BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC NHA TRANG

KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO THỰC TẬP CƠ SỞ

CHƯƠNG TRÌNH MÔ PHỎNG THUẬT TOÁN DUYỆT CÂY NHỊ PHÂN TRÊN MÔI TRƯỜNG ĐỒ HỌA

Sinh viên thực hiện : Phạm Minh Hoàng

Giảng viên hướng dẫn : ThS. Nguyễn Đình Hưng

Mã số sinh viên : 61131788

KHÁNH HÒA-2021

MỤC LỤC

DANH MỤC HÌNH VẼ, ĐỒ THỊ

LỜI CẢM ƠN

Để có thể hoàn thành đợt thực tập lần này, em xin chân thành cảm ơn đến quý thầy cô khoa Công nghệ Thông tin các bạn trong lớp đã tạo điều kiện và giúp đỡ trong suốt quá trình học tập và thực hiện đề tài. Sự quan tâm và động viên của mọi người là nguồn động lực to lớn giúp em từng bước hoàn thành đề tài lần này

Qua đây, em xin chân thành cảm ơn thầy Nguyễn Đình Hưng, người đã trực tiếp quan tâm và hướng dẫn chúng em hoàn thành tốt đợt thực tập trong thời gian qua.

Do kiến thức còn hạn chế và thời gian thực hiện còn ngắn nên bài báo cáo của em còn nhiều thiếu sót, kính mong sự góp ý của quý thầy cô.

Em xin chân thành cảm ơn!

LỜI NÓI ĐẦU

Cùng với sự phát triển của khoa học kĩ thuật , công nghệ thông tin nói chung và bộ môn cấu trúc dữ liệu và giải thuật nói riêng ngày càng được ứng dụng rộng rãi trong nhiều lĩnh vực. Với một cơ sở dữ liệu khổng lồ, việc đưa ra một phương pháp nhằm giải quyết vấn đề tìm kiếm dữ liệu có hiệu quả và nhanh chóng nhất luôn được sự quan tâm của các nhà phát triển phần mềm. Thông thường dữ liệu được biểu diễn dưới dạng danh sách liên kết. Việc truy xuất dữ liệu chưa đạt hiệu quả cao. Sử dụng cấu trúc dữ liệu cây là một giải pháp làm tăng hiệu suất trong các thao tác xử lý. Vấn đề đặt ra : với việc sử dụng cấu trúc dạng cây, chúng ta cần dùng giải thuật nào với từng dạng dữ liệu để đạt hiệu quả cao nhất. Để giải quyết vấn đề trên ta cùng tìm hiểu một số phương pháp duyệt cây nhị phân.

Cây nhị phân là một cấu trúc dữ liệu quan trọng mà trong môn Cấu trúc dữ liệu và giải thuật, nó được sử dụng rất rộng rãi trong lập trình vì các ứng dụng của nó.

Các cấu trúc dữ liệu khác như mảng, danh sách liên kết, ngăn xếp và hàng đợi là cấu trúc dữ liệu tuyến tính và lưu trữ dữ liệu theo một cách tuần tự. Để thực hiện bất kỳ hoạt động nào trong cấu trúc dữ liệu tuyến tính, độ phức tạp về thời gian sẽ tăng lên khi kích thước dữ liệu tăng lên. Các cấu trúc dữ liệu dạng cây khác nhau cho phép truy cập dữ liệu nhanh hơn và dễ dàng hơn vì nó là cấu trúc dữ liệu phi tuyến tính.

Toàn bộ mã nguồn của chương trình được tải lên theo địa chỉ:

https://github.com/HoangMinh-dell7566/TTCS

CHƯƠNG 1: TỔNG QUAN VẤN ĐỀ

1.1 TỔNG QUAN

Cây nhị phân được sử dụng vào nhiều mục đích khác nhau. Tuy nhiên việc sử dụng cây nhị phân để lưu giữ và tìm kiếm thông tin vẫn là một trong những ứng dụng quan trọng nhất. Và để phục vụ cho việc nghiên cứu đề tài lần này, em đã sử dụng phần mềm Dev C để có thể mô phỏng được quy trình duyệt một cây nhị phân tìm kiếm. Điểm đặc biệt của Dev C chính là việc phần mềm này có hỗ trợ thư viện đồ họa Graphic.h giúp việc mô phỏng thuật toán trở nên rõ ràng, cụ thể hơn trên màn hình console. Khi cài đặt thư viện này, mình cần lưu ý copy 2 file "graphics.h" và "winbgim.h" vào đường dẫn C:\Program Files (x86)\Dev-Cpp\MinGW64\x86_64-w64-mingw32\include 2 file này chưa được tích hợp sẵn trong thư viện của phần mềm Dev C nên ta phải tự thêm vào để sử dụng khi lập trình đồ họa.

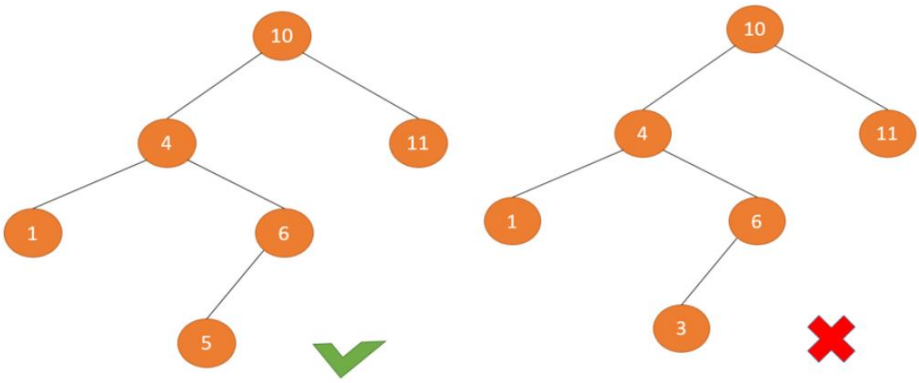
1.2 LÝ DO CHỌN ĐỀ TÀI

Với mong muốn ôn lại kiến thức kỹ thuật đồ họa, cấu trúc dữ liệu và giải thuật để phục vụ cho công việc sau này em đã quyết định chọn đề tài này. Đề tài vận dụng phương pháp lập trình xây dựng cấu trúc cây, đệ quy, hàng đợi được áp dụng để duyệt cây nhị phân tìm kiếm đó chính là tiền đề để em hoàn thành được đề tài lần này. Như chúng ta đã biết cây nhị phân (Binary Search Tree) là một cấu trúc rất phổ biến và được ứng dụng vào nhiều bài toán lập trình từ đơn giản đến phức tạp, việc được các thầy cô hướng dẫn để hoàn thành được đề tài lần này là cơ hội vô cùng quý báu để em có thể củng cố, trau dồi thêm kiến thức giúp nâng cao năng lực sau này.

