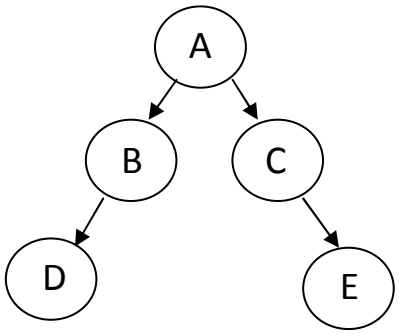
Ex. 1 以串列實作"二元樹"來儲存資料,
並且依前序/中序/後序進行深度優先搜尋.



資料結構

```
class node {
   String ^name;  //儲存字串
   node ^left, ^right;  //左邊子節點, 右邊子節點
};

   A->name = "A"
   A->left = B
   A->right= C

   B->name = "B"
   B->left = D
   B->right= nullptr

   C->name = "C"
   C->left = nullptr
   C->right= E

   D->name = "D"
   D->left = nullptr
   D->right= nullptr

   E->name = "E"
   E->left = nullptr
   E->right= nullptr
```

1. 前序搜尋 Preorder tree-traversal

訪問目前節點，左邊子節點，右邊子節點

```
Preorder_Traversal( node ^x ){
    if( x != nullptr ) {
        Visit( x );
        Preorder_Traversal( left );
        Preorder_Traversal( right );
    }
}
```

順序: ABDCE　　A(B(D))(C(E))

2. 中序搜尋 Inorder tree-traversal

訪問左邊子節點，目前節點，右邊子節點

```
Inorder_Traversal( node ^x ){
    if( x != nullptr ) {
        Inorder_Traversal( left );
        Visit( x );
        Inorder_Traversal( right );
    }
}
```

順序: DBACE　　((D)B)A(C(E))

3. 後序搜尋 Postorder tree-traversal

訪問左邊子節點，右邊子節點，目前節點

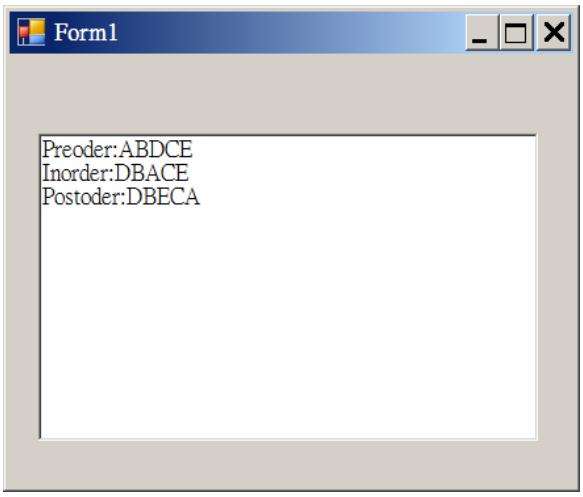
```
Postorder_Traversal( node ^x ){
    if( x != nullptr ) {
        Postorder_Traversal( left );
        Postorder_Traversal( right );
        Visit( x );
    }
}
```

順序: DBECA　　((D)B)((E)C)A

(程式碼)

