

Курс: Объектно-ориентированное программирование на C++»

Встреча №15

ТЕМА: ДИНАМИЧЕСКИЕ СТРУКТУРЫ ДАННЫХ – ОДНОСВЯЗНЫЙ СПИСОК



# Задание №1

Создайте шаблонный класс односвязного списка для работы с целыми значениями. Требуется создать реализации для типич- ных 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 пози- цию найденного элемента в случае успеха или NULL в случае не- удачи), 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;

struct Node

{

int val;

Node\* next;

};

class List

{

private:

Node\* Head;

Node\* Tail;

int count;

public:

List()

{

Head = Tail = NULL;

count = 0;

}

void AddToTail(int data)

{

Node\* temp = new Node;

temp->val = data;

temp->next = NULL;

if (Head != NULL)

{

Tail->next = temp;

Tail = temp;

}

else

{

Head = temp;

Tail = temp;

}

count++;

}

void AddInside(int data, int index)

{

if (index <= 0)

{

AddToHead(data);

return;

}

Node\* new\_node = new Node;

new\_node->val = data;

new\_node->next = NULL;

if (Head != NULL)

{

int x = 1;

Node\* tmp = Head;

while (tmp->next != NULL && x < index - 1)

{

tmp = tmp->next;

x++;

}

new\_node->next = tmp->next;

tmp->next = new\_node;

}

else

{

Head = new\_node;

Tail = new\_node;

}

count++;

}

void AddToHead(int data)

{

Node\* temp = new Node;

temp->val = data;

temp->next = NULL;

if (Head != NULL)

{

temp->next = Head;

Head = temp;

}

else

{

Head = temp;

Tail = temp;

}

count++;

}

Node\* LookingFor(int chislo)

{

Node\* tmp = Head;

if (tmp->val == chislo)

return tmp;

while (tmp != NULL)

{

tmp = tmp->next;

if (tmp == NULL)

break;

if (tmp->val == chislo)

return tmp;

}

tmp = NULL;

return tmp;

}

int RedactElement(int chislo, int new\_chislo)

{

int k = 0;

bool b = false;

Node\* tmp = Head;

if (tmp->val == chislo)

{

tmp->val = new\_chislo;

b = true;

k++;

}

while (tmp != NULL)

{

tmp = tmp->next;

if (tmp == NULL)

break;

if (tmp->val == chislo)

{

tmp->val = new\_chislo;

b = true;

k++;

}

}

if (!b)

return -1;

else

return k;

}

void SwapList()

{

List tmp;

Node\* index = Head;

while (index != NULL)

{

tmp.AddToHead(index->val);

index = index->next;

}

this->DeleteAll();

this->Head = tmp.Head;

this->Tail = tmp.Tail;

this->count = tmp.count;

}

void DeleteFromHead()

{

Node\* tmp = Head;

Head = Head->next;

delete tmp;

count--;

}

void DeleteFromTail()

{

Node\* end=Tail;

if (Head == NULL)

return;

Node\* tmp = Head;

while (tmp->next->next != NULL)

{

tmp = tmp->next;

}

Tail = tmp;

tmp->next = NULL;

delete end;

count--;

}

void DeleteAll()

{

while (Head != NULL)

{

DeleteFromHead();

}

}

void DeleteInside(int index)

{

if (index > count||index<=0)

{

cout << "Ви вийшли за межі списку!\n";

return;

}

if (index == 1)

{

DeleteFromHead();

return;

}

Node\* new\_node = new Node;

new\_node->next = NULL;

Node\* tmp = Head;

int x = 1;

while (tmp->next != NULL && x < index - 1)

{

tmp = tmp->next;

x++;

}

new\_node = tmp->next;

tmp->next = tmp->next->next;

delete new\_node;

}

void Show()

{

Node\* temp = Head;

while (temp != NULL)

{

cout << temp->val << endl;

temp = temp->next;

}

cout << endl << endl;

}

};

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

List lst;

lst.AddToTail(100);

lst.AddToTail(120);

lst.AddToTail(70);

lst.AddToTail(45);

lst.Show();

/\*if (lst.LookingFor(100) != NULL)

cout << "YES!\n";

else

cout << "NO!\n";\*/

/\*cout << lst.RedactElement(70, 1) << endl<<endl<<endl;\*/

lst.SwapList();

lst.Show();

system("pause");

return 0;

}