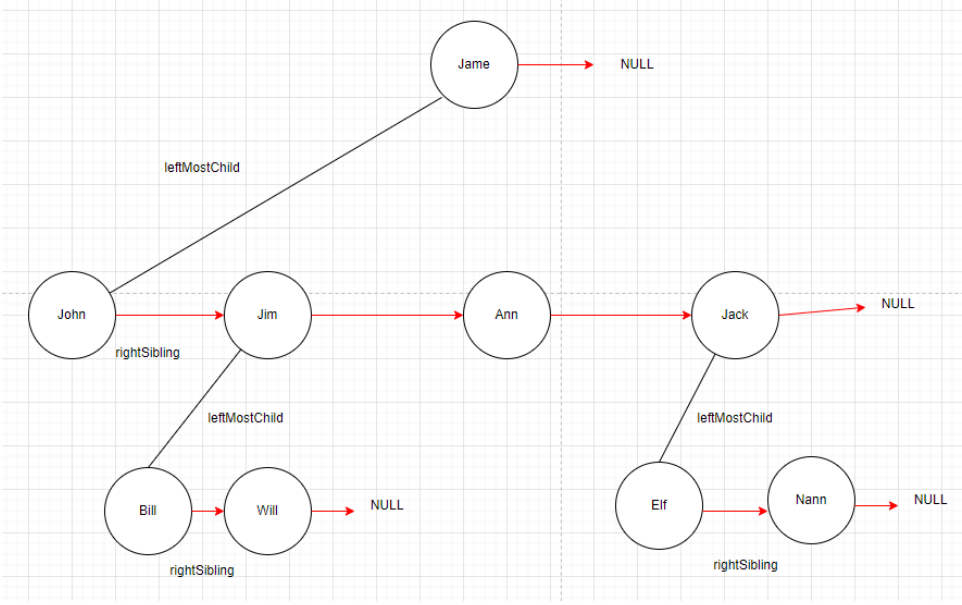
Nội dung

- Cây tổng quát và 1 số thuật toán cơ bản
  - Thêm nút
  - Tìm kiếm
  - Xóa nút




```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

typedef struct Node{
   char name[256];
   struct Node* leftMostChild;
   struct Node* rightSibling;
}Node;

// cap phat bo nho de luu tru nut moi
// gan ten va gia tri leftmostchild, rightsibling la NULL
Node* makeNode(const char* name){
   Node* p = (Node*)malloc(sizeof(Node));
   strcpy(p->name,name);
   p->leftMostChild = NULL; p->rightSibling = NULL;
   return p;
}

// tim va tra ve nut co gia tri la keyname
```

```
// neu name khong co tren cay --> tra ve NULL
// tim de quy = DFS, thoi gian co O(n)
Node* find(Node* r, const char* keyname){
   if(r == NULL) return NULL;
   if(strcmp(r->name,keyname) == 0) return r;
   // neu khong bang --> tim tiep tai cay con cua nut hien tai
   Node* p = r->leftMostChild;
   while(p != NULL){
      Node* q = find(p,keyname);
      if(q != NULL) return q;
      p = p->rightSibling;
   }
}

// them keyname la anh chi em nho nhat cua nut hien tai
// se la anh chi em phai nhat trong cac nut con --> duyet toi con phai nhat hien tai
// sau do se add them vao cuoi
Node* addLast(Node* p, const char* keyname){
   // cap phat nut moi va gan vao
   if(p == NULL) return makeNode(keyname);

   // di chuyen toi anh chi em phai nhat hien tai
   p->rightSibling = addLast(p->rightSibling, keyname);
   return p;
}

// them keyname la anh chi em nho nhat cua nut hien tai khong de quy
Node* addLast_loop(Node* p, const char* keyname)
{
   Node *newChild = makeNode(keyname);
   while(p->rightSibling!=NULL) p = p->rightSibling;
   p->rightSibling = newChild;
}

// them nut child la con nho nhat cua parent
void addChild(Node* root, const char*parent, const char* child){
   Node* r = find(root,parent);
   if(r == NULL) return;
   r->leftMostChild = addLast(r->leftMostChild,child);
}

// ham them con cua nut hien tai khong de quy
void addChild_loop(Node* root, const char* parent, const char* child){
   Node* r = find(root,parent);
   if(r == NULL) return;
   // tao nut con moi
   Node *newChild = makeNode(child);
   // neu nut r chua co con
```

```
    if(r->leftMostChild == NULL)r->leftMostChild = newChild;
    else  // neu da co con thi them vao con nho nhat
    {
        Node *p = r->leftMostChild;
        while(p->rightSibling!=NULL) p = p->rightSibling;
        p->rightSibling = newChild;
    }
}

void printTree(Node* r){
    if(r == NULL) return;
    printf("%s: ",r->name);
    Node* p = r->leftMostChild;
    while(p != NULL){
        printf("%s ",p->name);
        p = p->rightSibling;
    }
    printf("\n");
    p = r->leftMostChild;
    while(p != NULL){
        printTree(p);
        p = p->rightSibling;
    }
}
int main()
{
    Node *root = makeNode("Jame");
    addChild_loop(root, "Jame","John");
    addChild_loop(root, "Jame","Jim");
    addChild_loop(root, "Jame","Ann");
    addChild_loop(root, "Jame","Jack");
    addChild_loop(root, "Jim","Bill");
    addChild_loop(root, "Jim","Will");
    addChild_loop(root, "Jack","Elf");
    addChild_loop(root, "Jack","Nann");
    printTree(root);
    return 0;
}
```

