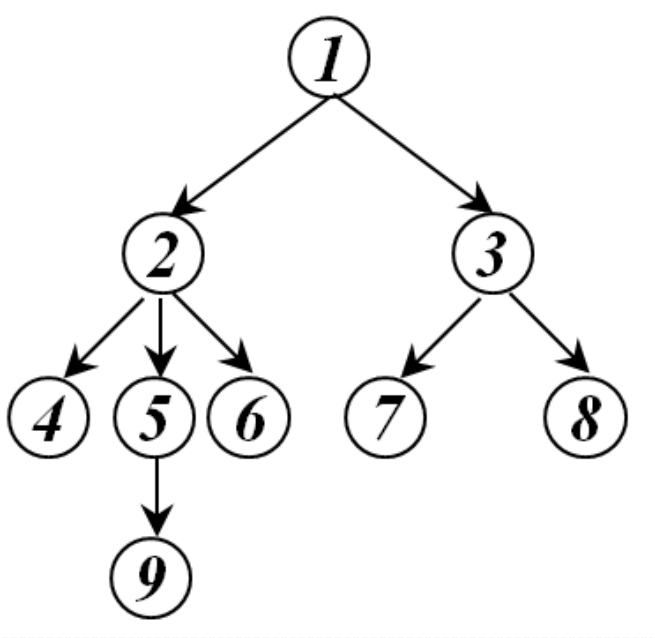
Cây tổng quát

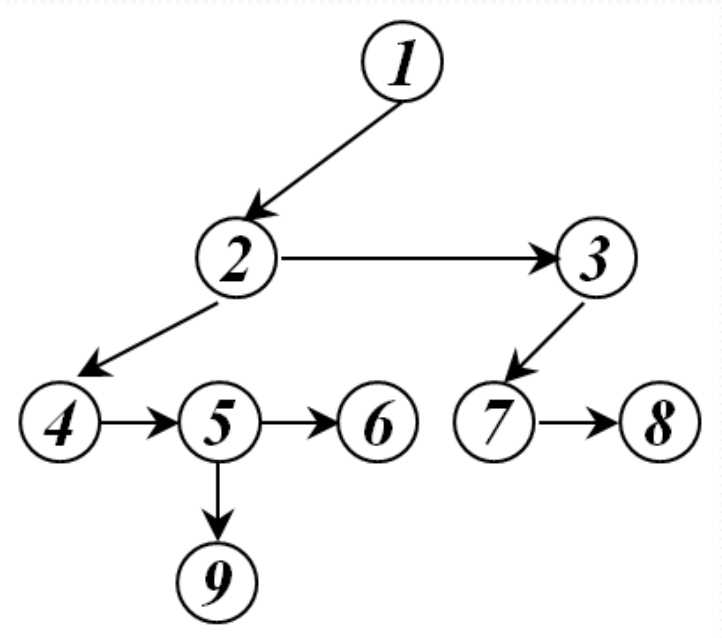
Nội dung

1. Tổ chức dữ liệu
2. Duyệt cây
    1. Duyệt theo chiều sâu
    2. Duyệt theo chiều rộng
3. Cây tìm kiếm tổng quát
4. Bài tập

1. Tổ chức dữ liệu

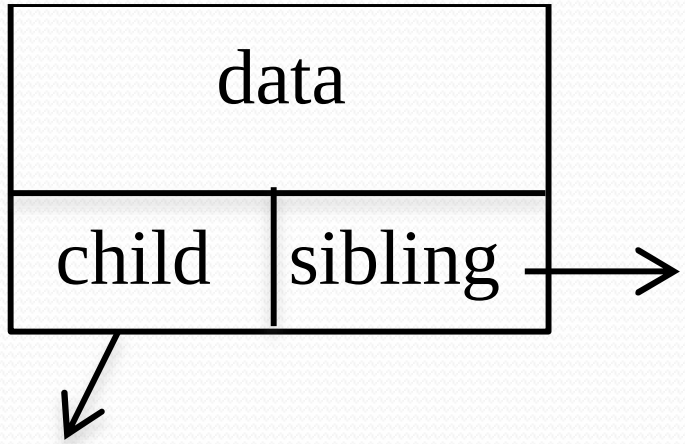
- Dùng liên kết các nút anh em





1. Tổ chức dữ liệu

- Mỗi nút gồm:
  - data: lưu một phần tử
  - child: lưu địa chỉ nút con đầu tiên (nếu có)
  - sibling: lưu địa chỉ nút em liền kề (nếu có)



