文章编号:1006-544%(2001)03-0288-04

一种离散事件模拟算法的实现

汤谷云

(广西师范大学数学与计算机科学学院,广西桂林 541004)

摘 要:计算机加工的对象是信息,如何对其正确表示,正确存储,并选择高效率算法进行处理是计算机解决问题的三个步骤。本文以企业商品生产经营管理为例,以上述三大步骤为基础,利用数据结构理论中堆栈和队列这种线性结构的后进先出(LIFO)和先进先出(FIFO)的特殊性,分析如何用计算机对生产经营管理这一过程中的信息进行处理,并对这一离散事件的算法设计模拟过程的实现进行描述。

关键词:堆栈;LIFO;数学模型;数据结构;存储结构

中图分类号: TP311.12 文献标识码: A^①

1 数学模型的建立

1.1 问题描述

在企业生产过程中,存放商品货架往往以堆栈的方式摆放,目的是使生产日期越近的商品越靠近堆栈底,出货时从堆栈顶取货,这样生产出来的商品就可以按时间的先后出售,从而达到对商品生产经营的规范管理。一天营业结束时,如果货架不满,则需要补充货物。如果直接将商品放到货架上,则会使生产日期越近的商品越靠近堆栈顶端,不能满足生产经营的规范要求。要进行规范化的经营就需要倒货架,在补充货物时必须仍使生产日期越近的商品越靠近堆栈底。为了保证每一次上货后始终保持生产日期越近的商品越靠近堆栈底,可用一个队列和一个临时栈作为周转。以一个二元组来表示问题的数学模型^{1]}:

DS- production =
$$(P, R)$$
,

其中:P 为生产的商品集合,R 为各商品间的关系集合,即

$$P = \{P_1, P_2, P_3, \dots, P_i, \dots, P_n\}$$

 $R = \{ \langle P_i, P_{i+1} \rangle \mid i = 1, 2, ..., n-1 \}, \langle P_i, P_{i+1} \rangle$ 为一序偶,商品与商品之间是一种线性关系, P_i 则是一个线性表。从问题的要求来看, P_i 必须在 P_{i+1} 之前卖出。

2.2 抽象数据类型[1]

ADT administration {

数据对象: $D = \{a_i | a_i \in$ 某商品集合, $i = 1, 2, 3, ..., n, n > = 0\}$

数据关系: $R = \{ \langle a_{i-1}, a_i \rangle \mid a_{i-1}, a_i \in D, i = 2, 3, 4, \ldots, n \}$

基本操作:Initial (& S_1 , & S_2 , & S_q)

操作结果:生成2个空栈和1个空队列;

```
Produce ( & S_q , e ) 初始条件:队列已经存在,操作结果:生产产品 e 放入队列;
Turnover ( & S_2 , & S_q ) 初始条件:队列、栈中已经存在商品,操作结果:队列中商品进周转栈;
Customerarrived ( & S_1 , & e ) 初始条件:栈中已经存在商品,操作结果:商品 e 被买走;
Supply ( & S_1 , & S_2 , & S_q ) 初始条件:队列、栈中已经存在商品,操作结果:队列中商品进周转栈之后按要求进入货架栈;
ADT administration.
```

2 系统设计

2.1 数据结构

由 1.1 可知,对商品销售的要求构成了线性结构的特殊性,为此可定义两个堆栈:stack1 为摆放商品的货架,stack2 为周转堆栈; S_q 为周转队列; p_c 为商品变量。生产的商品按生产日期的先后放入临时队列 S_q 中,然后根据 stack1 (货架)的大小,从队列中取出商品入 stack2,再依次进行出栈操作进入 stack1 中,这样就保证了 stack1 (货架)上的商品生产日期越近越靠近栈底。当一天营业结束时,如果 stack1 不满,则先将 stack1 中的剩余货物入 stack2 中,再从队列中取所需数量的商品入 stack2,再将 stack2 中的商品入 stack1 中。

```
# define maxsize 1 000; //最大栈的长度
# define maxqsize 1 500; //最大队列的长度
typedef struct commodity {
    char appellation [8]; //商品名称
    char pdate [8]; //生产日期
} commodity;
typedef struct {
    commodity * base; //初始化动态分配队列空间的首地址
    int front; //队列头指针
    int rear; //队列尾指针
    } Sqqueue; //队列的存储结构
    commodity stack1 [ maxsize ], stack2 [ maxsize ]; //堆栈空间
    int top1, top2; //栈顶指针
```

