

THỰC HÀNH - CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

Lưu hành nội bộ

AUGUST 1, 2018

FIT-UTC

**Bài 1. Thực hành bài Cấu trúc vector**

**Yêu cầu:**

a. Xây dưng lớp Vector, lớp bộ lặp của lớp Vector

b. Sử dụng lớp Vector và lớp bộ lăp của lớp vector xây dựng chương trình quản lý danh sác **số thực** có các chức năng sau:

1. Chèn 1 phần tử vào vector

2. Xóa 1 phần tử của vector

3. Thay thế một phần tử của vector

4. Lấy giá trị của một phần tử của vector

5. In danh sách các phần tử hiện có trong vector

**Các bước làm:**

**1.Tạo File Vector.cpp**

#ifndef VECTOR\_CPP

#define VECTOR\_CPP 1

template <class T>

class Vector{

private:

int N; //số chiều tối đa của vector

T \*V;//luu trữ dữ liệu

int n; //số phần tử hiện có trong Vector

public:

Vector();

~Vector();

int getAtRank(int r, T &o);

int replaceAtRank(int r, T o);

int insertAtRank(int r, T o);

int removeAtRank(int r, T &o);

int size();

int isEmpty();

};

template<class T>

Vector<T>::Vector()

{

V = new T[1];

N = 1;

n = 0;

}

template<class T>

Vector<T>::~Vector()

{

delete V;

}

//Ham insertAtRank

template<class T>

int Vector<T>::insertAtRank(int r, T o)

{

if(r<0 || r > n )

1

return 0;

if(n==N)

{ //Phat trien mang

T \*A;

N = 2\*N;

A = new T[N];

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

A[i] = V[i];

delete V;

V = A;

int k = n-1;

while(k>=r)

{

V[k+1] = V[k];

k--;

}

}

V[r]= o;

n++;

return 1;

}

//Ham lay ra mot phan tu

template<class T>

int Vector<T>::getAtRank(int r, T &o) {

if(r<0 || r>n-1)

return 0;

o = V[r];

return 1;

}

//Ham thay the mot phan tu

template<class T>

int Vector<T>::replaceAtRank(int r, T o) {

if(r<0 || r>n-1)

return 0;

V[r] = o;

return 1;

}

template<class T>

int Vector<T>::removeAtRank(int r, T &o) {

if(r<0 || r>n-1)

return 0;

o = V[r];

int k = r;

while(k<n-1)

{

V[k] = V[k+1];

k++;

}

n--;

return 1;

2

}

template<class T>

int Vector<T>::size(){ return n;}

template<class T>

int Vector<T>::isEmpty()

{

return n==0;

}

#endif

**2.Tạo File VITR.CPP**

#ifndef VectorItr\_cpp

#define VectorItr\_cpp 1

#include"Vector.cpp"

template <class T>

class VectorItr{

private :

Vector<T>\* theVector;

int cur\_Index;

public:

VectorItr(Vector<T>\*V1)

{

theVector = V1;

cur\_Index = 0;

}

int hasNext(){

if (cur\_Index<theVector->size())

return 1;

else

return 0;

}

T next(){

T o;

theVector->getAtRank(cur\_Index, o);

cur\_Index++;

return o;

}

};//End of class VectorItr

#endif

**3.Tạo File Student.cpp**

#ifndef STUDENT\_CPP

#include"conio.h"

#include"iostream"

using namespace std;

class Student

{

private:

int masv;

char hoten[30];

char gioi[4];

public:

friend istream & operator >>(istream &is, Student &s); friend ostream & operator <<(ostream &os, Student s);

3

};

istream & operator >>(istream &is, Student &s) {

cout<<"\nNhap ma sv:";

is>>s.masv;

cout<<"Nhap ho va ten:";

is.ignore(1);

is.get(s.hoten,30);

cout<<"Nhap gioi tinh:";

is.ignore(1);

is.get(s.gioi,4);

return is;

}

ostream & operator <<(ostream &os, Student s) {

os<<s.masv<<"\t"<<s.hoten<<"\t" <<s.gioi;

return os;

}

#endif

**4.Tạo File StuApp.cpp**

#include"conio.h"

#include"stdio.h"

#include"iostream"

#include"Vector.cpp"

#include"VItr.cpp"

#include"Student.cpp"

using namespace std;

class VectorApp //Lop ung dung lop vector va lop VectorItr {

private:

Vector<Student> v;

public:

int menu();

void run();

void GetElement();

void InsertElement();

void RemoveElement();

void ReplaceElement();

void ListElement();

};

int VectorApp::menu()

