**DANH SÁCH LIÊN KẾT**

**//a. them 1 ptu vao sau Y xuat hien cuoi cung**

void InsertZY(ElementType Z,ElementType Y,List &L){

Position P;

P=L;

int d2,d1=0;

while(P->Next!=NULL){

if(P->Next->Element==Y){

d1=d1+1;

cout <<"d1 = "<< d1 << '\n';

}

P=P->Next;

}

P=L;

while(P->Next!=NULL){

if(P->Next->Element==Y){

d2=d2+1;

if(d2==d1){

cout <<"d2 = "<< d2 << '\n';

P=P->Next;

InsertList(Z,P,L);

}

}

P=P->Next;

}

}

**//d. xoa tat ca pt X co trong danh sach(1)**

void DeleteX(ElementType X,List &L){

Position P;

P=L;

while (P->Next!=NULL){

if(P->Next->Element==X){

DeleteList(P,L);

}

else

P=P->Next;

}

}

**//f. In tu X xuat hien o lam thu K den cuoi danh sach**

void PrintXK(ElementType X, int K, List L){

Position P=L;

int d=1;

while (P->Next!=NULL){

if(P->Next->Element==X){

if(d==K){

while(P->Next!=NULL){

cout << P->Next->Element<<" ";

P=P->Next;

}

}

else{

d+=1;

}

}

P=P->Next;

}

}

**//g. tách 1 ds thành 2 ds chua nd tuong ung là chan le**

void Separate(List &L, List &O){

Position P=L;

Position K=O;

ElementType t;

while(P->Next!=NULL){

if(P->Next->Element%2==0){

t=P->Next->Element;

InsertList(t,K,O);

K=K->Next;

DeleteList(P,L);

}

P=P->Next;

}

}

**//h. ghep ds 2 vao chinh giua ds 1**

void Graft(List &L, List &O){

Position P=L;

Position K=O;

ElementType t;

int d1,d2=0;

while(P->Next!=NULL){

P=P->Next;

d1+=1;

cout << d1 << '\n';

}

P=L;

while(P->Next!=NULL){

if(d2==d1/2){

while(K->Next!=NULL){

cout << "111"<<'\n';

t=K->Next->Element;

InsertList(t,P,L);

P=P->Next;

DeleteList(K,O);

}

break;

}

else{

d2+=1;

P=P->Next;

}

}

}

**//a. sap xep ds tang dan**

void TangDan(List &L){

Position I=L;

Position J;

ElementType X;

while(I->Next!=Last(L)){

for(J=I->Next;J->Next!=NULL;J=J->Next){

if(I->Next->Element > J->Next->Element){

X = I->Next->Element;

I->Next->Element = J->Next->Element;

J->Next->Element = X;

}

}

I=I->Next;

}

}

**//sap xep ds giam dan**

void GiamDan(List &L){

Position I,J;

I=L->Next;

ElementType X;

while(I->Next!=NULL){

for(J=I->Next;J!=NULL;J=J->Next){

if(I->Element < J->Element){

X = I->Element;

I->Element = J->Element;

J->Element = X;

}

}

I=I->Next;

}

}

**//b. ghep 2 ds co thu tu giam dan thanh 1 ds co thu tu giam dan**

Ghép 2 ds thành 1 ds rồi dùng hàm sx giảm

**//d. tim ptu MIN**

ElementType MinList(List L){

ElementType X=L->Next->Element;

for (Position P=L->Next->Next; P!=NULL; P=P->Next){

if (P->Element < X){

X = P->Element;

}

}

return X;

}

**//g. cho bt ptu nho nhat la ptu thu may trong ds**

int ViTriMin(List L){

Position P=L->Next;

int d=1;

while (P!=NULL){

if (P->Element == MinList(L)){

return d;

}

else{

P=P->Next;

d++;

}

}

}

**//i. tong ptu co phan nguyen la boi so cua K**

ElementType TongBoiSo(int K, List L){

Position P=L->Next;

ElementType S=0;

while (P!=NULL){

if (K % int(P->Element) == 0){

S = S + P->Element;

}

P=P->Next;

}

return S;

}

**//xay dung cau truc theo ds lien ket**

struct Sinhvien{

int mssv;

char hoten[30];

float dtb;

