Министерство образования и науки Российской Федерации

Федеральное‌ ‌государственное‌ ‌бюджетное‌ ‌образовательное‌ ‌учреждение‌

высшего‌ ‌образования‌

**«Пермский национальный исследовательский**

**политехнический университет»**

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

**О Т Ч Ё Т**

**по лабораторной работе №11.1**

Дисциплина: «Основы алгоритмизации и программирования»

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

Вариант 18

Выполнил:

Студент группы РИС-20-1б

Тараканов Д. М.

Проверила:

Доцент кафедры ИТАС

Полякова О. А.

Пермь

2021 год

# **Цель работы**

1) Получить практические навыки работы с однонаправленными списками;

2) Получить практические навыки работы с двунаправленными списками;

3) Получить практические навыки работы со списками через STL;

# **Постановка задачи**

Задача состоит в разработке программы, в которой создаётся однонаправленный, двунаправленный список и список через STL, выполнить обработку списков в соответствии со своим вариантом:

(18 вариант)

Однонаправленный список

1. Сформировать однонаправленный список с информационным полем типа double.
2. Удалить из списка все элементы с четными номерами (2, 4, 6 и т.д.).

Двунаправленный список

1. Сформировать двунаправленный список с информационным полем типа char.
2. Добавить в список элемент с заданным номером.

Список через STL

1. Сформировать список через STL с информационным полем типа char.
2. Добавить в список элемент с заданным номером.

# **Анализ задачи**

Однонаправленный список

1. Определить какие предстоит выполнить действия:

* Создать структуру в виде однонаправленного списка.

struct List {

double data;

List\* next;

};

* Разработать функцию FillList для заполнения списка.

List\* FillList(int N) {

List\* first, \* p = new List;

cout << "\nВведите 1 элемент: ";

cin >> p->data;

first = p;

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

List\* k = new List;

p->next = k;

p = p->next;

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

cin >> p->data;

p->next = nullptr;

}

return first;

};

* Разработать функцию удаление элементов с четными индексами в списке.

void DeleteElement(List\* first) {

List\* p = first;

int count = 1;

while (p->next!=nullptr) {

count++;

if ((count%2) == 0) {

List\* q = p->next;

p->next = p->next->next;

delete q;

}

else {

p = p->next;

}

}

};

2. С какими типами данных действие надо сделать, в каком виде эти данные будут представлены:

* Для запоминания длины списка используем переменную size типа int.

int size=-1;

* Для данных в списке используем переменную data типа double.

double data;

* Для хранения адреса следующего элемента используем указатель next типа List.

List\* next;

* Для обращения к списку используем указатель list на первый элемент списка типа List.

List\* list = FillList(size);

* Для функции FillList используем следующие элементы.

1. Длину списка типа int.

Сама функция имеет тип List так, как входе её работы функция возвращает указатель на первый элемент списка.

List\* FillList(int N)

* Для функции PrintList используем следующие элементы.

1. Указатель на первый элемент списка типа List.

Сама функция имеет тип void так, как входе её работы функция не возвращает значений.

void PrintList(List\* first)

* Для функции DeleteElement используем следующие элементы.

1. Указатель на первый элемент списка типа List.

Сама функция имеет тип void так, как входе её работы функция не возвращает значений.

void DeleteElement(List\* first)

3. Для решения задачи данные были представлены в виде списка элементов типа double.

4. Реализация операций ввода и вывода данных:

* Ввод данных с консоли реализован при помощи оператора cin.

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

List\* k = new List;

p->next = k;

p = p->next;

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

cin >> p->data;

p->next = nullptr;

}

* Вывод данных на консоль реализован при помощи оператора cout.

while (p != nullptr) {

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

p = p->next;

}

5. Поставленные задачи будут решены при помощи следующих действий:

* Программа запрашивает у пользователя длину списка.

while (size < 0) {

cin >> size;

if (size < 0) {

cout << "Введите размер списка: ";

}

}

* При помощи функции FillList происходит заполнение списка с консоли.

