数 据 结 构 实 验 报 告 成绩_____

学号		姓名		授课教师	黄 欣
专业		实验报告递交日期			

实验题目

L 是带头结点的单链表，结点按整数增为序排列，插入值为 x 的结点，使 L 仍有序.

一. 需求分析

1．程序实现的功能：输入有序单链表与插入结点值，得有序单链表。

2. 编制函数：

1). 建立链表 L 的函数； linklist *creat(void)

2). 显示链表 L 的函数； void list(linklist *L)

3). 插入值为 x 的结点到 L 的函数； void insert(linklist *L,int x)

4). 释放单链表结点空间函数； void delete(linklist *L)

5). 主函数完成功能：

a). 调用 L=creat( ) ; b). 调用 list(L); c). 输入 x 值；

d). 调用 insert(L,x); e). 调用 list(L); f). 调用 delete(L).

3．数据输入的内容、输入形式与范围

以值从小到大为序，输入所创建的单链表中的数据，以及要插入的 x 的值，其类型是整型数；输入数据以回车符相隔，以'@'为输入结束符。

4．数据输出的内容与形式

输出创建单链表时单链表和插入 x 后的单链表中的结点序号与数据, 数据以回车符相隔。

二. 主要算法的算法思想.

1．创建单链表函数：

用尾插法建立单链表，每个新插入的结点都作为单链表的最后一个结点。读入结点的数值，生成的新结点插入表尾。

2. 显示链表 L 的函数：

从 L 的第一个结点开始往后依次输出结点序号与数据。

3. 插入值为 x 的结点到 L 的函数：

生成值为 x 的结点*p，在 L 中找*p 的前趋*q, 在*q 后插入*p。

4. 释放单链表 L 结点空间函数:

从 L 的开始结点起，定住后一个结点后，释放前一个结点的空间。

5. 主函数:

为单链表头结点开辟结点空间;依次调用创建单链表函数、显示单链表函数;

输入 x 值；调用插入函数、显示单链表函数，和释放单链表函数。

三. 设计：

1．线性表存储结构：单链表。

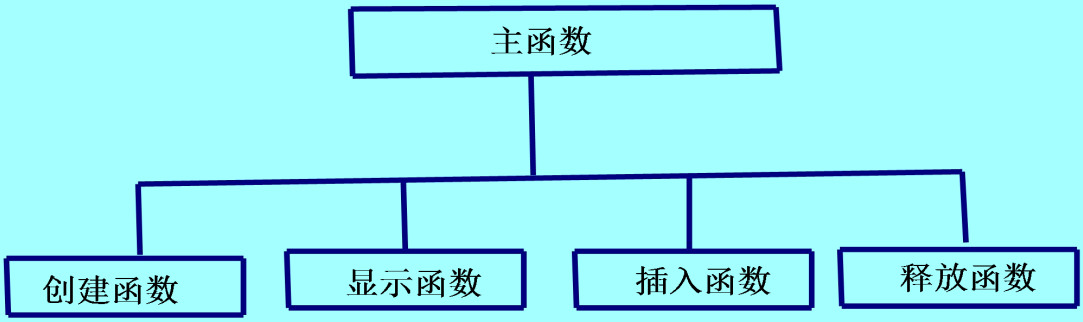
单链表结点类型定义：

```
typedef struct node
 {datatype data;
  struct node *next;
 }linklist;          /*单链表结点类型*/
```

2．参数表（列出所有的符号常量与全局变量）

参数名	数据传递方式	数据内容	传递	所属函数
NULL	符号常量	空指针	0	所有函数

3．函数间的调用关系图



4．列出每个函数的函数声明、函数作用、函数值、形参内容与形式、主要算法步骤等

1)．创建单链表函数

函数首部：void creat(linklist *h)

形参：h:单链表头指针

函数作用：创建存放原始数据的单链表

函数值：无

局部变量 r：作为尾指针，指向生成的新链表的尾结点

new：结点类型指针，指向新结点。*new 尾插到链表中

输入一个结点算法主要步骤：

(a) 开辟新结点空间 new 指向 new=(linklist *)malloc(sizeof(linklist));

(b) 新结点*new 插入表尾 r->next=new;

(c) 尾指针 r 指向新的表尾 r=new;

(d) 将输入数据放入新结点的数据域中 r->ch=atoi(numstr);

(e) 读入下一个结点的值 gets(numstr);

输入链表结点循环条件：numstr[0]!='@'

2)．输出单链表函数

函数首部：void list(linklist *h)

形参：h:单链表头指针

函数作用：将单链表输出在窗口上

函数值：无

局部变量　指针 p：初始 p 指向 h 的开始结点，依次指向 h 的各个结点

算法主要步骤：

判断是否为空链表：

(a)若空，则输出“empty list”

if(!p){printf("\n empty list");return;}

(b)若不为空，则依次扫描每一个结点并且将结点数据输出

printf("\nnode number %d, value %d",i++,p->data);

p=p->next;

3). 插入函数

函数首部：void insert(linklist *h, datatype x)

形参：h: 单链表头指针；　x: 插入结点的值。

函数作用：将值为 x 的结点插在单链表 h 的适当位置，使 h 仍有序。

函数值：无

局部变量：

指针 s：指向新结点；

指针 q：初始 q 指向 h 的头结点，依次找*s 的插入位置，循环结束*q 是*s 的前趋结点；

指针 p：初始 p 指向 h 的开始结点，依次指向*q 的原后继结点，即最后在*q 与*p 间插入*s.

