天津大学

数据结构上机实验报告

题目:模拟病人看病过程解决实际问题

学生姓名 王雨朦

学生学号 2016229082

学院名称 国际工程师学院

专业 计算机

时间 2016/10/8

目　　录

第一章　需求分析.....1
1.1 原题表述.....1
1.2 解决方案.....1
第二章　概要设计.....2
2.1 抽象数据类型.....2
第三章　详细设计.....4
3.1 程序代码.....4
第四章　调试分析.....7
4.1 调试过程.....7
第五章　测试结果.....8
5.1 测试过程.....8

第一章　需求分析

1.1 原题表述

按照如下要求编写一个病人看病模拟程序：

编写一个程序，反映病人到医院看病，排队看医生的情况。在病人排队过程中，主要重复两件事：

1) 病人到达诊室，将病历本交给护士，拍到等待队列中候诊。
2) 护士从等待队列中取出下一位病人的病例，该病人进入诊室就诊。

要求模拟病人等待就诊这一过程，程序采用菜单方式，其选项及功能说明如下：

1) 排队————输入排队病人的病历号，加入到病人排队队列中。
2) 就诊————病人排队队列中最前面的病人就诊，并将其从队列中删除。
3) 查看排队————从队首到队尾列出所有的排队病人的病历号。
4) 不再排队，余下依次就诊————从队首到队尾列出所有的排队病人的病历号，并退出运行。
5) 下班————退出运行。

