THỰC HÀNH - CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

Lưu hành nội bộ

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 myvector.cpp

```
#include<iostream>
#ifndef __vector__cpp__
#define __vector__cpp__
using namespace std;
template <class T>
class myvector_reverse_iterator
{
      T *curr;
      public:
            myvector_reverse_iterator(T *c=0) {curr=c;}
            myvector_reverse_iterator<T>
&operator=(myvector_reverse_iterator<T> it)
            {
                  this->curr=it.curr;
                  return *this;
            }
            myvector_reverse_iterator<T> operator++()//++curr
            {
                  curr--;
                  return curr;
            }
            myvector_reverse_iterator<T> operator++(int)//++curr
            {
                  myvector_reverse_iterator<T> tmp=curr;
                  curr--;
                  return tmp;
            }
            T &operator*() {return *curr;}
            bool operator!=(myvector_reverse_iterator<T> t) {return
curr!=t.curr;}
};

template<class T>
class myvector
```

```
{
	private:
		int cap,num;
		T *buff;
	public:
		myvector() {
			cap=1;
			num=0;
			buff=new T[1];
		}
		myvector(int k,T x) {
			cap=num=k;
			buff=new T[k];
			for(int i=0;i<k;i++)
				buff[i]=x;
		}
		~myvector() {if(buff) delete[]buff;}
		int capacity() {return cap;}
		int size() {return num;}
		bool empty() {return num==0;}
		void pop_back() {if(num>0) num--;}
		void extend(int newcap)
		{
			if(newcap<cap) return;
			cap=newcap;
			T *temp=new T[cap];
			for(int i=0;i<num;i++) temp[i]=buff[i];
			if(buff)
				delete []buff;
			buff= temp;
		}
		T &back() {return buff[num-1];}
		void push_back(T x)
		{
			if(num==cap) extend(cap*2);
			buff[num++]=x;
		}
		T &operator[](int k) {return buff[k];}
		void insert(int k,T x)
		{
			if(cap==num)
				extend(cap*2);
			for(int i=num-1;i>=k;i--)
				buff[i+1]=buff[i];
			buff[k]=x;
			num++;
		}
		void erase(int k)
		{
			if (k<0 || k>size())
				return;
			for(;k<size();k++)
				buff[k] = buff[k+1];
```

```
            num--;
        }
        myvector &operator=(myvector<T> V)
        {
            this->num=V.num;
            this->cap=V.cap;
            if(cap)
            {
                this->buff=new T[cap];
                for(int i=0;i<num;i++) this->buff[i]=V.buff[i];
            }
            else this->buff=0;
            return *this;
        }
        typedef T *iterator;
        iterator begin() {return buff;}
        iterator end() {return buff+num;}
        typedef myvector_reverse_iterator<T> reverse_iterator;
        reverse_iterator rbegin()
        {
            return reverse_iterator(buff+num-1);
        }
        reverse_iterator rend()
        {
            return reverse_iterator(buff-1);
        }

};
#endif
```

2.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:
    int getmasv(){
        return masv;
    }
    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
```

3.Tạo File StuApp.cpp

```
#include"conio.h"
#include"stdio.h"
#include"iostream"
#include"myvector.cpp"
#include"student.cpp"
using namespace std;
class VectorApp  //Lop ung dung lop vector va lop VectorItr
{
      private:
            myvector <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;
}
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;
      v.insert(r,x);
      cout<<"Chen phan tu thanh cong!";
}
void VectorApp::RemoveElement()
{
      myvector<Student>::iterator it;
      int ma;
      cout<<"Nhap ma cua sinh vien can xoa bo:";
      cin>>ma;
      int k=0;
      bool found =false;
      while(k<v.size() && found==false)
            if (v[k].getmasv()==ma)
                  found = true;
            else
                  k++;
            if(found)
            {
                  v.erase(k);
                  cout<<"Phan tu bi xoa di :"<<v[k];
            }
            else
```