1.3 GIỚI THIỆU CHUNG

Cây nhị phân là một tập hợp hữu hạn các node, trong đó có một node đặc biệt gọi là gốc (Root). Đúng như tên gọi của nó, cây nhị phân có bậc là 2 và mỗi nút trong cây nhị phân đều có bậc không quá 2 nút. Thuật toán Duyệt cây là một tiến trình để truy cập tất cả các nút của một cây và cũng có thể in các giá trị của các nút này. Cây nhị phân tìm kiếm, tiếng anh là Binary Search Tree (BST), là cây nhị phân mà trong đó, các phần tử của cây con bên trái đều nhỏ hơn phần tử hiện hành và các phần tử của cây con bên phải

đều lớn hơn phần tử hiện hành. Do đó, cây nhị phân tìm kiếm không được có phần tử cùng giá trị.



Hình 1. Ví dụ cây nhị phân tìm kiếm.

Trong mỗi nút của cây nhị phân sẽ bao gồm 3 thành phần như sau:

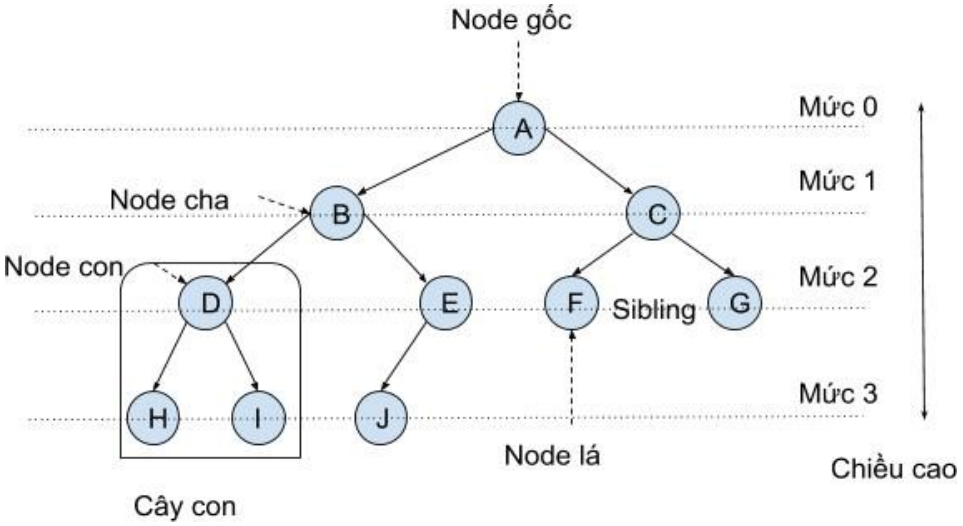
- Thành phần dữ liệu: có thể là bất kỳ kiểu dữ liệu nào. Thường là kiểu số.
- Thành phần liên kết trái: lưu trữ địa chỉ của nút gốc của cây con bên trái.
- Thành phần liên kết phải: lưu trữ địa chỉ của nút gốc của cây con bên phải.

Nhờ vào cấu trúc đặc biệt của cây nhị phân tìm kiếm, nên khi được sử dụng để tìm kiếm phần tử sẽ nhanh hơn (tương tự với tìm kiếm nhị phân) và chính xác hơn. Có nhiều cách duyệt cây nhị phân khác nhau, từ đó sẽ thu được nhiều mảng có thứ tự khác nhau. Chúng ta sẽ lần lượt tìm hiểu qua chúng.

Duyệt cây có nghĩa là đi qua từng nút trong cây. Ví dụ, ta có thể muốn xem tất cả các giá trị trong cây hoặc tìm giá trị lớn nhất. Để thực hiện tất cả các thao tác này, ta sẽ cần phải truy cập vào từng nút của cây. Ở hình 1, khi thực hiện thuật toán duyệt trung thứ tự của cây bên trái, ta sẽ được kết quả như sau: 1, 4, 5, 6, 10, 11.

1.4 MÔ TẢ BÀI TOÁN

Cấu trúc dữ liệu tuyến tính như mảng, ngăn xếp, hàng đợi và danh sách liên kết chỉ có một cách để đọc dữ liệu. Nhưng cấu trúc dữ liệu phân cấp như cây có thể được duyệt theo nhiều cách khác nhau. Chính vì thế, sau khi đã sinh 1 cây nhị phân tìm kiếm chúng ta sẽ bắt đầu duyệt qua cây theo 2 phương thức: duyệt theo chiều rộng và duyệt theo chiều sâu. Khi duyệt theo chiều rộng ta sẽ lần lượt đi qua từng bậc của cây, ứng với mỗi bậc ta sẽ in ra tuần tự các nút từ trái qua phải. Sau khi đã hoàn thành 1 bậc, thuật toán sẽ tiếp tục với bậc tiếp theo cho đến khi đi hết bậc của cây.



Hình 2. Cây nhị phân tìm kiếm

Ví dụ với cây nhị phân như ở hình 2, khi áp dụng thuật toán duyệt theo chiều sâu, các bước của thuật toán sẽ là:

- Đầu tiên duyệt nút gốc (mức 0) và in ra dữ liệu là A.
- Sau đó là duyệt tới mức 1, từ trái qua phải, dữ liệu nhận được là B, C.
- Tiếp tục duyệt đến mức 2 và in ra dữ liệu là D, E, F, G.
- Cuối cùng là kết thúc thuật toán ở mức 3 và dữ liệu nhận được là H, I, J.

