BÀI TẬP CẤU TRÚC DỮ LIỆU

TS. Lê Xuân Trường

BUỔI 1 - Chương 1: DANH SÁCH

Bài 1. Quản lý một danh sách có tối đa 100 phần tử, mỗi phần tử trong danh sách có kiểu int. (danh sách không có thứ tự)

a. Khai báo cấu trúc danh sách

b. Viết thủ tục nhập danh sách.

c. Viết thủ tục xuất danh sách ra màn hình

d. Viết thủ tục tìm một phần tử trong danh sách. Tính độ phức tạp của thuật toán

e. Viết thủ tục thêm một phần tử vào cuối danh sách.

f. Viết thủ tục xoá phần tử cuối danh sách.

g. Viết thủ tục xoá phàn tử tại vị trí thứ i trong danh sách. Tính độ phức tạp của thuật toán

h. Viết thủ tục thêm một phần tử vào danh sách tại vị trí i. Tính độ phức tạp của thuật toán.

i. Tìm một phần tử trong danh sách. Nếu tìm thấy, xoá phần tử đó. Tính độ phức tạp của thuật toán.

BUỔI 1 - Chương 1: DANH SÁCH

a.

```
#define max 100
int a[max];
int n;
```

b.

```
void nhap()
{
 cin>>n;

 for(int i=0;i<n;i++)
 {
     cout<<"a["<<i<<"]=";
     cin>>a[i];
  }
}
```

BUỔI 1 - Chương 1: DANH SÁCH

c. void xuat()

```
{
  for(int i=0;i<n;i++)
      cout<<a[i]<<endl;
}
```

d. int void tim(int x)

```
{ // x là giá trị cần tìm
   int i=0;
  while ((i<n)&&(a[i]!=x)) i++;
  if(i==n) return -1;
  else return i;
}
```

BUỔI 1 - Chương 1: DANH SÁCH

Bài 2: Quản lý một danh sách *có thứ tự* có tối đa 100 phần tử, mỗi phần tử trong danh sách có kiểu int.

a. Khai báo cấu trúc danh sách

b. Viết thủ tục thêm một phần tử vào danh sách

c. Viết thủ tục xuất các phần tử trong danh sách

d. Viết thủ tục tìm một phần tử trong danh sách (dùng phương pháp tìm kiếm tuần tự). Đánh giá độ phức tạp của thuật toán.

e. Viết thủ tục tìm một phần tử trong danh sách (dùng phương pháp tìm kiếm nhị phân). Đánh giá độ phức tạp của thuật toán.

f. Viết thủ tục tìm một phần tử trong danh sách. Nếu tìm thấy, xoá phần tử này.

BUỔI 1 - Chương 1: DANH SÁCH

Bài 3: Quản lý một stack có tối đa 100 phần tử, mỗi phần tử trong stack có kiểu int.

a. Khai báo cấu trúc stack

b. Viết thủ tục khởi tạo stack rỗng

c. Viết thủ tục xét stack rỗng

d. Viết thủ tục xét stack đầy

e. Viết thủ tục thêm một phần tử vào stack

f. Viết thủ tục xoá một phần tử trong stack

BUỔI 1 - Chương 1: DANH SÁCH

Bài 3:

a.

```
#define       max 100
int a[max];
int sp;
```

b.

```
void stack_empty()
{
   sp=-1;
}
```

BUỔI 1 - Chương 1: DANH SÁCH

Bài 3:

c.
```
void push(int x)
{
   if(sp<max-1)
      a[++sp]=x;
}
```

d.
```
int pop(int &x)
{
   if(sp!=-1)
   {
      x=a[sp--];
      return 1;
   }
  return 0;
}
```

BUỔI 2 - Chương 1: DANH SÁCH

Bài 4: Sử dụng stack đã xây dựng, đổi một số hệ thập phân sang hệ nhị phân.

HD: - Lấy số thập phân chia cho 2, phần dư đưa vào stack.

- Lấy thương chia tiếp cho 2, phần dư đưa vào stack. Lặp lại bước này cho đến khi thương bằng 0.
- Lấy các phần tử trong stack là số nhị phân-kết quả của việc chuyển đổi.

Bài 5: Sử dụng stack đã xây dựng, đổi một số hệ thập phân sang hệ bất kỳ.

