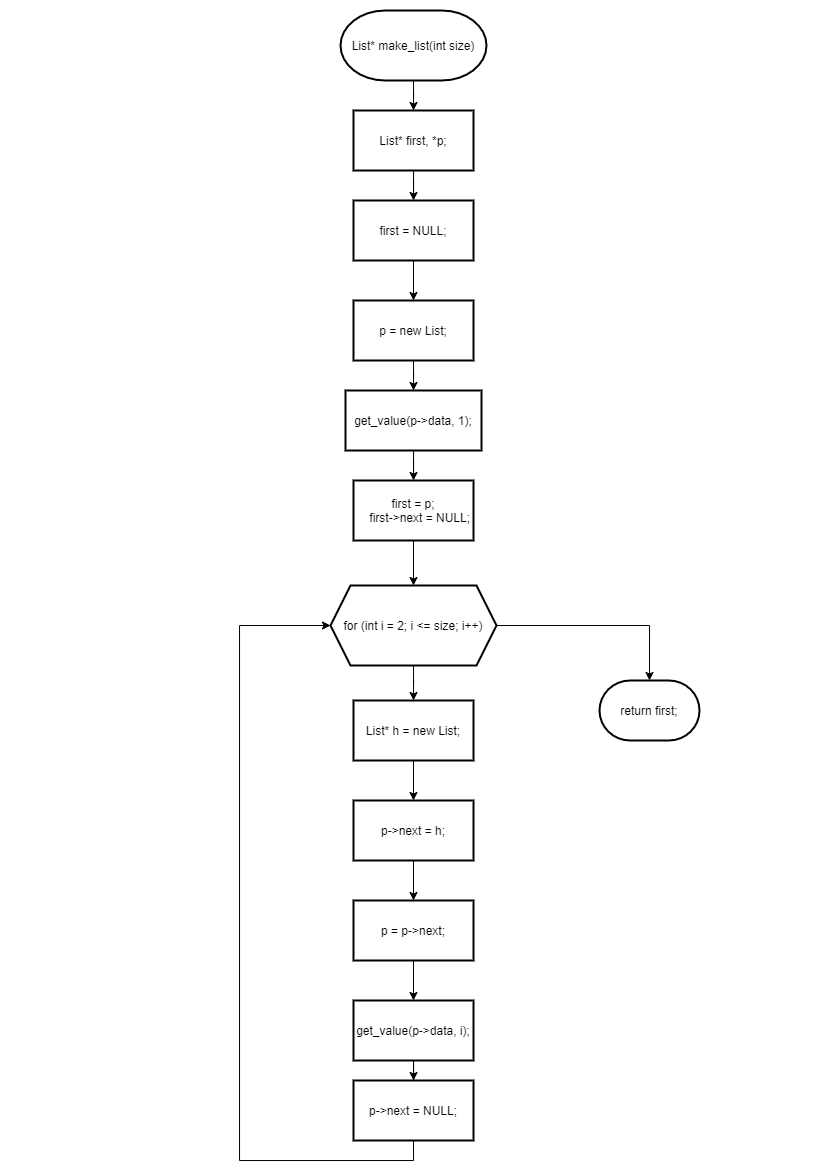
delete\_list(l);

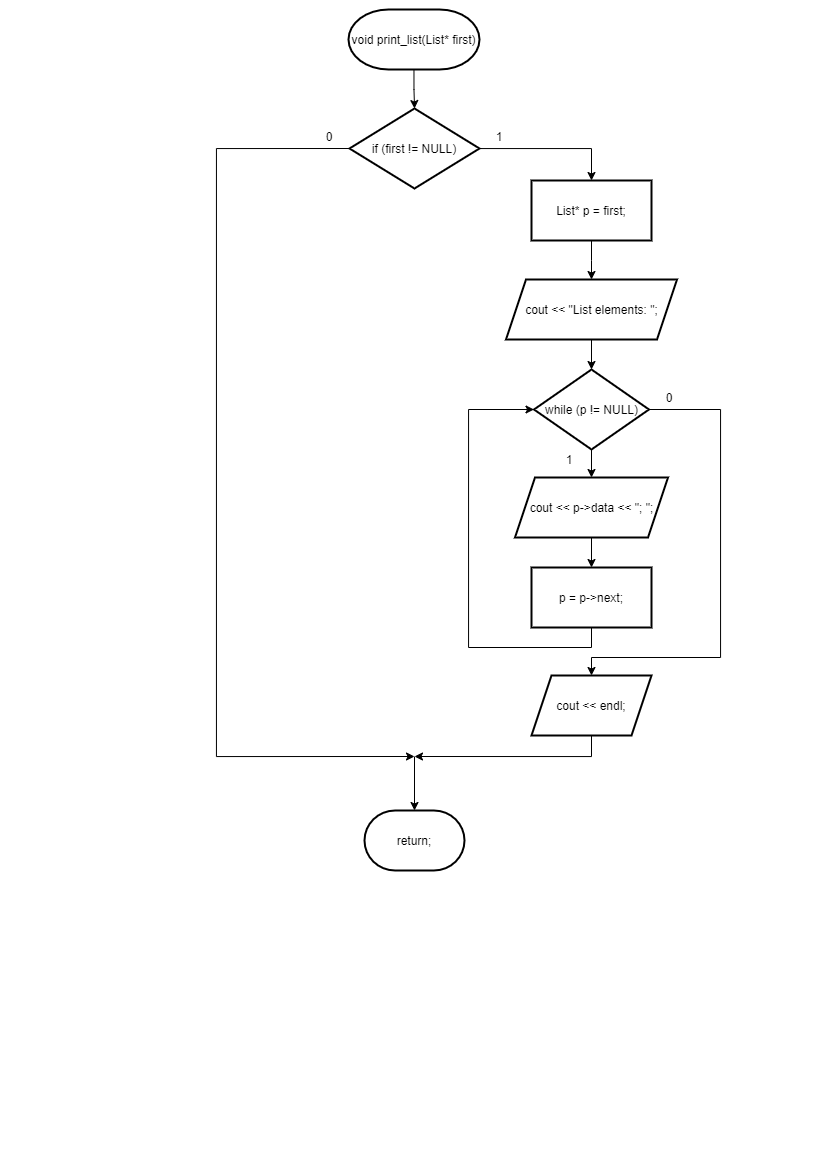
return 0;

