案例 1 基于链表的学生成绩管理系统

1.1 简介

在有数据库的情况下，将数据存到数据库中是一个很好的选择。但是，在没有数据库的情况下，我们也可以用链表存储数据，并完成增加、删除、修改、查询等数据管理功能。本案例就是开发一个基于链表的学生信息管理系统。在这个系统中，学生信息保存在链表中，可以增加、删除、修改、查询，也可以把链表中的数据保存到一个硬盘文件中，便于下次运行程序时读取。此外，还要求对学生的学号和姓名建立索引，然后根据索引快速查找需要的信息，根据索引以数据递增的次序显示各个记录。这不但复习巩固了折半查找算法，而且对索引有了更直观的认识。

1.2 数据结构的定义

```
typedef struct studentStru
{
    int id;//学号
    char name[30];//姓名
    float score[3];//成绩
    struct studentStru *next;//指向下个结点的指针
}StudentNode, *StudentLink;
typedef struct STUStru
{
    StudentLink Head; //指向表头的指针
    StudentLink Tail; //指向表位的指针
    int count; //已保存的记录的数量
}STU;
StudentLink *IdIndex; //学号索引数组的起始地址
StudentLink *NameIndex; //姓名索引数组的起始地址
```

1.3 算法的实现步骤

1.3.1 链表的相关算法

```
//获得链表的第 i 个元素
Status GetElem(LinkList L,int i,ElemType &e)
 { /* L为带头结点的单链表的头指针。当第i个元素存在时,其值赋给e并返回OK,否则返回
ERROR */
   int j=1; /* j为计数器*/
   LinkList p=L->next; /* p指向第一个结点*/
   while(p&&j<i) /* 顺指针向后查找,直到p指向第i个元素或p为空*/
```

案例 1 基于链表的学生成绩管理系统

1.1 简介

在有数据库的情况下，将数据存到数据库中是一个很好的选择。但是，在没有数据库的情况下，我们也可以用链表存储数据，并完成增加、删除、修改、查询等数据管理功能。本案例就是开发一个基于链表的学生信息管理系统。在这个系统中，学生信息保存在链表中，可以增加、删除、修改、查询，也可以把链表中的数据保存到一个硬盘文件中，便于下次运行程序时读取。此外，还要求对学生的学号和姓名建立索引，然后根据索引快速查找需要的信息，根据索引以数据递增的次序显示各个记录。这不但复习巩固了折半查找算法，而且对索引有了更直观的认识。

1.2 数据结构的定义

```
typedef struct studentStru
{
    int id;//学号
    char name[30];//姓名
    float score[3];//成绩
    struct studentStru *next;//指向下个结点的指针
}StudentNode, *StudentLink;
typedef struct STUStru
{
    StudentLink Head; //指向表头的指针
    StudentLink Tail; //指向表位的指针
    int count; //已保存的记录的数量
}STU;
StudentLink *IdIndex; //学号索引数组的起始地址
StudentLink *NameIndex; //姓名索引数组的起始地址
```

1.3 算法的实现步骤

1.3.1 链表的相关算法

```
//获得链表的第 i 个元素
Status GetElem(LinkList L,int i,ElemType &e)
 { /* L为带头结点的单链表的头指针。当第i个元素存在时,其值赋给e并返回OK,否则返回
ERROR */
   int j=1; /* j为计数器*/
   LinkList p=L->next; /* p指向第一个结点*/
   while(p&&j<i) /* 顺指针向后查找,直到p指向第i个元素或p为空*/
```

案例 1 基于链表的学生成绩管理系统

1.1 简介

在有数据库的情况下，将数据存到数据库中是一个很好的选择。但是，在没有数据库的情况下，我们也可以用链表存储数据，并完成增加、删除、修改、查询等数据管理功能。本案例就是开发一个基于链表的学生信息管理系统。在这个系统中，学生信息保存在链表中，可以增加、删除、修改、查询，也可以把链表中的数据保存到一个硬盘文件中，便于下次运行程序时读取。此外，还要求对学生的学号和姓名建立索引，然后根据索引快速查找需要的信息，根据索引以数据递增的次序显示各个记录。这不但复习巩固了折半查找算法，而且对索引有了更直观的认识。

1.2 数据结构的定义

```
typedef struct studentStru
{
    int id;//学号
    char name[30];//姓名
    float score[3];//成绩
    struct studentStru *next;//指向下个结点的指针
}StudentNode, *StudentLink;
typedef struct STUStru
{
    StudentLink Head; //指向表头的指针
    StudentLink Tail; //指向表位的指针
    int count; //已保存的记录的数量
}STU;
StudentLink *IdIndex; //学号索引数组的起始地址
StudentLink *NameIndex; //姓名索引数组的起始地址
```

1.3 算法的实现步骤

1.3.1 链表的相关算法

```
//获得链表的第 i 个元素
Status GetElem(LinkList L,int i,ElemType &e)
 { /* L为带头结点的单链表的头指针。当第i个元素存在时,其值赋给e并返回OK,否则返回
ERROR */
   int j=1; /* j为计数器*/
   LinkList p=L->next; /* p指向第一个结点*/
   while(p&&j<i) /* 顺指针向后查找,直到p指向第i个元素或p为空*/
```

案例 1 基于链表的学生成绩管理系统

1.1 简介

在有数据库的情况下，将数据存到数据库中是一个很好的选择。但是，在没有数据库的情况下，我们也可以用链表存储数据，并完成增加、删除、修改、查询等数据管理功能。本案例就是开发一个基于链表的学生信息管理系统。在这个系统中，学生信息保存在链表中，可以增加、删除、修改、查询，也可以把链表中的数据保存到一个硬盘文件中，便于下次运行程序时读取。此外，还要求对学生的学号和姓名建立索引，然后根据索引快速查找需要的信息，根据索引以数据递增的次序显示各个记录。这不但复习巩固了折半查找算法，而且对索引有了更直观的认识。

1.2 数据结构的定义

```
typedef struct studentStru
{
    int id;//学号
    char name[30];//姓名
    float score[3];//成绩
    struct studentStru *next;//指向下个结点的指针
}StudentNode, *StudentLink;
typedef struct STUStru
{
    StudentLink Head; //指向表头的指针
    StudentLink Tail; //指向表位的指针
    int count; //已保存的记录的数量
}STU;
StudentLink *IdIndex; //学号索引数组的起始地址
StudentLink *NameIndex; //姓名索引数组的起始地址
```

1.3 算法的实现步骤

1.3.1 链表的相关算法

```
//获得链表的第 i 个元素
Status GetElem(LinkList L,int i,ElemType &e)
 { /* L为带头结点的单链表的头指针。当第i个元素存在时,其值赋给e并返回OK,否则返回
ERROR */
   int j=1; /* j为计数器*/
   LinkList p=L->next; /* p指向第一个结点*/
   while(p&&j<i) /* 顺指针向后查找,直到p指向第i个元素或p为空*/
```

案例 1 基于链表的学生成绩管理系统

1.1 简介

在有数据库的情况下，将数据存到数据库中是一个很好的选择。但是，在没有数据库的情况下，我们也可以用链表存储数据，并完成增加、删除、修改、查询等数据管理功能。本案例就是开发一个基于链表的学生信息管理系统。在这个系统中，学生信息保存在链表中，可以增加、删除、修改、查询，也可以把链表中的数据保存到一个硬盘文件中，便于下次运行程序时读取。此外，还要求对学生的学号和姓名建立索引，然后根据索引快速查找需要的信息，根据索引以数据递增的次序显示各个记录。这不但复习巩固了折半查找算法，而且对索引有了更直观的认识。

1.2 数据结构的定义

```
typedef struct studentStru
{
    int id;//学号
    char name[30];//姓名
    float score[3];//成绩
    struct studentStru *next;//指向下个结点的指针
}StudentNode, *StudentLink;
typedef struct STUStru
{
    StudentLink Head; //指向表头的指针
    StudentLink Tail; //指向表位的指针
    int count; //已保存的记录的数量
}STU;
StudentLink *IdIndex; //学号索引数组的起始地址
StudentLink *NameIndex; //姓名索引数组的起始地址
```

1.3 算法的实现步骤

1.3.1 链表的相关算法

```
//获得链表的第 i 个元素
Status GetElem(LinkList L,int i,ElemType &e)
 { /* L为带头结点的单链表的头指针。当第i个元素存在时,其值赋给e并返回OK,否则返回
ERROR */
   int j=1; /* j为计数器*/
   LinkList p=L->next; /* p指向第一个结点*/
   while(p&&j<i) /* 顺指针向后查找,直到p指向第i个元素或p为空*/
```

案例 1 基于链表的学生成绩管理系统

1.1 简介

在有数据库的情况下，将数据存到数据库中是一个很好的选择。但是，在没有数据库的情况下，我们也可以用链表存储数据，并完成增加、删除、修改、查询等数据管理功能。本案例就是开发一个基于链表的学生信息管理系统。在这个系统中，学生信息保存在链表中，可以增加、删除、修改、查询，也可以把链表中的数据保存到一个硬盘文件中，便于下次运行程序时读取。此外，还要求对学生的学号和姓名建立索引，然后根据索引快速查找需要的信息，根据索引以数据递增的次序显示各个记录。这不但复习巩固了折半查找算法，而且对索引有了更直观的认识。

