学生信息管理系统 3.0 实验报告

1 需求分析

1.1 概述

该系统为学生信息管理系统 3.0（C 语言版），该信息包括姓名，性别，年龄，成绩等，可以对学生信息实现增，删，改，查，便于学校对学生的信息进行管理，本系统是基于链表的学生成绩管理系统。

1.2 数据需求

输入的数据需要具有代表性，例如出生日期的输入要加入闰年，平年的判断（主要是二月的判断）以及每个月有多少天的判断，输入学号时要判断之前是否存在。

1.3 功能性需求

演示程序以用户与计算机交互方式执行，即在计算机终端上显示"提示信息"之后，由用户在键盘上输入演示程序中规定的运算命令；相应的输入数据和运算，结果显示在其后。

1.4 其他需求

学生系统的设计需要满足人性化需要，要实现多种功能，每一项功能都要简单明了易懂，避免复杂冗长

2、概要设计

2.1 系统模块划分

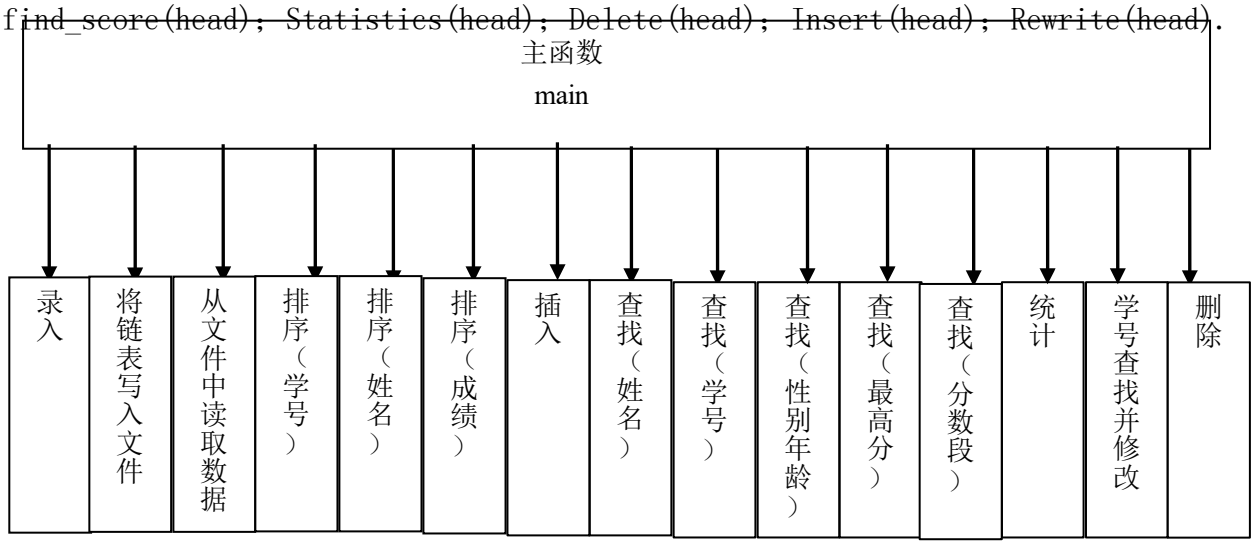
2.1.1 主程序模块

主程序主要是用来调用各个子程序

2.1.2 子程序模块

用了 15 个子程序实现对学生信息的增，删，改，查分别为 creat()；
print(head) ； sort_num(head) ； sort_name(head) ； sort_score(head) ；

find_name(head)； find_num(head)； find_age_sex(head)； find_max_score(head)；
find_score(head); Statistics(head); Delete(head); Insert(head); Rewrite(head).



