4.3 队列应用-舞伴问题

舞场上，一男一女为一对舞伴。最简单的配对方法为：男、女分别按到场顺序排队，然后男、女依次出队，同时出队的男、女为一对。如果有一队为空，另一队则需等待。

一、程序设计简介

验证程序采用两个属性表示舞者，结构如下：

```
struct dancer                          // 舞者信息
{    string name;                      // 姓名
     char sex;                         // 性别
};
```

用数组 dancer person[]存储所有舞者的信息。

用两个队列分别作男舞者和女舞者的队列。队列采用的是链队。

所有的源码都放在一个文件（DancePartner.cpp）中。根据问题求解需要，除主函数 main()之外，设计了以下 9 个函数：

（1）InitialLinkQueue(LinkQueue &Q)，创建一个空的链队，用于男队、女队的初始化。

（2）DestroyLinkQueue(LinkQueue &Q)，销毁链队，模拟舞会结束时撤消舞者排的队列。

（3）EnQueue(LinkQueue &Q,dancer &e)，入队，模拟舞者加入等待配对的队列。

（4）IsEmpty(LinkQueue Q)，测队空。

（5）DeQueue(LinkQueue &Q,dancer &e)，出队，模拟舞者配对出队，其中调用了 IsEmpty（）判断舞队是否为空，空队不能进行配对。

（6）GetHead(LinkQueue Q,dancer &e)，获取队首元素，模拟获取队首的舞者信息

（7）EntranHall(dancer person[],int num)，模拟舞者到场，由键盘输入舞者信息

（8）DancePartner(dancer person[],int num)，模拟舞者入声、配对的过程。其中调用了函数（1）~（6），调用流程如图 1.4.5 所示。

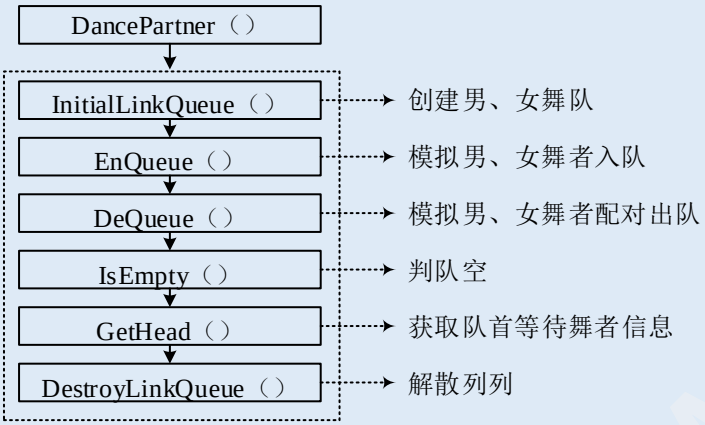


图 1.4.5 舞者配对流程

main（）函数调用函数 EntranHall（）和函数 DancePartner（），分别模拟舞者到场和舞者配对。

二、源程序

```
#include<iostream>
#include<string>
using namespace std;

struct dancer                          // 舞者信息
{     string name;                     // 姓名
      char sex;                        // 性别
};

struct Node                                  // 队列结点
{     dancer data;                     // 数据域
      Node* next;                            // 指针域，指向后继
}*front,*rear;                         // 队头、队尾

struct LinkQueue                       // 舞者队列
{       Node *front;
        Node *rear;
};

void InitialLinkQueue(LinkQueue &Q)            // 初始化队列
{
      Q.front=new Node;
      Q.front->next=NULL;
      Q.rear=Q.front;
}

void DestroyLinkQueue(LinkQueue &Q)            // 销毁队列
```