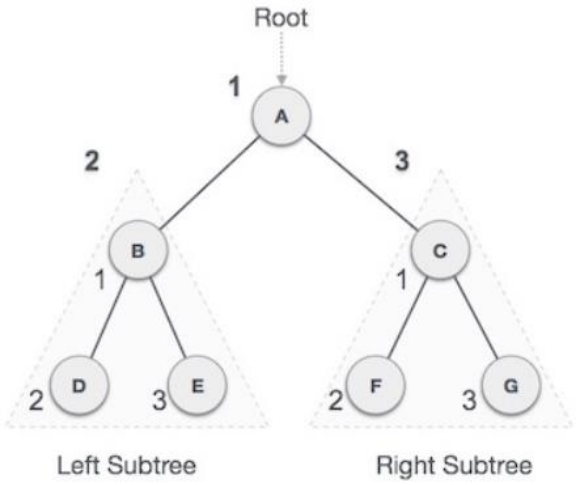
Kết quả khi kết thúc thuật toán là: A, B, C, D, E, F, G, H, I, J.

Đối với thuật toán duyệt cây theo chiều sâu, có 3 cách làm đó là: duyệt tiền thứ tự (NLR), duyệt trung thứ tự (LNR), duyệt hậu thứ tự (LRN). 3 cách này về cơ bản là hoàn toàn

giống nhau, điểm khác biệt duy nhất chính là thứ tự duyệt nút hiện hành so với 2 cây con của nút đó.

1.4.1 DUYỆT TIỀN THỨ TỰ (NLR)

Trong cách thức duyệt tiền thứ tự trong cây nhị phân, nút gốc được duyệt đầu tiên, sau đó sẽ duyệt cây con bên trái và cuối cùng sẽ duyệt cây con bên phải.



Hình 3. Duyệt tiền thứ tự NLR

Ở hình ví dụ minh họa trên, **A** là nút gốc. Chúng ta bắt đầu từ **A**, và theo cách thức duyệt tiền thứ tự, đầu tiên chúng ta truy cập chính nút gốc **A** này và sau đó di chuyển tới nút con bên trái **B** của nó. **B** cũng được duyệt theo cách thức duyệt tiền thứ tự. Và tiến trình tiếp tục cho tới khi tất cả các nút đều đã được truy cập. Kết quả của cách thức duyệt tiền thứ tự cây này sẽ là:

$$A \rightarrow B \rightarrow D \rightarrow E \rightarrow C \rightarrow F \rightarrow G$$

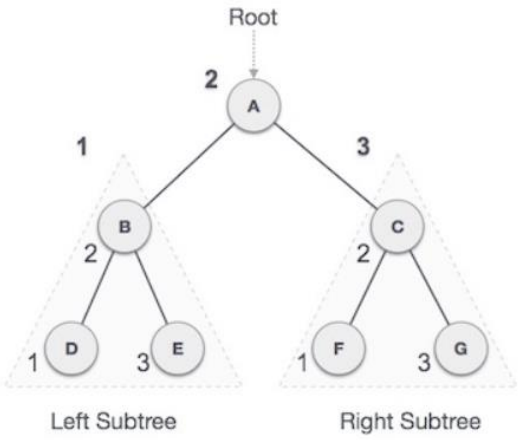
Giải thuật cho cách duyệt tiền thứ tự:

- Bước 1: Truy cập nút gốc.
- Bước 2: Duyệt các cây con bên trái một cách đệ quy.
- Bước 3: Duyệt các cây con bên phải một cách đệ quy.

1.4.2 DUYỆT TRUNG THỨ TỰ (LNR)

Trong cách duyệt này, cây con bên trái được truy cập đầu tiên, sau đó là nút gốc và sau đó là cây con bên phải. Bạn nên luôn luôn ghi nhớ rằng mỗi nút đều có thể biểu diễn một cây con.

Nếu một cây nhị phân được duyệt trung thứ tự, kết quả tạo ra sẽ là các giá trị khóa được sắp xếp theo thứ tự tăng dần.



Hình 4. Duyệt trung thứ tự LNR

Ở hình ví dụ minh họa trên, **A** là nút gốc. Với phương thức duyệt trung thứ tự, chúng ta bắt đầu từ nút gốc **A**, di chuyển tới cây con bên trái B của nút gốc. Tại đây, **B** cũng được duyệt theo cách thức duyệt trung thứ tự. Và tiến trình tiếp tục cho đến khi tất cả các nút đã được truy cập. Kết quả của cách thức duyệt trung thứ tự cho cây trên sẽ là:

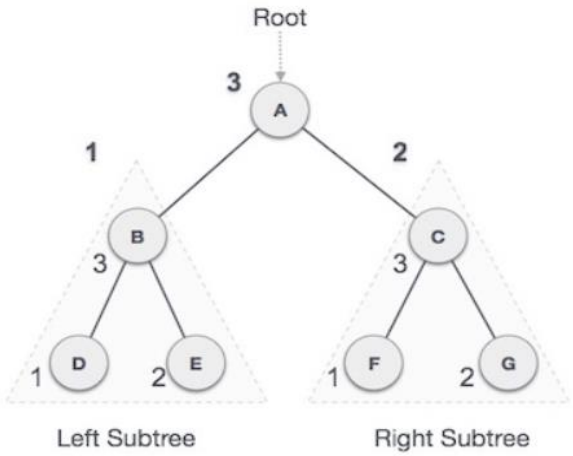
D → B → E → A → F → C → G

Giải thuật cho cách duyệt trung thứ tự:

- Bước 1: Duyệt các cây con bên trái một cách đệ quy.
- Bước 2: Truy cập nút gốc.
- Bước 3: Duyệt các cây con bên phải một cách đệ quy.

1.4.3 DUYỆT HẬU THỨ TỰ (LRN)

Trong cách thức duyệt hậu thứ tự trong cây nhị phân, nút gốc của cây sẽ được truy cập cuối cùng, do đó bạn cần chú ý. Đầu tiên, chúng ta duyệt cây con bên trái, sau đó sẽ duyệt cây con bên phải và cuối cùng là duyệt nút gốc.