2.1.3 文件模块

利用文件实现对数据的管理，将学生信息录入在一个文件中就可以防止每次使用系统每次录入学生信息，毕竟每次录入信息都不现实。

2.2 模块设计

2.2.1 主程序模块

主程序主要是显示命令，调用各个函数，还有就是对于用户的密码判断也是在主程序中进行的；

2.2.2 子程序模块

系统要实现的功能都写在子程序中，通过主程序调用子程序来实现函数的调用，防止主程序所包含的太多，运行速度慢；

2.2.3 文件模块

利用文件实现对学生信息的存储，便于对学生信息进行管理。

3.详细设计

3.1 数据定义

```
//***************定义变量*****************
enum sex1 {boy=1,girl};                                    //性别,枚举型
enum  course  {math=1,chinese,english,average};            //科目，枚举型
struct time                                                //时间,结构体数组
{
 int year;
 int month;
 int day;
};
struct stu
{
 char name[20];              // 姓名
 int  num;                   //学号
 enum sex1 sex;              //性别
 float grade[average+1];     //成绩
 struct time birthday;       //生日
 struct stu *next;           //指针类型，指向下一个信息
};  //定义一个结构数组
typedef stu* pstu;
 int current;                //实际输入学生的人数
struct   stu *head;                 //头指针
```

如上所示为变量的定义，使用了枚举型定义性别与科目，结构体数组定义时间以及学生的基本信息，使用指针指向下一个学生信息。

3.2 文件定义

```
//将链表写入文件
void SaveToFile(struct stu * head)
{
    if((fp=fopen("stu.dat","wb"))==NULL)
    {
       printf("cannot open file\n");
       return;
    }
```

```
    p=head;
    do
    {
      fwrite(p, sizeof(struct stu), 1, fp);
      p = p->next;
    }while(p != NULL);
    fclose(fp);
}


//从文件中读取数据放进链表里
struct stu * load()
{
    head=p= (struct stu *)malloc(sizeof(struct stu));
     if((fp=fopen("stu.dat","rb"))==NULL)
    {
       printf("cannot open file\n");
       return(0);
    }
    while (!feof(fp))
    {
        ne = (struct stu *)malloc(sizeof(struct stu));
        if(fread(ne,sizeof(struct
            stu),1,fp)>0){ p->next=ne;
            p=ne;
        }
        current++;
    }
    current--;
    free(ne);
    p->next=NULL;
    fclose(fp);  //文件读取完后,进行链表操作
    return (head);
}
```

