Nội dung

* Chạy lại code cây nhị phân tìm kiếm trong slide
* Bài tập ứng dụng

Cây nhị phân tìm kiếm được sử dụng trong các trường hợp việc thêm/xóa và tìm kiếm diễn ra liên tục

* Nếu KHÔNG thêm/xóa (hoặc diễn ra ít) --> nên dùng mảng hoặc danh sách liên kết đơn
* Nếu thêm/xóa diễn ra nhiều --> dùng cây nhị phân

Code mẫu

|  |
| --- |
| /\*Minh hoa dung cay nhi phan tim kiem de quan ly sinh vien\*/  #include <stdio.h>  #include <stdlib.h>  #include <string.h>  #include <ctype.h>  #define MAX\_L 256  #define MAX 100000  //Dinh nghia cau truc sinh vien don gian  typedef struct Node{  char name[MAX\_L];  char email[MAX\_L];  struct Node\* leftChild;  struct Node\* rightChild;  }Node;    // Ham cap phat dong thong tin 1 sinh vien moi  Node\* makeNode(char\* name, char\* email){  Node\* p = (Node\*)malloc(sizeof(Node));  strcpy(p->name,name); strcpy(p->email,email);  p->leftChild = NULL; p->rightChild = NULL;  return p;  }    // Ham them thong tin sinh vien moi vao cay  // Cay ban dau co the rong  // Cach 1. Tra ve qua ten ham  Node\* insert(Node\* r, char\* name, char\* email){  if(r == NULL) return makeNode(name,email); // cay ban dau rong  int c = strcmp(r->name,name);  if(c == 0){  printf("Student %s exists, do not insert\n",name); return r;  }else if(c < 0){  r->rightChild = insert(r->rightChild,name,email); return r;  }else{  r->leftChild = insert(r->leftChild,name,email); return r;  }  }    // Cach 2. tra ve qua tham so  // cay ban dau co the rong  void insert2(Node\*\* r, char\* name, char\* email){  if(\*r == NULL) \*r = makeNode(name,email); // cay ban dau rong  int c = strcmp((\*r)->name,name);  if(c == 0){  printf("Student %s exists, do not insert\n",name);  }else if(c < 0){  insert2(&((\*r)->rightChild),name,email);  }else{  insert2(&((\*r)->leftChild),name,email);  }  }  / \*Hàm insert2 truyền vào một con trỏ đến một con trỏ kiểu Node (Node\*\*) để có thể thay đổi giá trị của con trỏ được truyền vào (nếu cần thiết) và tạo ra một nút mới (nếu cần thiết) cho trường hợp cây rỗng. Trong khi đó, hàm insert3 truyền vào một con trỏ trực tiếp kiểu Node (Node\*) vì nó chỉ cần truy cập đến nút được truyền vào để tạo một nút mới (nếu cần thiết) và chèn nút mới vào cây. \*/  // Cach 3. Them nut moi vao cay, va cay chac chan khac rong!  void insert3(Node\* r, char\* name, char\* email){  int c = strcmp(r->name,name);  if(c == 0){  printf("Student %s exists, do not insert\n",name);  }else if(c < 0){  if(r->rightChild==NULL) r->rightChild = makeNode(name,email);  else insert3(r->rightChild,name,email);  }else{  if(r->leftChild==NULL) r->leftChild = makeNode(name,email);  else insert3(r->leftChild,name,email);  }  }    // Tim kiem va tra ve nut tren cay  Node\* find(Node\* r, char\* name){  if(r == NULL) return NULL;  int c = strcmp(r->name,name);  if(c == 0) return r;  if(c < 0) return find(r->rightChild,name);  return find(r->leftChild,name);  }    // tim nut trai nhat  Node\* findMin(Node\* r){  if(r == NULL) return NULL;  Node\* lmin = findMin(r->leftChild);  if(lmin != NULL) return lmin;  return r;  }    // ham xoa nut  // Co the viet lai bang cach tra ve qua tham so  Node\* removeStudent(Node\* r, char\* name){  if(r == NULL) return NULL;  int c = strcmp(r->name,name);  if(c > 0) r->leftChild = removeStudent(r->leftChild,name);  else if(c < 0) r->rightChild = removeStudent(r->rightChild,name);  else{  if(r->leftChild != NULL && r->rightChild != NULL){  Node\* tmp = findMin(r->rightChild);  strcpy(r->name,tmp->name); strcpy(r->email,tmp->email);  r->rightChild = removeStudent(r->rightChild,tmp->name);  }else{  Node\* tmp = r;  if(r->leftChild == NULL) r = r->rightChild; else r = r->leftChild;  free(tmp);  }  }  return r;  }      //Ham xoa tat ca cac nut tren cay  void freeTree(Node\* r){  if(r == NULL) return;  freeTree(r->leftChild);  freeTree(r->rightChild);  free(r);  }    // Ham doc du lieu tu file  