THỰC HÀNH CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

Họ tên: Khương Thanh Bình

MSSV:3123411034

Bài Tập Danh sách liên kết đơn

* Các tác vụ trong danh sách liên kết đơn

CODE

void init(Node\*& pHead)

{

pHead = NULL;

} //Khởi tạo danh sách rỗng

bool isEmpty(Node\* pHead)

{

return(pHead == NULL);

} //Kiểm tra danh sách, nếu rỗng trả về true, nếu không thì trả về false

Node\* find(Node\* pHead, int x) { //Tìm x trong danh sách

bool found = false; //Tạo biến found kiểu bool, gán giá trị false

Node\* p = pHead; //Tạo con trỏ p trỏ đến đầu danh sách

while ((p != NULL) && !found) { //Duyệt danh sách, nếu p khác NULL và found false thì thực hiện

if (p->info == x) found = true; //Nếu giá trị Node p bằng x, gán found = true

else p = p->next; //Nếu không, di chuyển con trỏ p đến Node tiếp theo

}

if (!found) p = NULL; //Nếu không tìm thấy, gán p = NULL

return p;

}//Trả về con trỏ p

Node\* findOrder(Node\* pHead, int x) { //Tìm kiếm x trong danh sách đã được sắp xếp

bool found = false; //Tạo biến found kiểu bool, gán giá trị false

Node\* p = pHead; //Tạo con trỏ p trỏ đến đầu danh sách

while ((p != NULL) && !found) //Duyệt danh sách, nếu p khác NULL và found false thì thực hiện

if (p->info == x)

found = true; //Nếu giá trị Node p bằng x -> Gán found = true

else if (p->info < x)

p = p->next; //Nếu giá trị Node p nhỏ hơn x -> Di chuyển con trỏ p đến Node tiếp theo

else

p = NULL; //Nếu không, gán p = NULL

if (!found)

p = NULL; //Nếu không tìm thấy, gán p = NULL

return p; //Trả về con trỏ p

}

Node\* createNode(int x) {

Node\* p = new Node;

p->info = x;

p->next = NULL;

return p;

} //Tạo 1 Node mới chứa giá trị x trỏ đến giá trị NULL

void ShowList(Node\* pHead) {

if (pHead == NULL) printf("Danh sach rong\n"); //Nếu danh sách rỗng thì thông báo, kết thúc hàm

else {

Node\* p = pHead; //Tạo con trỏ p trỏ đến đầu danh sách

while (p != NULL) {

cout << p->info << " ";

p = p->next;

} //Duyệt danh sách, in giá trị Node và di chuyển con trỏ p đến Node tiếp theo

}

} //In ra danh sách liên kết đơn

void insertFirst(Node\*& pHead, int x) {

Node\* p = createNode(x); //Tạo Node mới mang giá trị x

p->next = pHead; //Node mới trỏ đến Node đầu danh sách

pHead = p; //Node đầu danh sách trỏ đến Node mới

} //Chèn 1 Node mang giá trị x vào đầu danh sách

void insertAfter(Node\* p, int x) {

if (p != NULL) { //Nếu Node p không trỏ đến NULL -> Node p không ở cuối danh sách -> Có thể thực hiện chèn Node mới sau Node p

Node\* q = createNode(x); //Tạo Node mới mang giá trị x

q->next = p->next; //Node mới trỏ đến Node mà Node p đang trỏ

p->next = q; //Node p trỏ đến Node mới

} //Tức là q trỏ đến Node sau Node p, Node p trỏ đến Node q

}

void insertOrder(Node\*& pHead, int x) {

Node\* tp = NULL; //Tạo con trỏ tp trỏ đến NULL

Node\* p = pHead; //Tạo con trỏ p trỏ đến đầu danh sách

bool cont = true; //Tạo biến cont kiểu bool, gán giá trị true

while ((p != NULL) && cont) { //Duyệt ds, nếu p khác NULL và cont true thì thực hiện

if (p->info < x) {

tp = p;

p = p->next;

} //Nếu giá trị Node p nhỏ hơn x, tp trỏ đến p, p trỏ đến Node tiếp theo

else cont = false; //Nếu không, gán cont = false

}

if (p == pHead)

insertFirst(pHead, x); //Nếu p trỏ đến đầu danh sách, chèn Node mới vào đầu danh sách

else

insertAfter(tp, x); //Ngược lại chèn Node mới sau Node tp

}

void deleteFirst(Node\*& pHead) {

if (pHead == NULL) return; //Nếu danh sách rỗng thì kết thúc hàm

else {

Node\* p = pHead; //Tạo con trỏ p trỏ đến đầu danh sách

pHead = pHead->next; //Chuyển Node đầu danh sách thành Node tiếp theo

delete p; //Xóa Node p

}

}

void deleteAfter(Node\* p) {

if (p != NULL && p->next != NULL) { //Nếu Node p và Node sau p không trỏ đến NULL thì mới có Node sau nó để xóa

Node\* q = p->next; //Tạo Node q trỏ đến Node sau p

p->next = q->next; //Cập nhật lại liên kết giữa Node p và Node sau q

delete q; //Xóa Node q

}

}

void remove(Node\*& pHead, int x) {

if (pHead == NULL) return; //Nếu danh sách rỗng thì kết thúc

if (pHead->info == x) {

Node\* p = pHead; // Nếu giá trị đầu danh sách bằng x -> Tạo p trỏ tới pHead

pHead = pHead->next; //Chuyển Node đầu danh sách thành Node tiếp theo

delete p; //Xóa Node p

return;