{

cout<<"1.Them mot sinh vien moi";

cout<<"\n2.Xoa mot sinh vien";

cout<<"\n3.Thay the mot sinh vien";

cout<<"\n4. Lay thong tin mot sinh vien";

cout<<"\n5.In danh sach sinh vien";

cout<<"\n6.Ket thuc chuong trinh";

cout<<"\nChon chuc nang tu 1..6:";

int n;

cin>>n;

return n;

}

4

void VectorApp::run()

{

int ch;

do{

system("cls");

ch = menu();

system("cls");

switch(ch)

{

case 1:

InsertElement();

break;

case 2:

RemoveElement();

break;

case 3:

ReplaceElement();

break;

case 4:

GetElement();

break;

case 5:

ListElement();

break;

}

getch();

}while(ch!=6);

}

void VectorApp::InsertElement()

{

Student x;

int r;

cout<<"Nhap thong tin cua sinh vien:";

cin>>x;

cout<<"Vi tri chen:";

cin>>r;

if(v.insertAtRank(r,x)==1)

cout<<"Chen phan tu thanh cong!";

else

cout<<"Chen phan tu khong thanh cong. Loi do vi tri chen";

}

void VectorApp::RemoveElement()

{

Student x;

int r;

cout<<"Nhap vi tri cua sinh vien can xoa bo:";

cin>>r;

if(v.removeAtRank(r,x)==1)

cout<<"Phan tu bi xoa di :"<<x;

else

cout<<"Xoa phan tu khong thanh cong. Loi do vi tri xoa";

5

}

void VectorApp::ReplaceElement()

{

Student x;

int r;

cout<<"Nhap thong tin sinh vien:";

cin>>x;

cout<<"Nhap vi tri can thay the:";

cin>>r;

if(v.replaceAtRank(r,x)==1)

cout<<"Thay the phan tu thanh cong";

else

cout<<"Thay the phan tu khong thanh cong. Loi do vi tri thay the";

}

void VectorApp::GetElement()

{

Student x;

int r;

cout<<"Nhap vi tri can lay phan tu:";

cin>>r;

if(v.getAtRank(r,x)==1)

cout<<"Phan tu lay duoc:"<<x;

else

cout<<"Lay phan tu khong thanh cong. Loi do vi tri lay"; }

void VectorApp::ListElement()

{

VectorItr<Student> Itr(&v);

cout<<"Danh sach cac sinh vien:\n";

while(Itr.hasNext())

cout<<Itr.next()<<"\n";

}

**5.Tạo File Demo.cpp**

#include"conio.h"

#include"stdio.h"

#include"StuApp.cpp"

int main()

{

VectorApp x;

x.run();

return 0;

}

**6.Chạy file Demo**

6

**Bài 2**

**Thực hành bài Cấu trúc danh sách liên kết đơn (Single List)**

**Yêu cầu:**

a. Cài đặt lớp Node, lớp SingleList, bộ lặp của lớp SingleList

b. Xây dựng lớp ứng dụng sử dụng lớp Danh sách liên kết đơn để lưu trữ 1 danh sách sinh viên. Mỗi sinh viên gồm các thông tin sau: Mã sinh viên, họ tên, giới tính.

**Yêu cầu lớp có các chức năng sau:**

1. Thêm một sinh viên vào cuối DS

2. Thêm một sinh viên vào đầu DS

3. Xóa bỏ một sinh viên thu i khỏi DS

4. Thay thế sinh viên thứ i bằng một sinh viên mới

5. Xây dựng chương trình để chạy lớp ứng dụng

**Các bước làm:**

**1.Tạo File node.cpp**

#ifndef NODE\_CPP

#define NODE\_CPP 1

#include"iostream"

using namespace std;

template <class T>

class Node{

private:

T elem;

Node \*next;

public:

Node();

Node \*getNext(); //Tra lai dia chi cua doi tuong do thanh

//phan Next tro den

void setNext(Node<T>\*); //Dat thanh phan Next tro

//vao doi tuong la doi cua

phuong thuc

T getElem(); // Tra lai dia chi cua phan tu luu trong Node

void setElem(T); // Dat thay the phan tu luu trong Node

// bang phan tu moi

};

//------------------------------------------------------------------ template <class T>

Node<T>::Node(){

next = NULL;

}

//-----------------------------------------------------------------

7

template <class T>

Node<T> \*Node<T>::getNext(){

return next;

}

//------------------------------------------------------------------ template <class T>

void Node<T>::setNext(Node<T> \*p){

next = p;

}

//------------------------------------------------------------------ template <class T>

T Node<T>::getElem(){

return elem;

}

//------------------------------------------------------------------ template <class T>

void Node<T>::setElem(T e){

elem = e;