Bài 6: Sử dụng stack đã xây dựng, giải bài toán Tháp Hà nội. (*)

BUỔI 2 - Chương 1: DANH SÁCH

Bài 7: Quản lý một queue có tối đa 100 phần tử, mỗi phần tử trong queue có kiểu int

a. Khai báo cấu trúc queue

b. Viết thủ tục khởi tạo queue rỗng

c. Viết thủ tục xét queue rỗng

d. Viết thủ tục xét queue đầy

e. Viết thủ tục thêm một phần tử vào queue

f. Viết thủ tục xoá một phần tử trong queue

BUỔI 2 - Chương 1: DANH SÁCH

Bài 7:

a.

```
#define max 100
int a[max];
int front,rear;
//front la vi tri loai bo
//rear la vi tri them vao
```

b.

```
void queue_empty()
  {
    front=-1;
    rear=-1;
  }
```

BUỔI 2 - Chương 1: DANH SÁCH

Bài 7:

```
e. int insert_queue(int x)
    {
       if(rear-front==max-1 )
             return -1;
       else
           {
              if (front==-1)     //hang rong
                   front=0;  // rear=-1
              if(rear==max-1)  //doi tinh tien
                 {
                    for(int i=front;i<=rear; i++)
                        a[i-front]=a[i];
                    rear = max -1-front ;
                    front =0;
                  }
               a[++rear]=x;
              return rear;
           }
}
```

BUỔI 2 - Chương 1: DANH SÁCH

Bài 7:

f.
```
int delete_queue(int &x)
 {
    if(front==-1)
          return 0;
    else
     {
         x=a[front++];
          if(front>rear)
            {
                front=-1;
                rear=-1;
             }
         return 1;
     }
}
```

BUỔI 2 - Chương 1: DANH SÁCH

Bài 8: Quản lý một danh sách có số phần tử khá lớn, biến động. Mỗi phần tử có kiểu int. (Dùng cấu trúc Danh sách liên kết)

a. Khai báo cấu trúc danh sách.

b. Viết thủ tục khởi tạo danh sách rỗng.

c. Viết thủ tục xuất các phần tử trong danh sách.

d. Viết thủ tục tìm một phần tử trong danh sách.

e. Viết thủ tục thêm một phần tử vào đầu danh sách.

f. Viết thủ tục xoá phần tử đầu danh sách.

g. Viết thủ tục thêm một phần tử vào cuối danh sách.

h. Viết thủ tục xoá phần tử cuối danh sách.

i. Viết thủ tục tìm một phần tử trong danh sách. Nếu tìm thấy, hãy xoá phần tử này.

j. Từ danh sách trên chuyển thành danh sách có thứ tự. (*)

BUỔI 2 - Chương 1: DANH SÁCH

Bài 8:

a.

```
struct Node
{
  int info;
 Node *link;
};
Node *first;
```

b.

```
void list_empty()
{
  first = NULL;
}
```

BUỔI 2 - Chương 1: DANH SÁCH

Bài 8:

c.

```
void process_list()
{
   Node *p;
   p=first;
   while (p!=NULL)
    {
      cout<<p->info<<endl;
      p=p->link;
    }
}
```

BUỔI 2 - Chương 1: DANH SÁCH

Bài 8:

d.

```
Node *search(int x)
 {
    Node *p=first;
    while ((p!=NULL)&&(p->info!=x))
     p=p->link;
    return p;
}
```

BUỔI 2 - Chương 1: DANH SÁCH

Bài 8:

e.

```
void insert_first(int x)
{

  Node *p;
  p = new Node;
  p->info=x;
  p->link=first;
  first=p;
}
```

BUỔI 2 - Chương 1: DANH SÁCH

Bài 8:

f. int delete_first()

```
{
    if(first==NULL)
        return 0;
    else
      {
       Node *p;
       p=first;
       first=p->link;
       delete p;
      return 1;
      }
}
```

BUỔI 2 - Chương 1: DANH SÁCH

Bài 8:

g.
```
void insert_last(int x)
    {
      Node *p;
      p = new Node;
      p->info=x;
      p->link=NULL;
      if(first==NULL)
        first=p;
      else
       {
        Node *q=first;
        while(q->link!=NULL)
              q=q->link;
        q->link=p;
       }
   }
```

BUỔI 2 - Chương 1: DANH SÁCH

Bài 8:

h.
```
int delete_last()
    {
      if(first==NULL)
          return 0;
      else
        {
          Node *p,*q;
          p=first;
          while(p->link!=NULL)
             {
               q=p;
               p=p->link;
             }
         q->link=NULL;
         delete p;
        return 1;
    }
}
```