}SV;

typedef Sinhvien ElementType;

struct Node{

ElementType Element;

Node\* Next;

};

typedef Node\* Position;

typedef Node\* List;

void MakeNullList(List &L){

L = new Node;

L->Next = NULL;

}

bool EmptyList(List L){

return L->Next==NULL;

}

Position Last(List L){

Position P=L;

while (P->Next!=NULL)

P=P->Next;

return P;

}

void InsertList(ElementType X, Position P, List &L){

Position t;

t = new Node;

t->Element.mssv=X.mssv;

strcpy(t->Element.hoten,X.hoten);

t->Element.dtb=X.dtb;

t->Next=P->Next;

P->Next=t;

}

void ReadList(List &L){

ElementType X;

do{

cout << "Nhap ma so sinh vien: ";

cin >> X.mssv;

if (X.mssv==0)

break;

cin.ignore(1);

cout << "Nhap ho ten sinh vien: ";

cin.getline(X.hoten,30);

cout << "Nhap diem trung binh: ";

cin >> X.dtb;

InsertList(X,Last(L),L);

}while(X.mssv!=0);

}

void PrintList(List L){

if (EmptyList(L)!=0){

cout << "Danh sach rong" << endl;

}

else{

int i=0;

while (L->Next!=NULL){

cout << "Thong tin sinh vien thu " << i<<endl;

cout << "mssv: " << L->Next->Element.mssv << '\n';

cout << "ho ten: " << L->Next->Element.hoten << '\n';

cout << "diem trung binh: " << L->Next->Element.dtb << '\n';

i++;

L=L->Next;

}

}

}

**// dem so sinh vien co trong ds**

int Reccount(List L){

Position P=L;

int d=0;

while (P->Next!=NULL){

P=P->Next;

d++;

}

return d;

}

**//them 1 sv vao vi tri i**

void addSV(List &L,int i){

ElementType X;

cout << "Nhap ma so sinh vien: ";

cin >> X.mssv;

cin.ignore(1);

cout << "Nhap ho ten sinh vien: ";

cin.getline(X.hoten,30);

cout << "Nhap diem trung binh: ";

cin >> X.dtb;

if (i==0)

InsertList(X,L,L);

if (i>Reccount(L))

InsertList(X,Last(L),L);

if (i>0 && i<=Reccount(L)){

Position P=L;

while (i>0){

P=P->Next;

i--;

}

InsertList(X,P,L);

}

}

**STACK**

typedef int ElementType;

struct Node{

ElementType Element;

Node\* Next;

};

typedef Node\* Stack;

typedef Node\* Position;

void MakeNullStack(Stack &S){

S=NULL;

}

bool EmptyStack(Stack S){

if (S==NULL) return true;

else return false;

}

**//them 1 ptu vao stack**

void Push(ElementType X, Stack &S){

Position T;

T=new Node;

T->Element=X;

T->Next=S;

S=T;

}

**//lay ptu tren cung cua stack**

ElementType Top(Stack S){

if (EmptyStack(S)) return 0;

else return S->Element;

}

**//xoa 1 ptu tu stack neu ko con thi tu xoa chinh no**

void Pop(Stack &S){

if (!EmptyStack(S)){

Position T=S;

S=S->Next;

delete T;

}

}

void ReadStack(Stack &S){

ElementType X;

do{

cout << "Nhap noi dung ptu: ";

cin>>X;

if(X!=0) Push(X,S);

}while(X!=0);

}

void PrintStack(Stack &S){

if(EmptyStack(S)) cout << "Danh sach rong"<<endl;

else

while(S!=NULL){

cout << Top(S)<<" ";

Pop(S);

}

}

**//doi 10 sang 2**

void Dec\_Bin(int n, Stack &S){

int a;

while(n > 0){

a=n%2;

Push(a,S);

n=n/2;

}

cout << "nhi phan: ";

PrintStack(S);

cout << endl;

}

main(){

Stack S;

MakeNullStack(S);

int n;

cout << "Nhap so thap phan: ";

cin>>n;

Dec\_Bin(n,S);

}

**CÂY NHỊ PHÂN**

**//a. in cac nut la**

void LRootR\_Traverse(BinT\_Type BTree){

if (BTree==NULL) return;