Hình 5. Duyệt hậu thứ tự LRN

Ở hình ví dụ minh họa trên, **A** là nút gốc. Chúng ta bắt đầu từ **A**, và theo cách duyệt hậu thứ tự, đầu tiên chúng ta truy cập cây con bên trái **B**. **B** cũng được duyệt theo cách thứ duyệt hậu thứ tự. Và tiến trình sẽ tiếp tục tới khi tất cả các nút đã được truy cập. Kết quả của cách thức duyệt hậu thứ tự của cây con trên sẽ là:

$$D \rightarrow E \rightarrow B \rightarrow F \rightarrow G \rightarrow C \rightarrow A$$

Giải thuật cho cách duyệt hậu thứ tự

- Bước 1: Duyệt các cây con bên phải một cách đệ quy.
- Bước 2: Duyệt các cây con bên trái một cách đệ quy.
- Bước 3: Truy cập nút gốc.

CHƯƠNG 2. PHƯƠNG PHÁP NGHIÊN CỨU

2.1 CÀI ĐẶT THUẬT TOÁN:

2.1.1 KHAI BÁO THƯ VIỆN

Để thuận tiện cho việc viết chương trình, việc đầu tiên là phải khai báo 1 số thư viện cần thiết để quá trình cài đặt gặp ít khó khăn hơn.

```
#include<iostream>
#include<conio.h>
#include"graph.h"
#include<graphics.h>
//-- Ngon ngu C++
```

Hình 6. Khai báo thư viện

Ở đây em sử dụng thư viện iostream.h để dùng ngôn ngữ C++, conio.h hỗ trợ các hàm giúp thực hiện các thao tác input hoặc output từ màn hình console, graphic.h là thư viện không thể thiếu để lập trình trong môi trường đồ họa, còn graph.h là 1 thư viện tự định nghĩa khai báo các hàm giúp thu gọn chương trình main.

2.1.2 TẠO VÀ MÔ PHỎNG CÂY

```
struct node
{
    int data; // node mang gia tri kieu int.
    node *pleft,*pright; //--- 2 con tro tro den 2 cay con (trai - phai).
};
typedef node* tree;
//-----------------------------------------------
//-- Khoi tao cây
void init(tree &t)
{
    t=NULL;
}
//-----------------------------------------------
//-- Tao nut moi
node* get_node(int data)
{
    node *p=new node();
    p->data=data;
    p->pleft=NULL;
    p->pright=NULL;
    return p;
}
```

Hình 7. Các bước khởi tạo

Sau khi đã khai báo xong các thư viện cần thiết, việc tiếp theo là khai báo 1 kiểu dữ liệu cấu trúc struct, kiểu dữ liệu này bao gồm 1 biến data kiểu int dùng để lưu trữ giá trị của

nút hiện hành và 2 biến con trỏ, sẽ trỏ tới địa chỉ của 2 cây con trái và phải, sau đó sẽ khởi tạo cây ban đầu là 1 cây rỗng, dữ liệu sẽ được chèn vào cây sau khi người dùng nhập các giá trị từ bàn phím, mỗi giá trị nhập vào sẽ được tạo thành 1 nút thông qua hàm get_node(data).

```
//-- Chen 1 nut vao cay nhi phan
void insert(tree &t,node *p)
{
    //-- neu cay chua co nut thi chen vao goc
    if(t==NULL)
    {
        t=p;
    }
    //-- Neu co nut thì chen vao cay con
    else
    {
        if(p->data < t->data)
        {
            return insert(t->pleft,p);
        }
        else
        if(p->data > t->data)
        {
            return insert(t->pright,p);
        }
    }
}
```

Hình 8. Chèn nút vào cây

Tiếp đến là hàm chèn 1 nút bất kỳ vào cây, hàm này nhận đối số truyền vào là biến t lưu địa chỉ của nút gốc và 1 nút lưu địa chỉ của giá trị cần chèn vào cây. Nếu cây ban đầu chưa có cây thì nút đó sẽ được chèn thẳng vào gốc, ngược lại nếu đã tồn tại cây thì sẽ tìm vị trí của nút đó trong cây, sao cho thỏa điều kiện cho phép, nếu nhỏ hơn nút hiện hành thì sẽ tiếp tục xét sang cây con trái, nếu lớn hơn nút hiện hành thì sẽ xét sang cây con phải, lặp lại đệ quy như vậy cho đến khi tìm được vị trí.

```
void khoitao(tree &t)
{
    //-- Goi ham khoi tao
    init(t);
    int data;
    cout << "Nhap vao day so, nhap 0 de dung: \n";
    while(cin>>data)
    {
        if(data!=0)
        {
            insert(t,get_node(data));
        }
        else
        {
            break;
        }
    }
}
```

Hình 9. Khởi tạo cây

Sau khi đã có các hàm để phục vụ cho bước sinh 1 cây nhị phân, chúng ta bắt đầu tiến hành khởi tạo cây dựa trên các hàm đã xây dựng từ trước. Khởi tạo 1 cây rỗng, sau đó nhập vào từng giá trị của cây, khi giá trị bằng 0 sẽ kết thúc nhập và xuất ra cây nhị phân được tạo. Mỗi giá trị khi nhập vào sẽ được tạo thành 1 nút, nút đó được chèn vào cây như mô tả ở trên. Cuối cùng là mô phỏng bằng đồ họa thuật toán sinh cây dựa trên các hàm có sẵn của thư viện graphic.h, ở công đoạn này em đã tự xây dựng 1 thư viện là graph.h để thực thi đoạn chương trình liên quan đến đồ họa.

