Bài tập thực hành
Môn Cấu trúc Dữ liệu

- Thời lượng: 30 tiết
- Môi trường cài đặt: C-Free
- Lịch trình thực hành

Tuần (6 tiết/ tuần)	Nội dung thực hành	Ghi chú
1	Tìm kiếm	Sinh viên vắng mặt trên 2 buổi sẽ không được dự thi
2	Tìm kiếm	
3	Sắp xếp	
4	Sắp xếp	
5	Danh sách liên kết	
6	Danh sách liên kết –Kiểm tra	
7	Danh sách liên kết	
8	Danh sách liên kết	
9	Cây nhị phân tìm kiếm	
10	Cây nhị phân tìm kiếm	
11	Kiểm tra	

Lưu ý: Sinh viên phải có tài liệu này trong mỗi buổi học.

Phần I: Bài tập tìm kiếm và sắp xếp trên mảng 1 chiều (12 tiết)

Module 1

Bài 1 (02 tiết):

Viết chương trình cài đặt 2 giải thuật tìm kiếm: tuyến tính và nhị phân (*giả sử dãy số đầu vào có thứ tự tăng dần*).

Hướng dẫn: Xây dựng các hàm sau:

i) Tạo ngẫu nhiên mảng một chiều số nguyên có thứ tự tăng dần gồm N phần tử cho trước: **void PhatSinhMangTang(int a[], int N)**
ii) Xem mảng phát sinh: **void XuatMang(int a[], int N)**
iii) Tìm tuyến tính: **int TimTuyenTinh(int a[], int N, int X)**
iv) Tìm nhị phân: **int TimNhiPhan(int a[], int N, int X)**
v) Hàm chính (main()):
- Phát sinh mảng tăng a với kích thước N cho trước (*không phải sắp xếp*).
- Xuất mảng xem kết quả phát sinh.
- Nhập giá trị cần tìm x.

- Tìm x theo 2 phương pháp.
- In kết quả tìm: Nếu tìm thấy thì cho biết vị trí tìm thấy, ngược lại in kết quả không tìm thấy cho từng phương pháp.

Bài 2 (01 tiết):

Bổ sung **Bài 1** sao cho chương trình phải xác định được số lần so sánh và vị trí tìm thấy (nếu có) của phần tử cần tìm (*giả sử dãy số đầu vào có thứ tự tăng dần*).

Hướng dẫn: Thay đổi 2 hàm tìm trong **Bài 1** như sau:

i) Tìm tuyến tính có chèn vào giá trị ss tính số lần so sánh với phần tử cần tìm:
int TimTuyenTinh(int a[], int N, int X, int &ss)

ii) Tìm nhị phân có chèn vào giá trị ss tính số lần so sánh với phần tử cần tìm:
int TimNhiPhan(int a[], int N, int X, int &ss)

iii) Hàm chính (main()):
- Phát sinh mảng tăng a với kích thước N cho trước (*không phải sắp xếp*).
- Xuất mảng xem kết quả phát sinh.
- Nhập giá trị cần tìm x
- Tìm x theo 2 phương pháp
- In kết quả tìm: Gồm vị trí (nếu tìm thấy x) và số lần so sánh cho từng phương pháp.

Module 2

Bài 3 (03 tiết):

Cải tiến **Bài 2** sao cho: Nếu dãy không có thứ tự thì áp dụng phương pháp tìm tuyến tính, ngược lại dãy có thứ tự thì áp dụng phương pháp tìm nhị phân.