使用文件实现了数据的存储，上述为源代码中文件的编写。

3.3 模块详细设计

（注：本系统使用函数 DateJudge 实现年份的判断，在录入信息时调用了此函数）

```
/*****功能：判断年份是否正确；参数：year,month,day 为日期;flag 为判断标志*****/
int DateJudge(int year,int month,int day,int flag)
{
 if (year>1980&&year<2014)     //判断年份，将年份限制在 1880－2013 之间
 {
  switch (month)        //判断月份
  {
   case 1:
   case 3:
   case 5:
   case 7:
   case 8:
   case 10:
   case 12:if (day>31||day<=0)
             flag=0;
             break;          // 1,3,5,7,8,10,12 月份最多的天数为 31 天，输入错误时
flag=0
   case 4:
   case 6:
   case 9:
   case 11:if (day>30||day<=0)
             flag=0;
             break;       //4,6,9,11 月份最多的天数为 30 天，输入错误时 flag=0
   case 2:if(((year % 4 ==0)&&(year %100 !=0))||(year % 400 == 0))  //闰年
          {
           if (day > 29 || day <= 0)  flag=0;
          }
           else if (day > 28 || day <= 0) flag=0;  //平年
          break;
   default: flag=0; break;  //输入错误月份
  }
 }
 else flag=0;     //输入错误月份
 return(flag);  //返回 flag 的值
}

void print(stu *head)
```

```
{
    stu*p=head->next;
    printf("\n  姓名   学号   性别    数学    语文    英语    平均分       出生日期
");
    while (p)
    {
        printf("\n%5s%5d",p->name,p->num);
        switch(p->sex)
          {
           case 1:printf("     男");
                  break;
           case 2:printf("     女");
                  break;
           default:break;
          }

    printf("%8.1f%7.1f%7.1f%8.1f",p->grade[math],p->grade[chinese],p->grade[en
glish],p->grade[average]);
       printf("%9d%3d%3d",p->birthday.year,p->birthday.month,p->birthday.day);
        p=p->next;
    }
}
/******功能：第一次录入学生信息；链表 stu*****/
struct stu *creat()
{
 head=tail=(struct stu *) malloc(LEN);   //申请一个新结点的空间
 head=NULL;            //初始化
 printf("\n 请输入学生人数：");
 scanf("%d", &current);      //输入实际存储学生信息的人数
 for (i=1 ; i <= current; i++)
 {
  p1=(struct stu *) malloc(LEN);
  if (i==1)
   head=p1;
  else tail->next=p1;
  tail=p1;
  tail->next=NULL;
  printf("\n 请输入学号：");
  scanf("%d",&num);
```

```
  p2 = head;
  while (p2)                    //判断学号是否重复
  {
   if (p2->next && p2->num==num)
    {
     printf("此学号已存在，请重新输入 ");
      scanf("%d",&num);
     p2=head;
    }
   if (p2->next && p2->num!=num)
     p2=p2->next;
   if (!p2->next)
    break;
  }
  p1->num=num;
  printf("\n 请输入姓名：");
  scanf("%s",p1->name);
  printf("\n 请输入性别(1 代表男生，其他代表女生)");
  scanf("%d",&xuan_ze);
  switch (xuan_ze)
  {
   case 1:p1->sex=boy;
          break;
   default :p1->sex=girl;
            break;
  }
  printf("\n 请输入数学、语文、英语成绩(0~100 之间)：");
  scanf("%f%f%f",&p1->grade[math],&p1->grade[chinese],&p1->grade[english]);
  while (p1->grade[math] < 0 || p1->grade[math] > 100 || p1->grade[chinese] <
0 || p1->grade[chinese] > 100||    //判断分数是否在 0~100 之间
        p1->grade[english] < 0 || p1->grade[english] > 100)
  {
   printf("\n 输入错误，请重新输入");
   scanf("%f%f%f",&p1->grade[math],&p1->grade[chinese],&p1->grade[english]);
  }

p1->grade[average]=(p1->grade[math]+p1->grade[chinese]+p1->grade[english])/3;
  printf("\n 请输入出生日期(年份应在 1980~2014 之间，如：1992 7 7)");
  scanf("%d%d%d",&p1->birthday.year,&p1->birthday.month,&p1->birthday.day);
```

```
  flag=DateJudge(p1->birthday.year,p1->birthday.month,p1->birthday.day,flag);
//判断日期是否正确
  while (!flag)
  {
   printf("输入错误，请重新输入");
   scanf("%d%d%d",&p1    ->    birthday.year,&p1    ->    birthday.month,&p1    ->
birthday.day);
   flag=DateJudge(p1    ->    birthday.year,p1    ->    birthday.month,p1    ->
birthday.day,flag);
  }
 }
return(head);
}
/*****功能：学号按从小到大排序；参数：链表 stu*****/
void sort_num(stu *head)        //插入法排序
{
    headq=(struct stu *) malloc(LEN);       //新建链表头部
    q=(struct stu *) malloc(LEN);
    q=head;             //q 赋值给以前的头指针
    headq->next=q;      //新建指针，并将指针连接到以前头指针
    p=headq;
    head=head->next;    //head 向后移
    q->next=NULL;       //是新建的链表的尾指针为空
    while (head)      //当 head 不为空时
    {
        while (q && (q->num < head->num))
            //当 p 不为为指针时，若要插入点的学号小于 head 指针，则插入
        {
            p=q;                    //将 q 赋给 p
            q=q->next;
        }
        p->next=head;      //p 连接 head
        m=head->next;     //m 记录 head 下一个指针的数据
        head->next=q;     //将 head 插入到新链表中
        head=m;
        p=headq;          //p 重新回到新链表的头部
        q=headq->next;
    }
    head=headq->next;
```