1.2 数据结构的定义

```
typedef struct studentStru
{
    int id;//学号
    char name[30];//姓名
    float score[3];//成绩
    struct studentStru *next;//指向下个结点的指针
}StudentNode, *StudentLink;
typedef struct STUStru
{
    StudentLink Head; //指向表头的指针
    StudentLink Tail; //指向表位的指针
    int count; //已保存的记录的数量
}STU;
StudentLink *IdIndex; //学号索引数组的起始地址
StudentLink *NameIndex; //姓名索引数组的起始地址
```

1.3 算法的实现步骤

1.3.1 链表的相关算法

```
//获得链表的第 i 个元素
Status GetElem(LinkList L,int i,ElemType &e)
 { /* L为带头结点的单链表的头指针。当第i个元素存在时,其值赋给e并返回OK,否则返回
ERROR */
   int j=1; /* j为计数器*/
   LinkList p=L->next; /* p指向第一个结点*/
   while(p&&j<i) /* 顺指针向后查找,直到p指向第i个元素或p为空*/
```

案例 1 基于链表的学生成绩管理系统

1.1 简介

在有数据库的情况下，将数据存到数据库中是一个很好的选择。但是，在没有数据库的情况下，我们也可以用链表存储数据，并完成增加、删除、修改、查询等数据管理功能。本案例就是开发一个基于链表的学生信息管理系统。在这个系统中，学生信息保存在链表中，可以增加、删除、修改、查询，也可以把链表中的数据保存到一个硬盘文件中，便于下次运行程序时读取。此外，还要求对学生的学号和姓名建立索引，然后根据索引快速查找需要的信息，根据索引以数据递增的次序显示各个记录。这不但复习巩固了折半查找算法，而且对索引有了更直观的认识。

1.2 数据结构的定义

```
typedef struct studentStru
{
    int id;//学号
    char name[30];//姓名
    float score[3];//成绩
    struct studentStru *next;//指向下个结点的指针
}StudentNode, *StudentLink;
typedef struct STUStru
{
    StudentLink Head; //指向表头的指针
    StudentLink Tail; //指向表位的指针
    int count; //已保存的记录的数量
}STU;
StudentLink *IdIndex; //学号索引数组的起始地址
StudentLink *NameIndex; //姓名索引数组的起始地址
```

1.3 算法的实现步骤

1.3.1 链表的相关算法

```
//获得链表的第 i 个元素
Status GetElem(LinkList L,int i,ElemType &e)
 { /* L为带头结点的单链表的头指针。当第i个元素存在时,其值赋给e并返回OK,否则返回
ERROR */
   int j=1; /* j为计数器*/
   LinkList p=L->next; /* p指向第一个结点*/
   while(p&&j<i) /* 顺指针向后查找,直到p指向第i个元素或p为空*/
```

案例 1 基于链表的学生成绩管理系统

1.1 简介

在有数据库的情况下，将数据存到数据库中是一个很好的选择。但是，在没有数据库的情况下，我们也可以用链表存储数据，并完成增加、删除、修改、查询等数据管理功能。本案例就是开发一个基于链表的学生信息管理系统。在这个系统中，学生信息保存在链表中，可以增加、删除、修改、查询，也可以把链表中的数据保存到一个硬盘文件中，便于下次运行程序时读取。此外，还要求对学生的学号和姓名建立索引，然后根据索引快速查找需要的信息，根据索引以数据递增的次序显示各个记录。这不但复习巩固了折半查找算法，而且对索引有了更直观的认识。

1.2 数据结构的定义

```
typedef struct studentStru
{
    int id;//学号
    char name[30];//姓名
    float score[3];//成绩
    struct studentStru *next;//指向下个结点的指针
}StudentNode, *StudentLink;
typedef struct STUStru
{
    StudentLink Head; //指向表头的指针
    StudentLink Tail; //指向表位的指针
    int count; //已保存的记录的数量
}STU;
StudentLink *IdIndex; //学号索引数组的起始地址
StudentLink *NameIndex; //姓名索引数组的起始地址
```

1.3 算法的实现步骤

1.3.1 链表的相关算法

```
//获得链表的第 i 个元素
Status GetElem(LinkList L,int i,ElemType &e)
 { /* L为带头结点的单链表的头指针。当第i个元素存在时,其值赋给e并返回OK,否则返回
ERROR */
   int j=1; /* j为计数器*/
   LinkList p=L->next; /* p指向第一个结点*/
   while(p&&j<i) /* 顺指针向后查找,直到p指向第i个元素或p为空*/
```

案例 1 基于链表的学生成绩管理系统

1.1 简介

在有数据库的情况下，将数据存到数据库中是一个很好的选择。但是，在没有数据库的情况下，我们也可以用链表存储数据，并完成增加、删除、修改、查询等数据管理功能。本案例就是开发一个基于链表的学生信息管理系统。在这个系统中，学生信息保存在链表中，可以增加、删除、修改、查询，也可以把链表中的数据保存到一个硬盘文件中，便于下次运行程序时读取。此外，还要求对学生的学号和姓名建立索引，然后根据索引快速查找需要的信息，根据索引以数据递增的次序显示各个记录。这不但复习巩固了折半查找算法，而且对索引有了更直观的认识。

1.2 数据结构的定义

```
typedef struct studentStru
{
    int id;//学号
    char name[30];//姓名
    float score[3];//成绩
    struct studentStru *next;//指向下个结点的指针
}StudentNode, *StudentLink;
typedef struct STUStru
{
    StudentLink Head; //指向表头的指针
    StudentLink Tail; //指向表位的指针
    int count; //已保存的记录的数量
}STU;
StudentLink *IdIndex; //学号索引数组的起始地址
StudentLink *NameIndex; //姓名索引数组的起始地址
```

1.3 算法的实现步骤

1.3.1 链表的相关算法

```
//获得链表的第 i 个元素
Status GetElem(LinkList L,int i,ElemType &e)
 { /* L为带头结点的单链表的头指针。当第i个元素存在时,其值赋给e并返回OK,否则返回
ERROR */
   int j=1; /* j为计数器*/
   LinkList p=L->next; /* p指向第一个结点*/
   while(p&&j<i) /* 顺指针向后查找,直到p指向第i个元素或p为空*/
```

案例 1 基于链表的学生成绩管理系统

1.1 简介

在有数据库的情况下，将数据存到数据库中是一个很好的选择。但是，在没有数据库的情况下，我们也可以用链表存储数据，并完成增加、删除、修改、查询等数据管理功能。本案例就是开发一个基于链表的学生信息管理系统。在这个系统中，学生信息保存在链表中，可以增加、删除、修改、查询，也可以把链表中的数据保存到一个硬盘文件中，便于下次运行程序时读取。此外，还要求对学生的学号和姓名建立索引，然后根据索引快速查找需要的信息，根据索引以数据递增的次序显示各个记录。这不但复习巩固了折半查找算法，而且对索引有了更直观的认识。

1.2 数据结构的定义

```
typedef struct studentStru
{
    int id;//学号
    char name[30];//姓名
    float score[3];//成绩
    struct studentStru *next;//指向下个结点的指针
}StudentNode, *StudentLink;
typedef struct STUStru
{
    StudentLink Head; //指向表头的指针
    StudentLink Tail; //指向表位的指针
    int count; //已保存的记录的数量
}STU;
StudentLink *IdIndex; //学号索引数组的起始地址
StudentLink *NameIndex; //姓名索引数组的起始地址
```

1.3 算法的实现步骤

1.3.1 链表的相关算法

```
//获得链表的第 i 个元素
Status GetElem(LinkList L,int i,ElemType &e)
 { /* L为带头结点的单链表的头指针。当第i个元素存在时,其值赋给e并返回OK,否则返回
ERROR */
   int j=1; /* j为计数器*/
   LinkList p=L->next; /* p指向第一个结点*/
   while(p&&j<i) /* 顺指针向后查找,直到p指向第i个元素或p为空*/
```

案例 1 基于链表的学生成绩管理系统

1.1 简介

在有数据库的情况下，将数据存到数据库中是一个很好的选择。但是，在没有数据库的情况下，我们也可以用链表存储数据，并完成增加、删除、修改、查询等数据管理功能。本案例就是开发一个基于链表的学生信息管理系统。在这个系统中，学生信息保存在链表中，可以增加、删除、修改、查询，也可以把链表中的数据保存到一个硬盘文件中，便于下次运行程序时读取。此外，还要求对学生的学号和姓名建立索引，然后根据索引快速查找需要的信息，根据索引以数据递增的次序显示各个记录。这不但复习巩固了折半查找算法，而且对索引有了更直观的认识。

1.2 数据结构的定义

```
typedef struct studentStru
{
    int id;//学号
    char name[30];//姓名
    float score[3];//成绩
    struct studentStru *next;//指向下个结点的指针
}StudentNode, *StudentLink;
typedef struct STUStru
{
    StudentLink Head; //指向表头的指针
    StudentLink Tail; //指向表位的指针
    int count; //已保存的记录的数量
}STU;
StudentLink *IdIndex; //学号索引数组的起始地址
StudentLink *NameIndex; //姓名索引数组的起始地址
```

