บทที่ 4

การพัฒนาโปรแกรมสำหรับระบบ

1. ลักษณะของภาษาซี

ในหัวข้อนี้จะกล่าวถึง ลักษณะเฉพาะบางประการของภาษาซี ที่เกี่ยวข้องกับการพัฒนาโปรแกรมสำหรับระบบการเก็บและการค้นคืนสารสนเทศ

ในภาษาซีแบ่งตัวแปรออกเป็นชนิดต่างๆ เช่น char หมายถึงตัวแปรชนิดตัวอักษร int หมายถึงตัวแปรชนิดเลขจำนวนเต็มปกติ long หมายถึงตัวแปรชนิดเลขจำนวนเต็มที่มีความยาวเป็น 2 เท่า และสตริง (string) หมายถึงอะเรย์ของตัวแปรแบบ char

นอกจากตัวแปรแบบต่างๆ ที่กล่าวมาแล้ว ตัวแปรที่สำคัญอีกแบบหนึ่งในภาษาซีคือ ตัวแปรแบบพอยน์เตอร์ (pointer) ตัวแปรแบบพอยน์เตอร์ในภาษาซี จะทำหน้าที่ชี้ข้อมูล โดยทำหน้าที่ชี้หน่วยความจำโดยตรง ดังนั้นค่าในตัวแปรพอยน์เตอร์ จึงเป็นค่าแอดเดรสต้นของข้อมูล ในการกำหนดตัวแปรแบบพอยน์เตอร์ จะใช้สัญญลักษณ์ * และจะต้องบ่งด้วยว่าจะให้ชี้ที่ข้อมูลแบบใด เช่น int *v เป็นการกำหนดตัวแปร v ให้เป็นตัวแปรแบบพอยน์เตอร์สำหรับชี้ข้อมูลแบบเลขจำนวนเต็ม ซึ่งหมายถึง เป็นการกำหนดเนื้อที่ในหน่วยความจำและให้ชื่อว่า v สำหรับเก็บแอดเดรสของข้อมูลแบบเลขจำนวนเต็มที่เราจะกำหนดต่อไปในโปรแกรม เช่น กำหนดให้ onew เป็นตัวแปรแบบเลขจำนวนเต็ม เราจะได้ว่า v = &onew โดยที่สัญญลักษณ์ & หมายถึงแอดเดรสของตัวแปร onew ในการอ้างถึงตัวแปรแบบพอยน์เตอร์ เราจะใช้ v เมื่อต้องการอ้างถึงแอดเดรสของข้อมูล และ *v เมื่อต้องการอ้างถึงตัวข้อมูล นั่นคือ

```
        v = &onew
หมายถึง onew    แทนข้อมูล
        &onew   ชี้หน่วยความจำ
        v       ชี้หน่วยความจำ
        *v      แทนข้อมูล
        &v      ให้ค่าแอดเดรสของหน่วยความจำที่เป็นตัวแปรแบบพอยน์เตอร์
```

นอกจากนี้ในภาษาซียังมีข้อมูลชนิดโครงสร้าง (structure) ซึ่งเป็นชนิดของข้อมูลที่รวบรวมข้อมูลชนิดต่างๆ เข้าด้วยกัน การกำหนดตัวแปรแบบโครงสร้าง จะต้องเริ่มต้นด้วยคำว่า struct เช่น

```
struct   Directory_File
{        char  fname[8];
         char  desc[80];
};
```

เป็นการกำหนดว่า Directory_File เป็นตัวแปรแบบโครงสร้าง ประกอบด้วยสมาชิกคือ fname เป็นสตริงขนาด 8 ตัวอักษร และ desc เป็นสตริงขนาด 80 ตัวอักษร การอ้างถึงสมาชิกแต่ละตัวของข้อมูลชนิดโครงสร้าง มีรูปแบบดังนี้ คือ

ชื่อตัวแปร.ชื่อสมาชิก

และถือว่าเป็นตัวแปรหนึ่งตัว มีชนิดตามแบบสมาชิก เช่น

direct.fname เป็นตัวแปรแบบสตริงขนาด 8 ตัวอักษร
เมื่อ direct เป็นตัวแปรแบบ struct Directory_File

นอกจากนี้ในภาษาซี เราสามารถกำหนดชื่อใหม่ แทนชื่อชนิดของข้อมูลเดิมได้ โดยใช้ typedef เช่น เราต้องการกำหนดชื่อ DFIL ให้ใช้แทน struct Directory_File เราสามารถกำหนดได้ดังนี้

```
typedef  struct  Directory_File  DFIL;
```

