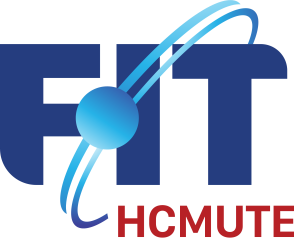
**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TPHCM**

**KHOA CÔNG NGHỆ THÔNG TIN**

**MÔN CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT**



## NGUYỄN THÀNH LỢI – 21110234

*Đề Tài :*

**XÁC THỰC GIAO DỊCH VỚI MERKLE TREE**

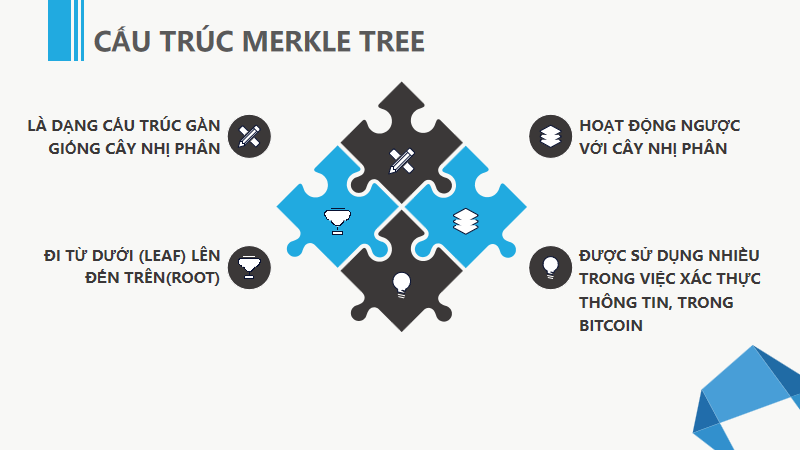
**BÁO CÁO ĐỒ ÁN CUỐI KỲ**

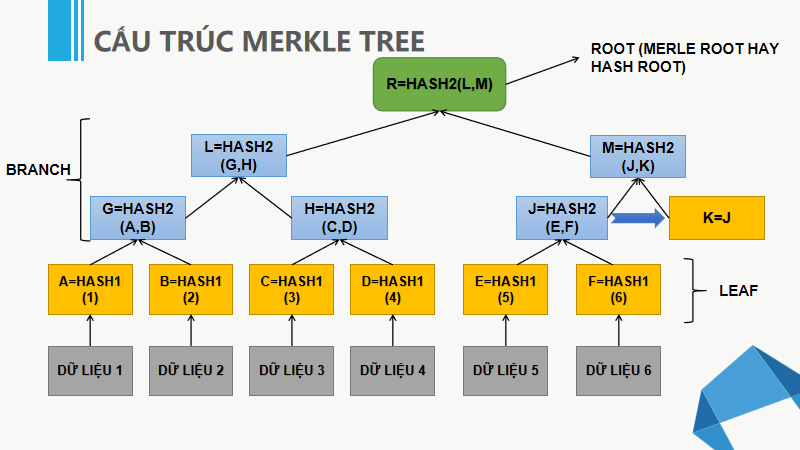
## GIÁO VIÊN HƯỚNG DẪN:

**Ths.NGUYỄN THỦY AN**

**HỌC KỲ 1, NH: 2022-2023**

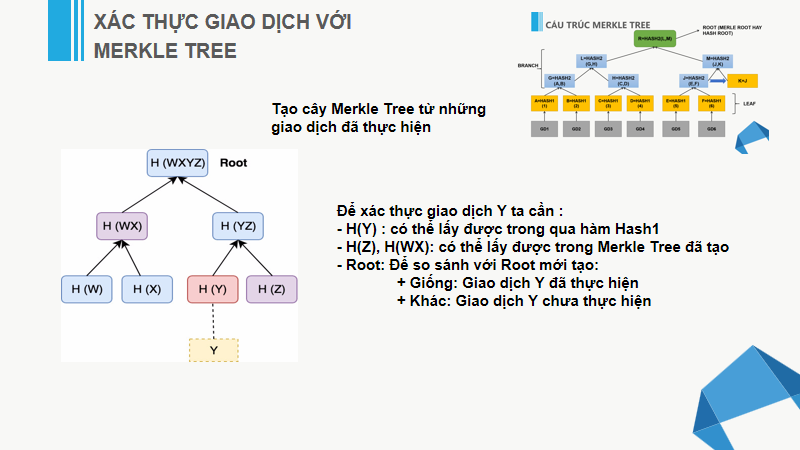
1. **PHẦN LÝ THUYẾT**
2. **CẤU TRÚC MERKLE TREE**





1. **XÁC THỰC GIAO DỊCH**





1. **CÁC CHỨC NĂNG HỖ TRỢ KHÁC**
2. **Tìm kiếm ID**

Tìm kiếm ID trên ListGD theo với Thuật toán tìm kiếm nhị phân, dựa trên MaGD của từng giao dịch, vì các giao dịch đều được sắp theo thứ tự tăng dần MaGD.