}

#endif

**2.Tạo File SList.CPP**

#ifndef SINGLELIST\_CPP

#define SINGLELIST\_CPP 1

#include "node.cpp"

#include "iostream"

using namespace std;

template <class T>

class SingleList{

private:

Node<T> \*header; // Con tro tro vao dau danh sach

Node<T> \*trailer; // Con tro tro vao phan tu cuoi cua danh sach long n; // so phan tu hien co cua danh sach public:

SingleList();

long size();

int isEmpty();

Node<T>\* first(); // Tra lai dia chi do con tro header tro den

Node<T>\* last(); // Tra lai dia chi do con tro trailer tro den void replace(Node<T> \*p, T e); //Thay the phan tu luu trong p bang phan tu e va tra lai dia chi p

Node<T>\* insertAfter(Node<T> \*p, T e); // Chen them node sau node do p tro toi

Node<T>\* insertFirst(T e); // Chen them 1 node vao dau danh sach Node<T>\* insertLast(T e); //Chen them mot node vao cuoi danh sach

void remove(Node<T> \*p); // Loai bo node do con tro p tro toi

Node<T>\* getNode(int i);

};

//--------------------------------------------------------------------- ----------------------------------------------------------

8

template <class T>

SingleList<T>::SingleList(){

header = NULL;

trailer = NULL;

n = 0;

}

//--------------------------------------------------------------------- ---------------------------------------------------------- template <class T>

long SingleList<T>::size (){

return n;

}

//--------------------------------------------------------------------- ---------------------------------------------------------- template <class T>

int SingleList<T>::isEmpty(){

return n == 0;

}

//--------------------------------------------------------------------- ---------------------------------------------------------- template <class T>

Node<T>\* SingleList<T>::first(){

return header;

}

template <class T>

Node<T>\* SingleList<T>::last(){

return trailer;

}

//--------------------------------------------------------------------- ---------------------------------------------------------- // Thay the phan tu luu trong node do co dia chi luu trong p bang phan tu co gia tri e

template <class T>

void SingleList<T>::replace(Node<T>\* p,T e){

p->setElem(e);

}

//--------------------------------------------------------------------- ---------------------------------------------------------- template <class T>

Node<T>\* SingleList<T>::insertAfter(Node<T>\* p, T e){

Node<T> \*q;

q = new Node<T>;

q->setElem(e);

q->setNext(p->getNext());

p->setNext(q);

if(p==trailer)

trailer = q;

n++;

return q;

}

9

//--------------------------------------------------------------------- ---------------------------------------------------------- template <class T>

Node<T>\* SingleList<T>::insertFirst(T e){

Node<T>\* q;

q = new Node<T>;

q->setElem(e);

if(isEmpty()){

header = q;

trailer = q;

}

else{

q->setNext(header);

header = q;

}

n++;

return q;

}

//--------------------------------------------------------------------- ---------------------------------------------------------- template <class T>

Node<T>\* SingleList<T>::insertLast(T e){

Node<T> \*q;

q = new Node<T>;

q->setElem(e);

if(isEmpty()){

header = q;

trailer = q;

}

else{

trailer->setNext(q);

trailer = q;

}

n++;

return q;

}

//--------------------------------------------------------------------- ---------------------------------------------------------- template <class T>

void SingleList<T>:: remove(Node<T>\* p){

Node<T> \*q;

if(p==header){

if(size()==1){

header = NULL;

trailer = NULL;

}

else

header = header->getNext();

}

else{

q = header;

10

while(q->getNext()!= p) // Vong lap tim den vi tri cua node truoc node p

q = q->getNext();

if(q!=NULL){ //Tim thay node can xoa

if(p!=trailer)

q->setNext(p->getNext());

else

trailer = q;

}

}

n--;

delete []p;

}

//--------------------------------------------------------------------- --

template<class T>

Node<T>\* SingleList<T>::getNode(int i)

{

if(i<0 || isEmpty())

return NULL;

Node<T> \*p;

int dem=0;

p = header;

while(dem<i && p!=NULL){

dem++;

p = p->getNext();

}

return p;

}

#endif

**3.Tạo file SLITR.cpp //Bộ lặp trên danh sách liên kết đơn**

#ifndef SINGLE\_LIST\_ITR\_H

#define SINGLE\_LIST\_ITR\_H 1

#include "slist.cpp"

template <class T>

class SingleListItr{

private:

SingleList<T>\* slist;

Node<T> \*cur;

public:

SingleListItr(SingleList<T>\* list){

slist = list;

cur = slist->first();

}

void reset(){

cur = slist->first();

}

//--------------------------------------------------------------- -

11

int hasNext(){

return cur!=NULL;

}

//--------------------------------------------------------------- -

T next(){

T o;

o = cur->getElem();

cur = cur->getNext();

return o;

