第二次第四周小班讨论（个人资料）

STL序列式容器研讨

队列

湖南大學
HUNAN UNIVERSITY

前言 PREFACE

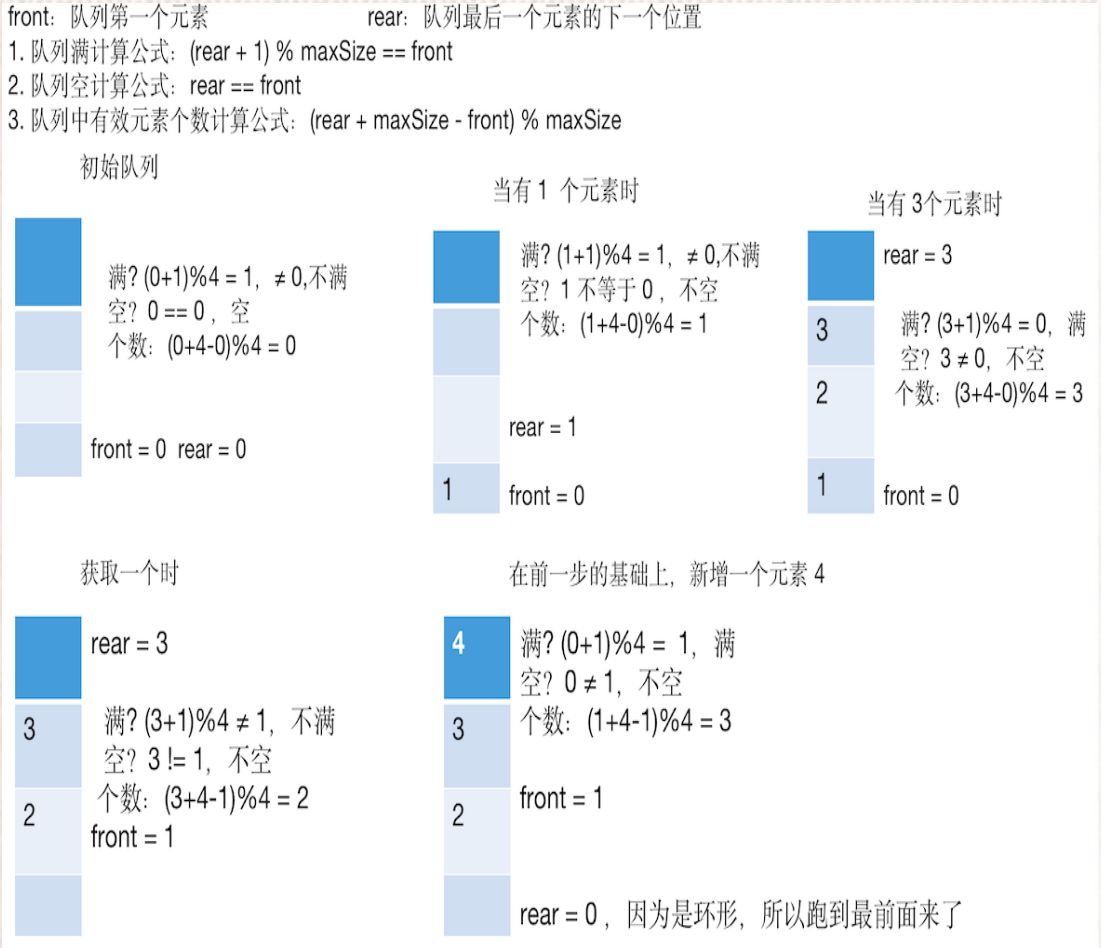
STL(Standard Template Library，标准模板库)是“容器”、算法和其他一些组件的集合。它是由 Alexander Stepanov、Meng Lee 和 David R Musser 在惠普实验室工作时所开发出来的。借助模板技术，STL把常用的数据结构及其算法都实现了一遍，并且做到了数据结构和算法的分离。
STL 的目的是标准化组件，这样就不用重新开发，可以使用现成的组件。STL 现在是 C++的一部分，因此无需额外安装什么。在实际开发过程中，数据结构本身的重要性不会逊于操作于数据结构的算法的重要性，当程序中存在着对时间要求很高的部分时，数据结构的选择就显得更加重要。 经典的数据结构数量有限，但是我们常常重复着一些为了实现向量、链表等结构而编写的代码，这些代码都十分相似，只是为了适应不同数据的变化而在细节上有所出入。STL 容器就为我们提供了这样的方便，它允许我们重复利用已有的实现构造自己的特定类型下的数据结构，通过设置一些模板类，STL 容器对最常用的数据结构提供了支持，这些模板的参数允许我们指定容器中元素的数据类型，可以将我们许多重复而乏味的工作简化。