```
                cout<<"Khong tim thay sinh vien can xoa";
}
void VectorApp::ReplaceElement()
{
      int ma;
      Student x;
      cout<<"Nhap ma cua sinh vien can thay the:";
      cin>>ma;
      int k=0;
      bool found =false;
      while(k<v.size() && found==false)
            if (v[k].getmasv()==ma)
                  found = true;
            else
                  k++;
            if(found)
            {
                  cout<<"Nhap thong tin can thay the:";
                  cin>>x;
                  v[k]=x;
            }
            else
                  cout<<"Khong tim thay sinh vien co ma "<<ma;
}
 void VectorApp::GetElement()
 {
      myvector<Student>::iterator it;
      int ma;
      cout<<"Nhap ma cua sinh vien :";
      cin>>ma;
      int k=0;
      bool found =false;
      while(k<v.size() && found==false)
            if (v[k].getmasv()==ma)
                  found = true;
            else
                  k++;
      if(found)
      {
            cout<<"Thong tin sinh vien :"<<v[k];
      }
      else
            cout<<"Khong tim thay sinh vien";
 }
 void VectorApp::ListElement()
 {
      myvector<Student>::iterator it;
      cout<<"Danh sach cac sinh vien:\n";
      it = v.begin();
      while(it!=v.end())
      {
            cout<<*it;
            it++;
```

```
      }

}
```

4.Tạo File Demo.cpp

```
#include"conio.h"
#include"stdio.h"
#include"StuApp.cpp"
int main()
{
   VectorApp x;
   x.run();
   return 0;
}
```

5.Chạy file Demo

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. Kiểm tra tính chính xác các phương thức của lớp Single List.

Các bước làm:

1. Tạo File snode.cpp

```
#ifndef __node_slist_cpp__
#define __node_slist_cpp__
template <class T>
class snode
{
   private:
         T elem;
         snode *next;
   public:
         snode<T>()
         {
               next = 0;
         }
         snode<T>(T x, snode<T> *Next=0)
         {
               elem = x;
               next = Next;
         }
         T & getelem()
         {
               return elem;
         }
         snode<T> * getnext()
         {
               return next;
         }
         void setelem(T x)
         {
               elem = x;
         }
         void setnext(snode<T> *Next)
         {
               next = Next;
         }
#endif
```

2. Tạo File slistiterator.cpp

```
#ifndef __slidt_ite__cpp
#define __slidt_ite__cpp
#include"snode.cpp"
template <class T>
class slist_ite
{
      private:
            snode<T> *curr;
      public:
            slist_ite(snode<T> *c=0)
            {
                  curr = c;
            }
            snode<T> *getCurr()
            {
                  return curr;
            }

            slist_ite<T> &operator =(slist_ite<T> *it)
            {
                  this->curr = it->getCurr();
                  return *this;
            }
            T & operator *()
            {
                  return curr->getelem();
            }
            slist_ite<T> operator ++(int)
            {
                  curr = curr->getnext();
                  return curr;
            }
            slist_ite<T> operator ++()
            {
                  slist_ite<T> it = curr;
                  curr = curr->getnext();
                  return it;
            }
            bool operator !=(slist_ite<T> it)
            {
                  return curr != it.getCurr();
            }
};
#endif
```

3. Tạo file slist.cpp

```
#ifndef __slist__cpp__
#define __slist__cpp__
#include"snode.cpp"
#include"slistiterator.cpp"
template <class T>
class slist
```

```
{
	private:
		snode<T> *head;
		snode<T> *trail;
		int num; //So phan tu hien co cua list
	public:
		slist<T>()
		{
			head = trail = 0; //NULL
			num = 0;
		}
		slist<T>(int n, T x)
		{
			head = trail = 0; //NULL
			num = 0;
			while(n>0)
			{
				push_back(x);
				n--;
			}
		}
		bool empty()
		{
			return num == 0;
		}
		int size()
		{
			return num;
		}
		T & front()
		{
			return head->getelem();
		}
		T & back()
		{
			return trail->getelem();
		}
		void push_front(T x)
		{
			snode<T>* newnode = new snode<T>(x);
			if(newnode==0)
				return;
			if(empty())
				head = trail = newnode;
			else
			{
				newnode->setnext(head);
				head =  newnode;
			}
			num++;
		}
		void push_back(T x)
		{
```

```
      snode<T>* newnode = new snode<T>(x);
      if(newnode==0)
            return;
      if(empty())
            head = trail = newnode;
      else
      {
            trail->setnext(newnode);
            trail = newnode;
      }
      num++;
}
void pop_front()
{
      if(num==1)
      {
            delete head;
            head = trail = 0;
      }
      else
      {
            snode<T> *p = head;
            head = head->getnext();
            delete p;
      }
      num--;
}
void pop_back()
{
      if(num==1)
      {
            delete trail;
            head = trail = 0;
      }
      else
      {
            snode<T> *p = head;
            while(p->getnext()!=trail)
                  p = p->getnext();
            p->setnext(0);
            delete trail;
            trail = p;
      }
      num--;
}

