实验四队列及其应用

实验周次：第11周 学时：2学时 地点：学院机房

学号：3190707121 姓名：武新纪 班级：人工智能191 序号：21

提示：请务必填写 以上个人信息。

**【实验目的】**

1.领会环形顺序队列存储结构和掌握环形顺序队列中各种基本运算算法设计。

2.领会链队存储结构和掌握链队中各种基本运算算法设计。

3.掌握顺序队在求解报数问题中的应用。

**【实验内容】**

实验题1.编写一个程序sqqueue.cpp,实现环形顺序队列中的各种基本运算（假设顺序栈的元素类型Elem Type为char,顺序队列中存放队列元素的数组长度为5即#define MaxSize 5），并在此基础上设计一个程序exp3-3.cpp完成以下功能。

1. 初始化环形顺序队列q。
2. 判断环形顺序队列q是否非空。
3. 依次进队元素a,b,c。
4. 出队一个元素，并输出该元素。
5. 依次进队元素d,e,f,g,h。
6. 输出出队序列。
7. 释放队列。

提示内容：不会请查看PPT中相关内容。

**程序；**

**//sqqueue.cpp**

#include <stdio.h>

#include <malloc.h>

#define MaxSize 100

typedef char ElemType;

//定义环队

typedef struct

{

    ElemType data[MaxSize];

    int front, rear;

} SqQueue;

//初始化队

void InitQueue(SqQueue \*&q)

{

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

    q->front = q->rear = 0;

}

//销毁队

void DestroyQueue(SqQueue \*&q)

{

    free(q);

}

//判断队是否为空

bool QueueEmpty(SqQueue \*q)

{

    return (q->front == q->rear);

}

//进队

bool enQueue(SqQueue \*&q, ElemType e)

{

    if ((q->rear + 1) % MaxSize == q->front)

        return false;

    q->rear = (q->rear + 1) % MaxSize;

    q->data[q->rear] = e;

    return true;

}

//出队

bool deQueue(SqQueue \*&q, ElemType &e)

{

    if (q->front == q->rear)

        return false;

    q->front = (q->front + 1) % MaxSize;

    e = q->data[q->front];

    return true;

}

//exp3-3.cpp

#include "sqqueue.cpp"

int main()

{

    ElemType e;

    SqQueue \*q;

    printf("环形队列基本运算如下:\n");

    printf("  (1)初始化队列q\n");

    InitQueue(q);

    printf("  (2)依次进队列元素a,b,c\n");

    enQueue(q, 'a');

    enQueue(q, 'b');

    enQueue(q, 'c');

    printf("  (3)队列为%s\n", (QueueEmpty(q) ? "空" : "非空"));

    if (deQueue(q, e) == 0)

        printf("队空,不能出队\n");

    else

        printf("  (4)出队一个元素%c\n", e);

    printf("  (5)依次进队列元素d,e,f\n");

    enQueue(q, 'd');

    enQueue(q, 'e');

    enQueue(q, 'f');

    printf("  (6)出队列序列:");

    while (!QueueEmpty(q))

    {

        deQueue(q, e);

        printf("%c ", e);

    }

    printf("\n");

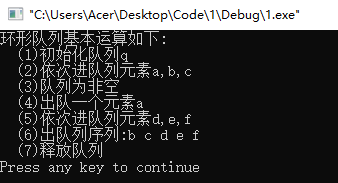
    printf("  (7)释放队列\n");

    DestroyQueue(q);

    return 0;

}

**调试结果；**

****

实验题2.编写一个程序linkqueue.cpp,实现链队的各种基本运算（假设链队中数据域元素类型Elem Type为char），并在此基础上设计一个程序exp3-4.cpp完成以下功能。

1. 初始化链队q。
2. 判断链队q是否非空。
3. 依次进链队元素a,b,c。
4. 出队一个元素，并输出该元素。
5. 依次进链队元素d,e,f,g,h。
6. 输出出队序列。
7. 释放链队。

提示内容：不会请查看PPT中相关内容。

**程序**:

//linkqueue.cpp

//链队运算算法

#include <stdio.h>

#include <malloc.h>

typedef char ElemType;

typedef struct DataNode

{

    ElemType data;

    struct DataNode \*next;

} DataNode;

typedef struct

{

    DataNode \*front;

    DataNode \*rear;

} LinkQuNode;

void InitQueue(LinkQuNode \*&q)

{

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

    q->front = q->rear = NULL;

}

void DestroyQueue(LinkQuNode \*&q)

{

    DataNode \*p = q->front, \*r;

    if (p != NULL)

    {

        r = p->next;

        while (r != NULL)

        {

            free(p);

            p = r;

            r = p->next;

        }

    }

    free(p);

    free(q);

}

bool QueueEmpty(LinkQuNode \*q)

{

    return (q->rear == NULL);