目录

湖南大学
HUNAN UNIVERSITY

第一部分

概述



队列

限定在一端进行插入，另一端进行删除特殊线性表。就像排队买东西，排在前面的人买完东西后离开队伍（删除），而后来的人总是排在队伍未尾（插入）。通常把队列的删除和插入分别称为出队和入队。允许出队的一端称为队头，允许入队的一端称为队尾。所有需要进队的数据项，只能从队尾进入，队列中的数据项只能从队头离去。由于总是先入队的元素先出队（先排队的人先买完东西），这种表也称为先进先出（FIFO）表。

队列的分类

顺序队列的定义

顺序队列使用一组地址连续的存储单元，依次存放从队头到队尾的数据元素。需要附设两个指针：**队头指针（front）和队尾指针（rear）**，分别指向队头元素和队尾元素。如果在插入元素E的基础上再插入元素F，将会插入失败，因为尾指针已经达到队列的最大长度。这种现象叫做**“假溢出”**，因为队列存储空间并未全部被占满。

循环队列的定义

为了解决“假溢出”现象，使得队列的存储空间得到充分利用，我们可以将顺序队列的数组看成一个头尾相接的循环结构。这样的队列称为**循环队列**。循环队列的判满和判空条件需要特别注意

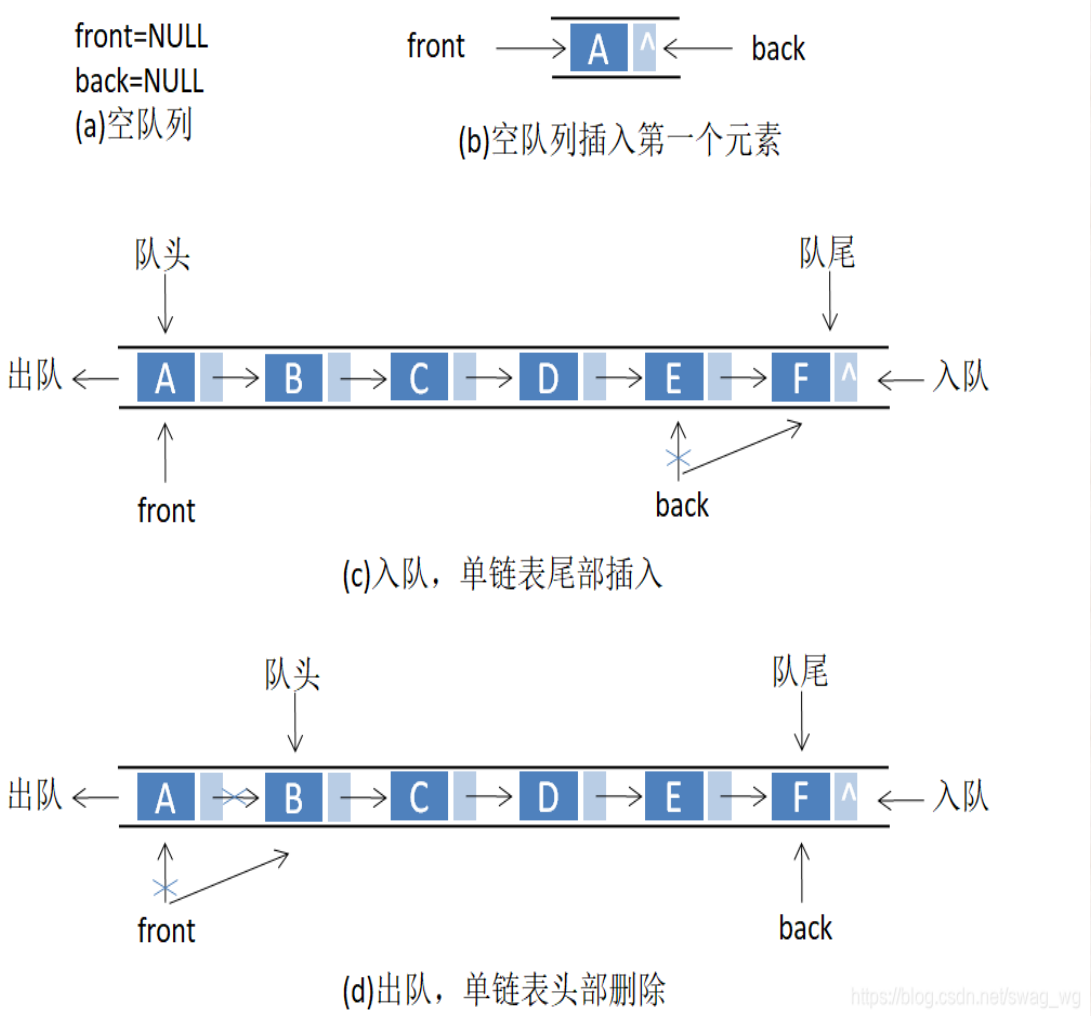
湖南大学
HUNAN UNIVERSITY

第二部分

物理结构的定义

标题二 队列的分类

副标题



链队列的定义

链队列采用链式存储结构实现，队头指针指向链队列的头结点，队尾指针指向终端结点。空队列时，front和rear都指向头结点，即front == rear。

顺序队列的定义

顺序队列使用一组地址连续的存储单元，依次存放从队头到队尾的数据元素。需要附设两个指针：**队头指针（front）和队尾指针（rear）**，分别指向队头元素和队尾元素。如果在插入元素E的基础上再插入元素F，将会插入失败，因为尾指针已经达到队列的最大长度。这种现象叫做**“假溢出”**，因为队列存储空间并未全部被占满。

循环队列的定义

为了解决“假溢出”现象，使得队列的存储空间得到充分利用，我们可以将顺序队列的数组看成一个头尾相接的循环结构。这样的队列称为**循环队列**。循环队列的判满和判空条件需要特别注意