}

Node\* p = pHead;//Tạo Node p trỏ đến đầu danh sách

while (p->next != NULL && p->next->info != x) p = p->next; //Duyệt danh sách, tìm Node có giá trị x

if (p->next != NULL) {

Node\* q = p->next; //Nếu tìm thấy Node có giá trị x -> Tạo Node q trỏ đến Node đó

p->next = q->next; //Cập nhật liên kết giữa Node p và Node sau q

delete q; //Xóa Node q

return;

}

}

void removeOrder(Node\*& pHead, int x) {

if (pHead == NULL) return; //Nếu danh sách rỗng thì kết thúc

if (pHead->info == x) { //Nếu giá trị đầu danh sách bằng x -> Xóa Node đầu danh sách

Node\* p = pHead;

pHead = pHead->next;

delete p;

return;

}

Node\* p = pHead;

while (p->next != NULL && p->next->info < x) p = p->next; //Duyệt danh sách, tìm Node có giá trị lớn hơn x

if (p->next->info == x) { //Nếu tìm thấy Node có giá trị x

Node\* q = p->next; //Tạo Node q trỏ đến Node đó

p->next = q->next; //Cập nhật liên kết giữa Node p và Node sau q

delete q; //Xóa Node q

}

}

void clearList(Node\*& pHead) {

Node\* p; //Tạo Node p

while (pHead != NULL) { //Duyệt danh sách, xóa từng Node

p = pHead; //Tạo Node p trỏ đến Node đầu danh sách

pHead = pHead->next; //Chuyển Node đầu danh sách thành Node tiếp theo

delete p;//Xóa Node p

}

}

void selectionSort(Node\* pHead) {

Node\* p, \* q, \* pmin; //Tạo con trỏ p,q, pmin

int vmin; //Tạo biến giá trị vmin

for (p = pHead; p->next != NULL;p = p->next) { //Trỏ p đến đầu danh sách, duyệt ds

vmin = p->info; //Gán vmin = giá trị Node p

pmin = p; //Gán pmin = p

for (q = p->next;q != NULL;q = q->next) { //Duyệt danh sách từ Node sau p

if (q->info < vmin) { //Nếu giá trị Node q nhỏ hơn vmin

vmin = q->info;

pmin = q;

}

}

pmin->info = p->info; //Sắp xếp, thay đổi giá trị Node pmin = giá trị Node p

p->info = vmin; //Thay đổi giá trị Node p = vmin

}

}

CODE TEST

void test1() {

cout << "TEST 1: KIEM TRA DANH SACH\n";

Node\* pHead;

init(pHead);

if (isEmpty(pHead))

cout << "Danh sach rong\n";

else

cout << "Danh sach da co phan tu\n";

}

void test2() {

cout << "TEST 2: CHEN, XOA, TIM KIEM TRONG DANH SACH\n";

cout << " CHEN TRONG DANH SACH: \n";

Node\* pHead;

init(pHead);

int a[] = { 9,2,4,5,6,3,7,8,1,10 }; int n = 10;

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

{

insertFirst(pHead, a[i]);

}

cout << "Danh sach sau khi chen vao dau: \n";

ShowList(pHead);cout << "\n";

insertAfter(pHead->next->next->next, 100);

cout << "Danh sach sau khi chen vao Node thu 4: \n";

ShowList(pHead);cout << "\n";

cout << "XOA TRONG DANH SACH\n";

deleteFirst(pHead);

cout << "Danh sach sau khi xoa Node dau tien: \n";

ShowList(pHead);cout << "\n";

deleteAfter(pHead->next);

cout << "Danh sach sau khi xoa Node so 2: \n";

ShowList(pHead);cout << "\n";

remove(pHead, 100);

ShowList(pHead);cout << "\n";

cout << "Tim kiem trong danh sach\n";

cout << "Tim so 1\n";

if (find(pHead, 1) == NULL) {

cout << "Khong tim thay\n";

}

else {

cout << "Tim thay\n";

}

cout << "Tim so 100\n";

if (find(pHead, 100) == NULL) {

cout << "Khong tim thay\n";

}

else {

cout << "Tim thay\n";

}

}

void test3() {

cout << "TEST 3: SAP XEP VA THUC HIEN CAC TAC VU KHI SAP XEP\n";

Node\* pHead;

init(pHead);

int a[] = { 3,5,8,7,9,2,1,4 }; int n = 8;

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

{

insertFirst(pHead, a[i]);

}

cout << "Danh sach sau khi chen: \n";

ShowList(pHead);cout << "\n";

selectionSort(pHead);

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

ShowList(pHead);cout << "\n";

cout << "Chen vao danh sach da sap xep: \n";

insertOrder(pHead, 6);

ShowList(pHead);cout << "\n";

cout << "Xoa trong danh sach da sap xep: \n";

removeOrder(pHead, 6);

ShowList(pHead);cout << "\n";

cout << "Tim kiem trong sap xep: \n";

cout << "Tim so 1: ";

if (findOrder(pHead, 1) == NULL) {

cout << "Không tìm thấy\n";

}

else {

cout << "Tim thay\n";

}

cout << "Tim so 6: ";

if (findOrder(pHead, 6) == NULL) {

cout << "Khong tim thay\n";

}

else {

cout << "Tim thay\n";

}

}

void test4() {

cout << "TEST 4: XOA TOAN BO DANH SACH\n";

Node\* pHead;

init(pHead);

int a[] = { 3,5,8,7,9,2,1,4 }; int n = 8;

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

{

insertFirst(pHead, a[i]);

}

cout << "Danh sach sau khi chen vao dau \n";

ShowList(pHead);cout << "\n";

clearList(pHead);

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

ShowList(pHead);cout << "\n";

}

RESULT

A screenshot of a computer program

AI-generated content may be incorrect.