TRƯỜNG ĐẠI HỌC VINH
VIỆN KỸ THUẬT VÀ CÔNG NGHỆ

TÀI LIỆU THỰC HÀNH

CẤU TRÚC DỮ LIỆU & GIẢI THUẬT

Giảng viên: Th.S Nguyễn Thị Uyên

Email: uyennt@vinhuni.edu.vn

Đơn vị: Viện Kỹ thuật và Công nghệ

KẾ HOẠCH THỰC HIỆN CÁC BÀI THỰC HÀNH

Tuần	Nội dung	Chuẩn đầu ra chi tiết	Bài đánh giá
Tuần 1	**Bài thực hành 1** Sử dụng các kiểu dữ liệu nền tảng	G1.3, G2.3, G8	A1, A3.1
Tuần 2	**Bài thực hành 2** Cài đặt giải thuật Đệ quy	G3.3, G3.4, G8	A1, A3.1
Tuần 3	**Bài thực hành 3** Cài đặt giải thuật sắp xếp	G4.3, G4.4, G8	A1, A3.1
Tuần 4	**Bài thực hành 3** *(tiếp)* Cài đặt giải thuật sắp xếp	G4.3, G4.4, G8	A1, A3.1
Tuần 5	**Bài thực hành 4** Cài đặt giải thuật Tìm kiếm	G4.3, G4.4, G8	A1, A3.1
Tuần 6	**Bài thực hành 5** Cài đặt các thao tác trên Danh sách liên kết	G5.3, G5.4, G8	A1, A3.1
Tuần 7	**Bài thực hành 6** Cài đặt các thao tác trên Ngăn xếp	G5.3, G5.4, G8	A1, A3.1
Tuần 8	**Bài thực hành 6** *(tiếp)* Cài đặt các thao tác trên Hàng đợi	G5.3, G5.4, G8	A1, A3.1
Tuần 9	**Bài thực hành 7** Cài đặt các thao tác trên Cây	G6.3, G6.4, G8	A1, A3.1
Tuần 10	**Bài thực hành 7** *(tiếp)* Cài đặt các thao tác trên Cây	G6.3, G6.4, G8	A1, A3.1
Tuần 11	**Bài thực hành 8** Cài đặt cấu trúc Đồ thị	G7.3, G7.4, G8	A1, A3.1
Tuần 12	**Bài thực hành 8** *(tiếp)* Cài đặt cấu trúc Đồ thị	G7.3, G7.4, G8	A1, A3.1

MỤC TIÊU CÁC BÀI THỰC HÀNH

Tuần	Nội dung	Chuẩn đầu ra chi tiết	Hoạt động đánh giá
Tuần 1	**BÀI THỰC HÀNH 1** **1.1**. Cài đặt và sử dụng các kiểu dữ liệu cơ bản và kiểu dữ liệu do người dùng định nghĩa trong C/C++ **1.2**. Cài đặt và sử dụng con trỏ trong C/C++ **1.3.** Sử dụng được ký hiệu Big O đánh giá độ phức tạp của các giải thuật cơ bản	**G1.3**. Áp dụng được các kiểu dữ liệu nền tảng và các cấu trúc điều khiển tuần tự; rẽ nhánh; lặp vào bài toán thực. **G2.3**. Áp dụng được ký hiệu "Big O" để ghi ra độ phức tạp của giải thuật cấu thành từ các cấu trúc điều khiển: tuần tự, rẽ nhánh và lặp.	- Đánh giá trên thái độ tham gia thí nghiệm, chuyên cần, coding style, và bài giải của sinh viên trên file nộp ở hệ thống LMS hay file nộp trực tiếp.
Tuần 2	**BÀI THỰC HÀNH 2** **2.1**. Cài đặt và sử dụng giải thuật đệ quy **2.2**. Cài đặt và sử dụng giải thuật đệ quy quay lui	**G3.3**. Áp dụng được giải thuật đệ quy, đệ quy quay lui để phân tích một số bài toán cơ bản **G3.4**. Phát triển được giải thuật đệ quy cho các phương thức cần thiết của các cấu trúc dữ liệu nền tảng.	-
Tuần 3	**BÀI THỰC HÀNH 3** **3.1**. Cài đăt giải thuật sắp xếp chèn, chọn. **3.2**. Phân tích và đánh giá các giải thuật sắp xếp	**G4.3**. Hiện thực được các giải thuật sắp xếp, tìm kiếm bằng ngôn ngữ lập trình C/C++. **G4.4**. Áp dụng được giải thuật sắp xếp và tìm kiếm trong bài toán thực.	-
Tuần 4	**BÀI THỰC HÀNH 3 (tiếp)** **3.3**. Cài đăt giải thuật sắp xếp nổi bọt, nhanh, vun đống **3.4**. Phân tích và đánh giá các giải	**G4.3**. Hiện thực được các giải thuật sắp xếp, tìm kiếm bằng ngôn ngữ lập trình C/C++. **G4.4**. Áp dụng được giải thuật	-

	thuật sắp xếp	sắp xếp và tìm kiếm trong bài toán thực.	
Tuần 5	**BÀI THỰC HÀNH 4** **4.1**. Cài đăt giải thuật tìm kiếm tuần tự và nhị phân **3.4**. Phân tích và đánh giá các giải thuật tìm kiếm	**G4.3**. Hiện thực được các giải thuật sắp xếp, tìm kiếm bằng ngôn ngữ lập trình C/C++. **G4.4**. Áp dụng được giải thuật sắp xếp và tìm kiếm trong bài toán thực.	-
Tuần 6	**BÀI THỰC HÀNH 5** **5.1** Cài đặt các thao tác cơ bản trên cấu trúc dữ liệu danh sách liên kết đơn, đôi **5.2**. Sử dụng danh sách liên kết đơn và đôi để giải quyết bài toán thực tế.	**G5.3**. Hiện thực được các cấu trúc danh sách, ngăn xếp và hàng đợi bằng ngôn ngữ lập trình C/C++. **G5.4.** Áp dụng được danh sách liên kết, ngăn xếp và hàng đợi để giải quyết bài toán thực.	-
Tuần 7	**BÀI THỰC HÀNH 6** **6.1** Thực hiện các thao tác cơ bản trên ngăn xếp **6.2**. Sử dụng ngăn xếp để giải quyết bài toán thực tế	**G5.3**. Hiện thực được các cấu trúc danh sách, ngăn xếp và hàng đợi bằng ngôn ngữ lập trình C/C++. **G5.4.** Áp dụng được danh sách liên kết, ngăn xếp và hàng đợi để giải quyết bài toán thực.	-
Tuần 8	**BÀI THỰC HÀNH 6 (tiếp)** **6.1** Thực hiện các thao tác cơ bản trên hàng đợi **6.2**. Sử dụng hàng đợi để giải quyết bài toán thực tế	**G5.3**. Hiện thực được các cấu trúc danh sách, ngăn xếp và hàng đợi bằng ngôn ngữ lập trình C/C++. **G5.4.** Áp dụng được danh sách liên kết, ngăn xếp và hàng đợi để giải quyết bài toán thực.	-