Bài tập 1. thêm hàm **loadDataFromFile**(const char *filename)
Để tạo ra cây từ file

Nội dung file dạng

```
David Jame Peter John $
```

```
Peter Mary Daisy $
Jame Mike Smith $
John Rick David $
Rick Poor $
Rick Vip $
Vip Ann $
$$
```

Đầu dòng chính là nút cha

Dòng đầu tiên là nút gốc và các con của gốc

Tên ngăn cách bởi dấu cách trống

Hàm in ra file dạng vòng lặp

```
// in file KHONG de quy, dung hang doi
void printTreeF_loop(Node* root, const char *filename){
   if(root == NULL) return;
   FILE* f = fopen(filename,"w");
   queue<Node*> Q;
   Q.push(root);

   while(!Q.empty())
   {
      Node* r = Q.front();
      Q.pop();

      fprintf(f,"%s ",r->name);
      Node* p = r->leftMostChild;
      while(p != NULL){
         fprintf(f,"%s ",p->name);
         if(p->leftMostChild!=NULL) Q.push(p);
         p = p->rightSibling;
      }
      fprintf(f,"$\n");
   }
   fprintf(f,"$$\n");
   fclose(f);
}
```

Hàm in ra các nút con của nút có value là keyname

```
// in ra cac nut con cua nut co ten la keyname
void listChildren(Node* root, const char *keyname){
```

```
    Node* p = find(root,keyname);
    if(p == NULL) printf("Not Found %s\n",keyname);
    else{
        printf("Found %s with children: ",keyname);
        Node* q = p->leftMostChild;
        while(q != NULL){
            printf("%s ",q->name);     q = q->rightSibling;
        }
    }
    printf("\n");
}
```

Các hàm xử lý trên cây thường bằng đệ quy (muốn khử đệ quy thì dùng thêm stack hoặc queue)

Đệ quy với các trường hợp

- Cây rỗng
- Cây khác rỗng (leftMostChild và RightSibling): ít nhất sẽ có 1 nút

Với hàm đếm số nút - countNodes

- Cây rỗng --> số nút là 0
- Cây khác rỗng?

Ít nhất sẽ có 1 nút

Số nút sẽ bằng tổng số nút trên các cây con của nó (nếu có) + 1

Tìm tổng số nút trên các cây con? ==> gọi đệ quy

Con đầu tiên sẽ là LeftMostChild và con tiếp thieo sẽ là RightSibling của LeftMostChild

```
// hàm dem so nut trên cây
int countNodes(Node* root){
    if(root == NULL) return 0;
    // cay khac rong
    int sum = 1;
    // con đầu tiên
    Node* q = root->leftMostChild;
    while(q != NULL){
        sum = sum + countNodes(q);
        q = q->rightSibling;
    }
    return sum;
}
```