1.3 算法的实现步骤

1.3.1 链表的相关算法

```
//获得链表的第 i 个元素
Status GetElem(LinkList L,int i,ElemType &e)
 { /* L为带头结点的单链表的头指针。当第i个元素存在时,其值赋给e并返回OK,否则返回
ERROR */
   int j=1; /* j为计数器*/
   LinkList p=L->next; /* p指向第一个结点*/
   while(p&&j<i) /* 顺指针向后查找,直到p指向第i个元素或p为空*/
```

案例 1 基于链表的学生成绩管理系统

1.1 简介

在有数据库的情况下，将数据存到数据库中是一个很好的选择。但是，在没有数据库的情况下，我们也可以用链表存储数据，并完成增加、删除、修改、查询等数据管理功能。本案例就是开发一个基于链表的学生信息管理系统。在这个系统中，学生信息保存在链表中，可以增加、删除、修改、查询，也可以把链表中的数据保存到一个硬盘文件中，便于下次运行程序时读取。此外，还要求对学生的学号和姓名建立索引，然后根据索引快速查找需要的信息，根据索引以数据递增的次序显示各个记录。这不但复习巩固了折半查找算法，而且对索引有了更直观的认识。

1.2 数据结构的定义

```
typedef struct studentStru
{
    int id;//学号
    char name[30];//姓名
    float score[3];//成绩
    struct studentStru *next;//指向下个结点的指针
}StudentNode, *StudentLink;
typedef struct STUStru
{
    StudentLink Head; //指向表头的指针
    StudentLink Tail; //指向表位的指针
    int count; //已保存的记录的数量
}STU;
StudentLink *IdIndex; //学号索引数组的起始地址
StudentLink *NameIndex; //姓名索引数组的起始地址
```

1.3 算法的实现步骤

1.3.1 链表的相关算法

```
//获得链表的第 i 个元素
Status GetElem(LinkList L,int i,ElemType &e)
 { /* L为带头结点的单链表的头指针。当第i个元素存在时,其值赋给e并返回OK,否则返回
ERROR */
   int j=1; /* j为计数器*/
   LinkList p=L->next; /* p指向第一个结点*/
   while(p&&j<i) /* 顺指针向后查找,直到p指向第i个元素或p为空*/
```

案例 1 基于链表的学生成绩管理系统

1.1 简介

在有数据库的情况下，将数据存到数据库中是一个很好的选择。但是，在没有数据库的情况下，我们也可以用链表存储数据，并完成增加、删除、修改、查询等数据管理功能。本案例就是开发一个基于链表的学生信息管理系统。在这个系统中，学生信息保存在链表中，可以增加、删除、修改、查询，也可以把链表中的数据保存到一个硬盘文件中，便于下次运行程序时读取。此外，还要求对学生的学号和姓名建立索引，然后根据索引快速查找需要的信息，根据索引以数据递增的次序显示各个记录。这不但复习巩固了折半查找算法，而且对索引有了更直观的认识。

1.2 数据结构的定义

```
typedef struct studentStru
{
    int id;//学号
    char name[30];//姓名
    float score[3];//成绩
    struct studentStru *next;//指向下个结点的指针
}StudentNode, *StudentLink;
typedef struct STUStru
{
    StudentLink Head; //指向表头的指针
    StudentLink Tail; //指向表位的指针
    int count; //已保存的记录的数量
}STU;
StudentLink *IdIndex; //学号索引数组的起始地址
StudentLink *NameIndex; //姓名索引数组的起始地址
```

1.3 算法的实现步骤

1.3.1 链表的相关算法

```
//获得链表的第 i 个元素
Status GetElem(LinkList L,int i,ElemType &e)
 { /* L为带头结点的单链表的头指针。当第i个元素存在时,其值赋给e并返回OK,否则返回
ERROR */
   int j=1; /* j为计数器*/
   LinkList p=L->next; /* p指向第一个结点*/
   while(p&&j<i) /* 顺指针向后查找,直到p指向第i个元素或p为空*/
```

案例 1 基于链表的学生成绩管理系统

1.1 简介

在有数据库的情况下，将数据存到数据库中是一个很好的选择。但是，在没有数据库的情况下，我们也可以用链表存储数据，并完成增加、删除、修改、查询等数据管理功能。本案例就是开发一个基于链表的学生信息管理系统。在这个系统中，学生信息保存在链表中，可以增加、删除、修改、查询，也可以把链表中的数据保存到一个硬盘文件中，便于下次运行程序时读取。此外，还要求对学生的学号和姓名建立索引，然后根据索引快速查找需要的信息，根据索引以数据递增的次序显示各个记录。这不但复习巩固了折半查找算法，而且对索引有了更直观的认识。

1.2 数据结构的定义

```
typedef struct studentStru
{
    int id;//学号
    char name[30];//姓名
    float score[3];//成绩
    struct studentStru *next;//指向下个结点的指针
}StudentNode, *StudentLink;
typedef struct STUStru
{
    StudentLink Head; //指向表头的指针
    StudentLink Tail; //指向表位的指针
    int count; //已保存的记录的数量
}STU;
StudentLink *IdIndex; //学号索引数组的起始地址
StudentLink *NameIndex; //姓名索引数组的起始地址
```

1.3 算法的实现步骤

1.3.1 链表的相关算法

```
//获得链表的第 i 个元素
Status GetElem(LinkList L,int i,ElemType &e)
 { /* L为带头结点的单链表的头指针。当第i个元素存在时,其值赋给e并返回OK,否则返回
ERROR */
   int j=1; /* j为计数器*/
   LinkList p=L->next; /* p指向第一个结点*/
   while(p&&j<i) /* 顺指针向后查找,直到p指向第i个元素或p为空*/
```

案例 1 基于链表的学生成绩管理系统

1.1 简介

在有数据库的情况下，将数据存到数据库中是一个很好的选择。但是，在没有数据库的情况下，我们也可以用链表存储数据，并完成增加、删除、修改、查询等数据管理功能。本案例就是开发一个基于链表的学生信息管理系统。在这个系统中，学生信息保存在链表中，可以增加、删除、修改、查询，也可以把链表中的数据保存到一个硬盘文件中，便于下次运行程序时读取。此外，还要求对学生的学号和姓名建立索引，然后根据索引快速查找需要的信息，根据索引以数据递增的次序显示各个记录。这不但复习巩固了折半查找算法，而且对索引有了更直观的认识。

1.2 数据结构的定义

```
typedef struct studentStru
{
    int id;//学号
    char name[30];//姓名
    float score[3];//成绩
    struct studentStru *next;//指向下个结点的指针
}StudentNode, *StudentLink;
typedef struct STUStru
{
    StudentLink Head; //指向表头的指针
    StudentLink Tail; //指向表位的指针
    int count; //已保存的记录的数量
}STU;
StudentLink *IdIndex; //学号索引数组的起始地址
StudentLink *NameIndex; //姓名索引数组的起始地址
```

1.3 算法的实现步骤

1.3.1 链表的相关算法

```
//获得链表的第 i 个元素
Status GetElem(LinkList L,int i,ElemType &e)
 { /* L为带头结点的单链表的头指针。当第i个元素存在时,其值赋给e并返回OK,否则返回
ERROR */
   int j=1; /* j为计数器*/
   LinkList p=L->next; /* p指向第一个结点*/
   while(p&&j<i) /* 顺指针向后查找,直到p指向第i个元素或p为空*/
```

案例 1 基于链表的学生成绩管理系统

1.1 简介

在有数据库的情况下，将数据存到数据库中是一个很好的选择。但是，在没有数据库的情况下，我们也可以用链表存储数据，并完成增加、删除、修改、查询等数据管理功能。本案例就是开发一个基于链表的学生信息管理系统。在这个系统中，学生信息保存在链表中，可以增加、删除、修改、查询，也可以把链表中的数据保存到一个硬盘文件中，便于下次运行程序时读取。此外，还要求对学生的学号和姓名建立索引，然后根据索引快速查找需要的信息，根据索引以数据递增的次序显示各个记录。这不但复习巩固了折半查找算法，而且对索引有了更直观的认识。

1.2 数据结构的定义

```
typedef struct studentStru
{
    int id;//学号
    char name[30];//姓名
    float score[3];//成绩
    struct studentStru *next;//指向下个结点的指针
}StudentNode, *StudentLink;
typedef struct STUStru
{
    StudentLink Head; //指向表头的指针
    StudentLink Tail; //指向表位的指针
    int count; //已保存的记录的数量
}STU;
StudentLink *IdIndex; //学号索引数组的起始地址
StudentLink *NameIndex; //姓名索引数组的起始地址
```

1.3 算法的实现步骤

1.3.1 链表的相关算法

```
//获得链表的第 i 个元素
Status GetElem(LinkList L,int i,ElemType &e)
 { /* L为带头结点的单链表的头指针。当第i个元素存在时,其值赋给e并返回OK,否则返回
ERROR */
   int j=1; /* j为计数器*/
   LinkList p=L->next; /* p指向第一个结点*/
   while(p&&j<i) /* 顺指针向后查找,直到p指向第i个元素或p为空*/
```

