**实验八　指针1**

|  |
| --- |
| 实验目标 |
| 1. 掌握指针的概念 2. 掌握指针的初始化和基本应用 3. 基本掌握指针用于函数传递参数 |

|  |
| --- |
| E-easy必做 N-normal必做 H-hard选做 |

1. **今天是哪天 E**

请编写一个程序，读入一个1~365的整数，忽略闰年。请判断今天是哪月哪天。比如输入：43，输出：2-12。为了练习指针传递参数，请用以下函数接口实现程序：

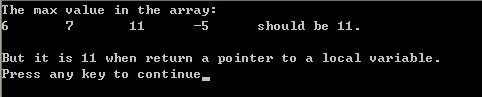
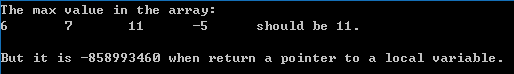
void getdate(int number,int \*month, int \*day);

* 1. 问题分析：请思考如何换算日期：switch~case，if else if，还是数组+循环方式

1. **调试2 M**

请下载程序returnlocalvaraddr.c。第一次直接运行程序，观察输出结果，即下图中红色部分；第二次将//extra for注释的循环指令去注释/\* \*/符号后激活，再次运行程序，观察输出，即下图中红色部分；体会返回不存在变量地址导致的问题。

