STRUCT AND ALGORITHMS

4 - 1

Binary Tree- Cây nhi phân

Phần 1: Cơ sở lý thuyết

Xem lại slide bài giảng

Phần 2: Thực hành

1. Nhập N ≤ 1000 số nguyên từ bàn phím.Viết chương trình thực hiện :
    a. Áp dụng thuật toán tạo cây nhị phân tìm kiếm để tạo cây từ dãy số trên.
    b. Viết thủ tục duyệt cây để kết quả in ra dãy số giảm dần.
    c. Đếm số nút có bậc lẻ trong cây.
    d. Yêu cầu người dùng nhập vào giá trị x cần huỷ và huỷ nó ra khỏi cây

```
//

#include <iostream>
using namespace std;
int menu()
{
     int option;
     cout << "-----Menu----";
     cout << "\n1. Init Tree";
     cout << "\n2. Add Tree";
     cout << "\n3. Push into Stack";
     cout << "\n4. Pop out Stack";
     cout << "\n5. Get item top Stack";
     cout << "\n6. Exit";
     cout << "seleted Option: "; cin >> option;
```

```
     return option;
}
typedef struct Node{
     int Data;
     Node *L, *R;
}NodeType;

typedef NodeType *NodePtr;
NodePtr CreateNode(int X)
{
     NodePtr P = (NodePtr)malloc(sizeof(NodeType));
     if (P != NULL)
     {
          P->Data = X;
          P->L = P->R = NULL;
     }
     return P;
}

void InitTree(NodePtr &Root)
{
     Root = NULL;
}
void InsertTree(NodePtr &Root, NodePtr P)
{
     if (Root == NULL) Root = P;
     else
     {
          if (Root->Data>P->Data)
               InsertTree(Root->L, P);
          else if (Root->Data<P->Data)
               InsertTree(Root->R, P);
          else {
               cout<<"\nGia tri bi trung lap ko the them vao
cay nhi phan";
               exit(1);
          }
     }
}
void Input(NodePtr &Root, int &N)
```

```
{
     cout << "Nhap so phan tu N = "; cin >> N;
     int X;
     for (int i = 0; i<N; i++)
     {
          cout << "Nhap phan tu " << i << ": ";
          cin >> X;
          NodePtr P = CreateNode(X);
          InsertTree(Root, P);
     }
}
void RNL(NodePtr Root)
{
     if (Root != NULL)
     {
          RNL(Root->R);
          cout << Root->Data;
          RNL(Root->L);
     }
}
int DemBacLe(NodePtr Root)
{
     if (Root != NULL)
     {
          int dem = 0;
          if ((Root->L != NULL && Root->R == NULL) ||
(Root->L == NULL && Root->R != NULL))
               dem++;
          return dem + DemBacLe(Root->L) +
DemBacLe(Root->R);
     }
     return 0;
}
void TimTheMang(NodePtr &Root, NodePtr &P)
{
     if (Root->R == NULL)
     {
          P->Data = Root->Data;
          Root = Root->L;
     }
```

```
     else TimTheMang(Root->R, P);
}
void DelNode(NodePtr &Root, int X)
{
     if (Root->Data>X)
          DelNode(Root->L, X);
     else if (Root->Data<X)
          DelNode(Root->R, X);
     else
     {
          if (Root->L == NULL)
               Root = Root->R;
          else if (Root->R == NULL)
               Root = Root->L;
          else
          {
               NodePtr P = Root;
               TimTheMang(Root->L, P);
          }
     }
}
void KQDel(NodePtr &Root)
{
     int X;
     cout<<"\nNhap gia tri nut muon xoa: ";
     cin>>X;
     DelNode(Root, X);
     cout<<"\nDanh sach cac node trong cay sau khi xoa: ";
     RNL(Root);
}
void main()
{
     int option = 0;
     do
     {
          option = menu();
          switch (option)
          {
          case 1:
```

```
            break;
        case 2:

            break;
        }
    } while (option != 6);
}
```

Bài 02:

Xây dựng cấu trúc cây để lưu số nguyên và thực hiện thao tác
nhập số N <=1000 từ bàn phím. viết chương trình thực hiện
//1. Nhập N số từ số từ bàn phìm và theo vào cây nhị phân.
//2.Viết thủ tục để duyệt cây và in ra dãy số giảm dần.
//3.Đếm số nút lẻ trong cây.
//4. xóa 1 nút có giá trị x nhập từ bàn phím từ cây

```
#include <iostream>
using namespace std;
int menu()
{
    int option;
    cout << "-----Menu----";
    cout << "\n1. Init Tree";
    cout << "\n2. Input Tree";
    cout << "\n3. Print Tree DESC";
    cout << "\n4. Count node in tree.";
    cout << "\n5. Delete node in tree";
    cout << "\n6. Exit";
    cout << "seleted Option: "; cin >> option;
    return option;
}
// khai báo cấu trúc của cây.
typedef int item;
typedef struct node
{
    item data;
```

```
     node * left;
     node * right;
}NODETREE;
typedef NODETREE * ptrNODE;
//Cấu trúc cây
typedef struct tree
{
     ptrNODE root;
     int count;
}TREE;
//hàm khởi tạo cây rỗng
void Init(TREE &tree)
{
     tree.root = NULL;
     tree.count = 0;
}
ptrNODE GetNode(item value)
{
     ptrNODE p = new NODETREE;//tạo biến con trỏ p kiểu
NODETREE
     if (p == NULL)
          exit(0);
     p->data = value;
     p->left = p->right = NULL;
     return p;
}
void InsertTree(ptrNODE &root, ptrNODE p)
{
     if (root == NULL)
          root = p;
     else
     {
          if (root->data > p->data)
               InsertTree(root->left, p);
          else if (root->data < p->data)
```

```
                InsertTree(root->right, p);
            else
            {
                cout << "Node exsisted";
            }
        }
}
void InputTree(ptrNODE &root,int &N)
{
        cout << "Nhap N: "; cin >> N;
        int value;
        for (int i = 0; i < N; i++)
        {
            cout << "Nhap gia tri thu " << i + 1 << ": ";
            cin >> value;
            ptrNODE p = GetNode(value);
            InsertTree(root, p);

        }
}
void RNL(ptrNODE root)
{
        if (root == NULL)
            return;
        RNL(root->right);
        if (root->data > 7)
            cout << root->data << " ";
        RNL(root->left);
}
int CountNode(ptrNODE root)
{
        int dem = 0;
        if (root != NULL)
        {
            dem = 0;
```

```
          if (root->left != NULL &&root->right == NULL ||
root->left == NULL &&root->right != NULL)
               dem++;
          return dem + CountNode(root->left) +
CountNode(root->right);
     }
     else
     {
          return 0;
     }
}
//dem trong cay co bao nhieu nut lá
int CountNodeLeaf(ptrNODE root)
{
     int dem = 0;
     if (root != NULL)
     {
          dem = 0;
          if (root->left == NULL &&root->right == NULL )
               dem++;
          return dem + CountNodeLeaf(root->left) +
CountNodeLeaf(root->right);
     }
     else
     {
          return 0;
     }
}
int CountNode2child(ptrNODE root)
{
     int dem = 0;
     if (root != NULL)
     {
          dem = 0;
          if (root->left != NULL &&root->right != NULL)
```

```
                dem++;
            return dem + CountNode2child(root->left) +
CountNode2child(root->right);
        }
        else
        {
            return 0;
        }
}
void main()
{
        int option = 0;
        TREE tree;
        int N = 0;
        do
        {
            option = menu();
            switch (option)
            {
            case 1:
                Init(tree);
                break;
            case 2:
                InputTree(tree.root, N);
                break;
            case 3:
                RNL(tree.root);
                break;
            case 4:
                cout << "So nut co con le: " <<
CountNode(tree.root);
                break;
            }
        } while (option != 6);
}
```

