4.4 综合练习_物联网 / 嵌入式工程师 - 慕课网

> 慕课网慕课教程 4.4 综合练习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。

球钟是一个利用球的移动来记录时间的简单装置。它有三个可以

容纳若干个球的指示器：分钟指示器，五分钟指示器，小时指示器。若分钟指示

器中有 2 个球，5 分钟指示器中有 6 个球，小时指示器中有 5 个球，则时间为 5:32。

工作原理：

每过一分钟，球钟就会从球队列的队首取出一个球放入分钟指示器，分钟指示器

最多可容纳 4 个球。当放入第五个球时，在分钟指示器内的 4 个球就会按照他们

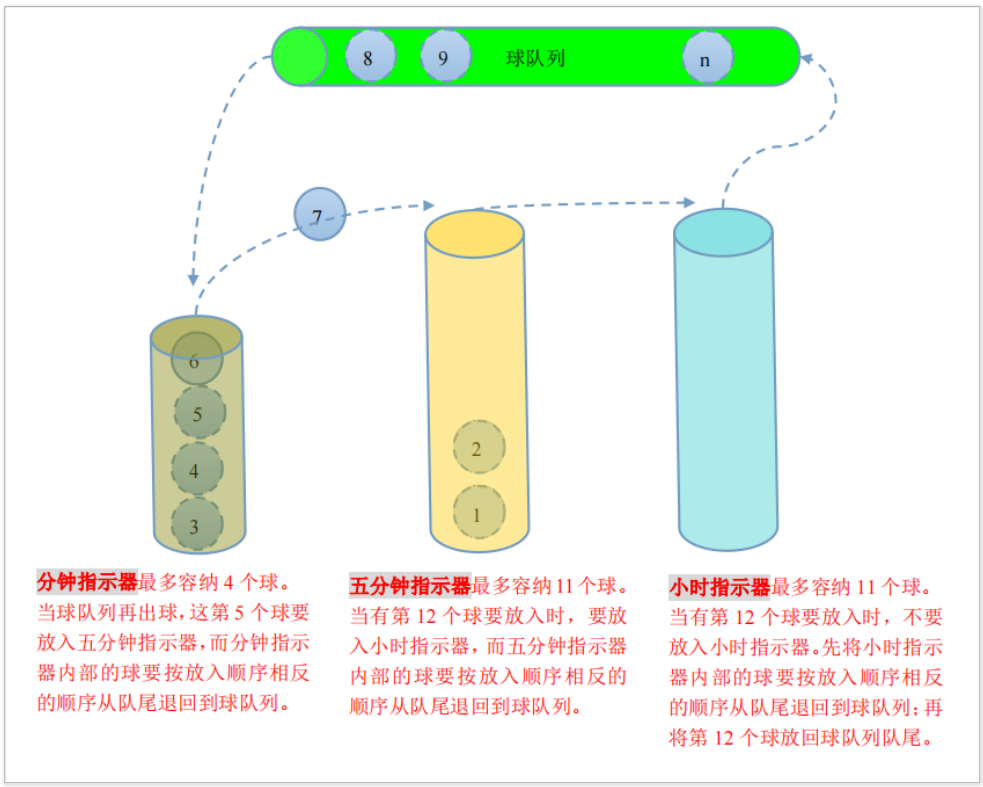
被放入时的相反顺序加入球队列的队尾。而第五个球就会进入五分钟指示器。按

此类推，五分钟指示器最多可放 11 个球，小时指示器最多可放 11 个球。当小时

指示器放入第 12 个球时，原来的 11 个球按照他们被放入时的相反顺序加入球队

列的队尾，然后第 12 个球也回到队尾。这时，三个指示器均为空，回到初始状

态，从而形成一个循环。因此，该球种表示的时间范围是从 00:00 到 11:59



要想表示 00:00 到 12：00 最少需要多少个球?

假设，指示器都为空，球队列需要多长时间能回到原来的状态?

即从初始球队列中球的顺序，经过球的循环后球队列中的球再次与初始顺序

相同。

head.h

```
#ifndef _HEAD_H_
#define _HEAD_H_

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define N 123

typedef int data_t;

typedef struct node
{
    data_t data;
    struct node *next;
}linknode_t;


typedef struct
{
    linknode_t *top;
    int n;
}linkstack_t;


typedef struct
{
    linknode_t *front;
    linknode_t *rear;
}linkqueue_t;


extern linkstack_t *create_empty_linkstack();
extern int is_empty_linkstack(linkstack_t *s);
extern int push_linkstack(linkstack_t *s,data_t data);
extern data_t pop_linkstack(linkstack_t *s);
extern data_t get_top_data(linkstack_t *s);


extern linkqueue_t *create_empty_linkqueue();
extern int is_empty_linkqueue(linkqueue_t *q);
extern int enter_linkqueue(linkqueue_t *q,data_t data);
extern data_t delete_linkqueue(linkqueue_t *q);

#endif
```