Để vẽ được 1 nút bằng đồ họa, chúng ta phải tính toán tọa độ của nút đó trên màn hình windows, ở đây em sử dụng hàm setcolor() để thiết lập màu cho nét vẽ, settextstyle() để thiết lập kiểu chữ, outtextxy() dùng để in chuỗi s theo vị trí (x,y) và hàm circle() để vẽ 1 đường tròn mô phỏng cho nút. Dưới đây là mẫu tạo nút gốc mang giá trị là .

```
setcolor(8);
//---- node goc (7)
settextstyle(0,0,4);
outtextxy(435,53,"7");
circle(450,70,25);
```

Hình 10. Vẽ 1 nút

Như vậy sau khi đã vẽ được nút gốc, chúng ta tiếp tục vẽ các nút còn lại của cây, tuy nhiên ở các nút sau ngoài việc phải vẽ các nút chúng ta còn phải vẽ thêm các đường nối giữa 2 nút có quan hệ “cha – con” với nhau để tạo thành cây hoàn chỉnh. Chính vì thế, trước khi vẽ nút mới chúng ta phải tính toán tọa độ của nút thứ tiếp theo, đồng thời tính toán tọa độ của đường nối sao cho điểm bắt đầu và kết thúc phải nằm đúng vị trí trên đường tròn và dễ nhìn. Từ đó, khi gọi hàm Sinhcay() chúng ta sẽ vẽ được 1 cây nhị phân tìm kiếm hoàn chỉnh, để phục vụ cho bước duyệt cây tiếp theo.

```
//-- Tu (7) ve ham line() toi 8

delay(dl);
line(467,87,583,153);
delay(dl);
settextstyle(0,0,4);
outtextxy(585,153,"8");
circle(600,170,25);

//--- Tu (7) ve ham line() toi 3
delay(dl);
line(433,87,317,153);
delay(dl);
settextstyle(0,0,4);
outtextxy(285,153,"3");
circle(300,170,25);
```

Hình 11. Vẽ đường nối và 2 nút con

2.1.3 DUYỆT VÀ MÔ PHỎNG DUYỆT CÂY

- Duyệt cây theo chiều rộng

Đối với thuật toán duyệt cây theo chiều rộng, em sử dụng hàng đợi để lưu vết sau đó xử lý từng nút mỗi khi duyệt qua.

```
//-- 1. Ham duyet cay nhi phan theo chieu rong
void DuyetCayTheoChieuRong(tree &t) //-- Duyet tu tren xuong, sau do tu trai qua.
{
    // Neu nhu cay co so phan tu (>=1)
    if(t!=NULL)
    {
        //-- Tao queue de luu vet (a)
        queue<tree> a;
        //-- Sau do them vao queue goc
        a.push(t);
        //-- Trong khi queue khac rong
        while(!a.empty())
        {
            //-- luu phan tu dau tien trong queue vao p
            node *p=a.front();
            //-- Xuat du lieu
            cout<<p->data<<" ";
            //-- xoa phan tu dau tien da duoc xuat
            a.pop();
            if(p->pleft!=NULL)
            {
                //day cay con trai vào queue
                a.push(p->pleft);
            }
            if(p->pright!=NULL)
            {
                //day cay con phai vao queue
                a.push(p->pright);
            }
        }
    }
}
```

Hình 12. Duyệt cây theo chiều rộng

Ở đây, đầu tiên em sẽ tạo ra 1 hàng đợi kiểu tree lưu vào biến a. Sau đó đẩy nút gốc vào a. Lúc này a đã có phần tử cho nên sẽ thực thi đoạn code trong while, vì cách hoạt động của hàng đợi là FIFO (First In First Out) nên chúng ta sẽ lần lượt thêm từng nút của mỗi mức vào hàng đợi, như thế khi xuất ra cây sẽ được duyệt từ trên xuống dưới, từ trái qua phải đến khi duyệt qua hết các nút thì dừng.

Sau khi duyệt xong chúng ta bắt đầu mô phỏng đồ họa quy trình duyệt cây, bằng cách khi duyệt tới nút nào thì nút đó sẽ sáng lên đồng thời in kết quả xuống bên dưới, để sáng nút em vận dụng hàm setcolor() thay đổi màu của nét vẽ và vẽ lại nút được duyệt, đồng thời cho 1 khoảng delay tạo chuyển động cho quá trình duyệt. Thao tác này được lặp đi lặp lại cho từng nút từ đó mô phỏng quá trình duyệt cây theo chiều rộng.

```
//-- sang nut 7 ------------------------
delay(dl);
setcolor(9);
settextstyle(0,0,4);
outtextxy(435,53,"7");
circle(450,70,25);

//--- in nut  7 ----------------------
setcolor(9);
delay(dl);
settextstyle(1,0,3);
outtextxy(225,378,"7");
circle(235,390,20);
```

Hình 13. Mô phỏng duyệt 1 nút

- Duyệt cây theo chiều sâu

Đối với thuật toán duyệt cây theo chiều sâu, sử dụng thuật toán đệ quy gọi lại hàm khi duyệt theo từng cách, duyệt tiền thứ tự, duyệt trung thứ tự và duyệt hậu thứ tự. Cả 3 thuật toán đều bắt đầu từ nút gốc và duyệt sâu xuống nút lá ngoài cùng, điểm khác biệt duy nhất là việc xuất dữ liệu của nút hiện hành đang được duyệt, đối với tiền thứ tự nút hiện hành được xuất ra đầu tiên, trung thứ tự nút hiện hành được xuất ra sau cây con trái và trước cây còn phải, cuối cùng là hậu thứ tự sẽ xuất ra cuối cùng sau khi đã duyệt xong 2 cây con.

```
void DuyetNLR(tree &t)
{
    if(t == NULL)
        return;
    cout << t->data << " ";
    DuyetLNR(t->pleft);
    DuyetLNR(t->pright);
}
//-------------------------------------------------
```

Hình 13. Thuật toán tiền thứ tự

```
void DuyetLNR(tree &t)
{
    if(t == NULL)
        return;
    DuyetLNR(t->pleft);
    cout << t->data << " ";
    DuyetLNR(t->pright);
}
//-------------------------------------------------
```

Hình 14. Thuật toán trung thứ tự

```
void DuyetLRN(tree &t)
{
    if(t == NULL)
        return;
    DuyetLNR(t->pleft);
    DuyetLNR(t->pright);
    cout << t->data << " ";
}
//-------------------------------------------------
```