1. **Xuất tất cả các giao dịch**

Chỉ việc xuất tất cả các giao dịch ra từ đầu đến cuối danh sách.

1. **PHẦN TRÌNH BÀY CODE**
2. **CÁC KIỂU DỮ LIỆU ĐỊNH NGHĨA**
3. C++

struct Node {

string Hash;

Node\* left;

Node\* right;

};

typedef Node\* MTree;

struct ID { //Luu cac Hash cần để mã hóa 1 giao dịch

string lHash[50];

int n;

};

struct GiaoDich {

string MaGD;

string TenKhach;

// string NhanVienPV[50];

// string NoiDung[100];

long SoTien;

ID HashXD;

};

struct ListGD {

GiaoDich A[100];

int n;

};

1. C#

public class ID

{ //Luu cac Hash cần để mã hóa 1 giao dịch

public string[] lHash { get; set; }

public int n { get; set; }

public ID()

{

this.lHash=new string[100];

}

}

public class GiaoDich

{

public string MaGD { get; set; }

public string TenKhach { get; set; }

// string NhanVienPV[50];

// string NoiDung[100];

public long SoTien { get; set; }

public ID HashXD { get; set; }

public GiaoDich()

{

this.HashXD = new ID();

}

public void CapNhat(long t)

{

this.SoTien = t;

}

}

public class ListGD

{

public GiaoDich[] A { get; set; }

public int n { get; set; }

public ListGD ()

{

this.A = new GiaoDich[100];

for (int i = 0; i < 100; i++)

{

this.A[i] = new GiaoDich();

}

this.n = 0;

}

public ListGD(int n)

{

this.A = new GiaoDich[100];

for (int i = 0; i < 100; i++) {

this.A[i] = new GiaoDich();

}

this.n = n;

}

}

1. **HÀM BĂM**
2. C++

//Hash

void Hash1(string& s, GiaoDich x) // giao dịch -> Hash(giao dịch)

{

s= x.MaGD + x.TenKhach;

for (int i = 0; i < s.length(); i++)

{

s[i] = (26 \* s[i] + x.SoTien) % 96 + 32;

if (s[i] == '\_') // '\_' để ngăn cắt các hash khác nhau trong ID

s[i]++;

}

}

void Hash2(string s1, string s2, string& kq) // kq=Hash(s1,s2)

{

int n = max(s1.length(),s2.length());

int i = s1.length();

while (i < n)

{

s1.push\_back( ('0' + i) % 96 + 32);

i++;

}

i = s2.length();

while (i < n)

{

s2.push\_back(('0' + i) % 96 + 32);

i++;

}

kq = s1;

kq.push\_back((s1[s1.length()-1] + s2[s2.length()-1]) % 96 + 32);

for (int i = 0; i < n; i++)

{

kq[i] = (28 \* (kq[i]+s2[i]) + 97) % 96 + 32;

if (kq[i] == '\_')

kq[i] ++;

}

}

1. C#

static class Hash

{

public static string Hash1(GiaoDich x)

{

string s = String.Concat(x.MaGD, x.TenKhach);

char[] s1 = s.ToCharArray();

for (int i = 0; i < s1.Length; i++)

{

long t = (26 \* s1[i] + x.SoTien) % 96 + 32;

s1[i] = (char)t ;

if (s1[i] == '\_') // '\_' để ngăn cắt các hash khác nhau trong ID

s1[i]++;

}

string l= s1[0].ToString();

for(int i=1; i<s1.Length; i++)

{

l= l + s1[i].ToString();

}

return l;

}

public static string Hash2(string s1, string s2)

{

int n = Math.Max(s1.Length, s2.Length);

int i = s1.Length;

while (i < n)

{

s1=s1+((char)(('0' + i) % 96 + 32));

i++;

}

char[] kq1 = new char[100];

kq1 = s1.ToCharArray();

i = s2.Length;

while (i < n)

{

s2 = s2 + ((char)(('0' + i) % 96 + 32));

i++;

}

char[] kq2 = s2.ToCharArray();

char[] kq = kq1;

char t= (char)((kq1[kq1.Length - 1] + kq2[kq2.Length - 1]) % 96 + 32);

for (i = 0; i < n; i++)

{

kq[i] = (char)((28 \* (kq[i] + kq2[i]) + 97) % 96 + 32);

if (kq[i] == '\_')

kq[i]++;

}

string l = kq[0].ToString();

for (int j = 1; j < kq.Length; j++)

{

l += kq[j].ToString();

}

return l + t.ToString();

}

}

1. **DỮ LIỆU LƯU TRỮ**

Dữ liệu có thể lưu trữ trong Excel, Word,....Nhưng để không mất thời gian cho việc nhập dữ liệu vào lưu trữ, em đã sử dụng Tạo dữ liệu thay cho việc Đọc và ghi File. Và sẽ được trình bày lần lượt ở phần IV và V

1. **TRÌNH BÀY CODE TRÊN NGÔN NGỮ C++**
2. **Hàm Tạo dữ liệu**

//Tạo dữ liệu

void TaoDuLieu(ListGD& l)

