1. C 语言贪吃蛇小游戏源码下载和思路解析

在《C 语言贪吃蛇小游戏演示和说明》一节中，我们对贪吃蛇游戏的玩法进行了介绍和演示，这节就来分析一下它的源码。

贪吃蛇源代码下载地址：https://pan.baidu.com/s/1pMk7nlx 密码：2yju

各位读者不妨先将源码下载下来浏览一遍，记住关键的几个函数，整理一下不了解的知识点，做到心中有数。

需要说明的是：贪吃蛇背景地图、食物、贪吃蛇本身都是由特殊字符组成（由 printf() 输出），并不是绘制出来的图形。C 语言标准库没有绘图函数，如果绘图的话，就需要使用第三方库，增加了大家的学习成本，所以我们采用了“投机取巧”的办法，用特殊字符来模拟不同的图形。

一. 关键知识点

下面请各位读者先学习一下该游戏中涉及到的几个关键知识点，有了这些必备条件，我们才好讲解贪吃蛇的设计思路。

1) 改变输出文本的颜色

贪吃蛇游戏的背景地图是是绿色的，边框是钻红色的，食物是红色的，贪吃蛇本身是黄色的，这就涉及到如何改变文本的输出颜色，请大家猛击《彩色版的 C 语言，让文字更漂亮》了解详情。

2) 在任意位置输出文本

在一般的程序中，字符都是依次输出的，例如当前控制台上显示的是“123456”，如果我们希望输出“abcd”，那么“abcd”就位于“123456”之后。在一般的程序中这是没有问题的，但是对于贪吃蛇游戏，我们需要自己来控制字符的输出位置，例如：

- 输出背景地图后，我们需要在背景地图中间输出贪吃蛇和食物；
- 要统计贪吃蛇吃掉的食物的数量，就必须不断改变同一位置的数字。

这是如何做到的呢？请大家猛击《C 语言在屏幕的任意位置输出字符》了解详情。

3) 键盘监听

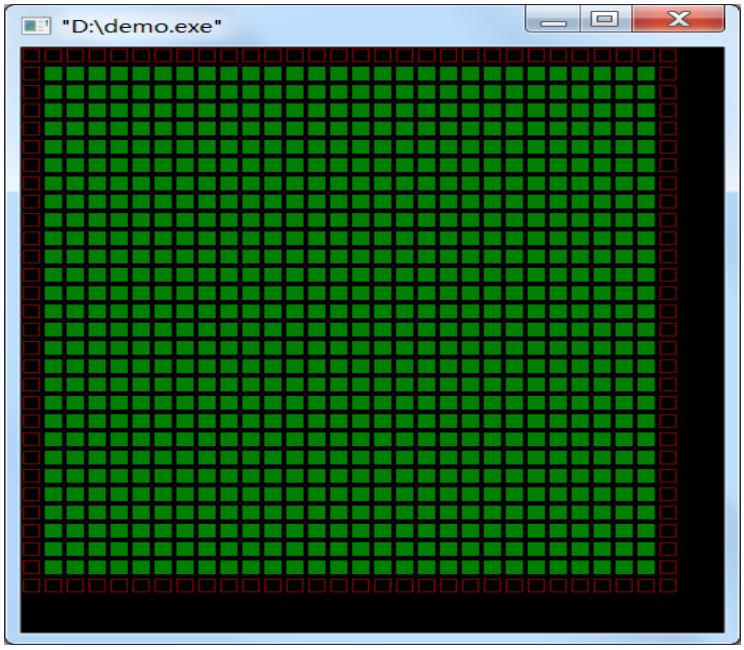
在贪吃蛇移动过程中，必须能够及时捕获用户按下的方向键，并改变移动方向，这是如何做到的呢？请大家猛击《C 语言非阻塞式键盘监听》了解详情。

4) 获取随机数

贪吃蛇的食物会随机出现在背景地图上的任意位置，没有任何规律，这就要求程序生成一对随机数值，来控食物所在的行和列。那么，随机数值是如何产生的呢？请大家猛击《C 语言获取随机数》了解详情。