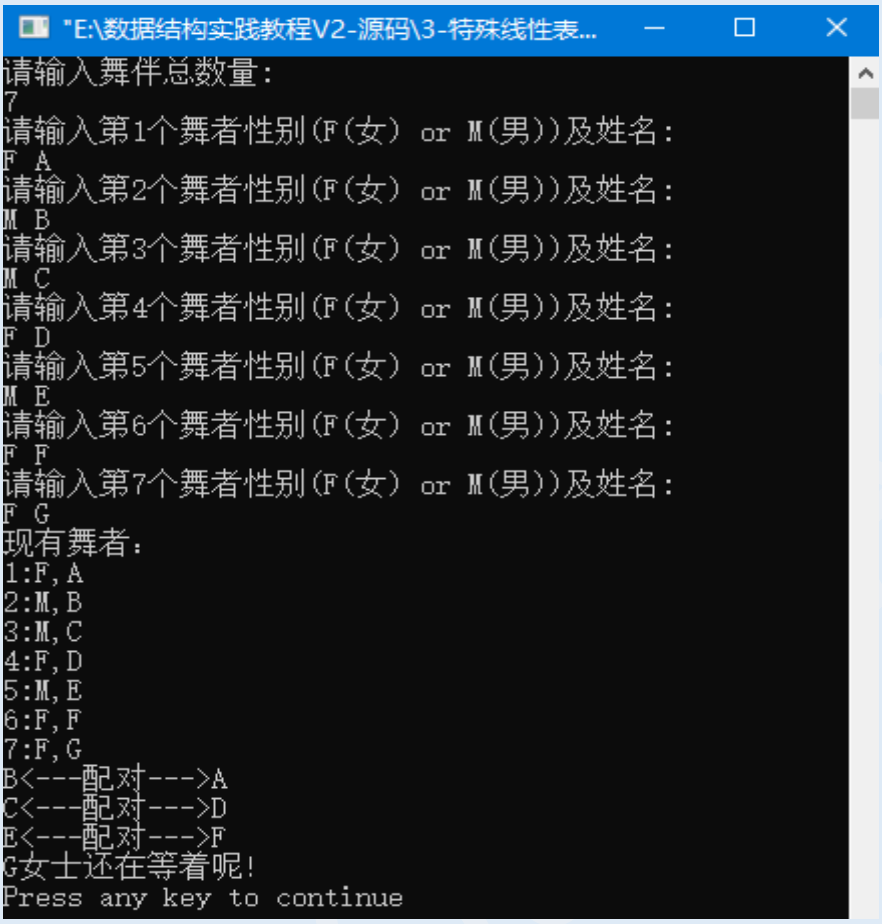
```
{     Node *p;
      while(Q.front!=NULL)
      {     p=Q.front;
            Q.front=Q.front->next;
            delete p;
      }
}

void EnQueue(LinkQueue &Q,dancer &e)                    // 入队
{
      Node *s=new Node;
      s->data=e;
      s->next=Q.rear->next;
      Q.rear->next=s;
      Q.rear=s;
}

bool IsEmpty(LinkQueue Q)                               // 判队空
{
      if (Q.front==Q.rear)
            return true;
      else
            return false;
}

bool DeQueue(LinkQueue &Q,dancer &e)                    // 出队
{
    Node *p;
    if (IsEmpty(Q))                                     // 队空，
    {
          cout<<"队列为空，无法出队列！";
          return false;
    }
    p=Q.front->next;
    e=p->data;
    Q.front->next=p->next;
    if (p==Q.rear)                                      // 只有一个元素，出队
          Q.rear=Q.front;                               // 修改队尾指针
    delete p;
    return true;
}

bool GetHead(LinkQueue Q,dancer &e)                     // 判队空
{     if(IsEmpty(Q))
```

```
    {
        cout<< "队列为空，无法取得队首元素！";
        return false;
    }
    e=Q.front->next->data;                                  // 返回队头元素
    return true;
}

void EntranHall(dancer person[],int num)                    // 舞者到场
{
    int i;
    for(i=0;i<num;i++)
    {
        cout<<"请输入第"<<i+1<<"个舞者性别(F(女) or M(男))及姓名:"<<endl;
        cin>>person[i].sex;
        cin>>person[i].name;
    }
    cout<<"现有舞者："<<endl;
    for(i=0;i<num;i++)
    {
        cout<<i+1<<":"<<person[i].sex
            <<","<<person[i].name<<endl;
    }

}

void DancePartner(dancer person[],int num)   // 算法 3.24  舞者配对
{
    dancer newdancer,m,f,p;
    LinkQueue GenQueue;
    LinkQueue LadyQueue;
    InitialLinkQueue(GenQueue);                                 // 初始化男队
    InitialLinkQueue(LadyQueue);                               // 初始化女队
    for(int i=0;i<num;i++)                                      // 舞者入场
     {   p=person[i];
        if(p.sex=='F')
            EnQueue(LadyQueue,p);
        else
            EnQueue(GenQueue,p);
     }
     while ( (!IsEmpty(GenQueue)) && (!IsEmpty(LadyQueue)) )        // 匹配舞者
     {
         DeQueue(GenQueue,m);                                   // 女士出队
         DeQueue(LadyQueue,f);                                  // 男士出队
```

```
            cout<<m.name <<"<---配对--->"<<f.name<<endl;          // 男、女配队
        }
        if (!IsEmpty(GenQueue))
        {
            GetHead(GenQueue,m);
            cout<<m.name<<"先生还在等着呢!"<<endl;
        }
        else if (!IsEmpty(LadyQueue))
        {
            GetHead(LadyQueue,f);
            cout<<f.name<<"女士还在等着呢!"<<endl;
        }
        else
            cout<<"配对完美结束!"<<endl;
        DestroyLinkQueue(GenQueue);                              // 销毁队列
        DestroyLinkQueue(LadyQueue);
}

int main()                                                       // 主函数
{
        dancer *person;
        int num;
        cout<<"请输入舞伴总数量:"<<endl;
        cin>>num;
        person=new dancer[num];
        EntranHall(person, num);                                 // 舞者入场
        DancePartner(person,num);                                // 舞者匹配
        return 0;
}
```

三、运行说明

一个测试用例的运行过程如图 1.4.6 所示。



1.4.6 舞者配对验证程序运行截图

Step 1：按屏幕提示，输入舞者总数。

Step 2：按屏幕提示，输入各个舞者信息。

Step3：按照已获得的舞者信息，屏幕显示配对信息。

Step 4：按任意键，结束运行。

四、思考题

1. 研读源程序，回答下列问题

（1）表示舞者的属性有几个？其中可以标识舞者的信息吗？

（2）舞者的信息存储在哪里？是在队列里吗？

（3）舞者到场是直接入队的吗？

（4）舞者入队的顺序是如何定的？

（5）如果有一队为空，程序中给出等待配对舞者信息了吗？

（6）分析配对算法的时间和空间复杂度。

2.运行程序，回答下列问题

针对下列情况，给出运行结果：

（7）男、女舞者等数；

（8）男舞者多；

（9）女舞者多。