BUỔI 3 - Chương 1: DANH SÁCH

Bài 9: Quản lý một danh sách *có thứ tự* có số phần tử khá lớn, biến động. Mỗi phần tử có kiểu int. (danh sách liên kết đơn)

a. Khai báo cấu trúc danh sách

b. Viết thủ tục khởi tạo danh sách rỗng

c. Viết thủ tục thêm một phần tử vào danh sách

d. Viết thủ tục xuất các phần tử trong danh sách

e. Viết thủ tục tìm một phần tử trong danh sách.

f. Viết thủ tục tìm một phần tử trong danh sách. Nếu tìm thấy, xoá một phần tử này.

BUỔI 3 - Chương 1: DANH SÁCH

Bài 10: Quản lý một stack có số phần tử khá lớn, biến động. Mỗi phần tử có kiểu int (danh sách liên kết đơn)

a. Khai báo cấu trúc stack

b. Viết thủ tục khởi tạo stack rỗng

c. Viết thủ tục xét stack rỗng

d. Viết thủ tục thêm một phần tử vào stack

e. Viết thủ tục xoá một phần tử trong stack

f. Áp dụng stack đã xây dựng, đổi số hệ thập sang hệ nhị phân

g. Áp dụng stack đã xây dựng, giải bài toán tháp Hà nội.(*)

BUỔI 3 - Chương 1: DANH SÁCH

Bài 10:

a.
```
struct Node
{
    int info;
    Node *link;
};

Node *sp;
```

b.
```
void stack_empty()
{
    sp = NULL;
}
```

BUỔI 3 - Chương 1: DANH SÁCH

Bài 10:

c.

```
void push(int x)
 {
        Node *p;
        p = new Node;
        p->info=x;
        p->link=sp;
        sp=p;
}
```

BUỔI 3 - Chương 1: DANH SÁCH

Bài 10:

d.

```
int pop(int &x)
 {
      if(sp==NULL)
            return 0;
      else
        {
            x=sp->info;
            Node *p;
            p=sp;
            sp=p->link;
            delete p;
            return 1;
        }
 }
```

BUỔI 3 - Chương 1: DANH SÁCH

Bài 11: Quản lý một queue có số phần tử khá lớn, biến động. Mỗi phần tử có kiểu int (danh sách liên kết đơn)

a. Khai báo cấu trúc queue

b. Viết thủ tục khởi tạo queue rỗng

c. Viết thủ tục xét queue rỗng

d. Viết thủ tục thêm một phần tử vào queue

e. Viết thủ tục xoá một phần tử trong queue

BUỔI 3 - Chương 1: DANH SÁCH

Bài 11:

a.
```
struct Node
    {
        int info;
        Node *link;
     };
     Node *front, *rear;
```

b.
```
void queue_empty()
    {
        front = NULL;
        rear = NULL;
    }
```

BUỔI 3 - Chương 1: DANH SÁCH

Bài 11:

c.
```
void push(int x)
    {
        Node *p;
        p = new Node;
        p->info=x;
        p->link=NULL;
        if (rear==NULL)
                front= p;
        else
                rear->link=p;
        rear=p;
     }
```

BUỔI 3 - Chương 1: DANH SÁCH

Bài 11:

d.
```
int pop(int &x)
    {
        if(front!=NULL)
            {
                Node *p=front;
                front=p->link;
                x=p->info;
                if(front==NULL)
                        rear = NULL;
                delete p;
                return 1;
            }
        else
                return -1;
    }
```

BUỔI 4 - Chương 1: DANH SÁCH

Bài 12: Quản lý một danh sách có số phần tử khá lớn, biến động. Mỗi phần tử có kiểu int. Thường có nhu cầu truy suất phần tử đứng trước và phần tử đứng sau phần tử đang truy xuất. (Danh sách liên kết kép)

a. Khai báo cấu trúc danh sách

b. Viết thủ tục khai báo danh sách rỗng

c. Xuất các phần tử trong danh sách.

d. Viết thủ tục thêm phần tử vào đầu danh sách

e. Viết thủ tục thêm một phần tử vào cuối danh sách.

f. Viết thủ tục xoá phần tử đầu danh sách

g. Viết thủ tục xoá phần tử cuối danh sách

h. Viết thủ tục tìm một phần tử trong danh sách. Nếu tìm thấy, xoá phần tử này

i. Viết thủ tục tìm một phần tử có giá trị bằng với giá trị x hoặc gần nhất và lớn hơn phần tử x nhập vào; Thêm một phần tử trước phần tử tìm thấy.

