**Постановка задачи.**

Реализовать связный список на с++.

**Введение.**

Связный список - это базовая динамическая структура данных в информатике, состоящая из узлов, содержащих данные и ссылки на следующий и/или предыдущий узел списка. Принципиальным отличием связного списка от массива является то, что порядок элементов в связном списке не обязательно совпадает с порядком расположения элементов в памяти компьютера.

Связный список обладает следующими методами:

* добавление элемента в начало списка;
* добавление элемента в конец списка;
* добавление элемента после заданного элемента;
* удаление первого элемента списка;
* удаление последнего элемента списка;
* удаление заданного элемента списка.

**Решение.**

#include <iostream>

using namespace std;

class LinkedListNode {

public:

int data;

LinkedListNode\* next;

};

class LinkedList {

public:

LinkedListNode\* head;

LinkedList() {

head = NULL;

}

// Добавление элемента в начало списка

void AddElementAtFront(int x) {

LinkedListNode\* current = new LinkedListNode;

current->data = x;

current->next = head;

head = current;

}

// Добавление элемента в конец списка

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;

}

// Добавление элемента после заданного элемента

void AddElementAfter(int x, int y) {

LinkedListNode\* current = new LinkedListNode;

current->data = y;

if (head == NULL) {

head = current;

return;

}

LinkedListNode\* temp = head;

while (temp != NULL && temp->data != x) {

temp = temp->next;

}

if (temp == NULL) {

cout << "Element not found" << endl;

return;

}

current->next = temp->next;

temp->next = current;

}

// Удаление первого элемента списка

void RemoveFirstElement() {

if (head == NULL) {

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

return;

}

LinkedListNode\* temp = head;

head = head->next;

delete temp;

}

// Удаление последнего элемента списка

void RemoveLastElement() {

if (head == NULL) {

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

return;

}

if (head->next == NULL) {

delete head;

head = NULL;

return;

}

LinkedListNode\* temp1 = head;

while (temp1->next != NULL) {

temp1 = temp1->next;

}

LinkedListNode\* temp2 = head;

while (temp2->next != temp1) {

temp2 = temp2->next;

}

delete temp1;

temp2->next = NULL;

}

// Удаление заданного элемента списка

void RemoveElement(int x) {

if (head == NULL) {

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

return;

}

if (head->data == x) {

RemoveFirstElement();

return;

}

LinkedListNode\* temp1 = head;

while (temp1 != NULL && temp1->data != x) {

temp1 = temp1->next;

}

if (temp1 == NULL) {

cout << "Element not found" << endl;

return;

}

else {

LinkedListNode\* temp2 = head;

while (temp2->next != temp1) {

temp2 = temp2->next;

}

temp2->next = temp1->next;

delete(temp1);

}

}

// Печать списка

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;

}

}

};

int main() {

// Создание объекта связного списка

LinkedList list;

// Добавление элементов в начало списка

list.AddElementAtFront(2);

list.AddElementAtFront(1);

// Добавление элементов в конец списка

list.AddElementAtEnd(4);

list.AddElementAtEnd(5);

// Добавление элемента после заданного элемента

list.AddElementAfter(2, 3);

//Печать списка

list.PrintList();

// Удаление первого элемента списка

list.RemoveFirstElement();

// Удаление последнего элемента списка

list.RemoveLastElement();

// Удаление заданного элемента списка

list.RemoveElement(2);

//Печать списка

list.PrintList();

}

C:\Users\Максим\Downloads\2023-06-06_17-22-04.png

**Заключение.**

В результате работы была создана структура данных – связный список. Были реализованы основные функции для работы со списком, такие как добавление новых элементов в список, удаление элементов из списка и поиск элементов в списке. Связный список на С++ является мощным инструментом для решения различных задач и может быть использован во многих областях, где требуется динамическая структура данных.