Tuần 9	**BÀI THỰC HÀNH 7** **7.1.** Cài đặt các thao tác cơ bản cấu trúc Cây nhị phân tìm kiếm **7.2.** Áp dụng cấu trúc Cây nhị phân tìm kiếm trong bài toán thực tế	**G6.3**. Hiện thực được các cấu trúc cây nhị phân và cây AVL bằng C/C++. **G6.4**. Áp dụng được cây nhị phân và cây AVL để giải quyết bài toán thực.	-
Tuần 10	**BÀI THỰC HÀNH 7 (tiếp)** 7.3. Cài đặt các thao tác trên Cây nhị phân AVL 7.4. Áp dụng cấu trúc Cây AVL trong bài toán thực tế	**G6.3**. Hiện thực được các cấu trúc cây nhị phân và cây AVL bằng C/C++. **G6.4**. Áp dụng được cây nhị phân và cây AVL để giải quyết bài toán thực.	-
Tuần 11	**BÀI THỰC HÀNH 8** 8.1. Cài đặt các thao tác trên đồ thị	**G7.3.** Hiện thực được cấu trúc Đồ thị bằng ngôn ngữ lập trình C/C++. **G7.**4. Áp dụng được lý thuyết Đồ thị để giải quyết một số bài toán thực tế.	
Tuần 12	**BÀI THỰC HÀNH 8 (tiếp)** 8.2. Một số bài toán tiêu biểu trên đồ thị	**G7.3.** Hiện thực được cấu trúc Đồ thị bằng ngôn ngữ lập trình C/C++. **G7.4.** Áp dụng được lý thuyết Đồ thị để giải quyết một số bài toán thực tế.	

BÀI THỰC HÀNH 1

KIỂU DỮ LIỆU NỀN TẢNG

1. **Mục tiêu**: Ôn tập và củng cố các kiến thức cơ bản đã học ở Ngôn ngữ lập trình C.
2. **Kỹ năng**: Sử dụng thành thạo Ngôn ngữ lập trình C.
3. **Yêu cầu**: Áp dụng được các kiểu dữ liệu cơ bản để giải quyết một số bài toán
4. **Thời lượng thực hành:** 2 tiết
5. **Tóm lược lý thuyết**

5.1. Các kiểu dữ liệu cơ sở

STT	Mô tả
1	***Kiểu cơ sở:*** Là các kiểu dữ liệu số học và bao gồm 2 kiểu chính: a) kiểu số nguyên và b) kiểu số thực dấu chấm động
2	***Kiểu liệt kê:*** là các kiểu số học và được dùng để định nghĩa các biến mà nó có thể được gán trước một số lượng nhất định giá trị số nguyên qua suốt chương trình.
3	***Kiểu void:*** Kiểu định danh *void* là kiểu đặc biệt thể hiện rằng không có giá trị nào.
4	***Kiểu dữ liệu khác*** a) Con trỏ b) Kiểu mảng, c) Kiểu cấu trúc, d) Kiểu union và e) Kiểu function (hàm).

5.2. Bài thực hành

Bài 1

Viết chương trình nhập vào 2 phân số a/b và c/d. Hãy tính tổng của phân số này, yêu cầu là phân số kết quả phải ở dạng tối giản.

Ví dụ :1/6+1/3=1/2

Bài 2

Viết một hàm đảo ngược thứ tự các phần tử của một mảng số nguyên.

Ví dụ: mảng nhập vào 1 2 3 4 5 7 9 10. Sau khi đảo mảng thành 10 9 7 5 4 3 2 1.

Bài 3

Viết chương trình nhập vào một mảng số tự nhiên. Hãy xuất ra màn hình:

- Dòng 1: gồm các số lẻ, tổng cộng có bao nhiêu số lẻ.
- Dòng 2: gồm các số chẵn, tổng cộng có bao nhiêu số chẵn.
- Dòng 3: gồm các số nguyên tố.
- Dòng 4: gồm các số không phải là số nguyên tố

Bài 4

Viết chương trình nhập vào một mảng, hãy xuất ra màn hình:

- Phần tử lớn nhất của mảng.
- Phần tử nhỏ nhất của mảng.
- Tính tổng của các phần tử trong mảng
- Tính trung bình cộng
- Tính tổng của các phần tử là số nguyên tố trong mảng
- Tính số lượng phần tử là số nguyên tố trong mảng
- Phần tử âm lớn nhất của mảng.
- Phần tử dương nhỏ nhất của mảng.
- Tổng các phần tử có căn bậc hai nguyên
- Đếm có bao nhiêu số lẻ.
- Đếm có bao nhiêu số chẵn.
- Kiểm tra tính đối xứng của mảng
- Tìm phân tử là số nguyên tố đầu tiên trong mảng.

BÀI THỰC HÀNH 2

GIẢI THUẬT ĐỆ QUY

1. Mục tiêu: Giới thiệu kỹ thuật phân tích và thiết kề giải thuật đệ qui, cách hoạt động, thiết kế giải thuật đệ qui và cài đặt các hàm đệ qui. Đồng thời hướng dẫn sinh viên thực hiện giải quyết các bài toán cơ bản trên ngôn ngữ lập trình C/C++

2. Kỹ năng: Biết lựa chọn phương pháp lưu trữ thích hợp và giải thuật cho từng bài toán. Đồng thời giúp sinh viên củng cố và phát triển kỹ năng phân tích và thiết kế một giải thuật đệ quy và cài đặt được bằng ngôn ngữ lập trình C/C++.

3. Yêu cầu: Sử dụng Giải thuật đệ qui đối với từng bài tập, sinh viên nêu được tên hàm, danh sách tham biến (tham trị), trình bày các bước, các thành phần của Giải thuật đệ qui, sử dụng lời gọi hàm đệ qui trong chương trình chính. Nắm được ưu nhược điểm của Giải thuật đệ quy.

4. Thời lượng thực hành: 2 tiết

5.Tóm lược lý thuyết

5.1. Khái niệm Đệ quy

Một đối tượng được gọi là đệ quy nếu nó được mô tả thông qua định nghĩa của chính nó. Nghĩa là, các đối tượng này được định nghĩa một cách quy nạp từ những khái niệm đơn giản nhất cùng dạng với nó.

Ví dụ: Hàm tính n!

a. *0! = 1*
b. *Nếu n>0, thì n! = n * (n-1)!*

5.2. Khái niệm Giải thuật Đệ quy

Nếu lời giải của một bài toán T được thực hiện bằng lời giải của một bài toán T', có dạng giống như T thì đó là một lời giải đệ qui. Giải thuật tương ứng với lời giải đệ qui được gọi là giải thuật đệ qui. Bài toán T' tuy có dạng giống T nhưng theo một nghĩa nào đó nó phải "nhỏ" hơn T.

5.3. Khái niệm Hàm Đệ quy

Định nghĩa của hàm đệ qui hay thủ tục đệ qui bao gồm hai phần sau:

1. ***Phần neo*** (hay Phần cố định), ứng với trường hợp suy biến. Trong đó tác động của hàm được đặc tả cho một hay nhiều tham số.
2. ***Phần đệ qui*** (hay Phần qui nạp, Phần hạ bậc), trong đó tác động cần được thực hiện cho giá trị hiện thời của các tham số được định nghĩa bằng các tác động hay giá trị được định nghĩa trước đây.