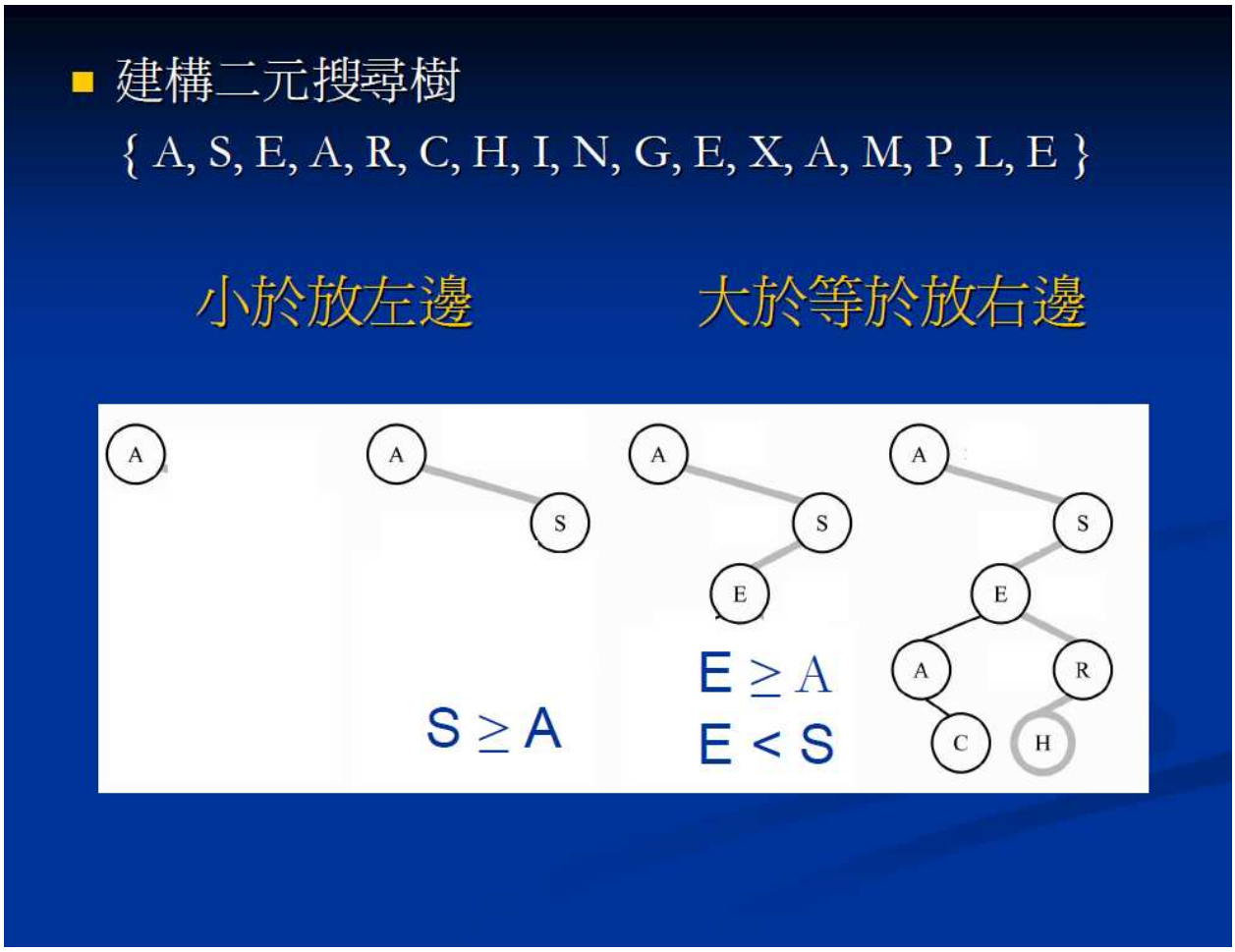
```
ref class node {
 public:
    String ^name;
    node   ^left, ^right;
    node( String ^s ){
        name=s;
        left=nullptr;
        right=nullptr;
    }
};
String^ Preorder_Traversal( node ^x ){
    String ^s;
    if( x != nullptr ) {
        s = x->name;
        s += Preorder_Traversal( x->left );
        s += Preorder_Traversal( x->right );
    } else s = "";
    return s;
}
String^ Inorder_Traversal( node ^x ){
    String ^s;
    if( x != nullptr ) {
        s  = Inorder_Traversal( x->left );
        s += x->name;
        s += Inorder_Traversal( x->right );
    } else s = "";
    return s;
}
String^ Postorder_Traversal( node ^x ){
    String ^s;
    if( x != nullptr ) {
        s = Postorder_Traversal( x->left );
        s += Postorder_Traversal( x->right );
        s += x->name;
    } else s = "";
    return s;
}
```

```
        node ^A, ^B, ^C, ^D, ^E;

private: System::Void Form1_Load(System::Object^  sender,
System::EventArgs^  e) {
           A=gcnew node("A");
           B=gcnew node("B");
           C=gcnew node("C");
           D=gcnew node("D");
           E=gcnew node("E");

           A->left = B;
           A->right = C;
           B->left = D;
           C->right = E;

           textBox1->Text  = "Preoder:"
                        + Preorder_Traversal( A ) + "\r\n";
           textBox1->Text += "Inorder:"
                        + Inorder_Traversal( A ) + "\r\n";
           textBox1->Text += "Postoder:"
                        + Postorder_Traversal( A ) + "\r\n";
        }
```



textBox1

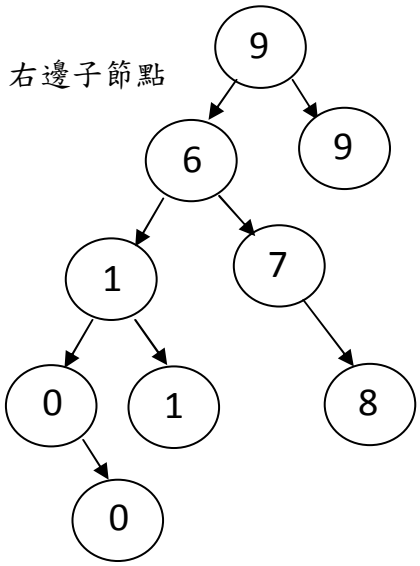
Ex. 2 實作二元搜尋樹，將任意陣列轉換為二元搜尋樹後，選擇其中一個陣列值進行搜尋。
(可參考演算法範例 7-3)