案例 1 基于链表的学生成绩管理系统

1.1 简介

在有数据库的情况下，将数据存到数据库中是一个很好的选择。但是，在没有数据库的情况下，我们也可以用链表存储数据，并完成增加、删除、修改、查询等数据管理功能。本案例就是开发一个基于链表的学生信息管理系统。在这个系统中，学生信息保存在链表中，可以增加、删除、修改、查询，也可以把链表中的数据保存到一个硬盘文件中，便于下次运行程序时读取。此外，还要求对学生的学号和姓名建立索引，然后根据索引快速查找需要的信息，根据索引以数据递增的次序显示各个记录。这不但复习巩固了折半查找算法，而且对索引有了更直观的认识。

1.2 数据结构的定义

```
typedef struct studentStru
{
    int id;//学号
    char name[30];//姓名
    float score[3];//成绩
    struct studentStru *next;//指向下个结点的指针
}StudentNode, *StudentLink;
typedef struct STUStru
{
    StudentLink Head; //指向表头的指针
    StudentLink Tail; //指向表位的指针
    int count; //已保存的记录的数量
}STU;
StudentLink *IdIndex; //学号索引数组的起始地址
StudentLink *NameIndex; //姓名索引数组的起始地址
```

1.3 算法的实现步骤

1.3.1 链表的相关算法

```
//获得链表的第 i 个元素
Status GetElem(LinkList L,int i,ElemType &e)
 { /* L为带头结点的单链表的头指针。当第i个元素存在时,其值赋给e并返回OK,否则返回
ERROR */
   int j=1; /* j为计数器*/
   LinkList p=L->next; /* p指向第一个结点*/
   while(p&&j<i) /* 顺指针向后查找,直到p指向第i个元素或p为空*/
```

案例 1 基于链表的学生成绩管理系统

1.1 简介

在有数据库的情况下，将数据存到数据库中是一个很好的选择。但是，在没有数据库的情况下，我们也可以用链表存储数据，并完成增加、删除、修改、查询等数据管理功能。本案例就是开发一个基于链表的学生信息管理系统。在这个系统中，学生信息保存在链表中，可以增加、删除、修改、查询，也可以把链表中的数据保存到一个硬盘文件中，便于下次运行程序时读取。此外，还要求对学生的学号和姓名建立索引，然后根据索引快速查找需要的信息，根据索引以数据递增的次序显示各个记录。这不但复习巩固了折半查找算法，而且对索引有了更直观的认识。

1.2 数据结构的定义

```
typedef struct studentStru
{
    int id;//学号
    char name[30];//姓名
    float score[3];//成绩
    struct studentStru *next;//指向下个结点的指针
}StudentNode, *StudentLink;
typedef struct STUStru
{
    StudentLink Head; //指向表头的指针
    StudentLink Tail; //指向表位的指针
    int count; //已保存的记录的数量
}STU;
StudentLink *IdIndex; //学号索引数组的起始地址
StudentLink *NameIndex; //姓名索引数组的起始地址
```

1.3 算法的实现步骤

1.3.1 链表的相关算法

```
//获得链表的第 i 个元素
Status GetElem(LinkList L,int i,ElemType &e)
 { /* L为带头结点的单链表的头指针。当第i个元素存在时,其值赋给e并返回OK,否则返回
ERROR */
   int j=1; /* j为计数器*/
   LinkList p=L->next; /* p指向第一个结点*/
   while(p&&j<i) /* 顺指针向后查找,直到p指向第i个元素或p为空*/
```

案例 1 基于链表的学生成绩管理系统

1.1 简介

在有数据库的情况下，将数据存到数据库中是一个很好的选择。但是，在没有数据库的情况下，我们也可以用链表存储数据，并完成增加、删除、修改、查询等数据管理功能。本案例就是开发一个基于链表的学生信息管理系统。在这个系统中，学生信息保存在链表中，可以增加、删除、修改、查询，也可以把链表中的数据保存到一个硬盘文件中，便于下次运行程序时读取。此外，还要求对学生的学号和姓名建立索引，然后根据索引快速查找需要的信息，根据索引以数据递增的次序显示各个记录。这不但复习巩固了折半查找算法，而且对索引有了更直观的认识。

1.2 数据结构的定义

```
typedef struct studentStru
{
    int id;//学号
    char name[30];//姓名
    float score[3];//成绩
    struct studentStru *next;//指向下个结点的指针
}StudentNode, *StudentLink;
typedef struct STUStru
{
    StudentLink Head; //指向表头的指针
    StudentLink Tail; //指向表位的指针
    int count; //已保存的记录的数量
}STU;
StudentLink *IdIndex; //学号索引数组的起始地址
StudentLink *NameIndex; //姓名索引数组的起始地址
```

1.3 算法的实现步骤

1.3.1 链表的相关算法

```
//获得链表的第 i 个元素
Status GetElem(LinkList L,int i,ElemType &e)
 { /* L为带头结点的单链表的头指针。当第i个元素存在时,其值赋给e并返回OK,否则返回
ERROR */
   int j=1; /* j为计数器*/
   LinkList p=L->next; /* p指向第一个结点*/
   while(p&&j<i) /* 顺指针向后查找,直到p指向第i个元素或p为空*/
```

案例 1 基于链表的学生成绩管理系统

1.1 简介

在有数据库的情况下，将数据存到数据库中是一个很好的选择。但是，在没有数据库的情况下，我们也可以用链表存储数据，并完成增加、删除、修改、查询等数据管理功能。本案例就是开发一个基于链表的学生信息管理系统。在这个系统中，学生信息保存在链表中，可以增加、删除、修改、查询，也可以把链表中的数据保存到一个硬盘文件中，便于下次运行程序时读取。此外，还要求对学生的学号和姓名建立索引，然后根据索引快速查找需要的信息，根据索引以数据递增的次序显示各个记录。这不但复习巩固了折半查找算法，而且对索引有了更直观的认识。

1.2 数据结构的定义

```
typedef struct studentStru
{
    int id;//学号
    char name[30];//姓名
    float score[3];//成绩
    struct studentStru *next;//指向下个结点的指针
}StudentNode, *StudentLink;
typedef struct STUStru
{
    StudentLink Head; //指向表头的指针
    StudentLink Tail; //指向表位的指针
    int count; //已保存的记录的数量
}STU;
StudentLink *IdIndex; //学号索引数组的起始地址
StudentLink *NameIndex; //姓名索引数组的起始地址
```

1.3 算法的实现步骤

1.3.1 链表的相关算法

```
//获得链表的第 i 个元素
Status GetElem(LinkList L,int i,ElemType &e)
 { /* L为带头结点的单链表的头指针。当第i个元素存在时,其值赋给e并返回OK,否则返回
ERROR */
   int j=1; /* j为计数器*/
   LinkList p=L->next; /* p指向第一个结点*/
   while(p&&j<i) /* 顺指针向后查找,直到p指向第i个元素或p为空*/
```

案例 1 基于链表的学生成绩管理系统

1.1 简介

在有数据库的情况下，将数据存到数据库中是一个很好的选择。但是，在没有数据库的情况下，我们也可以用链表存储数据，并完成增加、删除、修改、查询等数据管理功能。本案例就是开发一个基于链表的学生信息管理系统。在这个系统中，学生信息保存在链表中，可以增加、删除、修改、查询，也可以把链表中的数据保存到一个硬盘文件中，便于下次运行程序时读取。此外，还要求对学生的学号和姓名建立索引，然后根据索引快速查找需要的信息，根据索引以数据递增的次序显示各个记录。这不但复习巩固了折半查找算法，而且对索引有了更直观的认识。

1.2 数据结构的定义

```
typedef struct studentStru
{
    int id;//学号
    char name[30];//姓名
    float score[3];//成绩
    struct studentStru *next;//指向下个结点的指针
}StudentNode, *StudentLink;
typedef struct STUStru
{
    StudentLink Head; //指向表头的指针
    StudentLink Tail; //指向表位的指针
    int count; //已保存的记录的数量
}STU;
StudentLink *IdIndex; //学号索引数组的起始地址
StudentLink *NameIndex; //姓名索引数组的起始地址
```

1.3 算法的实现步骤

1.3.1 链表的相关算法

```
//获得链表的第 i 个元素
Status GetElem(LinkList L,int i,ElemType &e)
 { /* L为带头结点的单链表的头指针。当第i个元素存在时,其值赋给e并返回OK,否则返回
ERROR */
   int j=1; /* j为计数器*/
   LinkList p=L->next; /* p指向第一个结点*/
   while(p&&j<i) /* 顺指针向后查找,直到p指向第i个元素或p为空*/
```

案例 1 基于链表的学生成绩管理系统

1.1 简介

在有数据库的情况下，将数据存到数据库中是一个很好的选择。但是，在没有数据库的情况下，我们也可以用链表存储数据，并完成增加、删除、修改、查询等数据管理功能。本案例就是开发一个基于链表的学生信息管理系统。在这个系统中，学生信息保存在链表中，可以增加、删除、修改、查询，也可以把链表中的数据保存到一个硬盘文件中，便于下次运行程序时读取。此外，还要求对学生的学号和姓名建立索引，然后根据索引快速查找需要的信息，根据索引以数据递增的次序显示各个记录。这不但复习巩固了折半查找算法，而且对索引有了更直观的认识。