Giải phóng các nút trên cây

```
void freeTree(Node* r){
   if(r == NULL) return;
   Node* p = r->leftMostChild;
   while(p != NULL){
      Node* sp = p->rightSibling;
      freeTree(p);
      p = sp;
   }
   printf("free node %s\n",r->name); free(r);
   r = NULL;
}
```

Code:

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <queue>
using namespace std;

typedef struct Node{
   char name[256];
   struct Node* leftMostChild;
   struct Node* rightSibling;
}Node;

// cap phat bo nho de luu tru nut moi
// gan ten va gia tri leftmostchild, rightsibling la NULL
Node* makeNode(const char* name){
   Node* p = (Node*)malloc(sizeof(Node));
   strcpy(p->name,name);
   p->leftMostChild = NULL; p->rightSibling = NULL;
   return p;
}

// tim va tra ve nut co gia tri la keyname
// neu name khong co tren cay --> tra ve NULL
// tim de quy = DFS, thoi gian co O(n)
Node* find(Node* r, const char* keyname){
   if(r == NULL) return NULL;
   if(strcmp(r->name,keyname) == 0) return r;
   // neu khong bang --> tim tiep tai cay con cua nut hien tai
   Node* p = r->leftMostChild;
   while(p != NULL){
      Node* q = find(p,keyname);
      if(q != NULL) return q;
```

```
        p = p->rightSibling;
    }
}

// them keyname la anh chi em nho nhat cua nut hien tai
// se la anh chi em phai nhat trong cac nut con --> duyet toi con phai nhat hien tai
// sau do se add them vao cuoi
Node* addLast(Node* p, const char* keyname){
    // cap phat nut moi va gan vao
    if(p == NULL) return makeNode(keyname);

    // di chuyen toi anh chi em phai nhat hien tai
    p->rightSibling = addLast(p->rightSibling, keyname);
    return p;
}

// them keyname la anh chi em nho nhat cua nut hien tai khong de quy
Node* addLast_loop(Node* p, const char* keyname)
{
    Node *newChild = makeNode(keyname);
    while(p->rightSibling!=NULL) p = p->rightSibling;
    p->rightSibling = newChild;
}

// them nut child la con nho nhat cua parent
void addChild(Node* root, const char*parent, const char* child){
    Node* r = find(root,parent);
    if(r == NULL) return;
    r->leftMostChild = addLast(r->leftMostChild,child);
}

// ham them con cua nut hien tai khong de quy
void addChild_loop(Node* root, const char* parent, const char* child){
    Node* r = find(root,parent);
    if(r == NULL) return;
    // tao nut con moi
    Node *newChild = makeNode(child);
    // neu nut r chua co con
    if(r->leftMostChild == NULL)r->leftMostChild = newChild;
    else  // neu da co con thi them vao con nho nhat
    {
        Node *p = r->leftMostChild;
        while(p->rightSibling!=NULL) p = p->rightSibling;
        p->rightSibling = newChild;
    }
}

void printTree(Node* r){
```

```
   if(r == NULL) return;
   printf("%s: ",r->name);
   Node* p = r->leftMostChild;
   while(p != NULL){
      printf("%s ",p->name);
      p = p->rightSibling;
   }
   printf("\n");
   p = r->leftMostChild;
   while(p != NULL){
      printTree(p);
      p = p->rightSibling;
   }
}


// in file KHONG de quy, dung hang doi
void printTreeF_loop(Node* root, const char *filename){
   if(root == NULL) return;
   FILE* f = fopen(filename,"w");
   queue<Node*> Q;
   Q.push(root);

   while(!Q.empty())
   {
      Node* r = Q.front();
      Q.pop();

      fprintf(f,"%s ",r->name);
      Node* p = r->leftMostChild;
      while(p != NULL){
         fprintf(f,"%s ",p->name);
         if(p->leftMostChild!=NULL) Q.push(p);
         p = p->rightSibling;
      }
      fprintf(f,"$\n");
   }
   fprintf(f,"$$\n");
   fclose(f);
}

// in ra cac nut con cua nut co ten la keyname
void listChildren(Node* root, const char *keyname){
   Node* p = find(root,keyname);
   if(p == NULL) printf("Not Found %s\n",keyname);
   else{
      printf("Found %s with children: ",keyname);
      Node* q = p->leftMostChild;
```