1.2 解决方案

1) 用顺序队列进行看病过程模拟；
2) 每次新的病人信息加入队列尾部；
3) 取病人信息从队列头取，取出来并删除；
4) 退出程序时，删除队列。

第二章　概要设计

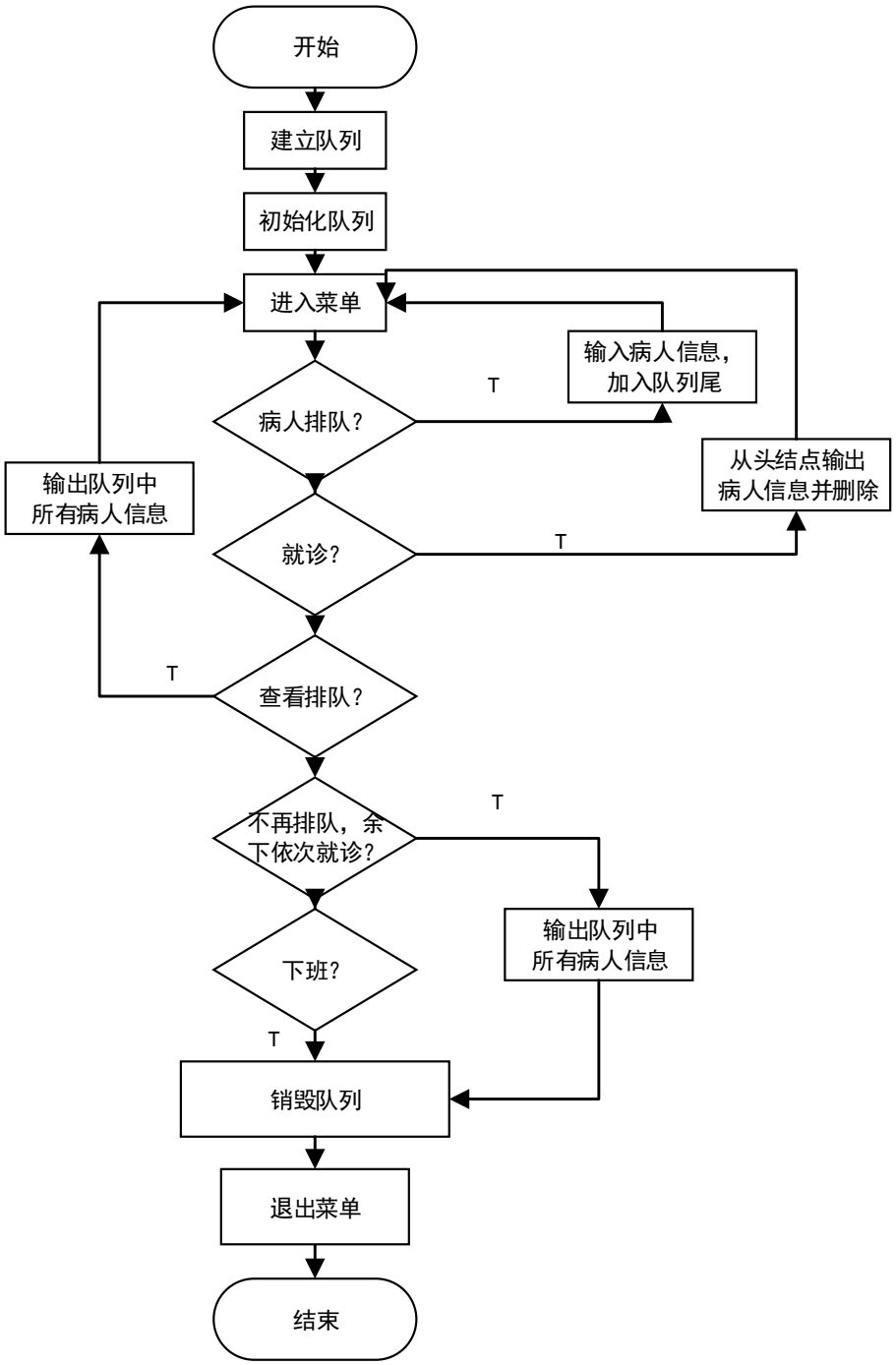
2.1 抽象数据类型

链式队列:

```
#define  MAXSIZE 1000 //  最大队列长度
typedef struct QNode{ //链队列结点的类型定义
    char  data[MAXSIZE];  //用于存放队列元素
    struct QNode *next;  //用于存放元素直接后继
                                    //结点的地址
}QNode,*QueuePtr;
typedef struct
{  //链队列的表头结点的的类型定义
    QueuePtr front;  //队头指针，指向链表的头结点
    QueuePtr rear;   //队尾指针，指向队尾结点
}LinkQueue;

int InitQueue_L(LinkQueue &Q);
int DestroyQueue_Lha(LinkQueue &Q)
int  QueueUp(LinkQueue &Q)
void VisitDoc(LinkQueue &Q)
void display(LinkQueue &q)
```

2.2 算法


开始
建立队列
初始化队列
进入菜单
输入病人信息，
加入队列尾
T
病人排队？
从头结点输出
病人信息并删除
输出队列中
所有病人信息
就诊？
T
T
查看排队？
T
不再排队，余
下依次就诊？
输出队列中
所有病人信息
下班？
T
销毁队列
退出菜单
结束

第三章　详细设计

3.1 程序代码

```
#include <iostream>
#include <stdio.h>
#include <cstdlib>
using namespace std;

#define  MAXSIZE 1000 //  最大队列长度
typedef struct QNode{ //链队列结点的类型定义
    char  data[MAXSIZE];  //用于存放队列元素
    struct QNode *next;  //用于存放元素直接后继
                                    //结点的地址
}QNode,*QueuePtr;
typedef struct
{  //链队列的表头结点的的类型定义
    QueuePtr front;  //队头指针，指向链表的头结点
    QueuePtr rear;   //队尾指针，指向队尾结点
}LinkQueue;

//with head node
int InitQueue_L(LinkQueue &Q) {//建一个空队列
    Q.front=Q.rear=(QueuePtr)malloc(sizeof(QNode));
    //为链队列的头结点分配空间
    if (!Q.front) exit(1);
    Q.front->next=NULL;
    return 1;
} //InitQueue_L

//带头结点
int DestroyQueue_Lha(LinkQueue &Q)  {           //销毁队列 Q
    if (!Q.front) return 0; // 链队列不存在
    QNode * p;
        while(Q.front->next) {   // 回收链队列的所有元素结点空间
            p=Q.front->next;
            Q.front->next=p->next;
            free(p);
        }
        free(Q.front );                         // 回收链队列头结点空间
        Q.front=Q.rear=NULL;
        return 1;
```

```
}//DestroyQueue_L

int  QueueUp(LinkQueue &Q) {      //在链队列 Q 队尾，插入新的队尾结点
    printf("请输入病历号和姓名：\n");

    QueuePtr p=(QueuePtr)malloc(sizeof(QNode));   //为新元素 e 分配结点空间
    if (!p) exit (1);      //存储分配失败
    scanf("%s",p->data);
    p->next=NULL;
    Q.rear->next=p;     //修改队尾指针，插入新队尾结点
    Q.rear=p;
    return 1;
}

void VisitDoc(LinkQueue &Q){ //dequeue
    printf("\n 下一个病人为：\n");
    if (Q.front ==Q.rear) exit (1);
             //若队列空，返回 ERROR
    QueuePtr p=Q.front->next;              // p 指向队头元素结点
    cout<<p->data<<endl;
    //rintf("eeee\n");
    Q.front->next=p->next; // 修改链队列头结点指针
    if(Q.rear == p)  Q.rear=Q.front;
// 对于链队列只有一个元素结点的情况,要同时修改队尾指针
    free(p);
}
void display(LinkQueue &q){
    printf("\n 当前所有排队的病人如下：\n");
    QueuePtr p=q.front->next;
    while(p!=NULL){
        cout<<p->data<<endl;
        p=p->next;
    }
    printf("\n");
}

int main(){
    LinkQueue q;
    InitQueue_L(q);//with head node
    int f=1,i;
    while(f){
        printf("\n 请输入你想进行的操作的序号：\n1、排队\n2、就诊\n3、查看队列\n4、
不再排队，余下依次就诊\n5、下班\n");
        cin>>i;
```

```
        if(i==1)
            QueueUp(q);
        if(i==2)
            VisitDoc(q);
        if(i==3)
            display(q);
        if(i==4){
            display(q);
            DestroyQueue_Lha(q);
            f=0;
        }
        if(i==5){
            DestroyQueue_Lha(q);
            f=0;
        }
    }

}
```