List\* FillList(int N) {

List\* first, \* p = new List;

cout << "\nВведите 1 элемент: ";

cin >> p->data;

first = p;

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

List\* k = new List;

p->next = k;

p = p->next;

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

cin >> p->data;

p->next = nullptr;

}

return first;

};

* При помощи функции DeleteElement удаляются элементы списка с четными индексами. Так как у списка нет индексов вводится новая переменная count. При помощи оператора while увеличивается переменная count и если она четна, то программа удаляет данный элеменет иначе переходит на следующий элемент списка. Цикл продолжается пока не достигнет конца списка.

void DeleteElement(List\* first) {

List\* p = first;

int count = 1;

while (p->next!=nullptr) {

count++;

if ((count%2) == 0) {

List\* q = p->next;

p->next = p->next->next;

delete q;

}

else {

p = p->next;

}

}

};

* При помощи функции PrintList программа выводит элементы списка на консоль.

void PrintList(List\* first) {

List\* p = first;

while (p != nullptr) {

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

p = p->next;

}

cout << endl;

};

Двунаправленный список

1. Определить какие предстоит выполнить действия:

* Создать структуру в виде двунаправленного списка.

struct List {

char data;

List\* next;

List\* prev;

};

* Разработать функцию FillList для заполнения списка.

List\* FillList(int N) {

List\* first, \* p = new List;

cout << "\nВведите 1 элемент: ";

cin >> p->data;

first = p;

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

List\* k = new List;

List\* l = p;

p->next = k;

p = p->next;

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

cin >> p->data;

p->next = nullptr;

p->prev = l;

}

return first;

};

* Разработать функцию AddElement для добавления в список элемента с заданным номером.

void AddElement(List\*& first, int size) {

int pos;

cout << "\nВведите номер элемента: ";

do {

cin >> pos;

if ((pos > size) || (pos < 1)) {

cout << "Введите номер элемента: ";

}

} while (pos > size || pos < 1); cout << "Введите элемент: ";

char el;

cin >> el;

List\* p = new List;

p->data = el;

List\* h = first;

for (int i = pos; i > 1; i--) {

h = h->next;

}

h->prev->next = p;

p->prev = h->prev;

p->next = h;

h->prev = p;

};

2. С какими типами данных действие надо сделать, в каком виде эти данные будут представлены:

* Для запоминания длины списка используем переменную size типа int.

int size=-1;

* Для данных в списке используем переменную data типа double.

double data;

* Для хранения адреса следующего элемента используем указатель next типа List.

List\* next;

* Для хранения адреса предыдущего элемента используем указатель prev типа List.

List\* prev;

* Для хранения нового элемента используем переменную el типа char.

char el;

* Для обращения к списку используем указатель list на первый элемент списка типа List.

List\* list = FillList(size);

* Для функции FillList используем следующие элементы.

1. Длину списка типа int.

Сама функция имеет тип List так, как входе её работы функция возвращает указатель на первый элемент списка.

List\* FillList(int N)

* Для функции PrintList используем следующие элементы.

1. Указатель на первый элемент списка типа List.

Сама функция имеет тип void так, как входе её работы функция не возвращает значений.

void PrintList(List\*& first)

* Для функции AddElement используем следующие элементы.

1. Указатель на первый элемент списка типа List.
2. Длину списка типа int.

Сама функция имеет тип void так, как входе её работы функция не возвращает значений.

void AddElement(List\*& first, int size)

3. Для решения задачи данные были представлены в виде списка элементов типа char.

4. Реализация операций ввода и вывода данных.

* Ввод данных с консоли реализован при помощи оператора cin.

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

List\* k = new List;

List\* l = p;

p->next = k;

p = p->next;

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

cin >> p->data;

p->next = nullptr;

p->prev = l;

}

* Вывод данных на консоль реализован при помощи оператора cout.

while (p != nullptr) {

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

p = p->next;

}

5. Поставленные задачи будут решены при помощи следующих действий:

* Программа запрашивает у пользователя длину двунаправленного списка.

while (size < 0) {

cin >> size;

if (size < 0) {

cout << "Введите размер списка: ";

}

}

* При помощи функции FillList происходит заполнение списка с консоли.

List\* FillList(int N)

{

List\* first, \* p = new List;

cout << "\nВведите 1 элемент: ";

cin >> p->data;

first = p;

p->next = nullptr;

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

List\* k = new List;

List\* l = p;

p->next = k;

p = p->next;

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

cin >> p->data;

p->next = nullptr;

p->prev = l;

}

return first;

};

* При помощи функции AddElement происходит добавления нового элемента перед элементом с указанным индексом. Программа запрашивает у пользователя номер элемента, перед которым ставится новый элемент, а затем с клавиатуры вводится новый элемент. При помощи оператора for указатель доходит до введённого номера элемента. Новый элемент подставляется путем изменения адресов элементов.