ในการพัฒนาโปรแกรมสำหรับระบบ จะแบ่งการทำงานของโปรแกรมออกเป็นส่วนย่อยๆ ซึ่งในภาษาซีเรียกว่า ฟังก์ชัน (function) การทำงานของฟังก์ชันที่มีการส่งค่ากลับ (return value) จะต้องกำหนดชนิดของฟังก์ชันให้สอดคล้องกับค่าที่ส่งกลับ เช่น

int openfile() หมายถึงฟังก์ชัน openfile มีการส่งค่ากลับเป็น int

ศูนย์วิทยทรัพยากร
จุฬาลงกรณ์มหาวิทยาลัย

1170087X3

2. การกำหนดตัวแปร

ตัวแปรที่สำคัญ และระเบียนของแฟ้มข้อมูลที่เกี่ยวข้อง ในการพัฒนาโปรแกรมของระบบการเก็บและการค้นคืนสารสนเทศ มีดังต่อไปนี้ คือ

```
struct   Text_Index
{        long  docadd;
         char  title[MAX];
};
typedef  struct  Text_Index  INDX;

struct   Link_List
{        long begin;
         int  docno;
         int  parano;
         int  wordno;
         long next;
};
typedef  struct  Link_List  LINK;

struct   node
{        int  cnt;
         char key[MM][MAXWORD];
         int  occur[MM];
         long list[MM];
         long ptr[MM+1];
};
typedef  struct node  NODE;

NODE  rootnode;
long  root;
```

ศูนย์วิทยทรัพยากร
จุฬาลงกรณ์มหาวิทยาลัย

MAXWORD เป็นจำนวนตัวอักษรสูงสุดที่เก็บได้ในแต่ละคำ

MAX เป็นจำนวนตัวอักษรสูงสุดใน 1 บรรทัด

M เป็นจำนวนคีย์ที่น้อยที่สุดในแต่ละโหนด ยกเว้นโหนดราก

MM หมายถึง 2M เป็นจำนวนคีย์สูงสุดที่เก็บได้ในแต่ละโหนด

Text_Index เป็นตัวแปรแบบโครงสร้างแทนระเบียนของแฟ้มข้อมูลดัชนี

Link_list เป็นตัวแปรแบบโครงสร้างแทนระเบียนของแฟ้มข้อมูลผกผัน

node เป็นตัวแปรแบบโครงสร้างแทนระเบียนของแฟ้มข้อมูลดิคชันนารี

rootnode และ root เป็นตัวแปรส่วนกลาง (global) ของโปรแกรม

โดยที่ rootnode เป็นตัวแปรแบบ NODE แทนโหนดรากของบีทรี

และ root เป็นตัวแปรแบบ long ใช้เก็บตำแหน่งของ rootnode

ในการทำงานเราจะกำหนดให้พารามิเตอร์ n, k_i, และ p_i แทนพารามิเตอร์ t.cnt, t.key[i], และ t.ptr[i] ตามลำดับ โดยที่พารามิเตอร์ t เป็นตัวชี้ ที่ชี้ไปยังโหนดที่เรากำลังพิจารณาอยู่ (current node) ดังนั้นเราจะได้

$1 <= n <= 2M$ สำหรับโหนดราก

$M <= n <= 2M$ สำหรับโหนดอื่นๆ

แต่ละโหนดมีโหนดลูกมากที่สุด 2M+1 โหนด ดังนั้นแต่ละโหนดจึงมีตัวชี้ 2M+1 ตัว และในแต่ละโหนด ตัวชี้ p_i และ คีย์ k_i จะจัดเรียงในลักษณะดังนี้

$p_0, k_0, p_1, k_1, \ldots, p_{n-1}, k_{n-1}, p_n$

โดยที่ $k_0 < k_1 < \ldots < k_{n-1}$

ค่า n เป็นจำนวนคีย์ที่มีในแต่ละโหนด และแต่ละโหนด ค่า p_i จะเป็นดังนี้

ถ้าเป็นโหนดใบ ค่า p_i แต่ละตัวในโหนดจะมีค่าเป็น null

ถ้าไม่ใช่โหนดใบ ค่า p_i จำนวน n+1 ตัวจะชี้ไปยังโหนดลูก ซึ่งถ้า i มากกว่า 0 ทุกๆ คีย์ในโหนดลูกที่ถูกชี้โดย p_i จะมีค่ามากกว่า k_{i-1} และในทำนองเดียวกัน ถ้า i น้อยกว่า n ทุกๆ คีย์ในโหนดลูกที่ถูกชี้โดย p_i จะมีค่าน้อยกว่า k_i

