



栈与队列

北京亚嵌教育研究中心

栈与队列

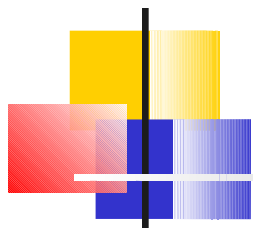
数据结构：

栈的实现

栈的应用

队列的实现

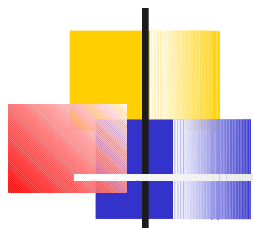
队列的应用



栈与队列

数据结构：

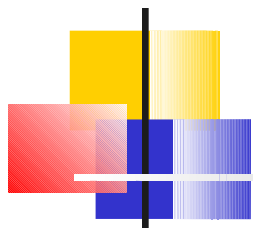
数据的组织办法，主要研究数据如何存储以及在此之上的数据操作方法。



栈与队列

数据结构：

数据的组织方式，主要研究数据之间的组织关系以及在此之上的数据操作方法。



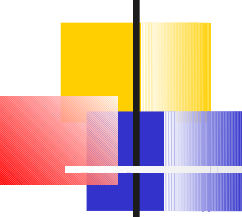
栈与队列

对于数据操作来说，主要就是进行数据的存入或取出，对于不同的数据结构而言，数据的存入或取出操作需要满足该数据结构的要求。



Section 1

栈的实现



对于某种数据结构主要是定义数据的存储方式并定义符合该数据结构的必要操作。

栈：是限定仅在一端进行插入或删除操作的数据结构。



一般操作有：

init ： 初始化

destroy ： 销毁

数据存入：**push**、**enqueue**

数据取出：**pop**、**dequeue**

判断存储区间是否为空、满



栈：是限定仅在一端进行插入或删除操作的数据结构。

栈底：不能进行数据操作的一端

栈顶：进行数据插入、删除的一端



总结：

只能通过栈顶完成数据的存
取

**FILO : first in last
out (后进先出)**



栈的实现：完成操作函数 --- 和具体的数据类型相关

如果栈中元素存储在数组中、存储在链表中则栈的操作函数实现不同，函数接口不需要改变。



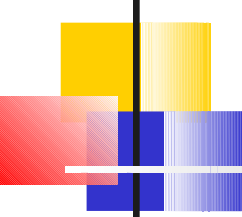
栈的实现：栈中元素存储在数组中
通过数组实现栈操作：
push 或 **pop** 本质上就是对数组
元素的存取操作。



栈元素：存储在数组中

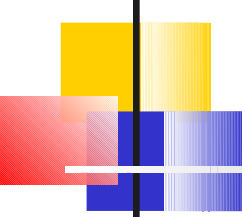
栈顶：数组下标

栈：空栈、满栈



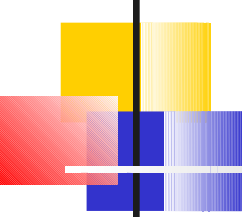
空位置栈：如果栈顶下标表示的是
栈中最后一个有效元素的下一个
位置，则这种栈成为空栈。

空位置栈：栈顶位置可以存元素

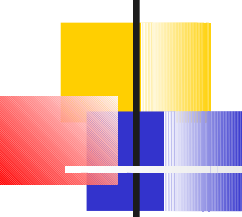


满位置栈：如果栈顶下标表示的是
栈中最后一个有效元素的位置，
则这种栈成为满栈。

满位置栈：栈顶位置已有有效元素



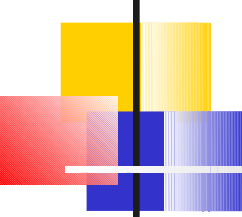
```
typedef int item_t;  
item_t stack[100];  
int top;  
void initstack(void);
```

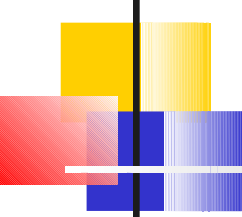



```
void push(item_t item);  
item_t pop(void);  
int is_empty(void);  
int is_full(void);
```



Section 2

- 
-
- 1、通过栈实现 10 进制向 16 进制的转换（只是显示与 10 进制对应的 16 进制形式）**
 - 2、通过栈实现逆波兰表达式**
 - 3、通过栈解决深度搜索迷宫问题**