Ví dụ. *Tính n!*

- *Phần neo : 0! = 1*
- *Phần đệ qui : Với n > 0: n! = n*(n-1)!*

4.4. Kỹ thuật thiết kế

Thực hiện 3 bước sau:

Bước 1: Tham số hóa bài toán

Bước 2: Tìm điều kiện dừng

Bước 3: Phân rã bài toán

Ví dụ: *Tính n!.*

Ta sẽ phân tích theo 3 bước như sau:

Bước 1: Tham số hóa bài toán

- Tìm các thông số biểu thị kích thước của bài toán
- Quyết định độ phức tạp của bài toán
- ***N tính trong hàm giai thừa của N***

Bước 2: Tìm điều kiện dừng

- Là trường hợp giải không đệ quy
- Là trường hợp kích thước bài toán nhỏ nhất
- ***0!=1 là điều kiện dừng***

Bước 3: Phân rã bài toán

- Hoặc không đệ quy
- Hoặc là bài toán trên nhưng kích thước nhỏ hơn

6. Bài tập thực hành

Bài 1

Sử dụng hàm viết dạng đệ qui tính Tổng=1+2+…+n, tính và in ra tổng S=1+(1+2)+….+(1+2+…+n), với n được nhập từ bàn phím.

Gợi ý:

```
1. int Tong(int n)
2. {
3. if(n==1) return 1;
4. else return (n + Tong(n-1));
5. }
6. for(i=1;i<=n;i++) s=s+Tong(i);
```

Bài 2

Nhập 2 số a và n. Tính S =$a^1 + a^2 + a^3 + \ldots + a^n$.

Gợi ý:

```
1. float Mu( float a, int
2. {if(n == 0)return 1;
3. return float Tong(float a ,int n)
4. if(n == 1)return a;
5. else return Tong(a,n-1) + Mu(a,n-1)*a;
```

Bài 3

Sử dụng hàm viết dạng đệ qui Tổng=1+3+…+2n-1, tính và in ra tổng S=1+(1+3)+….+(1+3+…+2n-1), với n được nhập từ bàn phím.

Gợi ý:

```
1. int Tong(int n)
2. {
3. if(n==1) return 1;
4. else return (2*n-1 + Tong(n-1));
5. for(i=1;i<=n;i++) s=s+Tong(i);
```

Bài 4

Sử dụng hàm viết dạng đệ qui Tổng=1/2+1/4+…+1/2n, tính và in ra tổng với n được nhập từ bàn phím.

Gợi ý:

```
float Tong(float n)
{if(n==1) return 0.5;
else return (1/(2*n) + Tong(n-1))};
```

Bài 5

Sử dụng hàm viết dạng đệ qui Tổng=$1^2+2^2+…+n^2$, tính và in ra tổng S=$1/1^2+1/(1^2+2^2)+…+1/(1^2+2^2+…+n^2)$, với n được nhập từ bàn phím.

Gợi ý:

```
float Tong(float n)
{if(n==1) return 1;
else return (1/(Tong(n-1)+n*n)+Tong(n-1));
}
```

Bài 6

Sử dụng hàm viết dạng đệ qui. Tổng=1/2+3/4+5/6+... +(2n+1)/(2n+2), tính và in ra Tổng với n được nhập từ bàn phím.

Gợi ý:

```
float Tong(float n)
{if(n==0) return 0.5;
 return ((2*n+1)/(2*n+2)+ Tong(n-1));
```

Bài 7

Sử dụng hàm viết dạng đệ qui tính Giaithua(n), tính và in ra S=1 + 1.2 + 1.2.3+.... + 1.2.3....n, với n được nhập từ bàn phím.

Gợi ý:

```
1. long Giaithua(int n) {if(n==0) return 1;
2. else return (n*Giaithua(n-1));
3. }
4. long Tong(int n)
5. if(n ==  return
6. else return Tong(n-1) + Giaithua(n-1)*n;
```

Bài 8

Viết chương trình nhập mảng các số nguyên gồm n phần tử. Sử dụng kỹ thuật đệ qui để xây dựng hàm tính tổng các phần tử của mảng.

Gợi ý:

```
1. int tong( int a[], int n)
2. { if(n==1) return a[0];
3. return (a[n-1]+tong(a,n-1));
```

Bài 9

Viết chương trình Tính $S(x,n) = x + x^2 + x^3 + ... + x^n$

Gợi ý:

```
1. float LuyThua(float x , int n) { if(n == 0)
   { return 1; }
2. return LuyThua(x,n-1)*x; }
3. float Tong(float x , int n) { if(n == 1) {
   return x; }
4. return Tong(x,n-1) + LuyThua(x,n-1)*x; }
```

Bài 10

Viết chương trình Tính $S(x,n) = x + x^2 + x^4 + ... + x^{2n}$

Gợi ý:

```
1. double LuyThua2(int x, int n)
2.
3. { if (n==1)  return pow(x,2*n);
4. }
5. return bai742(x,n-1) + pow(x,2*n); }
```

BÀI THỰC HÀNH 3

GIẢI THUẬT SẮP XẾP CƠ BẢN

1. Mục tiêu: Giúp sinh viên hiểu và áp dụng được các giải thuật sắp xếp cơ bản như: sắp xếp chèn, chọn, nổi bọt, vun đống vào một số bài toán cụ thể.

2. Kỹ năng: Cài đặt được các giải thuật sắp xếp đơn giản. Đồng thời tính toán được thời gian thực hiện của mỗi giải thuật.

3. Yêu cầu: Viết chương trình thực hiện minh họa một số giải thuật sắp xếp cơ bản.

4. Thời lượng thực hành: 4 tiết

5. Tóm lược lý thuyết:

- **Bài toán sắp xếp:** Xét một dãy a gồm N phần tử. Cần sắp xếp các phần tử của dãy để được một dãy có thứ tự (ví dụ tăng dần/giảm dần theo một khóa được chỉ định).
- **PHƯƠNG PHÁP SẮP XẾP CHỌN**

Ý tưởng:

- Chọn phần tử nhỏ nhất x trong N phần tử ban đầu, đảo vị trí của x với phần tử đầu dãy để đưa x về đầu dãy.
- Ta không quan tâm đến phần tử đầu dãy nữa, xem dãy bây giờ chỉ còn *N*-1 phần tử, bắt đầu từ vị trí thứ 2.
- Lặp lại xử lí trên cho đến khi dãy hiện hành chỉ còn một phần tử.

Ví dụ: Cho dãy gồm các số sau: 14, 33, 10, 35, 19, 42, 44. Hãy sắp xếp dãy theo thứ tự tăng dần sử dụng giải thuật sắp xếp chọn.

Khởi tạo	14	33	27	10	35	19	42	44
Bước 1	10	33	27	14	35	19	42	44
Bước 2	10	14	27	33	35	19	42	44
Bước 3	10	14	19	33	35	27	42	44
Bước 4	10	14	19	27	35	33	42	44
Kết quả	**10**	**14**	**19**	**27**	**33**	**35**	**42**	**44**

- PHƯƠNG PHÁP SẮP XẾP CHÈN

Ý tưởng

- Xét dãy a_1, a_2…., a_n, trong đó i-1 phần tử đầu tiên a_1, a_2,…,a_n đã có thứ tự. Tìm vị trí thích hợp để chèn phần tử ai vào vị trí thích hợp trong i-1 phần tử đã sắp để có dãy mới a_1, a_2,…,a_i trở nên có thứ tự.
- Vị trí này nằm giữa a_{k-1} và a_k thỏa $a_{k-1} \le a_i < a_k$ $(1 \le k < i)$.