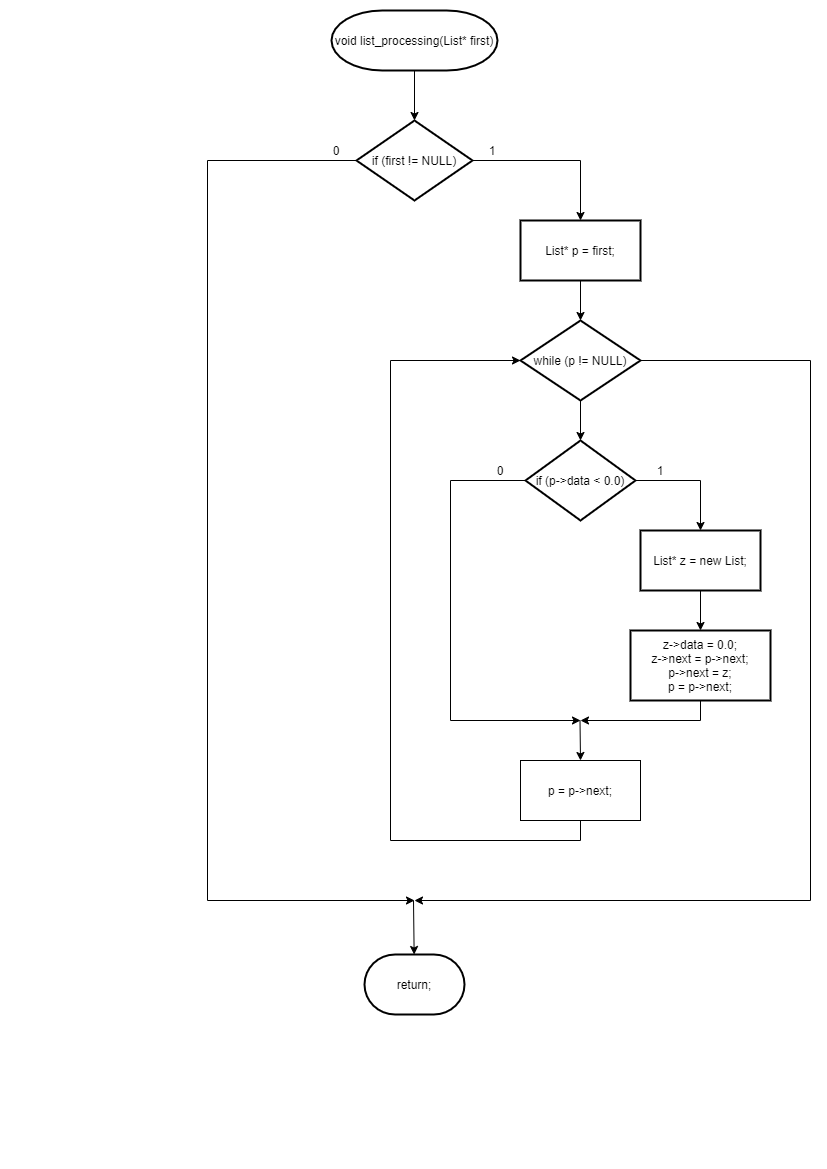
}

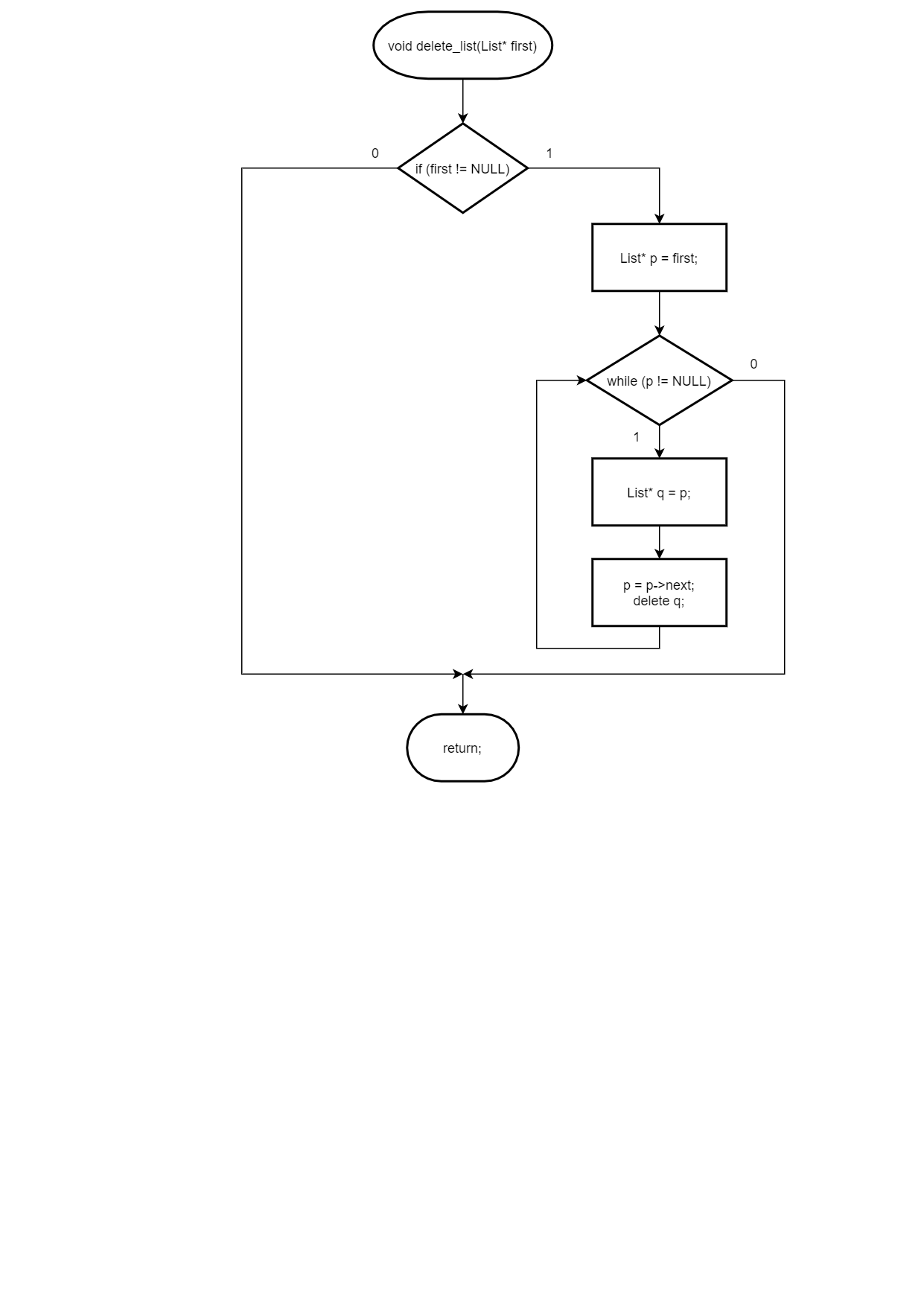
**Блок-схема**

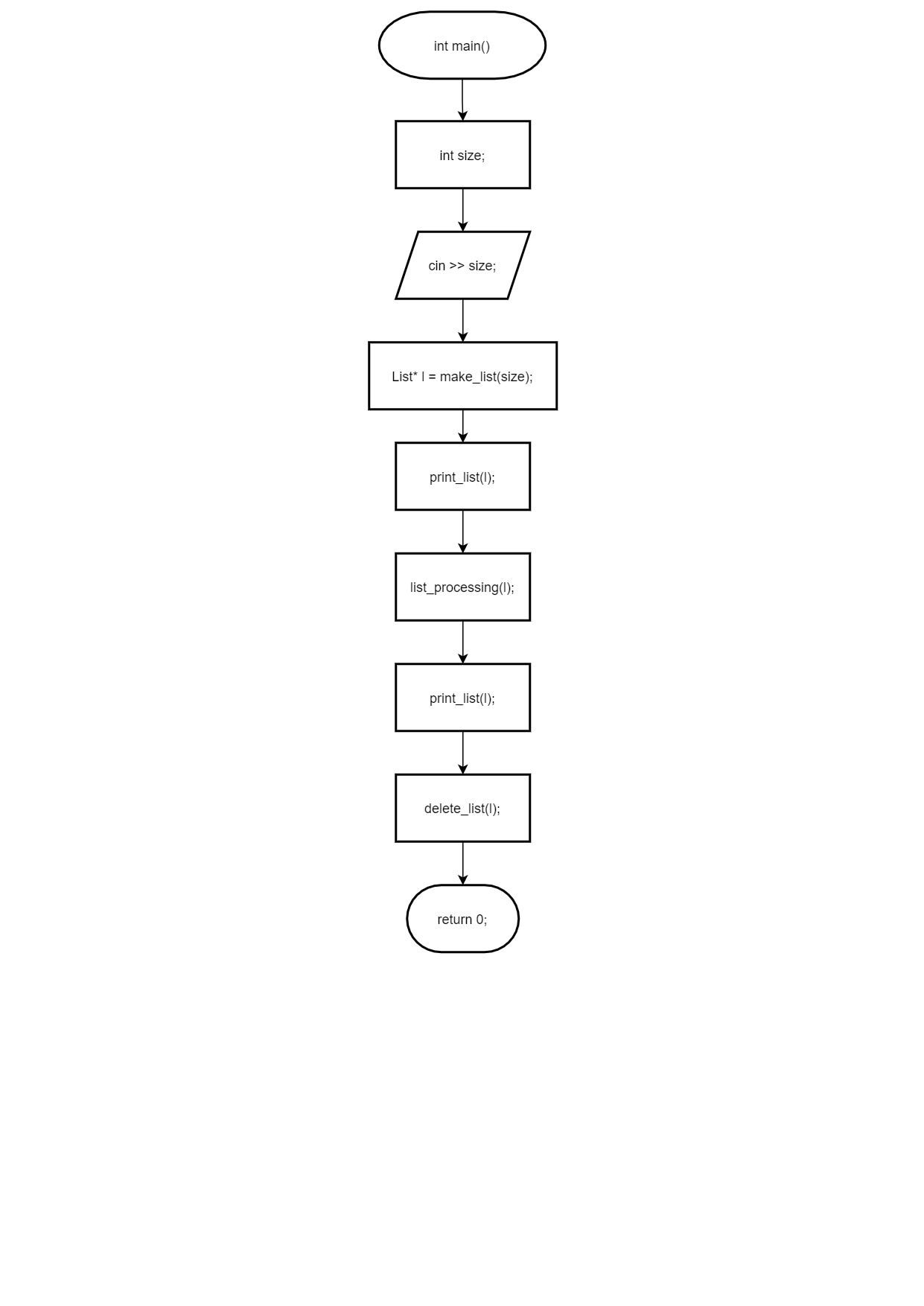
