**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN – ĐHQG.TPHCM**

**KHOA CÔNG NGHỆ THÔNG TIN**

Logo

Description automatically generated

**BÁO CÁO BÀI TẬP VỀ NHÀ TUẦN 4**

**BINARY SEARCH TREE**

| Giảng viên thực hành |

Lê Tuấn Thu

| Nhóm thực hiện |

Nguyễn Huy Hoàng – 21120458

Nguyễn Ngọc Như Huyền - 21120475

Thành phố Hồ Chí Minh – 2022

**QUẢN LÝ DANH SÁCH KHÁCH HÀNG VỚI CÂY NHỊ PHÂN TÌM KIẾM**

* **NHÓM HÀM DATA**
* Là nơi chứa những hằng số, cấu trúc dữ liệu do người lập trình định nghĩa và dùng xuyên suốt chương trình.
* **Khai báo thư viện**

|  |  |
| --- | --- |
|  | #include <iostream> |
|  | #include <fstream> |
|  | #include <string> |
|  | #include <ctime> |
|  | #pragma warning(disable : 4996) |
|  | using namespace std; |

* **Thư viện fstream:** sử dụng để đọc/ghi file.
* **Thư viện string:** sử dụng kiểu dữ liệu string và dùng hàm getline().
* **Thư viện ctime:** sử dụng để tính ngày hiện tại.
* **#pragma warning(disable : 1996):** khắc phục lỗi cảnh báo compiler khi sử dụng ctime.
* **Khai báo các cấu trúc**

|  |  |
| --- | --- |
|  | // Khai bao cau truc ngay sinh |
|  | struct NgaySinh { |
|  | int ngay, thang, nam; |
|  | }; |
|  | typedef NgaySinh NgayHeThong; |
|  |  |
|  | // Khai bao cau truc khach hang |
|  | struct KhachHang { |
|  | string name; |
|  | string sdt; |
|  | string diachi; |
|  | NgaySinh ngsinh; |
|  | }; |
|  |  |
|  | // Cau truc cay |
|  | struct Node { |
|  | KhachHang key; |
|  | Node\* left, \* right; |
|  | }; |
|  | typedef Node\* Tree; |

* **Struct NgaySinh:** lưu thông tin ngày tháng năm sinh của khách hàng.
* **Struct KhachHang:** lưu các thông tin cần có của khách hàng (tên, số điện thoại, địa chỉ, ngày sinh).
* **Struct Node:** lưu thông tin của một node (gồm thông tin khách hàng-khóa, địa chỉ node trái và node phải).
* **HÀM KHỞI TẠO**

|  |  |
| --- | --- |
|  | Node\* createNode(KhachHang key) { |
|  | Node\* node = new Node; |
|  | if (node != NULL) { |
|  | node->key = key; |
|  | node->left = node->right = NULL; |
|  | } |
|  | return node; |
|  | } |

* **createNode:** Khởi tạo node với các bước:

1. Khởi tạo vùng nhớ cho node.
2. Khởi tạo thành công thì cập nhật thông tin cho node (dữ liệu node-thông tin khách hàng, địa chỉ con trỏ trái và phải gán bằng NULL).
3. Hàm trả về địa chỉ của node.

* **CÁC HÀM NHẬP, XUẤT VÀ IN CÂY**
* **Hàm nhập thông tin khách hàng**

**Text

Description automatically generated**

* **Xuất danh sách khách hàng**

**Text

Description automatically generated**

* **In cây**

**Text

Description automatically generated**

* **CÁC HÀM ĐỌC THÔNG TIN KHÁCH HÀNG TỪ FILE**

|  |  |
| --- | --- |
|  | /\*Ham doc ngay sinh\*/ |
|  | void DocNgsinh(ifstream& filein, NgaySinh& ngsinh) { |
|  | filein >> ngsinh.ngay; |
|  | filein.ignore(1); |
|  | filein >> ngsinh.thang; |
|  | filein.ignore(1); |
|  | filein >> ngsinh.nam; |
|  | filein.ignore(1); |
|  | } |
|  | /\*Ham doc thong tin mot khach hang\*/ |
|  | void DocKhachHang(ifstream& filein, KhachHang& key) { |
|  | getline(filein, key.name, ','); |
|  | DocNgsinh(filein, key.ngsinh); |
|  | getline(filein, key.sdt, ','); |
|  | getline(filein, key.diachi); |
|  | } |
|  | /\*Ham doc danh ba\*/ |
|  | void DocDanhBa(ifstream& filein, Tree& root) { |
|  | KhachHang key; |
|  | while (filein.peek() != EOF) { |
|  | DocKhachHang(filein, key); |
|  | Node\* node = createNode(key); |
|  | insertNode(root, node); |
|  | } |
|  | } |