Ví dụ: Cho dãy sau: {14, 33, 27, 10, 35, 19, 42, 44}. Hãy sắp xếp dãy theo thứ tự tăng dần sử dụng giải thuật sắp xếp chèn.

Khởi tạo	14	33	27	10	35	19	42	44
Bước 1	14	33	27	10	35	19	42	44
Bước 2	14	27	33	10	35	19	42	44
Bước 3	10	14	27	33	35	19	42	44
Bước 4	10	14	19	27	33	35	42	44
Kết quả	**10**	**14**	**19**	**27**	**33**	**35**	**42**	**44**

- PHƯƠNG PHÁP SẮP XẾP NỔI BỌT

Ý tưởng:

- Xét dãy gồm *N* phần tử
- Xuất phát từ cuối dãy, đổi chỗ các cặp phần tử kế cận để đưa phần tử nhỏ hơn trong cặp phần tử đó về vị trí đầu dãy hiện hành.
- Ta không quan tâm đến phần tử đầu dãy nữa, xem dãy bây giờ chỉ còn *N*-1 phần tử, bắt đầu từ vị trí thứ 2.
- Lặp lại xử lí trên cho đến khi không còn cặp phần tử nào để xét.

Ví dụ: Cho dãy sau: {14, 33, 27, 10, 35, 19, 42, 44}. Hãy sắp xếp dãy theo thứ tự tăng dần sử dụng giải thuật sắp xếp nổi bọt.

Khởi tạo	14	33	27	10	35
Bước 1	14	27	33	10	35
Bước 2	14	27	10	33	35
Bước 3	14	10	27	33	35
Bước 4	10	14	27	33	35
Kết quả	**10**	**14**	**27**	**33**	**35**

- PHƯƠNG PHÁP SẮP XẾP ĐỔI CHỖ TRỰC TIẾP

Ý tưởng:

- Ý tưởng chính của giải thuật là xuất phát từ đầu dãy, tìm tất cả nghịch thế chứa phần tử này, triệt tiêu chúng bằng cách đổi chổ phần tử này với phần tử tương ứng trong cặp nghịch thế.
- Lặp lại xử lý trên với các phần tử kế tiếp theo trong dãy.

Ví dụ: *Cho dãy sau: {12, 4, 8, 9, 2}. Hãy sắp xếp dãy theo thứ tự tăng dần sử dụng giải thuật sắp xếp đổi chỗ trực tiếp.*

Khởi tạo	12	4	8	9	2
Bước 1	2	12	8	9	4
Bước 2	2	4	12	9	8
Bước 3	2	4	8	12	9
Bước 4	2	4	8	9	12
Kết quả	**2**	**4**	**8**	**9**	**12**

- PHƯƠNG PHÁP SẮP XẾP NHANH

Ý tưởng

- Chọn *x* là giá trị của một phần tử tùy ý trong dãy ban đầu. Vị trí phần tử thường được chọn là *k* = (*l* + *r*)/2.
- Thực hiện phân hoạch với *x*:
- Dãy ban đầu được chia làm 3 phần, trong đó dãy con thứ 2 đã có thứ tự. Dãy ban đầu chỉ có thứ tự nếu dãy con 1 và 3 cũng có thứ tự.
- Nếu dãy con 1 và 3 chỉ có một phần tử thì chúng đã có thứ tự, ngược lại, ta lần lượt tiến hành phân hoạch từng dãy con theo phương pháp phân hoạch như trên

Ví dụ: Cho dãy sau: {28, 16, 56, 30, 17, 32, 24, 18}. Chốt là phần tử ở giữa dãy= ([left]+[right])/2=(0+7)/2=A[3]=30

A[i]	0	1	2	3	4	5	6	7
Bước 1	28	16	56	30	17	32	24	18
Bước 2	28	16	18	30	17	32	24	56
Bước 3	28	16	18	24	17	32	30	56

Bước 4	**28**	**16**	**18**	**24**	**17**	30	**32**	**56**
Kết quả	**16**	**17**	**18**	**24**	**28**	30	**32**	**56**

- **PHƯƠNG PHÁP SẮP XẾP VUN ĐỐNG**

Ý tưởng:

- Xét dãy gồm N phần tử.
- Hiệu chỉnh dãy thành heap. Đảo vị trí của phần tử đầu dãy với phần tử cuối dãy.Để đưa phần tử lớn nhất về cuối dãy.
- Ta không quan tâm đến phần tử cuối dãy nữa, xem như dãy hiện hành chỉ gồm N-1 phần tử, tính từ 1.
- Lặp lại xử lí trên cho đến khi dãy hiện hành chỉ còn một phần tử.

Ví dụ: Cho mảng a= {2, 3, 5, 6, 4, 1, 7}, Ở đây n = 7. Các phần tử từ a[4] đến a[7].

Tạo đống							
Dãy	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]
Khởi tạo dãy	**2**	**3**	**5**	**6**	**4**	**1**	**7**
Vun gốc a[3]	2	3	7	6	4	1	5
Vun gốc a[2]	2	6	7	3	4	1	5
Vun gốc a[1]	7	6	5	3	4	1	2
		Heapsort					
Đổi chỗ **a[1]** với **a[7]**	2	6	5	3	4	1	7
Vun lại mảng **a[1..6]**	6	4	5	3	2	1	7
Đổi chỗ **a[1]** với **a[6]**	1	4	5	3	2	6	7
Vun lại mảng **a[1..5]**	5	4	1	3	2	6	7
Đổi chỗ **a[1]** với **a[5]**	2	4	1	3	5	6	7
Vun lại mảng **a[1..4]**	4	3	1	2	5	6	7
Đổi chỗ **a[1]** với **a[4]**	2	3	1	4	5	6	7
Vun lại mảng **a[1..3]**	3	2	1	4	5	6	7
Đổi chỗ **a[1]** với **a[3]**	2	1	3	4	5	6	7
Vun lại mảng **a[1..2]**	1	2	3	4	5	6	7
Kết quả	**1**	**2**	**3**	**4**	**5**	**6**	**7**

6. Bài tập thực hành

Bài 1

Viết chương trình thực hiện giải thuật sắp xếp một dãy gồm n nguyên bằng phương pháp **sắp xếp chọn (Selection Sort).** Với n nhập từ bàn phím. In ra màn hình từng bước của giải thuật.

Gợi ý:

```
1. void Selectionsort(int a[], int n)
2. {
3. for (int i = 0; i < n - 1; i++)
4. {
5. int min = i;
6. for (int j = i + 1; j < n; j++)
7. if (a[min]>a[j])
8. min = j;
9. swap(a[i], a[min]);
10.  }
11.  }
```

Bài 2

Viết chương trình thực hiện giải thuật sắp xếp một dãy gồm n nguyên bằng phương pháp **sắp xếp chèn (Insertion sort**). Với dữ liệu được đọc từ tệp. In kết quả từng bước của giải thuật vào một tệp khác.