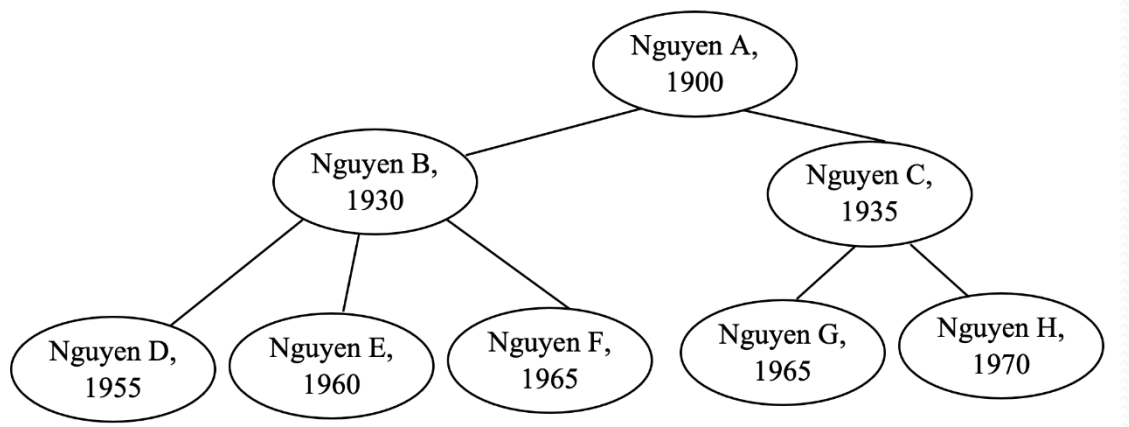
- Khai báo:

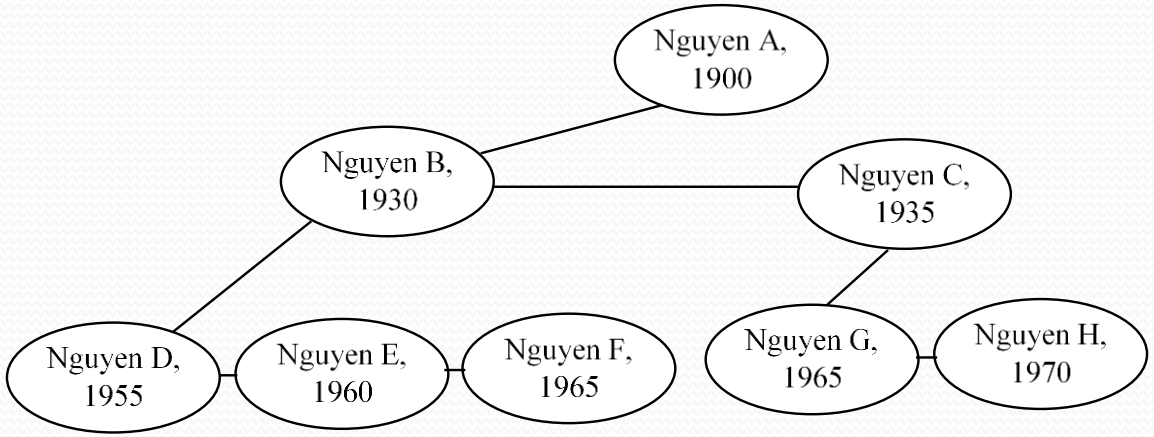
```
struct TreeNode
{
  ElementType data;
  TreeNode *child, *sibling;
};
```

