**Курс «Объектно-ориентированное программирование на C++»**

**Встреча №15**

Тема: динамические структуры данных – односвязный список

Задания для самостоятельной работы:

**Задание №1.** Создайте шаблонный класс односвязного списка для работы с целыми значениями. Требуется создать реализации для типичных операций над элементами:

* AddToHead – добавление элемента в голову
* AddToTail – добавление элемента в хвост
* DeleteFromHead – удаление элемента из головы
* DeleteFromTail – удаление элемента из хвоста
* DeleteAll – удаление всех элементов
* Show – отображение всех элементов списка на экран

**Задание №2.**  Добавить в класс из задания 1 следующие функции: вставка элемента в заданную позицию, удаление элемента по заданной позиции, поиск заданного элемента (функция возвращает позицию найденного элемента в случае успеха или NULL в случае неудачи), поиск и замена заданного элемента (функция возвращает количество замененных элементов в случае успеха или -1 в случае неудачи), переворот списка.

#include<windows.h>

#include<iostream>

using namespace std;

template<typename T>

struct Node

{

T info;

Node\* Next;

};

template<typename T>

class List

{

private:

Node\* head;

Node\* tail;

int count;

public:

List()

{

head = head = NULL;

count = 0;

}

void AddToTail(int info)

{

Node\* temp = new Node;

temp->info = info;

temp->Next = NULL;

if (head == NULL)

{

head = tail = temp;

}

else

{

tail->Next = temp;

tail = temp;

}

count++;

}

void AddToHead(int data)

{

Node\* temp = new Node;

temp->info = data;

temp->Next = head;

head = temp;

count++;

}

void DelFromTail()

{

Node\* temp;

temp = head;

for (int i = 0; i < count - 1; i++)

{

temp = temp->Next;

}

delete tail;

tail = temp;

temp->Next = NULL;

count--;

}

void DelFromHead()

{

Node\* temp = head;

head = head->Next;

delete temp;

count--;

}

void DeleteAll()

{

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

{

Node\* temp = head;

head = head->Next;

delete temp;

}

count = 0;

}

void getcount()

{

cout << count << endl;

}

void show()

{

cout << endl;

Node\* temp = head;

if (count != 0)

{

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

{

cout << temp->info << " ";

temp = temp->Next;

}

}

else

cout << "asdasd" << endl;

cout << endl;

}

void addinpos(int index, int num)

{

Node\* temp1 = head;

Node\* temp2 = new Node;

if (index == 0)

{

AddToHead(num);

count++;

}

else

{

for (int i = 0; i < index - 1; i++)

{

temp1 = temp1->Next;

}

temp2->info = num;

temp2->Next = temp1->Next;

temp1->Next = temp2;

count++;

}

}

void DellInPos(int index)

{

Node \* temp1 = head;

Node\* temp2 = head;

if (index <= -1 || index > count - 1)

{

cout << "NULL" << endl;

}

else

{

for (int i = 0; i < index - 1; i++)

{

temp2 = temp2->Next;

}

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

{

temp1 = temp1->Next;

}

temp2->Next = temp1->Next;

delete temp1;

count--;

}

}

int search(int num)

{

Node\* temp = head;

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

{

if (num == temp->info)

{

return i;

}

temp = temp->Next;

}

return NULL;

}

int replace(int find, int num)

{

int counter = 0;

Node\* temp1 = head;

Node\* temp2 = head;

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

{

Node\* replace = new Node;

replace->info = num;

if (temp1->Next != NULL)

{

if (temp1->Next->info == find)

{

temp2 = temp2->Next;

temp1->Next = replace;

replace->Next = temp2->Next;

counter++;

temp2 = temp1;

}

temp1 = temp1->Next;

temp2 = temp2->Next;

}

}

if (count > 0)

return counter;

else return -1;

}

void reverse()

{

Node\* temp1 = head;

Node\* temp2 = head;

Node\* newhead = tail;

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

{

temp1 = head;

temp2 = head;

for (int j = 0; j < count - i - 1; j++)

{

temp1 = temp1->Next;

}

for (int k = 0; k < count - i - 2; k++)

{

temp2 = temp2->Next;

}

temp1->Next = temp2;

}

temp1->Next = NULL;

tail = temp1;

head = newhead;

}

};

void main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

List<int> a;

for (int i = 0; i < 10; i++)

{

a.AddToTail(rand() % 100);

}

//cout << a.search(0) << endl;

//a.addinpos(5, 33);

//a.DellInPos(3);

//a.addinpos(8, 0);

a.show();

//a.replace(0, 212121);

a.reverse();

a.show();

}