{

l.n = 6;

string s1="0000";

string s2 = "Nguyen Van A";

long tien = 200000;

for (int i = 0; i < l.n; i++)

{

s1[3]++;

if (s1[3] > '9')

{

char t = s1[3]-'0';

s1[3] = t % 10+'0';

s1[2] += t / 10;

if (s1[2] > '9')

{

char t1 = s1[2] - '0';

s1[2] = t1 % 10 + '0';

s1[1] += t1 / 10;

if (s1[1] > '9')

{

char t2 = s1[1] - '0';

s1[1] = t2 % 10 + '0';

s1[0] += t2 / 10;

}

}

}

l.A[i].MaGD= s1;

s2[s2.length() - 1]++;

l.A[i].TenKhach= s2;

l.A[i].SoTien = 50000 + tien \* i;

l.A[i].HashXD.n = 0;

}

}

1. **Các Hàm Tạo tạo nên cây MerkleTree**

// Tạo Root

Node\* TaoRoot(Node\* LLeaf[], int n)

{

if (n == 1) return LLeaf[0];

else

{

Node\* LBranch[100];

int m = 0;

if (n % 2 == 1)

{

LLeaf[n] = new Node();

LLeaf[n]->Hash= LLeaf[n - 1]->Hash;

LLeaf[n]->left = NULL;

LLeaf[n]->right = NULL;

n++;

}

for (int i = 0; i < n; i = i + 2)

{

LBranch[m++] = TaoBranch(LLeaf[i], LLeaf[i + 1]);

}

return TaoRoot(LBranch, m);

}

}

// Tạo Branch

Node\* TaoBranch(Node\* x, Node\* y)

{

Node\* n = new Node();

n->left = x;

n->right = y;

Hash2(x->Hash, y->Hash, n->Hash);

return n;

}

//Tạo leaf

Node\* TaoLeaf(GiaoDich x)

{

Node\* n = new Node();

Hash1(n->Hash, x);

n->left = NULL;

n->right = NULL;

return n;

}

1. **Hàm liên quan đến ID**

//Tạo ID

void TaoID(ListGD& l, int start, int end, MTree T)

{

if (T->right->right == NULL && T->left->left != NULL)

{

for (int i = start; i <= end; i++)

{

int k = l.A[i].HashXD.n;

l.A[i].HashXD.lHash[k].append(T->right->Hash);

l.A[i].HashXD.n++;

}

TaoID(l, start, end, T->left);

}

else

if (T->left->left== NULL && T->right->right == NULL) {

l.A[start].HashXD.lHash[l.A[start].HashXD.n++].append(T->right->Hash);

l.A[end].HashXD.lHash[l.A[end].HashXD.n++].append(T->left->Hash);

return;

}

else

{

int m=(end-start+1)/2;

if (m % 2 == 1) m++;

m += start;

for (int i = start; i < m; i++)

{

int k = l.A[i].HashXD.n;

l.A[i].HashXD.lHash[k].append(T->right->Hash);

l.A[i].HashXD.n++;

}

TaoID(l, m, end, T->right);

for (int i = m; i <= end; i++)

{

int k = l.A[i].HashXD.n;

l.A[i].HashXD.lHash[k].append(T->left->Hash);

l.A[i].HashXD.n++;

}

TaoID(l, start, m-1, T->left);

}

}

// Vì Level càng thấp thì số ký tự trong Hash của node đó càng nhiều, nên để thuận tiện cho việc việc xác thực nên cần thêm 1 số hàm hỗ trợ khác nữa :

//sắp xếp các Hash theo thứ tự sử dụng

void sapxepID(ListGD& l)

{

for (int i = 0; i <= l.n; i++)

sapxep(l.A[i].HashXD);

}

void sapxep(ID& l)

{//sắp xếp theo chiều dài tăng dần;sử dụng thuật toán sắp xếp nổi bọt

for (int i =0; i<l.n-1; i++)

{

for (int j = l.n-1; j > i; j--)

if (l.lHash[j].length() < l.lHash[j - 1].length())

hoanvi(l.lHash[j], l.lHash[j - 1]);

}

}

void hoanvi(string& a, string& b)

{

string t = a;

a = b;

b = t;

}

1. **Hàm Nhập, xuất giao dịch**

//Nhập giao dịch

void NhapGD(GiaoDich& x)

{

cout << "Nhap Ma giao dich: ";

cin >> x.MaGD;

cout << "Nhap Ten khach: ";

cin.ignore();

getline(std::cin, x.TenKhach);

cout << "Nhap So tien giao dich: ";

cin >> x.SoTien;

cin.ignore();

string lHash;

cout << "Nhap Hash XD: ";

getline(std::cin, lHash);

// cout << lHash;

int vt1 = 0;

x.HashXD.n = 0;

for (int vt2 = 0; vt2 < lHash.length(); vt2++)

{

if (lHash[vt2] == '\_')

{

int k = x.HashXD.n;

x.HashXD.lHash[k].append(lHash.substr(vt1, vt2-vt1));

x.HashXD.n++;

vt1 = vt2 + 1;