第四章　调试分析

4.1 调试过程

Bug 名称	String 字符串输入错误
Bug 描述	把病人信息定义成 string 类型，用 scanf 无法输入
Bug 原因	String 是 c++中定义的字符类型，无法用 c 输入
Bug 解决方案	将病人信息定义成 char 数组
Bug 总结	基础知识，语言之间的差别不清晰

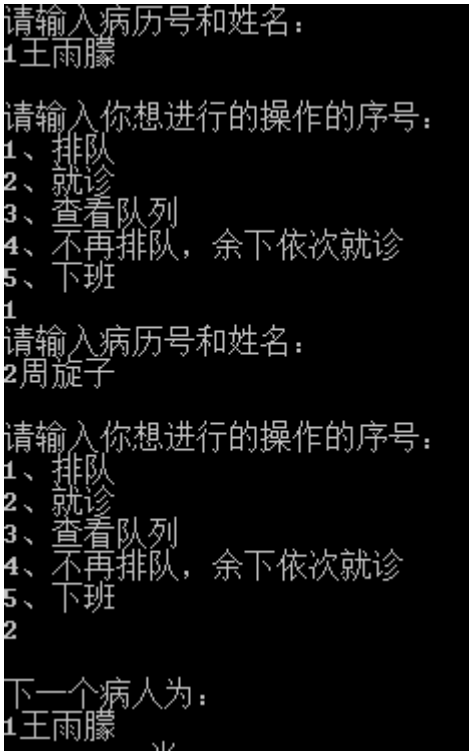
Bug 名称	类型不统一
Bug 描述	类型不一致导致程序中的菜单无线循环
Bug 原因	当结构体中的字符串改为 char 数组时，函数中有的定义的中间变量依然用 string 定义，两处变量类型不一致。
Bug 解决方案	统一定义的变量类型
Bug 总结	关于字符串定义读取和输出的知识不够清楚

Bug 名称	忘记销毁队列
Bug 描述	当菜单中选择下班时，只退出菜单，忘记销毁队列
Bug 原因	在思考程序设计时考虑的不够全面
Bug 解决方案	在菜单中选择不再排队或下班，都需要销毁队列
Bug 总结	此类经验较少，考虑不够周全

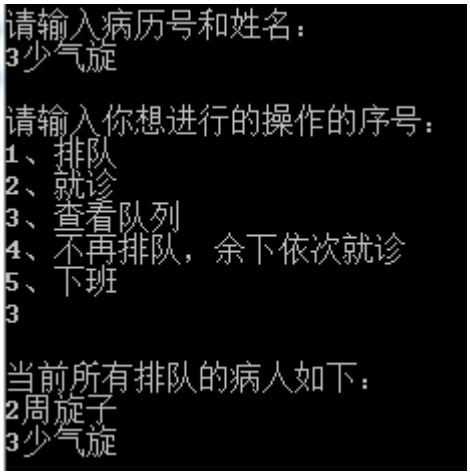
第五章　测试结果

5.1 测试过程

测试编号	1
测试对象	排队和就诊功能
测试输入参数	1 王雨朦 2 周旋子
测试步骤	先输入病人信息，再输出，看是不是正确
测试预期结果	1 王雨朦
测试输出结果	1 王雨朦
测试分析	程序正确



测试编号	2
测试对象	查看队列和不在排队功能
测试输入参数	3 少气旋
测试步骤	先增加一个病人信息，再输出队列信息
测试预期结果	2 周旋子 3 少气旋
测试输出结果	2 周旋子 3 少气旋
测试分析	程序正确



测试编号	3
测试对象	下班
测试输入参数	5
测试步骤	直接输入 5，看是不是退出菜单
测试预期结果	退出了菜单
测试输出结果	退出了菜单
测试分析	程序正确