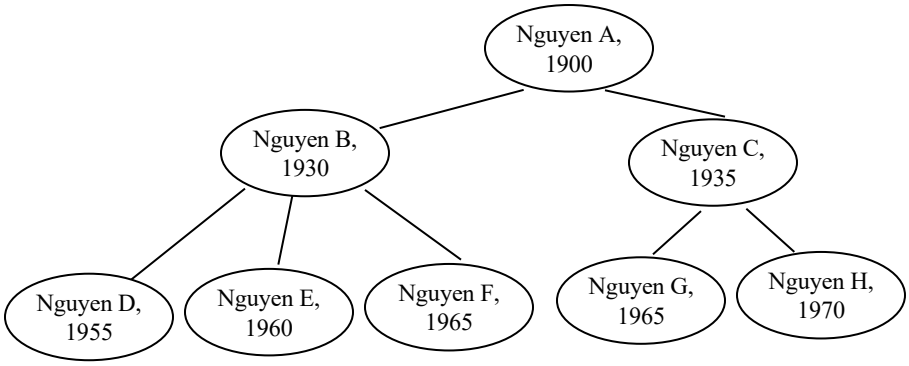
Bài thực hành số 09. CÂY TỔNG QUÁT

Mục tiêu: thành thạo với các thao tác trên cây tổng quát và làm quen với cây tìm kiếm tổng quát cụ thể là cây từ điển tiếng Anh,

Nội dung thực hành:

Bài 1. Cây gia phả

Để quản lý một cây gia phả, mỗi nút là một người gồm các thông tin: họ tên, năm sinh. Hãy khai báo tổ chức dữ liệu cây tổng quát bằng liên kết các nút anh em lưu cây gia phả.



Khai báo tổ chức dữ liệu và viết các hàm:

a) Tạo cây gia phả như hình.

b) In cây gia phả theo thứ tự thế hệ (dùng thuật toán duyệt theo chiều rộng).

c) Đếm số người có trong cây gia phả

d) Tính số thế hệ của cây gia phả

e) Tìm xem một người có họ tên x có trong cây gia phả không?

f) Kiểm tra người tên y có phải con của người tên x trong cây gia phả không?

g) Thêm người q vào con của người tên x trong cây. Biết rằng các nút con của một người được sắp theo thứ tự tăng của năm sinh.

h) Liệt kê con, cháu của một người có họ tên x.

i) In những người thuộc thế hệ thứ k.

j) Tính bậc của cây.

k) Xóa người trên cây có tên x (x không phải ông tổ của dòng họ).

Hướng dẫn:

Tổ chức dữ liệu:

```
struct Person
```

```
{
  string   name;
  int yearOfBirth;
};

struct FT  //Family Tree
{
  Person   data;
  FT  *child, *sibling;
};
```

a) Tạo cây gia phả

```
//create a node
FT* createNode(Person p, FT* child, FT* sibling)
{
  return new FT{p, child, sibling};
}

//create the specify FT
FT* createFT()
{
  FT *n1, *n2, *n3, *n4, *n5, *n6, *n7, *n8;

  n1 = createNode({"Nguyen H", 1970}, nullptr, nullptr);
  n2 = createNode({"Nguyen G", 1965}, nullptr, n1);
  n3 = createNode({"Nguyen F", 1965}, nullptr, nullptr);
  n4 = createNode({"Nguyen E", 1960}, nullptr, n3);
  n5 = createNode({"Nguyen D", 1955}, nullptr, n4);
  n6 = createNode({"Nguyen C", 1935}, n2, nullptr);
  n7 = createNode({"Nguyen B", 1930}, n5, n6);
  n8 = createNode({"Nguyen A", 1900}, n7, nullptr);
  return n8;
}
```

b) In cây gia phả theo thứ tự thế hệ thì dùng thuật toán duyệt cây tổng quát theo chiều rộng.

```
//prints a family tree in a breadth first search
void printFTBFS(FT* root)
```

```
{
  queue<FT*> q;
  FT *p;
    if (root)
    {
       q.push(root);
       while (!q.empty())
       {
         p = q.front();
         q.pop();
         cout << p->data.name << " " << p->data.yearOfBirth << endl;
         p = p->child;
         while (p)
         {
           q.push(p);
           p = p->sibling;
         }
       }
    }
}
```

c) Đếm số người trong cây gia phả:

Thuật toán:
+ Nếu cây rỗng thì số người là 0
+ Ngược lại: thì đếm số người trong từng cây con cộng lại và cộng thêm 1 là số người trong cả cây.

Cài đặt:

```
int countPerson(FT* root)
{




}
```

d) Tính số thế hệ trong cây gia phả:

Gợi ý: tìm số lớn trong chiều cao của các cây con rồi cộng thêm 1.

```
int height(FT* root)
{









}
```

e) Tìm 1 người trong cây gia phả theo họ tên:

Thuật toán:

+ Nếu cây rỗng thì không tìm thấy

+ Ngược lại:

Nếu nút gốc là người có họ tên bằng họ tên cần tìm thì tìm thấy tại nút gốc.

Ngược lại thì lần lượt tìm trong từng cây con, nếu tìm thấy ở cây con nào thì dừng ở đó với kết quả tìm thấy

Nếu tìm hết trong các cây con mà không tìm thấy thì kết quả là không tìm thấy trong cả cây.

Cài đặt:

```
//find a person by name
FT* search(FT *root, string name)
{
    FT *p, *result;
    if(!root)
      return nullptr;
    else
    if(root->data.name == name)
      return root;
     else
     {
         p = root->child;
         while(p)
         {
           result = search(p, name);
```

```
            if(result)
              return result;
            else
              p = p->sibling;
          }
          return nullptr;
      }
}
```

f) Kiểm tra người tên y có phải con của người tên x trong cây gia phả không?

Gợi ý: tìm người tên x trong cây tại p. Sau đó tìm người tên y trong cây nút gốc là p.

```
bool isChild(FT* root, string x, string y)
{




}
```

g) Thêm người q vào con của người tên x trong cây. Biết rằng các nút con của một người được sắp theo thứ tự tăng của năm sinh.

Gợi ý: tìm người tên x trong cây tại nút p. Duyệt các nút con của nút p để tìm vị trí phù hợp thêm người q vào.

```
void insertChild(FT* root, string x, Person q)
{




}
```

h) Liệt kê con, cháu của một người có họ tên x.

Gợi ý: tìm người tên x trong cây tại nút p. Sau đó in tất cả những cây có gốc là con của p.

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

```
}
```

i) In những người thuộc thế hệ thứ k.

```
void printByLevel(FT* root, int k)
{




}
```

j) Tính bậc của cây.

Gợi ý: duyệt từng nút, mỗi nút đếm số nút con sau đó chọn số nút con lớn nhất.

```
int degree(FT* root)
{




}
```

k) Xóa người trên cây có tên x (x không phải ông tổ của dòng họ).

Gợi ý: Tìm người tên x tại nút p và p1 là nút cha của p và p2 là nút anh liền kề của p. Nếu p là nút con trưởng thì chuyển nút con của p1 là em liền kề của p. Ngược lại chuyển em liền kề của p2 là em liền kề của p. Sau đó xóa toàn bộ cây nút gốc là p.

```
void deleteByName(FT* root, string x)
{




}
```

Bài 2. Cây từ điển tiếng Anh

Chương trình EngDict.cpp minh họa cây từ điển dùng cây tìm kiếm tổng quát với các thao tác: đọc từ điển từ tệp, kiểm tra một từ có trong từ điển không?