linkqueue.c

```
#include "head.h"

linkqueue_t *create_empty_linkqueue()
{
    linknode_t *head = NULL;
    linkqueue_t *q = NULL;

    head = (linknode_t *)malloc(sizeof(linknode_t));
    head->next = NULL;

    q = (linkqueue_t *)malloc(sizeof(linkqueue_t));
    q->front = q->rear = head;

    return q;
}

int is_empty_linkqueue(linkqueue_t *q)
{
    return q->front == q->rear;
}

int enter_linkqueue(linkqueue_t *q,data_t data)
{
    linknode_t *temp = NULL;

    temp = (linknode_t *)malloc(sizeof(linknode_t));
    temp->data = data;


    temp->next = NULL;
```

```
    q->rear->next = temp;

    q->rear = temp;

    return 0;
}

data_t delete_linkqueue(linkqueue_t *q)
{
 linknode_t *temp = NULL;
    data_t data;


    temp = q->front->next;
    data = temp->data;


    q->front->next = temp->next;
    free(temp);
    temp = NULL;


     if(q->front->next == NULL)
     {
         q->rear = q->front;
     }


    return data;
}
```

linkstack.c

```
#include "head.h"

linkstack_t *create_empty_linkstack()
{
    linkstack_t *s = NULL;

    s = (linkstack_t *)malloc(sizeof(linkstack_t));
    s->top = NULL;
    s->n = 0;

    return s;
}

int is_empty_linkstack(linkstack_t *s)
{
    return s->top == NULL;
}

int push_linkstack(linkstack_t *s,data_t data)
{
    linknode_t *temp = NULL;

    temp = (linknode_t *)malloc(sizeof(linknode_t));
    temp->data = data;

    temp->next = s->top;
    s->top = temp;

    s->n++;

    return 0;
}

data_t pop_linkstack(linkstack_t *s)
{
    linknode_t *temp = NULL;
    data_t data;

    temp = s->top;
    data = temp->data;

    s->top = temp->next;

    free(temp);
    temp = NULL;

    s->n--;

    return data;
}
```

```
data_t get_top_data(linkstack_t *s)
{
    return s->top->data;
}
```

ballclock.c

```
#include "head.h"

int print_linklist(linknode_t *head)
{
    linknode_t *p = head->next;

    while(p)
    {

        printf("%-3d\t",p->data);
        p = p->next;
    }
    putchar('\n');

    return 0;
}

int is_orginal_queue(linkqueue_t *q)
{
    int i = 1;
    linknode_t *p = q->front->next;

    for(i = 1;i <= N;i++)
    {
        if(i != p->data)
            return 0;

        p = p->next;
    }

    return 1;
}

int ball_clock()
{
    linkstack_t *min_stack = NULL,
                *min5_stack = NULL,
                *hour_stack = NULL;
    linkqueue_t *ball_queue = NULL;
    int half_day = 0;
    int ball = 0;

    min_stack = create_empty_linkstack();
    min5_stack = create_empty_linkstack();
    hour_stack = create_empty_linkstack();

    ball_queue = create_empty_linkqueue();

    for(ball = 1;ball <= N;ball++)
    {
        enter_linkqueue(ball_queue,ball);
    }

    print_linklist(ball_queue->front);

    while(1)
    {
        ball = delete_linkqueue(ball_queue);

        if(min_stack->n < 4)
        {
            push_linkstack(min_stack,ball);
            continue;
        }

        while(!is_empty_linkstack(min_stack))
        {
            enter_linkqueue(ball_queue,pop_linkstack(min_stack));
        }

        if(min5_stack->n < 11)
        {
            push_linkstack(min5_stack,ball);
            continue;
        }

        while(!is_empty_linkstack(min5_stack))
```

```
        {
            enter_linkqueue(ball_queue,pop_linkstack(min5_stack));
        }

        if(hour_stack->n < 11)
        {
            push_linkstack(hour_stack,ball);
            continue;
        }

        while(!is_empty_linkstack(hour_stack))
        {
            enter_linkqueue(ball_queue,pop_linkstack(hour_stack));
        }

        enter_linkqueue(ball_queue,ball);

        half_day++;

        if(is_orginal_queue(ball_queue))
            break;
    }

    return half_day / 2;
}

int main()
{
    int day_count = 0;

    day_count = ball_clock();

    printf("Restoring orignal queue need %d days\n",day_count);

    return 0;
}
```

全文完