Node\* load(char\* filename){  Node\* root = NULL;  FILE\* f = fopen(filename,"r");  if(f == NULL) printf("Load data -> file not found\n");  root = NULL;  while(!feof(f)){  char name[256], email[256];  fscanf(f,"%s%s",name, email); // Khong doc duoc ten co cach trong  root = insert(root,name,email);  }  fclose(f);  return root;  }  // Doc du lieu tu file co cach trong  // Thong tin ho ten, email tren cac dong rieng  Node\* load2(char\* filename){  FILE\* f = fopen(filename,"r");  if(f == NULL) printf("Load data -> file not found\n");  Node\* root = NULL;  while(!feof(f)){  char name[256], email[256];  //fscanf(f,"%s%s",name, email); // Khong doc duoc ten co cach trong  fgets(name,256,f);  // Xu ly dau xuong dong  if(!isalpha(name[strlen(name)-1]))name[strlen(name)-1]='\0';    fgets(email,256,f);  // Xu ly dau xuong dong  if(!isalpha(email[strlen(email)-1]))email[strlen(email)-1]='\0';    root = insert(root,name,email);  }  fclose(f);  return root;  }    Node\* loadFromFile()  {  char filename[256];  fflush(stdin);  printf("Ten file: ");  gets(filename);  Node\* root = load2(filename);  return root;  }    // duyet theo thu tu giua  void inOrder(Node\* r){  if(r == NULL) return;  inOrder(r->leftChild);  printf("%s, %s\n",r->name,r->email);  inOrder(r->rightChild);  }    // ghi ra file tu thu tu duyet giua  void inOrderF(Node\* r, FILE\* f){  if(r == NULL) return;  inOrderF(r->leftChild,f);  fprintf(f,"%s %s\n",r->name,r->email);  inOrderF(r->rightChild,f);  }  // moi thon tin tren 1 dong  void inOrderF2(Node\* r, FILE\* f){  if(r == NULL) return;  inOrderF2(r->leftChild,f);  fprintf(f,"%s\n%s\n",r->name,r->email);  inOrderF2(r->rightChild,f);  }    void printList(Node\* root){  inOrder(root);  printf("\n");  }  //ghi cay vao file  void WriteToFile(Node\* root){  char filename[256];  fflush(stdin);  printf("Ten file output: ");  scanf("%s",filename);  FILE\* f = fopen(filename,"w");  inOrderF2(root,f);  fclose(f);  }    //Them thong tin vao tu ban phim  Node\* processInsert(Node\* root){  char name[256], email[256];  fflush(stdin);  printf("Ten: ");  gets(name);  fflush(stdin);  printf("Email: ");  gets(email);  return insert(root,name,email);  }    //Xoa phan tu khoi cay  Node\* processRemove(Node\* root){  char name[256];  fflush(stdin);  printf("Ten can xoa: ");  gets(name);  return removeStudent(root,name);  }    //Tim kiem  void processFind(Node\* root){  char name[256];  fflush(stdin);  printf("Ten can tim: ");  gets(name);  Node \* ret = find(root,name);  if(ret!=NULL) printf("Sinh vien co trong danh sach!\n");  else printf("Khong ton tai ten sinh vien trong danh sach!\n");  }  int main(){  Node\* root = NULL;  while(1){  printf("Enter command: ");  char cmd[256];  scanf("%s",cmd);  if(strcmp(cmd,"Quit")==0) break;  else if(strcmp(cmd,"Load")==0) root = loadFromFile();  else if(strcmp(cmd,"Print")==0) printList(root);  else if(strcmp(cmd,"Find")==0) processFind(root);  else if(strcmp(cmd,"Insert")==0) root = processInsert(root);  else if(strcmp(cmd,"Remove")==0) root = processRemove(root);  else if(strcmp(cmd,"Store")==0) WriteToFile(root);    }  freeTree(root);    return 0;  } |

**Bài tập 1**. Thông tin đầu sách trong thư viện cần quản lý gồm

* Tên sách
* Loại sách
* Tác giả
* Số lượng sách

Thông tin được lưu trữ trong file dạng

|  |
| --- |
| Lap trinh C co ban  CNTT  Nguyen van A, Le thi B  12  Ngon ngu lap trinh C  CNTT  Le the Tanh  6  Co khi dai cuong  Tran Van E  Co khi  7  Hoa hoc can ban  Hoa hoc  Le van F  1  Tin hoc van phong  CNTT  Dao Hong Thai  6  \*\*\* |

Kết thúc file là \*\*\*

Hãy xây dựng cây nhị phân tìm kiếm để

* Quản lý thông tin sách có trong thư viện
* Tra cứu sách thông qua tên sách
* Mượn và trả sách thông qua tên sách
* Đọc/Ghi thông tin sách từ file
* Liệt kê danh sách các sách đang hết (số lượng =0)
* Liệt kê các sách trong 1 lĩnh vực
* Bổ sung thêm sách mới
* Xóa thông tin sách cũ