作业8

- **Ex1.** 用指针 pa 操纵长度为 5 的数组 a（pa = a），设 pa 为 0X20000000，那么 pa + 2（元素 a[2] 的地址）为多少？若有“q = &a[3];”，则 q-pa 的值为多少？

- 0X20000000 + 2*sizeof(a[0])
- 3

- **Ex2.** 为什么 scanf 库函数采用了指针型参数，而 printf 库函数没有采用？

- scanf：利用函数的副作用，通过库函数修改主调函数中待输入变量的值
- printf：不需要修改待输出变量的值

第12周自主训练任务

1. 判断下列对指针类型数据的初始化是否恰当，并上机验证。

```
int m = 3;
float f;
int *pi;                    // 未初始化！下面的自增有问题
++pi;
pi = &f;                    // 类型有问题
pi = &a;                    // a 未初始化
float *pf = &f;
int a[5];
char ch;
char *pch = &ch;
int *pa = &a;               // 类型有问题
int *pn = pa;
int *pv = 0;
pn = 100;                   // 类型有问题，有潜在风险
pn = (int *)4096;
printf("*pn = %d \n", *pn);
```

2. 用指针pa操纵数组a：

（1）pa++表示pa先作为操作数，并将指向下一个元素，那么*pa++与*(pa++)是什么含义？

（2）++pa表示pa指向下一个元素，后作为操作数，那么*(pa=pa+1)、*(pa+=1)与*(++pa)的含义分别是什么？

（3）(*pa)++的含义呢？

（4）分析下面程序片段的输出结果。

```
int a[5] = {0, 1, 2, 3, 5};
int *pa = a;
int sum = 0;
for( ; pa <= a+4; ++pa)
     sum += *pa;
printf("%d ", sum);
pa = a;
++pa;
printf("%d ", *pa);
printf("%d ", *pa++);
printf("%d ", *pa);
printf("%d ", *(++pa));
++(*pa);
printf("%d \n", a[3]);
```

执行结果：1 1 2 3 4

（略）

3. 调试课件中的例子程序，学习指针作为函数参数的用法，并体会传址调用方式下函数的副作用。

（略）

4. 编程实现将一个int型数组（用指针操纵）中存放的元素完全颠倒顺序存放。例如，对于int data[7] = {0,1,2,3,4,5,6}; 执行后，data[0]为6，data[6]为0。

5. 某同学写了如下程序，希望用0替换m、n中较大的数。请分析该程序输出结果，如果结果不正确，改正程序，并说明理由。

```
#include <stdio.h>
int *F(int m1, int n1);
int main( )
{
      int m = 5, n = 9;
      *F(m, n) = 0;   //相当于“int *p = F(m, n);”和“*p = 0;”
      printf("%d, %d \n", m, n);
      return 0;
}
int *F(int m1, int n1)
{     int *pm = &m1, *pn = &n1;
      if(m1 > n1)
            return pm;
      else
            return pn;
} //返回局部变量的地址，存于临时空间
```

局部变量不在了，其地址？

悬浮指针

传址调用（此修改方案没有产生副作用）

```
#include <stdio.h>

int *F(int m1, int n1);

int main( )
{
    int m = 5, n = 9;
    *F(&m, &n) = 0;
    printf("%d, %d \n", m, n);
    return 0;
}
```

```
int *F(int *m1, int *n1)
{
    //int *pm = &m1, *pn = &n1;
      if(*m1 > *n1)
            return m1;
      else
            return n1;
}//返回主调函数局部变量的地址
```

实际参数还在，临时空间的值是实际参数的地址，

7. 解释下面表述的含义：

```
int * (*pfnPfnPp)(int(*)(int *, int), int **);
```

```
int *
      (*pfnPfnPp)
                 (                                  );
                                          , int **
                   int
                       (*)
                            (int *, int)
```

函数声明中的形参名可省略

函数声明中的形参名可省略

模拟测验

1. 读C程序代码，写结果：

(1)

```
char ch = getchar();
if('A' <= ch && ch <= 'Z')
```

的功能是（判断ch是否为大写英文字母）

(2)

```
const double PI = 3.14;
double r=1.0;
printf("%.2f \n", 4/3*PI*r*r*r); //cout << 4/3*PI*r*r*r << endl;
```

的输出结果是（3.14）

- 2.下面的PerSqu函数是用来“判断一个正整数是否为完全平方数”，找出其中的 bug，用注释的方式在错误行右侧写出对应的正确代码：

```
bool PerSqu(i)
{
      if(sqrt(i) = (int)sqrt(i))
            bool key = true;
      else
            key = false;

}
```

```
bool PerSqu(int i)
{
      if(sqrt(i) == (int)sqrt(i))
            return true;
      else
            return false;
}
```