void AddElement(List\*& first, int size) {

int pos;

cout << "\nВведите номер элемента: ";

do {

cin >> pos;

if ((pos > size) || (pos < 1)) {

cout << "Введите номер элемента: ";

}

} while (pos > size || pos < 1);

cout << "Введите элемент: ";

char el;

cin >> el;

List\* p = new List;

p->data = el;

List\* h = first;

for (int i = pos; i > 1; i--) {

h = h->next;

}

h->prev->next = p;

p->prev = h->prev;

p->next = h;

h->prev = p;

};

* При помощи функции PrintList программа выводит элементы списка на консоль.

void PrintList(List\* first) {

List\* p = first;

while (p != nullptr) {

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

p = p->next;

}

cout << endl;

};

Список через STL

1. Определить какие предстоит выполнить действия:

* Создать список через STL.

list <char> List;

* Разработать функцию FillList для заполнения списка.

void FillList(int N) {

char el;

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

cout << "Введите " << i + 1 << " элемент: ";

cin >> el;

List.push\_back(el);

}

};

* Разработать функцию AddElement для добавления в список элемента с заданным номером.

void AddElement(int size){

char key;

int N = -1;

while ((N < 1) || (N > size)) {

if ((N < 1) || (N > size)) {

cout << "\nВведите номер элемента: ";

}

cin >> N;

}

list<char> :: iterator p;

p = List.begin();

advance(p, N-1);

list <char> AddList;

cout << "Введите элемент: ";

cin >> key;

AddList.push\_back(key);

List.insert(p, AddList.begin(), AddList.end());

}

2. С какими типами данных действие надо сделать, в каком виде эти данные будут представлены:

* Для запоминания длины списка используем переменную size типа int.

int size=-1;

* Для инициализации списка используем переменную List.

list <char> List;

* Для инициализации итератора для списка используем переменную p.

list<char> :: iterator p;

* Для хранения нового элемента используем переменную key типа char.

char key;

* Для функции FillList используем следующие элементы.

1. Длину списка типа int.

Сама функция имеет тип void так, как входе её работы функция не возвращает значений.

void FillList(int N)

* Для функции AddElement используем следующие элементы.

1. Длину списка типа int.

Сама функция имеет тип void так, как входе её работы функция не возвращает значений.

void AddElement(int size)

3. Для решения задачи данные были представлены в виде списка элементов типа char.

4. Реализация операций ввода и вывода данных.

* Ввод данных с консоли реализован при помощи оператора cin.

void FillList(int N) {

char el;

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

cout << "Введите " << i + 1 << " элемент: ";

cin >> el;

List.push\_back(el);

}

};

* Вывод данных на консоль реализован при помощи итератора copy().

copy(List.begin(), List.end(), ostream\_iterator<char>(cout, " "));

5. Поставленные задачи будут решены при помощи следующих действий:

* Программа запрашивает у пользователя длину списка.

while (size < 0) {

cin >> size;

if (size < 0) {

cout << "Введите размер списка: ";

}

}

* При помощи функции FillList происходит заполнение списка с консоли.

void FillList(int N) {

char el;

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

cout << "Введите " << i + 1 << " элемент: ";

cin >> el;

List.push\_back(el);

}

};

* При помощи функции AddElement происходит добавления нового элемента перед элементом с указанным индексом. Программа запрашивает у пользователя номер элемента, перед которым ставится новый элемент, а затем с клавиатуры вводится новый элемент. При помощи итератора advance указатель доходит до введённого номера элемента. Новый элемент подставляется при помощи итератора insert.

void AddElement(int size){

char key;

int N = -1;

while ((N < 1) || (N > size)) {

if ((N < 1) || (N > size)) {

cout << "\nВведите номер элемента: ";

}

cin >> N;

}

list<char> :: iterator p;

p = List.begin();

advance(p, N-1);

list <char> AddList;

cout << "Введите элемент: ";

cin >> key;

AddList.push\_back(key);

List.insert(p, AddList.begin(), AddList.end());