ศูนย์วิทยทรัพยากร
จุฬาลงกรณ์มหาวิทยาลัย

3. ขั้นตอนการทำงานของโปรแกรม

ขั้นตอนการทำงานของโปรแกรม สำหรับระบบการเก็บและการค้นคืนสารสนเทศ ประกอบด้วยส่วนของโปรแกรม ที่ทำหน้าที่หลักที่สำคัญ 4 ประการ คือ

1) การสร้างฐานข้อมูล
2) การค้นหาข้อมูล
3) การเพิ่มข้อมูล
4) การปรับปรุงแฟ้มข้อมูลสารบัญ

3.1 การสร้างฐานข้อมูล

ประกอบด้วยส่วนของการอ่านข้อมูลจากแฟ้มข้อมูลนำเข้า เพื่อสร้างแฟ้มข้อมูลดิคชันนารี และแฟ้มข้อมูลผกผัน การทำงานประกอบด้วยฟังก์ชันหลัก ต่อไปนี้

3.1.1 ฟังก์ชัน create

ฟังก์ชัน create จะทำการอ่านตัวอักษรจากแฟ้มข้อมูลนำเข้าทีละตัว จนกว่าจะหมดแฟ้ม และตรวจสอบว่า เป็นตัวอักษรที่อยู่ในชุดของตัวอักษร ที่กำหนดไว้หรือไม่ โดยโปรแกรมจะไม่สนใจสัญญลักษณ์พิเศษต่างๆ เช่น / < > ? : ; , " ' [ ] { } ( ) ถ้าเป็นตัวอักษรที่มีการกำหนดไว้ ก็จะเก็บไว้ในพารามิเตอร์ word เมื่ออ่านมาถึงตัวอักษรที่เป็นตัวแบ่งคำ ซึ่งได้แก่ ช่องว่าง (space หรือ blank) ตัวจบบรรทัด (end of line) หรือตัวจบแฟ้มข้อมูล (end of file) ก็จะทำการตรวจสอบพารามิเตอร์ word ถ้า word มีค่าเป็น .dh หมายถึงเป็นการเริ่มต้นเอกสารชุดใหม่ ก็จะเก็บตำแหน่งเริ่มต้นของเอกสาร พร้อมทั้งชื่อเรื่องของเอกสาร ไว้ในแฟ้มข้อมูลดัชนี (filename.INX) พร้อมทั้งเก็บตำแหน่งเริ่มต้นของเอกสารไว้ใน พารามิเตอร์ย่อยของลิงค์ลิสด้วย เพื่อใช้เป็นตำแหน่งอ้างอิงในการอ่านข้อความจากแฟ้มข้อมูล ในกรณีที่มีการค้นหาข้อมูล และเพิ่มค่าในตัวนับจำนวนเอกสารพร้อมทั้งกำหนดค่าตัวนับจำนวนตอน และตัวนับจำนวนคำ ให้เป็น 0 ถ้า word มีค่าเป็น .p หมายถึงเป็นการเริ่มต้นตอนใหม่ หรือย่อหน้าใหม่ในเอกสาร ก็จะเก็บตำแหน่งเริ่มต้นของตอนใหม่ไว้ในพารามิเตอร์ย่อยของลิงค์ลิส และเพิ่มค่าในตัวนับจำนวนตอน พร้อมทั้งกำหนดค่าตัวนับจำนวนคำให้เป็น 0 ถ้า word ไม่ใช่ .dh หรือ .p ก็จะหมายถึง word เป็นคีย์ก็จะเพิ่มค่าในตัวนับจำนวนคำ พร้อมทั้งเก็บตัวนับจำนวนเอกสาร ตัวนับจำนวนตอน และตัวนับจำนวนคำของ word ไว้ในพารามิเตอร์ย่อยของลิงค์ลิส จากนั้นจะเรียกใช้ฟังก์ชัน insert พร้อมทั้งส่งพารามิเตอร์ word และตำแหน่งของลิงค์ลิสไปยังฟังก์ชัน insert เพื่อนำไปใส่ในโหนดของบีทรี ในตอนสุดท้ายของฟังก์ชัน create จะทำการปรับปรุงค่าในระเบียนแรกของแฟ้มข้อมูล filename.DIC และ filename.INX เพื่อให้มีค่าที่ถูกต้อง