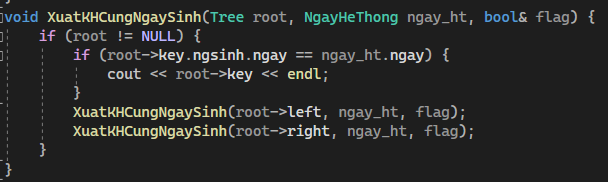
* **docNgSinh():** Hàm đọc ngày tháng năm sinh của khách hàng, được gọi lại trong hàm **docKhachHang()** để đọc thông tin đầy đủ của một khách hàng.
* **getline():** Đọc hết dòng đến dấu , thì ngừng.
* **ignore():** bỏ qua dấu ,
* **docDanhBa():** Hàm đọc danh bạ khách hàng từ file text và lưu vào cây nhị phân.
* **peek():** Đọc ký tự (không lưu ký tự) đến cuối file thì ngừng.
* **CÁC HÀM THÊM VÀ XÓA**

|  |  |
| --- | --- |
|  | //Them |
|  | void insertNode(Tree& root, Node\* node) { |
|  | if (root == NULL) |
|  | root = node; |
|  | else if (node->key.sdt > root->key.sdt) |
|  | insertNode(root->right, node); |
|  | else if (node->key.sdt < root->key.sdt) |
|  | insertNode(root->left, node); |
|  | else return; |
|  | } |

* **Các bước thêm:**
* Trường hợp cây rỗng thì nút thêm vào là nút gốc.
* Nếu nút thêm vào có khóa lớn hơn khóa của nút gốc thì gọi đệ quy thêm vào cây con bên phải.
* Nếu nút thêm vào có khóa nhỏ hơn khóa của nút gốc thì gọi đệ quy thêm vào cây con bên trái.
* Nếu hai nút có khóa bằng nhau thì kết thúc.

|  |  |
| --- | --- |
|  | /\*Ham tim node nho nhat cua cay con phai\*/ |
|  | void minNode(Node\* rightChild, Node\*& minnode) {//rightChild: cay con phai |
|  | if (rightChild->left == NULL) |
|  | minnode = rightChild; |
|  | else |
|  | minNode(rightChild->left, minnode); |
|  | } |
|  | /\*Ham xoa 1 khach hang co sdt cho truoc\*/ |
|  | void XoaKhachHang(Tree& root, string sdt) { |
|  | if (root == NULL) { |
|  | cout << "Khong tim thay so dien thoai can xoa" << endl; |
|  | return; |
|  | } |
|  | else if (root->key.sdt > sdt) |
|  | XoaKhachHang(root->left, sdt); |
|  | else if (root->key.sdt < sdt) |
|  | XoaKhachHang(root->right, sdt); |
|  | else { |
|  | if (root->left == NULL && root->right == NULL) |
|  | root = NULL; |
|  | else if (root->left == NULL) { |
|  | Node\* delnode = root; |
|  | root = root->right; |
|  | delete delnode; |
|  | } |
|  | else if (root->right == NULL) { |
|  | Node\* delnode = root; |
|  | root = root->left; |
|  | delete delnode; |
|  | } |
|  | else { |
|  | Node\* minnode = NULL; |
|  | minNode(root->right, minnode); |
|  |  |
|  | Node\* copy = createNode(minnode->key); |
|  | XoaKhachHang(root, minnode->key.sdt); |
|  | copy->left = root->left; |
|  | copy->right = root->right; |
|  |  |
|  | delete root; |
|  | root = copy; |
|  | } |
|  | } |
|  | } |

* Các bước xóa
* Trường hợp cây rỗng hoặc các nút đã được duyệt hết thì in kết quả không tìm thấy số điện thoại và kết thúc hàm.
* Nếu nút cần xóa có khóa nhỏ hơn khóa của nút gốc thì duyệt cây con bên trái, lớn hơn thì duyệt cây con bên phải.
* Trường hợp tìm thấy nút cần xóa:
* Nếu nút cần xóa là nút lá thì gán nút lá bằng NULL.
* Nếu nút cần xóa có một con thì cập nhật lại địa chỉ của nút là địa chỉ của nút con và xóa nút cần xóa.
* Nếu nút cần xóa có hai con:
* Tìm nút có khóa nhỏ nhất của cây con phải của nút cần xóa.
* Tạo nút mới có khóa bằng với khóa của nút nhỏ nhất vừa tìm được.
* Gọi đệ quy hàm xóa và xóa nút nhỏ nhất khỏi cây.
* Cập nhật địa chỉ cho cây con trái và cây con phải của nút mới tạo là địa chỉ của cây con trái và cây con phải của nút cần xóa.
* Xóa nút cần xóa.
* Cập nhật lại địa chỉ nút cần xóa là địa chỉ nút mới tạo.
* **HÀM TÌM NHỮNG KHÁCH HÀNG CÓ CÙNG NGÀY SINH Ở NGÀY HIỆN TẠI**

****

**Text

Description automatically generated**

* Dùng hàm hỗ trợ trong C++ ở thư viện c\_time để tìm ngày tháng năm hiện tại. Duyệt cây và so sánh ngày sinh của khách hàng với ngày hiện tại. Nếu trùng nhau thì in ra danh sách những khách hàng có ngày sinh trong ngày hiện tại, ngược lại thông báo kết quả không tìm thấy.

* **HÀM MENU ĐỂ CHẠY CHƯƠNG TRÌNH**
* Hàm menu chính được gọi trong hàm main để chạy chương trình

Text

Description automatically generated