}

}

}

//Xuất giao dịch

void XuatGD(GiaoDich x)

{

cout << endl;

cout << "Ma giao dich: " << x.MaGD << endl;

cout << "Ten khach hang: " << x.TenKhach << endl;

cout << "So tien giao dich: " << x.SoTien << " VND" << endl;

cout << "ID: ";

for (int j = 0; j < x.HashXD.n; j++)

{

cout << x.HashXD.lHash[j] << "\_";

}

cout << endl;

}

1. **Chức năng Tìm kiếm ID**

ID TimKiemID(ListGD l, int start, int end, int iMaGD)

{

if (start > end)

{

ID noExist;

noExist.n = -1;

return noExist;

}

int i = start+(end-start) / 2;

int y = strMaGDToIn(l.A[i].MaGD);

if (iMaGD == y)

return l.A[i].HashXD;

else if (iMaGD < y)

return TimKiemID(l, start, i - 1, iMaGD);

else

return TimKiemID(l, i+1, end, iMaGD);

}

int strMaGDToIn(string MaGD)

{

int tong = 0;

int n = MaGD.length();

for (int i = 0; i < n; i++)

{

int x = (MaGD[i] - '0');

if (x >= 10 || x < 0)

return -1;

tong += x \* pow(10, n - 1 - i);

}

return tong;

}

1. **Chức năng Xuất tất cả giao dịch**

//Xuất ra tất cả giao dịch đã thực hiện; // l: ListGD

for (int i = 0; i < l.n; i++)

XuatGD(l.A[i]);

1. **Chức năng xác thực giao dịch**

//Xác thực giao dịch

bool XacThuc(GiaoDich x, string HashRoot)

{

string kq;

Hash1(kq, x);

for (int i = 0; i < x.HashXD.n; i++)

Hash2(kq, x.HashXD.lHash[i], kq);

//cout << kq;

return kq == HashRoot;

}

1. **Menu**

//Menu Chon va xuat ket qua

void Menu(ListGD l, MTree T)

{

int bc = 0; //trở về bảng chon bc=0, không thì bc=1;

int c;

while (bc == 0)

{

cout << "Chon chuc nang: \n";

cout << "1.Xac thuc giao dich\n";

cout << "2.Tim kiem ID\n";

cout << "3.Xuat tat ca giao dich da thuc hien\n";

cout << "=>Chon: ";

cin >> c;

bool t = true;

if (c == 3)

{

//Xuất ra tất cả giao dịch đã thực hiện

for (int i = 0; i < l.n; i++)

XuatGD(l.A[i]);

}

else if (c == 1)

{

//Xác thực giao dịch đã được nhập vào

GiaoDich x;

do

{

NhapGD(x);

/\*cout << x.MaGD << " " << x.TenKhach << " " << x.SoTien << endl;

for (int i = 0; i < x.HashXD.n; i++)

{

cout << x.HashXD.lHash[i] << endl;

}\*/

if (XacThuc(x, T->Hash) == true)

cout << "Giao dich da duoc thuc hien !!\n";

else

{

cout << "Giao dich chua duoc thuc hien! Hay kiem tra lai thuong tin!\n";

int e;

cout << "1. Nhap lai.\n";

cout << "2.Thoat xac thuc.\n";

cout << "=>Chon: ";

cin >> e;

if (e == 1)

t = false;

else

t = true;

}

} while (t == false);

}

else if (c == 2)

{

//Tìm kiếm ID

do

{

string sMaGD;

cout << "\nNhap Ma giao dich: ";

cin >> sMaGD;

int iMaGD = strMaGDToIn(sMaGD);

ID x = TimKiemID(l, 0, l.n - 1, iMaGD);

if (x.n > 0)

{

cout << "ID: ";

for (int j = 0; j < x.n; j++)

{

cout << x.lHash[j] << "\_";

}

cout << endl;

}

else

{

cout << "\nNhap Ma giao dich sai!\n";

int e;

cout << "1. Nhap lai.\n";

cout << "2.Thoat Tim kiem ID.\n";

cout << "=>Chon: ";

cin >> e;

if (e == 1)

t = false;

else

t = true;

}

} while (t == false);

}

else

cout << "LOI!!!" << endl << endl;

cout << "\n0.Tro ve bang chon.\n";

cout << "1.Thoat toan bo.\n";

cout << "=>Chon: ";

cin >> bc;

cout << endl;

if (bc != 1 && bc != 0)

{

cout << "LOI!\n";

break;

}

}

}

1. **int main()**

int main()

{

ListGD l;

TaoDuLieu(l); // có thể sử dụng đọc file thay cho hàm này

//Chuyển hóa dữ liệu thành các Hash và lưu trữ trên Merkle Tree

Node\* LLeaf[100];

for (int i = 0; i < l.n; i++)

LLeaf[i] = TaoLeaf(l.A[i]);

MTree T = TaoRoot(LLeaf, l.n);

//Tạo ID cho các dịch giao dịch

TaoID(l, 0,l.n-1,T);

sapxepID(l);