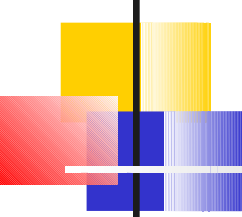


逆波兰表达式（后缀表达式）：每一个运算符都置于其运算对象之后

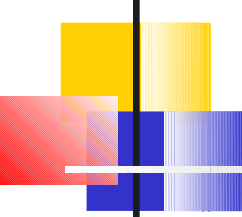
$a + b \rightarrow ab +$

$a + (b - c) \rightarrow abc - +$

$a + (b - c) * d \rightarrow abc - d * +$



逆波兰表达式（后缀表达式）优势
在于只用两种简单操作：入栈和
出栈就可以计算任何普通表达式
的运算。



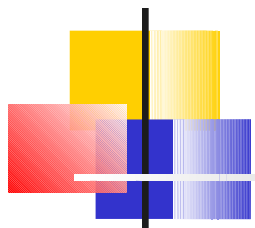
如果当前字符为变量或为数字则压栈，如果是运算符则将栈顶两个元素弹出做相应运算，结果再入栈，最后表达式扫描完后，栈里就是结果。



Section 3

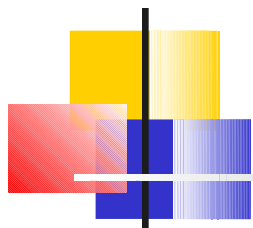
队列

队列的实现



队列的实现

队列：是一种先进先出的数据结构，在队列的一端（队尾）实现数据插入，在另一端（队头）进行数据删除。



队列的实现

队列：是一种先进先出的数据结构，在队列的一端（队尾）实现数据插入，在另一端（队头）进行数据删除。

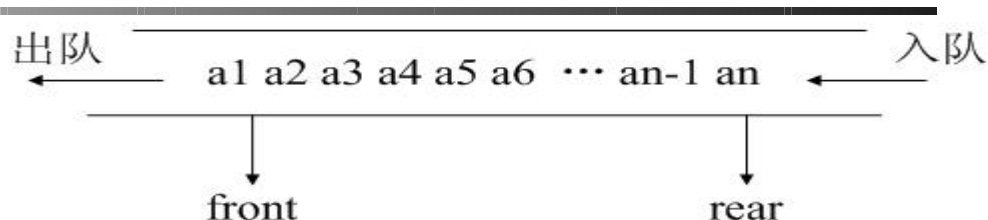
队列的实现

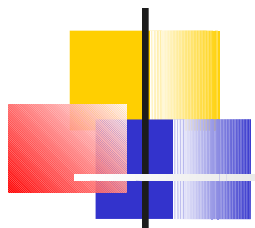
数组实现队列：

队列元素：数组元素

front：队头下标（删除数据）

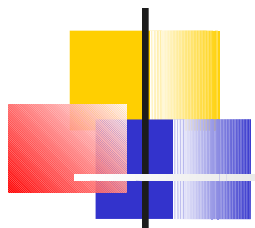
rear：队尾下标（插入数据）





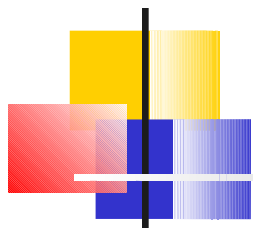
队列的实现

```
typedef int item_t;  
item_t queue[100];  
int front, rear;  
void initqueue(void);
```



队列的实现

```
void enqueue(item_t item);  
item_t dequeue(void);  
int is_empty(void);  
int is_full(void);
```



