实习报告规范

实习报告的开头应给出题目、班级、姓名、学号和完成日期，并包括以下七个内容：

1. 需求分析

以无歧义的陈述说明程序设计任务，强调的是程序要做什么？明确规定：

（1） 输入的形式和输入值的范围；

（2） 输出的形式；

（3） 程序所能达到的功能；

（4） 测试数据：包括正确的输入及其输出结果和含有错误的输入及其输出结果。

2. 概要设计

说明本程序中用到的所有抽象数据类型的定义、主程序的流程以及各程序模块之间的层次（调用）关系。

3. 详细设计

实现概要设计中定义的所有数据类型，对每个操作只需要写出伪码算法；对主程序和其他模块也都需要写出伪码算法（伪码算法达到的详细程度建议为：按照伪码算法可以在计算机键盘直接输入高级程序设计语言程序）；画出函数的调用关系图。

4. 调试分析

内容包括：

(1) 调试过程中遇到的问题是如何解决的以及对设计与实现的回顾讨论和分析；

(2) 算法的时空分析（包括基本操作和其他算法的时间复杂度和空间复杂度的分析）和改进设想；

(3) 经验和体会等。

5. 用户使用说明

说明如何使用你写的程序，详细列出每一步的操作步骤。

6. 测试结果

列出你的测试结果，包括输入和输出。这里的测试数据应该完整和严格，最好多于需求分析中所列。

7. 附录

带注释的源程序。如果提交源程序软盘，可以只列出程序文件名的清单。

在以下各实习单元中都提供了实习报告实例。值得注意的是，实习报告的各种文档资料，如：上述中的三个部分要在程序开发的过程中逐渐充实形成，而不是最后补写（当然也可以应该最后用实验报告纸誊清或打印）。

实习报告规范示例：

实习报告示例：2.9 题 迷宫问题

题目：编制一个求解迷宫通路的程序。

班级：计算机 95（1）　　姓名：丁一　　学号：954211　　完成日期：1996.10.9

一、 需求分析

（1）以二维数据 Maze[m+2][n+2]表示迷宫，其中：Maze[0][j]和 Maze[m+1][j]（0<=j<=n+1）及 Maze[i][0]和 Maze[i][n+1](0<=i<=m+1)为添加一圈障碍。数组中以元素值为 0 表示通路，1 表示障碍，限定迷宫的大小 m,n<=10。

（2）用户以文件的形式输入迷宫的数据：文件中第一行的数据为迷宫的行数 m 和列数 n；从第 2 行至第 m+1 行（每行 n 个数）为迷宫值，同一行中的两个数字之间用空白字符相隔。

（3）迷宫的入口位置和出口位置可由用户随时设定。

（4）若设定的迷宫存在通路，则以长方阵形式将迷宫及其通路输出到标准输出文件（即终端）上，其中，字符“#”表示障碍，字符“*”表示路径上的位置，字符“@”表示“死胡同”，即曾途径然而不能到达出口的位置，余者用空格符印出。若设定的迷宫不存在通路，则报告相应信息。

（5）本程序只求出一条成功的通路。然而，只需要对迷宫求解的函数作小量修改，便可求得全部路径。

（6）测试数据见原题，当入口位置为（1，1），出口位置为（9，8）时，输出数据应为：

*	*	#	@	@	@	#	
	*	#	@	@	@	#	
*	*	@	@	#	#		#
*	#	#	#			#	@
*	*	*	#	*	*	*	@
	#	*	*	*	#	*	#
	#	#	#	#		*	#
#	#				#	*	#
#	#					*	*

（7）程序执行的命令为：

1）创建迷宫；2）求解迷宫；3）输出迷宫的解。

二、 概要设计

1. 设定栈的抽象数据类型定义：

ADT Stack{

数据对象：D={a_i|ai∈ CharSet,i=1,2,…,n,n>=0}

数据关系：R1={<a_{i-1},a_i>|a_{i-1},a_i∈D,i=2,…,n}

基本操作：

InitStack(&S)

操作结果：构造一个空栈 S。

DestroyStack（&S）

初始条件：栈 S 已存在。

操作结果：销毁栈 S。

ClearStack(&S)

初始条件：栈 S 已存在。

操作结果：将 S 清为空栈。

StackLength(&S)

