**软件技术基础**

**实 验 报 告**

**学生姓名：孔迪 学 号：2016010908009 指导教师：居太亮**

## 实验项目名称：上机实验二

## 实验内容：

**完整程序训练**

ex2\_2 (1),(2):

#### 源代码

#include "stdafx.h"

#define TRUE 1

#define FALSE 0

typedef struct \_NODE

{

    int data;

    struct \_NODE \*next;

}node;

*//创建链表，输入0停止*

int list\_creat(node \*\*p\_head)

{

    int num;

    \*p\_head = (node\*)malloc(sizeof node);

    if (\*p\_head == NULL)    return FALSE;

    (\*p\_head)->data = 0;

    (\*p\_head)->next = NULL;

    node \*temp = \*p\_head;

    printf("Please input datas(input 0 to end): ");

    while (TRUE)

    {

        scanf("%d", &num);

        if (num == 0)   break;

        temp->next = (node\*)malloc(sizeof node);

        if (temp->next == NULL) return FALSE;

        temp->next->data = num;

        temp->next->next = NULL;

        temp = temp->next;

    }

    return TRUE;

}

*//按顺序打印链表中的值，p\_head为链表的头指针*

void list\_show(node \*p\_head)

{

    node \*p\_temp = p\_head;

    if (p\_temp->next == NULL)

        printf("\n\n\*\*\*\*\*\* The list's length is 0! \*\*\*\*\*\*\n\n");

    else

    {

        printf("The list is: ");

        while (p\_temp->next != NULL)

        {

            printf("%d ", p\_temp->next->data);

            p\_temp = p\_temp->next;

        }

    }

}

*//插入新的元素，p\_head为链表头指针的地址，addr表示要插入的位置，value为新值*

*//成功返回TRUE，失败返回FALSE*

int list\_insert(node \*p\_head, int addr, int value)

{

    int i = 1;

    node \*temp = p\_head, \*new\_ele = (node\*)malloc(sizeof node);

    if (new\_ele == NULL)    return FALSE;

    new\_ele->data = value;

    for (; i < addr && temp->next != NULL; i++) *//定位*

        temp = temp->next;

    if (i < addr)                                               *//插入位置超出链表长度*

    {                                                           *//*

        printf("\n------ Invalid Insert Address ------");       *//*

        return FALSE;                                           *//*

    }

    new\_ele->next = temp->next;

    temp->next = new\_ele;

    return TRUE;

}

*//删除链表中指定位置的元素，p\_head为链表头指针的地址，addr表示元素位置*

*//成功返回TRUE，失败返回FALSE*

int list\_del(node \*p\_head, int addr)

{

    node \*temp = p\_head, \*del = NULL;

    int i = 1;

    for (; i < addr && temp->next != NULL; i++) *//定位*

        temp = temp->next;

    if (i < addr)                                       *//指定位置超出链表长度*

    {                                                   *//*

        printf("\n------ Invalid Address ------");      *//*

        return FALSE;                                   *//*

    }

    del = temp->next;           *//将del指向待删除元素*

    temp->next = del->next;     *//重新连接链表*

    free(del);                  *//释放空间*

    return TRUE;

}

void list\_adj(node \*p\_head, int value)

{

    int addr = 0, exist = 0;

    node \*p\_temp = p\_head;

    while (p\_temp->next != NULL)

    {

        p\_temp = p\_temp->next;

        addr++;

        if (p\_temp->data == value)

        {

            list\_del(p\_head, addr);

            exist = 1;

            printf("\nDelete one element.");

            break;

        }

    }

    if (!exist)

    {

        list\_insert(p\_head, addr + 1, value);

        printf("\nInsert one element.");

    }

    printf("\nThe list after operation: ");

    list\_show(p\_head);

}

int main()

{

    int value = 0;

*//实验ex2\_2 (1), 新建链表*

    printf("ex2\_2 (1):\n");

    node \*p\_head = NULL;

    if (list\_creat(&p\_head) == FALSE)

    {

        printf("\n\*\*\*\*\*\* Failed to creat! \*\*\*\*\*\*\n\n");

        system("pause");

        return 0;

    }

    list\_show(p\_head);

*//实验ex2\_2 (2), 相关操作*

    printf("\n\n\nex2\_2 (2):\nplease input one number:");

    scanf("%d", &value);

list\_adj(p\_head, value);

    printf("\n\n\n");

