```
//7.1 表达式求值
typedef struct lnode
{
    char data;
    struct lnode *next;
}SNode,*SLink;
void Init(SLink *LS);
int Push(SLink *LS,char e);
int Pop(SLink *LS,char *e);
int GetTop(SLink S,char *e);
int BracketMatch(char exp[]);
int EmptyStack(SLink S);
int charMatch(char c1,char c2 );
int ExpToSuffix(char exp[],char suffix[]);
void Calculate(char suffix[],int *v);
int isDigit(char ch);
int Operate(char e1,char e2,char ch);

int _tmain(int argc, _TCHAR* argv[])
{
    char exp[]="3+((1+1)*(2-1))#";//"(((1+2)*(3-2)+1)+2-1+3-2)/2+2-2+1*1#";
    int i = BracketMatch(exp);
    char suffix[30];
    ExpToSuffix(exp,suffix);
    int v;
    Calculate(suffix,&v);
    return 0;
}
void Calculate(char suffix[],int *v)
{
    SLink S;
    Init(&S);
    char *p = suffix;
    char ch = *p;
    char e1,e2;
    int t;
    while(ch!='\0')
    {
        if(isDigit(ch))
            Push(&S,ch);
        else
        {
            Pop(&S,&e2);
            Pop(&S,&e1);
            t = Operate(e1,e2,ch);
            Push(&S,t+'0');
        }
        p++;
        ch = *p;
    }

    Pop(&S,&e1);
    *v = e1-'0';
}
int Operate(char e1,char e2,char ch)
{
    if(ch == '+')
        return e1-'0'+e2-'0';
    else if(ch == '-')
        return e1-'0'-(e2-'0');
    else if(ch == '*')
        return (e1-'0')*(e2-'0');
    else if(ch == '/')
        return (e1-'0')/(e2-'0');
    else
        return 10000;
}
int isDigit(char ch)
{
    if(ch>='0'&&ch<='9')
        return 1;
    else
        return 0;
}
int isOper(char ch)
{
    int j = ch ==' ('||ch==')' ||
        ch ==' +' || ch ==' -'|| ch ==' /' || ch ==' *';

    return j;
}

int level(char ch)
{
    if(ch ==' +' || ch ==' -')
        return 1;
    else if(ch ==' /' || ch ==' *')
        return 2;
    else if(ch == ' ('|| ch == '#' )
        return 0;
    else
        return -1;
}
int ExpToSuffix(char exp[],char suffix[])
{
    char *p = exp;
    char ch=*p;
    char e;
    int j=0;

    SLink S;
    Init(&S);
    Push(&S,'#');
    while(ch!='#')
    {
        if(!isOper(ch))
            suffix[j++]=ch;
        else
        {
            if(ch=='(')
               Push(&S,ch);
            else if(ch == ')')
            {
                while(Pop(&S,&e) && e !='(')
                    suffix[j++]=e;
            }
            else{
                while(GetTop(S,&e)&&level(ch)<=level(e))
                {
                    Pop(&S,&e);
                    suffix[j++]=e;
                    GetTop(S,&e);
                }
                if(ch !='#')Push(&S,ch);
            }
        }
        p++; ch = *p;
    }
```

```
    while(Pop(&S,&e)&& e!='#')
        suffix[j++]=e;
    suffix[j] ='\0';
    return 1;
}

int isEmptyStack(SLink S)
{
    return S == NULL;
}
int GetTop(SLink S,char *e)
{
    if(S==NULL)
        return 0;
    *e = S->data;
    return 1;
}
int charMatch(char c1,char c2 )
{
    if(c1 == '(' && c2==')')
        return 1;
    if(c1 == '{' && c2 == '}')
        return 1;
    if(c1=='[' && c2 == ']')
        return 1;
    return 0;
}
int BracketMatch(char exp[])
{
    char *p = exp;
    SLink S;
    Init(&S);
    while(*p)
    {
        if(*p == '(' || *p == '[' || *p == '{')
            Push(&S,*p);
        else if(*p == ')' || *p == ']' || *p == '}')