ขั้นตอนการทำงานของฟังก์ชัน create

```
create()
{   char  word[MAXWORD];  LINK  address;  INDX  index;
    long  add;  int  countdoc, countpara, countword;
    openfile;
    while (not end of file)
    {   read character from input file;
        if (character in character set)  store character in word;
        if (character = split character)
        {  if (word = .dh)
           {  store address of document in index.docadd;
              store document title in index.title;
              store address of document in address.begin;
              increament countdoc;
              countpara = 0;
              countword = 0;
           }
           if (word = .p)
           {  store address of document in address.begin;
              increament countpara;
              countword = 0;
           }
           else   /*  word is key, store word in B-tree.  */
           {  increament countword;
              store countdoc in address.docno;
              store countpara in address.parano;
              store countword in address.wordno;
              assign add to point node of link list;
              insert(word,add);
           }
        }
    }   /* end while */
```

```
    update the first record of index file and dictionary file;
    closefile;
}
```

3.1.2 ฟังก์ชัน insert และ ฟังก์ชัน ins

ฟังก์ชันสำหรับการแทรกข้อมูลในโหนดของบีทรี ค่อนข้างจะยุ่งยาก และซับซ้อน การทำงานประกอบด้วย 2 ฟังก์ชัน คือฟังก์ชัน insert และฟังก์ชัน ins โดยฟังก์ชัน insert จะถูกเรียกโดยฟังก์ชัน create และฟังก์ชัน ins ถูกเรียกโดยฟังก์ชัน insert

ขั้นตอนการทำงานของฟังก์ชัน insert

```
insert(char *word, long add)
{   long anew, tnew;  int code, onew;  char *wordnew;
    code = ins(word, add, root, &wordnew, &onew, &anew, &tnew);
    if (code = 1 or code = 2)   return
    /*  code = 0  */
    create new rootnode;
    store parameter wordnew, onew, and anew in new rootnode;
    update rootnode.ptr[0] to point old root;
    update rootnode.ptr[1] to point tnew;
    update root to point new rootnode;
}
```

การทำงานของฟังก์ชัน insert จะเรียกใช้ฟังก์ชัน ins และในการทำงานของฟังก์ชัน ins จะส่งค่ากลับ (return value) มายังฟังก์ชัน insert โดยเก็บไว้ในตัวแปร code ซึ่งมี 3 กรณี คือ ส่งค่ากลับเป็น 1 ถ้าการทำงานเสร็จเรียบร้อยสมบูรณ์ และส่งค่ากลับเป็น 2 ถ้าคีย์มีอยู่ในโหนดของบีทรีเรียบร้อยแล้ว ใน 2 กรณีนี้พารามิเตอร์ wordnew, onew, anew, และ tnew ไม่ได้ใช้

กรณีที่ 3 เป็นกรณีที่น่าสนใจมาก ฟังก์ชัน ins จะส่งค่ากลับเป็น 0 ซึ่งหมายถึง การทำงานยังไม่เสร็จสมบูรณ์ ถึงแม้ว่าคีย์อาจจะถูกใส่ในโหนดของบีทรีเรียบร้อยแล้วก็ตาม ในกรณีนี้ฟังก์ชันจะมีการผ่านค่าต่างๆ กลับมาในพารามิเตอร์ wordnew, onew, anew, และ tnew ด้วย (ในภาษาซีต้องส่งแอดเดรส (สัญญลักษณ์ &) ของ wordnew, onew, anew, และ tnew ไปเป็นอาร์กิวเมนท์)

