**1.         输入一个链表的头结点，从尾到头反过来输出每个结点的值。链表结点定义如下：**

struct ListNode

{

      int       m\_nKey;

      ListNode\* m\_pNext;

};

A:    递归方法逆序输出，栈方法逆序输出。

(任意实现一种既可)

void PrintListUsingRecursicve(pListNode head)

{

    if(head!=NULL)

    {

      PrintListUsingRecursicve(head->m\_pNext);

      printf("%d/n",head->m\_nKey);

    }

}

void PrintListUsingStack(pListNode head)

{

    Stack s;

    s.top=0;

    pListNode p=head;

   do{

       push(&s,p->m\_nKey);

       p=p->m\_pNext;

   }while(p!=NULL);

   while(!IsEmpty(&s))

   {

       printf("%d/n",pop(&s));

   }

}

**2.         二元树的深度**

**题目：输入一棵二元树的根结点，求该树的深度。从根结点到叶结点依次经过的结点（含根、叶结点）形成树的一条路径，最长路径的长度为树的深度**。

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include<time.h>

#define MAXLEN 100

#define MAXNUM 10

typedef int Tree[MAXLEN];

Tree bt;

int GetDeep(int i)

{

    int l=0,r=0;

    if(bt[i\*2]!=-1)

    {

      l=GetDeep(i\*2)+1;

    }

    if(bt[i\*2+1]!=-1)

    {

        r= GetDeep(i\*2+1)+1;

    }

    return l>r?l:r;

}

int main()

{

    int i=0;

    memset(bt,-1,sizeof(bt));

    for(i=1;i<=MAXNUM;i++)

      bt[i]=i;

   bt[(i-1)\*2]=i\*2;

   printf("%d /n",GetDeep(1));

   return 0;

}

**3.         整数的二进制表示中1的个数**

题目：输入一个整数，求该整数的二进制表达中有多少个1。例如输入10，由于其二进制表示为1010，有两个1，因此输出2。

（关键是能不能想到后面的那个方法，只要想到这个方法既可）

int Bit1inInt(int i)

{

 int result=0;

 do{

  result+=i&1;

 }while(i=i>>1);

 return result;

}

**4.         从上往下遍历二元树**

**题目：输入一颗二元树，从上往下按层打印树的每个结点，同一层中按照从左往右的顺序打印。**

**（先序，中序，后序三种方式实现）**

**如果从上往下，从左到右的话只有一种遍历的方式：广度优先遍历。**

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include<time.h>

#define MAXLEN 100

#define MAXNUM 10

typedef int Tree[MAXLEN];

Tree bt;

typedef struct queue

{

    int begin,end;

    int space[MAXLEN];

}Queue;

int main()

{

    int i=0;

    memset(bt,-1,sizeof(bt));

    for(i=1;i<=MAXNUM;i++)

      bt[i]=i;

    Queue qe;

    qe.begin=0;qe.end =0;

    qe.space[qe.end++]=bt[1];

    while(qe.begin!=qe.end)

    {

        if(bt[2\*qe.space[qe.begin]]!=-1)//lchild

        {

            qe.space[qe.end++]=bt[2\*qe.space[qe.begin]];

        }

        if(bt[2\*qe.space[qe.begin]+1]!=-1)//rchild

        {

            qe.space[qe.end++]=bt[2\*qe.space[qe.begin]+1];

        }

        qe.begin++;

    }

    printf("--------------------/n");

    for(i=0;i<qe.end;i++)

    printf("%d ",qe.space[i]);

   return 0;

}

先序，中序，后序三种方式的只是遍历二元树

typedef int Tree[MAXLEN];

Tree bt;

void PreOrderTraverse(int i)

{

   if(bt[i]==-1) {return ;}

        printf("%d ",bt[i]);

        PreOrderTraverse(i\*2);//lchild

        PreOrderTraverse(i\*2+1);//rchild

}

void InOrderTraverse(int i)

{

   if(bt[i]==-1) {return ;}

        InOrderTraverse(i\*2);//lchild

        printf("%d ",bt[i]);

        InOrderTraverse(i\*2+1);//rchild

}

void PostOrderTraverse(int i)

{

   if(bt[i]==-1) {return ;}

        PostOrderTraverse(i\*2);//lchild

        PostOrderTraverse(i\*2+1);//rchild

        printf("%d ",bt[i]);

}

 int main()

 {

    int i=0;

    memset(bt,-1,sizeof(bt));

    for(i=1;i<=MAXNUM;i++)

      bt[i]=i;

    printf("/n---------------/n");

    PreOrderTraverse(1);

    printf("/n---------------/n");