Ví dụ: Cây gia phả

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

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





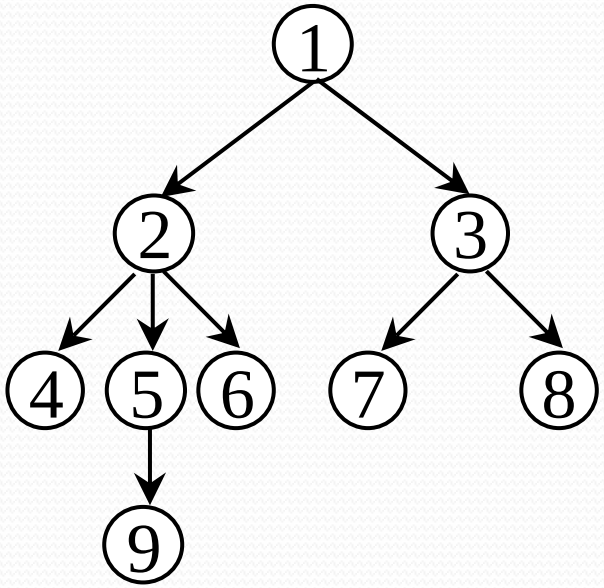
2. Duyệt cây

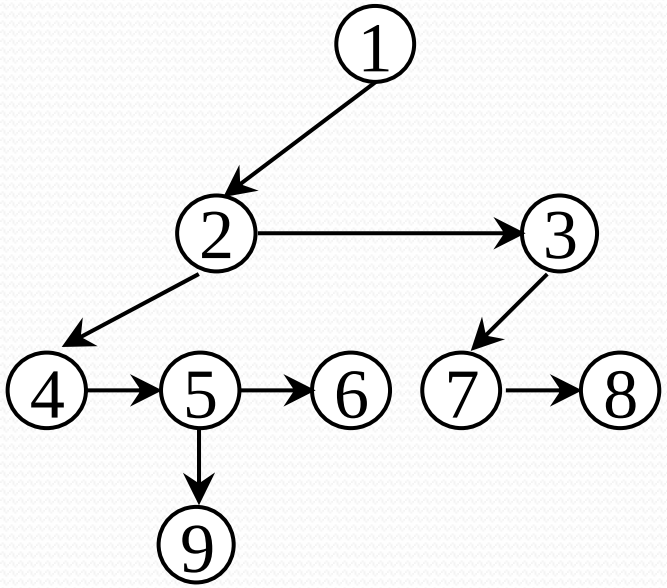
- Thuật toán duyệt cây tổng quát theo chiều sâu

```
Input: Cây cần duyệt có nút gốc là root
Output: Thứ tự các nút của cây duyệt theo chiều sâu
Action:
Nếu nút gốc khác rỗng:
   + Thăm nút gốc
   + Với mỗi nút con r của nút gốc:
       - Duyệt theo chiều sâu cho cây con có nút gốc là r
```

Ví dụ

- Duyệt theo chiều sâu: 1, 2, 4, 5, 9, 6, 3, 7, 8





Cài đặt:

```
void DFS(TreeNode *root)
{
	if(root != nullptr)
	{
		TreeNode *r;
		visit(root);
		r = root->child;
		while(r != nullptr)
		{
			DFS(r);
			r = r->sibling;
		}
	}
}
```

- Hoặc cách khác:

Nếu cây khác rỗng:

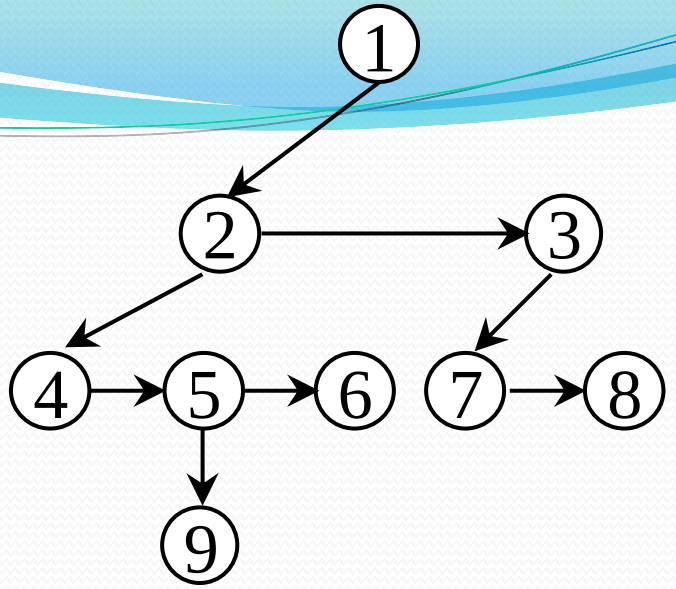
+ Thăm nút gốc

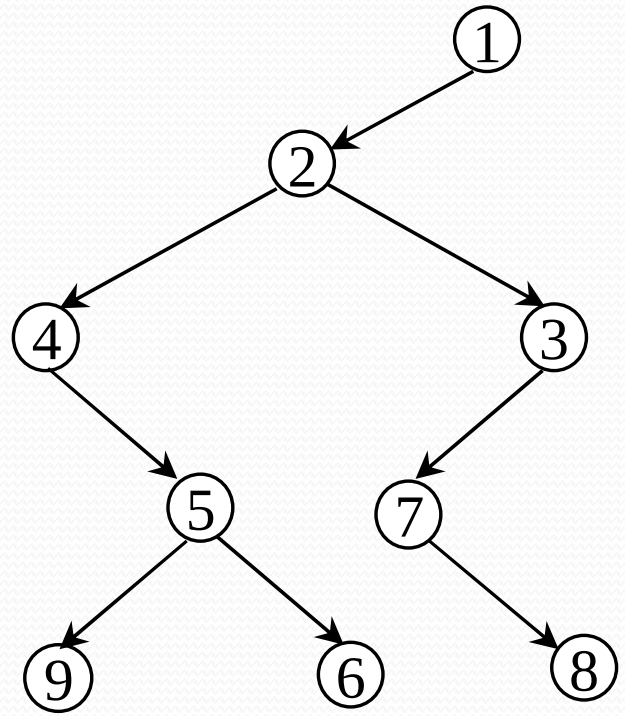
+ Duyệt theo chiều sâu xuất phát nút con

+ Duyệt theo chiều sâu xuất phát từ nút em

- Cài đặt:

```
void DFS2(TreeNode *root)
{
  if(root != nullptr)
  {
    visit(root);
    DFS2(root->child);
    DFS2(root->sibling);
  }
}
```





Duyệt theo chiều rộng

Input: Cây cần duyệt có nút gốc là root

Output: Thứ tự các nút của cây duyệt theo chiều rộng

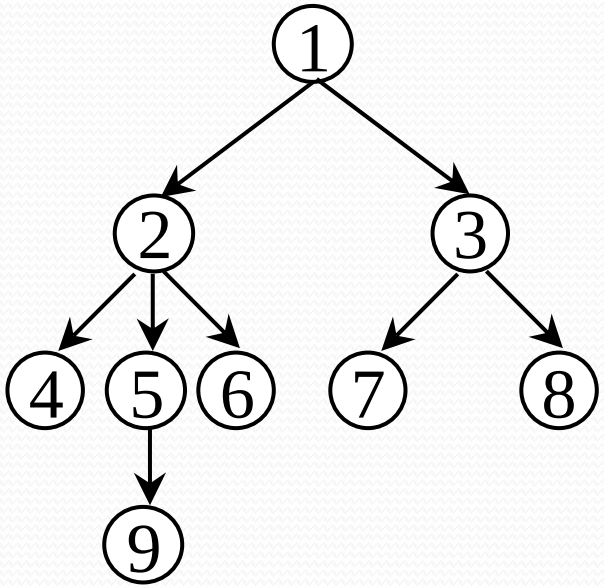
Action:

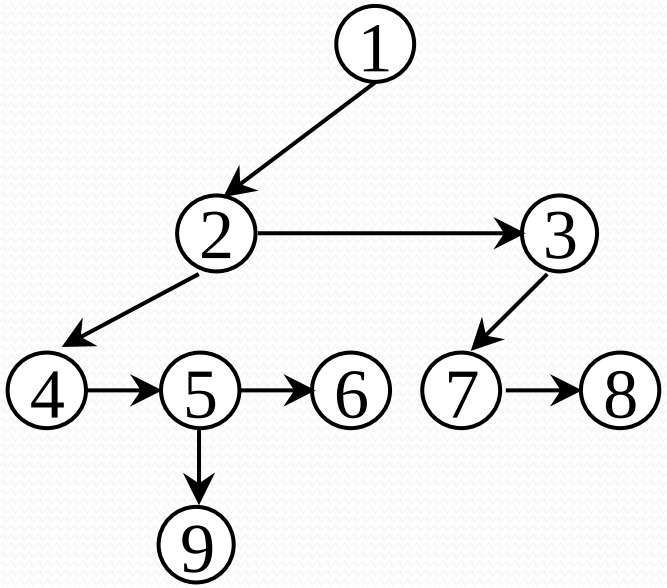
Nếu cây khác rỗng:

+ Thăm nút gốc
+ Thăm lần lượt các nút con của nút gốc
+ Với nút vừa thăm ở bước trên lần lượt thăm các nút con của nó
+ Lặp lại bước trên cho đến khi thăm hết các nút của cây

Ví dụ

- Duyệt theo chiều rộng: 1, 2, 3, 4, 5, 6, 7, 8, 9





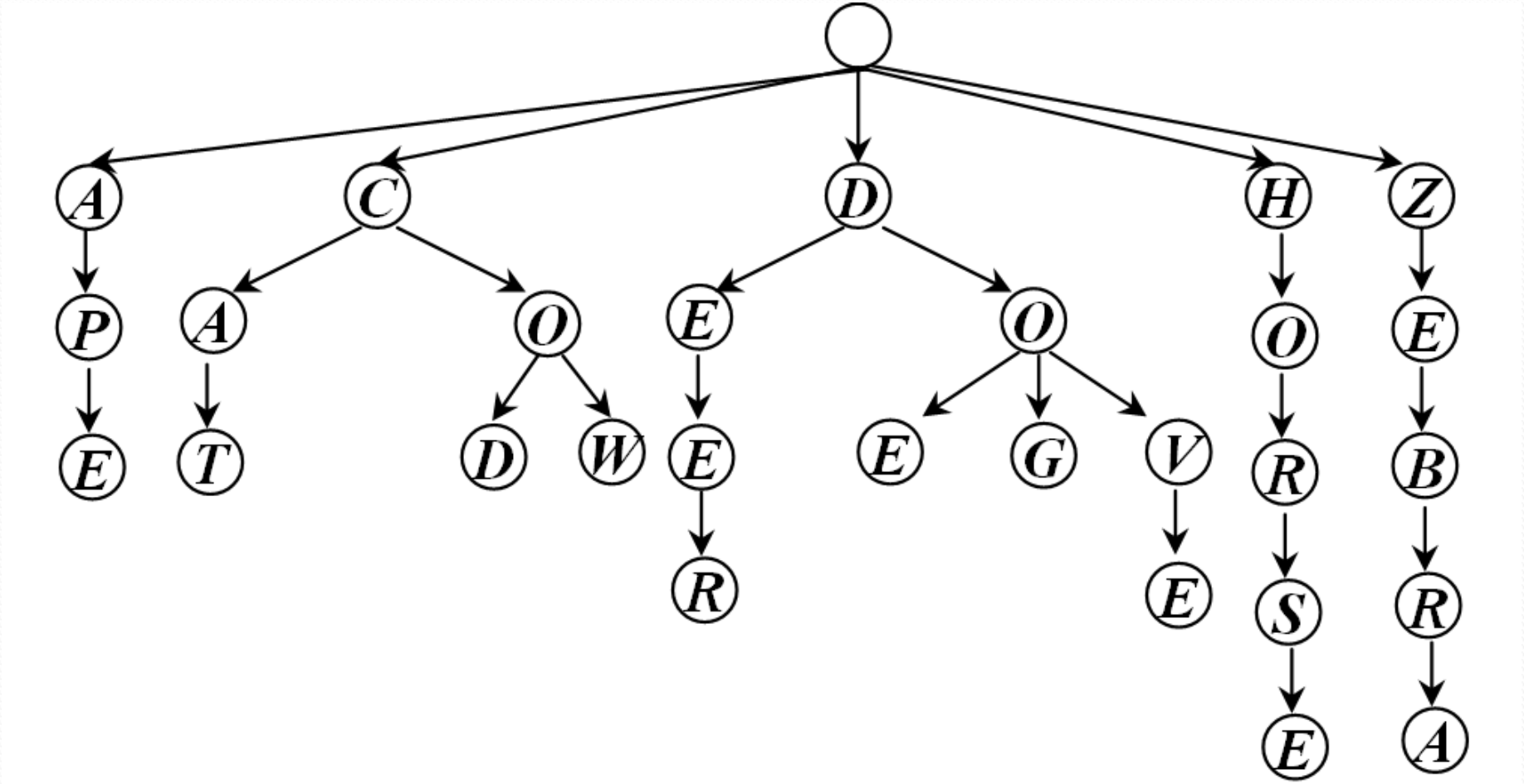
- Cài đặt: dùng hàng đợi q để lưu các nút trong quá trình duyệt.

```
void BFS(TreeNode *root)
{
  queue<TreeNode*> q;
  TreeNode *r;
  if (root != nullptr) {
    q.push(root);
    while (!q.empty()){
      r = q.front(); q.pop();
      visit(r);
      r = r->child;
      while (r != nullptr){
        q.push(r);
        r = r->sibling;
      }
    }
  }
}
```