注意：不同编译器下可能反应不同



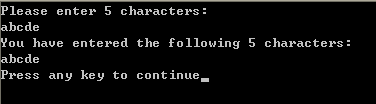
**Without executing function swap**

**executing function swap**

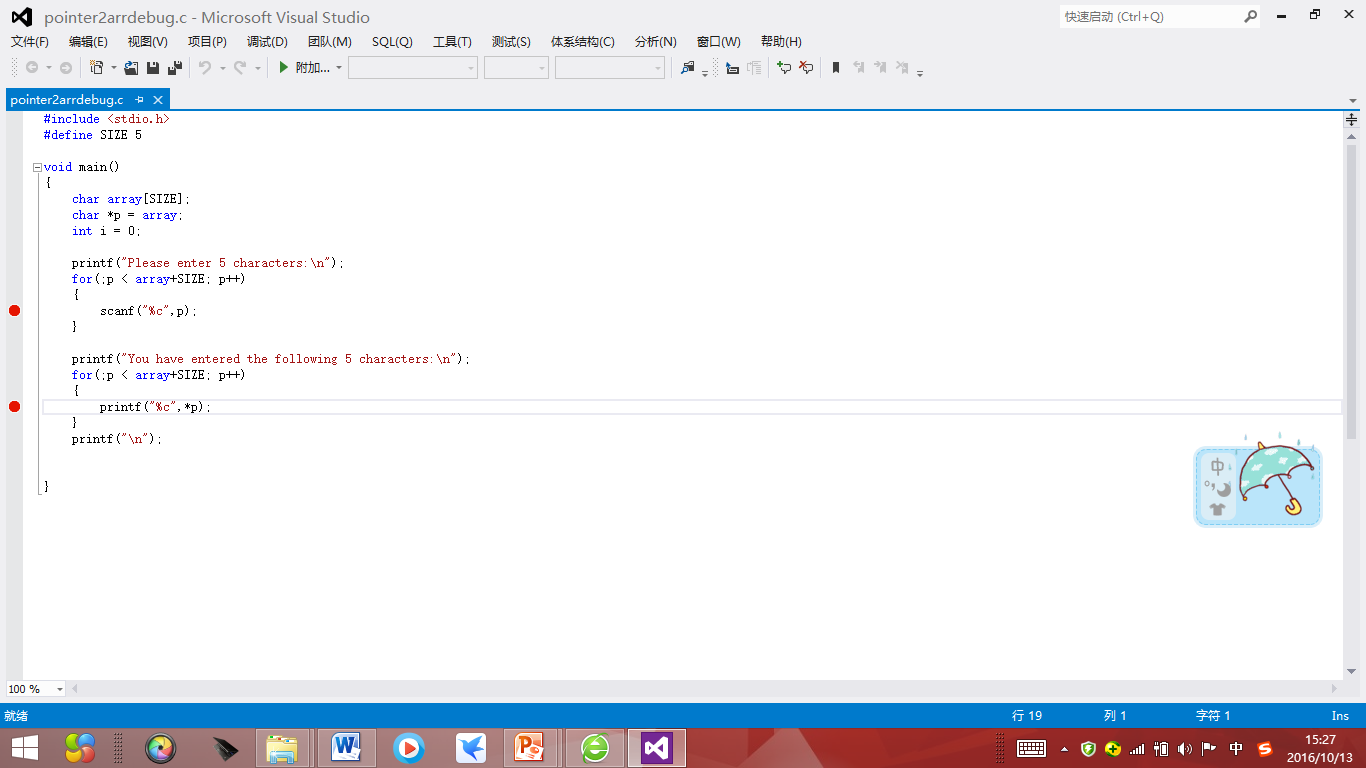
原因：

1. **调试 N**

下载框架程序pointer2arrdebug.c,程序预期输出效果如图

****

但实际情况并非如此，设置如图所示断点，



单步执行，并记录如下信息：

1号for循环：存入的5个元素地址为

1:p= 2:p= 3:p= 4:p= 5:p=

2号for循环：打印的5个元素地址为

1:p= 2:p= 3:p= 4:p= 5:p=

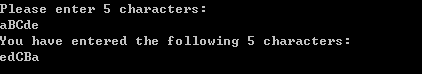
错误原因：

1. **指针练习 E**

下载框架程序prtbckwrdwithptr\_framework.c，用指针p完成对数组的初始化和倒序打印。

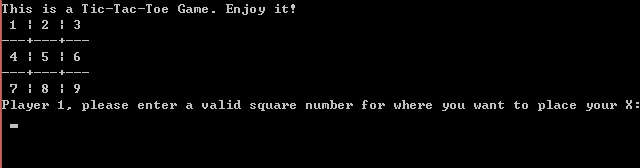
可以部分参考ppt page 59,60两个例子。

效果如图：



1. **Tic\_Tac\_Toe M**

Tic\_Tac\_Toe是三连棋游戏，游戏中两人轮流在一有九格方盘上划加字或圆圈, 谁先把三个同一记号排成横线、直线、斜线, 即是胜者。你可以先下载tic\_tac\_toe.exe试玩。



请将判断是否连成直线赢得比赛这一部分用函数完成，并且以指针方式传递数组。

* 1. 问题分析：程序的步骤非常明确，显示棋盘，读入用户棋子位置，刷新棋盘，判断输赢；以上步骤重复直至输赢确定或者9个位置填完,输出最后的结果。其中棋盘的表示是第一个难点，但图示已经给出了解决方案，即用1~9对位置进行编号,可以分别采用一维数组，二维数组，由于题设要求我们将输赢比较这一部分用指针完成，所以建议大家采用一维数组，愿意提升难度的同学可以选择二维数组；第二个难点是对棋盘的重画，这里可以采取不断在下方画出新棋盘，或者擦除老棋盘，画出新棋盘两种方式；第三个难点是对输赢的比较方案，需要分别搜索行、列和对角线，对角线1可以用board[0]==board[4],且board[4]=board[8],同理对角线2，行的比较可以用循环表示，同理列的比较也可以用循环表示。

行：board[i\*3+0] board[i\*3+1] board[i\*3+2]

列：board[i+3\*0] board[i+3] board[i+6]

根据题设要求，需要用指针方式传递数组，并在比较输赢这一函数中以指针方式实现数组元素的访问，因此需要将上述的数组下标模式转换为指针模式，比如board[i\*3+0]改写为\*(pboard+i\*3+0),并依次类推。

* 1. 下面分析主体程序伪代码：

main

define variables and board[9]

loop 9 times or winner==0

draw board

update player id:i%2+1

loop until get a valid input

read user input

verify user input :1~9 and board[input]is empty

call function :testplayerwin

if playerwin

update winner and break;

printf result

testplayerwin:

interface: return 1 if win, otherwise 0; paras:board

define variables

test diagonal

if win, return 1

loop from 0 to 2

test column

test row

if win, return 1

return 0(not win)

1. **EasyX教程6 N**
   1. 捕获按键，实现动画的简单控制

最常用的一个捕获按键的函数：getch()这个函数可以被当做“按任意键继续”来用，现在我们用变量保存这个按键：char c = getch();然后再判断即可。

不过程序执行到getch()是会阻塞的，知道用户有按键才能继续执行。可游戏中总不能因为等待按键而停止游戏执行吧？所以，要有一个函数，判断是否有用户按键：kbhit()这个函数返回当前是否有用户按键，如果有，再用getch()获取即可，这样是不会阻塞的。

即：

Char c;

If(kbhit())

C = getch();

举一个简单的例子，如果有按键，就输出相关按键。否则，输出“。”。每隔100毫秒输出一次。按esc退出。

注：esc的ASCII码是27.

完整代码如下：

#include<graphics.h>

#Include<stdio.h>

#include<conio.h>

Void main()

{

Char c = 0;

While(c !=27)

{

If(kbhit())

C = getch();

Else

C = ‘.’;

Printf(“%c”，c);

Sleep(100);

}

}

结合上一课简单动画，就可以做出来靠按键移动的图形了吧，看一下代码，实现a s控制圆的左右移动：

#include<graphics.h>

#include<conio.h>

Void main()

{

Initgraph(640,480);

Int x = 320;

//画初始图形

Setcolor(YELLOW);

Setfillstyle(GREEN);

Fillellipse(x,240,20,20);

Char c;

While(c != 27) //获取按键

{

C = getch();

//先擦出上次显示的旧图形

Setcolor(BLACK);

Setfillstyle(BLACK);

Fillellipse(x,240,20,20);

//根据输入，计算新的坐标

Switch(c)

{

Case ‘a’:

X -= 2; break;

case ‘d’:

x += 2; break;

}

//绘制新的图形

Setcolor(YELLOW);

Setfillstyle(GREEN);

Fillellipse(x,240,20,20);

//延时

Sleep(10);

}

Closegraph();

}

* 1. 请继续完成这个程序，实现以下功能：

1. 上下的控制

2. 边界的检测

3. 结合kbhit实现惯性移动（即按一下方向键，圆就会一直向这个方向移动）