****

****

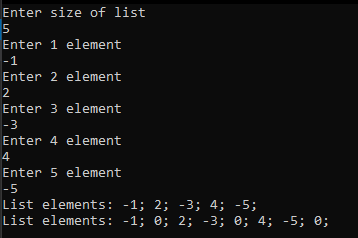
****

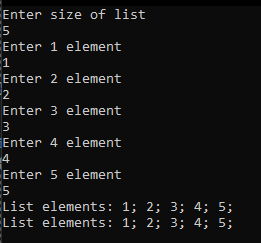
****

****

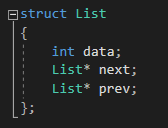
****

**Скриншоты тестов №1**

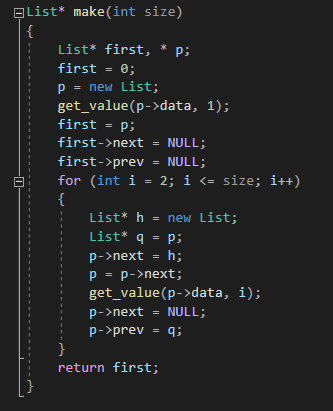
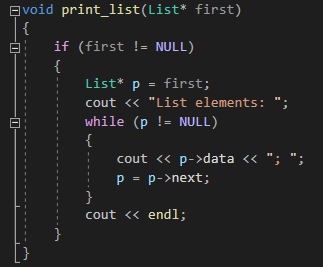
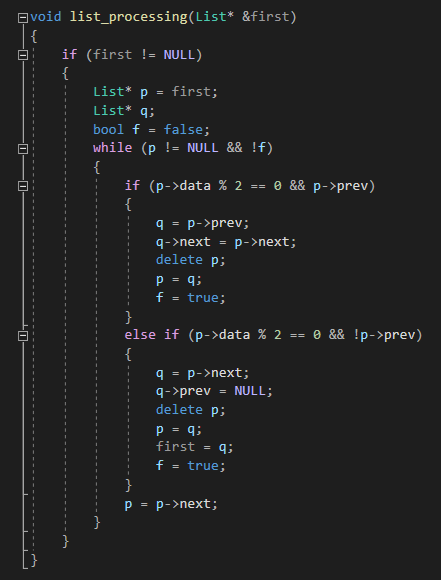
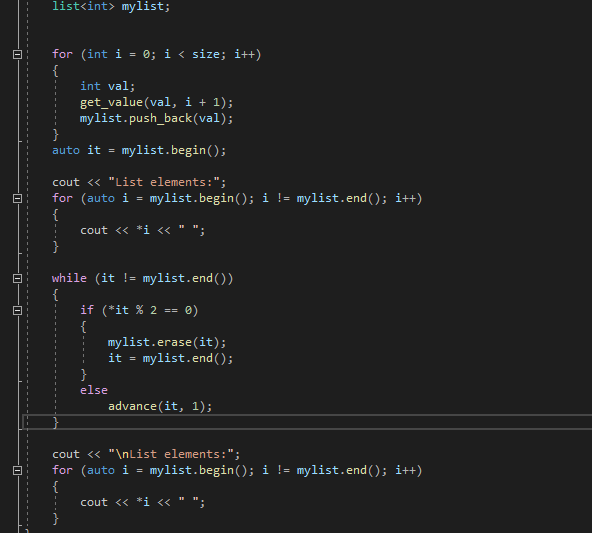
****