BUỔI 4 - Chương 1: DANH SÁCH

Bài 13. Quản lý một danh sách liên kết vòng.

a. Khai báo cấu trúc danh sách

b. Viết thủ tục khởi tạo danh sách rỗng

c. Viết thủ tục xuất các phần tử trong danh sách

d. Viết thủ tục thêm một phần tử vào đầu danh sách

e. Viết thủ tục xoá một phần tử trong đầu danh sách

f. Viết thủ tục thêm một phần tử vào cuối danh sách

g. Viết thủ tục xoá một phần tử trong cuối danh sách

h. Viết thủ tục tìm một phần tử trong danh sách. Nếu tìm thấy, xoá phần tử này.

BUỔI 4- Chương 1: DANH SÁCH

Bài 14: Dùng cấu trúc danh sách đặc quản lý một đa thức

a. Khai báo cấu trúc danh sách.

b. Viết thủ tục nhập đa thức

c. Viết thủ tục xuất đa thức

d. Viết thủ tục cộng hai đa thức

e. Viết thủ tục trừ hai đa thức

f. Viết thủ tục nhân hai đa thức

g. Viết thủ tục chia hai đa thức

BUỔI 4 - Chương 1: DANH SÁCH

Bài 15: Dùng cấu trúc danh sách liên kết quản lý một đa thức

a. Khai báo cấu trúc danh sách.

b. Viết thủ tục nhập đa thức

c. Viết thủ tục xuất đa thức

d. Viết thủ tục cộng hai đa thức

e. Viết thủ tục trừ hai đa thức

f. Viết thủ tục nhân hai đa thức

g. Viết thủ tục chia hai đa thức

BUỔI 4- Chương 1: DANH SÁCH

Bài 16: Dùng cấu trúc danh sách đặc quản lý tập hợp các phần tử số int

a. Viết thủ tục tìm tập hợp là hợp của hai danh sách

b. Viết thủ tục là tập hợp giao của hai danh sách

c. Viết thủ tục xuất danh sách chứa phần bù của hai danh sách

Bài 17: Dùng cấu trúc danh sách liên kết quản lý tập hợp các phần tử số int

a. Viết thủ tục tìm tập hợp là hợp của hai danh sách

b. Viết thủ tục là tập hợp giao của hai danh sách

c. Viết thủ tục xuất danh sách chứa phần bù của hai danh sách

BUỔI 4 - Chương 1: DANH SÁCH

Bài 18: Xây dựng chương trình xử lý văn bản với các chức năng nhập, mở, lưu,… văn bản

BUỔI 5 - Chương 2: XẾP THỨ TỰ - TÌM KIẾM

Bài 1: Quản lý một danh sách đăc 100 phần tử kiểu int.

a. Khai báo cấu trúc danh sách

b. Viết thủ tục nhập danh sách

c. Viết thủ tục xuất danh sách

d. Viết thủ tục xếp danh sách trên dùng phương pháp InsertionSort. Đánh giá độ phức tạp của thuật toán

e. Viết thủ tục xếp danh sách trên dùng phương pháp SelectionSort. Đánh giá độ phức tạp của thuật toán

f. Viết thủ tục xếp danh sách trên dùng phương pháp InterchangeSort. Đánh giá độ phức tạp của thuật toán

BUỔI 5 - Chương 2: XẾP THỨ TỰ - TÌM KIẾM

Bài 1: Quản lý một danh sách đăc 100 phần tử kiểu int.

d. InsertionSort

Bước 1: i = 1; //đoạn a[0] đã được xếp thứ tự

Bước 2: x = a[i]; Tìm vị trí pos thích hợp trong đoạn a[0] đến a[i-1] để chèn a[i] vào

Bước 3: Dời chỗ các phần tử từ a[pos] đến a[i-1] sang phải 1 vị trí để dành chổ cho a[i]

Bước 4: a[pos] = x; //có đoạn a[0]..a[i] đã được sắp

Bước 5: i = i+1;

Nếu i < n : Lặp lại Bước 2

BUỔI 5 - Chương 2: XẾP THỨ TỰ - TÌM KIẾM

Bài 1: Quản lý một danh sách đăc 100 phần tử kiểu int.