Bài 03: Cây với kiểu dữ liệu tự xây dựng

```
//2. trong file bài tập
// Xây dựng cây nhị phân tìm kiếm để lưu thông tin độc giả:
Mã độc giả (MaDG-int), Tên đọc giả (tenDocGia-char[30]),
ngày sinh (ngaySinh-Date), địa chỉ (diaChi-char[100]), ngày
lập thẻ (ngayLapThe -Date)
//hãy viết chương trình để:
//1. Lưu thông tin đọc giả.
//2. Tìm kiếm đọc giả có tuổi lớn nhất trong cây.
//3. tính chiều cao của cây.
//4. đếm số nút có đủ 2 con có ngày lập thẻ trong ngày
09-05-2013
//phân tích bài toàn
//1. cấu trúc dữ liệu:
//    - Cấu trúc Cây nhị phân tìm kiếm.
// - cấu trúc kiểu dữ liệu đọc giả
(maDG,tenDocGia,ngaySinh,diaChi,ngayLapThe).
// - cấu trúc kiểu dữ liệu ngày tháng (ngay (int),
thang(int), nam(int))
//2. Có các chức năng:
//2.1. Thêm dữ liệu đọc giả vào cây.
//2.2. Tìm kiếm theo tuổi lớn nhất.
//2.3.tính chiều cao của cây.
//2.4 đếm nút 2 con và có ngày lập thẻ là ngày chỉ định
#include <iostream>
using namespace std;
int menu()
{
     int option;
     cout << "-----Menu----";
     cout << "\n1. Init";
     cout << "\n2. Add";
     cout << "\n3. Print";
     cout << "\n4. Pop out Stack";
     cout << "\n5. Get item top Stack";
     cout << "\n6. Exit";
     cout << "seleted Option: "; cin >> option;
```

```
     return option;
}

//1. Xây dựng cấu trúc dữ liệu cho bài toán
typedef struct date
{
     int day;
     int month;
     int year;
}DATE;
//Xây dựng kiểu dữ liệu đọc giả
typedef struct docgia
{
     int maDG;
     char tenDocGia[30],diaChi[100];
     DATE ngaySinh, ngayLapThe;
}DOCGIA;
//Xây dựng cấu trúc NODE
typedef struct node
{
     DOCGIA data;
     node *left;
     node *right;
}NODE;
//Cấu trúc cây nhị phân tìm kiếm
typedef struct tree
{
     NODE * root;
     int count;
}TREE;
typedef NODE *ptrNode;//khai báo 1 kiểu dữ liệu mới dạng
con trỏ node
void Init(TREE &tree)
{
     tree.root = NULL;
     tree.count = 0;
}
//2. Xây dựng các phương thức để thực hiện lần lượt các
chức năng.
```

```
//2.1 Các phương thức để thêm 1 Đọc giả vào cây.
//-------------Begin---------------------------
//Hàm nhập kiểu dữ liệu ngày thàng
DATE NhapNgay()
{
     DATE _date;
     cout << "Day: ";cin >> _date.day;
     cout << "Month: "; cin >> _date.month;
     cout << "Year: "; cin >> _date.year;
     return _date;
}
//2.1.1. Phương thức Nhập thông tin 1 đọc giả
DOCGIA NhapMotDocGia(int i)
{
     DOCGIA _docGia;
     cout << "Nhap thong tin doc gia thu " << i<<": ";
     cout << "Ma doc gia:"; cin >> _docGia.maDG;
     cout << "Ma ten doc gia:"; fflush(stdin);
cin.getline(_docGia.tenDocGia,30);
     cout << "Ma ngay sinh:"; _docGia.ngaySinh = NhapNgay();
     cout << "Ma dia chi:"; fflush(stdin);
cin.getline(_docGia.diaChi,100);
     cout << "Ma ngay lap the:";
_docGia.ngayLapThe=NhapNgay();
     return _docGia;
}
//xuất ngày tháng
void XuatNgayThang(DATE _date)
{
     cout << _date.day << "/" << _date.month << "/" <<
_date.year;
}
//2.1.2 Phương thức xuất thông tin đọc giả
void XuatMotDocGia(DOCGIA _docGia)
{
     cout << _docGia.maDG << "-" << _docGia.tenDocGia <<
"-";
     XuatNgayThang(_docGia.ngaySinh);
     cout << "-" << _docGia.diaChi << "-";
     XuatNgayThang(_docGia.ngayLapThe);
```

```
}
//2.2 Các phương thức để tạo node và thêm node vào câu
//2.2.1. Phương thức tạo node
ptrNode GetNode(DOCGIA _docGia)
{
     ptrNode p = new NODE;
     if (p == NULL)
          exit(1);
     p->data = _docGia;
     p->left = p->right = NULL;
}
//2.2.2 Phương thực thêm node mới vào Tree.
//Thêm node p vào cây có gốc là root.
void AddTree(ptrNode &root, ptrNode p)
{
     if (root == NULL)
          root = p;
     else
     {
          if (root->data.maDG > p->data.maDG)
               AddTree(root->left, p);
          else if (root->data.maDG < p->data.maDG)
               AddTree(root->right, p);
          else
          {
               //trường hợp trùng khóa.