typedef slist_ite<T> iterator;

iterator begin()
{
      return head;
}
iterator end()
```

```
        {
            return iterator(0);
        }

        void insert(iterator it,T x)
        {
            snode<T>* newnode =  new snode<T>(x);
            if(newnode==0)
                return ;
            if(it.getCurr()==trail)
            {
                trail->setnext(newnode);
                trail = newnode;
            }
            else
            {
                snode<T> *p= it.getCurr();
                newnode->setnext(p->getnext());
                p->setnext(newnode);
            }
            num++;
        }
        void erase(iterator it)
        {
            if(it.getCurr()==head)
                pop_front();
            if(it.getCurr()==trail)
                pop_back();
            snode<T> *p = head;
            while(p->getnext()!=it.getCurr())
                p = p->getnext();
            p->setnext(it.getCurr()->getnext());
            num--;

        }
    };

#endif
```

4. **Chạy file Demo**

```
#include"slist.cpp"
#include<bits/stdc++.h>
using namespace std;
int main()
{
  slist<int> s;
  s.push_back(100);
  s.push_back(200);
  s.push_back(300);
  s.push_back(1000);
  s.push_front(1000);
```

```
    for(slist<int>::iterator it = s.begin();
    it!=s.end(); it++)
         cout<< *it<<"\t";
}
```

Bài 3
Thực hành bài Cấu trúc danh sách liên kết kép (Double List)

Yêu cầu:

a. Cài đặt lớp Node, lớp double list, bộ lặp của lớp double List
b. Xây dựng lớp sinh viên, mỗi sinh viên có các thông tin Mã sinh viên, Họ và tên, năm sinh.
c. Kiểm tra tính chính xác các phương thức của lớp double List bằng cách tổ chức lưu trữ danh sách n sinh viên

Các bước làm:

1. Tạo File dnode.cpp

```
#ifndef __dnode__cpp__
#define __dnode__cpp__
template <class T>
class dnode
{
   private:
         T elem;
         dnode<T> *next;
         dnode<T> *prev;
   public:
         dnode<T> (T x, dnode<T> *n = 0, dnode<T> *p = 0)// 0 - NULL
         {
               elem = x;
               next = n;
               prev = p;
         }
         T & getelem() { return elem; }
         dnode<T> * getnext() { return next; }
         dnode<T> * getprev() { return prev;}
         void setnext(dnode<T> *n){ next = n; }
         void setprev(dnode<T> *p){ prev = p; }
         void setelem(T x){ elem = x; }
};

#endif
```

2.Tạo File dlistiter.cpp

```
#ifndef __dlist_ite__cpp__
#define __dlist_ite__cpp__
#include"dnode.cpp"
template <class T>
class dlist_ite
{
   private:
         dnode<T> *curr;
   public:
         dlist_ite<T>(dnode<T> *c=0)
         {
               curr = c;
         }
         dnode<T> *getcurr()
         {
               return curr;
         }
         dlist_ite<T> & operator =(dlist_ite *it)
         {
               this->curr = it->getcurr();
               return *this;
         }
         T &operator *() { return curr->getelem();}
         dlist_ite<T> operator ++(int)
         {
               curr = curr->getnext();
               return curr;
         }
         dlist_ite<T> operator ++()
         {
               dlist_ite<T> temp = curr;
               curr = curr->getnext();
               return temp;
         }
         bool operator !=(dlist_ite it)
         {
               return curr!= it.getcurr();
         }

};

template <class T>
class dlist_reite
{
   private:
         dnode<T> *curr;
   public:
         dlist_reite<T>(dnode<T> *c)
         {
               curr = c;
         }
```

```
        dnode<T> * getcurr()
        {
              return curr;
        }
        dlist_reite<T> & operator =(dlist_reite<T> *it)
        {
              this->curr = it->getcurr();
              return *this;
        }
        T &operator *() { return curr->getelem() ;}
        dlist_reite<T> operator ++(int)
        {
              curr = curr->getprev();
              return curr;
        }
        dlist_reite<T> operator ++()
        {
              dlist_reite<T> temp = curr;
              curr = curr->getprev();
              return temp;
        }
        bool operator !=(dlist_reite<T> it)
        {
              return curr!=it.getcurr();
        }

