**THỰC HÀNH CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT 2025**

**Cài các tác vụ danh sách liên kết đơn**

Tên : Trịnh Tâm Như

Mssv: 3123411215

Code

#include <iostream>

using namespace std;

// Định nghĩa cấu trúc Node cho danh sách liên kết

struct Node {

int info; // Giá trị của nút

Node\* next; // Con trỏ tới nút tiếp theo

};

// Hàm khởi tạo danh sách liên kết rỗng

void Init(Node\*& pHead) {

pHead = NULL; // Gán con trỏ đầu danh sách là NULL, nghĩa là danh sách rỗng

}

// Kiểm tra xem danh sách có rỗng không

bool IsEmpty(Node\* pHead) {

return (pHead == NULL); // Nếu con trỏ đầu là NULL thì danh sách rỗng

}

// Hàm tạo một nút mới

Node\* CreateNode(int X) {

Node\* p = new Node; // Cấp phát bộ nhớ cho nút mới

p->info = X; // Gán giá trị cho nút mới

p->next = NULL; // Con trỏ next của nút này trỏ đến NULL (nút cuối cùng)

return p; // Trả về con trỏ tới nút mới

}

// Hàm tìm kiếm phần tử trong danh sách

Node\* Find(Node\* pHead, int X) {

Node\* temp = pHead;

while (temp != NULL) { // Duyệt qua tất cả các phần tử trong danh sách

if (temp->info == X) { // Nếu tìm thấy phần tử với giá trị X

return temp; // Trả về con trỏ tới phần tử đó

}

temp = temp->next; // Tiến tới phần tử kế tiếp

}

return NULL; // Nếu không tìm thấy phần tử

}

// Hàm tìm kiếm phần tử trong danh sách theo thứ tự

Node\* FindOrder(Node\* pHead, int X) {

Node\* temp = pHead;

int index = 0; // Biến chỉ mục để theo dõi thứ tự phần tử trong danh sách

while (temp != NULL) { // Duyệt qua danh sách

if (temp->info == X) { // Nếu tìm thấy phần tử với giá trị X

cout << "Phan tu " << X << " o vi tri " << index << " trong danh sach." << endl;

return temp; // Trả về con trỏ tới phần tử đó

}

temp = temp->next; // Tiến tới phần tử kế tiếp

index++; // Tăng chỉ mục

}

cout << "Khong tim thay phan tu co gia tri " << X << " trong danh sach." << endl;

return NULL; // Nếu không tìm thấy phần tử

}

// Thêm phần tử vào đầu danh sách

void InsertFirst(Node\*& pHead, int X) {

Node\* newNode = CreateNode(X); // Tạo nút mới

newNode->next = pHead; // Nút mới trỏ tới nút đầu của danh sách cũ

pHead = newNode; // Cập nhật con trỏ đầu danh sách trỏ tới nút mới

}

// Thêm phần tử vào sau một phần tử cụ thể

void InsertAfter(Node\* pHead, int X, int afterX) {

Node\* temp = Find(pHead, afterX); // Tìm phần tử có giá trị là afterX

if (temp != NULL) { // Nếu tìm thấy phần tử

Node\* newNode = CreateNode(X); // Tạo nút mới

newNode->next = temp->next; // Nút mới trỏ tới nút tiếp theo của phần tử tìm thấy

temp->next = newNode; // Nút tìm thấy trỏ tới nút mới

}

}

// Thêm phần tử vào danh sách sao cho danh sách vẫn được sắp xếp theo thứ tự

void InsertOrder(Node\*& pHead, int X) {

Node\* newNode = CreateNode(X); // Tạo nút mới

if (IsEmpty(pHead) || pHead->info >= X) { // Nếu danh sách rỗng hoặc giá trị nút đầu lớn hơn hoặc bằng X

newNode->next = pHead; // Nút mới trỏ tới nút đầu hiện tại

pHead = newNode; // Cập nhật con trỏ đầu danh sách

}

else {

Node\* temp = pHead;

while (temp->next != NULL && temp->next->info < X) { // Duyệt qua danh sách để tìm vị trí thích hợp

temp = temp->next;