湖南大学
HUNAN UNIVERSITY

第三部分

基本操作

队列的基本操作

初始化队列（InitQueue）

将队列初始化为空队列。

判断队列是否为空（IsEmpty）

若队列为空则返回1，否则返回0。

判断队列是否已满（IsFull）

若队列为满则返回1，否则返回0。

取队首元素（GetHead）

返回队列的队头元素值

入队操作（EnterQueue）

在队列的队尾插入元素。

出队操作（DeleteQueue）

将队列的队头元素出队，并返回出队元素的值。

清空队列（ClearQueue）

将队列置为空队列。

代码实现

```c
#define MaxSize 10
typedef int DataType;
typedef struct Queue {
    DataType Queue[MaxSize];
    int front; // 队头指针
    int rear; // 队尾指针
} SeqQueue;

// 初始化队列
void InitQueue(SeqQueue* SQ) {
    SQ->front = SQ->rear = 0;
}

// 判断队列是否为空
int IsEmpty(SeqQueue* SQ) {
    return SQ->front == SQ->rear;
}

// 判断队列是否为满
int IsFull(SeqQueue* SQ) {
    return SQ->rear == MaxSize;
}
```

```c
// 入队
void EnterQueue(SeqQueue* SQ, DataType data) {
    if (IsFull(SQ)) {
        printf("队列已满\n");
        return;
    }
    SQ->Queue[SQ->rear] = data;
    SQ->rear++;
}

// 出队
int DeleteQueue(SeqQueue* SQ, DataType* data) {
    if (IsEmpty(SQ)) {
        printf("队列为空！\n");
        return 0;
    }
    *data = SQ->Queue[SQ->front];
    SQ->front++;
    return 1;
}
```

```c
// 获取队首元素
int GetHead(SeqQueue* SQ, DataType* data) {
    if (IsEmpty(SQ)) {
        printf("队列为空!\n");
        return 0;
    }
    *data = SQ->Queue[SQ->front];
    return 1;
}

// 清空队列
void ClearQueue(SeqQueue* SQ) {
    SQ->front = SQ->rear = 0;
}

// 打印队列中的元素
void PrintQueue(SeqQueue* SQ) {
    int i = SQ->front;
    while (i < SQ->rear) {
        printf("%-3d", SQ->Queue[i]);
        i++;
    }
    printf("\n");
}
```

湖南大学
HUNAN UNIVERSITY

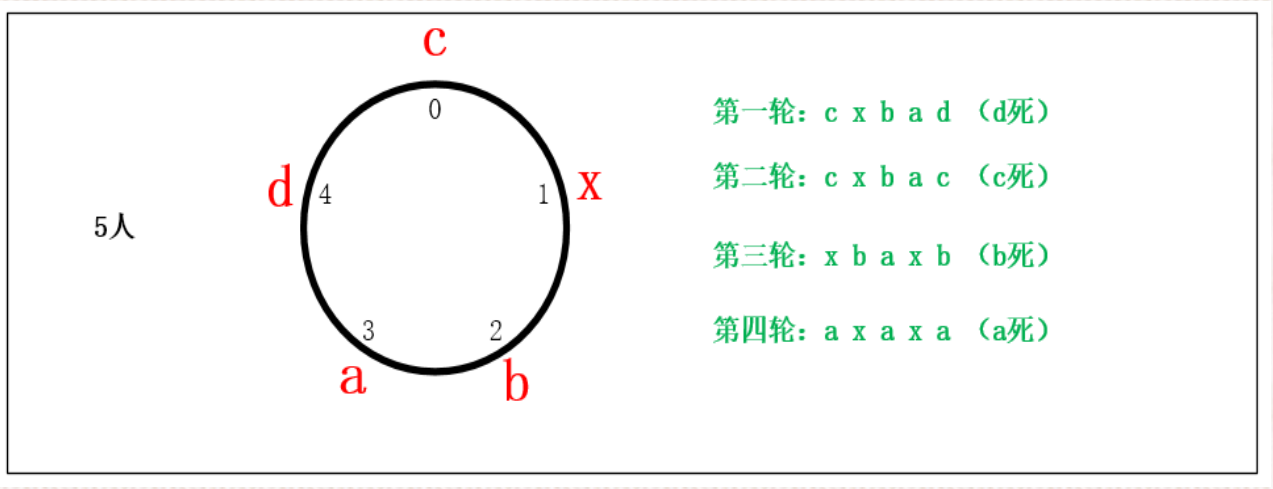
第四部分

应用

约瑟夫问题

问题描述：

设有 n 个人依次围成一圈，从第 1 个人开始报数，数到第 m 个人出列，然后从出列的下一个人开始报数，数到第 m 个人又出列，…，如此反复到所有的人全部出列为止。设 n 个人的编号分别为1，2，…，n，打印出列的顺序。



算法分析：

本题我们可以用数组建立标志位等方法求解，但如果用上数据结构中循环链的思想，则更贴切题意，解题效率更高。 n 人围成一圈，把一人看成一个结点， n 人之间的关系采用链接方式，即每一结点有一个前继结点和一个后继结点，每一个结点有一个指针指向下一个结点，最后一个结点指针指向第一个结点。这就是单循环链的数据结构。当 m 人出列时，将 m 结点的前继结点指针指向 m 结点的后继结点指针，即把 m 结点驱出循环链。

