**软件技术基础**

**实验1、线性表应用**

**1、实验内容**

实现课本上的8-11页，线性表例程。

主要算法： 1、查找操作

2、插入操作

3、删除操作

**2 工程构建**

创建一个工程，将\*.cpp文件加入工程。



**3 程序代码**

# include "stdio.h"

#define maxlen 5

int data[maxlen+1];

int search(int x, int v[maxlen+1]);

void insqlist(int x, int i, int v[maxlen+1]);

void delsqlist(int i, int v[maxlen+1]);

void main()

{

 int i,x;

 printf(" 请输入数据:\n");

 for (i=1;i<=maxlen;i++)

 scanf("%d",&data[i]);

 for (i=1;i<=maxlen;i++)

 printf("%d|",data[i]);

 printf("\n");

 data[0]=maxlen;

/\* 在线性表V中查找值为 3 的数据元素\*/

 printf("查找3是否在表内， 返回0，表示没有。\n");

 x=3;

 int find=search(,data);

 printf("%d 的下标为 %d\n",x,find);

/\* 在线性表V中底第3个位置，插入值为 8 的数据元素，溢出表满了。\*/

 printf("在线性表V中底第3个位置，插入值为 8 的数据元素。溢出，表满了\n");

 x=8;

 i=3;

 insqlist(x,i);

 for (i=1;i<=data[0];i++)

 printf(" %d |",data[i]);

 printf("\n");

/\* 在线性表V中删除第3个位置的数据元素\*/

 i=3;

 delsqlist(i,);

 for (i=1;i<=data[0];i++)

 printf("%d ",data[i]);

 printf("\n");

/\* 在线性表V中底第3个位置，插入值为 9 的数据元素\*/

 x=9;

 i=3;

 insqlist(x,i,data);

 for (i=1;i<=data[0];i++)

 printf("%d |",data[i]);

 printf("\n");

}

int search(int x, int v[maxlen+1])

{

 int i=1;

 while (i<=v[0])

 {

 if (v[i]!=x)

 i=i+1;

 else

 return i; /\* 找到元素ai \*/

 }

 return 0;

}

/\*在第i个位置插入一个新结点\*/

void insqlist(int x, int i, ])

{

 int j;

 if (v[0]==maxlen) printf("溢出!");

 else

 {

 if ((i<1) || (i>v[0])) printf("% d 是一个不正确的插入号码\n",i);

 else

 {

 for (j=v[0];j>=i;j--) v[j+1]=v[j]; /\* 元素后移 \*/

 v[i]=x; /\* 插入元素 \*/

 v[0]=v[0]+1; /\* 表长度 加 1 \*/

 }

 }

}

/\*删除线性表中的第i个结点\*/

void delsqlist(int i, int v[maxlen+1])

{

 int j;

 if (v[0]<1)

 printf("下溢!");

 else

 {

 if (i<1 || i>v[0])

 printf("% d 是一个不正确的删除号码\n",i);

 else

 {

 for (j=i; j<v[0];j++)

 v[j]=v[j+1]; /\* 元素前移，覆盖被删除的元素 \*/

 v[0]=v[0]-1; /\* 表长度 减 1 \*/

 }

 }

}

 **软件技术基础实验2、堆栈应用**

**1、实验内容**

实现课本上的13-15页，堆栈应用例程。

主要算法： 1、堆栈数据结构的建立

2、出栈操作

3、进栈操作

**2 工程构建**

创建一个工程，将\*.cpp文件加入工程。



**3 程序代码**

#include "stdio.h"

#define maxsize 10 /\* 栈空间为 10 \*/

int data[maxsize+1]; /\* data[0]--栈顶指针 \*/

/\* 在栈顶插入数据\*/

void push\_stack(int x, int s[maxsize+1])

{

 printf(" 压栈前: top=%d x=%d\n", s[0], x);

 if (s[0]==maxsize)

 printf("OVERFLOW\n");

 else

 {

 s[0]=s[0]+1; s[s[0]]=x;

 printf(" 压栈后: ");

 printf(" top=%d s[top]=%d\n", s[0], s[s[0]]);

 printf("\n");

 }

}

/\*弹出栈顶元素\*/

void pop\_stack(int s[maxsize+1])

{

 if (s[0]==0)

 printf("栈空!\n");

 else

 {

 s[0]=s[0]-1;

 printf("出栈的元素是: %d; 当前栈顶是: %d", s[s[0]+1], s[0]);

 }

}

void main()

{

 for (k=1;k<=num;k++)

 {

 scanf("%d",&x);

 push\_stack(x,data);

 }

 printf("\n");

 printf("栈中元素为:");

 for (k=1;k<=maxsize;k++)

 printf("%d-",data[k]);

 printf("\n");

 /\*把8压入栈元素\*/

 d=8;

 push\_stack(d, data);

 /\*输出栈中元素\*/

 printf("栈中元素为:");

 for (k=1;k<=maxsize;k++)

 printf("%d|", data[k]);

 printf("\n");