1.2 数据结构的定义

```
typedef struct studentStru
{
    int id;//学号
    char name[30];//姓名
    float score[3];//成绩
    struct studentStru *next;//指向下个结点的指针
}StudentNode, *StudentLink;
typedef struct STUStru
{
    StudentLink Head; //指向表头的指针
    StudentLink Tail; //指向表位的指针
    int count; //已保存的记录的数量
}STU;
StudentLink *IdIndex; //学号索引数组的起始地址
StudentLink *NameIndex; //姓名索引数组的起始地址
```

1.3 算法的实现步骤

1.3.1 链表的相关算法

```
//获得链表的第 i 个元素
Status GetElem(LinkList L,int i,ElemType &e)
 { /* L为带头结点的单链表的头指针。当第i个元素存在时,其值赋给e并返回OK,否则返回
ERROR */
   int j=1; /* j为计数器*/
   LinkList p=L->next; /* p指向第一个结点*/
   while(p&&j<i) /* 顺指针向后查找,直到p指向第i个元素或p为空*/
```

案例 1 基于链表的学生成绩管理系统

1.1 简介

在有数据库的情况下，将数据存到数据库中是一个很好的选择。但是，在没有数据库的情况下，我们也可以用链表存储数据，并完成增加、删除、修改、查询等数据管理功能。本案例就是开发一个基于链表的学生信息管理系统。在这个系统中，学生信息保存在链表中，可以增加、删除、修改、查询，也可以把链表中的数据保存到一个硬盘文件中，便于下次运行程序时读取。此外，还要求对学生的学号和姓名建立索引，然后根据索引快速查找需要的信息，根据索引以数据递增的次序显示各个记录。这不但复习巩固了折半查找算法，而且对索引有了更直观的认识。

1.2 数据结构的定义

```
typedef struct studentStru
{
    int id;//学号
    char name[30];//姓名
    float score[3];//成绩
    struct studentStru *next;//指向下个结点的指针
}StudentNode, *StudentLink;
typedef struct STUStru
{
    StudentLink Head; //指向表头的指针
    StudentLink Tail; //指向表位的指针
    int count; //已保存的记录的数量
}STU;
StudentLink *IdIndex; //学号索引数组的起始地址
StudentLink *NameIndex; //姓名索引数组的起始地址
```

1.3 算法的实现步骤

1.3.1 链表的相关算法

```
//获得链表的第 i 个元素
Status GetElem(LinkList L,int i,ElemType &e)
 { /* L为带头结点的单链表的头指针。当第i个元素存在时,其值赋给e并返回OK,否则返回
ERROR */
   int j=1; /* j为计数器*/
   LinkList p=L->next; /* p指向第一个结点*/
   while(p&&j<i) /* 顺指针向后查找,直到p指向第i个元素或p为空*/
```

案例 1 基于链表的学生成绩管理系统

1.1 简介

在有数据库的情况下，将数据存到数据库中是一个很好的选择。但是，在没有数据库的情况下，我们也可以用链表存储数据，并完成增加、删除、修改、查询等数据管理功能。本案例就是开发一个基于链表的学生信息管理系统。在这个系统中，学生信息保存在链表中，可以增加、删除、修改、查询，也可以把链表中的数据保存到一个硬盘文件中，便于下次运行程序时读取。此外，还要求对学生的学号和姓名建立索引，然后根据索引快速查找需要的信息，根据索引以数据递增的次序显示各个记录。这不但复习巩固了折半查找算法，而且对索引有了更直观的认识。

1.2 数据结构的定义

```
typedef struct studentStru
{
    int id;//学号
    char name[30];//姓名
    float score[3];//成绩
    struct studentStru *next;//指向下个结点的指针
}StudentNode, *StudentLink;
typedef struct STUStru
{
    StudentLink Head; //指向表头的指针
    StudentLink Tail; //指向表位的指针
    int count; //已保存的记录的数量
}STU;
StudentLink *IdIndex; //学号索引数组的起始地址
StudentLink *NameIndex; //姓名索引数组的起始地址
```

1.3 算法的实现步骤

1.3.1 链表的相关算法

```
//获得链表的第 i 个元素
Status GetElem(LinkList L,int i,ElemType &e)
 { /* L为带头结点的单链表的头指针。当第i个元素存在时,其值赋给e并返回OK,否则返回
ERROR */
   int j=1; /* j为计数器*/
   LinkList p=L->next; /* p指向第一个结点*/
   while(p&&j<i) /* 顺指针向后查找,直到p指向第i个元素或p为空*/
```

案例 1 基于链表的学生成绩管理系统

1.1 简介

在有数据库的情况下，将数据存到数据库中是一个很好的选择。但是，在没有数据库的情况下，我们也可以用链表存储数据，并完成增加、删除、修改、查询等数据管理功能。本案例就是开发一个基于链表的学生信息管理系统。在这个系统中，学生信息保存在链表中，可以增加、删除、修改、查询，也可以把链表中的数据保存到一个硬盘文件中，便于下次运行程序时读取。此外，还要求对学生的学号和姓名建立索引，然后根据索引快速查找需要的信息，根据索引以数据递增的次序显示各个记录。这不但复习巩固了折半查找算法，而且对索引有了更直观的认识。

1.2 数据结构的定义

```
typedef struct studentStru
{
    int id;//学号
    char name[30];//姓名
    float score[3];//成绩
    struct studentStru *next;//指向下个结点的指针
}StudentNode, *StudentLink;
typedef struct STUStru
{
    StudentLink Head; //指向表头的指针
    StudentLink Tail; //指向表位的指针
    int count; //已保存的记录的数量
}STU;
StudentLink *IdIndex; //学号索引数组的起始地址
StudentLink *NameIndex; //姓名索引数组的起始地址
```

1.3 算法的实现步骤

1.3.1 链表的相关算法

```
//获得链表的第 i 个元素
Status GetElem(LinkList L,int i,ElemType &e)
 { /* L为带头结点的单链表的头指针。当第i个元素存在时,其值赋给e并返回OK,否则返回
ERROR */
   int j=1; /* j为计数器*/
   LinkList p=L->next; /* p指向第一个结点*/
   while(p&&j<i) /* 顺指针向后查找,直到p指向第i个元素或p为空*/
```

案例 1 基于链表的学生成绩管理系统

1.1 简介

在有数据库的情况下，将数据存到数据库中是一个很好的选择。但是，在没有数据库的情况下，我们也可以用链表存储数据，并完成增加、删除、修改、查询等数据管理功能。本案例就是开发一个基于链表的学生信息管理系统。在这个系统中，学生信息保存在链表中，可以增加、删除、修改、查询，也可以把链表中的数据保存到一个硬盘文件中，便于下次运行程序时读取。此外，还要求对学生的学号和姓名建立索引，然后根据索引快速查找需要的信息，根据索引以数据递增的次序显示各个记录。这不但复习巩固了折半查找算法，而且对索引有了更直观的认识。

1.2 数据结构的定义

```
typedef struct studentStru
{
    int id;//学号
    char name[30];//姓名
    float score[3];//成绩
    struct studentStru *next;//指向下个结点的指针
}StudentNode, *StudentLink;
typedef struct STUStru
{
    StudentLink Head; //指向表头的指针
    StudentLink Tail; //指向表位的指针
    int count; //已保存的记录的数量
}STU;
StudentLink *IdIndex; //学号索引数组的起始地址
StudentLink *NameIndex; //姓名索引数组的起始地址
```

1.3 算法的实现步骤

1.3.1 链表的相关算法

```
//获得链表的第 i 个元素
Status GetElem(LinkList L,int i,ElemType &e)
 { /* L为带头结点的单链表的头指针。当第i个元素存在时,其值赋给e并返回OK,否则返回
ERROR */
   int j=1; /* j为计数器*/
   LinkList p=L->next; /* p指向第一个结点*/
   while(p&&j<i) /* 顺指针向后查找,直到p指向第i个元素或p为空*/
```

案例 1 基于链表的学生成绩管理系统

1.1 简介

在有数据库的情况下，将数据存到数据库中是一个很好的选择。但是，在没有数据库的情况下，我们也可以用链表存储数据，并完成增加、删除、修改、查询等数据管理功能。本案例就是开发一个基于链表的学生信息管理系统。在这个系统中，学生信息保存在链表中，可以增加、删除、修改、查询，也可以把链表中的数据保存到一个硬盘文件中，便于下次运行程序时读取。此外，还要求对学生的学号和姓名建立索引，然后根据索引快速查找需要的信息，根据索引以数据递增的次序显示各个记录。这不但复习巩固了折半查找算法，而且对索引有了更直观的认识。

1.2 数据结构的定义

```
typedef struct studentStru
{
    int id;//学号
    char name[30];//姓名
    float score[3];//成绩
    struct studentStru *next;//指向下个结点的指针
}StudentNode, *StudentLink;
typedef struct STUStru
{
    StudentLink Head; //指向表头的指针
    StudentLink Tail; //指向表位的指针
    int count; //已保存的记录的数量
}STU;
StudentLink *IdIndex; //学号索引数组的起始地址
StudentLink *NameIndex; //姓名索引数组的起始地址
```

1.3 算法的实现步骤

1.3.1 链表的相关算法

```
//获得链表的第 i 个元素
Status GetElem(LinkList L,int i,ElemType &e)
 { /* L为带头结点的单链表的头指针。当第i个元素存在时,其值赋给e并返回OK,否则返回
ERROR */
   int j=1; /* j为计数器*/
   LinkList p=L->next; /* p指向第一个结点*/
   while(p&&j<i) /* 顺指针向后查找,直到p指向第i个元素或p为空*/
```

