### Постановка задачи.

Реализовать стек на основе связного списка на с++.

### Введение.

Стек - это абстрактный тип данных, который представляет собой список элементов, организованных по принципу LIFO (Last In First Out). Это означает, что последний добавленный элемент будет первым удаленным.

Существует несколько методов для работы со стеком. Например, push используется для добавления элемента в стек, а pop используется для удаления элемента из стека. [Еще один метод peek используется для получения значения верхнего элемента стека без его удаления](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B5%D0%BA).

### Решение

#include <iostream>

using namespace std;

class LinkedListNode {

public:

int data;

LinkedListNode\* next;

};

class LinkedList {

public:

LinkedListNode\* head;

LinkedList() {

head = NULL;

}

// Добавление элемента в конец стека

void AddElementAtEnd(int x) {

LinkedListNode\* current = new LinkedListNode;

current->data = x;

current->next = NULL;

if (head == NULL) {

head = current;

return;

}

LinkedListNode\* temp = head;

while (temp->next != NULL) {

temp = temp->next;

}

temp->next = current;

}

//Данный метод необходимо немного изменить, чтобы вместо удаление последнего элемента он возвращал его.

// Возврат последнего элемента стека

int peackLastElement() {

if (head == NULL) {

cout << "List is empty" << endl;

return 0;

}

if (head->next == NULL) {

return head->data;

}

LinkedListNode\* temp1 = head;

while (temp1->next != NULL) {

temp1 = temp1->next;

}

return temp1->data;

}

//Данный метод необходимо немного изменить, чтобы вместе с удалением последнего элемента он возвращал его.

// Возврат последнего элемента стека

int RemoveLastElement() {

if (head == NULL) {

cout << "List is empty" << endl;

return 0;

}

if (head->next == NULL) {

int toReturn = head->data;

delete head;

head = NULL;

return toReturn;

}

LinkedListNode\* temp1 = head;

while (temp1->next != NULL) {

temp1 = temp1->next;

}

LinkedListNode\* temp2 = head;

while (temp2->next != temp1) {

temp2 = temp2->next;

}

int toReturn = temp1->data;

delete temp1;

temp2->next = NULL;

return toReturn;

}

//Печать стека

void PrintList() {

if (head == NULL) {

cout << "List is empty" << endl;

return;

}

else {

LinkedListNode\* current = head;

while (current != NULL) {

cout << current->data << " ";

current = current->next;

}

cout << endl;

}

}

};

class stack {

public:

LinkedList listInsideStack;

//Вставка в конец

void push(int x) {

listInsideStack.AddElementAtEnd(x);

}

//Удаланение и получение последнего элемента.

int pop() {

return listInsideStack.RemoveLastElement();

}

int peack() {

return listInsideStack.peackLastElement();

}

//Печать очереди

void printStack() {

listInsideStack.PrintList();

}

};

int main() {

// Создание объекта очереди

stack exmapleStack;

// Добавление элементов в конец очереди

exmapleStack.push(1);

exmapleStack.push(2);

exmapleStack.push(3);

exmapleStack.push(4);

//Печать стека

exmapleStack.printStack();

// Удаление элементов из конца стека

exmapleStack.pop();

// Удаление элементов из начала стека с получение последнего элемента.

cout << exmapleStack.pop() << endl;

//Получение последнего элемента без его удаления

cout << exmapleStack.peack()<<endl;

//Печать очереди

exmapleStack.printStack();

}

### Заключение.

### C:\Users\Максим\Downloads\2023-06-07_10-24-12.png

В результате выполнения данной задачи, я успешно реализовала стек на основе массива на языке программирования C++. В результате получилось эффективное решение, которое можно использовать в приложениях, где нужно хранить и обрабатывать большие объемы данных.