 /\*出栈元素\*/

 pop\_stack(data);

 printf("栈中元素为:");

 for (k=1;k<=maxsize;k++)

 printf("%d|", data[k]);

 printf("\n");

}

**软件技术基础**

**实验3、队列应用**

**1、实验内容**

实现课本上的19-21页，队列应用例程。

主要算法： 1、置空队列

2、入队操作

3、出队操作

**2 工程构建**

创建一个工程，将\*.cpp文件加入工程。



**3 程序代码**

#include "stdio.h"

#include "math.h"

#define maxsize 10

int data[maxsize+3]; /\* data[maxsize+1]:队头;

 data[maxsize+2]:队尾 \*/

 /\*循环队列从0到maxsize \*/

/\*置空队列\*/

void makenull\_sequeue(int sq[maxsize+3])

{

 int k;

 sq[maxsize+1]=0; sq[maxsize+2]=0;

 for (k=0;k<=maxsize;k++)

 sq[k]=-1;

}

/\* 入队---在队尾加入元素\*/

void add\_sequeue(int x, int sq[maxsize+3])

{

 printf(" 当前队尾是%d 及其欲加入的元素:%d\n", sq[maxsize+2],x);

 if (fmod(sq[maxsize+2]+1,maxsize+1)==sq[maxsize+1])

 printf(" 上溢!\n"); /\* 尾指针加1等于头指针 \*/

 else

 {

 sq[maxsize+2]=(int)fmod(sq[maxsize+2]+1,maxsize+1); /\* 尾指针加1 \*/

 sq[sq[maxsize+2]]=x; /\* 入队 \*/

 printf(" 入队后:");

 printf(" 队头=%d 队尾=%d\n", sq[maxsize+1],sq[maxsize+2]);

 printf("\n");

 }

 return;

}

/\* 出队--删除队头元素 \*/

void del\_sequeue(int sq[maxsize+3])

{

 if (sq[maxsize+1]==sq[maxsize+2])

 printf("队列\*已空!\n");

 else

 {

 sq[maxsize+1]=(int)fmod(sq[maxsize+1]+1,maxsize+1); /\* 头指针加1 \*/

 printf("出队:\n 队头:%d", sq[maxsize+1]);

 sq[sq[maxsize+1]]=-1;

 printf(" 队尾:%d\n", sq[maxsize+2]);

 }

}

void main()

{

 int k,mem,x,num;

/\*

 double a=-4.2,b=3.0;

 double c=fmod(a, b);

 printf("%lf\n",c);

 printf( "------------------------------------------\n");

\*/

 makenull\_sequeue(data);

 del\_sequeue(data);

 printf("队列长最大值=%d\n", maxsize);

 printf("请输入比队长实际值少 1 的数字（比如实际队长是3，则请输入2）:");

 scanf("%d", &num);

 printf("请输入%d个数据", num+1);

 for (k=0; k<=num; k++)

 {

 }

 printf("\n");

 printf("队列中的元素:\n");

 for (k=0;k<=maxsize; k++)

 printf(" %d |", data[k]);

 printf("\n");

 for (k=1;k<=num+1;k++)

 del\_sequeue(data);

 printf("-------------12 入队 ---------\n");

 mem=12;

 add\_sequeue(mem,data);

 printf("-------------34 入队 ---------\n");

 mem=34;

 add\_sequeue(mem,data);

 printf("数组中的元素:");

 for (k=0;k<=maxsize; k++)

 printf(" %d |", data[k]);

 printf("\n");

}

**软件技术基础**

**实验4、链表应用**

**1、实验内容**

实现课本上的24-30页，链表应用例程。

主要算法： 1、链表节点结构

2、动态建立链表操作

3、查找节点操作

4、插入节点操作

5、删除节点操作

**2 工程构建**

创建一个工程，将\*.cpp文件加入工程。



**3 程序代码**

/\*改正了书中程序的错误！

使用head[0], 而不是 head[1]

 2014年10月16日

\*/

#include "stdlib.h"

#include "stdio.h"

#define NULL 0

/\*链表的节点结构\*/

struct link

{

 int data;

 struct link \*next;

};

/\*链表头指针\*/

struct link \* h[1];

/\*求线性表的长度\*/

int len\_slist(struct link \*h[1])

{

 int len=0;

 struct link \* p;

 p=h[0];

 if(p == NULL)

 {

 return 0;

 }

 while (p != NULL)

 {

 len++;

 p=p->next;

 }

 return len;

}

/\*创建链表， 输入 0 零结束\*/

void creat(struct link \* h[1])

{

 struct link \*r,\*s; int x;

 h[0]=NULL; /\*初始化，建立一个空表\*/

 printf("请输入数据，如2 4 5 3 0，（输入 0 零结束）: ");

 scanf("%d",&x);

 while (x!=0)

 {

 r->data=x; /\*生成一个数据域为data的结点\*/

 r->next=h[0];

 h[0]=r; /\*将新结点插入到单向链表的第一个结点之前\*/

 scanf("%d",&x);

 }

 printf("\n");

 s=h[0];

 printf("生成的单链表为: ");

 while (s!=NULL)