****

**Анализ задачи №2**

1. Определить какие действия необходимо выполнить для решения задачи.
   1. Реализация списка производится через структуру List. 
   2. Создать двунаправленный список.



* 1. Заполнить список целыми числами. 
  2. Вывести все элементы списка в консоль. 
  3. Далее необходимо найти первый чётный элемент. 
  4. В конце выводятся все элементы списка.
  5. 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 и предыдущий элемент является экземпляром структуры.



**Код №2**

#include <iostream>

#include <list>

#include <iterator>

using namespace std;

struct List

{

int data;

List\* next;

List\* prev;

};

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);

}

List\* make(int size)

{

List\* first, \* p;

first = 0;

p = new List;

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

first = p;

first->next = NULL;

first->prev = NULL;

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

{

List\* h = new List;

List\* q = p;

p->next = h;

p = p->next;

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

p->next = NULL;

p->prev = q;

}

return first;

}

void print\_list(List\* first)

{

if (first != NULL)

{

List\* p = first;

cout << "List elements: ";

while (p != NULL)

{

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

p = p->next;

}

cout << endl;

}

}

void list\_processing(List\* &first)

{

if (first != NULL)

{

List\* p = first;

List\* q;

bool f = false;

while (p != NULL && !f)

{

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

{

q = p->prev;

q->next = p->next;

delete p;

p = q;

f = true;

}

else if (p->data % 2 == 0 && !p->prev)

{

q = p->next;

q->prev = NULL;

delete p;

p = q;

first = q;

f = true;

}

p = p->next;

}

}

}

int main()

{

int size;

do

{

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

cin >> size;

if (cin.fail())

{

cin.clear();

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

size = -1;

}

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

} while (size <= 0);

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

List\* l = make(size);

print\_list(l);

list\_processing(l);

print\_list(l);

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

list<int> mylist;

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

{

int val;

get\_value(val, i + 1);

mylist.push\_back(val);

}

auto it = mylist.begin();

cout << "List elements:";

for (auto i = mylist.begin(); i != mylist.end(); i++)

{

cout << \*i << " ";

}

while (it != mylist.end())

{

if (\*it % 2 == 0)

{

mylist.erase(it);

it = mylist.end();

}

else

advance(it, 1);

}

cout << "\nList elements:";

for (auto i = mylist.begin(); i != mylist.end(); i++)