- **3.下面代码片段的功能是“比较两个时刻的早晚（时分秒分别存储在变量h1、m1、s1和h2、m2、s2中）”，试对比做法一与做法二的优劣。**
- **做法一：**

```
int t1, t2;
t1 = s1 + m1*60 + h1*3600;
t2 = s2 + m2*60 + h2*3600;

if(t2 > t1)
      r = 1;
else if(t2 < t1)
      r = -1;
else
      r = 0;
```

此种做法计算量大，善于用比较运算往往能减少计算量(见下页)

- 做法二：

```
if(h2 > h1)          //先比较两个时刻的小时数，后者大
     r = 1;
else if(h2 < h1)     //后者小
     r = -1;
else if(m2 > m1)     //h1 == h2，比较两个时刻的分钟数
     r = 1;
else if(m2 < m1)
     r = -1;
else if(s2 > s1)     //后者大
     r = 1;
else if(s2 < s1)     //后者小
     r = -1;
else                 //相等
     r = 0;
```

风格好（有注释...）

- **4. 一辆卡车违反交通规则，撞人后逃跑。现场有三人目击事件，但都没有记住车号，只记下车号的一些特征。甲说：牌照的前两位数字是相同的；乙说：牌照的后两位数字是相同的，但与前两位不同；丙是数学家，他说：四位的车号刚好是一个整数的平方。请根据以上线索求出车号。**

```
int i, j, k, c;
for(i=0; i <= 9; ++i)              //i:车号前二位的取值
     for(j=0; j <= 9; ++j)         //j:车号后二位的取值
          if(i != j)
          {
               k = i*1000 + i*100 + j*10 + j;
               for(c=1; c*c < k; ++c); //判断是否为平方数
               if(c*c == k)
                    …
          }
```

请注意此分号！

- （选做）用多个函数实现一个日历显示程序。说明：年、月由用户输入，用按键增减月份、日历自动更新，日历形式如下（其中的汉字原样输出）：

2016 年 10 月

日	一	二	三	四	五	六
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

- 提示：有些开发环境提供的库函数 getche 或 _getche 可用来捕获光标键，并且不回显在屏幕上，先返回一个224再返回 72/80 时对应向上/下光标键。
- 可按公式 iWeek = ((c/4)-2*c*+*y*+(*y*/4)+(26(iMonth+1)/10)+iDay-1) % 7 将某年（iYear）某月（iMonth）的第一日（iDay）换算成星期（iWeek）。公式中：
  - 当iMonth为1时，iYear改为iYear-1，iMonth改为13
  - 当iMonth为2时，iYear改为iYear-1，iMonth改为14
  - c=iYear/100，c>15
  - y=iYear%100
  - 当iWeek<0时，iWeek改为iWeek+7
  - iWeek为0表示星期日

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>

int Year = 2016;
int Month = 10;

……//函数声明
```

```
int main(void)
{    scanf("%d%d", &Year, &Month);
     while(1)
     {
          DisMonthDays();//DisMonthDays(Year, Mont

          GetKey();
          printf("\n");
     }
     return 0;
}
```

```
void DisMonthDays() // void DisMonthDays(Year, Month)
{     int leap = 0;
      if((Year % 4 == 0 && Year % 100 != 0 || Year % 400 == 0))
            leap = 1;
      int monthday;
      switch(Month)
      {   ……
         case 2: monthday = (28 + leap); break;
         ……
         case 12: monthday = 31; break;
      }
      int week = 0;
      printf("        %d 年 %d 月\n", Year, Month);
      printf("日  一  二  三  四  五  六\n");
```

```
int MonthDays[][2] = {{0, 31,  28, 31, 30, 31, 30, 31, 3
                      {1, 31,  29, 31, 30, 31, 30, 31, 3
```

MonthDays[leap][Month]

```
    week = GetFirstDay(Year, Month, 1);
    for(int i=0; i < week; ++i)
        printf("    ");
    for(int i=1; i <= monthday; ++i)
    {
        printf("%2d  ", i);
        if(week >= 6)
        {    week = 0;          //清0，为下一行做准备
             printf("\n");
        }                       //换行
        else
             ++week;
    }
}
```

```
int GetFirstDay(int year, int month, int day)
{
    int week = 0;
    int c=0, y=0;
    if(month == 1)          //计算前先调整！！
    {
        --year;
        month = 13;
    }
    if(month == 2)
    {
        --year;
        month = 14;
    }
```

```
    c = year/100;
    y = year%100;
    week = (c/4)-2*c+y+(y/4)+(26*(month+1)/10)+day-1;
    week %= 7;
    if(week < 0)
    {
        week += 7;
    }
    return week;
}
```

```
void GetKey()
{     int key = getche();
      if(key == 224)        //判断是否为功能键
      {
            key = getche();
            if(key == 72)   //上键
            {
                  ++Month;
                  if(Month > 12)
                  {
                        Month = 1;
                        ++Year;
                  }
            }
```

```
while(input = getche())
{
      if(input == 72)
      {
            ……
//这样写，第一次循环 if 不成立
```