Gợi ý:

```
1. void InsertionSort(int a[], int n)
2. {
3. for (int i = 1; i < n; i++)
4. {
5. int x = a[i];
6. int j = i-1;
7. while (j >=0 && a[j ] > x) // tăng dần {
8. a[j+1] = a[j];
9. j--;
10.  }
11.  a[j+1] = x;
12.  }}
```

Bài 3

Viết chương trình thực hiện giải thuật sắp xếp một dãy gồm n nguyên bằng phương pháp **sắp xếp nổi bọt (Bubble sort**). Với n nhập từ bàn phím. In ra màn hình từng bước của giải thuật.

Gợi ý:

```
1. void Bubblesort(int a[], int n)
```

```
2. {
3. int i, j;
4. for (int i = 1; i<n - 1; i++)
5. {
6. for (int j = n - 1; j-1>i; j--)
7. {
8. if (a[j] < a[j - 1])
9. {
10.  swap(a[j], a[j - 1]);
11.  }
```

Bài 4

Viết chương trình thực hiện giải thuật sắp xếp một dãy gồm n nguyên bằng phương pháp **sắp xếp nhanh (Quick sort)**. Với n nhập từ bàn phím. In ra màn hình từng bước của giải thuật.

Gợi ý:

```
1. void quickSort(int *a, int l, int r)
2. srand(time(NULL));   //khoi tao tham so ham
   rand()
3. int key = a[l + rand() % (r-l+1)]; //lay khoa
   la gia tri ngau nhien tu a[l] ->
4. a[r]
5. //int key = a[(l+r)/2];
6. int i = l, j = r;
7. if (i < r) quickSort(a, i, r); // lam lai voi
   mang a[i]....a[r]
8. }
```

Bài 5

Viết chương trình thực hiện giải thuật sắp xếp một dãy gồm n nguyên bằng phương pháp **sắp xếp vun đống (Heap sort)**. Với n nhập từ bàn phím. In ra màn hình từng bước của giải thuật.

Gợi ý:

```
1. voi builHeap((A, int last)

2. {
3. for(int i=last/2;i>0;i--)
4. maxHeapify(A,last);
5. }
6. }
-------------------------------------
1. void maxHeapify(A,parent,last)
2.
3. {
4. int child=2*parent;
5. While (child<=last)
6. {
7. if(child+1<=last&&A[child+1]>A[child])
8. Child ++;
9. if(A[child]>A[parent])
```

```
10.  ĐổiChỗ(A[child], A[parent]);
11.  Parent=child;
12.  Child=2*parent;
```

Bài 6

Mỗi sinh viên được quản lý ở một hệ thống trường X được bao gồm: *họ và tên, năm sinh và địa chỉ email*. Để dễ dàng trong việc quản lý, hãy viết một chương trình thực hiện sắp xếp danh sách sinh viên theo thứ tự tăng dần theo trường *họ và tên*. (*Lựa chọn một trong các phương pháp sắp xếp đã học).*

BÀI THỰC HÀNH 4

GIẢI THUẬT TÌM KIẾM

1. Mục tiêu: Giúp sinh viên hiểu và áp dụng được các giải thuật tìm kiếm cơ bản như: tìm kiếm tuần tự và tìm kiếm nhị phân.

2. Kỹ năng: Cài đặt được các giải thuật tìm kiếm đơn giản. Đồng thời tính toán được thời gian thực hiện của giải thuật tìm kiếm cơ bản.

3. Yêu cầu: Viết chương trình thực hiện minh họa một số giải thuật tìm kiếm cơ bản.

4. Thời lượng thực hành: 2 tiết

5. Tóm lược lý thuyết:

- **Bài toán sắp xếp:** Cho dãy a gồm N phần tử, cần tìm x trong dãy a.
- **PHƯƠNG PHÁP TÌM KIẾM TUẦN TỰ**

Ý tưởng

So sánh x lần lượt với phần tử thứ 1, thứ 2,…của dãy a cho đến khi gặp phần tử có khóa cần tìm, hoặc đã tìm hết dãy mà không thấy x.

- **PHƯƠNG PHÁP TÌM KIẾM NHỊ PHÂN**

Ý tưởng:

- Đối với những dãy số đã có thứ tự (tăng dần), các phần tử đã có quan hệ $a_{i-1} \le a_i \le a_{i+1}$. Nếu $x > a_i$ thì x chỉ có thể xuất hiện trong đoạn $[a_{i+1}, a_N]$, ngược lại nếu $x < a_i$ thì x chỉ có thể xuất hiện trong đoạn $[a_1, a_{i-1}]$.
- Giải thuật áp dụng nhận xét trên để giới hạn phạm vi tìm kiếm sau mỗi lần so sánh x với một phần tử trong dãy.
- Tại mỗi bước, so sánh x với phần tử nằm giữa dãy tìm kiếm hiện hành, dựa vào kết quả so sánh để quyết định giới hạn của dãy tìm kiếm ở bước kế tiếp là nửa trên hay nửa dưới của dãy hiện hành.

6. Bài tập thực hành

Bài 1

Viết chương trình thực hiện tìm kiếm dãy gồm n số với khóa bất kỳ bằng phương pháp **tìm kiếm tuần tự**. Với n được nhập từ bàn phìm. In ra màn hình kết quả từng bước giải thuật.=

Gợi ý:

```
1. int Timkiemtuantu(int *A, int x, int n)
2. {
3. register i,temp;
4. for (i=0; i<n ; i ++)
5. {
6. if (A[i] == X)
7. return(i);
8. }
9. return(-1);
```

Bài 2

Viết chương trình thực hiện tìm kiếm dãy gồm n số với khóa bất kỳ bằng phương pháp **tìm kiếm nhị phân**. Với n được nhập từ bàn phìm. In ra màn hình kết quả từng bước giải thuật.

Gợi ý:

```
1.   int Timkiemnhiphan(int a[], int n, int x)
2.   {
3.   int First=1, Last=n;
4.   while(First <=Last)
5.   {
6.   int Mid=(First + Last)/2;
7.   if(x==a[Mid])
8.   return Mid;
9.   if(x<a[Mid])
10.  Last=Mid-1;
11.  else
12.  First=Mid+1;
13.  }
14.  return -1;
```

Bài 3

Cho một dãy số nguyên dương a_1, a_2, ..., a_N (10 < N < 100.000), $a_i \leq 10.000$ với mọi i=1..N và một số nguyên dương S (S < 100.000.000). Viết chương trình tìm độ dài nhỏ nhất của dãy con chứa các phần tử liên tiếp của dãy mà có tổng các phần tử lớn hơn hoặc bằng S. (*Sử dụng giải thuật tìm kiếm nhị phân*).

Bài 4 BÀI TOÁN CỔ

"Vừa gà vừa chó

Bó lại cho tròn

Ba mươi sáu con

Một trăm chân chẵn

Hỏi có bao nhiêu gà bao nhiêu chó?"

Hãy áp dụng giải thuật tìm kiếm nhị phân để giải quyết bài toán trên.