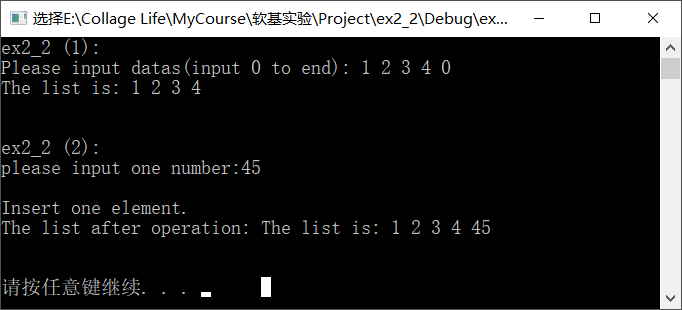
    system("pause");

    return 0;

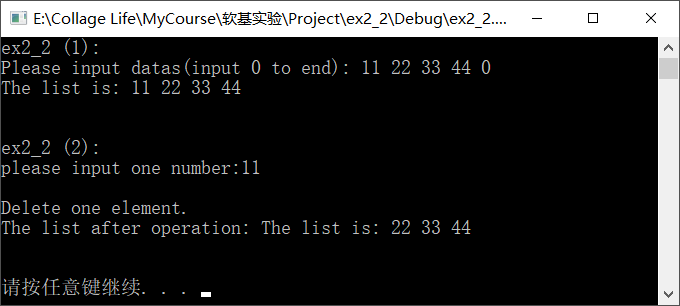
}

#### 测试结果

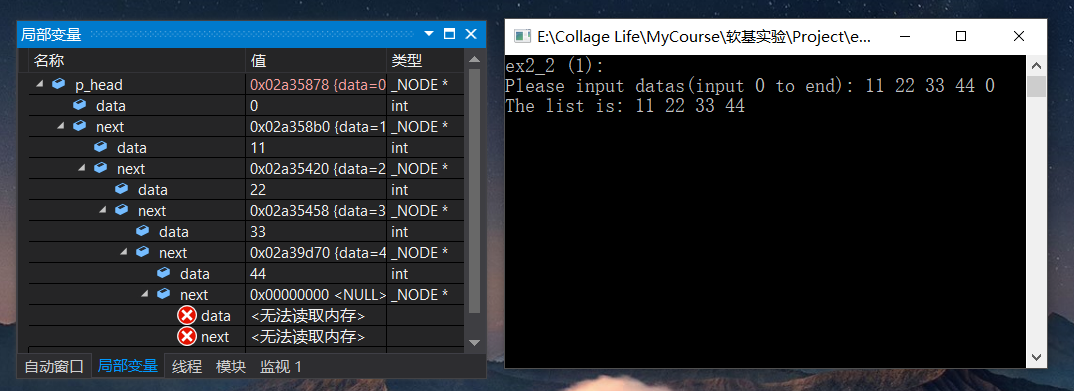
Test1：新建链表并插入新数：

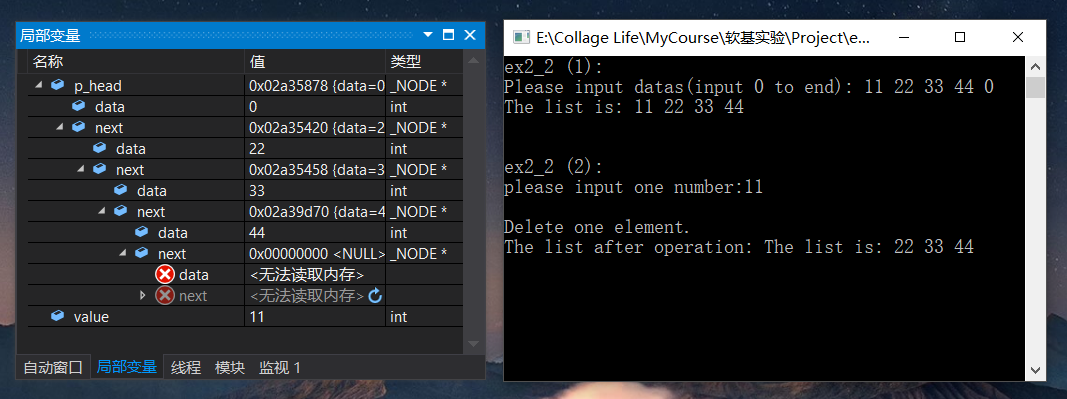


Test2：新建链表并删除其中一个数：



Test3：一个验证：





#### 上机时遇到的问题

* 1. 链表首节点存放了有效数据，末节点未存放有效数据，导致后续的插入和删除写起来非常繁琐。

**解决办法：**更换思路，链表首节点不存放有效数据（用0填充p\_head->data,0为无效数据）,末节点存放了有效数据。

ex2\_2(3):

#### 源代码

#include "stdafx.h"

#include <stdlib.h>

#define TRUE 1

#define FALSE 0

typedef struct \_NODE

{

    int data;

    struct \_NODE \*next;

}node;

*//创建并初始化链表*

int list\_creat(node \*\*p\_head)

{

    int i=0, element[6] = { 6,17,21,29,33,47 };

    \*p\_head = (node\*)malloc(sizeof node);

    if (\*p\_head == NULL)    return FALSE;

    (\*p\_head)->data = 0;

    (\*p\_head)->next = NULL;

    node \*temp = \*p\_head;

    for(;i<6;i++)

    {

        temp->next = (node\*)malloc(sizeof node);

        if (temp->next == NULL) return FALSE;

        temp->next->data = element[i];

        temp->next->next = NULL;

        temp = temp->next;

    }

    return TRUE;

}

*//按顺序打印链表中的值，p\_head为链表的头指针*

void list\_show(node \*p\_head)

{

    node \*p\_temp = p\_head;

    if (p\_temp->next == NULL)

        printf("\n\n\*\*\*\*\*\* The list's length is 0! \*\*\*\*\*\*\n\n");