               cout << "Doc gia da ton tai tren cay";
exit(1);
          }
     }
}
//2.2.3 Phương thức Nhập nhiều độc giả vào cây.
void InsertTree(TREE &tree)
{
     DOCGIA _docgia;
     char option;

     do
     {
          _docgia = NhapMotDocGia(tree.count+1);
```

```
          ptrNode p = GetNode(_docgia);
          AddTree(tree.root, p);
          tree.count++;
          cout << "Nhap tiep hay dung (y/n)"; cin >> option;
     } while (option!='y');
}
//2.2.4 In dữ liệu của cây.
//int theo NLR
void Print(ptrNode root)
{
     XuatMotDocGia( root->data);
     Print(root->left);
     Print(root->right);
}
//-------------End-----------------------------
//3. Các phương thức xử lý cây
//-------------BEGIN---------------------------
//3.1 tìm kiếm địa chỉ độc giả có tuổi lớn nhất.
//3.1.1 Tìm đọc giả có tuổi lớn nhất (theo năm)
void TuoiMax(ptrNode root, int &Max)
{
     if (root != NULL)
     {
          TuoiMax(root->left, Max);
          if (Max >= root->data.ngaySinh.year)
               Max = root->data.ngaySinh.year;
          TuoiMax(root->right, Max);
     }
}
//3.1.2 In địa chỉ của đọc giả có tuổi max.
void PrintDocGia(ptrNode root, int Max)
{
     if (root != NULL)
     {
          if (root->data.ngaySinh.year == Max)
               XuatMotDocGia(root->data);
          PrintDocGia(root->left, Max);
          PrintDocGia(root->right, Max);
     }
}
```

```
//3.1.3. Hàm tìm đọc giả
void TimDiaChiDocGia(ptrNode root)
{
     int Max = root->data.ngaySinh.year;
     TuoiMax(root, Max);
     cout << "Thong tin doc gia co tuoi lon nhat: \n";
     PrintDocGia(root, Max);
}
//-------------END-----------------------------
//4. Tính chiều cao của cây
int Max(int a, int b)
{
     return (a > b) ? a : b;
}
int TinhChieuCao(ptrNode root)
{
     if (root == NULL)
          return 0;
     else
     {
          return Max(TinhChieuCao(root->left),
TinhChieuCao(root->right))+1;
     }
}
//5. Đếm node 2 con có ngày làm thẻ là 9/5/2013
int SoSanhNgayThang(DATE _date1, DATE _date2)
{
     //0 bằng;
     //1: _date1>_date2;
     //-1 _date2>_date1
     if (_date1.year > _date2.year)
          return 1;
     else if (_date1.year<_date2.year)
          return -1;
     else//==
     {
          if (_date1.month > _date2.month)
               return 1;
          else if (_date1.month < _date2.month)
               return -1;
```

```
          else
          {
               if (_date1.day > _date2.day)
                    return 1;
               else if (_date1.day<_date2.day)
                    return -1;
               else
                    return  0;
          }
     }

}
//So sánh ngày tháng
int SoSanhNgay(DATE _date1, DATE _date2)
{
     int soNgayDate1, soNgayDate2;
     soNgayDate1 = _date1.day + (_date1.month - 1) * 30 +
(_date1.year - 1900) * 365;
     soNgayDate2 = _date2.day + (_date2.month - 1) * 30 +
(_date2.year - 1900) * 365;
     if (soNgayDate1 > soNgayDate2)
          return 1;
     else if (soNgayDate1 < soNgayDate2)
          return -1;
     else
          return 0;
}
//Đếm số node 2 con thỏa điều kiện.
int DemNodeHaiConThaoDieuKien(ptrNode root)
{
     int dem = 0;
     if (root == NULL)
     {
          return dem;
     }
     else
     {
          if (root->left != NULL&&root->right !=
NULL&&root->data.ngayLapThe.day == 9 &&
root->data.ngayLapThe.month == 5 &&
```

```
root->data.ngayLapThe.year == 2013)
                dem++;
           return dem + DemNodeHaiConThaoDieuKien(root->left)
+ DemNodeHaiConThaoDieuKien(root->right);
     }
}

void main()
{
     int option = 0;
     TREE tree;
     do
     {
          option = menu();
          switch (option)
          {
          case 1:
               Init(tree);
               break;
          case 2:
               break;
          }
     } while (option != 6);
}
```