    InOrderTraverse(1);

    printf("/n---------------/n");

    PostOrderTraverse(1);

     return 0;

 }

**5.         查找链表中倒数第k个结点**

**题目：输入一个单向链表，输出该链表中倒数第k个结点。链表的倒数第0个结点为链表的尾指针。链表结点定义如下：**

struct ListNode

{

      int       m\_nKey;

      ListNode\* m\_pNext;

};

         （最快的方法，只遍历一遍）

int FindCoundDownInList(pListNode head,int num)

{

    pListNode p1,p2;

    p1=p2=head;

    while(num-->0 && p1!=NULL) p1=p1->m\_pNext;

    if(p1==NULL) return 0;

    else{

    while(p1!=NULL)

    {

        p1=p1->m\_pNext;

        p2=p2->m\_pNext;

    }

    return p2->m\_nKey;

    }

}

**6.         求三角形面积**

**给出三角形的三个边长为a、b、c，求三角形的面积。**

**（注意考虑是不是三角形）**

     double GetArea(int a,int b,int c)

{

    if(a-b>=c || a+b<=c)

    return -0.1;

    else{

    double s=0.5\*(a+b+c);

    double area=sqrt(s\*(s-a)\*(s-b)\*(s-c));

    return area;

    }

}

**7.         压缩字符串**

**例如字串”aaabbbbccccc”，转换成相邻字符+个数的形式压缩，成为”a3b4c5”。**

**（如果有10个数相同）**

**假设需要考虑解压缩**

char \*MergeString(const char \* ch)

{

    char \*s=(char \*)malloc(sizeof(ch));

    if(s!=NULL)

    {

     int len=strlen(ch), i=0,j=0,k=0;

     for(;i<len;i=j)

     {

        int num=0;

        while(ch[j]==ch[i]) j++,num++;

        s[k++]=ch[i];

        sprintf(s+k,"%d",num);

        k=strlen(s);

     }

    }

   return s;

}

**8.         如何判断一个单向链表是否有环。**

int ISCircle(pListNode head)

{

    pListNode p1=head;

    p1=p1->m\_pNext;

    while(p1!=NULL)

    {

        if(p1==head) return 1;

        else p1=p1->m\_pNext;

    }

    return 0;

}

**9.         判断一个字符串是否对称。**

aabbaa , efffe返回true

aabac返回false

int   SymmetricString( const char \*ch)

{

    int len=strlen(ch);

    int i=0,j=len-1;

    if(len%2!=0) return 0;

    for(i=0,j=len-1;i<=len/2;i++,j--)

    {

        if(ch[i]!=ch[j]) return 0;

    }

    return 1;

}

**10.     判断一个字符串是否是另一个字符串的字串**

int next[20];

void get\_next(const char\* T,int next[])

{

    int i=0,j=-1;next[0]=-1;

    int len=strlen(T);

    while(i<len)

    {

        if(j==-1||T[i]==T[j]) {++i;++j;next[i]=j;}

        else j=next[j];

    }

}

int index\_KMP(const char \* S,const char \* T)

{

    int  i=0,j=0;

    get\_next(T,next);

    int lens=strlen(S),lent=strlen(T);

    while(i<lens &&j<lent){

    if(j==-1 ||S[i]==T[j]){++i;++j;}

    else j=next[j];    }

    if(j>=lent) return i-lent;

    else return -1;

}

链表的定义，栈的定义：

typedef struct stack

{

    int top;

    int space[MAXLEN+1];

}Stack;

int push(Stack \*s,int num)

{

    if(s->top>=sizeof(s->space)/sizeof(int)) return -1;//Error

    s->space[s->top++]=num;

    return num;

}

int pop(Stack \*s)

{

    if(s->top<0) return -1;

    return s->space[--s->top];

}

int IsEmpty(Stack \*s)

{

    return s->top==0;

}

typedef struct ListNode

{

    int m\_nKey;

    struct ListNode \*m\_pNext;

}ListNode,\*pListNode;

pListNode CreateList()

{

    srand((unsigned long)time(NULL));

    pListNode head,p1,p2;

    head=(pListNode)malloc(sizeof(ListNode));

    p1=head;p2=p1;

   int i=MAXLEN;

   while(i--){

       p2=(pListNode)malloc(sizeof(ListNode));

   p2->m\_nKey= rand()\*rand()%(MAXLEN\*rand());

   p1->m\_pNext=p2;

   p1=p2;

   }

   p2->m\_pNext=NULL;

   return head;

}

void PrintList(pListNode head)

{

    pListNode p=head;

   do{

       printf("%d/n",p->m\_nKey);

       p=p->m\_pNext;

   }while(p!=NULL);

}