Hình 15. Thuật toán hậu thứ tự

Sau khi đã duyệt xong cây theo chiều sâu, bắt đầu đến bước tiếp theo mô phỏng từng các duyệt theo đồ họa, ở đây cách mô phỏng cũng tương tự như mô phỏng duyệt theo chiều rộng, duyệt nút nào thì nút đó sáng đèn và in ra kết quả xuống bên dưới, và để phân biệt quá trình duyệt của từng giải thuật, em đã xây dựng 1 hàm tắt hết các nút trước khi duyệt, vì như đoạn mã ở trên mô phỏng duyệt 1 nút thì chỉ có bật mà chưa có tắt. Để tắt 1 nút em sẽ cho nút đó về lại màu ban đầu khi vẽ cây bằng hàm setcolor(), và hàm này áp dụng cho tất cả các nút và không có delay.

```
setcolor(8);
settextstyle(0,0,4);
outtextxy(185,253,"2");
circle(200,270,25);
```

Hình 16. Mô phỏng tắt đèn

2.1.4 CHƯƠNG TRÌNH CHÍNH

Tạo menu cho người dùng, menu cho phép người dùng chọn 2 phương thức duyệt theo chiều rộng hay theo chiều sâu, người dùng có thể lựa chọn các phương thức màn hình đồ họa sẽ hiển thị song song bên cạnh.

```
void DuyetCay(tree t)
{
    //------------------ Phan Sinh Duyet Cay --------------------------------
    //-- Dua ra OPTIONS -----------
    cout << "\n\n\nLua chon cua ban la gi?: ";
    cin >> tmp;
    //-- 1. Sinh va Duyet Cay Theo Chieu rong (bfs)
    if(tmp == 1)
    {
        cout << "1. Duyet cay theo chieu rong: ";
        DuyetCayTheoChieuRong(t);
        DoHoaMoPhongDuyetCayChieuRong();
    }
    //-- 2. Sinh va Duyet cay theo Chieu sau (dfs)
    if(tmp == 2)
    {
        cout << "2. Co 3 cach duyet theo chieu sau, lan luot la:";
        cout << "\n  2.1. Duyet theo LNR: ";
        DuyetLNR(t);
        cout << "\n  2.2. Duyet theo LRN: ";
        DuyetLRN(t);
        cout << "\n  2.3. Duyet theo NLR: ";
        DuyetNLR(t);
        DoHoaMoPhongDuyetCayChieuSau();
    }
}
```

Hình 18. Hàm gọi duyệt cây

Hàm DuyetCay() được dùng để đóng gói cả 2 phương thức duyệt cây, khi gọi hàm sẽ xuất hiện menu cho người dùng lựa chọn, nếu chọn 1 sẽ thực hiện duyệt cây theo chiều rộng và trả về kết quả trong màn hình console, đồng thời mô phỏng đồ họa song song bên cạnh, nếu chọn 2 sẽ thực hiện thuật toán và đồ họa mô phỏng của cả 3 phương thức tiền thứ tự, trung thứ tự và hậu thứ tự.

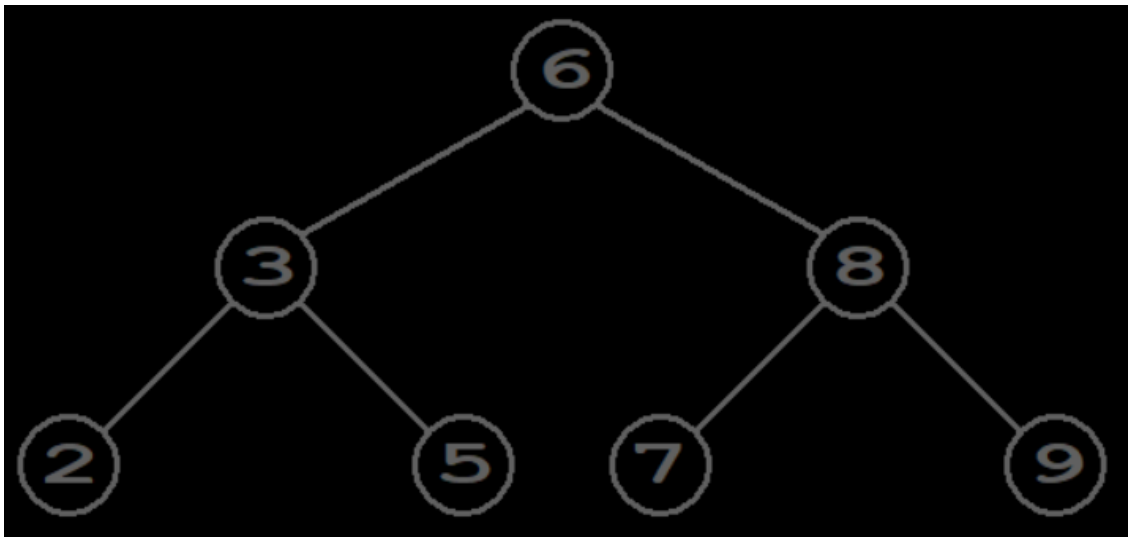
```
int main()
{
    initwindow(900,700);
    //------------------------------
    tree t;
    khoitao(t);
    DoHoaMoPhongVeCay();
    //--- Phan menu tinh ----
    textcolor(2);
    cout << "Cac chuc nang chinh cua chuong trinh:"<<endl;
    textcolor(6);
    cout << "1. Duyet cay theo chieu rong"<<endl;
    cout << "2. Duyet cay theo chieu sau"<<endl;
    //-------------------------------------------------------------
    DuyetCay(t);
    while(continuee == 1)
    {
        cout << "\n\n\n\nchon 1 de tiep tuc, 0 de dung lai: ";
        cin >> continuee;
        if(continuee == 1)
        {
            DuyetCay(t);
        }
        else
        {
            cout << "\nKet thuc chuong trinh>>>";
            cleardevice();
            //xóa màn hình
        }
    }
    getch();
    closegraph();
    return 0;
}
```

Hình 19. Chương trình chính

Chương trình chính sẽ khởi tạo cửa sổ đồ họa sau đó thực hiện khởi tạo 1 cây nhị phân và mô phỏng quá trình sinh cây dựa trên những giá trị được nhập vào, tiếp đến gọi các hàm đã được đóng gói, hiển thị 1 menu cho người dùng chọn lựa, sẽ duyệt theo phương thức nào. Nếu chọn 1 sẽ duyệt theo chiều rộng, chọn 2 để duyệt theo chiều sâu. Cuối cùng sau khi người dùng đã thực hiện hết các chức năng và muốn dừng chương trình gọi hàm cleardevice() để xóa màn hình và kết thúc.