Tổ chức dữ liệu: mỗi nút gồm mảng 26 nút con tương ứng cho 26 chữ cái và trường isEnOfWord để đánh dấu nút kết thúc của một từ.

```
#define ALPHABET_SIZE 26

struct TrieNode {
    TrieNode* children[ALPHABET_SIZE];
    bool isEndOfWord;
};
```

Tạo nút của cây từ điển:

```
TrieNode* createNode() {
    TrieNode* node = new TrieNode;
    node->isEndOfWord = false;
    for (int i = 0; i < ALPHABET_SIZE; i++) {
        node->children[i] = nullptr;
    }
    return node;
}
```

Thêm một từ vào cây từ điển:

Duyệt từng ký tự của từ cần thêm, kiểm tra có ký tự trong cây từ điển ở cấp tương ứng không? Nếu chưa có thì tạo nút mới. Cuối từ tạo nút kết thúc từ.

```
void insert(TrieNode* root, const string word) {
    TrieNode* node = root;
    for (int i = 0; i < word.size(); i++) {
        int index = word[i] - 'a';
        if (node->children[index] == nullptr) {
            node->children[index] = createNode();
        }
        node = node->children[index];
    }
    node->isEndOfWord = true;
}
```

Đọc một từ điển từ tệp:

```
int readDictFromFile(string fileName, TrieNode* root)
{
  ifstream file(fileName);
  string word;
  int count = 0;
  while (getline(file, word))
  {
    insertWord(root, word);
    count++;
  }
  file.close();
  return count;
}
```

Tìm một từ trong cây từ điển

```
bool search(TrieNode* root, const string word) {
    TrieNode* node = root;
    for (int i = 0; i < word.size(); i++) {
        int index = word[i] - 'a';
        if (node->children[index] == nullptr) {
            return false;
        }
        node = node->children[index];
    }
    return node->isEndOfWord;
}
```

In từ điển:

Duyệt từng nút trong cây từ điển, với mỗi nút lấy ký tự tương ứng ghép vào từ prefix. Khi gặp nút kết thúc từ thì in từ prefix ra.

```
void printWords(TrieNode* node, string prefix) {
        if (node == nullptr) {
            return;
        }
        if (node->isEndOfWord) {
            cout << prefix << endl;
```

```
        }
        for (int i = 0; i < ALPHABET_SIZE; i++) {
            if (node->children[i] != nullptr) {
                char ch = 'a' + i;
                printWords(node->children[i], prefix + ch);
            }
        }
    }

    void printAllWords(TrieNode *root) {
        printWords(root, "");
    }
```

Hàm main

```
int main()
{
  TrieNode *root;int k; string w="";
  root = createNode();
  cout << "Loading...";
  k = readDictFromFile("words_alpha.txt", root);
  cout << k << " words finished" << endl;
  cout << "Searching..." << endl;
  bool b = find(root, "dichlorodiphenyltrichloroethane");
  if (b)
    cout << "found the word \'dichlorodiphenyltrichloroethane\' in
dictionary" << endl;
  else
    cout <<     "not found the word \'dichlorodiphenyltrichloroethane\'
in dictionary" << endl;
  printEngDict(root, w);
}
```

Sửa lại chương trình để nhập vào một từ rồi kiểm tra từ này có trong từ điển không?

Cho biết chiều cao của cây từ điển nạp vào là bao nhiêu?

Cây có bao nhiêu từ?

Nút có ít nút con nhất là bao nhiêu (trừ các nút lá)?

Liệt kê tất cả những từ bắt đầu là xâu s.

Liệt kê những từ dài nhất trong từ điển.

Bài 3. Cây thư mục

Khai báo tổ chức dữ liệu để lưu cây thư mục của một máy tính gồm các thư mục và các tệp. Viết các hàm thực hiện các chức năng:

a) Tạo một cây thư mục cụ thể.
b) In cây thư mục.
c) Tìm một tập tin hay thư mục trong cây.
d) Tính tổng kích thước các tập tin trong cây thư mục.
e) Liệt kê tất cả đường dẫn đến tập tin tên fileName trong cây thư mục.
f) Xóa một tập tin trong cây thư mục.
g) Xóa một thư mục trong cây thư mục.