案例 1 基于链表的学生成绩管理系统

1.1 简介

在有数据库的情况下，将数据存到数据库中是一个很好的选择。但是，在没有数据库的情况下，我们也可以用链表存储数据，并完成增加、删除、修改、查询等数据管理功能。本案例就是开发一个基于链表的学生信息管理系统。在这个系统中，学生信息保存在链表中，可以增加、删除、修改、查询，也可以把链表中的数据保存到一个硬盘文件中，便于下次运行程序时读取。此外，还要求对学生的学号和姓名建立索引，然后根据索引快速查找需要的信息，根据索引以数据递增的次序显示各个记录。这不但复习巩固了折半查找算法，而且对索引有了更直观的认识。

1.2 数据结构的定义

```
typedef struct studentStru
{
    int id;//学号
    char name[30];//姓名
    float score[3];//成绩
    struct studentStru *next;//指向下个结点的指针
}StudentNode, *StudentLink;
typedef struct STUStru
{
    StudentLink Head; //指向表头的指针
    StudentLink Tail; //指向表位的指针
    int count; //已保存的记录的数量
}STU;
StudentLink *IdIndex; //学号索引数组的起始地址
StudentLink *NameIndex; //姓名索引数组的起始地址
```

1.3 算法的实现步骤

1.3.1 链表的相关算法

```
//获得链表的第 i 个元素
Status GetElem(LinkList L,int i,ElemType &e)
 { /* L为带头结点的单链表的头指针。当第i个元素存在时,其值赋给e并返回OK,否则返回
ERROR */
   int j=1; /* j为计数器*/
   LinkList p=L->next; /* p指向第一个结点*/
   while(p&&j<i) /* 顺指针向后查找,直到p指向第i个元素或p为空*/
```

案例 1 基于链表的学生成绩管理系统

1.1 简介

在有数据库的情况下，将数据存到数据库中是一个很好的选择。但是，在没有数据库的情况下，我们也可以用链表存储数据，并完成增加、删除、修改、查询等数据管理功能。本案例就是开发一个基于链表的学生信息管理系统。在这个系统中，学生信息保存在链表中，可以增加、删除、修改、查询，也可以把链表中的数据保存到一个硬盘文件中，便于下次运行程序时读取。此外，还要求对学生的学号和姓名建立索引，然后根据索引快速查找需要的信息，根据索引以数据递增的次序显示各个记录。这不但复习巩固了折半查找算法，而且对索引有了更直观的认识。

1.2 数据结构的定义

```
typedef struct studentStru
{
    int id;//学号
    char name[30];//姓名
    float score[3];//成绩
    struct studentStru *next;//指向下个结点的指针
}StudentNode, *StudentLink;
typedef struct STUStru
{
    StudentLink Head; //指向表头的指针
    StudentLink Tail; //指向表位的指针
    int count; //已保存的记录的数量
}STU;
StudentLink *IdIndex; //学号索引数组的起始地址
StudentLink *NameIndex; //姓名索引数组的起始地址
```

1.3 算法的实现步骤

1.3.1 链表的相关算法

```
//获得链表的第 i 个元素
Status GetElem(LinkList L,int i,ElemType &e)
 { /* L为带头结点的单链表的头指针。当第i个元素存在时,其值赋给e并返回OK,否则返回
ERROR */
   int j=1; /* j为计数器*/
   LinkList p=L->next; /* p指向第一个结点*/
   while(p&&j<i) /* 顺指针向后查找,直到p指向第i个元素或p为空*/
```

案例 1 基于链表的学生成绩管理系统

1.1 简介

在有数据库的情况下，将数据存到数据库中是一个很好的选择。但是，在没有数据库的情况下，我们也可以用链表存储数据，并完成增加、删除、修改、查询等数据管理功能。本案例就是开发一个基于链表的学生信息管理系统。在这个系统中，学生信息保存在链表中，可以增加、删除、修改、查询，也可以把链表中的数据保存到一个硬盘文件中，便于下次运行程序时读取。此外，还要求对学生的学号和姓名建立索引，然后根据索引快速查找需要的信息，根据索引以数据递增的次序显示各个记录。这不但复习巩固了折半查找算法，而且对索引有了更直观的认识。

1.2 数据结构的定义

```
typedef struct studentStru
{
    int id;//学号
    char name[30];//姓名
    float score[3];//成绩
    struct studentStru *next;//指向下个结点的指针
}StudentNode, *StudentLink;
typedef struct STUStru
{
    StudentLink Head; //指向表头的指针
    StudentLink Tail; //指向表位的指针
    int count; //已保存的记录的数量
}STU;
StudentLink *IdIndex; //学号索引数组的起始地址
StudentLink *NameIndex; //姓名索引数组的起始地址
```

1.3 算法的实现步骤

1.3.1 链表的相关算法

```
//获得链表的第 i 个元素
Status GetElem(LinkList L,int i,ElemType &e)
 { /* L为带头结点的单链表的头指针。当第i个元素存在时,其值赋给e并返回OK,否则返回
ERROR */
   int j=1; /* j为计数器*/
   LinkList p=L->next; /* p指向第一个结点*/
   while(p&&j<i) /* 顺指针向后查找,直到p指向第i个元素或p为空*/
```

案例 1 基于链表的学生成绩管理系统

1.1 简介

在有数据库的情况下，将数据存到数据库中是一个很好的选择。但是，在没有数据库的情况下，我们也可以用链表存储数据，并完成增加、删除、修改、查询等数据管理功能。本案例就是开发一个基于链表的学生信息管理系统。在这个系统中，学生信息保存在链表中，可以增加、删除、修改、查询，也可以把链表中的数据保存到一个硬盘文件中，便于下次运行程序时读取。此外，还要求对学生的学号和姓名建立索引，然后根据索引快速查找需要的信息，根据索引以数据递增的次序显示各个记录。这不但复习巩固了折半查找算法，而且对索引有了更直观的认识。

1.2 数据结构的定义

```
typedef struct studentStru
{
    int id;//学号
    char name[30];//姓名
    float score[3];//成绩
    struct studentStru *next;//指向下个结点的指针
}StudentNode, *StudentLink;
typedef struct STUStru
{
    StudentLink Head; //指向表头的指针
    StudentLink Tail; //指向表位的指针
    int count; //已保存的记录的数量
}STU;
StudentLink *IdIndex; //学号索引数组的起始地址
StudentLink *NameIndex; //姓名索引数组的起始地址
```

1.3 算法的实现步骤

1.3.1 链表的相关算法

```
//获得链表的第 i 个元素
Status GetElem(LinkList L,int i,ElemType &e)
 { /* L为带头结点的单链表的头指针。当第i个元素存在时,其值赋给e并返回OK,否则返回
ERROR */
   int j=1; /* j为计数器*/
   LinkList p=L->next; /* p指向第一个结点*/
   while(p&&j<i) /* 顺指针向后查找,直到p指向第i个元素或p为空*/
```

案例 1 基于链表的学生成绩管理系统

1.1 简介

在有数据库的情况下，将数据存到数据库中是一个很好的选择。但是，在没有数据库的情况下，我们也可以用链表存储数据，并完成增加、删除、修改、查询等数据管理功能。本案例就是开发一个基于链表的学生信息管理系统。在这个系统中，学生信息保存在链表中，可以增加、删除、修改、查询，也可以把链表中的数据保存到一个硬盘文件中，便于下次运行程序时读取。此外，还要求对学生的学号和姓名建立索引，然后根据索引快速查找需要的信息，根据索引以数据递增的次序显示各个记录。这不但复习巩固了折半查找算法，而且对索引有了更直观的认识。

1.2 数据结构的定义

```
typedef struct studentStru
{
    int id;//学号
    char name[30];//姓名
    float score[3];//成绩
    struct studentStru *next;//指向下个结点的指针
}StudentNode, *StudentLink;
typedef struct STUStru
{
    StudentLink Head; //指向表头的指针
    StudentLink Tail; //指向表位的指针
    int count; //已保存的记录的数量
}STU;
StudentLink *IdIndex; //学号索引数组的起始地址
StudentLink *NameIndex; //姓名索引数组的起始地址
```

1.3 算法的实现步骤

1.3.1 链表的相关算法

```
//获得链表的第 i 个元素
Status GetElem(LinkList L,int i,ElemType &e)
 { /* L为带头结点的单链表的头指针。当第i个元素存在时,其值赋给e并返回OK,否则返回
ERROR */
   int j=1; /* j为计数器*/
   LinkList p=L->next; /* p指向第一个结点*/
   while(p&&j<i) /* 顺指针向后查找,直到p指向第i个元素或p为空*/
```

案例 1 基于链表的学生成绩管理系统

1.1 简介

在有数据库的情况下，将数据存到数据库中是一个很好的选择。但是，在没有数据库的情况下，我们也可以用链表存储数据，并完成增加、删除、修改、查询等数据管理功能。本案例就是开发一个基于链表的学生信息管理系统。在这个系统中，学生信息保存在链表中，可以增加、删除、修改、查询，也可以把链表中的数据保存到一个硬盘文件中，便于下次运行程序时读取。此外，还要求对学生的学号和姓名建立索引，然后根据索引快速查找需要的信息，根据索引以数据递增的次序显示各个记录。这不但复习巩固了折半查找算法，而且对索引有了更直观的认识。