}

void enQueue(LinkQuNode \*&q, ElemType e)

{

    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, ElemType &e)

{

    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;

}

**//exp3-4.cpp**

#include "liqueue.cpp"

int main()

{

    ElemType e;

    LinkQuNode \*q;

    printf("链队的基本运算如下:\n");

    printf("  (1)初始化链队q\n");

    InitQueue(q);

    printf("  (2)依次进链队元素a,b,c\n");

    enQueue(q, 'a');

    enQueue(q, 'b');

    enQueue(q, 'c');

    printf("  (3)链队为%s\n", (QueueEmpty(q) ? "空" : "非空"));

    if (deQueue(q, e) == 0)

        printf("\t提示:队空,不能出队\n");

    else

        printf("  (4)出队一个元素%c\n", e);

    printf("  (5)依次进链队元素d,e,f\n");

    enQueue(q, 'd');

    enQueue(q, 'e');

    enQueue(q, 'f');

    printf("  (6)出链队序列:");

    while (!QueueEmpty(q))

    {

        deQueue(q, e);

        printf("%c ", e);

    }

    printf("\n");

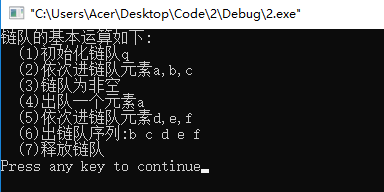
    printf("  (7)释放链队\n");

    DestroyQueue(q);

    return 0;

}

**运行结果；**



实验3.设有n个人站成一排，从左向右的编号分别为1～n，现在从左往右报数“1，2，1，2，…”，数到“1”的人立即站到队伍的最右端，数到“2”的出列。报数过程反复进行，直到n个人都出列为止。要求给出他们的出列顺序。要求给出实现算法函数代码，并用main()函数调用，也可以不用函数，直接在main()函数中实现。要求给出调试运行结果截图。

例如，当n=8时，初始序列为

1 2 3 4 5 6 7 8

提示内容：不会请查看PPT中相关内容。

**程序：**

#include <stdio.h>

#include <malloc.h>

#define MaxSize 100

typedef int ElemType;

typedef struct

{

    ElemType data[MaxSize];

    int front, rear;

} SqQueue;

//初始化队列

void InitQueue(SqQueue \*&q)

{

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

    q->front = q->rear = -1;

}

//销毁队列

void DestroyQueue(SqQueue \*&q)

{

    free(q);

}

//判断队是否为空

bool QueueEmpty(SqQueue \*q)

{

    return (q->front == q->rear);

}

//入队

bool enQueue(SqQueue \*&q, ElemType e)

{

    if (q->rear == MaxSize - 1)

        return false;

    q->rear++;

    q->data[q->rear] = e;

    return true;

}

//出队

bool deQueue(SqQueue \*&q, ElemType &e)

{

    if (q->front == q->rear)

        return false;

    q->front++;

    e = q->data[q->front];

    return true;

}

//主函数

int main()

{

    SqQueue \*q;

    int a, count = 1;

    InitQueue(q);

    for (int i = 1; i <= 10; i++)

        enQueue(q, i);

    printf("出队的序列为:\n");

    while (!QueueEmpty(q))

    {

        deQueue(q, a);

        if (count % 2 != 0)

            enQueue(q, a);

        else

            printf("%d ", a);

        count++;

}

printf("\n");

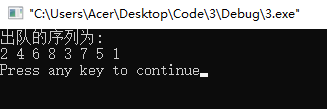
    DestroyQueue(q);

    return 0;

}

**调试结果:**

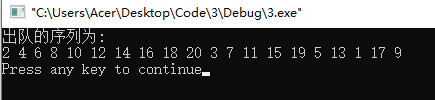
1. **设置不同的n，分别是n=8,10,20，调试结果如下，均符合预期。**
2. **n为8时**

****

**（2）n为10时**

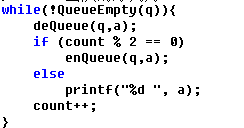
****

**（3）n为20时**

****

**调试结果:调试结果符合需求，程序没有问题**

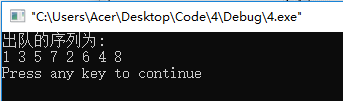
**2、修改主函数部分代码如下**

****

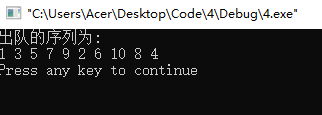
**即设置“数到“1”的出列，数到“2”的人立即站到队伍的最右端”。**

**当n分别为8,10,20时调试结果如下,结果符合预期，程序没有问题。**

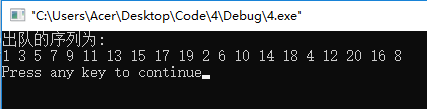
**（1）当n为8时**

****

**（2）当n为10时**

****

**（3）当n为20时**

****