Menu(l, T);

return 0;

}

1. **Thư viện sử dụng**

#include<iostream>

#include<string>

#include<math.h>

1. **TRÌNH BÀY CODE TRÊN NGÔN NGỮ C#**
2. **Class XuLyGD**
3. *Hàm Tạo dữ liệu*

//Tạo dữ liệu

static public void TaoDuLieu(ref ListGD l)

{

l.n = 6;

string s1 = "0000";

char[] a = s1.ToCharArray();

string s2 = "Nguyen Van A";

char[] b = s2.ToCharArray();

long tien = 200000;

for (int i = 0; i < l.n; i++)

{

a[3]++;

if (a[3] > '9')

{

char t = (char)(a[3] - '0');

a[3] = (char)(t % 10 + '0');

a[2] += (char)(t / 10);

if (a[2] > '9')

{

char t1 = (char)(a[2] - '0');

a[2] = (char)(t1 % 10 + '0');

a[1] += (char)(t1 / 10);

if (a[1] > '9')

{

char t2 = (char)(s1[1] - '0');

a[1] = (char)(t2 % 10 + '0');

a[0] += (char)(t2 / 10);

}

}

}

for (int j = 0; j < a.Length; j++)

l.A[i].MaGD = l.A[i].MaGD + a[j];

b[b.Length - 1]++;

for (int j = 0; j < b.Length; j++)

l.A[i].TenKhach = l.A[i].TenKhach + b[j];

l.A[i].CapNhat(50000 + (tien \* i));

l.A[i].HashXD.n = 0;

}

}

1. *Tạo ID*

//Tạo ID

static public void TaoID(ref ListGD l, int start, int end, MTree T)

{

if (T.right.right == null && T.left.left != null)

{

for (int i = start; i <= end; i++)

{

int k = l.A[i].HashXD.n;

l.A[i].HashXD.lHash[k]=T.right.hash;

l.A[i].HashXD.n++;

}

TaoID(ref l, start, end, T.left);

}

else

if (T.left.left == null && T.right.right == null)

{

l.A[start].HashXD.lHash[l.A[start].HashXD.n++]=T.right.hash;

l.A[end].HashXD.lHash[l.A[end].HashXD.n++]=T.left.hash;

return;

}

else

{

int m = (end - start + 1) / 2;

if (m % 2 == 1) m++;

m += start;

for (int i = start; i < m; i++)

{

int k = l.A[i].HashXD.n;

l.A[i].HashXD.lHash[k]=T.right.hash;

l.A[i].HashXD.n++;

}

TaoID(ref l, m, end, T.right);

for (int i = m; i <= end; i++)

{

int k = l.A[i].HashXD.n;

l.A[i].HashXD.lHash[k]=T.left.hash;

l.A[i].HashXD.n++;

}

TaoID(ref l, start, m - 1, T.left);

}

}

1. *Sắp xếp ID*

//sắp xếp các Hash theo thứ tự sử dụng

//sắp xếp theo chiều dài tăng dần //sử dụng thuật toán sắp xếp nổi bọt

static public void sapxepID(ref ListGD l)

{

for (int i = 0; i <= l.n; i++)

for (int j = 0; j < l.A[i].HashXD.n- 1; j++)

{

for (int k = l.A[i].HashXD.n - 1; k > j; k--)

if (l.A[i].HashXD.lHash[k].Length < l.A[i].HashXD.lHash[k -1].Length)

XuLyGD.hoanvi(ref l.A[i].HashXD.lHash[k], ref l.A[i].HashXD.lHash[k- 1]);

}

}

static public void hoanvi(ref string a, ref string b)

{

string t = a;

a = b;

b = t;

}

1. *Xác thực Giao dịch*

//Xác thực giao dịch

static public bool XacThuc(GiaoDich x, string HashRoot)

{

string kq;

kq=Hash.Hash1(x);

for (int i = 0; i < x.HashXD.n; i++)

kq=Hash.Hash2(kq, x.HashXD.lHash[i]);

//cout << kq;

return kq == HashRoot;

}

1. *Tìm kiếm ID*

//Tìm kiếm ID trong ListGD

// Không thể tìm bằng họ tên vì khóa chính là MaGD

//Mã giao dịch được tạo theo thứ tự tăng dần nên tìm theo thuật toán tìm kiếm nhị phân

static public ID TimKiemID(ListGD l, int start, int end, int iMaGD)

{

if (start > end)

{

ID noExist=new ID();

noExist.n = -1;

noExist.lHash = null;

return noExist;

}

int i = start + (end - start) / 2;

int y = int.Parse(l.A[i].MaGD);

if (iMaGD == y)

return l.A[i].HashXD;

else if (iMaGD < y)

return TimKiemID(l, start, i - 1, iMaGD);

else

return TimKiemID(l, i + 1, end, iMaGD);

}

1. **Class MTree**
2. **Fields, Constructors, Finalizer**

public string hash;

public MTree left;

public MTree right;

public MTree()

{

this.left = null;

this.right = null;