```
        else if(key == 80)   //下键
        {
            --Month;
            if(Month < 1)
            {
                Month = 12;
                --Year;
            }
        }
    }
}
```

11.（选做）八皇后问题：编程寻找在国际象棋的棋盘上放置八个皇后的正确样式（每种样式中的任何一个皇后所处的行、列及斜线上都不能有其他皇后），并用'Q'、' □'、空白符等符号模拟显示每种样式。

```
int main()
{
      Queen8(0);//从第一行开始放置
      return 0;
}
```

```
#include <stdio.h>
#define N 8
bool chessboard[N][N] = { false };

void Print()
{
    static int count = 1;
    printf("Picture %d:\n", count);
    ++count;
    for(int i = 0; i < N; ++i)
    {
      for(int j = 0; j < N; ++j)
        chessboard[i][j] == true?printf("%c ",'Q'):printf("□");
      printf("\n");
    }
    printf("\n");
}
```

```
#include <stdlib.h>
#include <stdbool.h>

bool IsOK(int i, int j)
{
    for(int m = 0; m < i; ++m) //与前面i个皇后( m 行 n 列)冲突否
    {
        int n = columnQ[m];
        if(n == j || abs(i - m) == abs(j - n))//同列、正反对角线
            return false;
    }
    return true;
}
```

```
int columnQ[N] = { -1 };          //存储第i行皇后的列号
void Queen8(int i)
{
    if(i >= N) //输出一局，结束本次递归，继续执行剩下的可能的 j
        Print();
    else        //递归地逐行放置
        for(int j = 0; j < N; ++j)     //逐列尝试
        {
            chessboard[i][j] = true; //假定可以
            if(IsOK(i, j))           //的确可以
            {
                columnQ[i] = j;
                Queen8(i + 1); //继续放置下一行，直到输出一局
            }
            chessboard[i][j] = false;//为下一列，以及下一局做准备
        }
}
```

落棋子

提棋子，或者 为下一局清除此处棋子

12.（选做）设计2048游戏程序。（游戏规则：在一个4×4的棋盘上，初值除了有两个2之外，其余数字均为0；玩家每次可以选择上下左右一个方向，每选一个方向，所有的数字往所选择的方向移动一次；系统同时会在0位置随机出现一个2或4，相同的数字在相遇时会相加；玩家要想办法在这小小的16格范围中凑出一个整数“2048”。程序要按一定的评分规则给玩家打分，例如，初始化一个总分，每选择一次方向减1分，至游戏结束给出最后得分，或者每使得数字相加时得10分，至游戏结束给出最后得分。凑出“2048”或没有0位置且没有相邻数字相同时，游戏结束。不需要画棋盘。）

步骤：

```
int step = 0;          //step记录步数，用于计分
bool deal = false;     //是否合并或移动过
初始化
输出
```

初始化棋盘 & 随机出数

```
void ini(int arr[][4], int n
{
	srand(time(0));
	int i = rand() % 4;
	int j = rand() % 4;
	arr[i][j] = 2;
	do
	{	i = rand() % 4;
		j = rand() % 4;
	} while(arr[i][j] != 0);
	 arr[i][j] = 2;
}
```

```
void ran(int arr[][4], int n)
{
	int a, b;
	srand(time(0));
	int i = 2 * (rand() % 2 + 1);
	do
	{
		a = rand() % 4;
		b = rand() % 4;
	} while(arr[a][b] != 0);
	arr[a][b] = i;
}
```

可继续玩吗?

```
bool game(int arr[][4], int n)
{
     bool go = false;
     for(int i = 0; i <= 2; ++i)
          for(int j = 0; j <= 2; ++j)
               if(arr[i][j] == 0 || arr[i + 1][j] == arr[i][j]
                              || arr[i][j + 1] == arr[i][j])
                    go = true;
     return go;
}
```

```
void up(int arr[][4], int n)
{    for(int k = 1; k <= 3; ++k)    //比较三次
          for(int i = 0; i <= 2; ++i)    //下面三行
               for(int j = 0; j <= 3; ++j)
                    if(arr[i][j] != 0)
                    {    if(arr[i][j] == arr[i + 1][j])
                         {    arr[i][j] *= 2;
                              arr[i + 1][j] = 0;
                              deal = true;
                         } //合并
                    }
                    else
                    {    arr[i][j] = arr[i + 1][j];
                         arr[i + 1][j] = 0;
                         if(arr[i][j] != 0)
                              deal = true;
                    }  //上移
}
```

main

```
bool go = true;
while(go == true)                    //游戏可继续
{       {键入方向；分别处理；}
        if(deal == true)                    //合并或移动过
            {计分；放随机数；deal = false;}
        {system("cls"); 输出}       //#include <conio.h>
        for(int i = 0; i <= 3; ++i)
            for(int j = 0; j <= 3; ++j)
                if(board[i][j] == 2048)
                    {win! 显示分数；return 0;}
        if((go = game(arr, 4)) != true)     //游戏无法继续
            {game over! 显示分数；return 0;}
}
```

Thanks !