3. Cây tìm kiếm tổng quát

- Cây tìm kiếm tổng quát:
  - Là cây tổng quát
  - Các nút con của 1 nút sắp theo thứ tự của khóa
- Ví dụ: xét một từ điển gồm các từ sau đây: APE, CAT, COD, COW, DEER, DOE, DOG, DOVE, HORSE, và ZEBRA.

- APE, CAT, COD, COW, DEER, DOE, DOG, DOVE, HORSE, và ZEBRA

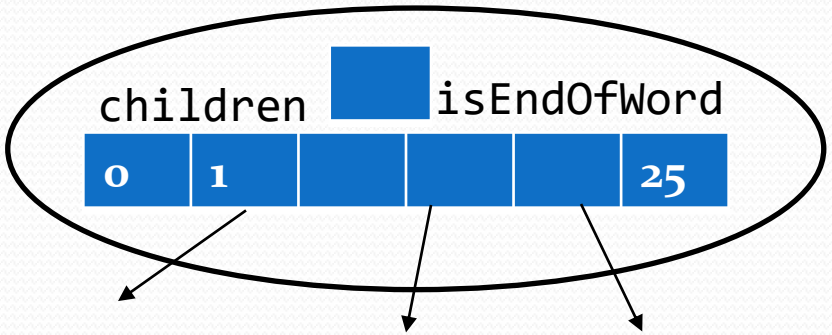


Ví dụ ứng dụng: Cây từ điển

- Cây từ điển tiếng Anh.

```
#define ALPHABET_SIZE 26

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



- Hàm tạo một nút trong Trie:
  - Cấp phát bộ nhớ cho một nút
  - Gán trường isEndOfWord là false
  - Gán các nút con rỗng

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

- Hàm chèn một từ word vào từ điển:
  - Xuất phát từ nút gốc của Trie lưu từ điển
  - Duyệt từng ký tự trong từ cần thêm
  - Xác định vị trí của ký tự trong mảng các nút con
  - Nếu từ đó chưa có trong nút con thì thêm vào nút con ứng với ký tự đang xét
  - Chuyển nút đến nút ứng với ký tự đang xét
  - Đánh dấu kết thúc từ cho nút đang xét

```
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;
}
```

- Hàm tìm một từ trong từ điển:

- Xuất phát từ nút gốc của trie lưu từ điển
- Duyệt từng ký tự của từ cần tìm
- Nếu ký tự không có trong nút con của nút đang xét thì trả về false
- Chuyển nút đang xét sang nút con tương ứng với ký tự đang xét
- Trả về trường isEndOfWord của nút đang xét

```
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;
}
```

- Hàm in tất cả các từ trong từ điển:

Duyệt từ nút gốc của cây với từ ban đầu prefix là rỗng.

+ Nếu là nút rỗng thì dừng

+ Nếu là nút kết thúc một từ thì in từ đang có trong prefix

+ Duyệt qua các nút con:

- Mỗi lần qua một nút thì thêm ký tự tương ứng với nút vào prefix rồi duyệt đệ quy cho nút con này