Một số bài tập về cây nhị phân:

1. Viết chương trình chuyển đổi một cây nhị phân chứa số nguyên sang danh sách liên kết kép sao cho các nút chứa giá trị tăng dần
2. Nhập N ≤ 1000 số nguyên từ bàn phím. Viết chương trình thực hiện:
   a) Áp dụng thuật toán tạo cây nhị phân tìm kiếm để tạo cây từ dãy số trên.
   b) Viết thủ tục duyệt cây để kết quả in ra dãy số giảm dần
   c) Đếm số nút có bậc lẻ trong cây.
   d) Yêu cầu người dùng nhập vào giá trị x cần huỷ và huỷ nó ra khỏi cây.
3. Một cửa hàng quản lý các hàng hóa theo các thông tin sau:
   - Mã hàng: chuỗi gồm tối đa 5 kí tự
   - Tên hàng: chuỗi gồm tối đa 20 kí tự

- Số lượng tồn: số nguyên
- Đơn giá: số thực

Hãy định nghĩa và xây dựng cây nhị phân tìm kiếm để lưu trữ N ≤ 1000 sản phẩm hàng hóa với khóa là mã hàng. Lập trình cài đặt các thao tác nghiệp vụ sau:

a) Nhập đầy đủ thông tin cho N sản phẩm hàng hóa gồm: mã hàng, tên hàng, số lượng, đơn giá và lưu trữ vào cây.

b) Yêu cầu người dùng nhập vào mã hàng cần hủy và hãy hủy nó ra khỏi cây.

4. Một cửa hàng quản lý các hàng hóa theo các thông tin sau:
   - Mã hàng: chuỗi gồm tối đa 5 kí tự
   - Tên hàng: chuỗi gồm tối đa 20 kí tự
   - Số lượng tồn: số nguyên
   - Đơn giá: số thực

   Hãy định nghĩa và xây dựng cây nhị phân tìm kiếm để lưu trữ N ≤ 1000 sản phẩm hàng hóa với khóa là mã hàng. Lập trình cài đặt các thao tác nghiệp vụ sau:

   a) Nhập đầy đủ thông tin cho N sản phẩm hàng hóa gồm: mã hàng, tên hàng, số lượng, đơn giá và lưu trữ vào cây.

   b) Tính tổng trị giá của tất cả hàng hóa. Biết rằng trị giá = số lượng * đơn giá.