Gợi ý:

```
1. d:=0; c:=36; x:=100; 2.Trong khi d<c thì
2. g:= (d+c) div 2;
3. Nếu x=2*g+4*(36-g) thì y:=c-g; Số gà là g; Số
   chó là y 2.3.Nếu x>2*g+4*(36-g) thì c:=g
4. Nếu x<2*g+4*(36-g) thì d:=g
5. Quay lại bước 2
6. Kết thúc
```

Bài 7

Mỗi sinh viên được quản lý ở một hệ thống trường X được bao gồm: *họ và tên, năm sinh và địa chỉ email*. Để dễ dàng trong việc quản lý, người ta cần một chương trình thực hiện được các thao tác cơ bản tìm kiếm:

- Thực hiện tìm kiếm tuyến tính theo tên, năm sinh và địa chỉ email.
- Thực hiện tìm kiếm nhị phân trên danh sách sau khi đã sắp xếp.

BÀI THỰC HÀNH 5

DANH SÁCH LIÊN KẾT

1. Mục tiêu: Giúp sinh viên hiểu và áp dụng được cấu trúc danh sách liên kết

2. Kỹ năng: Cài đặt được các thao tác trên danh sách liên kết

3. Yêu cầu: Thực hiện cài đặt một số thao tác và giải quyết các bài toán thực tế

4. Thời lượng thực hành: 2 tiết

5. Tóm tắt lý thuyết:

5.1. Danh sách liên kết- List

- **Danh sách liên kết đơn:**

Mỗi Node sẽ lưu trữ 2 thông tin:

- Thông tin dữ liệu: Lưu trữ các thông tìn về chính Node đó.
- Thông tin liên kết: Lưu trữ địa chỉ của phần tử kế tiếp trong danh sách, hoặc lưu trữ giá trị NULL nếu phần tử đó nằm cuối danh sách.

- **Danh sách liên kết đôi:**

Danh sách liên kết đôi (Doubly Linked List) là một biến thể của Danh sách liên kết (Linked List), trong đó hoạt động duyệt qua các nút có thể được thực hiện theo hai chiều: về trước và về sau một cách dễ dàng khi so sánh với Danh sách liên kết đơn.

- Link: mỗi link của một Danh sách liên kết có thể lưu giữ một dữ liệu và được gọi là một phần tử.
- Next: mỗi link của một Danh sách liên kết có thể chứa một link tới next link và được gọi là Next.
- Prev: mỗi link của một Danh sách liên kết có thể chứa một link tới previous link và được gọi là Prev.
- First và Last: một Danh sách liên kết chứa link kết nối tới first link được gọi là First và tới last link được gọi là Last.

- **Các thao tác trên danh sách liên kết**

1. **Tạo danh sách rỗng**
2. **Tạo và cấp phát bộ nhớ cho một nút**
3. **Thêm 1 node vào đầu danh sách liên kết**
4. **Thêm 1 node vào cuối danh sách**

5. Thêm 1 node vào vị trí bất kỳ
6. Xóa 1 node ở đầu danh sách
7. Xóa 1 node ở cuối danh sách
8. Xóa 1 node vào vị trí bất kỳ
9. Duyệt danh sách
10. Một số thao tác tìm kiếm và sắp xếp khác

6. Bài thực hành

Bài 1

Viết chương trình xây dựng và quản lý danh sách liên kết đơn. Thành phần quản lý gồm con trỏ firt và last (đầu và cuối danh sách). Hiển thị menu thực hiện các chức năng sau (mỗi chức năng thực hiện bằng hàm). Thành phần dữ liệu trong mỗi Node là giá trị kiểu số nguyên. Viết chương trình thực hiện các thao tác sau:

- Thêm một node vào đầu danh sách
- Thêm một node vào cuối danh sách
- Thêm nhiều node vào đầu danh sách
- Thêm nhiều node vào cuối danh sách
- Hiển thị giá trị node thứ n
- Tìm một node dựa theo giá trị nhập vào
- Thêm một node vào sau một node nào đó (nhập giá trị để tìm)
- Đếm số lượng node trong danh sách.
- Xóa node đầu danh sách
- Xóa node cuối danh sách
- Xóa toàn bộ danh sách

Gợi ý:

```
  //Khai báo node struct node
1. int data; node*next;
  //Khởi tạo danh sách struct dslk
2. node* head; node* tail;
  //Thủ tục tạo node node*
3. createNode(int x)
4. node *p; p=new node; if(p==NULL)return NULL;
5. p->data=x; p->next=NULL;
6. return p;
  // Hàm khởi tạo một danh sách
7. void init( dslk &l){
```

```
8. l.head=NULL;
9. l.tail=NULL;
10.    //Xóa node đầu
11.    int deleteFirst(dslk&l){
12.    if(l.head==NULL)
13.    return -1;
14.    node*p=l.head;//phan tu can xoa
15.    l.head=p->next;
16.    if(l.head==NULL)
17.    l.tail=NULL;
18.    delete p;
19.    return 1;
```

//Thêm node đầu

```
20.    void addFirst(dslk &l,node*p){
21.    if(l.head==NULL) l.head=l.tail=p;
22.    else
23.    p->next=l.head;
24.    l.head=p;
```

//Thêm node cuối

```
25.    void addLast(dslk &l,node*p){
26.    if(l.head==NULL) l.head=l.tail=p; else{
27.    l.tail->next=p;
28.    l.tail=p;
```

//Thêm node bất kỳ

```
29.    void addAfter(dslk&l,node
30.    *q,node*new_node){
31.    if(l.head==NULL&&q==NULL)
32.    l.head=l.tail=new_node;
33.    if (q!=NULL){
34.    new_node->next = q->next;
35.    q->next = new_node;
36.    if(q == l.tail) l.tail = new_node;
37.    }
```

Bài 2

Viết chương trình xây dựng và quản lý danh sách liên kết vòng đôi. Thành phần quản lý gồm con trỏ first. Hiển thị menu thực hiện các chức năng sau (mỗi chức năng thực hiện bằng hàm). Thành phần dữ liệu trong mỗi Node là giá trị kiểu số nguyên. Viết chương trình thực hiện các thao tác sau đây:

- Thêm một node vào đầu danh sách
- Thêm một node vào cuối danh sách
- Thêm nhiều node vào đầu danh sách
- Thêm nhiều node vào cuối danh sách
- Hiển thị giá trị node thứ n

Bài 3

Viết chương trình xây dựng và quản lý danh sách liên kết đơn.Thành phần quản lý gồm con trỏ first và last. Hiển thị menu thực hiện các chức năng sau (mỗi chức năng thực hiện bằng hàm). Thành phần dữ liệu trong mỗi Node là giá trị kiểu bản ghi gồm các trường: *Họ tên, mã sinh viên, Điểm.*

Viết chương trình thực hiện các thao tác sau:

- Thêm một sinh viên vào đầu danh sách
- Xóa sinh viên theo Điểm >9.
- Sắp xếp danh sách sinh viên theo thứ tự tăng dần của Điểm.

Bài 4

Viết chương trình xây dựng và quản lý danh sách liên kết đơn. Thành phần quản lý gồm con trỏ first, last. Thành phần dữ liệu trong mỗi node là thông tin một sinh viên, bao gồm các trường:

+Mã sinh viên (int)

+Họ tên sinh viên (string)

+Lớp (string)

+Điểm Toán (float)

+Điểm Lý (float)

+Điểm Hóa (float)

Hiển thị menu thực hiện các chức năng sau (mỗi chức năng thực hiện bằng hàm).