算法主要步骤：

(a) 找*s 的前趋*q　while(p &&p->data<x)

{q=p;p=p->next;}

(b) 在*q 与*p 间插入*s　s->next=p;　q->next=s;

4）. 释放函数：

函数首部：void del(charnode *h)

形参：h：单链表头指针

函数作用：释放删除单链表结点空间

函数值：无

局部变量：

指针 p：指针 p 指向剩余链表的开始结点，初值：p=h;

指针 q：指针 q 指向*p 的后继

算法主要步骤：

(a) q 指向*p 的后继　q=p->next;

(b) 释放 p 指向的结点空间　free(p);

(c) p 指向剩余链表的开始结点　p=q;

四. 调试分析：

1．调试中出现的问题，解决的办法 （略）

1）.

2）.

3）.

2．每个函数的时、空复杂性分析

1）. void creatd(linklist *h)建单链表函数

T(n)=0(n)， S(n)=0(n)；

2）. void list(linklist *h) 输出单链表函数

T(n)=0(n)， S(n)=0(1)；

3）. void insert(linklist *h, datatype x) 插入函数

T(n)=0(n) ， S(n)=0(1)；

4）. void del(linklist *h) 删除单链表

T(n)=0(n) ， S(n)=0(1)；

5）. main() 主函数

T(n)=0(n) ， S(n)=0(n).

3．改进设想，经验体会

(略)

五. 使用说明：如何使用你编制的程序、操作步骤.

编译程序成功后，按界面提示输入单链表结点数据与值 x。

六. 测试结果：

输入输出数据内容：

窗口显示如下：(下划线部分为输入部分，其余为输出部分)

测试数据一：

creat a linklist:
type '@' to end
input the value of node= 2↙
input the value of node= 15↙
input the value of node= 43↙
input the value of node= @↙
list the linklist:
node number 1, value 2
node number 2, value 15
node number 3, value 43
input x=20↙
list new linklist:
node number 1, value 2
node number 2, value 15
node number 3, value 20
node number 4, value 43
delete the list.

测试数据二：

```
creat    a linklist:
type '@' to end
input    the value of node= @↙
list the linklist:
empty list
input x=10↙
list new linklist:
node number 1, value 10
delete the list.
```

七．源代码清单

```
#include "stdio.h"
#include "stdlib.h"
#define NULL 0
typedef int datatype;
typedef struct node
  {datatype data;
    struct node *next;
  }linklist;          /*单链表结点类型*/

main()
{void creat(linklist *h);
  void list(linklist *h);
  void insert(linklist *h, datatype x);
  void del(linklist *h);
  linklist *head;
  datatype x;
  head=(linklist *)malloc(sizeof(linklist)); /*开辟头结点*/
  head->next=NULL;
  printf("creat    a linklist:\n");
  creat(head);
  printf("list the linklist:");
  list(head);
  printf("\ninput x= ");
  scanf("%d",&x);
  insert(head,x);
  printf("list new linklist:");
  list(head);
  printf("\n delete the list.");
  del(head);
}

void creat(linklist *h)
{char numstr[8];
```

```
  linklist *r,*new;
  r=h;
  printf("type '@' to end\n");
  printf("input    the value of node= ");
  gets(numstr);
  while(numstr[0]!='@')
      {new=(linklist *)malloc(sizeof(linklist));
       r->next=new;
       r=new;
       r->data=atoi(numstr);
       printf("input    the value of node= ");
       gets(numstr);
      }
     r->next=NULL;
}

void list(linklist *h)
{int i=1;
  linklist *p;
  p=h->next;
  if(!p) {printf("\nempty list");return;}
  do
  { printf("\nnode number %d, value %d",i++,p->data);
    p=p->next;
  }while(p);
}

void insert(linklist *h, datatype x)
    {linklist *p,*q,*s;
    s=(linklist *)malloc(sizeof(linklist));
    s->data=x;
    q=h;
    p=q->next;
    while(p &&p->data<x)
        {q=p;p=p->next;}
    s->next=p;      /*  在*q 与*p 间插入*s    */
    q->next=s;
}

void del(linklist *h);
  {linklist *p=h,*q;
    while(p)
        {q=p->next;
         free p;
```

```
        p=q;
      }
  }
```