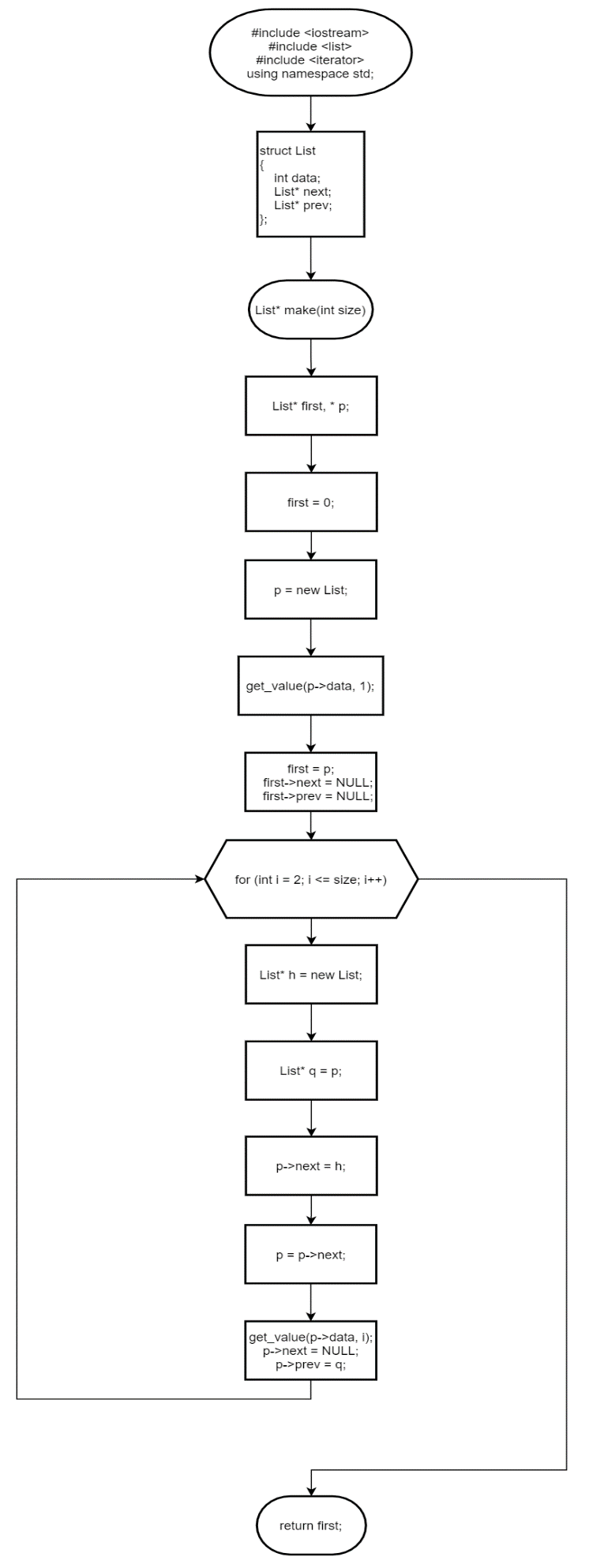
{

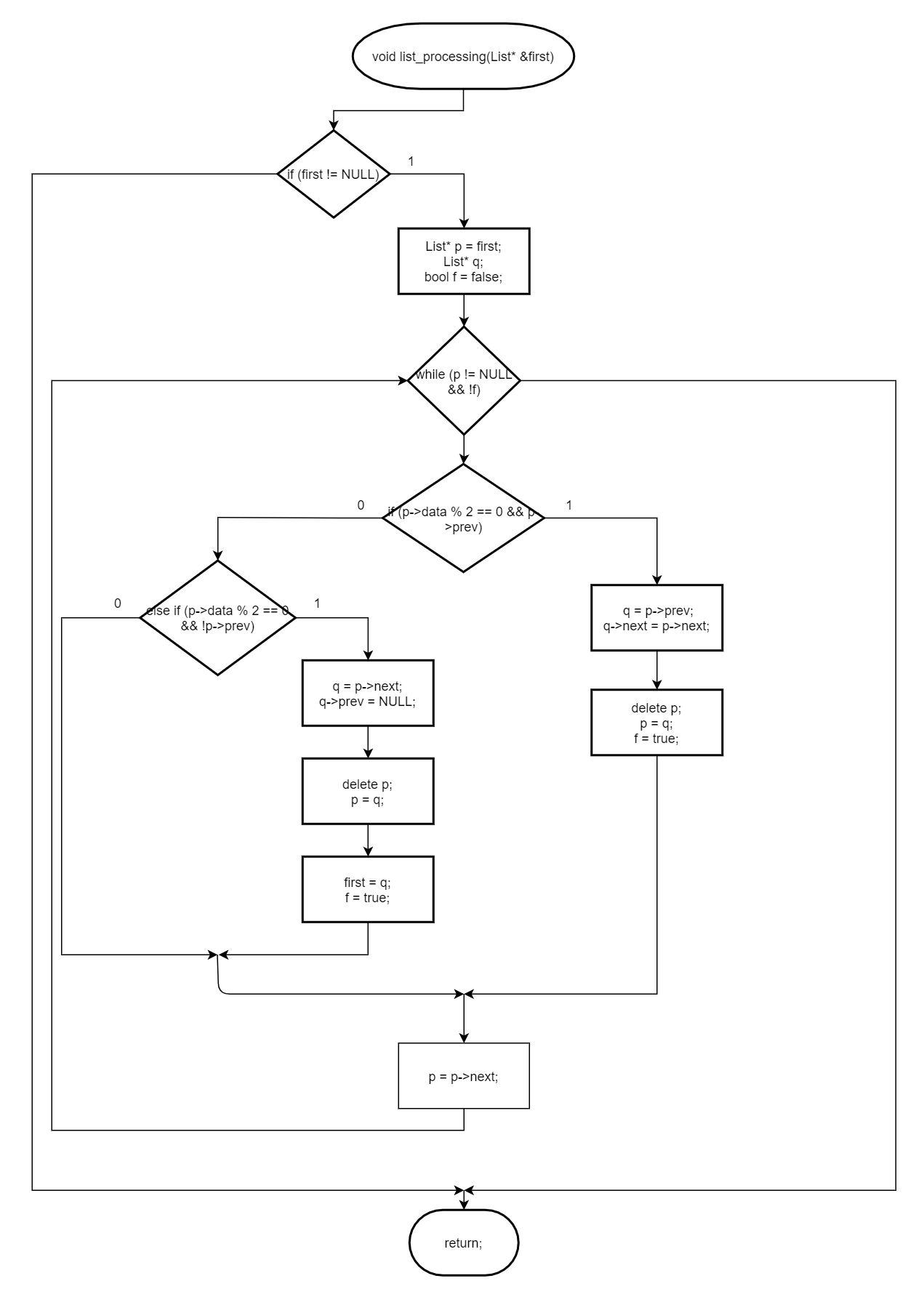
cout << \*i << " ";