e. SelectionSort

Bước 1: i = 0;

Bước 2: Tìm phần tử **a[min]** nhỏ nhất trong dãy hiện hành từ a[i] đến a[n]

Bước 3 : Đổi chỗ a[min] và a[i]

Bước 4 : i = i+1

Nếu i < N-1 thì lặp lại Bước 2;

BUỔI 5 - Chương 2: XẾP THỨ TỰ - TÌM KIẾM

Bài 1: Quản lý một danh sách đăc 100 phần tử kiểu int.

f. InterchangeSort

Bước 1: i = 0;

Bước 2: j = i+1;

Bước 3:

```
Trong khi j < N thực hiện
        Nếu a[j]<a[i]
                Swap(a[i],a[j]);
    j = j+1;
```

Bước 4: i = i+1;

Nếu i < N-1 lặp lại Bước 2.

BUỔI 5 - Chương 2: XẾP THỨ TỰ - TÌM KIẾM

Bài 2:

a. Viết thủ tục xếp danh sách trên dùng phương pháp BubbleSort. Đánh giá độ phức tạp của thuật toán

b. Viết thủ tục tìm một phần tử trong danh sách có thứ tự (Dùng phương pháp tìm kiếm tuần tự). Đánh giá độ phứp tạp của thuật toán.

c. Viết thủ tục tìm một phần tử trong danh sách có thứ tự (Dùng phương pháp tìm kiếm nhị phân). Đánh giá độ phứp tạp của thuật toán.

BUỔI 5 - Chương 2: XẾP THỨ TỰ - TÌM KIẾM

Bài 2:

a. BubbleSort.

Bước 1 : i = 0;

Bước 2 : j = N-1;

Trong khi (j > i) thực hiện:

Nếu a[j]<a[j-1]

swap(a[j],a[j-1]);

j = j-1;

Bước 3 : i = i+1;

Nếu i <N-1thì lặp lại Bước 2.

BUỔI 5 - Chương 2: XẾP THỨ TỰ - TÌM KIẾM

Bài 2:

b. Tìm tuần tự

```
int LinearSearch(int a[],int n, int x)
        {
                int i=0;
                while((i<n)&&(a[i]!=x))
                        i++;
                if(i<n) return i; //tìm thấy tại vị trí i
                  return  -1;
        }
```

BUỔI 6 - Chương 2: XẾP THỨ TỰ - TÌM KIẾM

Bài 2:

c. Tìm kiếm nhị phân

```
int BinarySearch(int a[],int n,int x)
      {   int left, right, mid; left=0; right=n-1;
          do{
                        mid=(left+right)/2;
                        if(a[mid]==x) return 1;
                        else    if(a[mid]<x)  left=mid+1;
                                else  right=mid-1;
          }while(left<=right);
          return 0;
      }
```

BUỔI 6- Chương 3: CÂY NHỊ PHÂN TÌM KIẾM

Bài 1: Quản lý một cây nhị phân, mỗi phần tử có kiểu int.

a. Khai báo cấu trúc cây nhị phân

b. Viết thủ tục khởi tạo cây rỗng

c. Viết thủ thục thêm một phần tử vào cây.

d. Viết thủ tục tìm một phần tử trong cây.

e. Viết thủ tục xoá một nút trong cây

f. Viết thủ tục duyệt cây theo thứ tự NLR (dùng đệ quy)

g. Viết thủ tục duyệt cây theo thứ tự LNR (dùng đệ quy)

h. Viết thủ tục duyệt cây theo thứ tự LRN (dùng đệ quy)

BUỔI 6- Chương 3: CÂY NHỊ PHÂN TÌM KIẾM

Bài 1:

a.
```
struct Node
    {
        int info;
        Node *left, *right;
    };
    typedef Node *StNODE;
   Node *root;
```

b.
```
void tree_empty()
    {
       root = NULL;
    }
```

BUỔI 6- Chương 3: CÂY NHỊ PHÂN TÌM KIẾM

Bài 1:

c.
```
int InsertNode(StNODE &p,int x)
    {
        if(p)
        {
            if(p->info==x) return 0; //da co
            else
                if(p->info>x)
                    return InsertNode(p->left,x);
                else
                    return InsertNode(p->right,x);
        }
        p=new Node;
        if(p==NULL) return -1;
        p->info=x;
        p->left=NULL;
        p->right=NULL;
        return 1;
    }
```