Hướng dẫn: Xóa hàm **PhatSinhMangTang** và bổ sung thêm một số hàm sau:

i) Tìm nhị phân cho trường hợp dãy giảm dần (*trường hợp dãy tăng dần sử dụng lại hàm **TimNhiPhan** ở **Bài 2***):
int TimNhiPhan2(int a[], int N, int X, int &ss)

ii) Kiểm tra xem mảng có thứ tự tăng? (trả về ***true***: nếu tăng, ngược lại trả về ***false***)
bool KiemTraTang(int a[], int N)

iii) Kiểm tra xem mảng có thứ tự giảm? (trả về ***true***: nếu giảm, ngược lại trả về ***false***)
bool KiemTraGiam(int a[], int N)

iv) Phát sinh mảng ngẫu nhiên, sao cho có thể tăng, giảm hoặc ngẫu nhiên
void PhatSinhMang(int a[], int N)

v) Hàm chính (main()):
- Phát sinh mảng a với kích thước N cho trước.
- Xuất mảng xem kết quả phát sinh.
- Nhập giá trị cần tìm x
- Kiểm tra nếu mảng có thứ tự tăng thì gọi hàm **TimNhiPhan**
  Ngược lại, nếu mảng có thứ tự giảm thì gọi hàm **TimNhiPhan2**
  Trường hợp còn lại thì gọi hàm **TimTuyenTinh** (*mảng không có thứ tự*)
- In kết quả như **Bài 2**

Module 3

Bài 4 (03 tiết):

Cài đặt các giải thuật sắp xếp theo các phương pháp:

1. Chọn trực tiếp.
2. Chèn trực tiếp.
3. Đổi chỗ trực tiếp.
4. Nổi bọt.
5. Quicksort.

**** Yêu cầu 1:***

- *Dữ liệu thử phát sinh ngẫu nhiên (**Dùng hàm phát sinh của Bài 3**).*
- *In ra kết quả chạy từng bước của từng giải thuật.*
- *Tính số lần so sánh và số phép gán của từng giải thuật.*

**** Yêu cầu 2:***

- *Dữ liệu thử phát sinh có thứ tự tăng dần (Dùng hàm phát sinh của **Bài 1**).*
- *In ra kết quả chạy từng bước của từng giải thuật.*
- *Tính số lần so sánh và số phép gán của từng giải thuật.*

**** Yêu cầu 3:***

- *Dữ liệu thử phát sinh có thứ tự giảm dần.*
- *In ra kết quả chạy từng bước của từng giải thuật.*
- *Tính số lần so sánh và số phép gán của từng giải thuật.*

Lập bảng sau cho các trường hợp (yêu cầu 1, 2, 3) khi chạy chương trình:

Stt	Phương pháp	Trường hợp					
		Tốt nhất (dãy tăng)		Xấu nhất (dãy giảm)		Dãy ngẫu nhiên	
		Số phép so sánh	Số phép gán	Số phép so sánh	Số phép gán	Số phép so sánh	Số phép gán
1	Đổi chỗ trực tiếp						
2	Chọn trực tiếp						
3	Chèn trực tiếp						
4	Nổi bọt						
5	QuickSort						

Module 4

Bài 5 (03 tiết): Cho mảng 1 chiều quản lý thông tin các sinh viên của 1 lớp học (tối đa 50 sinh viên). Mỗi sinh viên gồm các thông tin: MSSV, họ và tên, giới tính, địa chỉ và điểm trung bình. Viết chương trình thực hiện các yêu cầu sau:

1. Nhập các sinh viên vào danh sách.
2. In ra danh sách sinh viên.
3. Xóa 1 sinh viên với mã số x cho trước khỏi danh sách.
4. Sắp xếp danh sách sinh viên theo thứ tự tăng dần của điểm trung bình (*Dùng giải thuật sắp xếp chèn trực tiếp*).
5. Sắp xếp danh sách sinh viên theo thứ tự tăng dần của họ và tên (*Dùng giải thuật sắp xếp chọn trực tiếp*).

   Hướng dẫn:

   i) Khai báo cấu trúc thông tin sinh viên:

```
struct ttsinhvien
{   char MSSV[10], hoten[30];
    int gioitinh; //1: nữ, 0: nam
    char diachi[50];
    float dtb;
};
typedef struct ttsinhvien SINHVIEN;
```

   ii) Viết các hàm sau:

   void Nhap1SV(SINHVIEN &sv); //Nhập thông tin 1 sinh viên
   void NhapDSSV(SINHVIEN dssv[], int &n); //Nhập danh sách sinh viên
   void Xuat1SV(SINHVIEN sv); //Xuất thông tin 1 sinh viên
   void XuatDSSV(SINHVIEN dssv[], int n); //Xuất danh sách sinh viên
   int TimSV(SINHVIEN dssv[], int n, char maso[]); //Tìm sinh viên
   void XoaSV(SINHVIEN dssv[], int n, char maso[]); //Hàm xóa
   void SapTheoDTB(SINHVIEN dssv[], int n); //Sắp xếp theo điểm tb
   void SapTheoHoTen(SINHVIEN dssv[], int n); //Sắp xếp theo họ tên
   void Hoanvi(SINHVIEN &a, SINHVIEN &b); // Hoán vị 2 sinh viên

   Lưu ý: Dùng hàm ***stricmp()*** để so sánh 2 chuỗi

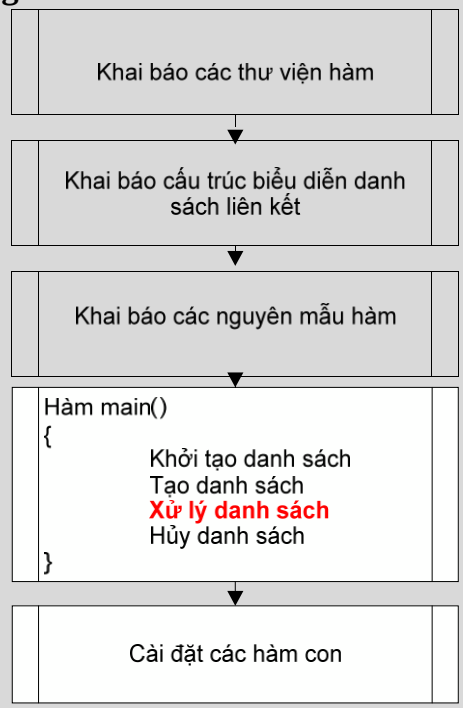
   iii) Hàm chính (main()):

   - Nhập danh sách sinh viên.
   - Xuất danh sách.
   - Nhập mã số sinh viên (x) cần xóa.
   - Xóa x.
   - Xem kết quả sau khi xóa.
   - Sắp xếp theo điểm trung bình, xuất và xem kết quả.
   - Sắp xếp theo họ tên, xuất và xem kết quả.

Module 5, 6, 7, 8

Phần II: Bài tập danh sách liên kết – ngăn xếp, hàng đợi (12 tiết)

Cấu trúc tổng quát của chương trình:



Chương trình mẫu: Nhập và xuất danh sách liên kết đơn các số nguyên

```
#include <iotream.h>
#include <stdlib.h>
struct tNODE
{
	int Key;
	struct tNODE *pNext;
};
typedef struct tNODE NODE;
struct tList
{
	NODE *pHead, *pTail;
};
typedef struct tList LIST;

void KhoiTao(LIST &l);
void Huy(LIST &l);
NODE *TaoNode(int x);
void ThemDau(LIST &l, NODE *p);
void Nhap(LIST &l);
void Xuat(LIST l);
```

```
void main()
{
	LIST l;
	Nhap(l);
	cout<<"\nDanh sach vua nhap: ";
	Xuat(l);
	Huy(l);
}
void KhoiTao(LIST &l)
{
	l.pHead=l.pTail=NULL;
}
void Huy(LIST &l)
{
	NODE *p;
	while(l.pHead)
	{
		p=l.pHead;
		l.pHead=l.pHead->pNext;
		delete p;
	}
}
NODE *TaoNode(int x)
{
	NODE *p;
	p=new NODE;
	if(p==NULL)
	{
		cout<<"Khong cap phat duoc vung nho, ket thuc";
		exit(0);
	}
	p->Key=x;
	p->pNext=NULL;
	return p;
}
void ThemDau(LIST &l, NODE *p)
{
	if(l.pHead==NULL)
		l.pHead=l.pTail=p;
	else
	{
		p->pNext=l.pHead;
		l.pHead=p;
	}
}
void Nhap(LIST &l)
{
	int x; NODE
	*p;
	KhoiTao(l);
	do{
		cout<<"Nhap gia tri vao danh sach (Nhap 0 ket thuc): ";
		cin>>x;
		if(x==0)
			break;
		p=TaoNode(x);
		ThemDau(l,p);
	}while(true);
}
```

```
void Xuat(LIST l)
{
	NODE *p=l.pHead;
	while(p)
	{
		cout<<p->Key<<" ";
		p=p->pNext;
	}
}
```