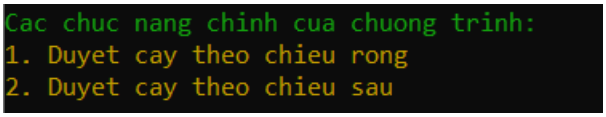
CHƯƠNG 3. KẾT QUẢ THỰC HIỆN

3.1 Mô phỏng vẽ cây



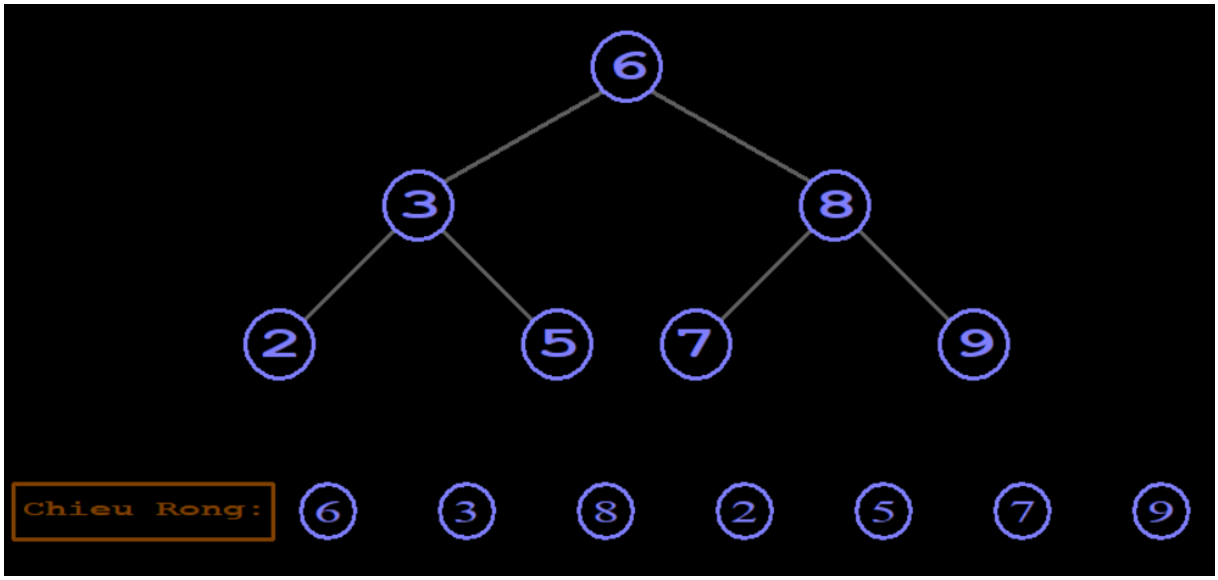
Hình 20. Mô phỏng vẽ cây

Sau khi người dùng kết thúc việc nhập chuỗi dữ liệu để tạo cây thì hàm đồ họa mô phỏng quá trình tạo cây sẽ thực hiện. Tiếp theo đó, menu cho người dùng chọn lựa các phương thức duyệt sẽ hiện ra và người dùng tiến hành chọn để tiếp tục thực hiện chương trình.