ขั้นตอนการทำงานของฟังก์ชัน ins

```
int  ins(char *word,long add,long t,char **y,int *v,long *x,long *u)
{   long anew, tnew;  int code, onew;  char *wordnew;
    if (t = null)  return 0;
    /*  select pointer p[i] and try to insert word in the subtree
        of which p[i] is the root.  */
    readnode(t);
    i = binsearch(word, k, *n);
    if (word = k[i])
    {  duplicate key, create link list of word's location;
       return 2;
    }
    code = ins(word, add, p[i], &wordnew, &onew, &anew, &tnew);
    if (code = 1 or code = 2)  return code;

    /*  insertion in subtree did not completely succeed,
        try to insert wordnew, onew, anew, and tnew
        in the current node at i position.  */
    if (n < MM)       /*  current node was not full  */
    {   i = binsearch(wordnew, k, *n);
        shift items at location i..n-1  one position to the right;
        insert items wordnew, onew, anew, and tnew in the current node
            at i position;
        return 1;  /*  insertion completely successful  */
    }
```

```
/*  current node was already full  */
if (i = MM)
{  /* we try to insert item at the end of node which already full,
          so : */
   insert items in temporary variable;
}
else
{  /* insert item at i position of already full node, so : */
   move the last items to temporary variable;
   shift items at location i..MM-2 one position to the right;
   insert items wordnew, onew, anew, and tnew into node
        at i position;
}

/*  now node is full and the last items of node is in temporary
    variable, so split node  */
move the middle items of node to parameter y, v, and x
create new node with pointer u;
move the second half of items in the node to new node;
move items from temporary variable to new node;
return 0;   /*  to report that insertion was not completed,
                since we have to insert the middle items into
                the one level higher node */
}
```

จากที่กล่าวมาแล้ว ถึงการทำงานของฟังก์ชัน ins ข้างต้น เราต้องการเพิ่ม word ในโหนดของทรี หรือสับทรีที่มีโหนดรากเป็น t ซึ่งถ้าการแทรกเสร็จสมบูรณ์ พารามิเตอร์ y, v, x, และ u จะไม่ถูกใช้ และฟังก์ชัน ins จะส่งค่ากลับเป็น 1 แต่ถ้าการแทรกไม่สำเร็จ ฟังก์ชัน ins จะส่งค่ากลับเป็น 0 พร้อมกับผ่านค่าต่างๆ กลับไปโดยพารามิเตอร์ y, v, x, และ u

กรณีหนึ่งที่ฟังก์ชันส่งค่ากลับเป็น 0 คือ เมื่อ t เป็น null ในกรณีนี้เราไม่ต้องทำอะไรมาก เนื่องจากไม่มีโหนดในสับทรีที่ t ชี้อยู่ แต่ถ้า t ไม่เป็น null ก็จะชี้ไปที่โหนด เราจะหาตำแหน่ง p_i ของโหนด t เพื่อที่จะแทรก word ในโหนดที่มี p_i เป็นโหนดราก โดยเปรียบเทียบ word กับค่าคีย์ที่อยู่ในโหนด t (โดยเรียกใช้ฟังก์ชัน binsearch) และฟังก์ชัน ins จะทำการเรียกตัวเองมาทำงาน (recursive call)

ในกรณีที่ฟังก์ชัน ins ที่เรียกตัวเองส่งค่ากลับเป็น 1 หมายถึง word ถูกใส่ในสับทรีที่มี p_i เป็นโหนดรากเรียบร้อยแล้ว ดังนั้นฟังก์ชัน ins จะส่งค่า 1 กลับทันที และถ้าฟังก์ชัน ins ที่เรียกตัวเองส่งค่ากลับเป็น 2 ก็หมายความว่า word มีอยู่ในสับทรีเรียบร้อยแล้ว ฟังก์ชัน ins จะส่งค่า 2 กลับทันทีเช่นกัน

ส่วนในกรณีที่ 3 ฟังก์ชัน ins ที่เรียกตัวเองส่งค่ากลับเป็น 0 เราจะพิจารณาพารามิเตอร์ wordnew, onew, anew, และ tnew ซึ่งเราจะต้องพยายามใส่ลงในโหนดที่ t ชี้อยู่ ซึ่งอาจจะทำได้หรือทำไม่ได้ ถ้าสามารถทำได้จะส่งค่ากลับเป็น 1 ตลอด และการเพิ่มข้อมูลก็จะเสร็จสมบูรณ์ ถ้าไม่สามารถทำได้เราจะแบ่งโหนด t โดยการสร้างโหนดใหม่ขึ้นมา 1 โหนด และแบ่งคีย์ในครึ่งหลังของโหนด t ไปใส่ในโหนดใหม่ที่สร้างขึ้นมา ดังนั้นจึงต้องเก็บตำแหน่งของโหนดใหม่ ไว้ในพารามิเตอร์ u หรืออีกนัยหนึ่ง u จะชี้ไปยังโหนดใหม่ที่มีการสร้างขึ้นมา เมื่อโหนด t ตัวเก่าเต็ม และเก็บค่าต่างๆ ที่อยู่ตรงกลางของโหนด ไว้ในพารามิเตอร์ y, v, และ x เพื่อนำไปใส่ในโหนดที่อยู่สูงขึ้นไปอีก 1 ระดับ และส่งค่ากลับเป็น 0