初始条件：栈 S 已存在。

操作结果：返回栈 S 的长度。

StackEmpty(&S)

初始条件：栈 S 已存在。

操作结果：若 S 为空栈，则返回 TRUE，否则返回 FALSE。

GetTop(S,&e)

初始条件：栈 S 已存在。

操作结果：若栈 S 不空，则以 e 返回栈顶元素。

Push(&S,e)

初始条件：栈 S 已存在。

操作结果：在栈 S 的栈顶插入新的栈顶元素 e。

Pop(&S,&e)

初始条件：栈 S 已存在。

操作结果：删除 S 的栈顶元素，并以 e 返回其值。

StackTraverse(S,visit())

初始条件：栈 S 已存在。

操作结果：从栈底到栈顶依次对 S 中的每个元素调用函数 visit( )。

}**ADT** Stack

2. 设定迷宫的抽象数据类型为：

ADT maze{

数据对象：D={$a_{i,j}$|$a_{i,j}$∈{ ‘ ’、‘#’、‘@’、‘*’ }，0≤i≤m+1,0≤j≤n+1,m,n≤10}

数据关系：R={ROW，COL}

ROW={<$a_{i-1,j}$,$a_{i,j}$>|$a_{i-1,j}$，$a_{i,j}$∈D,i=1,……，m+1,j=0,……，n+1}

COL={<$a_{i,j-1}$,$a_{i,j}$>|$a_{i,j-1}$,$a_{i,j}$∈D,i=0,……，m+1,j=1,……，n+1}

基本操作：

InitMaze(&M,a,row,col)

初始条件：二维数组 a[row+2][col+2]已存在，其中自第 1 行至第 row+1 行、每行中自第 1 列至第 col+1 列的元素已有值，并且以值 0 表示通路，以值 1 表示障碍。

操作结果：构成迷宫的字符型数组，以空白字符表示通路，以字符‘#’表示障碍，并在迷宫四周加上一圈障碍。

MazePath(&M)

初始条件：迷宫 M 已被赋值。

操作结果：若迷宫 M 中存在一条通路，则按如下规定改变迷宫 M 的状态：以字符“*”表示路径上的位置，字符“@”表示“死胡同”；否则迷宫的状态不变。

PrintMaze(M)

初始条件：迷宫 M 已存在。

操作结果：以字符形式输出迷宫。

}**ADT** maze;

3. 本程序包含三个模块

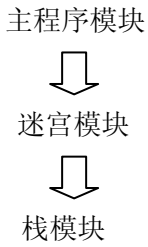
1） 主程序模块：

```
void main( )
{
    初始化；
    do{
        接受命令；
        处理命令；
    }while(命令！=“退出”);
}
```

2） 栈模块——实现栈抽象数据类型

3） 迷宫模块——实现迷宫抽象数据类型

各模块之间的调用关系如下：



4. 求解迷宫中一条通路的伪码算法：

```
设定当前位置的初值为入口位置；
do{
    若当前位置可通，
    则{将当前位置插入栈顶；              //纳入路径
       若该位置是出口位置，则结束；     //求得路径存放在栈中
       否则切换当前位置的东邻方块为新的当前位置；
      }
    否则{
        若栈不空且栈位置尚有其他方向未被探索，
            则设定新的当前位置为沿顺时针方向旋转找到的栈顶位置的下一相邻块；
        若栈不空但栈顶位置的四周均不可通，
            则{删去栈顶位置；              //后退一步，从路径中删去该通道块，
```

若栈不空，则重新测试新的栈顶位置，
直到找到一个可通的相邻块或出栈至栈空；
}
}
}**while**(栈不空)；
{栈空说明没有路径存在}

三、 详细设计

1. 坐标位置类型

```
typedef struct{
  int r,c;      //迷宫中行、列的范围
}PosType;
```

2. 迷宫类型

```
typedef struct{
   int    m,n;
   char   arr[RANGE][RANGE];  //各位置取值‘ ’,‘#’,‘@’或‘*’
}MazeType;
void initmaze(MazeType &maze,int a[][],int row,int col)
  //按照用户输入的 row 行和 col 列的二维数组（元素值为 0 或 1）
  //设置迷宫的初值，包括加上边缘一圈的值
bool mazepath(MazeType &maze,PosType start,PosType end)
  //求解迷宫 maze 中，从入口 start 到出口 end 的一条路径
  //若存在，则返回 TRUE；否则返回 FALSE
void printmaze(mazetype maze)
  //将迷宫以字符型方阵的形式输出到标准输出文件上
```