Bài 1: Cho danh sách liên kết đơn gồm các phần tử là số nguyên, viết chương trình thực hiện các yêu cầu sau:

1. Thêm một phần tử vào đầu danh sách.
   void ThemDau(LIST &l, NODE *p);
2. Xuất danh sách ra màn hình.
   void Xuat(LIST l);
3. Liệt kê các phần tử mang giá trị chẵn.

```
void XuatChan(LIST &l)
{
	NODE *p=l.pHead;
	while(p)
	{
		Nếu p->Key chẵn
			in giá trị p->Key
		p=p->pNext;
	}
}
```

4. Tìm phần tử có giá trị lớn nhất.

```
NODE *TimMax(LIST l)
{
	NODE *pmax=l.pHead;
	for(NODE *p=l.pHead->pNext; p; p=p->pNext)
		Nếu giá trị của pmax < giá trị của p thì
			gán lại pmax = p;
	return max;
}
```

5. Đếm số lượng số nguyên tố trong danh sách.
   bool LaSNT(int x); //Kiểm tra x có phải là số nguyên tố
   int DemSNT(LIST l);//Đếm số lượng số nguyên tố trong danh sách
6. Thêm phần tử có giá trị nguyên X vào trước phần tử có giá trị chẵn đầu tiên trong danh sách. Nếu không có phần tử chẵn thì thêm vào đầu danh sách.
   NODE *TimChanDau(LIST l);//Tìm chẵn đầu trong danh sách
   void ThemkTruocp(LIST &l, NODE *p, NODE *k);//Thêm k vào trước p
   void ThemXTruocChanDau(LIST &l, int X)//Thêm X vào trước chẵn đầu

```
{	NODE *k=TaoNode(X);//Phần tử cần thêm
	NODE *p=TimChanDau(l);//Node có giá trị chẵn đầu tiên
	if(p==NULL)
		ThemDau(l, k);
		else                                    }
```

ThemkTruocp(l p, k);

Ví dụ cách sử dụng hàm ThemXTruocChanDau()

```
void main()
{
	LIST l;
	int x;
	Nhap(l);
	cout<<"Danh sach vua nhap: \n";
	Xuat(l);
	cout<<"\nNhap gia tri can them vao truoc chan dau: ";
	cin>>x;
	ThemXTruocChanDau(l, x);
	cout<<"\nDanh sach sau khi them vao truoc chan dau:\n";
	Xuat(l);
}
```

7. Thêm phần tử có giá trị nguyên X vào sau phần tử có giá trị lẻ cuối cùng trong danh sách. Nếu không có phần tử lẽ thì thêm vào cuối danh sách.
   NODE *TimLeCuoi(LIST l);//Tìm lẻ cuối cùng trong danh sách **void ThemCuoi(LIST &l, NODE *p);**//Thêm p vào cuối danh sách **void ThemkSaup(LIST &l, NODE *p, NODE *k);**//Thêm k vào sau p **void ThemXSauLeCuoi(LIST &l, int X);**//Thêm X vào sau lẻ cuối
8. Xóa phần tử nhỏ nhất trong danh sách (Nếu trùng chỉ xóa phần tử nhỏ nhất đầu tiên).
   NODE *TimMin(LIST l);//Tìm node có giá trị nhỏ nhất
   void XoaDau(LIST &l);//Xóa node đầu của danh sách
   void XoaCuoi(LIST &l);//Xóa node cuối của danh sách
   void Xoap(LIST &l, NODE *p);//Xóa node p
   void XoaMin(LIST &l);//Xóa phần tử nhỏ nhất trong danh sách
9. Nhập vào phần tử X, xóa phần tử đứng sau và đứng trước phần tử X trong danh sách.
   NODE *TimX(LIST l, int X);//Tìm X
   void XoakTruocp(LIST &l, NODE *p, NODE *k);//Xóa k trước p
   void XoakSaup(LIST &l, NODE *p, NODE *q);//Xóa k sau p
10. Tách danh sách thành 2 danh sách, sao cho:
    - Danh sách thứ nhất chứa các phần tử là số nguyên tố.
    - Danh sách thứ hai chứa các phần tử còn lại.

```
void Tach(LIST l, LIST &l1, LIST &l2)
{
	KhoiTao(l1);
	KhoiTao(l2);
	NODE *p=l.pHead, *pAdd;
	while(p)
	{
		int k = p->Key;
		pAdd=TaoNode(k);
		Nếu k là số nguyên tố thì
			ThemDau(l1, pAdd);
		Ngược lại
			ThemDau(l2, pAdd);
		p trỏ đến node kế tiếp
	}
}
```