}

};

#endif

**4.Tạo File student.Cpp**

#ifndef STUDENT\_CPP

#include"conio.h"

#include"iostream"

using namespace std;

class Student

{

private:

int masv;

char hoten[30];

char gioi[4];

public:

friend istream & operator >>(istream &is, Student &s); friend ostream & operator <<(ostream &os, Student s); };

istream & operator >>(istream &is, Student &s)

{

cout<<"\nNhap ma sv:";

is>>s.masv;

cout<<"Nhap ho va ten:";

is.ignore(1);

is.get(s.hoten,30);

cout<<"Nhap gioi tinh:";

is.ignore(1);

is.get(s.gioi,4);

return is;

}

ostream & operator <<(ostream &os, Student s)

{

os<<s.masv<<"\t"<<s.hoten<<"\t" <<s.gioi;

return os;

}

#endif

**5.Tạo File App.cpp**

#include "conio.h"

#include "stdio.h"

#include "iostream"

#include "Slist.cpp"

#include "SLItr.cpp"

#include "Student.cpp"

12

using namespace std;

#define SIZE 1

#define INSERT\_FIRST 2

#define INSERT\_LAST 3

#define INSERT\_AFTER 4

#define REPLACE 5

#define REMOVE 6

#define DISPLAY\_LIST 7

class AppSingleList{

private:

SingleList<Student> \*list;

public:

AppSingleList();

void listSize();

void insertFirst();

void insertLast();

void insertAfter();

void replace();

void remove();

void displayList();

int menu();

void run();

};

//------------------------------------------------------------------ --------------------

AppSingleList::AppSingleList(){

list = new SingleList<Student>();

}

//------------------------------------------------------------------ --------------------

void AppSingleList::listSize(){

cout<<"So sinh vien hien co la:"<<list->size();

}

//------------------------------------------------------------------ --------------------

void AppSingleList::insertFirst(){

Student x;

cout<<"Nhap thong tin sinh vien:";

cin>>x;

if(list->insertFirst(x))

cout<<"Da them sinh vien vao dau thanh cong.";

else

cout<<"Them sinh vien khong thanh cong.";

}

//------------------------------------------------------------------ --------------------

void AppSingleList::insertLast(){

Student x;

cout<<"Nhap thong tin cua sinh vien:";

cin>>x;

if(list->insertLast(x))

cout<<"Da them sinh vien vao cuoi thanh cong.";

13

else

cout<<"Them sinh vien khong thanh cong.";

}

//------------------------------------------------------------------ --------------------

void AppSingleList::replace(){

Student x;

Node<Student> \*p;

int n;

cout<<"Nhap thong tin sinh vien can thay:";

cin>>x;

cout<<"Thay the sinh vien thu may (<="<<list->size()<<"):"; cin>>n;

p = list->getNode(n);

if(p!=NULL){

list->replace(p,x);

cout<<"Thay the thanh cong.";

}

else

cout<<"Thay the khong thanh cong.";

}

//------------------------------------------------------------------ --------------------

void AppSingleList::insertAfter(){

Student x;

Node<Student> \*p;

int n;

cout<<"Nhap thong tin sinh vien:";

cin>>x;

cout<<"Chen sau sinh vien thu may(<="<<list->size()<<"):"; cin>>n;

p = list->getNode(n);

if(p!=NULL){

list->insertAfter(p,x);

cout<<"Chen sinh vien thanh cong.";

}

else

cout<<"Chen sinh vien khong thanh cong.";

}

//------------------------------------------------------------------ --------------------

void AppSingleList::remove(){

int n;

Node<Student> \*p;

cout<<"Xoa sinh vien o vi tri thu may:";

cin>>n;

p = list->getNode(n);

if(p!=NULL){

list->remove(p);

cout<<"Da xoa thanh cong.";

}

else

cout<<"Xoa khong thanh khong.";

14

}

//------------------------------------------------------------------ --------------------

void AppSingleList::displayList(){

SingleListItr<Student> itr(list);

cout<<"Cac sinh vien hien co trong danh sach: ";

while(itr.hasNext())

cout<<"\n"<<itr.next();

}

//------------------------------------------------------------------ --------------------

int AppSingleList::menu(){

int n;

cout<<"1. Kich thuoc hien tai cua danh sach.\n";

cout<<"2. Chen phan tu vao dau danh sach. \n";

cout<<"3. Chen phan tu vao cuoi danh sach. \n";

cout<<"4. Chen phan tu vao sau mot phan tu nao do trong danh sach.\n";

cout<<"5. Thay the phan tu hien co cua mot node. \n"; cout<<"6. Loai bo mot node khoi danh sach.\n";

cout<<"7. Hien thi toan bo danh sach:\n";

cout<<"8. Ket thuc.\n";

cout<<" Hay nhap vao so 1..8:";

cin>>n;

return n;