 {

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

 }

 printf("^\n");

}

/\*在链表中 查找search\_key，如找到，\*found = 1 。\*/

void search\_slist(struct link \*h[1], int search\_key, int \*found)

{

 struct link \*p;

 p=h[0]; /\*初始化，p指向第一个元素结点\*/

 \*found=0;

 while ((p!=NULL) && (! \*found))

 {

 if (search\_key!=p->data)

 p=p->next;/\*顺指针往后寻找直至找到或指针指向空\*/

 else

 \*found=1;

 }

}

/\*在链表中 第i个位置，插入intser\_key 。\*/

void insert\_slist(struct link \*h[1], int i, int insert\_key)

{

 struct link \*p,\*q,\*l;

 int j;

 q->data=insert\_key; /\* 生成数据域为insert\_key的结点\*/

 p=h[0]; l=p;j=1; /\* 初始化，p指向头结点\*/

 while ((p!=NULL) && (j<i)) /\*寻找第i个结点\*/

 {

 l=p;

 p=p->next;

 j=j+1;

 }

 if (p!=NULL)

 {

 /\*第27页，图b 图c 情况。在中间和头部插入\*/

 q->next=p->next; p->next=q;

 printf("插入的位置:%d,插入的数据:%d\n",i,q->data);

 }

 else

 {

 /\*第27页，图a 情况。在尾部插入\*/

 l->next=q; q->next=p;

 }

}

 /\*删除单向链表中第i个结点，head为头指针\*/

void delete\_slist(struct link \*h[1], int i)

{

 struct link \*p,\*q;

 int j;

 p=h[0];

 j=1;

 while ((p!=NULL) && (j<=i-1))

 { /\* 寻找第ｉ-１个结点，令P指向它 \*/

 q=p; p=p->next; j=j+1;

 }

 if (p==NULL) printf("WITHOUT!"); /\* i＞表长 \*/

 else

 {

 if (p==h[0])

 {

 /\*第28页，图a 情况。删除第一个节点。\*/

 h[0]=p->next;

 }

 else

 {

 /\*第28页，图b 图c 情况。删除中间或尾部节点。\*/

 q->next=p->next;

 }

 free(p); /\* 释放第i个结点 \*/

 }

 p=h[0]; printf("删除第%d结点后的表: ",i);

 while (p!=NULL)

 {

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

 }

 printf("^\n");

}

void main()

{

 struct link \*p;

 int x,key, exist;

/\*创建链表\*/

 creat(h);

 /\*在链表中找 30 \*/

 //printf("表长度 : %d \n ",len\_slist(h));

 printf("查找 30 的结果 : ");

 key=30;

 exist=0;

 search\_slist(h,key,&exist);

 printf("是否找到: %d\n",exist);

 /\*在链表中第2个位置，插入8\*/

 x=8;

 insert\_slist(h,2,x);

 printf("在位置 2 后 ， 插入 8 后的表: ");

 p=h[0];

 while (p!=NULL)

 {

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

 p=p->next;

 }

 printf("^\n");

/\*在链表中 删除第1个元素\*/

 x=1;

 delete\_slist(h,x);

}

**软件技术基础**

**实验5、二叉树应用**

**1、实验内容**

实现课本上的51-55页，二叉树应用例程。

主要算法： 1、二叉树节点结构

2、二叉树建立

3、先序遍历操作

4、中序遍历操作

5、后序遍历操作

**2 工程构建**

创建一个工程，将\*.cpp文件加入工程。



**3 程序代码**

#include "stdio.h"

#include "stdlib.h"

#include "malloc.h"

#define NULL 0

struct link

{

 struct link \*lchild,\*rchild;

 char data;

};

struct link \*root;

char ch;

struct link \*newnode()

{

 return((struct link \*) malloc (sizeof(struct link)));

}

struct link \*creattree()

{

 struct link \*tree;

 scanf("%c",&ch);

 printf("%c",ch);

 if (ch!='.')

 {

 tree->data=ch;

 tree->lchild=creattree();

 tree->rchild=creattree();

 }

 else

 tree=NULL;

 return(tree);

}

void preorder(struct link \*tree)

{

 if (tree!=NULL)

 {

 printf(" %c ",tree->data);

 preorder(tree->lchild);

 preorder(tree->rchild);

 }

}

void inorder(struct link \*tree)

{

 if (tree!=NULL)

 {

 inorder(tree->lchild);

 printf(" %c ",tree->data);

 inorder(tree->rchild);

 }

}

void postorder(struct link \*tree)

{

 if (tree!=NULL)

 {

 postorder(tree->lchild);

 postorder(tree->rchild);

 printf(" %c ",tree->data);

 }

}

void main()

{

 printf("开始创建书，请输入（书上例子ABD...CE.G..FH..I..）：\n");

 root=creattree();

 printf("\n");

 printf("先序遍历：");

 preorder(root);

 printf("\n");

 printf("中序遍历：");

 inorder(root);

 printf("\n");

 printf("后序遍历：");

 postorder(root);

 printf("\n");

}