    else

        while (p\_temp->next != NULL)

        {

            printf("%d ", p\_temp->next->data);

            p\_temp = p\_temp->next;

        }

}

*//插入新元素，不破坏原链表的递增性*

int list\_insert(node \*p\_head, int value)

{

    int i = 1;

    node \*temp = p\_head, \*new\_ele = (node\*)malloc(sizeof node);

    if (new\_ele == NULL)    return FALSE;

    new\_ele->data = value;

    while(temp->next != NULL && value>temp->next->data) *//定位*

        temp = temp->next;

    new\_ele->next = temp->next;

    temp->next = new\_ele;

    return TRUE;

}

int main()

{

    int value = 0;

    node \*p\_head = NULL;

    if (list\_creat(&p\_head) == FALSE)

    {

        printf("\n\*\*\*\*\*\* Failed to creat! \*\*\*\*\*\*\n\n");

        system("pause");

        return 0;

    }

    printf("\nThe list is initialed as: ");

    list\_show(p\_head);

    printf("\nplease input one number:");

    scanf("%d", &value);

    list\_insert(p\_head, value);

    printf("\nThe list after insert is: ");

    list\_show(p\_head);

    printf("\n\n\n");

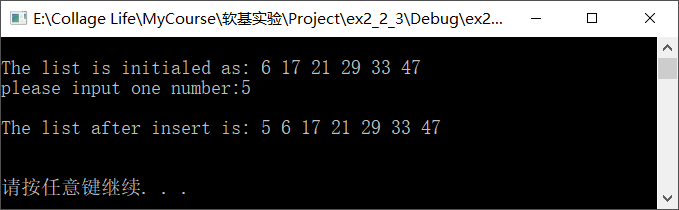
    system("pause");

    return 0;

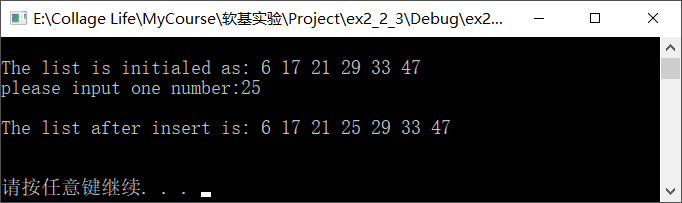
}

#### 测试结果

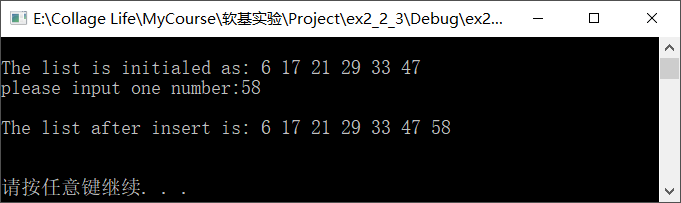
Test1，插入最小数：



Test2，插入中间数：



Test3，插入最大数：



#### 上机时遇到的问题

无。

ex2\_3 (1):