3. 栈类型

```
typedef struct{
    int         step;          //当前位置在路径上的“序号”
    PosType     seat;          //当前的坐标位置
    directiveType di;          //往下一坐标位置的方向
}ElemType;                 //栈的元素类型
typedef struct NodeType{
    ElemType    data;
    NodeType    *next;
}NodeType,*LinkType; //结点类型，指针类型
typedef struct{
    linktype   top;
    int    size;
}Stack;      //栈类型
```

栈的基本操作设置如下：

```
void InitStack(Stack &S)
    //初始化，设 S 为空栈（S.top=NULL）
void DestroyStack(Stack &S)
    //销毁栈 S，并释放所占空间
void ClearStack(Stack S)
    //将 S 清为空栈
int StackLength(Stack S)
    //返回栈 S 的长度 S.size
Status Stackempty(Stack S)
    //若 S 为空栈（S.top==NULL），则返回 TRUE；否则返回 FALSE
Status GetTop(Stack S,ElemType e)
    //若栈 S 不空，则以饿带回栈顶元素并返回 TRUE,否则返回 FALSE
Status Push(Stack &S,ElemType e)
    //若分配空间成功，则在 S 的栈顶插入新的栈顶元素 e，并返回 TRUE；
    //否则栈不变，并返回 FALSE
Status Pop(Stack &S,ElemType &e)
    //若栈不空，则删除 S 的栈顶元素并以 e 带回其值，且返回 TRUE
    //否则返回 FALSE
Status StackTraverse(Stack S,Status(*visit)(ElemType e))
    //从栈底到栈顶依次对 S 中的每个结点调用函数 visit
```

其中部分操作的算法：

```
Status Push(Stack &S,ElemType e)
{
    //若分配空间成功，则在 S 的栈顶插入新的栈顶元素 e，并返回 TRUE；
    //否则栈不变，并返回 FALSE
    if(MakeNode(p,e)){
        P->next=S.top;S.top=p;
        S.size++;return TRUE;
    }
    else return FALSE;
}
Status Pop(Stack & S, ElemType & e)
{
    //若栈不空，则删除 S 的栈顶元素并以 e 带回其值，且返回 TRUE，
    //否则返回 FALSE，且 e 无意义
    if (StackEmpty(S)) return FALSE;
    else {
        p = S.top; S.top = S.top->next;
        e = p->data; S.size--; return TRUE;
    }
}
```