資料結構

```
class node {
    int  key;  //儲存鍵值
    String ^name; //儲存其他資料
    node ^left, ^right;  //左邊子節點，右邊子節點
};
```

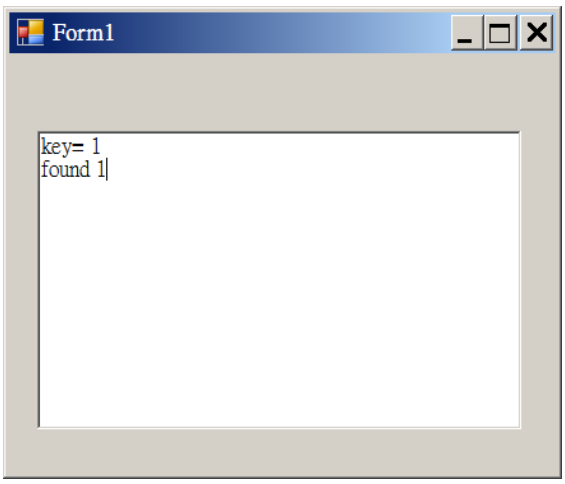
陣列： 9 6 1 0 1 0 7 8 9
二元搜尋樹



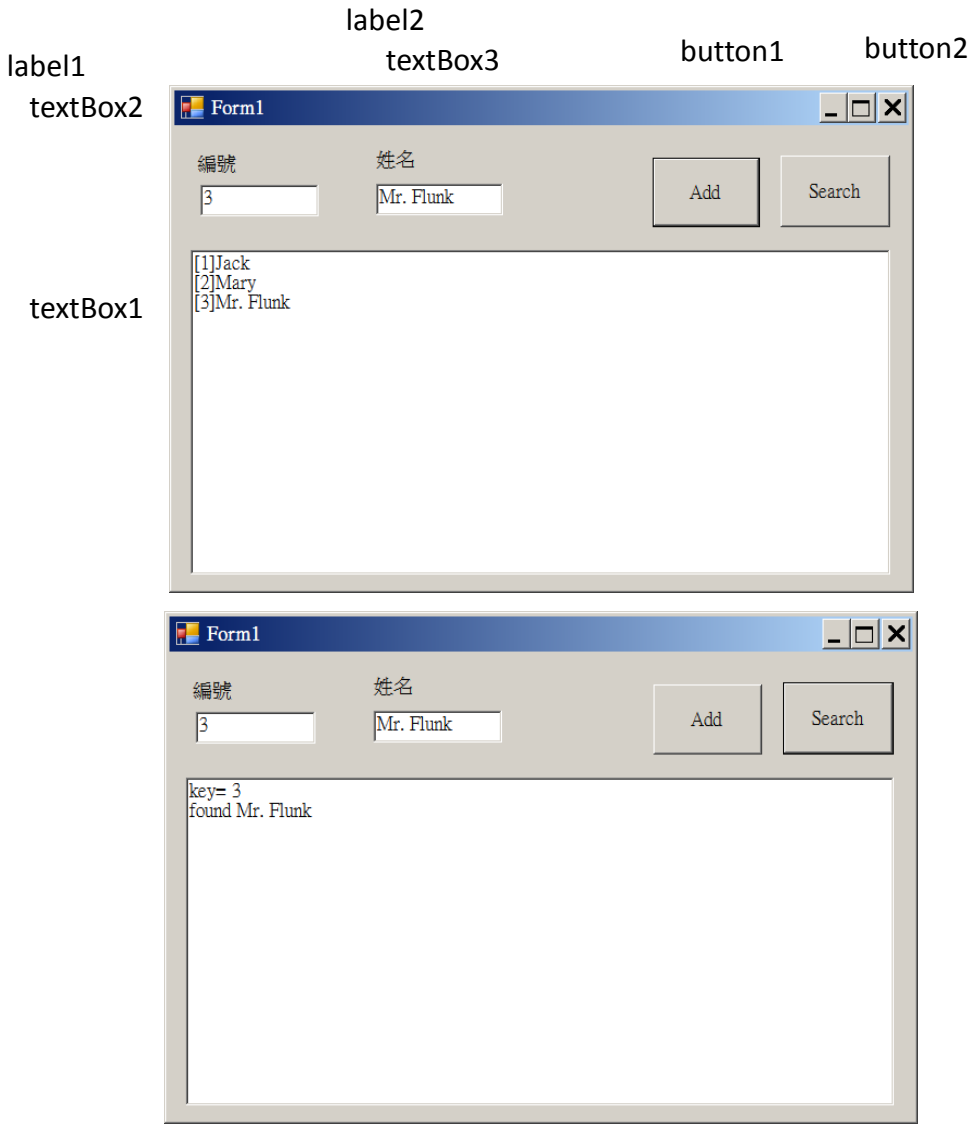
(程式碼)

```
ref class node {
 public:
   int    key;
   String ^name;
   node    ^left, ^right;
   node( int k, String ^s ){
      key=k;
      name=s;
      left=nullptr;
      right=nullptr;
   }
};

ref class BST { //Binary Search Tree
private:
   node ^root;
public:
   BST(){  root = nullptr; }

   void Add( int k, String ^s ){
      node ^n = gcnew node(k, s);

      if( root == nullptr ) root = n;
      else {
         node ^parent, ^x=root;

         while( x != nullptr) {
            parent = x;
            if ( k < x->key) x = x->left;
            else x = x->right;
         }
         if ( k < parent->key ) parent->left = n;
         else parent->right = n;
      }
   }
```

```
        String^ Search( int k ){
            node ^x = root;

            while( x != nullptr ){
                if( k == x->key ) return x->name;
                else if( k< x->key ) x = x->left;
                else x = x->right;
            }

            return "";
        }
    };

    BST ^A;

private: System::Void Form1_Load(System::Object^  sender,
                                 System::EventArgs^  e) {
        A=gcnew BST();

        int B[]={ 9, 6, 1, 0, 1, 0, 7, 8, 9 };
        int N = 9;

        int k;
        for(k=0; k<N; k++ )
            A->Add( B[k], B[k].ToString() );

        textBox1->Text = "key= " + B[ N/2 ] + "\r\n";

        String ^s = A->Search( B[ N/2 ] );

        if( s!="" )
            textBox1->Text += "found " + s;
        else
            textBox1->Text += "not found ";
    }
```


