数据结构实验作业

**exp3-5**

#include<stdio.h>

#include<stdlib.h>

#define MAX 10

int map[6][6] = {{1, 1, 1, 1, 1, 1}, //迷宫地图

                 {1, 0, 0, 0, 1, 1},

                 {1, 0, 1, 0, 0, 1},

                 {1, 0, 0, 0, 1, 1},

                 {1, 1, 0, 0, 0, 1},

                 {1, 1, 1, 1, 1, 1}};

struct

{

    int i, j; // i单元格的横坐标，j单元格的纵坐标;

    int di;

}path[MAX],minpath[MAX];

int top = -1;//栈顶

int n;    //代表路径个数

int path\_length = MAX;

void find\_path(int xi,int yi,int xe,int ye)

{

    int i, j, i1, j1, di,k;

    bool find;

    top++;

    path[top].i = xi;

    path[top].j = yi;

    path[top].di = -1;

    map[xi][yi] = -1;

    while(top!=-1)

    {

        i = path[top].i;

        j = path[top].j;

        di =path[top].di;

        if(i==xe&&j==ye)

        {

            printf("第%d条路径为：", n);

            n++;

            for (k = 0; k < top; k++)

            {

                printf("(%d,%d)", path[k].i, path[k].j);

            }

            printf("\n");

            if(top+1 < path\_length)

            {

                for (k = 0; k < top + 1;k++)

                {

                    minpath[k] = path[k];

                }

                path\_length = top + 1;

            }

        map[i][j] = 0;

        top--;

        i = path[top].i;

        j = path[top].j;

        di = path[top].di;

        }

        find = false;

        while (di < 4&&!find)

        {

            di++;

            switch(di)

            {

                 case 0:i1 = i - 1;j1 = j;break;

                 case 1:i1 = i    ;j1 = j+1;break;

                 case 2:i1 = i + 1;j1 = j;break;

                 case 3:i1 = i    ;j1 = j-1;break;

            }

            if(map[i1][j1]==0)

            {

                find = true;

            }

        }

        if(find)

        {

            path[top].di = di;

            top++;

            path[top].i = i1;

            path[top].j = j1;

            path[top].di = -1;

            map[i1][j1] = -1;

        }

        else

        {

            map[i][j] = 0;

            top--;

        }

    }

    printf("最短路径长度为：%d", path\_length);

    printf("\n");

    printf("最短路径为:");

    for (k = 0; k < path\_length;k++)

    {

        printf("(%d,%d)", minpath[k].i, minpath[k].j);

    }

}

int main()

{

    find\_path(1,1,4,4);

    return 1;

}

**exp3-6**

#include<stdio.h>

#include<stdlib.h>

#define MAXN 10

#define MaxSize 20

typedef struct qnode

{

    int data;

    struct qnode \*next;

} DataNode;

typedef struct

{

    DataNode \*front;

    DataNode \*rear;

} LinkQuNode;

void InitQueue(LinkQuNode\*&q)//初始化队列

{

    q = (LinkQuNode \*)malloc(sizeof(LinkQuNode));

    q->front = q->rear = NULL;

}

bool QueueEmpty(LinkQuNode\*q)//判断队列是否为空

{

    return (q->rear ==NULL);

}

void enQueue(LinkQuNode\*&q,int e)//进队列

{

    DataNode \*p;

    p = (DataNode \*)malloc(sizeof(DataNode));

    p -> data = e;

    p->next = NULL;

    if(q->rear==NULL)

    {

        q->front = q->rear = p;

    }

    else

    {

        q->rear->next = p;

        q->rear = p;

    }

}

bool deQueue(LinkQuNode\*&q,int &e)//出队列

{

    DataNode \*t;

    if(q->rear==NULL)

        return false;

    t = q->front;

    if(q->front==q->rear)

        q->front = q->rear = NULL;

    else

        q->front = q->front->next;

    e = t->data;

    free(t);

    return true;

}

void paidui(LinkQuNode\*q)

{

    int number;

    printf("请输入病人的病历号");

    scanf("%d",&number);

    enQueue(q, number);

    printf("病历号为%d\n", number);