4. 求迷宫路径的伪码算法：

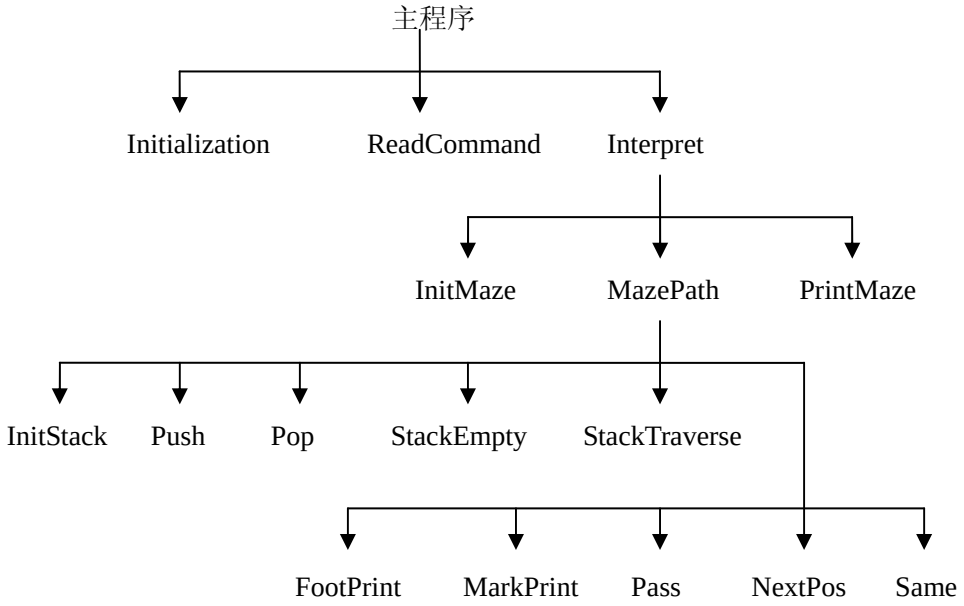
```
Status MazePath (MazeType maze, PosType start, PosType end)
{
    //若迷宫 maze 中存在从入口 start 到出口 end 的通道，则求得一条存放在栈中
    //（从栈底到栈顶为从入口到出口的路径），并返回 TRUE；否则返回 FALSE
    InitStack(S); curpos = start;        //设定“当前位置”为“入口位置”
    curstep = 1; found = FALSE;      //探索第一步
    do{
        if(Pass(maze, curpos)){
            //当前位置可以通过，即是未曾走到过的通道块留下足迹
            FootPrint (maze,curpos);
            e=(curstep,curpos,1);
            Push(S,e);          //加入路径
            if(Same(curpos,end) found=TRUE;    //到达终点（出口）
            else{
                curpos=NextPos(curpos,1);        //下一位置是当前位置的东邻
                curstep++;          //探索下一步
            }//else
        }//if
        else            //当位置不能通过
            if(!StackEmpty(S)){
                Pop(S,e);
                while(e.di==4&&!StackEmpty(S)){
                    MarkPrint(maze,e,seat);Pop(S,e);
                    Curstep--;          //留下不能通过的标记，并退回一步
                }//while
                if(e.di<4){
                    e.di++;Push(S,e);        //换下一个方向探索
                    curpos=NextPos(e.seat,e.di);      //设定当前位置是该新方向上
        的相邻块
                }//if
            }//if
    }while(!StackEmpty(S)&&!found);
    return found;
}//MazePath
```

5.主函数和其他函数的伪码算法

```
void main()
{
    //主程序
    Initialization();    //初始化
    do{
        ReadCommand(cmd);//读入一个操作命令符
        Interpret(cmd);          //解释执行操作命令符
    }while(cmd!='q'&&cmd!='Q');
```

```
}//main
void Initialization()
{
    //系统初始化
    clrscr();//清屏
    在屏幕上方显示操作命令清单：
    CreatMaze—c MazePath—m PrintMaze—p Quit—q;
    在屏幕下方现实操作命令提示框；
}//Initialization
void ReadCommand(char &cmd)
{
    //读入操作命令符
    显示键入操作命令符的提示信息；
    do{
        Cmd=getche();
    }while(cmd(['c','C','m','M','p','P','q','Q']));
}//ReadCommand
void Interpret(char cmd)
{
    //解释执行操作命令 cmd
    switch(cmd){
        case 'c','C': 提示用户输入“迷宫数据的文件名 filename”;
                      从文件读入数据分别存储在 rnum，cnum 和二维数组 a2 中；
                      InitMaze（ma，a2，rnum，cnum）；  //创建迷宫
                      输出迷宫建立完毕的信息；
                      break;
        case 'm','M': 提示用户输入迷宫的入口 from 和出口 term 的坐标位置；
                       if(MazePath(ma,from,term))      //存在路径
                          提示用户察看迷宫；
                       else 输出该迷宫没有从给定的入口到出口的路径的信息；
                       break;
        case 'p','P': PrintMaze(ma);      //将标记路径信息的迷宫输出到终端
    }//switch
}//Interpret
```

6.函数的调用关系图反映了演示程序的层次结构：



四、 调试分析

1．本次作业比较简单，只有一个核心算法，即求迷宫的路径，所以总的调试比较顺利，只在调试 MazePath 算法时，遇到两个问题：其一是，起初输出的迷宫中没有加上‘@’的记号，后发现是因为在 MarkPrint 函数中的迷宫参数丢失“变参”的原因；其二是，由于回退时没有将 curpos 随之减一，致使栈中路径上的序号有错。