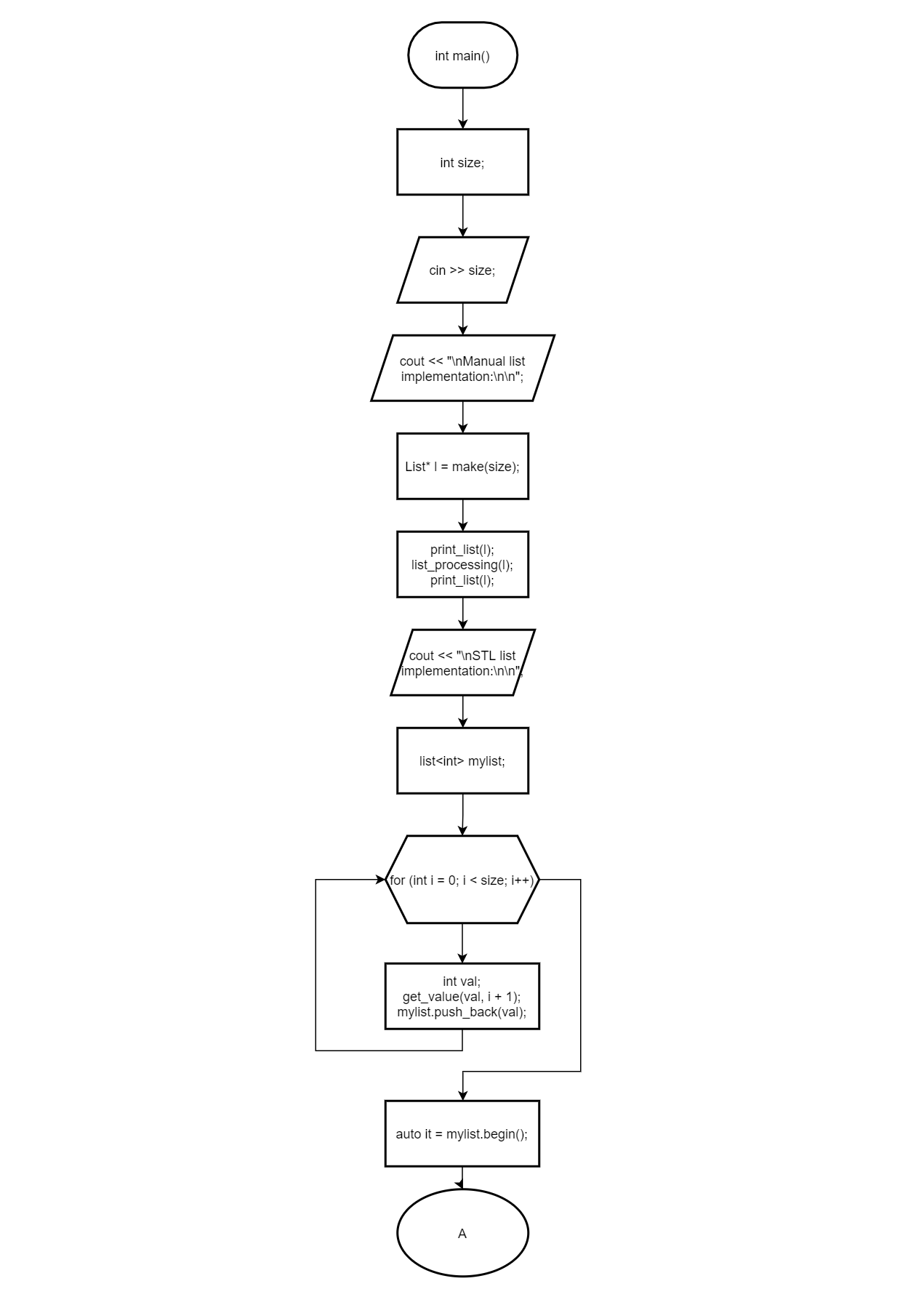
}

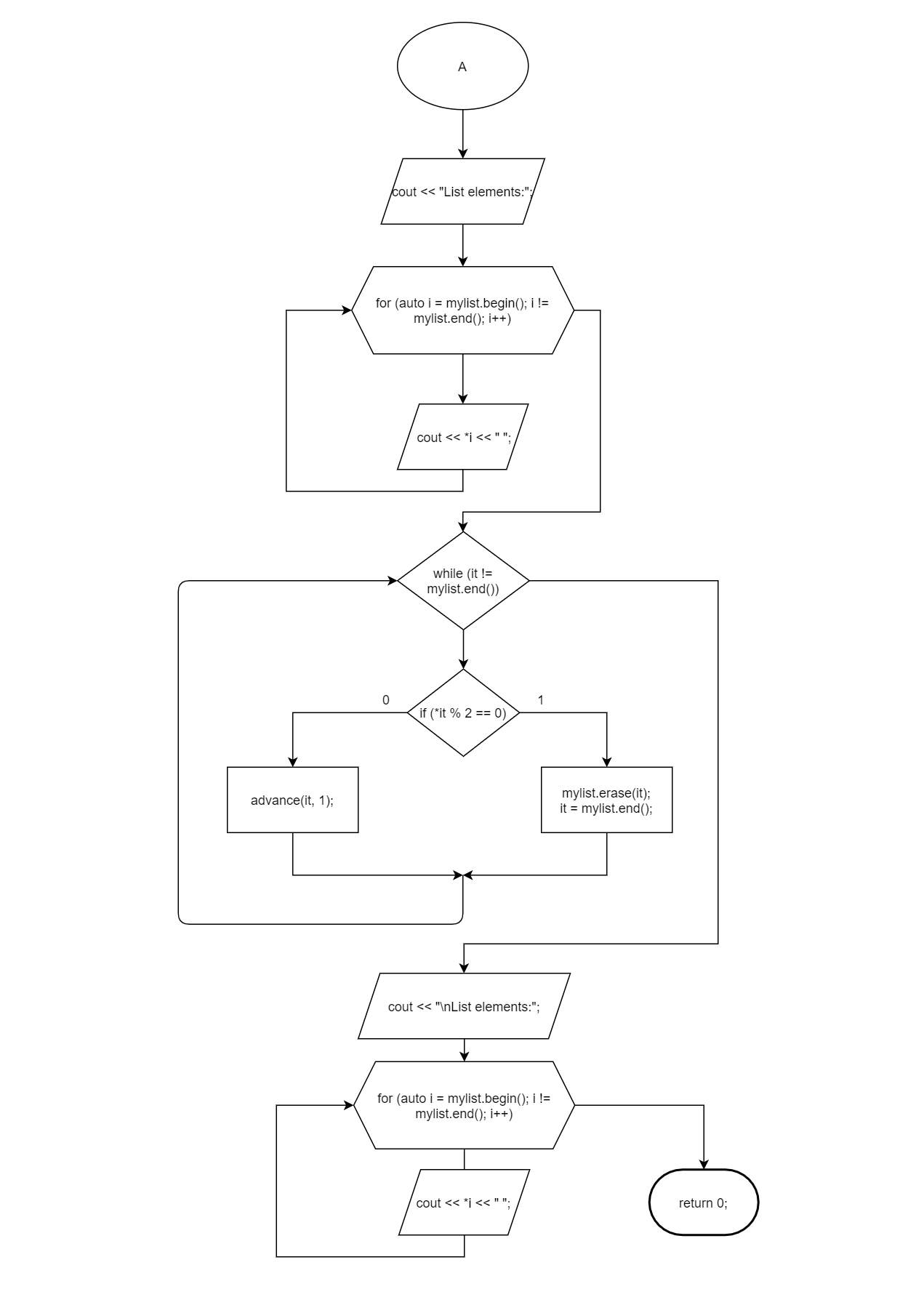
}

**Блок схема №2**

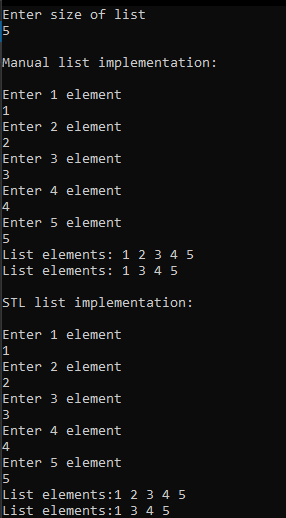
****

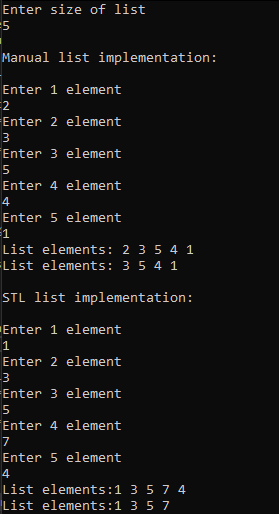
****

****

****

**Тесты №3**

****

****