1.2 数据结构的定义

```
typedef struct studentStru
{
    int id;//学号
    char name[30];//姓名
    float score[3];//成绩
    struct studentStru *next;//指向下个结点的指针
}StudentNode, *StudentLink;
typedef struct STUStru
{
    StudentLink Head; //指向表头的指针
    StudentLink Tail; //指向表位的指针
    int count; //已保存的记录的数量
}STU;
StudentLink *IdIndex; //学号索引数组的起始地址
StudentLink *NameIndex; //姓名索引数组的起始地址
```

1.3 算法的实现步骤

1.3.1 链表的相关算法

```
//获得链表的第 i 个元素
Status GetElem(LinkList L,int i,ElemType &e)
 { /* L为带头结点的单链表的头指针。当第i个元素存在时,其值赋给e并返回OK,否则返回
ERROR */
   int j=1; /* j为计数器*/
   LinkList p=L->next; /* p指向第一个结点*/
   while(p&&j<i) /* 顺指针向后查找,直到p指向第i个元素或p为空*/
```

案例 1 基于链表的学生成绩管理系统

1.1 简介

在有数据库的情况下，将数据存到数据库中是一个很好的选择。但是，在没有数据库的情况下，我们也可以用链表存储数据，并完成增加、删除、修改、查询等数据管理功能。本案例就是开发一个基于链表的学生信息管理系统。在这个系统中，学生信息保存在链表中，可以增加、删除、修改、查询，也可以把链表中的数据保存到一个硬盘文件中，便于下次运行程序时读取。此外，还要求对学生的学号和姓名建立索引，然后根据索引快速查找需要的信息，根据索引以数据递增的次序显示各个记录。这不但复习巩固了折半查找算法，而且对索引有了更直观的认识。

1.2 数据结构的定义

```
typedef struct studentStru
{
    int id;//学号
    char name[30];//姓名
    float score[3];//成绩
    struct studentStru *next;//指向下个结点的指针
}StudentNode, *StudentLink;
typedef struct STUStru
{
    StudentLink Head; //指向表头的指针
    StudentLink Tail; //指向表位的指针
    int count; //已保存的记录的数量
}STU;
StudentLink *IdIndex; //学号索引数组的起始地址
StudentLink *NameIndex; //姓名索引数组的起始地址
```

1.3 算法的实现步骤

1.3.1 链表的相关算法

```
//获得链表的第 i 个元素
Status GetElem(LinkList L,int i,ElemType &e)
 { /* L为带头结点的单链表的头指针。当第i个元素存在时,其值赋给e并返回OK,否则返回
ERROR */
   int j=1; /* j为计数器*/
   LinkList p=L->next; /* p指向第一个结点*/
   while(p&&j<i) /* 顺指针向后查找,直到p指向第i个元素或p为空*/
```

案例 1 基于链表的学生成绩管理系统

1.1 简介

在有数据库的情况下，将数据存到数据库中是一个很好的选择。但是，在没有数据库的情况下，我们也可以用链表存储数据，并完成增加、删除、修改、查询等数据管理功能。本案例就是开发一个基于链表的学生信息管理系统。在这个系统中，学生信息保存在链表中，可以增加、删除、修改、查询，也可以把链表中的数据保存到一个硬盘文件中，便于下次运行程序时读取。此外，还要求对学生的学号和姓名建立索引，然后根据索引快速查找需要的信息，根据索引以数据递增的次序显示各个记录。这不但复习巩固了折半查找算法，而且对索引有了更直观的认识。

1.2 数据结构的定义

```
typedef struct studentStru
{
    int id;//学号
    char name[30];//姓名
    float score[3];//成绩
    struct studentStru *next;//指向下个结点的指针
}StudentNode, *StudentLink;
typedef struct STUStru
{
    StudentLink Head; //指向表头的指针
    StudentLink Tail; //指向表位的指针
    int count; //已保存的记录的数量
}STU;
StudentLink *IdIndex; //学号索引数组的起始地址
StudentLink *NameIndex; //姓名索引数组的起始地址
```

1.3 算法的实现步骤

1.3.1 链表的相关算法

```
//获得链表的第 i 个元素
Status GetElem(LinkList L,int i,ElemType &e)
 { /* L为带头结点的单链表的头指针。当第i个元素存在时,其值赋给e并返回OK,否则返回
ERROR */
   int j=1; /* j为计数器*/
   LinkList p=L->next; /* p指向第一个结点*/
   while(p&&j<i) /* 顺指针向后查找,直到p指向第i个元素或p为空*/
```

案例 1 基于链表的学生成绩管理系统

1.1 简介

在有数据库的情况下，将数据存到数据库中是一个很好的选择。但是，在没有数据库的情况下，我们也可以用链表存储数据，并完成增加、删除、修改、查询等数据管理功能。本案例就是开发一个基于链表的学生信息管理系统。在这个系统中，学生信息保存在链表中，可以增加、删除、修改、查询，也可以把链表中的数据保存到一个硬盘文件中，便于下次运行程序时读取。此外，还要求对学生的学号和姓名建立索引，然后根据索引快速查找需要的信息，根据索引以数据递增的次序显示各个记录。这不但复习巩固了折半查找算法，而且对索引有了更直观的认识。

1.2 数据结构的定义

```
typedef struct studentStru
{
    int id;//学号
    char name[30];//姓名
    float score[3];//成绩
    struct studentStru *next;//指向下个结点的指针
}StudentNode, *StudentLink;
typedef struct STUStru
{
    StudentLink Head; //指向表头的指针
    StudentLink Tail; //指向表位的指针
    int count; //已保存的记录的数量
}STU;
StudentLink *IdIndex; //学号索引数组的起始地址
StudentLink *NameIndex; //姓名索引数组的起始地址
```

1.3 算法的实现步骤

1.3.1 链表的相关算法

```
//获得链表的第 i 个元素
Status GetElem(LinkList L,int i,ElemType &e)
 { /* L为带头结点的单链表的头指针。当第i个元素存在时,其值赋给e并返回OK,否则返回
ERROR */
   int j=1; /* j为计数器*/
   LinkList p=L->next; /* p指向第一个结点*/
   while(p&&j<i) /* 顺指针向后查找,直到p指向第i个元素或p为空*/
```

案例 1 基于链表的学生成绩管理系统

1.1 简介

在有数据库的情况下，将数据存到数据库中是一个很好的选择。但是，在没有数据库的情况下，我们也可以用链表存储数据，并完成增加、删除、修改、查询等数据管理功能。本案例就是开发一个基于链表的学生信息管理系统。在这个系统中，学生信息保存在链表中，可以增加、删除、修改、查询，也可以把链表中的数据保存到一个硬盘文件中，便于下次运行程序时读取。此外，还要求对学生的学号和姓名建立索引，然后根据索引快速查找需要的信息，根据索引以数据递增的次序显示各个记录。这不但复习巩固了折半查找算法，而且对索引有了更直观的认识。

1.2 数据结构的定义

```
typedef struct studentStru
{
    int id;//学号
    char name[30];//姓名
    float score[3];//成绩
    struct studentStru *next;//指向下个结点的指针
}StudentNode, *StudentLink;
typedef struct STUStru
{
    StudentLink Head; //指向表头的指针
    StudentLink Tail; //指向表位的指针
    int count; //已保存的记录的数量
}STU;
StudentLink *IdIndex; //学号索引数组的起始地址
StudentLink *NameIndex; //姓名索引数组的起始地址
```

1.3 算法的实现步骤

1.3.1 链表的相关算法

```
//获得链表的第 i 个元素
Status GetElem(LinkList L,int i,ElemType &e)
 { /* L为带头结点的单链表的头指针。当第i个元素存在时,其值赋给e并返回OK,否则返回
ERROR */
   int j=1; /* j为计数器*/
   LinkList p=L->next; /* p指向第一个结点*/
   while(p&&j<i) /* 顺指针向后查找,直到p指向第i个元素或p为空*/
```

案例 1 基于链表的学生成绩管理系统

1.1 简介

在有数据库的情况下，将数据存到数据库中是一个很好的选择。但是，在没有数据库的情况下，我们也可以用链表存储数据，并完成增加、删除、修改、查询等数据管理功能。本案例就是开发一个基于链表的学生信息管理系统。在这个系统中，学生信息保存在链表中，可以增加、删除、修改、查询，也可以把链表中的数据保存到一个硬盘文件中，便于下次运行程序时读取。此外，还要求对学生的学号和姓名建立索引，然后根据索引快速查找需要的信息，根据索引以数据递增的次序显示各个记录。这不但复习巩固了折半查找算法，而且对索引有了更直观的认识。

1.2 数据结构的定义

```
typedef struct studentStru
{
    int id;//学号
    char name[30];//姓名
    float score[3];//成绩
    struct studentStru *next;//指向下个结点的指针
}StudentNode, *StudentLink;
typedef struct STUStru
{
    StudentLink Head; //指向表头的指针
    StudentLink Tail; //指向表位的指针
    int count; //已保存的记录的数量
}STU;
StudentLink *IdIndex; //学号索引数组的起始地址
StudentLink *NameIndex; //姓名索引数组的起始地址
```

