**8、【银行业务活动的模拟】**

问题描述:客户业务分为两种。一种是申请从银行得到一笔资金，即取款或借款。一种是向银行投入一笔资金，即存款或还款。银行有两个服务窗口，相应地有两个队列。客户到达银行后先排第一个队列。处理每个客户业务时，如果属于第一种，且申请额超出银行现存资金总额而得不到满足，则立刻排入第二个队等候，直至满足时才离开银行；否则业务处理完后立刻离开银行。每接待完一个第二种业务的客户，则顺序检查和处理（如果可能）第二个队列中的客户，对能满足的申请者予以满足，不能满足者重新排列第二个队列的队尾。注意，在此检查过程中，一旦银行资金总额少于或等于刚才第一个队列中最后一个客户(第二种业务)被接待之前的数额，或者本次已将第二个队列检查或处理了一遍，就停止检查（因为此时已不可能还有能满足者）转而继续接待第一个对列的客户。任何时刻都只开一个窗口，假设检查不需要时间。营业时间结束时所有客户立即离开银行。

写一个上述银行业务的模拟程序，通过模拟方法求出用户在银行内逗留的平均时间。

要求 :利用链式存储结构实现模拟

**1、【算术表达式求值】**

**问题描述：**一个算术表达式是由操作数(operand)、运算符(operator)和界限符(delimiter)组成的。假设操作数是正整数，运算符只含加减乘除等四种运算符，界限符有左右括号和表达式起始、结束符“#”，如：#（7+15）\*（23-28/4）#。引入表达式起始、结束符是为了方便。编程利用“算符优先法”求算术表达式的值。

**基本要求：**从键盘读入一个合法的算术表达式，输出正确的结果；显示输入序列和栈的变化过程，操作数类型扩充到实数。

"(3.14159/2+sqrt(1/3^2+4)+1/2^2\*ln(1/1.1\*(2+sqrt(1/3^2+4))))\*23.45@";

**选作内容：**扩充运算符集合；引入变量操作数；

**5、二叉排序树和二叉平衡树**

(1)给定一组数

(2)编程实现二叉排序树的创建、插入、删除和查询

(3)编程实现二叉平衡树的创建、插入、删除和查询

(4)对于给定的这组数分别在二叉排序树和二叉平衡树上进行查找，给出两种情况下的查找成功和不成功时的ASL

**4、这是一个古典问题。**

假设有n个修道士和n个野人准备渡河，但只有一条能容纳c人的小船，为了防止野人侵犯修道士，要求无论在何处，修道士的个数不得少于野人的人数（除非修道士个数为0）。如果两种人都会划船，试设计一个算法，确定他们能否渡过河去，若能，则给出一个小船来回次数最少的最佳方案。

要求：

（1）用一个三元组（x1,x2,x3）表示渡河过程中各个状态。其中，x1表示起始岸上修道士个数，x2表示起始岸上野人个数，x3表示小船位置（0——在目的岸，1——在起始岸）。例如（2,1,1）表示起始岸上有两个修道士，一个野人，小船在起始岸一边。采用邻接表做为存储结构，将各种状态之间的迁移图保存下来。

（2）采用广度搜索法，得到首先搜索到的边数最少的一条通路。

（3）输出数据：若问题有解（能渡过河去），则输出一个最佳方案。用三元组表示渡河过程中的状态，并用箭头指出这些状态之间的迁移：

目的状态←…中间状态←…初始状态。

若问题无解，则给出“渡河失败”的信息。

1. 求出所有的解。

提示：

（1）需求分析

* 1. 用一个三元组（x1,x2,x3）表示渡河过程中各个状态。其中，x1表示起始岸上修道士个数，x2表示起始岸上野人个数，x3表示小船位置（0——在目的岸，1——在起始岸）。
  2. 采用广度搜索法，得到首先搜索到的边数最少的一条通路。
  3. 输出数据