}

newNode->next = temp->next; // Nút mới trỏ tới nút tiếp theo của nút tìm thấy

temp->next = newNode; // Nút tìm thấy trỏ tới nút mới

}

}

// Xóa phần tử đầu tiên trong danh sách

void DeleteFirst(Node\*& pHead) {

if (pHead != NULL) { // Nếu danh sách không rỗng

Node\* temp = pHead; // Lưu con trỏ đầu

pHead = pHead->next; // Cập nhật con trỏ đầu danh sách

delete temp; // Xóa nút đầu

}

}

// Xóa phần tử sau một phần tử cụ thể

void DeleteAfter(Node\* pHead, int afterX) {

Node\* temp = Find(pHead, afterX); // Tìm phần tử sau phần tử có giá trị afterX

if (temp != NULL && temp->next != NULL) { // Nếu tìm thấy và có nút tiếp theo

Node\* deleteNode = temp->next; // Lưu nút cần xóa

temp->next = deleteNode->next; // Nút tìm thấy trỏ tới nút sau nút cần xóa

delete deleteNode; // Xóa nút

}

}

// Xóa một phần tử cụ thể

void Remove(Node\*& pHead, int X) {

if (pHead == NULL) return; // Nếu danh sách rỗng thì không làm gì

if (pHead->info == X) { // Nếu phần tử đầu tiên cần xóa

DeleteFirst(pHead);

return;

}

Node\* temp = pHead;

while (temp->next != NULL && temp->next->info != X) { // Duyệt danh sách để tìm phần tử cần xóa

temp = temp->next;

}

if (temp->next != NULL) { // Nếu tìm thấy phần tử cần xóa

Node\* deleteNode = temp->next; // Lưu nút cần xóa

temp->next = deleteNode->next; // Nút trước trỏ tới nút sau nút cần xóa

delete deleteNode; // Xóa nút

}

}

// Xóa phần tử theo thứ tự

void RemoveOrder(Node\*& pHead, int X) {

if (pHead == NULL) return; // Nếu danh sách rỗng thì không làm gì

Node\* temp = pHead;

Node\* prev = NULL;

while (temp != NULL && temp->info != X) { // Duyệt danh sách để tìm phần tử cần xóa

prev = temp;

temp = temp->next;

}

if (temp != NULL) { // Nếu tìm thấy phần tử cần xóa

if (prev == NULL) { // Nếu phần tử cần xóa là phần tử đầu

pHead = temp->next; // Cập nhật con trỏ đầu

}

else {

prev->next = temp->next; // Cập nhật con trỏ của phần tử trước trỏ tới phần tử sau nút cần xóa

}

delete temp; // Xóa phần tử

}

}

// Duyệt danh sách và in ra các phần tử

void ShowList(Node\* pHead) {

Node\* temp = pHead;

while (temp != NULL) { // Duyệt qua danh sách

cout << temp->info << " -> "; // In giá trị của phần tử hiện tại

temp = temp->next; // Tiến tới phần tử kế tiếp

}

cout << "NULL" << endl; // In ra kết thúc danh sách

}

// Hủy bỏ danh sách (giải phóng bộ nhớ)

void ClearList(Node\*& pHead) {

Node\* temp;

while (pHead != NULL) { // Duyệt qua danh sách và giải phóng bộ nhớ cho từng nút

temp = pHead;

pHead = pHead->next; // Cập nhật con trỏ đầu danh sách

delete temp; // Xóa nút hiện tại

}

}

// Sắp xếp danh sách theo thứ tự tăng dần (Selection Sort)

void SelectionSort(Node\*& pHead) {

Node\* i, \* j;

int temp;

for (i = pHead; i != NULL; i = i->next) { // Duyệt qua danh sách

for (j = i->next; j != NULL; j = j->next) { // Duyệt qua các phần tử sau phần tử i

if (i->info > j->info) { // Nếu giá trị của phần tử i lớn hơn giá trị của phần tử j

temp = i->info;

i->info = j->info;

j->info = temp; // Hoán đổi giá trị của phần tử i và j

}

}

}

}

int main() {

Node\* head;

Init(head); // Khởi tạo danh sách liên kết rỗng

// Thêm các phần tử vào danh sách

InsertFirst(head, 10);

InsertFirst(head, 5);

InsertFirst(head, 7);

InsertFirst(head, 2);