ย้อนกลับมาพิจารณาฟังก์ชัน insert ในกรณีที่โหนดเต็มและฟังก์ชัน ins ส่งค่ากลับเป็น 0 พร้อมทั้งผ่านค่าในพารามิเตอร์ y, v, x, และ u มายังพารามิเตอร์ wordnew, onew, anew, และ tnew ซึ่งหมายถึง เราต้องทำการสร้างโหนดรากตัวใหม่เพื่อเก็บพารามิเตอร์ wordnew, onew, anew, และ tnew ซึ่งจะทำให้ความสูงของบีทรีเพิ่มขึ้นอีก 1 ระดับ และในกรณีนี้เราต้องปรับปรุงค่าใน root ให้ชี้ไปยังโหนดรากตัวใหม่

3.1.3 ฟังก์ชัน binsearch

เป็นฟังก์ชันที่ใช้สำหรับการค้นหาตำแหน่ง ที่จะแทรกข้อมูลในโหนดของบีทรี (ในกรณีที่เป็นการสร้างฐานข้อมูล) หรือค้นหาตำแหน่ง ที่จะอ่านข้อมูลจากโหนดของบีทรี (ในกรณีที่เป็นการค้นหาข้อมูล) โดยทำการค้นหาแบบทวิภาค (binary seach) ซึ่งเหมาะสำหรับชุดของคีย์ในโหนดของบีทรี ที่มีการเก็บแบบเรียงลำดับจากน้อยไปหามาก

สมมติมีชุดของคีย์ a[i] (i = 0, 1, ..., n-1)
จะได้ว่า a[0] < a[1] < ... < a[n-1]
และกำหนดตัวแปร char a[n], word; โดยที่ word คือคีย์ที่ต้องการค้นหา

การทำงานของฟังก์ชัน binsearch จะเริ่มต้นที่คีย์ที่อยู่ตรงกลางของโหนด โดยการเปรียบเทียบกับคีย์ที่ต้องการ ถ้าไม่ใช่คีย์ที่ต้องการก็จะทำการค้นหาต่อไปในครึ่งซ้าย หรือครึ่งขวาของโหนด โดยขึ้นอยู่กับคีย์ที่ต้องการว่ามีค่าน้อยกว่า หรือมากกว่าคีย์ที่อยู่ตรงกลางของโหนด ซึ่งในการทำงานจริงๆ ค่าคีย์ที่ต้องการ อาจไม่มีอยู่ในโหนด ดังนั้นในการทำงานของฟังก์ชัน binsearch จึงต้องแจ้งให้ผู้ใช้ทราบด้วยว่า มีคีย์อยู่ในโหนดหรือไม่ โดยมีการส่งค่าบางค่ากลับไป และในกรณีที่ไม่มีคีย์อยู่ในโหนด ก็จะแจ้งให้ทราบด้วยว่าตำแหน่งที่คีย์ควรจะอยู่คือตำแหน่งใดของโหนด ดังนั้นฟังก์ชัน binsearch จึงสามารถใช้ได้ทั้งในกรณีที่เป็นการแทรกข้อมูลในโหนดของบีทรี หรือการค้นหาข้อมูลในโหนดของบีทรี ในการทำงานจะส่งค่ากลับเป็น i ดังนี้

```
i = 0  ถ้า  word <= a[0]
i = n  ถ้า  word > a[n-1]
i = j  ถ้า  a[j-1] < word <= a[j]  สำหรับ j (1 <= j <= n-1)
```

ในการทำงานเราจะหลีกเลี่ยงความผิดพลาดที่อาจเกิดขึ้น 2 กรณี คือ กรณีที่ได้ผลลัพธ์ที่ผิดพลาด และกรณีที่มีการวนลูปไม่รู้จบ (endless loop) ดังนั้นถ้าเราทำการทดสอบค่า 2 กรณีคือ word <= a[0] และ word > a[n-1] ไว้ล่วงหน้า กรณีที่เหลือที่จะต้องพิจารณาก็คือ

```
a[0] < word <= a[n-1]
```

ซึ่งเราสามารถจะลดเงื่อนไขนี้ลงมาได้เป็น

a[j-1] < word <= a[j]