```
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, "");
    }
```

```
int main() {
    TrieNode* root = createNode();

    insert(root, "apple");
    insert(root, "banana");
    insert(root, "cat");
    insert(root, "dog");

    cout << search(root, "apple") << endl; // prints 1
    cout << search(root, "pear") << endl; // prints 0

  printAllWords(root);
    return 0;
}
```

4. Bài tập

Bài 7.1 Khai báo cây gia phả kiểu cây tổng quát mỗi nút là 1 người gồm họ tên và năm sinh. Trình bày thuật toán và cài đặt các thao tác:

a) Đếm số người trong cây gia phả
b) Tính số thế hệ của cây gia phả
c) Tìm một người tên x trong cây gia phả
d) Kiểm tra người tên y có phải con người tên x không?
e) Tính thế hệ của người tên x
f) Liệt kê tất cả những người con của người tên x
g) Liệt kê những người thuộc thế hệ thứ k
h) Cho biết người tên x và y có phải là anh em không?
i) Thêm một người q là con của người tên x.

Bài 7.1a

Thuật toán:

Input: Cây gia phả cần đếm có nút gốc là root

Output: Số người trong cây gia phả

Action:

+ 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.

```
int countPersons(FT *root)
{
	if(root == nullptr) return 0;
	else
	{
		int d = 1;
		FT *p = root->child;
		while (p != nullptr)
		{
			d = d + countPersons(p);
			p = p->sibling;
		}
		return d;
	}
}
```

Bài 7.1b

Thuật toán:

Input: Cây gia phả cần đếm có nút gốc là root

Output: Số thế hệ của cây gia phả

Action:

+ Nếu cây rỗng thì số thế hệ là 0

+ Ngược lại thì tìm số thế hệ lớn nhất trong từng cây con và cộng thêm 1 là số thế hệ của cây.

```
int height(FT *root)
{
	if(root == nullptr) return 0;
	else
	{
		int h, max = 0;
		FT *p = root->child;
		while (p != nullptr)
		{
			h = height(p);
			if (h > max) max = h;
			p = p->sibling;
		}
		return max + 1;
	}
}
```

Bài 7.1c

Thuật toán:

- Input: Cây gia phả, họ tên cần tìm x
- Output: True nếu có người tên x trong cây gia phả, False nếu ngược lại
- Action:
- Nếu cây rỗng thì trả về False
- Ngược lại:
  - Nếu nút gốc là người có họ tên bằng họ tên cần tìm thì trả về True.
  - Ngược lại thì lần lượt tìm trong từng cây con, nếu tìm thấy ở cây con nào thì trả về True
  - Nếu tìm hết trong các cây con mà không tìm thấy thì trả về False.

```
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;
     }
}
```

Bài 7.1d

- Kiểm tra người tên y có phải con người tên x không?

Thuật toán:

Input: Cây gia phả có nút gốc root, x, y là tên của hai người

Output: True nếu y là con người tên x; False ngược lại

Action:

Tìm nút r chứa người tên x trong cây có gốc là root

Nếu r rỗng thì trả về False

Ngược lại:

    Duyệt các nút con của r

        Nếu y là tên của nút đang xét thì trả về True

    Trả về False

```
bool isParent(FT* root, string x, string y)
{
   FT* r;
   r = search(root, x);
   if (r == nullptr) return False;
   else{
     FT* p = r->child;
     while (p != nullptr){
       if (p->data.name == y) return True;
       else p = p->sibling;
    }
    return False;
 }
}
```

Bài 7.1e

- Tính thế hệ người tên x

Thuật toán:

Input: Cây gia phả, họ tên x

Output: thế hệ người tên x nếu có, hoặc 0 nếu không có.

Action:

- Nếu cây rỗng thì trả về 0
- Ngược lại:
  - Nếu nút gốc là người tên x thì trả về 1
  - Ngược lại:
    - Lần lượt tìm thế hệ người tên x ở các cây con
    - Nếu tìm thấy ở cây con với thế hệ là h thì trả về h+1
    - Nếu không có ở các cây con thì trả về 0