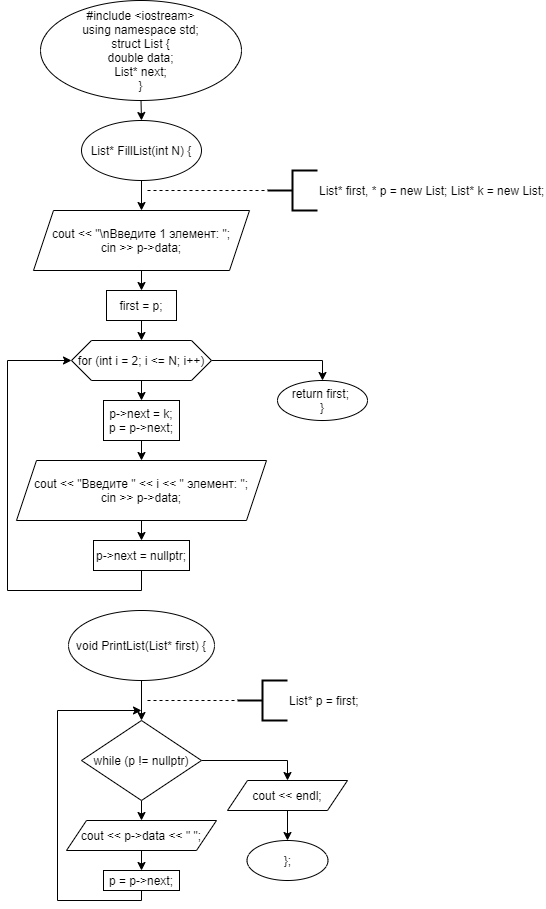
}

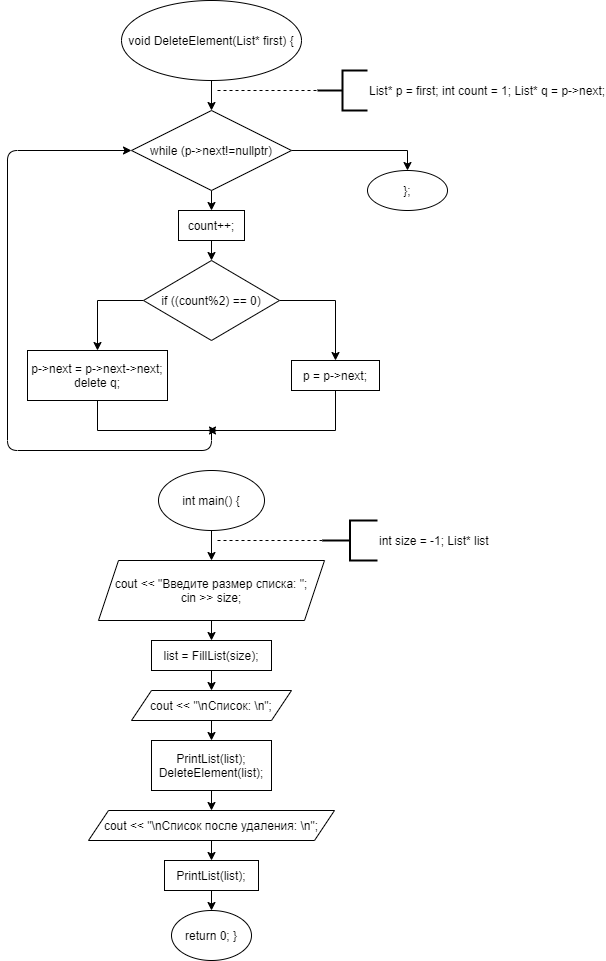
* При помощи итератора copy элементы списка выводятся на консоль.

copy(List.begin(), List.end(), ostream\_iterator<char>(cout, " "));