}

//----------------------------------------------------------------- ---------------------

void AppSingleList::run(){

int choise;

do{

system("cls");

choise = menu();

system("cls");

switch(choise){

case SIZE:

listSize();

break;

case INSERT\_FIRST:

insertFirst();

break;

case INSERT\_LAST:

insertLast();

break;

case INSERT\_AFTER:

insertAfter();

break;

case REPLACE:

replace();

break;

case REMOVE:

remove();

15

break;

case DISPLAY\_LIST:

displayList();

break;

}

getch();

}while(choise!=8);

}

**6.Tạo File Demo.cpp**

#include "app.cpp"

#include "stdio.h"

int main(){

AppSingleList x;

x.run();

return 0;

}

**7.Chạy file Demo**

16

**Bài 3**

**Thực hành thuật toán sắp xếp O(n2)**

**Yêu cầu:**

a. Cài đặt các thuật toán sắp xếp nổi bọt, sắp xếp chọn, sắp xếp chèn b. Nhập vào dãy số, lần lượt sử dụng các thuật toán để sắp xếp dãy số tăng dần hoặc giảm dần

c. Nhập vào một danh sách sinh viên, lần lượt sử dụng các thuật toán để sắp xếp theo mã sinh viên hoặc theo tên, biết mỗi sinh viên gồm các thông tin: mã sinh viên, họ tên, giới tính.

**Các bước làm:**

**1. File Array.cpp**

#ifndef ARRAY\_H

#define ARRAY\_H 0

#include"iostream"

using namespace std;

template <class T>

void InputArr(T \*a, int n, char \*c){

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

cout<<c<<"["<<i<<"]=";

cin>>a[i];

}

}

template <class T>

void PrintArr(T \*a, int n, int xuongdong){

//xuongdong=1 thi in ra theo cot, nguoc lai in ra theo hang

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

if (xuongdong)

cout<<a[i]<<"\n";

else

cout<<a[i]<<" ";

}

#endif

**2. File sortnn.cpp**

#ifndef SORT\_NN\_H

#define SORT\_NN\_H 1

#include"array.cpp"

#include"conio.h"

17

template <class T>

void Swap(T &a, T &b)

{

T tg =a;

a= b;

b=tg;

}

template <class T>

void BubbleSort(T \*a, int n, int (\*comp)(T,T)){ int i, j;

for (i=0;i<n-1;i++)

for(j=n-1;j>i;j--)

if(comp(a[j],a[j-1]))

Swap(a[j],a[j-1]);

}

template<class T>

void SelectionSort(T \*a,int n, int (\*comp)(T,T)){ int i,j,min;

for(i=0;i<=n-2;i++){

min=i;

for(j=i+1;j<n;j++)

if(comp(a[min],a[j])) min=j; if(min!=i)

Swap(a[i],a[min]);

}

}

18

template<class T>

void InsertionSort(T \*a,int n, int (\*comp)(T,T)){ T x;

int i, j;

for(i=1; i<=n-1;i++){

j = i-1;

x = a[i];

while(comp(a[j],x) && j>=0){

a[j+1]=a[j];

j--;

}

a[j+1]= x;

}

}

#endif

**3. File demo1.cpp**

#include"conio.h"

#include"stdio.h"

#include"iostream"

#include"sortnn.cpp"

#include"array.cpp"

using namespace std;

int compare0(float x, float y){

if (x<y)

return 1;

else

return 0;

}

int compare1(float x, float y){

if (x>y)

return 1;

else

return 0;

}

19

int main(){

float \*a;

int n;

system("cls");

cout<<"Nhap n=";

cin>>n;

a = new float[n];

InputArr(a, n, "a");

system("cls");

cout<<"Day so ban dau:";

PrintArr(a,n,0);

BubbleSort(a,n,compare0); //SelectionSort(a,n,compare1); //InsertionSort(a,n,compare); //cout<<"\nDay so duoc sep:"; cout<<"\n";

PrintArr(a,n,0);

BubbleSort(a,n,compare1); cout<<"\n";

PrintArr(a,n,0);

getch();

return 0;

}

**4. Demo2**

- **Tạo file student.cpp**

#ifndef STUDENT\_CPP

#include"conio.h"

#include"iostream"

using namespace std;

class Student

{

private:

int masv;

char hoten[30];

char gioi[4];