}

void jiuzhen(LinkQuNode\*q)

{

    int e1;

    deQueue(q, e1);

    printf("病历号为%d的病人就诊",e1);

}

void chakanpaidui(LinkQuNode\*q)

{

    int number;

    while(!QueueEmpty(q))

    {

        deQueue(q, number);

        printf("病人的病历号为%d", number);

        printf("\n");

    }

}

void buzaipaidui(LinkQuNode\*q)

{

    printf("不再排队\n");

    int number;

    while(!QueueEmpty(q))

    {

        deQueue(q, number);

        printf("病人的病历号为%d", number);

        printf("\n");

    }

}

int main()

{

    LinkQuNode \*qu;

    InitQueue(qu);

    int choose=0;

    printf("排队看医生程序，请选择您想要运行的功能\n");

    printf("1:排队，2：就诊,3:查看排队，4：不再排队，5：退出");

    scanf("%d", &choose);

    while(choose!=5)

    {

        switch(choose)

        {

        case 1:

            paidui(qu);

            break;

        case 2:

            jiuzhen(qu);

            break;

        case 3:

            chakanpaidui(qu);

            break;

        case 4:

            buzaipaidui(qu);

            return 1;

        case 5:

            return 1;

        }

        scanf("%d", &choose);

    }

}

**exp3-7**

#include<stdio.h>

#include<stdlib.h>

#define MAX 10

typedef struct linknode

{

    int data;

    struct linknode \*next;

} LinkStNode;

void InitStack(LinkStNode\*&s)//初始化栈

{

    s = (LinkStNode \*)malloc(sizeof(LinkStNode));

    s->next = NULL;

}

void DestroyStack(LinkStNode \*& s)//销毁栈

{

    LinkStNode \*pre = s, \*p = s->next;

    while(p!=NULL)

    {

        free(pre);

        pre = p;

        p = pre->next;

    }

    free(pre);

}

bool StackEmpty(LinkStNode \*s)//判断栈是否为空

{

    return (s->next == NULL);

}

void Push(LinkStNode \*&s,int e)//进栈

{

    LinkStNode \*p;

    p=(LinkStNode \*)malloc(sizeof(LinkStNode));

    p->data = e;

    p->next = s->next;

    s->next = p;

}

bool Pop(LinkStNode \*&s,int &e)//出栈

{

    LinkStNode \*p;

    if(s->next==NULL)

        return false;

    p = s->next;

    e = p->data;

    s->next = p->next;

    free(p);

    return true;

}

bool GetTop(LinkStNode \*s,int e)

{

    if(s->next==NULL)

        return false;

    e = s->next->data;

    return true;

}

void sort(LinkStNode \*&st)

{

    LinkStNode \*t;

    InitStack(t);

    int number1,number2;

    while(!StackEmpty(st))

    {

        Pop(st, number1);

        printf("将%d从st中出栈\n", number1);

        while(!StackEmpty(t))

        {

            GetTop(t, number2);

            printf("取t栈的栈顶元素%d\n", number2);

            if(number1<number2)

            {

                Pop(t, number2);

                Push(st, number2);

                printf("因为%d<%d", number1, number2);

                printf("所以将%d从t栈中弹出，并将其压入st栈\n",number2);

            }

            else

            {

                break;

            }

        }

        Push(t, number1);

        printf("将%d压入t栈\n", number1);

    }

    printf("最后将t栈的数出栈压入st栈\n");

    while(!StackEmpty(t))

    {

        Pop(t, number1);

        Push(st, number1);

    }

    DestroyStack(t);

}

int main()

{

    LinkStNode \*st;

    InitStack(st);

    int n;

    printf("将1，2，3，4依次进栈\n");

    Push(st, 1);

    Push(st, 2);

    Push(st, 3);

    Push(st, 4);

    sort(st);

    printf("排序完后的顺序为\n");

    while(!StackEmpty(st))

    {

        Pop(st, n);

        printf("%d", n);

    }

    DestroyStack(st);

    return 1;

}