BUỔI 6 - Chương 3: CÂY NHỊ PHÂN TÌM KIẾM

Bài 1:

d.

```
Node * SearchNode(Node *p,int x)
 {
        if(p) {
                if(p->info==x) return p;
                if(p->info>x) return SearchNode(p->left,x);
                        else return SearchNode(p->right,x);
        }
        return NULL;
 }
```

BUỔI 6 - Chương 3: CÂY NHỊ PHÂN TÌM KIẾM

Bài 1:

e.

```
void searchStandFor(StNODE &p, StNODE &q)
 {
        if(q->left)
                searchStandFor(p,q->left);
        else
        {
                p->info=q->info;
                p=q;
                q=q->right;
        }
 }
```

BUỔI 6 - Chương 3: CÂY NHỊ PHÂN TÌM KIẾM

```
int delNode(StNODE &T, int x)
  {
        if(T==NULL) return 0;
        if(T->info>x)
              return delNode(T->left,x);
        if(T->info<x)
              return delNode(T->right,x);
        else
        {
              Node *p =T;
              if(T->left==NULL)
                    T=T->right;
              else
                    if(T->right==NULL)
                          T=T->left;
                    else
                    { //co ca 2 con
                          Node *q =T->right;
                          searchStandFor(p,q);
                    }
              delete p;
              return 1;
        }
  }
```

BUỔI 6 - Chương 3: CÂY NHỊ PHÂN TÌM KIẾM

f.

```
void NLR(Node *p)
 {
        if(p!=NULL)
        {
                cout<<p->info<<endl;
                NLR(p->left);
                NLR(p->right);
        }
}
```

BUỔI 6 - Chương 3: CÂY NHỊ PHÂN TÌM KIẾM

g.

```
void LNR(Node *p)
  {
        if(p!=NULL)
        {
              LNR(p->left);
              cout<<p->info<<endl;
              LNR(p->right);

        }
 }
```

BUỔI 6 - Chương 3: CÂY NHỊ PHÂN TÌM KIẾM

h.

```
void LRN(Node *p)
 {
        if(p!=NULL)
        {
                LRN(p->left);
                LRN(p->right);
                cout<<p->info<<endl;

        }
}
```

BUỔI 7 - Chương 4: BẢNG BĂM

Bài 1: Dùng cấu trúc bảng băm – phương pháp kết nối trực tiếp, quản lý 5000 phần từ kiểu số int.

a. Khai báo cấu trúc bảng băm

b. Viết thủ tục khởi bảng băm rỗng

c. Viết thủ tục thêm một phần tử vào bảng băm

d. Viết thủ tục tìm kiếm một phần tử trong bảng băm

BUỔI 7 - Chương 4: BẢNG BĂM

Bài 1:

a.

```
#define M 501

struct Node
{
    int key;
    Node *next;
};
Node *heads[M];
Node *z;
```

BUỔI 7 - Chương 4: BẢNG BĂM

Bài 1:

b.

```
void init()
 {
    z=new Node;
    z->next=z;
    for(int i=0;i<M;i++)
      {
          heads[i]=new Node;
          heads[i]->next=z;
      }
 }
```

BUỔI 7 - Chương 4: BẢNG BĂM

Bài 1:

c.

```
Node *Insert(int k, Node *t)
 {
    Node *x,*p;
    z->key=k;
    p=t->next;
    while (p->key<k)
     {
        t=t->next;
        p=t->next;
      }
    x=new Node;
    x->next=p;
    t->next=x;
    x->key=k;
    return x;
}
```

BUỔI 7 - Chương 4: BẢNG BĂM

Bài 1:

d.

```
Node *search(int k, Node *t)
  {
      z->key=k;
      do {            {
            t=t->next;
       }while (t->key<k);
      if (k==t->key)
          return t;
      else
          return z;
  }
```

BUỔI 8 - Chương 4: BẢNG BĂM

Bài 2: Quản lý một tự điển Anh – Việt gồm khoảng 50.000 từ

a. Khai báo cấu trúc tự điển
b. Viết thủ tục tìm kiếm một từ
c. Viết thủ tục load các từ từ file lên bộ nhớ
d. Viết thủ thêm một từ mới

BUỔI 9 – ÔN TẬP VÀ DỰ TRỮ

(-- Hết --)