20

public:

int getMaSV(){ return masv;}

char\* getHoten(){ return hoten;}

char\* getGioi(){ return gioi;}

friend istream & operator >>(istream &is, Student &s); friend ostream & operator <<(ostream &os, Student s); };

istream & operator >>(istream &is, Student &s) {

cout<<"\nNhap ma sv:";

is>>s.masv;

cout<<"Nhap ho va ten:";

is.ignore(1);

is.get(s.hoten,30);

cout<<"Nhap gioi tinh:";

is.ignore(1);

is.get(s.gioi,4);

return is;

}

ostream & operator <<(ostream &os, Student s) {

os<<s.masv<<"\t"<<s.hoten<<"\t" <<s.gioi;

return os;

}

#endif

- **Tạo file demo2.cpp**

#include"conio.h"

#include"stdio.h"

21

#include"string.h"

#include"iostream"

#include"sortnn.cpp"

#include"array.cpp"

#include"student.cpp"

using namespace std;

int compare(Student x, Student y){ if (x.getMaSV()<y.getMaSV())

return 1;

else

return 0;

}

int compare\_Name(Student x, Student y){ if (strcmp(x.getHoten(),y.getHoten())<0) return 1;

else

return 0;

}

int main(){

Student \*a;

int n;

system("cls");

cout<<"Nhap so sinh vien n="; cin>>n;

a = new Student[n];

InputArr(a, n, "Nhap SV thu "); system("cls");

cout<<"Danh sach sinh vien:\n";

22

PrintArr(a,n,1);

//BubbleSort(a,n,compare);

BubbleSort(a,n,compare\_Name);

// SelectionSort(a,n,compare);

//InsertionSort(a,n,compare);

cout<<"\nDanh sach sinh vien sau khi sap xep:\n"; PrintArr(a,n,1);

getch();

return 0;

}

23

**Bài 4**

**Thực hành thuật toán sắp xếp O(nlogn) Yêu cầu:**

a. Cài đặt các thuật toán sắp xếp Quick sort, Merge sort, Heap sort. b. Nhập vào dãy số, lần lượt sử dụng các thuật toán để sắp xếp dãy số tăng dần hoặc giảm dần

**Các bước làm:**

**1. Tạo file Sortnlogn.cpp**

template <class T>

void Swap(T &a, T &b)

{

T tg =a;

a= b;

b=tg;

}

template <class T>

void Partition (T \*A, int i, int j, int &right) {

T p = A[i];

int left = i;

right = j;

while( left < right)

{

while(A[left]<=p && left<=right) left++; while(A[right]>p) right--;

if(left < right)

Swap(A[left],A[right]);

}

if(i!=right)

Swap(A[i],A[right]);

}

template <class T>

void QuickSort(T \*a,int i, int j)

{

int k;

if(i<j)

{

Partition(a,i,j,k);

24

QuickSort(a,i,k-1);

QuickSort(a,k+1,j);

}

}

//Thuat toan sap xep tron

template <class T>

void Merge( T \*A, T \*B, int i, int k, int j) {

int left=i;

int right=k+1;

int t = i;

while(left<=k && right<=j)

{

if (A[left]<A[right])

{

B[t] = A[left];

left++;

t++;

}

else

{

B[t] = A[right];

right++;

t++;

}

}//ket thuc while

if(left>k)

for(int r=right;r<=j;j++) {

B[t]=A[r];

t++;

}

else

for(int r=left; r<=k; r++) {

B[t]=A[r];

t++;

}

for(int r =i; r<=j ;r++)

A[r] = B[r] ;

}

template <class T>

25

void Mergesort(T \*A, T \*B, int i, int j) {

if(i<j)

{

int k=(i+j)/2;

Mergesort(A,B,i, k);

Mergesort(A,B,k+1,j);

Merge(A,B, i, k, j);

}

}

//Sap xep heap sort

template<class T>

void Pushdown (T \*A, int i, int n) {

int j = i;

int kt=0;

int max;

while (j<=n/2 && kt==0)

{

if(2\*j==n)

max = 2\*j;

else

if (A[2\*j]<=A[2\*j+1]) max = 2\*j+1;

else

max = 2\*j;

if (A[j]<A[max])

{

Swap (A[j], A[max]); j = max;

}

else

kt=1;

}

}

template<class T>

void HeapSort(T \*A, int n)

{

int i;

for(i=(n-1)/2; i>= 0;i--) Pushdown(A,i, n-1);

26

for(i=n-1;i>=2;i--)

{

Swap(A[0],A[i]);

Pushdown(A,0,i-1);

}

}

**2. File demo1.cpp**

#include"conio.h"

#include"stdio.h"

#include"iostream"

#include"sortnlogn.cpp"

#include"array.cpp"

using namespace std;

int compare0(float x, float y){

if (x<y)

return 1;

else

return 0;