};
#endif
```

3. Tạo file dlist.cpp

```
#ifndef __dlist__cpp__
#define __dlist__cpp__
#include"dnode.cpp"
#include"dlistite.cpp"
template <class T>
class dlist
{
      private:
            dnode<T> *head;
            dnode<T> *trail;
            int num; //so phan tu hien co list
      public:
      dlist<T>()
      {
            head = trail = 0;//NULL
            num = 0;
      }
      dlist<T>(T x, int n)
      {
            head = trail = 0;
            num = 0;
            int i = 0;
```

```
      while (i<n)
      {
            push_front(x);
            i++;
      }
}
bool empty(){ return num==0;}
int size() { return num; }
T & front() { return head->getelem();}
T & back() { return trail->getelem();}
void push_front(T x)
{
      if(empty())
            head = trail = new dnode<T>(x);
      else
      {
            dnode<T> *newnode =  new dnode<T>(x);

            newnode->setnext(head);
            head->setprev(newnode);
            head = newnode;
            /*
            head =  new dnode<T>(x,head,0);
            head->getnext()->setprev(head);*/
      }
      num++;
}
void push_back(T x)
{
      if(empty())
            head = trail = new dnode<T>(x);
      else
      {
            dnode<T> *newnode =  new dnode<T>(x);
            newnode->setprev(trail);
            trail->setnext(newnode);
            trail = newnode;
      }
      num++;
}
void pop_front()
{
      if(num==1)
            head = trail = 0;
      else
            head = head->getnext();
      num--;
}
void pop_back()
{
      if(num==1)
            head = trail = 0;
      else
```

```
                trail = trail->getprev();
            num--;
        }
        //Bo lap xuoi
        typedef dlist_ite<T> iterator;
        iterator begin()
        {
            return dlist_ite<T>(head);
        }
        iterator end()
        {
            return dlist_ite<T>(0);
        }
        typedef dlist_reite<T> reverse_iterator;
        reverse_iterator rbegin()
        {
            return dlist_reite<T>(trail);
        }
        reverse_iterator rend()
        {
            return dlist_reite<T>(0);
        }
        void insert(iterator it, T x)
        {
            if(it.getcurr()==head)
                push_front(x);
            else
            {
                dnode<T> *newnode = new dnode<T>(x);
                dnode<T> *p = it.getcurr();
                dnode<T> *q = p->getprev();

                p->setprev(newnode);
                newnode->setnext(p);
                q->setnext(newnode);
                newnode->setprev(q);
                num++;
            }
        }
        void erase(iterator it)
        {
            if(it.getcurr()==head)
                return pop_front();
            if(it.getcurr()==trail)
                return pop_back();
            dnode<T> *pre = it.getcurr()->getprev();
            dnode<T> *nex = it.getcurr()->getnext();
            pre->setnext(nex);
            nex->setprev(pre);
            num--;
        }
};
#endif
```

4.Chạy file Demo

```
#include<bits/stdc++.h>
#include"dlist.cpp"
using namespace std;
class Sinhvien
{
    private:
        string ma;
        string hoten;
        int namsinh;
    public:
    friend istream &operator >>(istream & is, Sinhvien
&s)
    {

        cout<<"Ma sinh vien:";

        getline(is, s.ma);
        cout<<"Nhap ho ten:";
        getline(is, s.hoten);
        cout<<"Nhap nam sinh:";
        is>>s.namsinh;
        is.ignore(1);
        return is;
    }
    friend ostream & operator <<(ostream &os, Sinhvien
&s)
    {
        os<<s.ma<<"\t"<<s.hoten<<"\t"<<s.namsinh<<"\n";
        return os;
    }
};
int main()
{
    dlist<Sinhvien> dl;
    Sinhvien x;
    int n;
    cout<<"So sinh vien:";
```

```
    cin>>n;
    for(int i=0;i<n;i++)
    {
        cin>>x;
        dl.push_front(x);
    }
    for(dlist<Sinhvien>::iterator it = dl.begin(); it
!=dl.end(); ++it)
        cout<<*it;

}
```

Bài 4
Thực hành thuật toán sắp xếp O(n^2)

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"
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]);
        }
    }
    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;
      }
      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];
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"
#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";
	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;
}
```

Bài 5.
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);
```

```
            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>
```

```
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);
```

```
    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";
```

```
        PrintArr(a,n,0);
        BubbleSort(a,n,compare1);
        cout<<"\n";
        PrintArr(a,n,0);
        getch();
        return 0;
}
```

Bài 6.
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 kiếm 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

```
#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>
```

```
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);
     }
```

```
}
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)
                    {
```

```
                        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())
```

```
   {
         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
```

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;
}
```