/\*----------------------------------------------------------

哈希表的建立(链式解决冲突)

bluelithium 2011/12/18 15:30

哈希函数: H(k) = k % INIT\_MAXSIZE;

输入样例

25

14 54 67 8 34 22 16 45 87 60 39 28 71 30 107 875 43 21 14 11 9 89 79 66 55

8

10 20 30 40 50 60 70 80

----------------------------------------------------------\*/

#include <stdio.h>

#include <malloc.h>

#define INIT\_MAXSIZE 10

typedef struct Lnode

{

int data;

struct Lnode \*next;

}Lnode, \*ListLink; //建立链表结点

typedef struct

{

int pos;

ListLink firstnode; //建立数组结点

}HashBox;

typedef struct

{

HashBox HArrary[INIT\_MAXSIZE]; //建立哈希数组(哈希表的地址表头)

}HashArray;

void InitHashList(HashArray &l, int input[], int account); //建立哈希表

void VistHashList(HashArray &l); //遍历输出哈希表

int main(void)

{

int account = 0, i = 0, input[256];

HashArray l;

printf("请输入要插入哈希表元素的个数:");

scanf("%d", &account);

printf("请输入要插入哈希表的元素:");

for (i = 0; i < account; i++)

{

scanf("%d", &input[i]);

}

InitHashList(l, input, account);

printf("\n哈希表如下:\n");

VistHashList(l);

return 0;

}

void InitHashList(HashArray &l, int input[], int account) //建立哈希表

{

int i = 0, j = 0, pos = 0;

ListLink q, p;

char ch = 'A';

for (i = 0; i < INIT\_MAXSIZE; i++) //初始化哈希表头

{

l.HArrary[i].pos = ch++;

l.HArrary[i].firstnode = NULL;

}

for (i = 0; i < account; i++)

{

pos = input[i] % INIT\_MAXSIZE; //计算元素地址

q = new Lnode; //申请结点

q->data = input[i];

q->next = NULL;

if(l.HArrary[pos].firstnode == NULL) //判断当前地址表头是否还没有元素连入

{

l.HArrary[pos].firstnode = q;

}

else

{

p = l.HArrary[pos].firstnode;

while (p->next != NULL)

{

p = p->next; //找到链表表尾

}

p->next = q; //将要插入的结点接入表尾

}

}

}

void VistHashList(HashArray &l) //输出哈希表

{

ListLink p;

int i;

for (i = 0; i < INIT\_MAXSIZE; i++)

{

printf("%c. ", l.HArrary[i].pos);

p = l.HArrary[i].firstnode;

while (p != NULL)

{

printf("->%d", p->data);

p = p->next;

}

printf("\n");

}

}