}

int compare1(float x, float y){

if (x>y)

return 1;

else

return 0;

}

int main(){

float \*a;

int n;

system("cls");

cout<<"Nhap n=";

cin>>n;

a = new float[n];

InputArr(a, n, "a");

system("cls");

cout<<"Day so ban dau:";

PrintArr(a,n,0);

Quicksort(a,0, n-1); //Thay hàm MergeSort, HeapSort

cout<<"\nDay so duoc sep:";

cout<<"\n";

27

PrintArr(a,n,0);

BubbleSort(a,n,compare1); cout<<"\n";

PrintArr(a,n,0);

getch();

return 0;

}

28

**Bài 5**

**Thực hành cây tìm kiếm nhị phân**

**Yêu cầu:**

a. Cài đặt lớp Node, lớp Btree

b. Cài đặt chương trình, tạo cây nhị phân với khóa tìm kiềm có giá trị nguyên và giá trị là các chuỗi ký tự bất kỳ. Hãy duyệt cây và in ra mìn hình giá trị của cây theo thứ tự duyệt trước, giữa, sau.

c.

**Các bước làm:**

**1. File BNode.cpp**

#ifndef NODE\_H

#define NODE\_H 1

#include"stdio.h"

template <class Keys, class T>

class BNode{

private:

Keys key;

T elem;

BNode<Keys,T> \*parent;

BNode<Keys,T> \*left;

BNode<Keys,T> \*right;

public:

BNode(){

parent = NULL;

left = NULL;

right = NULL;

}

BNode<Keys,T> \*getParent() { return parent;}

BNode<Keys,T> \*getLeft() { return left;}

BNode<Keys,T> \*getRight() { return right;}

void setLeft(BNode<Keys,T>\* p){ left = p;}

void setRight(BNode<Keys,T>\* p) {right = p;}

void setParent(BNode<Keys,T>\* p) {parent= p;}

int hasLeft() { return left!=NULL; }

int hasRight() { return right!=NULL;}

T getElem(){ return elem;}

void setElem(T e) { elem =e;}

Keys getKey(){ return key;}

void setKey(Keys k){ key = k; }

};

#endif

**2. File BTree.cpp**

29

#ifndef BTREE\_H

#define BTREE\_H 1

#include"stdio.h"

#include"BNode.cpp"

template <class Keys, class T>

class BTree{

private:

BNode<Keys,T> \*root;

long count;

void inOrder(BNode<Keys,T>\*,BNode<Keys,T>\*&first,int &kt);

void remove(BNode<Keys,T>\*&);

public:

BTree();

BNode<Keys,T> \*getRoot();

int size();

int isEmpty();

int isInternal(BNode<Keys,T>\*);

int isExternal(BNode<Keys,T>\*);

int isRoot(BNode<Keys,T>\*);

void preOrder(BNode<Keys,T>\*, void (\*visit)(BNode<Keys,T>\*));

void inOrder(BNode<Keys,T>\*, void (\*visit)(BNode<Keys,T>\*));

void postOrder(BNode<Keys,T>\*,void (\*visit)(BNode<Keys,T>\*));

BNode<Keys,T>\*search(Keys, BNode<Keys,T>\*);

BNode<Keys,T>\* insert(Keys, T);

void remove(Keys);

};

//------------------ Cai dat cac phuong thuc -------------- template <class Keys, class T>

BTree<Keys,T>::BTree()

{

root = NULL;

count=0;

}

template <class Keys, class T>

BNode<Keys,T>\* BTree<Keys,T>::getRoot()

{

return root;

}

template <class Keys, class T>

int BTree<Keys,T>::size()

{

return count;

}

template <class Keys, class T>

30

int BTree<Keys,T>::isEmpty()

{

return root==NULL;

}

template <class Keys, class T>

int BTree<Keys,T>::isInternal(BNode<Keys,T>\* p)

{

return p->hasLeft()||p->hasRight();

}

template <class Keys, class T>

int BTree<Keys,T>::isExternal(BNode<Keys,T>\*p)

{

if(!p->hasLeft() && !p->hasRight())

return 1;

else

return 0;

}

template <class Keys, class T>

int BTree<Keys,T>::isRoot(BNode<Keys,T>\* p)

{

if(p->getParent()==NULL)

return 1;

else

return 0;

}

template <class Keys, class T>

void BTree<Keys,T>::preOrder(BNode<Keys,T>\*p,void (\*visit)(BNode<Keys,T>\*))

{

if(p!=NULL)

{

visit(p);

preOrder(p->getLeft(),visit);

preOrder(p->getRight(),visit);

}

}

template <class Keys, class T>

void BTree<Keys,T>::inOrder(BNode<Keys,T>\*p,void (\*visit)(BNode<Keys,T>\*))