}

public MTree(string hash1, MTree left, MTree right) //Thay cho hàm tạo LLeaf

{

this.hash = hash1;

this.left = left;

this.right = right;

}

~MTree()

{

}

1. **Tạo Branch**

// Tạo Branch

public static MTree TaoBranch(MTree x, MTree y)

{

MTree n = new MTree();

n.left = x;

n.right = y;

n.hash=Hash.Hash2(x.hash, y.hash);

return n;

}

1. **Tạo Root**

//Tạo Root

public static MTree TaoRoot(MTree[] LLeaf, int n)

{

if (n == 1) return LLeaf[0];

else

{

if (n % 2 == 1)

{

LLeaf[n] = new MTree();

LLeaf[n].hash = LLeaf[n - 1].hash;

LLeaf[n].left = null;

LLeaf[n].right = null;

n++;

}

int m = 0;

MTree[] LBranch = new MTree[10];

for (int i = 0; i < n; i = i + 2)

{

LBranch[m++] = MTree.TaoBranch(LLeaf[i], LLeaf[i + 1]);

}

return TaoRoot(LBranch, m);

}

}

1. **Duyệt**

//Duyệt

public static void xuatMTree(MTree T)

{

if (T == null) return;

else

{

Console.WriteLine(T.hash);

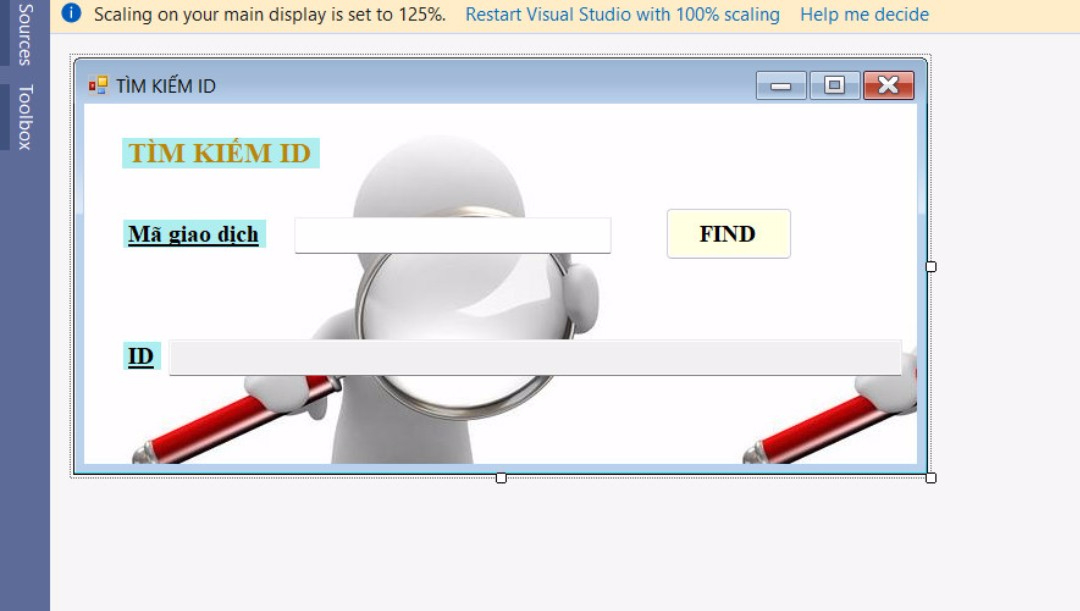
xuatMTree(T.left);

xuatMTree(T.right);

}

}

1. **Form Tìm kiếm ID**



Form được thiết kế như trên hình và để tìm kiếm ID của 1 giao dịch ta phải nhập vào textbox ở chỗ mã giao dịch với mã giao dịch phù hợp và click nút button Find để tìm kiếm ID. Code trong sự kiện click button này trình bày như sau:

private void button1\_Click(object sender, EventArgs e)

{

int n;

if (int.TryParse(textBox1.Text, out n) == false)

{

textBox2.Clear();

textBox2.Text="Không tìm thấy, vui lòng nhập lại";

return;

}

MTree T = new MTree();

ListGD l = new ListGD(0);

XuLyGD.TaoDuLieu(ref l);

MTree[] LLeaf = new MTree[100];

for (int i = 0; i < l.n; i++)

{

string h = Hash.Hash1(l.A[i]);

LLeaf[i] = new MTree(h, null, null);

}

T = MTree.TaoRoot(LLeaf, l.n);

XuLyGD.TaoID(ref l, 0, l.n - 1, T);

XuLyGD.sapxepID(ref l);

ID x = XuLyGD.TimKiemID(l, 0, l.n - 1, n);

if (x.n < 0)

{

textBox2.Clear();

textBox2.Text="Không tìm thấy, vui lòng nhập lại";

}

else

{

string t = x.lHash[0] + "\_";

for (int i = 1; i < x.n; i++)

t += x.lHash[i] + "\_";

textBox2.Clear();

textBox2.Text = t;