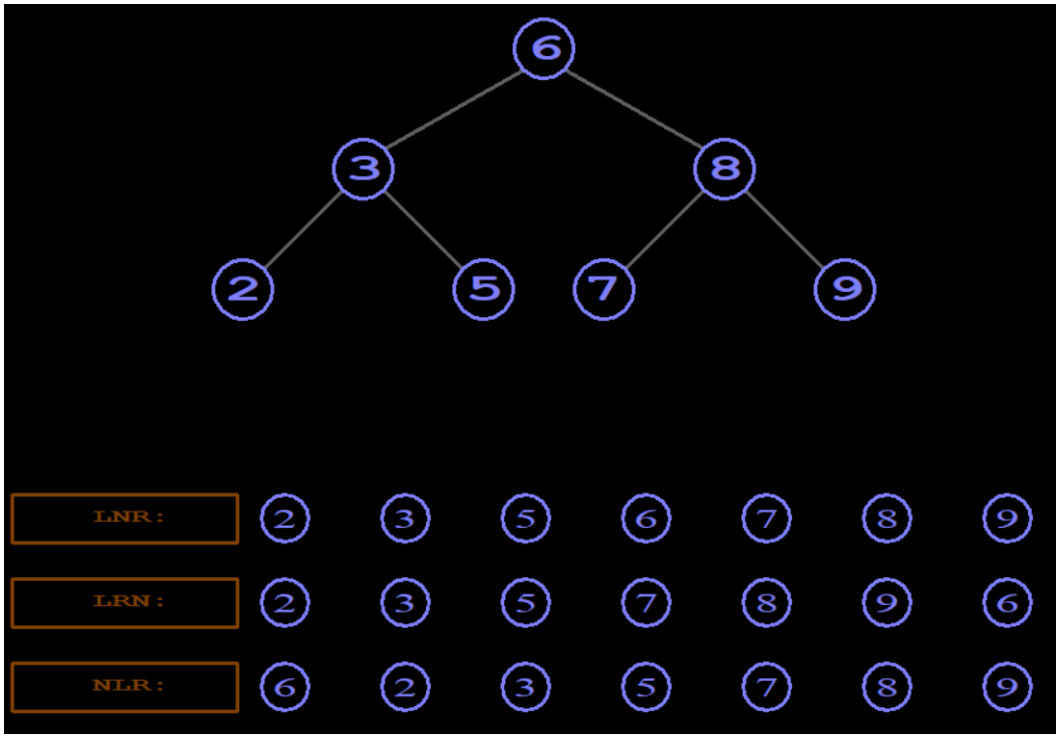
Hình 21. Menu chọn lựa

3.2 Mô phỏng duyệt cây



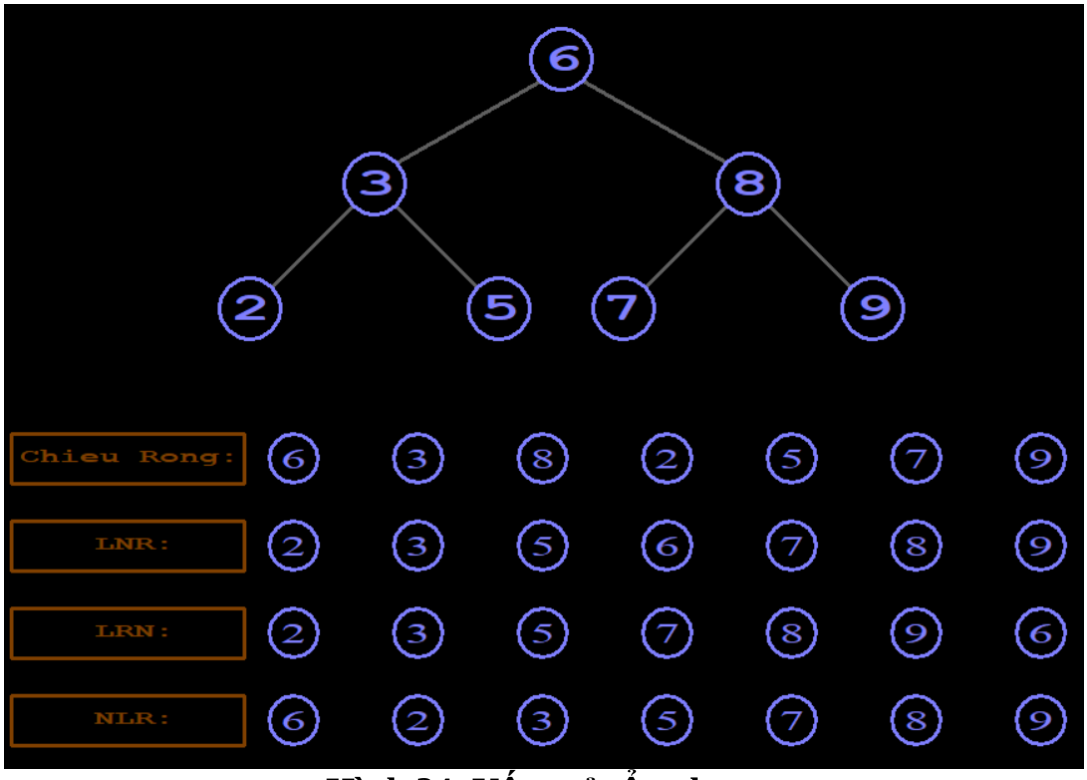
Hình 22. Kết quả duyệt cây theo chiều rộng

Khi người dùng chọn 1, cửa sổ đồ họa sẽ thực hiện mô phỏng duyệt theo chiều rộng và cho ra kết quả như ở hình 22. Ngược lại nếu chọn 2, cửa sổ sẽ thực hiện mô phỏng duyệt theo chiều sâu và cho ra kết quả ở hình 23 .



Hình 23. Kết quả duyệt cây theo chiều sâu

Bên cạnh đó người dùng có thể chọn lần lượt cả 2 phương thức, kết quả vẫn sẽ được xuất ra cửa sổ đồ họa.



Hình 24. Kết quả tổng hợp

CHƯƠNG 4. KẾT LUẬN

Sản phẩm cơ bản đáp ứng các yêu cầu đề ra nhất định đó là mô phỏng việc xây dựng và duyệt một cây nhị phân tìm kiếm đơn giản với số nút tối đa là 7.

Trong lần chọn đề tài này, bản thân em đã tự lường trước cho mình một nhiệm vụ không mấy dễ dàng, bởi đây là một giải thuật tương đối trừu tượng và rất rộng, cộng thêm sử dụng kỹ thuật lập trình mà trước đó đã vốn rất khó khăn.

Chương trình này đúng hơn hết là những cóp nhặt có chọn lọc của những tác giả đi trước về mặt giải thuật và những nhận định đôi khi mang tính chủ quan. tuy nhiên chưa hoàn thiện về mặt hình thức trang trí. Trong tương lai, sẽ phát triển sản phẩm hoàn thiện hơn, có các chức năng khác như chèn 1 nút bất kỳ, cân bằng cây, xoay cây, mô phỏng nhiều đồ họa khác nhau và linh hoạt hơn... Toàn bộ quy trình thiết kế, thực hiện sản phẩm được hoàn thiện trong thời gian 4 tuần của đợt thực tập cơ sở.

TÀI LIỆU THAM KHẢO

1. Giáo trình HƯỚNG DẪN GIẢI TIẾT VÀ LẬP TRÌNH [KỸ THUẬT ĐỒ HOẠ] – thầy Đoàn Vũ Thịnh.
2. Giáo trình CẤU TRÚC DỮ LIỆU – thầy Nguyễn Đức Thuần.
3. Sách CẤU TRÚC DỮ LIỆU, PHÂN TÍCH THUẬT TOÁN VÀ PHÂN TÍCH PHẦN MỀM – [Hồ Thuần, Hồ Cẩm Hà, Trần Thiên Thành].
4. https://text.123docz.net/document/2623543-de-tai-bao-cao-cay-nhi-phan.htm
5. https://khiemle.dev/cay-nhi-phan-trong-cpp/?fbclid=IwAR3PmWskdqUSFh7GC20gzVluKNDErgNLt1LCySvstl3rAJ6 PkQnLUEAvZRY#Cay_nhi_phan
6. https://www.slideshare.net/ANHMATTROI/mot-so-ham-do-hoa-trong-c-c?fbclid=IwAR3S54I_ItH5MEnjTpl4KJZq6a26BkfJ3tf8ZWqJfX_5aok- d3nK5iErJAQ
7. Duyệt cây trong cấu trúc dữ liệu và giải thuật from <https://vietjack.com/cau-truc-du-lieu-va-giai-thuat/duyet-cay.jsp>
8. https://vimentor.com/vi/lesson/duyet-cay-theo-chieu-rong-1 - Duyệt cây theo chiều rộng