Form1
key= 1
found 1

Ex. 3 接 Ex. 2 實作二元搜尋樹，將輸入的資料儲存在二元搜尋樹內，並進行搜尋。



(程式碼)

```
ref class node {
public:
    int key;
    String ^name;
    node ^left, ^right;
    node( int k, String ^s ){
        key=k;   name=s;
        left=nullptr;
        right=nullptr;
    }
};
```

```
ref class BST { //Binary Search Tree
private:
    node ^root;
public:
    BST(){ root = nullptr; }
    void Add( int k, String ^s ){
        node ^n = gcnew node(k, s);
        if( root == nullptr ) root = n;
        else {
            node ^parent, ^x=root;
            while( x != nullptr) {
                parent = x;
                if ( k < x->key) x = x->left;
                else x = x->right;
            }
            if ( k < parent->key ) parent->left = n;
            else parent->right = n;
        }
    }
    String^ Search( int k ){
        node ^x = root;
        while( x != nullptr ){
            if( k == x->key ) return x->name;
            else if( k< x->key ) x = x->left;
            else x = x->right;
        }
        return "";
    }
    String^ Preorder( node ^x ){
        String ^s;
        if( x != nullptr ) {
            s = "[" + x->key + "]" + x->name + "\r\n";
            s += Preorder( x->left );
            s += Preorder( x->right );
        } else s = "";
        return s;
    }
```

```
        String^ Show(){
            return Preorder( root );
        }
    };
    BST ^A;

private: System::Void Form1_Load(System::Object^
                                                sender,
            System::EventArgs^ e) {
                A=gcnew BST();
        }
private: System::Void button1_Click(System::Object^
                        sender, System::EventArgs^  e) {
            int x;
            if( int::TryParse(textBox2->Text, x) ){
                A->Add( x, textBox3->Text );
            }
            textBox1->Text = A->Show();
        }

private: System::Void button2_Click(System::Object^
                        sender, System::EventArgs^  e) {
        int x;
        if( int::TryParse(textBox2->Text, x) ){
               textBox1->Text = "key= " + x + "\r\n";
               String ^s = A->Search( x );
               if( s!="" )
                  textBox1->Text += "found " + s;
               else
                  textBox1->Text += "not found ";
        }
    }
```