#### 源代码

#include "stdafx.h"

#include <stdlib.h>

#define TRUE 1

#define FALSE 0

typedef struct \_NODE

{

    int data;

    struct \_NODE \*next;

}node;

*//创建链表，输入0停止*

int list\_creat(node \*\*p\_head)

{

    int num;

    \*p\_head = (node\*)malloc(sizeof node);

    if (\*p\_head == NULL)    return FALSE;

    (\*p\_head)->data = 0;

    (\*p\_head)->next = NULL;

    node \*temp = \*p\_head;

    printf("Please input datas(input 0 to end): ");

    while (TRUE)

    {

        scanf("%d", &num);

        if (num == 0)   break;

        temp->next = (node\*)malloc(sizeof node);

        if (temp->next == NULL) return FALSE;

        temp->next->data = num;

        temp->next->next = NULL;

        temp = temp->next;

    }

    return TRUE;

}

*//按顺序打印链表中的值，p\_head为链表的头指针*

void list\_show(node \*p\_head)

{

    node \*p\_temp = p\_head;

    if (p\_temp->next == NULL)

        printf("\n\n\*\*\*\*\*\* The list's length is 0! \*\*\*\*\*\*\n\n");

    else

        while (p\_temp->next != NULL)

        {

            printf("%d ", p\_temp->next->data);

            p\_temp = p\_temp->next;

        }

}

*//删除链表中的负数*

void list\_del(node \*p\_head)

{

    node \*temp = p\_head, \*del = NULL;

    while (temp != NULL && temp->next != NULL)

    {

        if (temp->next->data < 0)

        {

            del = temp->next;

            temp->next = del->next;

            free(del);

            continue;

        }

        temp = temp->next;

    }

}

int main()

{

    int value = 0;

*//新建链表*

    printf("ex2\_2 (1):\n");

    node \*p\_head = NULL;

    if (list\_creat(&p\_head) == FALSE)

    {

        printf("\n\*\*\*\*\*\* Failed to creat! \*\*\*\*\*\*\n\n");

        system("pause");

        return 0;

    }

    printf("\nThe list is: ");

    list\_show(p\_head);

*//删除链表中负数*

    list\_del(p\_head);

    printf("\nDelete all negative numbers:");

    list\_show(p\_head);

    printf("\n\n\n");

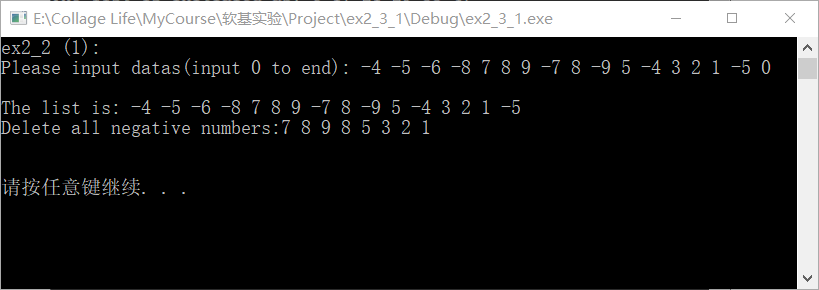
    system("pause");

    return 0;

}

#### 测试结果

Test：



#### 上机时遇到的问题

* 1. 测试时发现若输入连续的负数时会出现间隔删除的情况，例如输入-1 -2 -3 -4 -5 6 7 8 -4 1 0;则输出为-2 -4 6 7 8 1

**解决办法：**调试模式，查看变量，发现为逻辑不严谨，在list\_del函数中的if()中添加continue，解决问题。

### ex2\_3 (2):

#### 源代码

#include "stdafx.h"

#include <stdlib.h>

#define FALSE 0

#define TRUE 1

typedef struct \_NODE

{

    int data;

    struct \_NODE \*prior, \*next;

