BÀI THỰC HÀNH SỐ 7

1. MỤC TIÊU:

- Giúp SV làm quen với các giải thuật trên cây:
  - Tạo cây
  - Duyệt cây
  - Thêm node mới
  - Xóa node

2. LÝ THUYẾT CẦN GHI NHỚ

Cây nhị phân tìm kiếm là cây nhị phân mà giá trị (khóa) của phần tử bên trái của một node có giá trị nhỏ hơn giá trị (khóa) của node, giá trị (khóa) của các phần tử bên phải của một node thì lớn hơn giá trị (khóa) của node đó.Trong một cây nhị phân tìm kiếm sẽ không có 2 phần tử có giá trị bằng nhau

Các thao tác trên cây:

- Tạo cây
- Thêm node mới
- Duyệt cây
- Xóa node

3. BÀI TẬP THỰC HÀNH CƠ BẢN

SV tạo project "Win32 Console Application" trên VS 2015, với tên là "**Lab7**".

Tạo file source .cpp có tên: **hovaten_mssv_lab7.cpp**

*** Cách tạo project và cấu trúc chương trình C++: Xem lại file hướng dẫn thực hành lab 1

Bài 1:

1. Khai báo cấu trúc cây nhị phân tìm kiếm.
   Cây nhị phân tìm kiếm gồm có nhiều node, mỗi node gồm có 3 thành phần sau:

Info: chứa thông tin trong node

Left: liên kết (chứa địa chỉ) đến node bên trái

Right: liên kết (chứa địa chỉ) đến node bên phải

Khai báo cấu trúc của một node như sau:

```
// cấu trúc 1 node
struct Node
{
        int info; // lưu trữ giá trị của phần tử, giá trị khóa của node
        Node *left; // left lưu địa chỉ phần tử bên trái (cây con trái)
        Node *right; //right lưu trữ địa chỉ phần tử bên phải (cây con phải)
};
Node *root;
```

2. Viết thủ tục khởi tạo cây rỗng.
   Cây rỗng là cây mà có node gốc có giá trị NULL

```
void tree_empty() {
    root = NULL; // node root được khai báo toàn cục
}
```

3. Viết thủ tục thêm một phần tử vào cây (dùng đệ quy).
   Để thêm phần tử x vào cây cần kiểm tra:
   - Nếu phần tử thêm vào có giá trị bằng với một node bất kỳ trong cây ➔ không thêm vào (1)
   - Nếu phần tử thêm vào có giá trị < giá trị của phần tử gốc ➔ duyệt cây con bên trái phần tử gốc để tìm vị trí cần thêm vào phù hợp (2)
   - Nếu phần tử thêm vào có giá trị > giá trị phần tử gốc ➔ duyệt cây con bên phải của phần tử gốc để tìm vị trí cần thêm vào phù hợp (3)
   - Lặp lại bước (1), (2), (3) đối với cây con trái và cây con phải của phần tử gốc cho đến khi gặp giá trị NULL thì thêm phần tử vào

- Hàm thêm phần tử vào cây như sau, trong đó x là giá trị cần thêm vào, p ban đầu là con trỏ trỏ đến phần tử gốc

```
void InsertNode(Node*& p, int x) {
    if (p == NULL) { // insert node goc
        p = new Node;
        p->info = x;
        p->left = NULL;
        p->right = NULL;
    }
    else {
        if (p->info == x) return;
        else if (p->info < x) InsertNode(p->right,
        else InsertNode(p->left, x);
    }
}
```

4. Viết thủ tục tìm một phần tử trong cây (dùng đệ quy).
   Để tìm 1 phần tử trong cây nhị phân tìm kiếm:
   - So sánh phần tử cần tìm với phần từ gốc của cây, nếu tìm thấy ➔ trả về (0)
   - Nếu phần tử cần tìm nhỏ hơn phần từ gốc của cây ➔ tìm ở cây con bên trái của cây (1)
   - Nếu phần tử cần tìm lớn hơn phần tử gốc của cây ➔ tìm ở cây con bên phải của cây (2)