2．栈的元素中的 step 域没有太多用处，可以省略。

3．StackTraverse 在调试过程中很有用，它可以插入在 MazePath 算法中多处，以察看解迷宫过程中走的路径是否正确，但对最后的执行版本没有用。

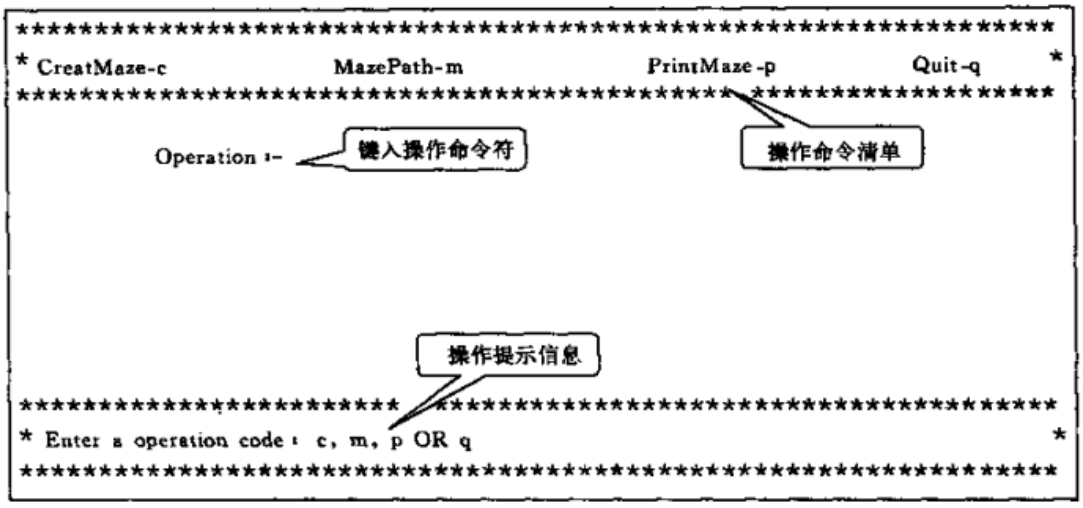
4．本题中三个主要算法：InitMaze,MazePath 和 PrintMaze 的时间复杂度均为 0（m*n），本题的空间复杂度亦为 0（m*n）（栈所占最大空间）

5．经验体会：借助 DEBUG 调试器和数据观察窗口，可以加快找到程序中疵点。

五、 用户手册

1． 本程序的运行环境为 DOS 操作系统，执行文件为：TestMaze.exe.

2． 进入演示程序后，即现实文本方式的用户界面：



3. 进入“产生迷宫（CreateMzae）”的命令后，即提示键入迷宫数据的文件名，结束符为“回车符”，该命令执行之后输出“迷宫已建成”的信息行。
4. 进入“求迷宫路径（MazePath）”的命令后，即提示键入入口位置（行号和列号，中间用空格分开，结束符为“回车符”）和出口位置（行号和列号，中间用空格分开，结束符为“回车符”），该命令执行之后输出相应信息。请注意：若迷宫中存在路径，则执行此命令之后，迷宫状态已经改变，若需要重复执行此命令，无论是否改变入口和出口的位置，均需要重新输入迷宫数据。
5. 输入“显示迷宫”的命令后，随即输出当前的迷宫，即迷宫的初始状态或求出路径之后的状态。

六、 测试结果

三组测试数据和输出结果分别如下：

1. 输入文件名为：m1.dat，其中迷宫数据为：

   3 2
   0 0
   0 0
   0 0
   入口位置：1 1
   出口位置：3 2
   求解路径后输出的迷宫：

*	*
	*
	*

2. 输入文件名：m2.dat，其中迷宫数据为：

   3 4
   0 0 0 0
   0 0 1 1
   0 0 0 0
   入口位置：1 1

出口位置：3 4

求解路径后输出的迷宫：

*	*	@	@
	*	#	#
	*	*	*

3. 输入文件名：m3.dat，其中迷宫数据同题目中的测试数据。

入口位置：1 1

出口位置：9 8

求解路径后输出的迷宫正确，并和需求分析中所列相同。

4. 输入文件名：m4.dat，其中迷宫数据为：

4 9

0 0 0 0 0 0 1 0 0

0 1 0 0 0 1 0 0 0

0 0 1 1 1 0 0 1 1

0 0 1 1 1 0 1 0 0

入口位置：1 1

出口位置：4 9

输出信息为：此迷宫从入口到出口没有路径。

七、 附录

源程序文件名清单：

base.H //公用的常量和类型

stkpas.H //栈类型

maze.H //迷宫类型

testmaze.C //主程序