LRootR\_Traverse(BTree->Left);

if(BTree->Left==NULL && BTree->Right==NULL)

cout << BTree->Key << " ";

LRootR\_Traverse(BTree->Right);

}

**//b. in cac nut o muc K**

void InMucK(int K, BinT\_Type BTree){

if (BTree==NULL) return;

if (K==1){

cout << BTree->Key<<" ";

return;

}

InMucK(K-1,BTree->Left);

InMucK(K-1,BTree->Right);

}

**//d. in cac nut con trai cua cay**

void InTrai(BinT\_Type BTree){

if (BTree==NULL) return;

InTrai(BTree->Left);

InTrai(BTree->Right);

if(BTree->Left!=NULL)

cout << BTree->Left->Key<<" ";

}

**//f. dem so nut bac 2 cua cay**

int DemNutBac2(BinT\_Type BTree){

if (BTree==NULL) return 0;

int v1=DemNutBac2(BTree->Left);

int v2=DemNutBac2(BTree->Right);

if(BTree->Left!=NULL && BTree->Right!=NULL)

return v1+v2+1;

else

return v1+v2;

}

**//i. cho biet nut lon nhat trong tree**

Info NutMax(BinT\_Type BTree){

if (BTree==NULL) return 0;

int v1=NutMax(BTree->Left);

int v2=NutMax(BTree->Right);

int max = BTree->Key;

if (max<v1) max=v1;

if (max<v2) max=v2;

return max;

}

**//j. tinh tong cac nut co noi dung le**

Info TongNutLe(BinT\_Type BTree){

if (BTree==NULL) return 0;

int v1=TongNutLe(BTree->Left);

int v2=TongNutLe(BTree->Right);

if (BTree->Key % 2 != 0)

return v1+v2+BTree->Key;

else

return v1+v2;

}

**//l. cho biet muc nao co nhieu nut nhat**

**//dem so nut o muc k**

int SoNutMuck(BinT\_Type BTree, int k){

if (BTree==NULL) return 0;

if (k==1) return 1;

int v1 = SoNutMuck(BTree->Left,k-1);

int v2 = SoNutMuck(BTree->Right,k-1);

return v1+v2;

}

Info MucNutMax(BinT\_Type BTree){

int \*a;

int h = Tree\_Height(BTree);

a = new int [h];

for(int i=0; i<=h-1; i++)

a[i] = SoNutMuck(BTree,i+1);

int max=a[1];

for(int i=0; i<=h-1; i++)

if(a[i]>max)

max = a[i];

cout << max<<endl;

for(int i=0; i<=h-1; i++)

if(a[i]==max) cout << i+1 << " ";

}

**//m. chieu dai duong di trong**

Info RoadIn(BinT\_Type BTree, Info m=1){

if (BTree==NULL) return 0;

int v1 = RoadIn(BTree->Left,m+1);

int v2 = RoadIn(BTree->Right,m+1);

return v1+v2+m;

}

**//n. chieu dai duong di ngoai**

Info RoadOut(BinT\_Type BTree, Info m=1){

if (BTree==NULL) return m;

int v1 = RoadIn(BTree->Left,m+1);

int v2 = RoadIn(BTree->Right,m+1);

return v1+v2;

}

**//tao BST tu cay nhi phan co san (duyệt NLR, add từng ptu qua BST)**

void BT\_BST(BinT\_Type BT, BinT\_Type &BST){

if (BT==NULL)

return;

Add\_Node(BST, BT->Key);

BT\_BST(BT->Left, BST);

Add\_Node(BST, BT->Key);

BT\_BST(BT->Right, BST);

Add\_Node(BST, BT->Key);

}

**CÂY NHỊ PHÂN TÌM KIẾM**

**BST**

typedef int Info;

struct BST\_Node{

BST\_Node\* Left;

BST\_Node\* Right;

int Count;

Info Key;

};

typedef BST\_Node\* BST\_Type;

void Initialize(BST\_Type &BST){

BST = NULL;

}

BST\_Type Create\_Node(Info NewData){

BST\_Type BSTNode = new BST\_Node;

if (BSTNode != NULL){

BSTNode->Left = NULL;

BSTNode->Right = NULL;

BSTNode->Key = NewData;

}

return BSTNode;