และเราสามารถกำหนด 2 เงื่อนไขนี้ใหม่ ให้เป็นเงื่อนไขเฉพาะได้เป็น

a[left] < word <= a[right] ..... (1)

ดังนั้นเราจึงเริ่มต้นการค้นหาที่ left = 0 และ right = n-1 และในทุกรอบของการค้นหา เราจะสามารถลดช่วงของ left, ..., right ลงได้ครึ่งหนึ่ง โดยการคำนวณ

middle = (right + left)/2

และให้ค่า i คือค่า middle จากนั้นเราจะทำการค้นหาเช่นเดิมอีก จนกระทั่งเราได้

right - left = 1 ..... (2)

เมื่อมาถึงจุดนี้ ทั้ง (1) และ (2) จะเป็นจริง ซึ่งหมายความว่า right คือค่า i ที่เราต้องการ

ขั้นตอนการทำงานของฟังก์ชัน binsearch

```
int binsearch(char *word, char *a[], int n)
{   int i, left, right;
    if (word <= a[0])   return 0;
    if (word > a[n-1])  return n;
    left = 0;   right = n-1;
    while (right-left > 1)
    {   i = (rigth+left)/2;
        if (word <= a[i])  then  right = i;  else  left = i;
    }
    return right;
}
```

ในการทำงานเราให้ค่าที่อยู่ตรงกลาง คือ i เป็น right ถ้า

word <= a[i]

ดังนั้นเราจะได้ word <= a[right]

และในทำนองเดียวกัน เราให้ค่า i เป็น left ถ้า

a[i] < word

ดังนั้นเราจะได้ a[left] < word

เนื่องจากการทำงานในลูป เราไม่ได้กำหนดค่าอื่นใดให้กับตัวแปร left และ right ดังนั้นเงื่อนไขที่ (1) จึงเป็นจริงเสมอ เราจะพิจารณาเงื่อนไขการหยุดทำงานของลูป เนื่องจากแต่ละรอบของการทำงาน ส่วนของลูปจะประกอบด้วยช่วงของ

left, ..., right

ซึ่งจะทำให้การคำนวณค่า middle เป็นจริงเสมอ คือ

left < middle < right

จากวิธีการทำงานดังกล่าวข้างต้น จะเป็นการเพิ่มค่าของ left ขึ้นเสมอ หรือลดค่าของ right ลงเสมอ ดังนั้นจึงเป็นการรับประกันได้ว่า ความยาวของช่วง ที่เรากำลังพิจารณาอยู่ จะลดลงทุกรอบของการทำงาน และจากเงื่อนไขที่ (2) ที่เป็นการหยุดการทำงานของลูป ช่วงที่เรากำลังพิจารณา จะประกอบด้วยค่าอย่างน้อย 2 ค่า และเมื่อความยาวของช่วงเป็น 1 ซึ่งหมายถึง a[n-1] เป็นค่าเดียวกับ a[0] จะจบการทำงานของลูป และฟังก์ชัน binsearch จะจบการทำงาน

3.2 การค้นหาข้อมูล

ประกอบด้วยส่วนของการค้นหาข้อมูลในแฟ้มข้อมูล filename.DIC และอ่านข้อมูลจากแฟ้มข้อมูลข้อความ การทำงานประกอบด้วยฟังก์ชันหลักต่อไปนี้ คือ

3.2.1 ฟังก์ชัน search

เป็นฟังก์ชันที่รับข้อความในการค้นหาข้อมูลจากผู้ใช้ ซึ่งผู้ใช้สามารถป้อนข้อความได้ 2 ลักษณะ คือ ป้อนคำที่ต้องการค้นหา หรือป้อนคำสั่ง ในกรณีที่ป้อนคำที่ต้องการค้นหา โปรแกรมจะแสดงผลลัพธ์เป็นจำนวนซ้ำของคำที่เกิดขึ้น ในกรณีที่เป็นการป้อนคำสั่ง สามารถป้อนได้ 2 รูปแบบ คือ

1) ป้อนคำสั่ง .p เพื่อระบุรายละเอียดในการแสดงผลลัพธ์ ซึ่งสามารถระบุได้ 3 ลักษณะ ดังนี้

- .p lo/คำ หมายถึง ให้แสดงผลลัพธ์เป็น ตำแหน่งของคำ
- .p ti/คำ หมายถึง ให้แสดงผลลัพธ์เป็น หมายเลขของเอกสาร พร้อมชื่อเรื่องของเอกสาร ที่คำปรากฏอยู่
- .p pa/คำ หมายถึง ให้แสดงข้อความในตอนของเอกสารที่คำปรากฏอยู่