```
int level(FT* root, string x)
{
   if (roor == nullptr) return 0;
   else
     if (root->data.name == x) return 1;
     else
     {
        p = root->child;
        while (p != nullptr)
        {
           int h = level(p, x);
           if (h > 0) return h+1;
           else p = p->sibling;
        }
     }
}
```

Bài 7.1f

- Liệt kê tất cả những người con của người tên x
- Thuật toán:

Input: Cây gia phả có gốc là root, tên x cần liệt kê con cháu

Output: Tất cả con cháu của người tên x (nếu có)

Action:

Tìm r là nút chứa người tên x trong cây gốc root

Nếu r khác rỗng thì:

    Duyệt các nút con p của nút r:

        In tên những người trong nút p

```
void printChildren(FT* root, string x)
{
    FT* r = search(root, x);
    if (r != nullptr)
    {
        FT* p = r->child;
        while (p != nullptr)
        {
            cout << p->data.name << endl;
            p = p->sibling;
        }
    }
}
```

Bài 7.1g

- Liệt kê những người thuộc thế hệ thứ k trong cây gia phả

Thuật toán:

Input: Cây gia phả có nút gốc root, số nguyên dương k

Output: Những người thế hệ k trong cây gia phả

Action:

Nếu cây khác rỗng:

    Nếu k = 1 thì in nút gốc

    Ngược lại:

        Duyệt từng cây con

            In thế hệ thứ k-1 của cây con

```
void printLevel(FT* root, int k)
{
  if (root != nullptr)
    if (k == 1)
     cout <<root->data.name <<“ “<<root->data.age<<“ ”<<root->data.gpa <<endl;
    else
    {
      p = root->child;
      while (p != nullptr)
      {
         printLevel(p, k-1);
         p = p->sibling;
      }
    }
}
```

Bài 7.1h

- Cho biết người tên x và y có phải là anh em không?
- Thuật toán:

Input: Cây gia phả gốc là root, tên hai người x, y
Output: True nếu x và y là anh em
Action:
Nếu cây rỗng thì trả về False
Ngược lại:
    Nếu x, y có trong con của gốc thì trả về True
    Ngược lại duyệt từng nút con r của gốc:
        Kiểm tra x và y có phải là anh em trong cây gốc r không, nếu có trả về True
    Trả về False

```
bool isSibling(FT* root, string x, string y)
{
    if (root == nullptr) return False;
    FT* r = root->child;
    bool chkx = chky = False;
    while (r != nullptr)
    {
        if (r->data.name == x)
        {
            chkx = True;
            if (chky) return True;
        }
        if (r->data.name == y)
        {
            chky = True;
            if (chkx) return True;
        }
        r = r->sibling;
    }
```

```
    r = root->child;
    while (r != nullptr)
    {
        chk = isSibling(r, x, y);
        if (chk) return True;
        r = r->sibling;
    }
}
```

Bài 7.1i

- Thêm một người q là con của người tên x (nếu có).

Thuật toán:

Input: Cây gia phả có gốc là root, người q, tên x

Output: Cây sau khi thêm q vào con người tên x (nếu có)

Action:

Tìm nút r chứa người tên x trong cây gốc root

Nếu r khác rỗng:

Duyệt các con của r để thêm q vào các nút con của r theo thứ tự tăng của năm sinh

```
void insertChild(FT* root, Person p, string x)
{
  FT* r = search(root, x);
  if (r != nullptr){
      FT *q = FT{p, nullptr, nullptr};
      FT* p1 = nullptr, p = r->child;
      while (p!=nullptr && p->data.yearOfBirth < p.yearOfBirth){
           p1 = p; p = p->sibling;
      }
      if (p1 == nullptr){ // thêm vào con đầu
          q->sibling = r->child;
          r->child = q;
      }
      else{
         q->sibling = p;
         p1.sibling = q;
     }
  }
}
```

Bài tập

Bài 7.2 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.

Bài 7.3 Sử dụng cấu trúc cây trie viết chương trình đếm số lần xuất hiện từng từ trong một tệp, chỉ lấy các từ chứa các chữ cái latinh.

Tổng kết

- Cây tổng quát cho phép lưu trữ cây nhiều nút con
- Sử dụng liên kết các nút anh em
- Nhiều thao tác trên cây tổng quát tương tự cây nhị phân