解题思路：

1、建立循环链表。当用数组实现本题链式结构时，数组a[i]作为”指针”变量来使用，a[i]存放下一个结点的位置。设立指针j指向当前结点，则移动结点过程为j=a[j]，当数到m时，m结点出链，则a[j]=a[a[j]]。 当直接用链来实现时，则比较直观，每个结点有两个域：一个数值域，一个指针域，当数到m时，m出链，将m结点的前继结点指针指向其后继结点；

2、设立指针，指向当前结点，设立计数器，计数数到多少人；

3、沿链移动指针，每移动一个结点，计数器值加 1 ，当计数器值为m时， 则m结点出链，计数器值置为 1 。

4、重复 3 ，直到n个结点均出链为止。

参考代码

```
#include <bits/stdc++.h>
using namespace std;
int n,m;            //设有n人,报到m人出列
int a[110],j,k=1,p=0;
int main()
{ cin>>n>>m;  j=n;
  for (int i=1;i<n;i++) a[i]=i+1;      //建立链表
  a[n]=1;       //第n人指向第1人，形成一个环
  while (p<n)                  //n个人均出队为止
  {
     while(k<m)                //报数,计数器加1
     {    k++;  j=a[j]; }
          printf("%d ",a[j]); p++;      //数到m,此人出队,计数器置1
          a[j]=a[a[j]]; k=1;
  }
  return 0;
}//样例输入4 17输出1 3 4 2
```

机器翻译

题目描述

小晨的电脑上安装了一个机器翻译软件，他经常用这个软件来翻译英语文章。这个翻译软件的原理很简单，它只是从头到尾，依次将每个英文单词用对应的中文含义来替换。对于每个英文单词，软件会先在内存中查找这个单词的中文含义，如果内存中有，软件就会用它进行翻译；如果内存中没有，软件就会在外存中的词典内查找，查出单词的中文含义然后翻译，并将这个单词和译义放入内存，以备后续的查找和翻译。假设内存中有M个单元，每单元能存放一个单词和译义。每当软件将一个新单词存入内存前，如果当前内存中已存入的单词数不超过M-1，软件会将新单词存入一个未使用的内存单元；若内存中已存入M个单词，软件会清空最早进入内存的那个单词，腾出单元来，存放新单词。假设一篇英语文章的长度为N个单词。给定这篇待译文章，翻译软件需要去外存查找多少次词典？假设在翻译开始前，内存中没有任何单词。

算法分析

维护内存单元中的单词：状态数组 + 队列。
需要查询一个单词时，如果已在队列中，跳过；否则，存入新单词(如果队列满，先清空最早进入的单词)。
M <= 100，N <=1000。
队列中最多有M个元素，但大小须定义到N，存在空间的浪费。
改进：使用循环队列普通队列
循环队列 出队： head++; head = (head + 1) % M
入队： tail++; tail = (tail + 1) % M

机器翻译

代码参考

```
#include <cstdio>
const int maxm=110;const int maxn=1010;
int m,n,p[maxn],k,ans;
int q[maxm],head,tail;  //循环队列
int main(){
    scanf("%d%d",&m,&n);
    for(int i=0;i<maxn;i++) p[i]=-1;
    scanf("%d",&k);
    head=0; tail=1; q[1]=k;
    p[k]=0; ans=1;
    for(int i=1;i<n;i++){
        scanf("%d",&k);
        if(p[k]==-1){
            ans++;
            if(head==tail){     //队列满
                head=(head+1)%m;
                                                p[q[head]]=-1;  //队首出队

            }
                    tail=(tail+1)%m;
                                            q[tail]=k; p[k]=tail;//k入队
        }
    }
    printf("%d\n",ans); return 0;
}
```

谢 谢 观 看

thank you for watching

参考文献

郭艳燕，童向荣，孙雪姣，等.程序设计基础与数据结构两门课程的'教学衔接[J].计算机教育，2014(10)：47-50.

高贤强，化希耀，陈立平.引入计算思维的《数据结构》教学改革研究[J].现代计算机：专业版，2015(7)：16-19.

严蔚敏.数据结构C语言版[M].清华大学出版社，2007.