}

void Read\_Tree (BST\_Type &BST){

Info X;

cin >> X;

if (X!=0){

BST\_Type BT;

BT = Create\_Node(X);

BST = BT;

cout << "Nhap con trai cua "<<X<<" : ";

Read\_Tree(BST->Left);

cout << "Nhap con phai cua "<<X<<" : ";

Read\_Tree(BST->Right);

}

else BST=NULL;

}

void InLNR(BST\_Type BST){

if (BST == NULL) return;

InLNR(BST->Left);

cout << BST->Key << " ";

InLNR(BST->Right);

}

**//a. in tu goc den nut K**

void RtoK(BST\_Type BST,Info K){

if (BST!=NULL){

cout << BST->Key << " ";

if (BST->Key < K)

RtoK(BST->Right,K);

else RtoK(BST->Left,K);

}

}

**//b. in tu K ve goc**

void KtoR(BST\_Type BST,Info K){

if (BST!=NULL){

if(BST->Key < K)

KtoR(BST->Right,K);

else KtoR(BST->Left,K);

cout << BST->Key << " ";

}

}

**//c. in tu goc den nut max**

void RtoMax(BST\_Type BST){

if (BST!=NULL){

cout << BST->Key<<" ";

RtoMax(BST->Right);

}

}

**//d. in tu nut mic ve goc**

void MintoR(BST\_Type BST){

if (BST!=NULL){

MintoR(BST->Left);

cout << BST->Key<<" ";

}

}

**//e. in tu min den max**

void MintoMax(BST\_Type BST){

MintoR(BST);

RtoMax(BST->Right);

}

**//g. in cac nut lon hon K**

void RK(BST\_Type BST , Info K){

if (BST!=NULL){

if(BST->Key==K)

InLNR(BST->Right);

else if(BST->Key < K)

RK(BST->Right, K);

else{

RK(BST->Left,K);

cout << BST->Key <<" ";

InLNR(BST->Right);

}

}

}

**//tim kiem 1 nut**

BST\_Type Searching(BST\_Type BST, Info SearchData){

BST\_Type CurNode = BST;

while (CurNode != NULL && CurNode->Key != SearchData)

if (CurNode->Key > SearchData)

CurNode = CurNode->Left;

else

CurNode = CurNode->Right;

return CurNode;

}

**//them mot nut**

void Add\_Node (BST\_Type &BST, Info NewData){

if (BST == NULL){

BST = new BST\_Node;

if(BST!=NULL){

BST->Left = NULL;

BST->Right = NULL;

BST->Key = NewData;

}

}

else

if (BST->Key == NewData) return;

else

if (BST->Key > NewData)

Add\_Node(BST->Left, NewData);

else

Add\_Node(BST->Right, NewData);

}

**VÔ HƯỚNG ĐẦY**

**//a. cho biet dinh nao co so canh noi la K**

void DinhSoCanhNoiK(Graph g, int n, int e, int K){

int i;

for(i=1; i<=n; i++){

if(CountEdge(g,n,e,i) == K)

cout << i << " ";

}

}

**//b. in ra cac dinh co canh noi voi dinh nhap vao**

void DinhCoCN(Graph g, int n ,int e, int d){

for(int i=1; i<=n; i++){

if(g[d][i]==1 && d!=i)

cout << i << " ";

}

}

**//c. nhap 2 dinh, ktra canh noi**

bool KTCN(Graph g, int n, int e, int d1, int d2){

if(g[d1][d2]==1) return true;

else return false;

}

**//d. dem cac dinh treo**

int DemDT(Graph g, int n ,int e){

int dem;

for(int i=1; i<=n; i++){

if(CountEdge(g,n,e,i)==1)

dem++;

return dem;

}

}

**//e. in ra cac dinh co canh noi voi cac dinh treo**

void InDinhCanhDT(Graph g, int n, int e){

for(int i=1; i<=n; i++){

if(CountEdge(g,n,e,i)==1)

for(int j=1; j<=n; j++)

if(g[i][j]==1 && i!=j){

cout << j << " ";

break;

}

}

}

**HỮU HƯỚNG THƯA**

**//a. cho bt dinh co so cung den cac dinh khac nhieu nhat**

void cungXPmax(Graph g, int n, int e){

int \*a;

a = new int[n];