InsertFirst(head, 8);

InsertFirst(head, 9);

InsertFirst(head, 1);

// Duyệt danh sách

cout << "Danh sach sau khi them phan tu: ";

ShowList(head);

// Xóa phần tử đầu tiên

DeleteFirst(head);

cout << "Danh sach sau khi xoa phan tu dau tien: ";

ShowList(head);

// Xóa một phần tử

Remove(head, 7);

cout << "Danh sach sau khi xoa phan tu 7: ";

ShowList(head);

// Sắp xếp danh sách

SelectionSort(head);

cout << "Danh sach sau khi sap xep: ";

ShowList(head);

//Xóa sau khi đã sắp xếp

RemoveOrder(head, 8);

cout << "Danh sach sau khi xoa trong danh sach da duoc sap xep: ";

ShowList(head);

// Tìm kiếm phần tử không có trong danh sách

int searchVal = 20; // Giá trị tìm kiếm

Node\* result = Find(head, searchVal);

if (result != NULL) {

cout << "Phan tu co gia tri " << searchVal << " da duoc tim thay trong danh sach." << endl;

}

else {

cout << "Khong tim thay phan tu co gia tri " << searchVal << " trong danh sach." << endl;

}

// Xóa danh sách

ClearList(head);

cout << "Danh sach sau khi xoa: ";

ShowList(head);

return 0;

}

TEST CASE CODE

#include <iostream>

using namespace std;

// Định nghĩa cấu trúc Node cho danh sách liên kết

struct Node {

int info; // Giá trị của nút

Node\* next; // Con trỏ tới nút tiếp theo

};

// Hàm khởi tạo danh sách liên kết rỗng

void Init(Node\*& pHead) {

pHead = NULL; // Gán con trỏ đầu danh sách là NULL, nghĩa là danh sách rỗng

}

// Kiểm tra xem danh sách có rỗng không

bool IsEmpty(Node\* pHead) {

return (pHead == NULL); // Nếu con trỏ đầu là NULL thì danh sách rỗng

}

// Hàm tạo một nút mới

Node\* CreateNode(int X) {

Node\* p = new Node; // Cấp phát bộ nhớ cho nút mới

p->info = X; // Gán giá trị cho nút mới

p->next = NULL; // Con trỏ next của nút này trỏ đến NULL (nút cuối cùng)

return p; // Trả về con trỏ tới nút mới

}

// Hàm thêm phần tử vào đầu danh sách

void InsertFirst(Node\*& pHead, int X) {

Node\* newNode = CreateNode(X); // Tạo nút mới

newNode->next = pHead; // Nút mới trỏ tới nút đầu danh sách cũ

pHead = newNode; // Cập nhật con trỏ đầu danh sách trỏ tới nút mới

}

// Hàm xóa phần tử đầu tiên trong danh sách

void DeleteFirst(Node\*& pHead) {

if (pHead != NULL) { // Nếu danh sách không rỗng

Node\* temp = pHead; // Lưu con trỏ đầu

pHead = pHead->next; // Cập nhật con trỏ đầu danh sách

delete temp; // Xóa nút đầu

}

}

// Hàm xóa phần tử cụ thể

void Remove(Node\*& pHead, int X) {

if (pHead == NULL) return; // Nếu danh sách rỗng thì không làm gì

if (pHead->info == X) { // Nếu phần tử đầu tiên cần xóa

DeleteFirst(pHead);

return;

}

Node\* temp = pHead;

while (temp->next != NULL && temp->next->info != X) { // Duyệt danh sách để tìm phần tử cần xóa

temp = temp->next;

}

if (temp->next != NULL) { // Nếu tìm thấy phần tử cần xóa

Node\* deleteNode = temp->next; // Lưu nút cần xóa

temp->next = deleteNode->next; // Nút trước trỏ tới nút sau nút cần xóa

delete deleteNode; // Xóa nút

}

}

// Hàm sắp xếp danh sách theo thứ tự tăng dần (Selection Sort)

void SelectionSort(Node\*& pHead) {

Node\* i, \* j;

int temp;

for (i = pHead; i != NULL; i = i->next) { // Duyệt qua danh sách

for (j = i->next; j != NULL; j = j->next) { // Duyệt qua các phần tử sau phần tử i

if (i->info > j->info) { // Nếu giá trị của phần tử i lớn hơn giá trị của phần tử j

temp = i->info;

i->info = j->info;

j->info = temp; // Hoán đổi giá trị của phần tử i và j

}

}

}

}

// Hàm xóa phần tử theo thứ tự

void RemoveOrder(Node\*& pHead, int X) {

if (pHead == NULL) return; // Nếu danh sách rỗng thì không làm gì

Node\* temp = pHead;

