**实验1**

**实验题目：单链表的插入和删除**

**实验目的：**

了解和掌握线性表的逻辑结构和链式存储结构，掌握单链表的基本算法及相关的时间复杂度分析方法。

**实验要求：**

阅读所提供的代码，根据提示补全缺少的部分，完成下列功能：

1）用头插法建立一个带头结点的单链表。要求在建立单链表的过程中，不允许插入重复的字符串；

2）根据输入字符串，查找该字符串在单链表中的位置，若不存在则返回NULL；

3）根据输入字符串，删除单链表中具有相同字符串的结点；

4）根据输入字符串，若不存在于当前单链表中，则插入单链表表头。

**实验主要步骤：**

1. 分析、理解给出的程序，根据提示填写缺少部分的代码；
2. 调试程序，并且编译通过；
3. 使用以下输入数据：

bat↙ cat↙ hat↙ jat↙ lat↙ jat↙ mat↙ #↙

建立单链表，其中“#”表示停止输入的符号。然后删除包含“bat”元素的结点。然后将包含“pat”元素的结点插入单链表。

**提示：**字符串的比较和赋值分别可以使用 strcmp 和 strcpy函数

课后实验报告提交要求：

1. 提交时将所有函数写在一个源文件(.c文件或.cpp文件)中。

程序代码（请同学们填写空缺部分的代码）:

#define \_CRT\_SECURE\_NO\_WARNINGS 1

using namespace std;

#include"stdio.h"

#include"string.h"

#include"stdlib.h"

#include"ctype.h"

#define OK 1

typedef struct node //定义结点

{

char data[10]; //结点的数据域为字符串

struct node\* next; //结点的指针域

}ListNode;

typedef ListNode\* LinkList; // 自定义LinkList单链表类型

LinkList CreatList(void); //函数，用头插入法建立带头结点的单链表

LinkList LocateNode(LinkList L, char\* ch); //函数，按值查找结点

void DeleteList(LinkList L, char\* ch); //函数，删除指定值的结点

void Printlist(LinkList L); //函数，打印链表中的所有值

void DeleteAll(LinkList L); //函数，删除所有结点，释放内存

LinkList AddNode(LinkList L); //修改程序：增加节点。用头插法，返回头指针

//==========主函数==============

int main()

{

char ch[10], num[5];

LinkList head;

head = CreatList(); //用头插入法建立单链表，返回头指针

Printlist(head); //遍历链表输出其值

printf(" Delete node (y/n):"); //输入"y"或"n"去选择是否删除结点

scanf("%s", num);

if (strcmp(num, "y") == 0 || strcmp(num, "Y") == 0) {

printf("Please input Delete\_data:");

scanf("%s", ch); //输入要删除的字符串

DeleteList(head, ch);

Printlist(head);

}

printf(" Add node ? (y/n):"); //输入"y"或"n"去选择是否增加结点

scanf("%s", num);

if (strcmp(num, "y") == 0 || strcmp(num, "Y") == 0)

{

head = AddNode(head);

}

Printlist(head);

DeleteAll(head); //删除所有结点，释放内存

return OK;

}

//==========用头插入法建立带头结点的单链表===========

LinkList CreatList(void)

{

char ch[100];

LinkList head, p;

head = new ListNode;

head->next = NULL;

while (1)

{

printf("Input # to end ");

printf("Please input Node\_data:");

scanf("%s", ch);

p = head->next;

if (strcmp(ch, "#") == 0) {

break;

}

if (LocateNode(head, ch) == NULL) {

p = (ListNode\*)malloc(sizeof(ListNode));

strcpy(p->data, ch);

p->next = head->next;

head->next = p;

}

//根据输入字符串，插入单链表中。以“#”为结束标志（头插法）

}

return head;

}

//==========按值查找结点，找到则返回该结点的位置，否则返回NULL==========

LinkList LocateNode(LinkList head, char\* key)

{

LinkList p = head->next; //从首元结点比较

//扫描单链表，按值查找结点。若p=NULL则查找失败，否则p指向找到的值为key的结点

while (p != NULL && strcmp(p->data, key) != 0) {

p = p->next;

}

return p; //若p=NULL则查找失败，否则p指向找到的值为key的结点

}

//==========按照输入值增加节点=======

LinkList AddNode(LinkList head)

{

char ch[10];

LinkList s, pp;

printf("\nPlease input a New Node\_data:");

scanf("%s", ch); //输入各结点的字符串

//若输入字符串与当前链表结点元素均不同，则插入到表头位置

if (LocateNode(head, ch) == NULL) {

s = new ListNode;

strcpy(s->data, ch);

s->next = head->next;

head->next = s;

}

return head;

}

//==========删除带头结点的单链表中的指定结点=======

void DeleteList(LinkList head, char\* key)

{

LinkList p, r, q = head;

p = LocateNode(head, key); //按key值查找结点的

if (p == NULL) { //若没有找到结点，退出

printf("position error");

exit(0);

}

//若找到结点，则将其删除

while (q->next != p) {

q = q->next;

}

r = q->next;

q->next = r->next;

delete r;

}

//===========打印链表=======

void Printlist(LinkList head)

{

LinkList p = head->next; //从开始结点打印

while (p) {

printf("%s, ", p->data);

p = p->next;

}

printf("\n");

}

//==========删除所有结点，释放空间===========

void DeleteAll(LinkList head)

{

LinkList p = head, r;

while (p->next) {

r = p->next;

delete p;

p = r;

}

delete p;

}

**实验结果：**

根据实验步骤中的提示，将屏幕输出内容截取，并粘贴到word文档中。