}

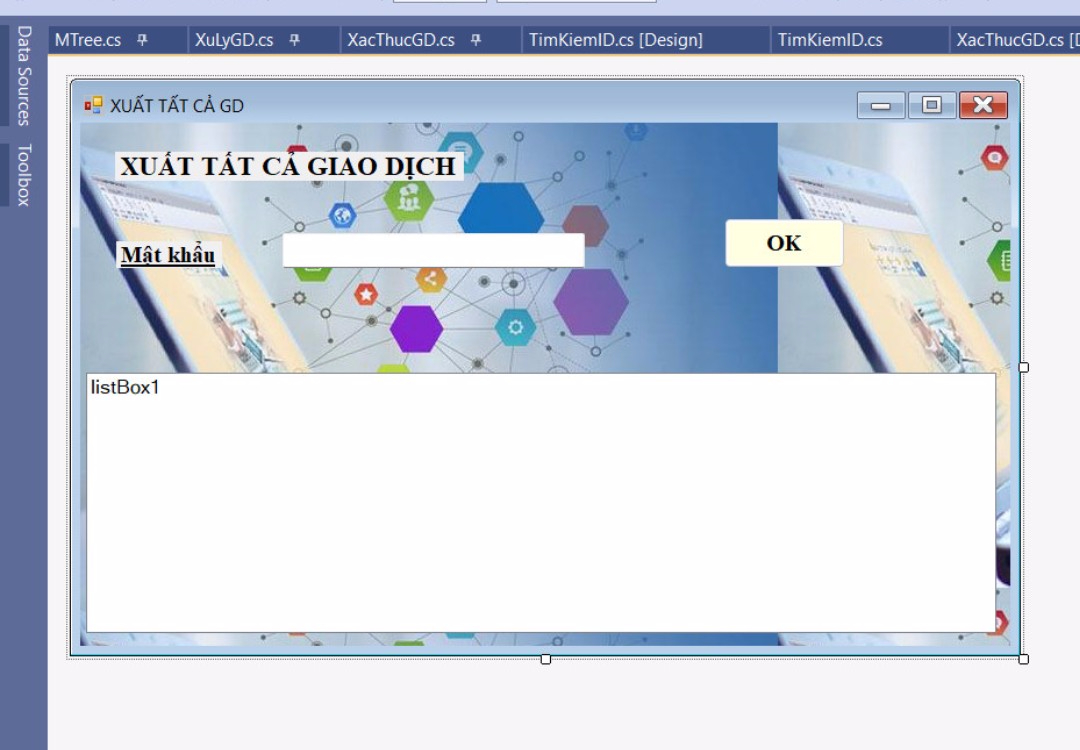
}

//textBox1 là textBox tại vị trí Mã giao dịch

//TextBox2 là textBox tại vị trí ID

//Button1 là button Find

1. **Form Xuất tất cả Giao dịch**



Form được thiết kế như trên hình và để xuất tất cả các giao dịch ta phải nhập vào textbox ở chỗ Mật khẩu với mật khẩu chính xác và click nút button OK để xuất ra được tất cả các giao dịch . Code trong sự kiện click button này trình bày như sau:

private void button1\_Click(object sender, EventArgs e)

{

listBox1.Items.Clear();

if(textBox1.Text!="12345") //Mật khẩu là 12345;

{

listBox1.Items.Add("Nhập mật khẩu sai, vui lòng nhập lại !!");

}else

{

MTree T = new MTree();

ListGD l = new ListGD(0);

XuLyGD.TaoDuLieu(ref l);

MTree[] LLeaf = new MTree[100];

for (int i = 0; i < l.n; i++)

{

string h = Hash.Hash1(l.A[i]);

LLeaf[i] = new MTree(h, null, null);

}

T = MTree.TaoRoot(LLeaf, l.n);

XuLyGD.TaoID(ref l, 0, l.n - 1, T);

XuLyGD.sapxepID(ref l);

for(int i=0; i<l.n; i++)

{

string m = "Mã Giao dịch: " + l.A[i].MaGD ;

listBox1.Items.Add(m);

m = "Tên khách: " + l.A[i].TenKhach ;

listBox1.Items.Add(m);

m = "Số tiền: " + l.A[i].SoTien;

listBox1.Items.Add(m);

m = "ID: ";

for (int j=0; j < l.A[i].HashXD.n; j++)

{

m += l.A[i].HashXD.lHash[j]+ "\_";

}

listBox1.Items.Add(m);

listBox1.Items.Add("");