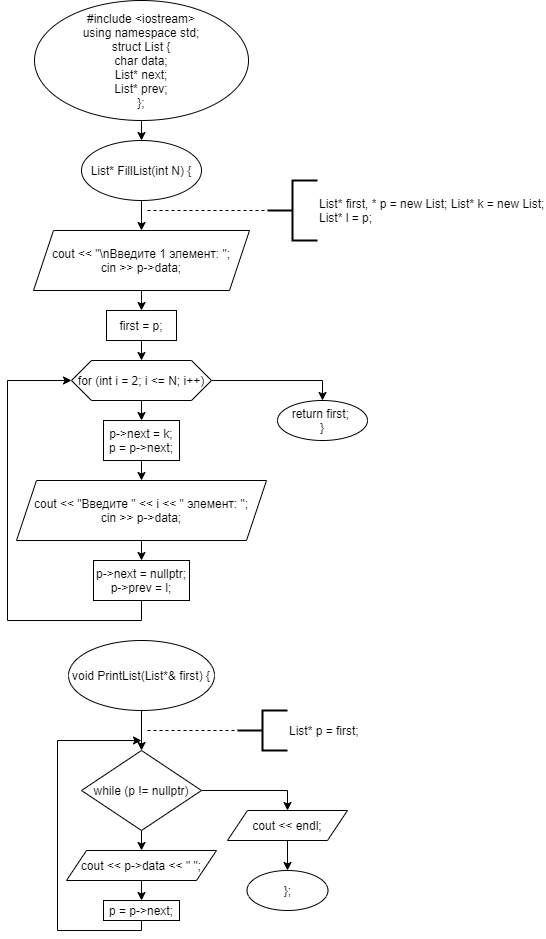
# **Блок схема**

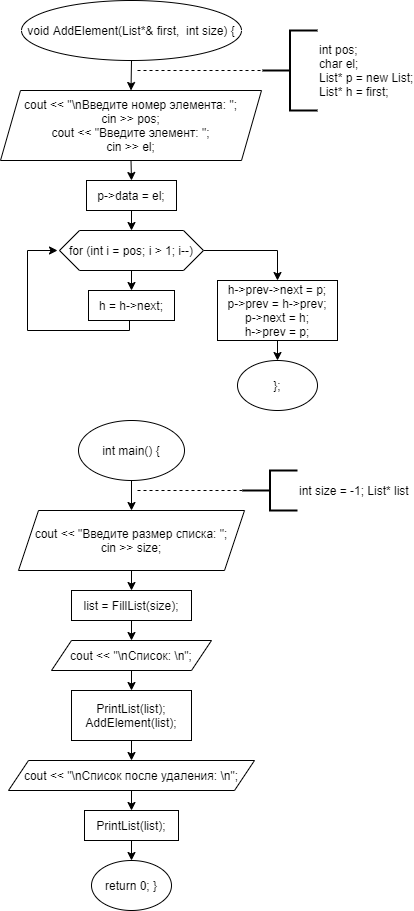
Однонаправленный список



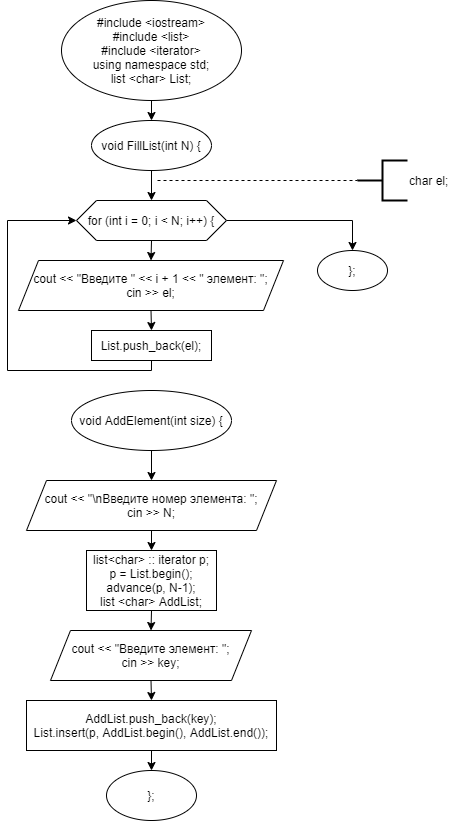


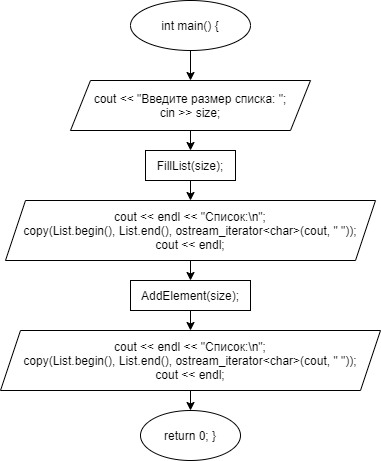
Двунаправленный список





Список через STL





# **Код программы на языке C++**

Однонаправленный список

#include <iostream>

using namespace std;

struct List {

double data;

List\* next;

};

List\* FillList(int N)

{

List\* first, \* p = new List;

cout << "\nВведите 1 элемент: ";

cin >> p->data;

first = p;

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

List\* k = new List;

p->next = k;

p = p->next;

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

cin >> p->data;

p->next = nullptr;

}

return first;

};

void PrintList(List\* first)

{

List\* p = first;

while (p != nullptr) {

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

p = p->next;

}

cout << endl;

};

void DeleteElement(List\* first)

{

List\* p = first;

int count = 1;

while (p->next!=nullptr) {

count++;

if ((count%2) == 0) {

List\* q = p->next;

p->next = p->next->next;

delete q;

}

else {

p = p->next;