1.3 算法的实现步骤

1.3.1 链表的相关算法

```
//获得链表的第 i 个元素
Status GetElem(LinkList L,int i,ElemType &e)
 { /* L为带头结点的单链表的头指针。当第i个元素存在时,其值赋给e并返回OK,否则返回
ERROR */
   int j=1; /* j为计数器*/
   LinkList p=L->next; /* p指向第一个结点*/
   while(p&&j<i) /* 顺指针向后查找,直到p指向第i个元素或p为空*/
```

案例 1 基于链表的学生成绩管理系统

1.1 简介

在有数据库的情况下，将数据存到数据库中是一个很好的选择。但是，在没有数据库的情况下，我们也可以用链表存储数据，并完成增加、删除、修改、查询等数据管理功能。本案例就是开发一个基于链表的学生信息管理系统。在这个系统中，学生信息保存在链表中，可以增加、删除、修改、查询，也可以把链表中的数据保存到一个硬盘文件中，便于下次运行程序时读取。此外，还要求对学生的学号和姓名建立索引，然后根据索引快速查找需要的信息，根据索引以数据递增的次序显示各个记录。这不但复习巩固了折半查找算法，而且对索引有了更直观的认识。

1.2 数据结构的定义

```
typedef struct studentStru
{
    int id;//学号
    char name[30];//姓名
    float score[3];//成绩
    struct studentStru *next;//指向下个结点的指针
}StudentNode, *StudentLink;
typedef struct STUStru
{
    StudentLink Head; //指向表头的指针
    StudentLink Tail; //指向表位的指针
    int count; //已保存的记录的数量
}STU;
StudentLink *IdIndex; //学号索引数组的起始地址
StudentLink *NameIndex; //姓名索引数组的起始地址
```

1.3 算法的实现步骤

1.3.1 链表的相关算法

```
//获得链表的第 i 个元素
Status GetElem(LinkList L,int i,ElemType &e)
 { /* L为带头结点的单链表的头指针。当第i个元素存在时,其值赋给e并返回OK,否则返回
ERROR */
   int j=1; /* j为计数器*/
   LinkList p=L->next; /* p指向第一个结点*/
   while(p&&j<i) /* 顺指针向后查找,直到p指向第i个元素或p为空*/
```

案例 1 基于链表的学生成绩管理系统

1.1 简介

在有数据库的情况下，将数据存到数据库中是一个很好的选择。但是，在没有数据库的情况下，我们也可以用链表存储数据，并完成增加、删除、修改、查询等数据管理功能。本案例就是开发一个基于链表的学生信息管理系统。在这个系统中，学生信息保存在链表中，可以增加、删除、修改、查询，也可以把链表中的数据保存到一个硬盘文件中，便于下次运行程序时读取。此外，还要求对学生的学号和姓名建立索引，然后根据索引快速查找需要的信息，根据索引以数据递增的次序显示各个记录。这不但复习巩固了折半查找算法，而且对索引有了更直观的认识。

1.2 数据结构的定义

```
typedef struct studentStru
{
    int id;//学号
    char name[30];//姓名
    float score[3];//成绩
    struct studentStru *next;//指向下个结点的指针
}StudentNode, *StudentLink;
typedef struct STUStru
{
    StudentLink Head; //指向表头的指针
    StudentLink Tail; //指向表位的指针
    int count; //已保存的记录的数量
}STU;
StudentLink *IdIndex; //学号索引数组的起始地址
StudentLink *NameIndex; //姓名索引数组的起始地址
```

1.3 算法的实现步骤

1.3.1 链表的相关算法

```
//获得链表的第 i 个元素
Status GetElem(LinkList L,int i,ElemType &e)
 { /* L为带头结点的单链表的头指针。当第i个元素存在时,其值赋给e并返回OK,否则返回
ERROR */
   int j=1; /* j为计数器*/
   LinkList p=L->next; /* p指向第一个结点*/
   while(p&&j<i) /* 顺指针向后查找,直到p指向第i个元素或p为空*/
```

案例 1 基于链表的学生成绩管理系统

1.1 简介

在有数据库的情况下，将数据存到数据库中是一个很好的选择。但是，在没有数据库的情况下，我们也可以用链表存储数据，并完成增加、删除、修改、查询等数据管理功能。本案例就是开发一个基于链表的学生信息管理系统。在这个系统中，学生信息保存在链表中，可以增加、删除、修改、查询，也可以把链表中的数据保存到一个硬盘文件中，便于下次运行程序时读取。此外，还要求对学生的学号和姓名建立索引，然后根据索引快速查找需要的信息，根据索引以数据递增的次序显示各个记录。这不但复习巩固了折半查找算法，而且对索引有了更直观的认识。

1.2 数据结构的定义

```
typedef struct studentStru
{
    int id;//学号
    char name[30];//姓名
    float score[3];//成绩
    struct studentStru *next;//指向下个结点的指针
}StudentNode, *StudentLink;
typedef struct STUStru
{
    StudentLink Head; //指向表头的指针
    StudentLink Tail; //指向表位的指针
    int count; //已保存的记录的数量
}STU;
StudentLink *IdIndex; //学号索引数组的起始地址
StudentLink *NameIndex; //姓名索引数组的起始地址
```

1.3 算法的实现步骤

1.3.1 链表的相关算法

```
//获得链表的第 i 个元素
Status GetElem(LinkList L,int i,ElemType &e)
 { /* L为带头结点的单链表的头指针。当第i个元素存在时,其值赋给e并返回OK,否则返回
ERROR */
   int j=1; /* j为计数器*/
   LinkList p=L->next; /* p指向第一个结点*/
   while(p&&j<i) /* 顺指针向后查找,直到p指向第i个元素或p为空*/
```

案例 1 基于链表的学生成绩管理系统

1.1 简介

在有数据库的情况下，将数据存到数据库中是一个很好的选择。但是，在没有数据库的情况下，我们也可以用链表存储数据，并完成增加、删除、修改、查询等数据管理功能。本案例就是开发一个基于链表的学生信息管理系统。在这个系统中，学生信息保存在链表中，可以增加、删除、修改、查询，也可以把链表中的数据保存到一个硬盘文件中，便于下次运行程序时读取。此外，还要求对学生的学号和姓名建立索引，然后根据索引快速查找需要的信息，根据索引以数据递增的次序显示各个记录。这不但复习巩固了折半查找算法，而且对索引有了更直观的认识。

1.2 数据结构的定义

```
typedef struct studentStru
{
    int id;//学号
    char name[30];//姓名
    float score[3];//成绩
    struct studentStru *next;//指向下个结点的指针
}StudentNode, *StudentLink;
typedef struct STUStru
{
    StudentLink Head; //指向表头的指针
    StudentLink Tail; //指向表位的指针
    int count; //已保存的记录的数量
}STU;
StudentLink *IdIndex; //学号索引数组的起始地址
StudentLink *NameIndex; //姓名索引数组的起始地址
```

1.3 算法的实现步骤

1.3.1 链表的相关算法

```
//获得链表的第 i 个元素
Status GetElem(LinkList L,int i,ElemType &e)
 { /* L为带头结点的单链表的头指针。当第i个元素存在时,其值赋给e并返回OK,否则返回
ERROR */
   int j=1; /* j为计数器*/
   LinkList p=L->next; /* p指向第一个结点*/
   while(p&&j<i) /* 顺指针向后查找,直到p指向第i个元素或p为空*/
```

案例 1 基于链表的学生成绩管理系统

1.1 简介

在有数据库的情况下，将数据存到数据库中是一个很好的选择。但是，在没有数据库的情况下，我们也可以用链表存储数据，并完成增加、删除、修改、查询等数据管理功能。本案例就是开发一个基于链表的学生信息管理系统。在这个系统中，学生信息保存在链表中，可以增加、删除、修改、查询，也可以把链表中的数据保存到一个硬盘文件中，便于下次运行程序时读取。此外，还要求对学生的学号和姓名建立索引，然后根据索引快速查找需要的信息，根据索引以数据递增的次序显示各个记录。这不但复习巩固了折半查找算法，而且对索引有了更直观的认识。

1.2 数据结构的定义

```
typedef struct studentStru
{
    int id;//学号
    char name[30];//姓名
    float score[3];//成绩
    struct studentStru *next;//指向下个结点的指针
}StudentNode, *StudentLink;
typedef struct STUStru
{
    StudentLink Head; //指向表头的指针
    StudentLink Tail; //指向表位的指针
    int count; //已保存的记录的数量
}STU;
StudentLink *IdIndex; //学号索引数组的起始地址
StudentLink *NameIndex; //姓名索引数组的起始地址
```

1.3 算法的实现步骤

1.3.1 链表的相关算法

```
//获得链表的第 i 个元素
Status GetElem(LinkList L,int i,ElemType &e)
 { /* L为带头结点的单链表的头指针。当第i个元素存在时,其值赋给e并返回OK,否则返回
ERROR */
   int j=1; /* j为计数器*/
   LinkList p=L->next; /* p指向第一个结点*/
   while(p&&j<i) /* 顺指针向后查找,直到p指向第i个元素或p为空*/
```