МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «Полоцкий государственный университет»

Факультет информационных технологий

Кафедра технологий программирования

**Отчёт по лабораторной работе № 1 по курсу «Алгоритмы и структуры данных»**

«Реализация линейной структуры данных «Список» и основные

алгоритмы обработки.»

Вариант №6

ВЫПОЛНИЛ студент группы 21-ИТ-1

Макеёнок Д.И.

ПРОВЕРИЛ преподаватель

Виноградова А.Д.

Полоцк, 2022 г.

**Цель работы:** ознакомиться с основами линейной структуры данных «Список», изучить основные алгоритмы обработки ЛСД «Список», научиться применять полученные знания на практике.

**Теоретические сведения:**

1. Определение понятия список.

Связный список является простейшим типом данных динамической

структуры, состоящей из элементов (узлов). Каждый узел включает в себя в

классическом варианте два поля:

• данные (в качестве данных может выступать переменная, объект

класса или структуры и т. д.)

• указатель на следующий узел в списке.

Элементы связанного списка можно помещать и исключать произвольным

образом.

1. Классификация списков по количеству полей указателей.

По количеству полей указателей различают однонаправленный(односвязный) и двунаправленный (двусвязный) списки. Связный список, содержащий только один указатель на следующий элемент, называется односвязным. Связный список, содержащий два поля указателя – на следующий элемент и на предыдущий, называется двусвязным.

1. Классификация списков по способу связи элементов.

различают линейные и циклические списки. Связный список, в котором, последний элемент указывает на NULL, называется линейным. Связный список, в котором последний элемент связан с первым, называется циклическим.

1. Виды списков. Отличие списков.

Односвязный линейный список (ОЛС): Каждый узел ОЛС содержит 1

поле указателя на следующий узел.

Односвязный циклический список (ОЦС):

Каждый узел ОЦС содержит 1 поле указателя на следующий узел.

Двусвязный линейный список (ДЛС): Каждый узел ДЛС содержит

два поля указателей: на следующий и на предыдущий узел.

Двусвязный циклический список (ДЦС):

Каждый узел ДЦС содержит два поля указателей:

на следующий и на предыдущий узел.

1. Основные действия, производимые над элементами односвязного

линейного списка.

Основные действия, производимые над элементами ОЛС:

• Инициализация списка

• Добавление узла в список

• Удаление узла из списка

• Удаление корня списка

• Вывод элементов списка

• Взаимообмен двух узлов списка

1. Какие аргументы принимает функция добавления узла в список

(ОЛС).

Функция добавления узла в список принимает два аргумента:

• Указатель на узел, после которого происходит добавление

• Данные для добавляемого узла.

1. Сколько полей указателей содержит каждый узел двусвязного

линейного списка?

Каждый узел двунаправленного (двусвязного) линейного списка (ДЛС)

содержит два поля указателей — на следующий и на предыдущий узлы.

1. Сколько полей указателей содержит каждый узел односвязного

линейного списка?

Каждый узел однонаправленного (односвязного) линейного списка (ОЛС)

содержит одно поле указателя на следующий узел.

1. Основные действия, производимые над элементами двусвязного

линейного списка.

Основные действия, производимые над узлами ДЛС:

• Инициализация списка

• Добавление узла в список

• Удаление узла из списка

• Удаление корня списка

• Вывод элементов списка

• Вывод элементов списка в обратном порядке

• Взаимообмен двух узлов списка

**Код программы:**

#include <stdlib.h>

#include <iostream>

using namespace std;

struct Node

{

int x;

Node \*Next, \*Prev;

};

class List

{

Node \*Head, \*Tail;

public:

List() :Head(NULL), Tail(NULL) {};

void Show();

void Add(int x);

void del();

};

void List::Add(int x)

{

Node \*temp = new Node; //Выделение памяти под новый элемент структуры

temp->Next = NULL; //Указываем, что изначально по следующему адресу пусто

temp->x = x; //Записываем значение в структуру

if (Head != NULL) //Если список не пуст

{

temp->Prev = Tail; //Указываем адрес на предыдущий элемент в соотв. поле

Tail->Next = temp; //Указываем адрес следующего за хвостом элемента

Tail = temp; //Меняем адрес хвоста

}

else

{

temp->Prev = NULL; //Предыдущий элемент указывает в пустоту

Head = Tail = temp; //Голова=Хвост=тот элемент, что сейчас добавили

}

}

void List::Show()

{

//ВЫВОДИМ СПИСОК С НАЧАЛА

Node \*temp = Head; //Временно указываем на адрес первого элемента

while (temp != NULL) //Пока не встретим пустое значение

{

cout << temp->x << " "; //Выводим каждое считанное значение на экран

temp = temp->Next; //Смена адреса на адрес следующего элемента

}

cout << "\n";

}

void List::del() {

Node \*temp = Head;

Node \*p, \*o;

while (temp != NULL) {

if (temp->x < 0) {

if (Head->x < 0) {

Head = Head->Next;

Head->Prev = NULL;

}

else if (Tail == temp && Tail->x < 0) {

Tail = Tail->Prev;

Tail->Next = NULL;

}

else {

p = temp;

p->Prev->Next = temp->Next;

p->Next= temp->Prev;

}

}

temp = temp->Next;

}

}

int main()

{

int x;

List lst; //Объявляем переменную, тип которой есть список

for (int i = 0; i < 100; i++) {

cin >> x;

lst.Add(x); //Добавляем в список элементы

}

system("CLS");

cout << "initial list:" << endl;

lst.Show();

lst.del();

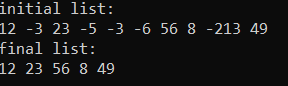
cout << "final list:" << endl;

lst.Show(); //Отображаем список на экране

return 0;

}

**Результат работы программы:**



**Вывод:** я ознакомился с основами линейной структуры данных «Список», изучил основные алгоритмы обработки ЛСД «Список», научился применять полученные знания на практике.