3.2 哈希表之开放地址法_物联网 / 嵌入式工程师 - 慕课网

> 慕课网慕课教程 3.2 哈希表之开放地址法涵盖海量编程基础技术教程，以图文图表的形式，把晦涩难懂的编程专业用语，以通俗易懂的方式呈现给用户。

2. 哈希表之开放地址法

```
int a[7] = {10,22,13,11,24,7,14};

f(key) = key % 7;
f(10) = 3;
f(22) = 1;
f(13) = 6;
f(11) = 4;
f(24) = 3 ;
```

含义：所谓开放地址方解释一旦发生了冲突，就去寻址下一个空的散列地址。只要 散列表足够大，空的散列地址总是能够找到，并且将其记录在内。

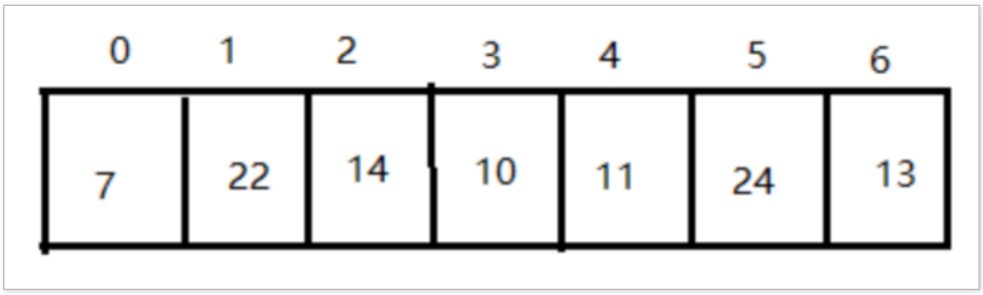
重新构造哈希函数公式：

f(key) = (f(key) + d) % p (d = 1,2,3,4,…n) ; (n <= p – 1)

```
int a[7] = {10,22,13,11,24,7,14};
f(key) = key % 7;
f(10) = 3;
f(22) = 1;
f(13) = 6;
f(11) = 4;
f(24) = 3 ;

新的函数：f(key) = (f(key) + d) % 7;

f(24) =f (f(24) + 1) % 7 =(3 + 1) % 7 = 4 ;还是冲突，继续移动。
f(24) = (f(24) + 2) % 7 = 5;
f(7) = 0;
f(14) = 0;
f(14) = (f(14) + 1 ) % 7 = 1;
f(14) = (f(14) + 2) % 7 = 2;
```



```
#define  MAX  10

#define  NULL_KEY  -1

typedef int datatype_t;

typedef struct node{
    datatype_t *elem_p;
    int n;
}hashtable_t;
```

```c
hashtable_t *create_hashtable()
{
        int i = 0;
        hashtable_t *h = NULL;
        h = (hashtable_t *)malloc(sizeof(hashtable_t));

        h->elem_p = (datatype_t *)malloc(MAX * sizeof(datatype_t));
        h->n = 0;


        for(i = 0;i < MAX;i++)
        {
            h->elem_p[i] = NULL_KEY;
        }

        return h;
}



int is_full_hashtable(hashtable_t  * h)
{
        return h->n == MAX ? 1 : 0;
}



void insert_data_hash(hashtable_t  *h, datatype_t key)
{

        if(is_full_hashtable(h))
        {
                printf("hash table is full!\n");
                return;
        }


        int index = 0;
        index = key % MAX;



        while(h->elem_p[index] != NULL_KEY)
        {
                index = (index + 1) % MAX;
        }


        h->elem_p[index] = key;
        h->n++;
        return ;
}



void printf_hash_table(hashtable_t *h)
{
        int i = 0;

        for(i = 0;i < MAX;i++)
        {
                printf("%d ",h->elem_p[i]);
        }
        printf("\n");
        return;
}



int search_hash_table(hashtable_t *h, datatype_t key)
{

        int index = key % MAX;


        while(h->elem_p[index] != key)
        {
                index = (index + 1) % MAX;


                if((h->elem_p[index] == NULL_KEY) || index == key % MAX)
                        return -1;
        }
        return index;
}
```

```
int main(int argc, const char *argv[])
{
        hashtable_t *h = NULL;
        datatype_t data[MAX] = {13,29,27,28,26,30,38,16,14,19};
        datatype_t value = 0;
        datatype_t ret;
        int i = 0;

        h = create_hashtable();

        for(i = 0;i < MAX;i++)
        {
                insert_data_hash(h,data[i]);
        }

        printf_hash_table(h);
        printf("please input you want to find value : ");
        scanf("%d",&value);
        ret = search_hash_table(h,value);
        if(ret < 0){
                printf("no such data in the hash_tabled!\n");
        }else{
                printf("hashtable index is %d\n",ret);
        }
        insert_data_hash(h,800);
        free(h->elem_p);
        free(h);
        h = NULL;
        return 0;
}
```

运行结果：

```
30 38 16 13 14 19 26 27 28 29
please input you want to find value : 26
hashtable index is 8
hash table is full!
```