Bài thực hành số 07. CÂY NHỊ PHÂN

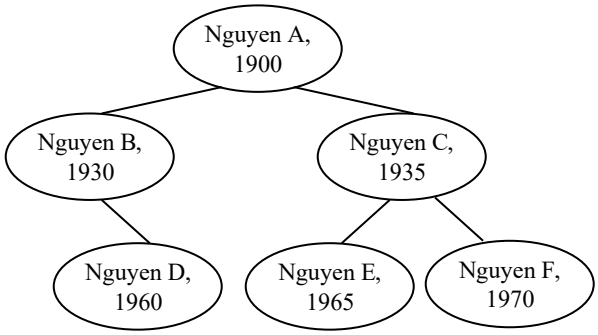
Mục tiêu: thành thạo các thao tác trên cây nhị phân làm cơ sở cho thao tác với các cấu trúc cây tiếp theo.

Nội dung thực hành:

Bài 1. Cây gia phả nhị phân

Cho tổ chức dữ liệu của một cây nhị phân mà mỗi nút là một người gồm các thông tin: họ tên, năm sinh.

Khai báo tổ chức dữ liệu và các hàm thực hiện các thao tác sau:



a) Tạo cây như hình vẽ.
b) In danh sách những người có trong cây.
c) Đếm số người trong cây.
d) Tính số thế hệ của cây.
e) Đếm số người sinh trước năm x.
f) Tìm một người trong cây khi biết họ tên.
g) Kiểm tra người tên y có phải con của người tên x không?
h) Cho biết người tên x thuộc thế hệ thứ mấy trong cây.
i) Kiểm tra người tên y có phải con cháu của người tên x không?
j) Liệt kê tất cả con cháu của người tên x.
k) Thay người tên x bằng một người khác trong cây gia phả.
l) Kiểm tra hai người tên x, y có phải anh em không?
m) Liệt kê những người trong cây thuộc thế hệ thứ k
n) Kiểm tra hai cây gia phả có giống nhau không?
o) Thêm người ng vào con của người tên x. Nếu người tên x đã đủ 2 con thì không thêm.

Khai báo kiểu dữ liệu:

```
struct Person
{
  string   name;
  int yearOfBirth;
};
struct BFT  //Binary Family Tree
```

```
{
  Person   data;
  BFT    *left, *right;
};
```

a) Tạo cây

```
//create a node
BFT* createNode(Person p, BFT* left, BFT* right)
{
  return new BFT{p, left, right};
}

//create the specify BFT
BFT* createBFT()
{
  BFT *n1, *n2, *n3, *n4, *n5, *n6;

  n1 = createNode({"Nguyen D", 1960}, nullptr, nullptr);
  n2 = createNode({"Nguyen B", 1930}, nullptr, n1);
  n3 = createNode({"Nguyen E", 1965}, nullptr, nullptr);
  n4 = createNode({"Nguyen F", 1970}, nullptr, nullptr);
  n5 = createNode({"Nguyen C", 1935}, n3, n4);
  n6 = createNode({"Nguyen A", 1900}, n2, n5);
  return n6;
}
```

b) Hàm in cây

Dùng thuật toán duyệt theo thứ tự trước với thao tác thăm 1 nút là in dữ liệu tại nút đó lên màn hình (gồm in họ tên và năm sinh).

```
//prints a binary family tree in a pre-order traversal
void printBFT(BFT* root)
{
    if (root)
    {
    cout << root->data.name << " " << root->data.yearOfBirth << endl;
    printBFT(root->left);
    printBFT(root->right);
```

```
    }
}
```

c) Đếm số người trong cây

```
//count persons in BFT
int countPersons(BFT* root)
{
    if (!root) return 0;
    return 1 + countPersons(root->left) + countPersons(root->right);
}
```

d) Tính số thế hệ của cây

```
//height of BFT
int height(BFT* root)
{
    if (!root) return 0;
    return 1 + max(height(root->left), height(root->right));
}
```

e) Đếm số người sinh trước năm x

Thuật toán:

Nếu cây rỗng thì số người sinh trước năm x là 0

Ngược lại:

Nếu nút gốc là người sinh trước năm x thì:

Gọi d1 là số người sinh trước năm x ở cây con trái

Gọi d2 là số người sinh trước năm x ở cây con phải

Số người sinh trước năm x của cây bằng d1+d2+1

Ngược lại số người sinh trước năm x là 0

Cài đặt:

```
//Count the number of people whose birth year is less than a specific
year
int countPersonsLessThan(BFT* root, int year)
{
    if (!root) return 0;
    else
    if (root->data.yearOfBirth < year)
```

```
      return      countPersonsLessThan(root->left,      year)      +
countPersonsLessThan(root->right, year)+1;
    else
      return 0;
}
```

f) Tìm một người theo họ tên

Thuật toán:
Nếu cây rỗng thì không tìm thấy
Ngược lại:
  Nếu nút gốc chứa người có họ tên x thì tìm thấy tại nút gốc
  Ngược lại:
    Tìm người tên x ở cây con trái
    Nếu tìm được thì kết quả tìm thấy
    Ngược lại thì tìm người tên x ở cây con phải