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

a[i] = CountArcFromVertice(g,n,e,i);

int max = a[1];

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

if (a[i]>max)

max = a[i];

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

if(a[i]==max)

cout << i << " ";

}

**//b. cho bt dinh co so cung den tu cac dinh khac it nhat**

void dinhcungdenmin(Graph g, int n, int e){

int min = CountArcToVertice(g,n,e,1);

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

if (min > CountArcToVertice(g,n,e,i))

min = CountArcToVertice(g,n,e,i);

cout << "dinh co cung den tu cac dinh khac it nhat: ";

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

if (CountArcToVertice(g,n,e,i)==min)

cout << i <<" ";

}

**//c. dem so khuyen**

bool ktkhuyen(List t, int d){

while (t->Next!=NULL && t->Next->Element!=d)

t=t->Next;

if (t->Next!=NULL)

return true;

else return false;

}

int demkhuyen(Graph g, int n, int e){

int dem=0;

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

if (ktkhuyen(g[i],i))

dem++;

return dem;

}

**//d. cho biet co bao nhieu dinh treo trong do thi**

int demdinhtreo(Graph g, int n, int e){

int dem=0;

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

if (CountArcFromVertice(g,n,e,i)==0 && CountArcFromVertice(g,n,e,i))

dem++;

return dem;

}

**//e. in ra cac dinh co cung den dinh d**

void indinh(Graph g, int n, int e, int d){

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

if(ktkhuyen(g[i],d))

cout << i << " ";

}

**Duyệt cây NLR và LNR không đệ qui**

**//Cai dat Cay nhi phan**

typedef int Info;

struct BinT\_Node{

Info Key;

BinT\_Node \*Left;

BinT\_Node \*Right;

};

typedef BinT\_Node \*BinT\_Type;

void Initialize(BinT\_Type &BTree){

BTree=new BinT\_Node;

BTree->Left=NULL;

BTree->Right=NULL;

}

void ReadTree(BinT\_Type &BTree){

Info X;

cin>>X;

if(X!=0){

BTree=new BinT\_Node;

BTree->Key=X;

BTree->Left=NULL;

BTree->Right=NULL;

cout<<"Nhap con trai cua "<<X<<": ";

ReadTree(BTree->Left);

cout<<"Nhap con phai cua "<<X<<": ";

ReadTree(BTree->Right);

}

}

**//Cai dat Stack**

typedef BinT\_Type ElementType;

struct Node{

ElementType Element;

Node \*Next;

};

typedef Node \*Position;

typedef Node \*Stack;

void MakeNullStack(Stack &S){

S=NULL;

}

bool EmptyStack(Stack S){

return S==NULL;

}

void Push(ElementType X, Stack &S){

Position Temp = new Node;

Temp->Element =X;

Temp->Next=S;

S=Temp;

}

BinT\_Type Top(Stack S){

return S->Element;

}

void Pop(Stack &S){

Position Temp;

Temp=S;

S=Temp->Next;

delete Temp;

}

void DuyetNLR(BinT\_Type BTree){ //Node-Left-Right

BinT\_Type Current=new BinT\_Node;

Stack S;

Current=BTree;

MakeNullStack(S);

while(Current!=NULL){

cout<<Current->Key<<"\t";

if(Current->Right!=NULL)

Push(Current,S);

if(Current->Left != NULL)

Current=Current->Left;

else{

Current=Top(S)->Right;

Pop(S);

}

}

}

void DuyetLNR(BinT\_Type BTree){

bool flag=false;

Stack S;

BinT\_Type Current=new BinT\_Node;

Current=BTree;

MakeNullStack(S);

while(Current!=NULL){

if(flag==false)

Push(Current, S);

if(Current->Left!=NULL){

Current=Current->Left;

}

else{

Current=Top(S);

Pop(S);

cout<<Current->Key<<"\t";

flag=true;

if(Current->Right!=NULL){

Current=Current->Right;

flag=false;

}

}

}

}

main(){

BinT\_Type BTree;

// Stack S;

Initialize(BTree);

ReadTree(BTree);

// cout<<"Duyet cay Node-Left-Right: ";

// DuyetNLR(BTree, S);

cout<<"Duyet cay Left-Node-Right: ";

DuyetLNR(BTree);

}