2) ป้อนคำสั่ง .q เพื่อเลิกการค้นหาข้อมูล

หลังจากผู้ใช้ป้อนข้อความในการค้นหาข้อมูลเรียบร้อยแล้ว โปรแกรมจะทำการตรวจสอบข้อความที่ผู้ใช้ป้อน ถ้าข้อความเริ่มต้นด้วยสัญญลักษณ์จุด (.) หมายถึงเป็นคำสั่ง โปรแกรมจะทำการตรวจสอบรูปแบบของคำสั่ง ถ้าข้อความไม่ได้เริ่มต้นด้วยจุด จะหมายถึงข้อความคือคำที่ต้องการค้นหา หลังจากตรวจสอบข้อความในการค้นหาเรียบร้อยแล้ว โปรแกรมจะทำการค้นหาคำ ในแฟ้มข้อมูลดิคชันนารี โดยเริ่มต้นค้นหาที่โหนดรากของบีทรี ทำการเปรียบเทียบค่าที่ต้องการค้นหา กับค่าคีย์ในโหนด โดยเรียกใช้ฟังก์ชัน binsearch ในหัวข้อ 3.1.3 และในการทำงานของฟังก์ชัน binsearch จะส่งค่ากลับมาเป็น i ซึ่งอาจจะหมายถึงตำแหน่งที่ค้นพบคำที่ต้องการ แต่ถ้า i ไม่ใช่ตำแหน่งที่ค้นพบคำที่ต้องการ ก็จะทำการค้นหาต่อไปในสับทรีที่มี p[i] เป็นโหนดราก

ขั้นตอนการทำงานของฟังก์ชัน search

```
search()
{   char word[MAXWORD];  long t;
    openfile;
    t = root;
    get command;
    check command;
    while not(t = null)
    {   readnode(t);
        i = binsearch(word, k, n);
        if (word = k[i])    found(word);
        else  t = node.ptr[i];
    }
    closefile;
}
```

3.3 การเพิ่มข้อมูล

เป็นส่วนของการเพิ่มข้อมูลในแฟ้มข้อมูลข้อความ การทำงานประกอบด้วยฟังก์ชันหลัก คือ

3.3.1 ฟังก์ชัน appendtext

เป็นฟังก์ชันที่ให้ผู้ใช้ใส่ชื่อแฟ้มข้อมูลที่ต้องการเพิ่มข้อความลงไป และใส่ชื่อแฟ้มข้อมูลที่จะนำข้อความมาเพิ่ม โดยในการเพิ่มข้อมูล โปรแกรมจะทำการอ่านข้อมูลจากแฟ้มข้อมูลที่นำมาเพิ่ม และนำไปเพิ่มในแฟ้มข้อมูลที่ต้องการเพิ่ม โดยการเพิ่มต่อท้ายข้อความเดิมที่มีอยู่แล้ว หลังจากเพิ่มข้อมูลเรียบร้อยแล้ว ข้อมูลทั้งหมดจะเก็บอยู่ในแฟ้มข้อมูลที่ต้องการเพิ่ม

ขั้นตอนการทำงานของฟังก์ชัน appendtext

```
appendtext()
{   openfile(source);
    openfile(destination);
    seek(end of file, source);
    while not(end of file destination)
    {   read text from destination;
        write text into source;
    }
    closefile(source);
    closefile(destination);
}
```

3.4 <u>การปรับปรุงแฟ้มข้อมูลสารบัญ</u>

การปรับปรุงแฟ้มข้อมูลสารบัญ จะเป็นส่วนที่อนุญาตให้ผู้ใช้เพิ่ม หรือลบรายชื่อของแฟ้มข้อมูลข้อความที่มีการเก็บไว้ในแฟ้มข้อมูลสารบัญ หรือพิมพ์รายชื่อของแฟ้มข้อมูลข้อความที่มีการเก็บไว้ในแฟ้มข้อมูลสารบัญ

การทำงานของฟังก์ชันต่างๆ ถูกควบคุมโดยฟังก์ชันหลัก (main function) ซึ่งขั้นตอนการทำงานได้แสดงไว้ในบทที่ 5

ศูนย์วิทยทรัพยากร
จุฬาลงกรณ์มหาวิทยาลัย