Bài 2: Cho 2 danh sách liên kết đơn l1 và l2 gồm các phần tử là số nguyên, viết chương trình thực hiện các yêu cầu sau:

1. Sắp xếp l1 và l2 tăng dần.
   void SapXep(LIST &l);
2. Nối l1 và l2 thành l3 sao cho l3 vẫn có thứ tự tăng dần.
   void Noi(LIST l1, LIST l2, LIST &l3);

Bài 3: Cho danh sách liên kết đơn quản lý thông tin của các sinh viên của 1 lớp học (tối đa 50 sinh viên). Mỗi sinh viên gồm các thông tin: MSSV, họ và tên, giới tính, địa chỉ và điểm trung bình. Viết chương trình thực hiện các yêu cầu sau:

1. Thêm 1 sinh viên vào danh sách.
2. In ra danh sách sinh viên.
3. Xóa 1 sinh viên với MSSV cho trước khỏi danh sách.
4. Sắp xếp danh sách sinh viên theo thứ tự tăng dần của điểm trung bình.
5. Liệt kê các sinh viên có điểm trung bình >=5.0.
6. Đếm số lượng sinh viên nam.
7. Cập nhật điểm trung bình của một sinh viên thông qua mã số sinh viên.

Bài 4 (Bài tập làm thêm): Dùng danh sách liên kết đơn để biểu diễn 2 số lớn (số có vài chục chữ số trở lên), viết chương trình thực hiện các yêu cầu sau:

1. Cộng
2. Trừ
3. Nhân
4. Chia

hai số trên.

Bài 5 (Bài tập làm thêm): Cài đặt lại câu 1 của phần II dùng danh sách liên kết kép.

Bài 6: Dùng kỹ thuật mảng để cài đặt minh họa các thao tác cơ bản: pop, push, … trên ngăn xếp (hoặc hàng đợi)

Bài 7: Ứng dụng bài 6 để cài đặt chương trình cho phép nhận vào biểu thức gồm các số, các toán tử +, -, *, /, các dấu đóng mở ngoặc và tính giá trị của biểu thức này

Ví dụ:

- Nhập biểu thức: ***(2*3)+9-(3+4)***
- In kết quả của biểu thức: ***8***

Module 9,10

Phần III: Bài tập cây nhị phân tìm kiếm (06 tiết)

Bài 1: Khai báo cấu trúc dữ liệu cây nhị phân (các node có giá trị là số nguyên) và viết chương trình thực hiện các yêu cầu sau:

1. Nhập và duyệt cây theo các thứ tự: trước, giữa và sau.
2. Tìm node có giá trị x trên cây.
3. Tìm node có giá trị nhỏ nhất.
4. Tìm node có giá trị lớn nhất.
5. Tính độ cao của cây.
6. Đếm số nút lá của cây.
7. Đếm số nút có đúng 2 cây con.
8. Đếm số nút có đúng 1 cây con.
9. Xóa nút có giá trị x.

Bài 2 (Bài tập làm thêm):Viết chương trình tạo và tra cứu từ điển Anh – Việt đơn giản.