}

}

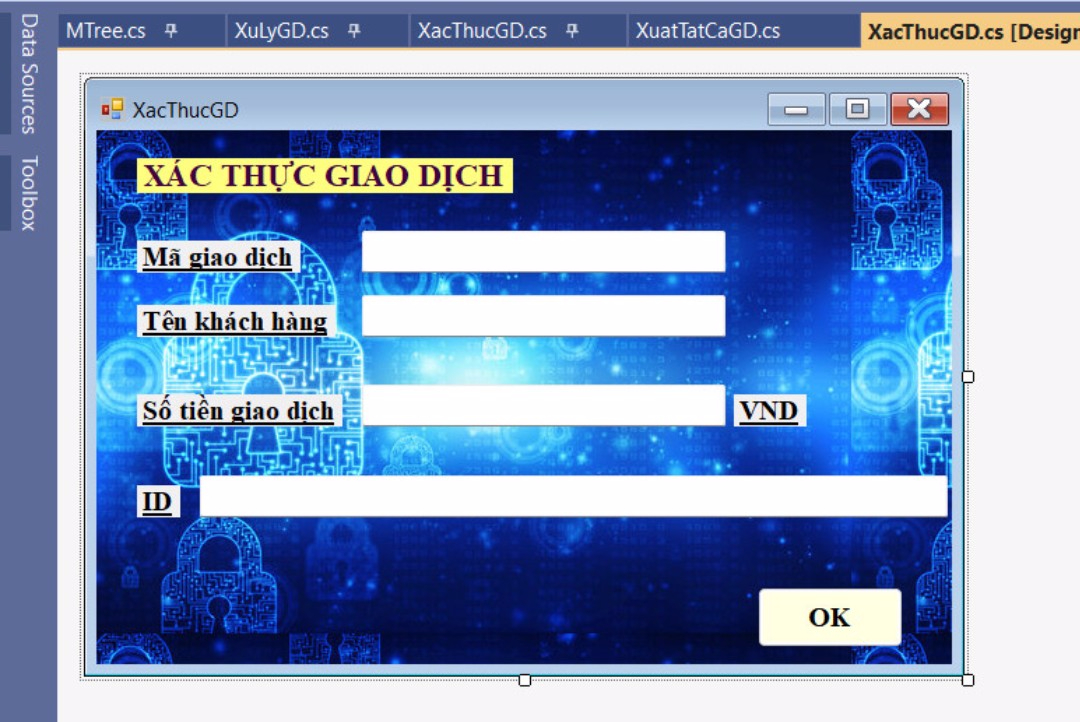
}

//listBox1 là list box trên hình

//TextBox1 là textbox tại chỗ mật khẩu

//Button1 là button OK

1. **Form Xác thực giao dịch**



Form được thiết kế như trên hình và để xác thực các giao dịch ta phải nhập vào textbox ở chỗ Mã giao giao dịch, Tên khách hàng, Số tiền giao dịch, ID chính xác và click nút button OK . Code trong sự kiện click button này trình bày như sau:

private void button1\_Click(object sender, EventArgs e)

{

MTree T = new MTree();

ListGD l = new ListGD(0);

XuLyGD.TaoDuLieu(ref l);

MTree[] LLeaf = new MTree[100];

for (int i = 0; i < l.n; i++)

{

string h = Hash.Hash1(l.A[i]);

LLeaf[i] = new MTree(h,null,null);

}

T = MTree.TaoRoot(LLeaf, l.n);

XuLyGD.TaoID(ref l, 0, l.n - 1, T);

XuLyGD.sapxepID(ref l);

GiaoDich x = new GiaoDich();

x.MaGD = textBox1.Text;

x.TenKhach = textBox2.Text;

x.SoTien = long.Parse(textBox3.Text);

int vt1 = 0;

x.HashXD.n = 0;

char[] lH = textBox4.Text.ToCharArray();

for (int vt2 = 0; vt2 < lH.Length; vt2++)

{

if (lH[vt2] == '\_')

{

int k = x.HashXD.n;

string str = textBox4.Text.Substring(vt1, vt2-vt1);

x.HashXD.lHash[k] = str;

x.HashXD.n++;

vt1 = vt2 + 1;

}

}

if (XuLyGD.XacThuc(x, T.hash))

MessageBox.Show("Giao dịch đã được thực hiện !!");

else

MessageBox.Show("Giao dich chưa được thực hiện!Hãy kiểm tra lại thông tin!");

}

//textBox1 là textBox ở Mã giao dịch

//textBox2 là textBox ở Tên khách hàng

//textBox3 là textBox ở Số tiền giao dịch

//textBox4 là textBox ở ID

//Button1 là button OK

1. **Form Main**



Chỉ là giao diện để gọi các form khác, các Form khác sẽ được hiện lên khi Click vào các Button sẽ hiện ra Form tương ứng từng button và không thể hiện 1 lần nhiều Form. Và code trong từng Button:

private void button1\_Click\_1(object sender, EventArgs e)

{

XacThucGD f= new XacThucGD();

f.ShowDialog();

}

private void button2\_Click(object sender, EventArgs e)

{

TimKiemID f = new TimKiemID();

f.ShowDialog();

}

private void button3\_Click(object sender, EventArgs e)

{

XuatTatCaGD f = new XuatTatCaGD();

f.ShowDialog();

}

// button1 là button Xác thực giao dịch xuất hiện Form Xác thực giao dịch

// button2 là button Tìm kiếm ID xuất hiện Form Tìm kiếm ID

// button3 là button Xuất tất cả các giao dịch xuất hiện Form Xuất tất cả giao dịch

1. **Thư viện sử dụng**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;