Cài đặt:

```
//find a person in BFT by name
BFT* findPerson(BFT* root, string name)
{
    if (!root) return nullptr;
    if (root->data.name == name) return root;
    BFT* left = findPerson(root->left, name);
    if (left) return left;
    return findPerson(root->right, name);
}
```

g) Kiểm tra người tên cName có phải con của người tên pName không?

```
//check pName is parent of cName
bool isParent(BFT* root, string pName, string cName)
{
    BFT* parent;
    parent = findPerson(root, pName);
    if (!parent)
    return false;
    else
    return (parent->left && parent->left->data.name==cName) ||
```

```
    (parent->right && parent->right->data.name==cName);
}
```

Hàm main()

```
int main()
{
  BFT *root, *result;
  root = createBFT();
  printBFT(root);
  cout << "Number of persons in BTS is: " << countPersons(root) << endl;
  cout << "Height of BTS is: " << height(root) << endl;
  result = findPerson(root, "Nguyen E");
  if (result)
  {
    cout << result->data.name <<" "<<result->data.yearOfBirth<< endl;
  }
  else
    cout << "Not found!";
  cout << "Number person YOB < 1965 is: " << countPersonsLessThan(root,
1965) << endl;

  string pName = "Nguyen B", cName = "Nguyen E";
  bool chk = isParent(root, pName, cName);
  if (chk)
    cout << pName << " is parent of " << cName << endl;
  else
    cout << pName << " is not parent of " << cName << endl;
  return 0;
}
```

h) Cho biết người tên x thuộc thế hệ thứ mấy trong cây.

Hướng dẫn: sử dụng thuật toán tương tự thuật toán tìm một người trong cây theo họ tên.

Thuật toán:

- Nếu cây rỗng thì kết quả là 0 (người x không có trong cây thì quy ước thế hệ 0).
- Ngược lại:
  + Nếu nút gốc là người tên x thì thế hệ là 1
  + Ngược lại:

Nếu thế hệ của người x trong cây con trái là h1 khác 0 thì thế hệ của người x trong cây là h1+1

Ngược lại

Nếu thế hệ của người x trong cây con phải là h2 khác 0 thì thế hệ của người x trong cây là h2+1

Ngược lại thì thế hệ của người x trong cây là 0.

Cài đặt:

```
int levelOfPerson(BFT* root, string name)
{




}
```

i) Kiểm tra người tên y có phải con cháu của người tên x không?

Hướng dẫn: sử dụng hàm findPerson().

```
bool isDescendants(BFT* root, string x, string y)
{




}
```

j) Liệt kê tất cả con cháu của người tên x.

```
void printDescendants(BFT* root, string x)
{




}
```

k) Thay người tên x bằng một người khác trong cây gia phả.

```
void setPerson(BFT* root, string x, Person p)
{
```

```
}
```

l) Kiểm tra hai người tên x, y có phải anh em không?

Gợi ý: duyệt từng nút của cây nếu nút có hai nút con và hai con này có tên là x và y thì x và y là anh em.

```
bool isSibling(BFT* root, string x, string y)
{




}
```

m) Liệt kê những người trong cây thuộc thế hệ thứ k

Gợi ý: viết hàm đệ quy theo cấp của nút đang duyệt. Nếu cấp của nút đang duyệt bằng k thì in người tại nút đó, nếu chưa bằng k thì xuống nút con với cấp đang duyệt tăng lên 1.

```
void printPersonsAtLevel(BFT* root, int k, int currLevel)
{




}
```

n) Kiểm tra hai cây gia phả có giống nhau không?

Gợi ý: hai cây giống nhau nếu dữ liệu tại gốc giống nhau và cây con trái giống nhau, cây con phải giống nhau.

```
bool isSameBFT(BFT* root1, BFT* root2)
{



```

```
}
```

o) Thêm người ng vào con của người tên x. Nếu người tên x đã đủ 2 con thì không thêm.

Gợi ý: tìm nút chứa người tên x. Kiểm tra nút đó có nút con rỗng thì tạo nút chứa người cần thêm và gắn vào như nút con.

```
bool addPerson(BFT* root, string x, Person p)
{




}
```

Bài 2. Xây dựng cây nhị phân

Cho một cây nhị phân mà mỗi nút là một ký tự. Cho biết thứ tự duyệt theo thứ tự trước và thứ tự giữa của cây. Hãy xây dựng cây.

```
```