```
      while(q != NULL){
         printf("%s ",q->name);     q = q->rightSibling;
      }
   }
   printf("\n");
}

// hàm tính chiều cao của nút p trên cây
// chieu cao nút lá tính là 0
int height(Node* p){
   if(p == NULL) return -1;
   int maxH = -1;
   Node* q = p->leftMostChild;
   while(q != NULL){
      int h = height(q);
      maxH = maxH < h ? h : maxH;
      q = q->rightSibling;
   }
   return maxH + 1;
}

// hàm dem so nut trên cây
int countNodes(Node* root){
   if(root == NULL) return 0;
   // cay khac rong
   int sum = 1;
   // con dầu tiên
   Node* q = root->leftMostChild;
   while(q != NULL){
      sum = sum + countNodes(q);
      q = q->rightSibling;
   }
   return sum;
}

void freeTree(Node* r){
   if(r == NULL) return;
   Node* p = r->leftMostChild;
   while(p != NULL){
      Node* sp = p->rightSibling;
      freeTree(p);
      p = sp;
   }
   printf("free node %s\n",r->name); free(r);
   r = NULL;
}
```

```
Node *doc_file (Node *r, const char filename[])
{
   char Child[256],Parent[256];
   FILE *f = fopen (filename,"r");
   fscanf (f,"%s",Parent);
   r=makeNode(Parent);
   while (strcmp(Parent,"$$")!=0)
   {
      fscanf (f,"%s",Child);
      if (strcmp(Child,"$")!=0)
      {
         addChild (r,Parent,Child);
      }
      else
      {
         fscanf (f,"%s",Parent);
      }
   }
   fclose (f);
   return r;
}

int main()
{
   Node *root = makeNode("Jame");
   addChild_loop(root, "Jame","John");
   addChild_loop(root, "Jame","Jim");
   addChild_loop(root, "Jame","Ann");
   addChild_loop(root, "Jame","Jack");
   addChild_loop(root, "Jim","Bill");
   addChild_loop(root, "Jim","Will");
   addChild_loop(root, "Jim","Rick");
   addChild_loop(root, "Jim","Habert");
   addChild_loop(root, "Jack","Elf");
   addChild_loop(root, "Jack","Nann");
   printTree(root);
   printTreeF_loop(root,"data.txt");
   listChildren(root,"Jack");

   Node *p = find(root,"Jame");
   printf("Chieu cao nut %s la %d\n",p->name, height(p));

   printf("So luong nut tren cay la %d\n",countNodes(root));

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

Bài tập 2. Tìm và in ra các nút có nhiều con nhất trên cây

VD. In ra số lượng nút con lớn nhất trên cây

Với cây trong ảnh thì số lượng nút con lớn nhất là 4

Cây rỗng --> -1

Cây là nút lá --> trả về 0

Cây khác nút lá --> số lượng con của nút hiện tại sẽ đếm từ leftMostChild

Số lượng nút lá lớn nhất sẽ bằng MAX của số lượng nút con trên nút hiện tại và các nút con của nó (gọi đệ quy)

```
// hàm tìm số lượng nút con lớn nhất của nút trên cây
int maxChildren(Node* root){
   if(root == NULL) return -1;
   int count = 0;
   int maxChild = 0;
   Node* q = root->leftMostChild;
   while(q != NULL){
      count ++;
      int countChild = maxChildren(q);
      maxChild = maxChild < countChild ? countChild : maxChild;
      q = q->rightSibling;
   }
   if(maxChild<count) return count;
   else return maxChild;
}

// tim va in ra nhan cua nut co max children
void findNodeMaxChildren(Node* root, int maxChildren)
{
   if(root == NULL) return;
   int count = 0;
   Node* q = root->leftMostChild;
   while(q != NULL){
      count ++;
      findNodeMaxChildren(q, maxChildren);
      q = q->rightSibling;
   }
   if(maxChildren==count) printf("Node have max children %s\n",root->name);
}
```

Tối ưu code chỉ cần duyệt 1 lần?

Dùng hàng đợi

Bài tập 3. Tìm và in ra các nút lá trên cây

Bài tập 4. Tìm và in ra nút lá ở mức nông nhất/ sâu nhất trên cây

Bài tập 5. Đếm số lượng nút trong trên cây (nút có ít nhất 1 con)