{

if(p!=NULL)

{

inOrder(p->getLeft(),visit);

visit(p);

inOrder(p->getRight(),visit);

}

31

}

template <class Keys, class T>

void BTree<Keys,T>::postOrder(BNode<Keys,T>\*p, void (\*visit)(BNode<Keys,T>\*))

{

if(p!=NULL)

{

postOrder(p->getLeft(),visit);

postOrder(p->getRight(),visit);

visit(p);

}

}

template <class Keys, class T>

BNode<Keys,T>\* BTree<Keys,T>::search(Keys key, BNode<Keys,T>\* p)

{

if(p!=NULL)

{

if(p->getKey()>key)

return search(key,p->getLeft());

else

if(p->getKey()<key)

return search(key,p->getRight());

else

return p;

}

else

return NULL;

}

template <class Keys, class T>

BNode<Keys,T>\* BTree<Keys,T>::insert(Keys key, T elem) {

BNode<Keys, T>\*p;

BNode<Keys, T>\*q = new BNode<Keys, T>();

q->setKey(key);

q->setElem(elem);

if(root==NULL)

{

root = q;

count++;

}

else

{

p = root;

while(p != NULL)

{

if(key< p->getKey())

if(p->getLeft()==NULL)

{

32

q->setParent(p);

p->setLeft(q);

count++;

p = NULL; //d?t p=Null d? k?t thúc

}

else

p = p->getLeft();

else

if(key> p->getKey()) // nam ben cay con ben

phai

if(p->getRight()== NULL)

{

q->setParent(p);

p->setRight(q);

count++;

p = NULL;

}

else

p = p->getRight();

else

{

delete q;

p=NULL;

}

}

}

return q;

}

template <class Keys, class T>

void BTree<Keys,T>::inOrder(BNode<Keys,T> \*p, BNode<Keys,T> \*&first, int &kt)

{

if(p!=NULL && kt!=1)

{

inOrder(p->getLeft(),first, kt);

if(kt==0)

{

first = p;

kt=1;

}

//inOrder(p->getRight(),first, kt);

}

}

template <class Keys, class T>

void BTree<Keys,T>::remove(BNode<Keys,T> \*&v)

{

BNode<Keys,T> \*p;

if (!v->hasLeft() && !v->hasRight())

33

{

p=v->getParent();

if(p!=NULL)

if(v == p->getLeft())

p->setLeft(NULL);

else

p->setRight(NULL);

else

v = NULL;

}

if(v->hasLeft() && !v->hasRight())

{

p=v->getParent();

v->getLeft()->setParent(p);

if(p->getLeft()==v)

p->setLeft(v->getLeft());

else

p->setRight(v->getLeft());

}

if((!v->hasLeft()) && v->hasRight())

{

p=v->getParent();

v->getRight()->setParent(p);

if(p->getLeft()==v)

p->setLeft(v->getRight());

else

p->setRight(v->getRight());

}

delete v;

}

template <class Keys, class T>

void BTree<Keys,T>::remove(Keys key) {

BNode<Keys,T>\*v = search(key, root);

if(v==NULL) return;

if(v->hasLeft()&& v->hasLeft())//Có ca hai con {

BNode<Keys,T> \*first;

int kt=0;

inOrder(v->getRight(), first, kt);

v->setKey(first->getKey());

v->setElem(first->getElem());

remove(first);

}

else

remove(v);

count--;

}

#endif

34

**3. File Demo.cpp**

#include "conio.h"

#include "Btree.cpp"

#include "iostream"

using namespace std;

void visit(BNode<int,char\*>\*p)

{

cout<<p->getElem()<<" ";

}

int main(){

BTree<int,char\*> tree;

BNode<int,char\*> \*p;

//BNode<int,float> \*p;

system("cls");

tree.insert(100,"100-1");

tree.insert(70,"70a");

tree.insert(150,"150x");

tree.insert(120,"120-y");

//tree.preOrder(tree.getRoot(),visit);

cout<<"\nDuyet cay theo thu tu giua:";

tree.inOrder(tree.getRoot(),visit);

// cout<<"\n";

// tree.postOrder(tree.getRoot(),visit);

//tree.remove(150);

cout<<"\n";

//tree.inOrder(tree.getRoot(),visit);

p = tree.search(120, tree.getRoot());

if(p!=NULL)

cout<<"Tim thay khoa 120, Co gia tri "<<p->getElem(); else

cout<<"Khong tim thay khoa 120.";

tree.remove(120);

//tree.remove(100);

tree.remove(70);

cout<<"\nDuyet cay theo thu tu giua:";

tree.inOrder(tree.getRoot(),visit);

getch();

return 0;

}