}

}

};

int main()

{

system("chcp 1251");

int size = -1;

cout << "Введите размер списка: ";

while (size < 0) {

cin >> size;

if (size < 0) {

cout << "Введите размер списка: ";

}

}

List\* list = FillList(size);

cout << "\nСписок: \n";

PrintList(list);

DeleteElement(list);

cout << "\nСписок после удаления: \n";

PrintList(list);

system("pause");

return 0;

}

Двунаправленный список

#include <iostream>

using namespace std;

struct List {

char data;

List\* next;

List\* prev;

};

List\* FillList(int N)

{

List\* first, \* p = new List;

cout << "\nВведите 1 элемент: ";

cin >> p->data;

first = p;

p->next = nullptr;

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

List\* k = new List;

List\* l = p;

p->next = k;

p = p->next;

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

cin >> p->data;

p->next = nullptr;

p->prev = l;

}

return first;

};

void PrintList(List\*& first)

{

List\* p = first;

while (p != nullptr) {

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

p = p->next;

}

cout << endl;

};

void AddElement(List\*& first, int size) {

int pos;

cout << "\nВведите номер элемента: ";

do {

cin >> pos;

if ((pos > size) || (pos < 1)) {

cout << "Введите номер элемента: ";

}

} while (pos > size || pos < 1);

cout << "Введите элемент: ";

char el;

cin >> el;

List\* p = new List;

p->data = el;

List\* h = first;

for (int i = pos; i > 1; i--) {

h = h->next;

}

h->prev->next = p;

p->prev = h->prev;

p->next = h;

h->prev = p;

};

int main()

{

system("chcp 1251");

int size = -1;

cout << "Введите размер списка: ";

while (size < 0) {

cin >> size;

if (size < 0) {

cout << "Введите размер списка: ";

}

}

if (size == 0) {

cout << "Список пуст!\n";

}

else {

List\* list = FillList(size);

cout << "\nСписок: \n";

PrintList(list);

AddElement(list,size);

cout << "\nСписок после добавления: \n";

PrintList(list);

}

system("pause");

return 0;

}

Список через STL

#include <iostream>

#include <list>

#include <iterator>

using namespace std;

list <char> List;

void FillList(int N) {

char el;

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

cout << "Введите " << i + 1 << " элемент: ";

cin >> el;

List.push\_back(el);

}

};

void AddElement(int size){

char key;

int N = -1;

while ((N < 1) || (N > size)) {

if ((N < 1) || (N > size)) {

cout << "\nВведите номер элемента: ";

}

cin >> N;

}

list<char> :: iterator p;

p = List.begin();

advance(p, N-1);

list <char> AddList;

cout << "Введите элемент: ";

cin >> key;

AddList.push\_back(key);

List.insert(p, AddList.begin(), AddList.end());

}

int main()

{

system("chcp 1251");

int size = -1;

while (size < 0) {

if (size < 0) {

cout << "Введите размер списка: ";

}

cin >> size;

}

cout << endl;

if (size == 0) {

cout << "Список пуст!\n";

}

else {

FillList(size);

cout << endl << "Список:\n";

copy(List.begin(), List.end(), ostream\_iterator<char>(cout, " "));

cout << endl;

AddElement(size);

cout << endl << "Список:\n";

copy(List.begin(), List.end(), ostream\_iterator<char>(cout, " "));

cout << endl;

}

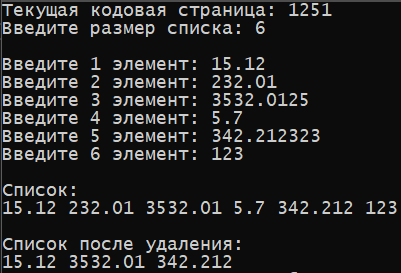
system("pause");

return 0;

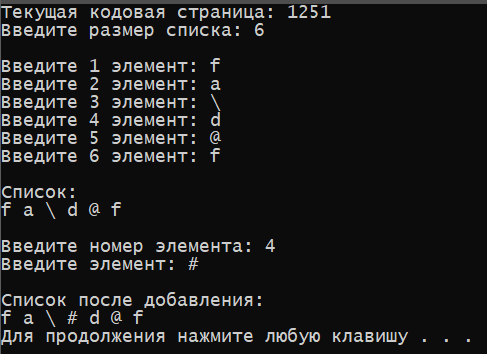
}

# **Скриншоты тестов**

Однонаправленный список



Двунаправленный список



Список через STL