1. Hiển thị toàn bộ danh sách
2. Tìm một sinh viên theo mã sinh viên (nhập vào)
3. Xóa một sinh viên khỏi danh sách từ mã sinh viên (nhập vào)
4. Nhập một lớp. Hiển thị danh sách sinh viên thuộc về lớp đó
5. Tính tổng số sinh viên có điểm toán >=5
6. Hiển thị toàn bộ danh sách sinh viên chứa tên nhập vào

Bài 5.

Để quản lý các quyển sách trong thư viện N. Người ta sử dụng một danh sách liên kết để lưu trữ thông tin các quyển sách. Mỗi nút của danh sách là một bản ghi gồm 5 trường:

+ **ID** lưu số thứ tự quyển sách
+ **MaSach** lưu mã của quyển sách
+ **TenSach** lưu tên quyển sách
+ **NamXB** lưu số năm xuất bản quyển sách
+ **SoLuong** lưu trữ số lượng quyển sách
+ Tạo danh sách nhân viên theo kiểu LIFO
+ Viết hàm Insert để chèn một quyển sách vào cuối danh sách
+ Viết hàmSearch để tìm thông tin một quyển sách thông qua Tên quyển sách
+ Viết hàm Sort để sắp xếp danh sách các quyển sách theo thứ tự tăng dần của Số lượng

Bài 6

Sử dụng danh sách liên kết đơn lưu trữ kiểu ngăn xếp - Stack để quản lý khách hàng cho một nhà ga. Mỗi thành phần thông tin lưu trữ cho khách hàng gồm: S*ố CMND khác hàng* (10 ký tự), *Tên* k*hách hàng, Ga đến, Giá tiền*.

Hệ thống menu gồm các mục:

1.Tạo danh sách nhân viên theo kiểu LIFO

2.Thêm một khách hàng mới vào hàng đợi mua vé.

3.Bán một vé cho khách hàng. Chỉ bán cho người đăng ký trước.

4.Hủy một khách hàng ra khỏi danh sách.(Khách hàng không mua vé nữa).

BÀI THỰC HÀNH 6

NGĂN XẾP VÀ HÀNG ĐỢI

1. **Mục tiêu:** Giúp sinh viên hiểu và áp dụng được cấu trúc danh sách liên kết
2. **Kỹ năng:** Cài đặt được các thao tác trên danh sách liên kết
3. **Yêu cầu**: Thực hiện cài đặt một số thao tác và giải quyết các bài toán thực tế
4. **Thời lượng thực hành:** 4 tiết
5. **Tóm tắt lý thuyết:**

6.1. Ngăn xếp- Stack

- **Khái niệm:** Một ngăn xếp là một cấu trúc dữ liệu trừu tượng (Abstract Data Type – viết tắt là ADT), hầu như được sử dụng trong hầu hết mọi ngôn ngữ lập trình. Đặc điểm này làm cho ngăn xếp trở thành cấu trúc dữ liệu dạng LIFO. LIFO là viết tắt của Last-In-First-Out. Ở đây, phần tử được đặt vào (được chèn, được thêm vào) cuối cùng sẽ được truy cập đầu tiên.
- Trong thuật ngữ ngăn xếp, hoạt động chèn được gọi là hoạt động PUSH và hoạt động xóa được gọi là hoạt động POP.
- **Các hoạt động chính trên Stack**
    1. **Hoạt động push(): lưu giữ một phần tử trên ngăn xếp.**
    2. **Hoạt động pop(): xóa một phần tử từ ngăn xếp.**
    3. **Hoạt động peek(): lấy phần** tử dữ liệu ở trên cùng của ngăn xếp, mà không xóa phần tử này.
    4. **Hoạt động isFull(): kiểm tra xem ngăn xếp đã đầy hay chưa.**
    5. **Hoạt động isEmpty(): kiểm tra xem ngăn xếp là trống hay không**

6.2. Hàng đợi – Queue

- **Khái niệm:** Hàng đợi (Queue) là một cấu trúc dữ liệu trừu tượng. Khác với ngăn xếp, hàng đợi là mở ở cả hai đầu. Một đầu luôn luôn được sử

dụng để chèn dữ liệu vào (hay còn gọi là sắp vào hàng) và đầu kia được sử dụng để xóa dữ liệu (rời hàng). Cấu trúc dữ liệu hàng đợi tuân theo phương pháp First-In-First-Out, tức là dữ liệu được nhập vào đầu tiên sẽ được truy cập đầu tiên.

- **Các thao tác chính trên Queue**
  1. **Hoạt động enqueue()**: thêm một phần tử vào trong hàng đợi.
  2. **Hoạt động dequeue()**: xóa một phần tử từ hàng đợi.
  3. **Phương thức peek()**: lấy phần tử ở đầu hàng đợi, mà không xóa phần tử này
  4. **Phương thức isFull()**: kiểm tra xem hàng đợi là đầy hay không.
  5. **Phương thức isEmpty()**: kiểm tra xem hàng đợi là trống hay hay không.

6. Bài thực hành

Bài 1 Cài đặt các thao tác trên ngăn xếp

Bài 2 Cài đặt các thao tác trên hàng đợi

Bài 3 Cài đặt chương trình chuyển biểu thức trung tố về hậu tố

Bài 4 Cài đặt chương trình chuyển biểu thức trung tố về hậu tố

Bài 5 Cài đặt chương trình chuyển biểu thức tiền tố về hậu tố

Bài 6 Cài đặt định giá trị biểu thức hậu tố

BÀI THỰC HÀNH 7

CÂY

1. **Mục tiêu:** Giúp sinh viên hiểu và áp dụng được cấu trúc Cây, Cây nhị phân tìm kiếm.
2. **Kỹ năng:** Cài đặt được các thao tác trên Cây nhị phân tìm kiếm
3. **Yêu cầu**: Thực hiện cài đặt một số thao tác và giải quyết các bài toán thực tế
4. **Thời lượng thực hành:** 4 tiết
5. **Tóm tắt lý thuyết:**

- **Khái niệm Cây:**

  - *Định nghĩa 1*: cây là một tập hợp T các phần tử (gọi là nút của cây) trong đó có 1 nút đặc biệt được gọi là gốc, các nút còn lại được chia thành những tập rời nhau T1, T2 , ... , Tn theo quan hệ phân cấp trong đó Ti cũng là một cây. Mỗi nút ở cấp i sẽ quản lý một số nút ở cấp i+1. Quan hệ này người ta còn gọi là quan hệ cha-con.

  - *Định nghĩa 2*: cấu trúc cây với kiểu cơ sở T là một nút cấu trúc rỗng được gọi là cây rỗng (NULL). Một nút mà thông tin chính của nó có kiểu T, nó liên kết với một số hữu hạn các cấu trúc cây khác cũng có kiểu cơ sở T. Các cấu trúc này được gọi là những cây con của cây đang xét

- **Cây nhị phân- Cây nhị phân tìm kiếm**

- Cây nhị phân: Cây nhị phân là cây mà mỗi nút có tối đa 2 cây con

- Cách duyệt Cây nhị phân:

1. Duyệt theo thứ tự trước
2. Duyệt theo thứ tự giữa
3. Duyệt theo thứ tự sau

- **Cây nhị phân tìm kiếm:** Cây nhị phân tìm kiếm là cây nhị phân rỗng hoặc thoả mãn đồng thời các điều kiện sau:

- ✓ Khoá của các đỉnh thuộc cây con trái nhỏ hơn khoá node gốc.
- ✓ Khoá của node gốc nhỏ hơn khoá của các đỉnh thuộc cây con phải của của gốc.