Node\* prev = NULL;

while (temp != NULL && temp->info != X) { // Duyệt danh sách để tìm phần tử cần xóa

prev = temp;

temp = temp->next;

}

if (temp != NULL) { // Nếu tìm thấy phần tử cần xóa

if (prev == NULL) { // Nếu phần tử cần xóa là phần tử đầu

pHead = temp->next; // Cập nhật con trỏ đầu

}

else {

prev->next = temp->next; // Cập nhật con trỏ của phần tử trước trỏ tới phần tử sau nút cần xóa

}

delete temp; // Xóa phần tử

}

}

// Hàm tìm kiếm phần tử trong danh sách

Node\* Find(Node\* pHead, int X) {

Node\* temp = pHead;

while (temp != NULL) { // Duyệt qua tất cả các phần tử trong danh sách

if (temp->info == X) { // Nếu tìm thấy phần tử với giá trị X

return temp; // Trả về con trỏ tới phần tử đó

}

temp = temp->next; // Tiến tới phần tử kế tiếp

}

return NULL; // Nếu không tìm thấy phần tử

}

// Duyệt danh sách và in ra các phần tử

void ShowList(Node\* pHead) {

Node\* temp = pHead;

while (temp != NULL) { // Duyệt qua danh sách

cout << temp->info << " -> "; // In giá trị của phần tử hiện tại

temp = temp->next; // Tiến tới phần tử kế tiếp

}

cout << "NULL" << endl; // In ra kết thúc danh sách

}

// Xóa danh sách (giải phóng bộ nhớ)

void ClearList(Node\*& pHead) {

Node\* temp;

while (pHead != NULL) { // Duyệt qua danh sách và giải phóng bộ nhớ cho từng nút

temp = pHead;

pHead = pHead->next; // Cập nhật con trỏ đầu danh sách

delete temp; // Xóa nút hiện tại

}

}

int main() {

Node\* head;

Init(head); // Khởi tạo danh sách liên kết rỗng

// Thêm các phần tử vào danh sách

InsertFirst(head, 10);

InsertFirst(head, 5);

InsertFirst(head, 7);

InsertFirst(head, 2);

InsertFirst(head, 8);

InsertFirst(head, 9);

InsertFirst(head, 1);

cout << "Danh sach sau khi them phan tu: ";

ShowList(head);

DeleteFirst(head);

cout << "Danh sach sau khi xoa phan tu dau tien: ";

ShowList(head);

Remove(head, 7);

cout << "Danh sach sau khi xoa phan tu 7: ";

ShowList(head);

SelectionSort(head);

cout << "Danh sach sau khi sap xep: ";

ShowList(head);

RemoveOrder(head, 8);

cout << "Danh sach sau khi xoa trong danh sach da duoc sap xep: ";

ShowList(head);

// Tìm kiếm phần tử trong danh sách

int searchVal = 5; // Giá trị cần tìm

Node\* result = Find(head, searchVal);

if (result != NULL) {

cout << "Phan tu co gia tri " << searchVal << " da duoc tim thay trong danh sach." << endl;

}

else {

cout << "Khong tim thay phan tu co gia tri " << searchVal << " trong danh sach." << endl;

}

// Tìm kiếm phần tử không có trong danh sách

searchVal = 20; // Giá trị cần tìm

result = Find(head, searchVal);

if (result != NULL) {

cout << "Phan tu co gia tri " << searchVal << " da duoc tim thay trong danh sach." << endl;

}

else {

cout << "Khong tim thay phan tu co gia tri " << searchVal << " trong danh sach." << endl;

}

ClearList(head);

cout << "Danh sach sau khi xoa: ";

ShowList(head);

return 0;

}

KẾT QUẢ

A screenshot of a computer

AI-generated content may be incorrect.