}dl\_node;

*//双向链表创建程序，其中(\*head)->data用来存储链表的长度，便于进行后面的排序操作*

int dl\_liset\_creat(dl\_node\*\*head)

{

    int num;

    if ((\*head = (dl\_node\*)malloc(sizeof dl\_node)) == NULL)

        return FALSE;

    (\*head)->data = 0;

    (\*head)->prior = NULL;

    (\*head)->next = NULL;

    dl\_node\*temp = \*head;

    printf("Please input:");

    while (TRUE)

    {

        scanf("%d", &num);

        if (num == 0)   break;

        temp->next = (dl\_node\*)malloc(sizeof dl\_node);

        temp->next->data = num;

        temp->next->prior = temp;

        temp->next->next = NULL;

        temp = temp->next;

        (\*head)->data++;

    }

    return TRUE;

}

void dl\_list\_show(dl\_node\*head)

{

    dl\_node\*temp = head;

    while (temp->next != NULL)

    {

        printf("%d ", temp->next->data);

        temp = temp->next;

    }

*//printf(" The list's length is %d.", head->data);*

}

void dl\_list\_sort(dl\_node\*head)

{

    int length = head->data, end = 1;

    dl\_node\*p1=NULL, \*p2=NULL,\*sign=NULL;

    for(int count=1;count<length;count++)

    {

        end = 1;

        for (p1 = head->next, p2 = head->next->next; p2 != sign; p1 = p1->next, p2 = p2->next)

        {

            if (p1->data > p2->data)

            {

                p1->prior->next = p2;

                if (p2->next != NULL)

                    p2->next->prior = p1;

                p1->next = p2->next;

                p2->prior = p1->prior;

                p1->prior = p2;

                p2->next = p1;

                dl\_node\*temp = p1;

                p1 = p2;

                p2 = temp;

                end = 0;

            }

        }

        if (end)break;

        printf("\nCycle %d: ", count);

        dl\_list\_show(head);

        sign = p1;

    }

    printf("\n\nSort complish!!");

}

int main()

{

    dl\_node\*head = NULL;

    dl\_liset\_creat(&head);

    printf("\nThe list you have inputed is: ");

    dl\_list\_show(head);

    printf("\n\n\* \* \* \* \* \* SORTing... ... \* \* \* \* \* \*\n\n");

    dl\_list\_sort(head);

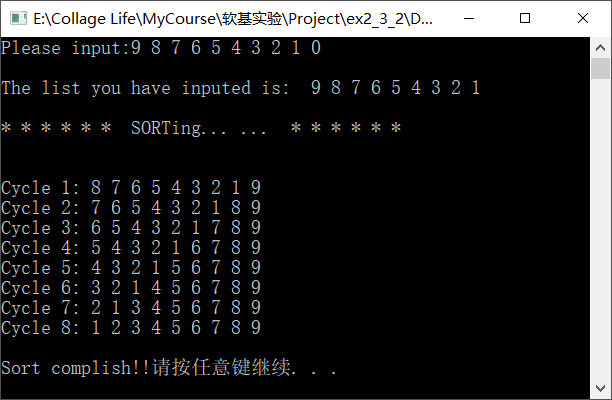
    system("pause");

return 0;

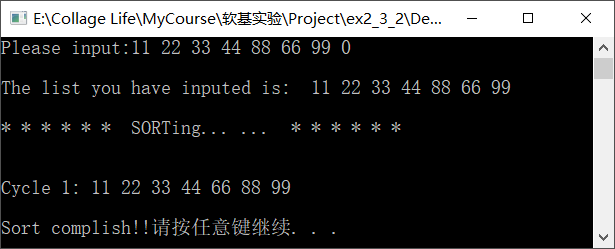
}

#### 测试结果

Test1：



Test2：



#### 上机时遇到的问题

* 1. 一开始努力尝试理清排序时各个变量交换顺序，以及控制循环的条件，多次尝试遭遇失败

**解决办法：**网上搜索，借鉴别人写的代码，得以启发，同时加上新的尝试和调试，问题得以解决。

## 总结及心得体会：

1. 编译器提供的调试功能是修改bug的好工具。
2. 适当借鉴别人的优秀代码，对提高自己的能力很有帮助。