队列的实现

存在的问题：设数组长度为 **LEN**

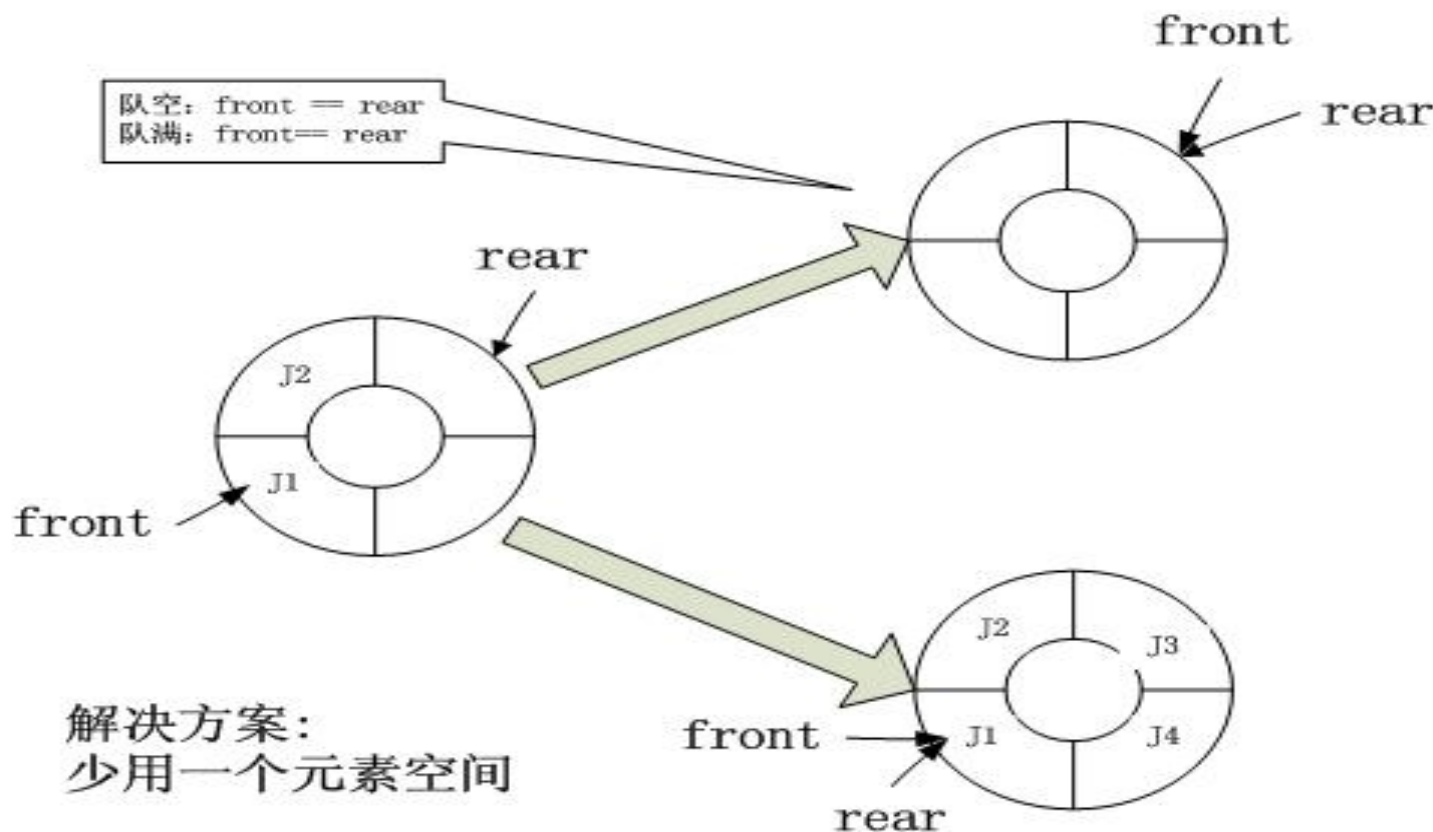
当 **front = 0, rear = LEN - 1**, 再有元素入队将发生溢出——
真溢出

当 **fornt != 0, rear = LEN - 1**, 再有元素入队将发生溢出——
假溢出

解决方案：

循环队列：把队列设想成环形，**num[0]** 与 **num[LEN-1]** 相
邻

队列的实现

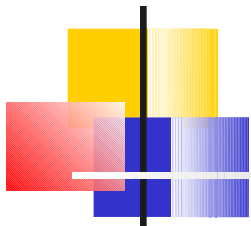




Section 4

队列

队列的应用



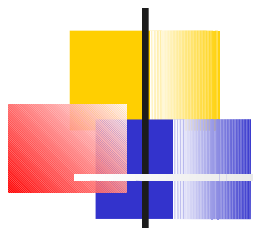
队列的应用

1、队列实现广度搜索迷宫

调试器 gdb

调试器 gdb

- ✓ gdb filename & file filename
- ✓ list
- ✓ run & start
- ✓ next & step
- ✓ quit
- ✓ print & display
- ✓ break & info breakpoint & delete breakpoint n



Let's DO it!

Thanks for listening!

