# ЛАБОРАТОРНАЯ РАБОТА № 27

# Обработка списков

#include <iostream>

using namespace std;

struct Item {

int info;

Item\* next;

};

int main()

{

Item \*first = 0; //Указатель на начало списка

Item \*p;

int i;

// Создать список

for (;;) {

// Вводить числа, пока не введем 0

cin >> i;

if (!i) break;

// Создать новый элемент списка

p = new Item;

p->info = i;

// Присоединить новый элемент к началу списка

p->next = first;

first = p;

}

// Пройти список и вывести элементы

p = first;

while (p) {

cout << p->info << " ";

p = p->next;

}

// Пройти список и удалить элементы

while (first) {

p = first;

first = first->next;

delete p;

}

return 0;

system("pause");

}

Другие виды списков:

* список с указателями на первый и последний элементы — позволяет добавлять и удалять элементы с обеих сторон списка;
* двунаправленный список — позволяет перемещаться по списку в обоих направлениях;
* кольцевой список — позволяет достичь любой элемент списка,  
  начав движение с любого места в списке.

Двусвязный список это список, который основан на односвязном, только адресных полей в нем два.

#include <iostream>

using namespace std;

struct Node //Структура являющаяся звеном списка

{

int x; //Значение x будет передаваться в список

Node \*Next, \*Prev; //Указатели на адреса следующего и предыдущего элементов списка

};

class List //Создаем тип данных Список

{

Node \*Head, \*Tail; //Указатели на адреса начала списка и его конца

public:

List() :Head(NULL), Tail(NULL) {}; //Инициализируем адреса как пустые

~List(); //Деструктор

void Show(); //Функция отображения списка на экране

void Add(int x); //Функция добавления элементов в список

};

List::~List() //Деструктор

{

while (Head) //Пока по адресу на начало списка что-то есть

{

Tail = Head->Next; //Резервная копия адреса следующего звена списка

delete Head; //Очистка памяти от первого звена

Head = Tail; //Смена адреса начала на адрес следующего элемента

}

}

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 //Если список пустой

{

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

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

}

}

void List::Show()

{

//ВЫВОДИМ СПИСОК С КОНЦА

Node \*temp = Tail;

//Временный указатель на адрес последнего элемента

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

{

cout << temp->x << " "; //Выводить значение на экран

temp = temp->Prev; //Указываем, что нужен адрес предыдущего элемента

}

cout << "\n";

//ВЫВОДИМ СПИСОК С НАЧАЛА

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";

}

int main()

{

system("CLS");

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

lst.Add(100); //Добавляем в список элементы

lst.Add(200);

lst.Add(900);

lst.Add(888);

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

system("PAUSE");

}

# ЗАДАНИЕ

### Задание 1. Однонаправленные списки

Написать программу по созданию, просмотру, добавлению и решению поставленной задачи для однонаправленного линейного списка (стек и/или очередь).

1. Создать список из случайных чисел, вычислить среднее арифметическое и заменить им первый элемент.

### Задание 2. Двунаправленные списки

Написать программу по созданию, просмотру, добавлению и решению поставленной задачи для двунаправленного линейного списка.

1. Создать список из случайных целых чисел. Удалить из списка элементы с повторяющимися более одного раза значениями.