- Lặp lại bước (0), (1), (2) đối với cây con bên trái và cây con bên phải cho đến khi tìm thấy hoặc không tìm thấy (NULL)
- Code tìm như sau:

```
Node* search(Node* p, int x) {
    if (p != NULL) {
        if (p->info == x) return p; // tim thay
        else
            if (p->info < x) search(p->left, x);// de quy tim o cay con trai
            else search(p->right, x); // de quy tim o cay con ben phai
    }
    return NULL;
}
```

*** Lưu ý:

- Giá trị cần tìm là x
- Ban đầu p là node trỏ về node gốc

5. Viết thủ tục xóa một nut trong cây (dùng đệ quy).
   Tìm node cần xóa, nếu tìm không thấy ➔ trả về kết quả không xóa
   Nếu tìm thấy:
   - Nếu node cần xóa là node lá ➔ delete node này (chuyển liên kết của node cha đến node này là NULL và xóa node)
   - Nếu node cần xóa là node cha của 1 hoặc 2 node lá (node bậc 1) thì chỉnh liên kết từ node cha của node cần xóa đến node con của node cần xóa, sau đó xóa node cần xóa đi
   - Nếu node cần xóa là node bậc 2 và có 2 con: tìm node cực trái của nhánh con bên phải – gọi là node thế mạng (là node có giá trị gần với node cần xóa nhất). Chép giá trị của node thế mạng vào giá trị của node cần xóa. Sau đó xóa node thế mạng
   Code như sau:

```
int Delete(Node*& T, int x)
{
    if (T == NULL) return 0;
    if (T->info == x) {
        Node* p = T;
        if (T->left == NULL) T = T->right;
        else if (T->right == NULL) T = T->left;
        else searStandfor(p, T->right);
        delete(p);
        return 1;
    }
    if (T->info < x) return Delete(T->right, x);
    else return Delete(T->left, x);

}
```

```
void searStandfor(Node*& p, Node*& q) {
    if (q->left == NULL) {
        p->info = q->info;
        p = q;
        q = q->right;
    }
    else
        searStandfor(p, q->left);
}
```

6. Viết thủ tục duyệt cây theo thứ tự NLR (dùng đệ quy)
Đi qua từng phần tử trong cây và in giá trị này ra màn hình theo thứ tự: in giá trị ở phần tử gốc trước, sau đó in ra phần tử ở bên trái của gốc, rồi in ra phần tử bên phải của gốc

```
void duyetNLR(Node *p)
{
        if (p != NULL)
        {
                cout<<p->info<<" ";
                duyetLNR(p->left);
                duyetLNR(p->right);
        }
}
```

7. Viết thủ tục duyệt cây theo thứ tự LNR (dùng đệ quy)
Đi qua từng phần tử trong cây và in giá trị này ra màn hình theo thứ tự: in giá trị ở phần tử bên trái gốc trước, sau đó in ra phần tử gốc, rồi in ra phần tử bên phải của gốc

```
void duyetLNR(Node *p)
{
        if (p != NULL)
        {
                duyetLNR(p->left);
                cout<<p->info<<" ";
                duyetLNR(p->right);
        }
}
```

8. Viết thủ tục duyệt cây theo thứ tự LRN (dùng đệ quy)
   Đi qua từng phần tử trong cây và in giá trị này ra màn hình theo thứ tự: in giá trị ở phần tử bên trái gốc trước, sau đó in ra phần tử bên phải của gốc, rồi in ra phần tử gốc

```
void duyetLRN(Node *p)
{
        if (p != NULL)
        {
                duyetLNR(p->left);
                duyetLNR(p->right);
                cout<<p->info<<" ";
        }
}
```

4. BÀI TẬP NÂNG CAO

Bài 1:

a. Đếm tổng số node trong cây
b. Đếm những node có giá trị là chẵn/lẻ/nguyên tố/chính phương trên cây nhị phân tìm kiếm
c. Nhập vào một chuỗi các ký tự(ví dụ: abcfwhij), xây dựng cây nhị phân tìm kiếm với mỗi node là một ký tự của chuỗi vừa nhập