- **Các thao tác trên cây nhị phân tìm kiếm:**
    1. Khởi tạo cây
    2. Chèn node
    3. Tạo cây nhị phân tìm kiếm
    4. Tìm node
    5. Duyệt cây
    6. Xóa node

6.Bài thực hành

Bài 1

Viết chương trình thực hiện các thao tác sau đây:

1. Khởi tạo một cây nhị phân tìm kiếm với giá trị các node là các số nguyên được nhập từ bàn phím
2. Tìm kiếm một phần tử có giá trị bất kỳ nhập từ bàn phí
3. Duyệt các phần tử trên cây BST trên theo thứ tự: đầu, giữa, cuối.
4. Xóa một phần tử có giá trị bất kỳ trong Cây

Bài 2

Viết chương trình xây dựng và quản lý cây nhị phân tìm kiếm (Binary Search Tree). Hiển thị menu thực hiện các chức năng sau (mỗi chức năng thực hiện bằng hàm). Thành phần dữ liệu trong mỗi Node là giá trị kiểu Integer.

1. Thêm một node vào cây (giá trị nhập vào). Nếu node này ñã có giá trị thì thông báo không thêm vào node ñã có.
2. Tìm giá trị trung bình của danh sách
3. Xuất danh sách theo thứ tự
    - Xuất danh sách theo thứ tự preorder
    - Xuất danh sách theo thứ tự inorder
    - Xuất danh sách theo thứ tự postorder

Bài 3

Viết chương trình thực hiện các thao tác sau:

1. Nhập dữ liệu cây từ file
2. Tính số lượng node của tree
3. Tính chiều cao của cây
4. Tìm giá trị nhỏ nhất
5. Tìm giá trị lớn nhất
6. Tìm một node theo giá trị nhập vào
7. Hiển thị giá trị tăng dần toàn bộ cây
8. Thống kê số lượng node: là số chẵn, là số lẻm là số nguyên tố

Bài 4

Viết chương trình xây dựng và quản lý danh sách sinh viên dựa trên cây nhị phân tìm kiếm (Binary Search Tree). Mỗi sinh viên chứa các thông tin: *mã sv (char), tên sv (char), Điểm toán, Điểm lý, Điểm hóa*. Hiển thị menu thực hiện các chức năng sau (mỗi chức năng thực hiện bằng hàm). Lập chỉ mục Index cho cơ sở dữ liệu theo mã sinh viên. Không có 2 sinh viên nào trùng mã với nhau.

1. Thêm một SV mới
2. Xuất danh sách SV tăng dần theo mã SV
3. Tìm 1 SV theo mã. Nếu tìm ra hiển thị menu con chứa các mục
    - Hiển thị thông tin sinh viên: tên, các điểm và điểm trung bình
    - Cập nhật (sửa) thông tin SV (tên, điểm toán, điểm lý)
4. Lưu danh sách sinh viên xuống file
5. Đọc danh sách sinh viên từ file
6. Xuất danh sách sinh viên tăng dần theo tên SV.

Bài 5

Viết chương trình xây dựng và quản lý cây nhị phân tìm kiếm (Binary Search Tree). Hiển thị menu thực hiện các chức năng sau (mỗi chức năng thực hiện bằng hàm). Thành phần dữ liệu trong mỗi Node là giá trị kiểu integer.

1. Thêm một node vào cây (giá trị nhập vào).
2. Lưu cây vào file

3. Đọc cây từ file
4. Xuất danh sách
5. Inorder
6. Preorder
7. Tìm một node trên cây
8. Hủy một node trên cây. (chọn node trái cùng nhánh bên phải)
9. Hủy một node trên cây. (chọn node phải cùng nhánh bên phải)

Bài 6

Viết chương trình xây dựng và quản lý danh sách lớp sử dụng cây nhị phân tìm kiếm (Binary Search Tree). Hiển thị menu thực hiện các chức năng sau (mỗi chức năng thực hiện bằng hàm).

Mỗi sinh viên gồm các thành phần:

+Mã SV: char[10];

+Mã Lớp : int

+Tên SV: char[255];

+DiemToan

+DiemLy

+DiemHoa

1. Đọc danh sách sinh viện từ file.
2. Thêm một sinh viên
3. Tìm một sinh viên theo mã SV
4. Hiển thị danh sách sinh viên
    - Tăng dần theo mã SV
    - Giảm dần theo mã SV

BÀI THỰC HÀNH 7

ĐỒ THỊ

1. **Mục tiêu:** Giúp sinh viên hiểu và áp dụng cấu trúc Đồ thị và các ứng dụng của Đồ thị trong thực tế.
2. **Kỹ năng:** Hiện thực được cấu trúc Đồ thị bằng ngôn ngữ lập trình C/C++.
3. **Yêu cầu**: Viết một số chương trình ứng dụng
4. **Thời lượng thực hành:** 4 tiết
5. **Tóm tắt lý thuyết:**

Bài 1. Nhập vào ma trận kề của một đơn đồ thị (từ bàn phím và đọc từ tập tin).

a. Kiểm tra tính hợp lệ của đồ thị (giá trị trên đường chéo chính đều bằng 0).

b. Kiểm tra xem đồ thị là vô hướng hay có hướng?

c. Nếu ma trận kề được nhập từ bàn phím thì xuất ra thành tập tin matranke.txt

d. Nếu ma trận được đọc từ tập tin thì xuất kết quả ma trận ra màn hình hiển thị.

e. Xuất ra bậc của tất cả các đỉnh của đồ thị (số cạnh nối tới đỉnh).

f. Kiểm tra tính liên thông của đồ thị? Xuất ra tất cả các thành phần liên

Bài 2. Nhập vào ma trận trọng số của một đơn đồ thị (từ bàn phím và đọc từ tập tin).

a. Kiểm tra tính hợp lệ của đồ thị (giá trị trên đường chéo chính đều bằng 0).

b. Kiểm tra xem đồ thị là vô hướng hay có hướng?

c. Nếu ma trận kề được nhập từ bàn phím thì xuất ra thành tập tin trongso.txt

d. Nếu ma trận được đọc từ tập tin thì xuất kết quả ma trận ra màn hình hiển thị.

e. Xuất ra cạnh có trọng số nhỏ nhất và lớn nhất.

Bài 3.

Cài đặt thuật toán duyệt theo chiều sâu và chiều rộng với đồ thị được cho bởi ma trận kề đọc từ tập tin matranke.txt, chương trình cho phép nhập vào đỉnh xuất phát và hiển thị kết quả duyệt.

Bài 4.

Cài đặt chương trình cho phép nhập vào 2 đỉnh x và y của đồ thị, kiểm tra xem có đường đi từ x tới y (và ngược lại) hay không?