        {
            char e;
            if(isEmptyStack(S)==0)
            {
                GetTop(S,&e);
                if(charMatch(e,*p))
                    Pop(&S,&e);
                else
                    break;
            }
            else if(isEmptyStack(S))
                break;
        }
        p++;
    }
    if(isEmptyStack(S) && *p == '\0')
        return 1;
    else
        return 0;
}
int Pop(SLink *LS,char *e)
{
    if(*LS == NULL)
        return 0;
    *e = (*LS)->data;
    *LS=(*LS)->next;
    return 1;
}
void Init(SLink *LS)
{
    *LS = NULL;
}
int Push(SLink *LS,char e)
{
    SLink s = NULL;
    s = new SNode;
    if(!s)
        return 0;
    s->data = e;
    s->next = *LS;
    *LS = s;
    return 1;
}
/*7.2 迷宫求解*/
typedef int ARRAY[MAXSIZE][MAXSIZE];
ARRAY maze=
{{1,1,1,1,1,1,1,1,1,1},
{1,0,0,1,0,0,0,1,0,1},
{1,0,0,1,0,0,0,1,0,1},
{1,0,0,0,0,1,1,0,0,1},
{1,0,1,1,1,0,0,0,1,1},
{1,0,0,0,1,0,0,0,1,1},
{1,0,1,0,0,0,1,0,0,1},
{1,1,1,1,1,0,1,1,0,1},
{1,0,0,0,0,0,0,0,0,1},
{1,1,1,1,1,1,1,1,1,1}};
typedef struct
{
    int m,n,d;
}POI;
typedef struct
{
    POI elem[MAXSIZE];
    int top;
}SqStack;


int GetTop(SqStack S,POI *e);
int Pop(SqStack *S,POI *e);
void Init(SqStack *S);
int Push(SqStack *S,POI e);
void Print(SqStack S);
void FindPath(ARRAY maze,SqStack *LS);

int _tmain(int argc, _TCHAR* argv[])
{
    SqStack s;
    Init(&s);
    FindPath(maze,&s);
    return 0;
}
void Init(SqStack *S)
{
    S->top = -1;
}
int GetTop(SqStack S,POI *e)
{
    if(S.top == -1)
        return 0;
    if(S.top>-1)
```

```
        *e = S.elem[S.top];
    return 1;
}
int Push(SqStack *S,POI e)
{
    if(S->top == MAXSIZE-1)
        return 0;
    S->elem[++S->top]=e;
    return 1;
}
int Pop(SqStack *S,POI *e)
{
    if(S->top == -1)
        return 0;
    *e = S->elem[S->top--];
    return 1;
}

void FindPath(ARRAY maze,SqStack *LS)
{
    int flag[10][10]={0};
    int i,j;

    i=1,j=1;
    POI e;
    e.m = 1;
    e.n = 1;
    e.d = 1;

    do{
        if(e.d<4 && !maze[e.m][e.n]&&!flag[e.m][e.n])
        {
```

```
        Push(LS,e);
        flag[e.m][e.n]=1;
        if(e.m==8 && e.n==8)
            break;
        else
         {   e.d=1;
             e.n++;
        }
    }
    else
    {
        if(LS->top!=-1)
        {
            Pop(LS,&e);
            if(e.d==1)
            {
                e.d++;
                Push(LS,e);
                e.m++;
                e.d = 1;
            }
            else if(e.d==2)
            {   e.d++;
                Push(LS,e);
                e.n--;
                e.d = 1;
            }
            else if(e.d==3)
            {   e.d++;
                Push(LS,e);
                e.m--;
                e.d = 1;
                }
                else
                {
                    GetTop(*LS,&e);
                    if(e.d==1)
                   {
                    e.d++;
                    e.m++;}
                    else if(e.d==2)
                    {   e.d++;
                    e.n--;
                    }
                    else if(e.d==3)
                {   e.d++;
                    e.m--;
                }
                }
            }
        }
    }while(LS->top!=-1);
}
```