若问题有解（能渡过河去），则输出一个最佳方案。用三元组表示渡河过程中的状态，并用箭头指出这些状态之间的迁移：

目的状态←…中间状态←…初始状态

* 1. 求出所有的解。
  2. 程序测试：用户输入修道士与野人个数以及一条船可容纳的人数，则程序输出可行的渡河状态图并输出最优解，程序最后给出可行解的个数。

注意：

1. 程序的输出格式严格按照三元组的形式，给出状态变迁图
2. 必须采用广度搜索算法

（2）设计

**设计思想**

1. 存储结构
   1. 定义一个结构体，用于存放各个时刻的状态

typedef struct

{

int xds;//修道士

int ymr;//野蛮人

int zt;//状态

}DataType;

* 1. 用邻接表存储结构实现图的操作，其存储结构为：

typedef struct Node

{

int dest; //邻接表的弧头结点序号

struct Node \*next;

}Edge; //邻接表单链表的结点结构体

typedef struct

{

DataType data; //结点数据元素

int sorce; //邻接表的弧尾结点序号

Edge \*adj; //邻接边的头指针

int pre; //指向此点的点的序号

}AdjLHeight; //数组的数据元素类型结构体

typedef struct

{

AdjLHeight a[10000]; //邻接表数组

int numOfVerts; //结点个数

int numOfEdges; //边个数

}AdjLGraph; //邻接表结构体

1. 基本思想
2. 由题意知，数据结构选用图较为合理，题中图的结点数目较大且边的数目远小于相同结点的完全图的边数，因此采用图的邻接表存储结构效率较高
3. 由于在每个状态时刻均要判断修道士是否安全，可考虑将安全检查编写为一个函数

**设计表示法**

1. 过程或函数调用关系图

main → work → jiancha → print

1. 基于数据结构的操作组

该程序数据结构相对简单，只运用了邻接表结构的图，work()函数建立一个广度表，实现广度搜索算法；jiancha()函数用于检查各个状态下修道士是否安全；print()函数打印安全渡河的过程

1. 过程或函数接口规格说明

void work(AdjLGraph \*p) //广搜建立表

int jiancha(DataType x) //检查当前情况下，修道士是否安全

int print(AdjLGraph \*p,int g) //打印安全渡河的过程

**（3）测试实例**：

请输入野蛮人和修道士人数N: 3

请输入船可乘人数C: 2

运行结果如下：

( 3 3 1 ) → ( 0 2 ) → ( 3 1 0 )

( 3 1 0 ) ← ( 0 1 ) ← ( 3 2 1 )

( 3 2 1 ) → ( 0 2 ) → ( 3 0 0 )

( 3 0 0 ) ← ( 0 1 ) ← ( 3 1 1 )

( 3 1 1 ) → ( 2 0 ) → ( 1 1 0 )

( 1 1 0 ) ← ( 1 1 ) ← ( 2 2 1 )

( 2 2 1 ) → ( 2 0 ) → ( 0 2 0 )

( 0 2 0 ) ← ( 0 1 ) ← ( 0 3 1 )

( 0 3 1 ) → ( 0 2 ) → ( 0 1 0 )

( 0 1 0 ) ← ( 1 0 ) ← ( 1 1 1 )

( 1 1 1 ) → ( 1 1 ) → ( 0 0 0 )

( 0 0 0 )

渡河成功! …… ……

有4种渡河方案。

**3、【散列文件的插入、删除和查找】**

功能要求：

（1）初始化三列文件；

（2）向散列文件中插入一个元素；

（3）从散列文件中删除一个元素；

（4）从散列文件中查找一个元素。

散列文件通常采用链接法处理冲突，并且把保存每个单链表表头指针的表头向量用一个文件单独存储起来，称此为散列表文件，把所有单链表中的结点用一个文件单独存储起来，称为散列主文件。

散列文件中每个节点的类型定义为：

Struct FLNode { //散列主文件中的节点类型

ElemType data ; //值域

Int next; //指向下一个节点的指针域

};

其中data域用来存储待散列的元素，next域用来存储下一个同义词元素在散列主文件中的存储位置，即所在节点的位置号。