2.2 总体模块结构

系统主要包括 5 大模块:主程序模块及 4 个子模块,其中补充商品模块还需要调用 turnover()子函数。

设

3

```
补充商品;
    }
   算法主要设计思想:
       \{ 当队列 S_q 不满 ,
        则 EnQueue (Sq) //生产出的商品放入队列;
       {当堆栈 stack2 不满
           {若 not QueueEmpty(Sq)/ 队列 Sq 不空
              则 DeQueue ( & S_q , & Pi );
                  Push ( & stack2, Pi ); //从 Sq 中取商品进入 stack2 中;}
      }
       {当堆栈 stack1 不满
         {若 not StackEmpty (stack2) // 堆栈 stack2 不空
            则 Pop ( & stack2 , & Pi );
              Push ( & stack1, Pi); // stack2 中取商品进入 stack1中;}
         }
     计
在上述数据结构下实现商品生产销售事件模拟的部分算法语句描述:
    void produce()
       {
        while ((S_q. rear + 1)% maxqsize < > S_q. front) // 队列不满
         \{ \sin > pc : appellation ; \}
          cin > pc. pdate;
          S_q. base [ S_q. rear ]. appellation = pc. appellation;
          S_q. base [ S_q. rear ]. pdate = pc. pdate;
          S_q. rear = (S_q. rear + 1)% maxqsize;
          }
        } //模拟生产商品
    void turnover ( )
        while ( top2 < = maxsize )
           {
          if (Sq. front <> Sq. rear)
               { stack2 [ top2 ]. appellation = S_q. base [ S_q. rear ]. appellation;
                stack2 [ top2 ]. pdate = S_q. base [ S_q. rear ]. pdate;
                top2 = top2 + 1;}
        } //队列中商品进周转栈
        while ( top1 < = maxsize )
           {
            if (top2 < > 0)
                 \{ top1 = top1 + 1 ;
```

```
stack1 [top1]. appellation = stack2 [top2]. appellation;
         stack1 [top1]. pdate = stack2 [top2]. pdate;
          top2 = top2 - 1;
                } //临时栈的商品放入货架
          return 0 ;}
void supply()//模拟一天营业结束后上货过程
 {
 if (top1 < maxsize) & & (top1 > 0)
 {
  while (top1 > 0)
     {
      top2 = top2 + 1;
     stack2 [top2]. appellation = stack1 [top1]. appellation;
     stack2 [top2]. pdate = stack1 [top1]. pdate;
      top1 = top1 - 1;
        }
        } // stack1 中的剩余商品先入 stack2 中
      turnover(); //调用周转算法
   if(top1 = 0) turnover(); //若货架上无剩余商品时
```

4 结束语

本系统是理论和实践相结合的一个典型的应用问题,着重考虑的是从现实生活的具体问题中如何提取操作对象,找出它们之间的关系并确定运算方法。抓住了算法中的关键点,合理利用了堆栈的 LIFO 的特性及循环队列的优势,在计算机的理论教学过程中,能够引导学生如何应用理论知识,并提供了一个很好的实践的样板模型。

参考文献:

```
[1] 严蔚敏,吴伟民.数据结构[M].北京:清华大学出版社,1996.18~65.
```

[2] 殷人昆,陶永雷,谢若阳,等.数据结构[M].北京:清华大学出版社,1999.396~397.

The simulating of realization of a discrete event algorithm

TANG Gu-yun

(College of Mathematics and Computer Science, Guangxi Normal University, Guilin 541004, China) Abstract: Correct expression, storge and selection of efficient alsorithm are the three steps for computer processing. Based on the above steps, and making full use of special property ("last in first out" and "first in first out"), a case of administration and management of commodities is discussed to analyze the information process of administration and management of commodities with a simulating algorithm.

Key words: stack; LIFO; model of mathematics; data structure; storage structure