二. 输出贪吃蛇背景地图

贪吃蛇背景地图的最终效果如下图所示：



钻红色空心方框表示边框，绿色实心方框表示贪吃蛇的活动区域。实现代码如下：

```
#include <stdio.h>
#include <conio.h>
#include <windows.h>
int main(){
    int width = 30, height = width;  //宽度和高度
    int x, y;  //x、y 分别表示当前行和列
    HANDLE hConsole = GetStdHandle(STD_OUTPUT_HANDLE);

    //设置窗口大小
    system("mode con: cols=64 lines=32");

    //打印背景，按行输出
    for(x=0; x<width; x++){
        for(y=0; y<height; y++){
            if(y==0 || y==width-1 || x==0 || x==height-1){  //输出边框
                SetConsoleTextAttribute(hConsole, 4 );
                printf("□");
            }else{  //贪吃蛇活动区域
                SetConsoleTextAttribute(hConsole, 2 );
                printf("■");
            }
```

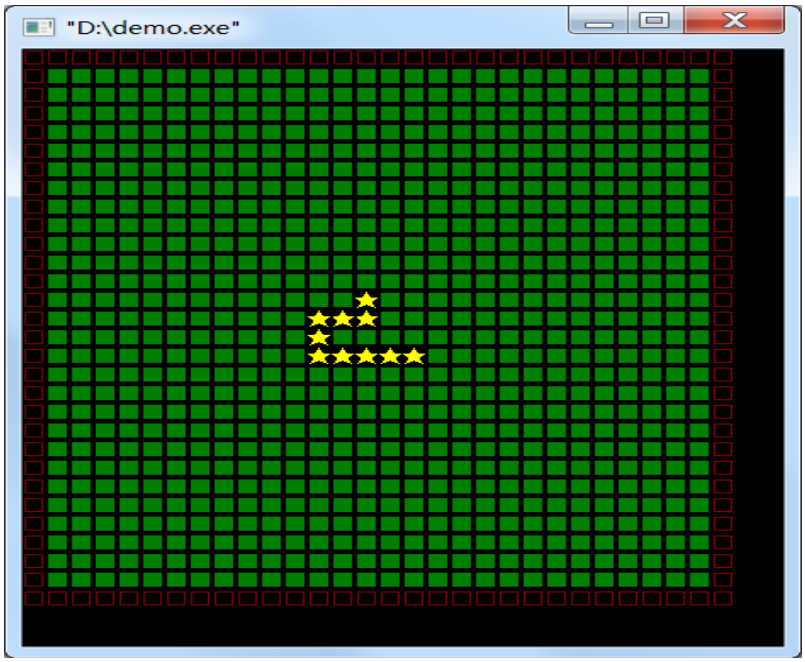
```
        }
        printf("\n");
    }

    //暂停
    getch();
    return 0;
}
```

程序的关键是两层嵌套的循环。x=0 时，内层循环执行 30 次，输出第 0 行；x=1 时，内层循环又执行 30 次，输出 1 行。以此类推，直到 x=30，外层循环不再执行（内存循环当然也就没机会执行），输出结束。

注意，□和■虽然都是单个字符，但它们不在 ASCII 码范围内，是宽字符，占用两个字节，用 putchar 等输出 ASCII 码（一个字节）的函数输出时可能会出现问题，所以作为字符串输出。

三. 让贪吃蛇移动起来

接下来，我们来让一条长度为 n 的贪吃蛇移动起来，而且可以用 WASD 四个键控制移动方向，如下图所示：



其实，移动贪吃蛇并不需要移动所有节点，只需要添加蛇头、删除蛇尾，就会有动态效果，这样会大大提高程序的效率。

我们可以定义一个结构体来表示贪吃蛇一个节点在控制台上的位置（也即所在行和列）：

```
struct POS{
    int x;  //所在行
    int y;  //所在列
}
```

然后再定义一个比贪吃蛇长的数组来保存贪吃蛇的所有节点：

```
struct POS snakes[n+m];
```

并设置两个变量 headerIndex、tailIndex，分别用来表示蛇头、蛇尾在数组中的下标坐标，这样每次添加蛇头、删除蛇尾时只需要改变两个变量的值就可以。如下图所示：

headerIndex 和 tailIndex 都向前移动，也就是每次减 1。如果 headerIndex=0，也就是指向数组的头部，那么下次移动时 headerIndex = arrayLength - 1，也就是指向数组的尾部，就这样一圈一圈地循环，tailIndex 也是如此。这相当于把数组首尾相连成一个圆圈，贪吃蛇在这个圆圈中不停地转圈。

由于这部分的演示代码较长，请大家到百度网盘下载：http://pan.baidu.com/s/1bouZGoZ　提取密码：4g74

对代码的说明

1) 贪吃蛇的最大长度为绿色方框的个数，所以我们将容纳贪吃蛇的数组 snakes 的长度定义为(HEIGHT-2) * (WIDTH-2)。

2) □、■、★ 占用两个字符的宽度，所以在 setPosition() 中该变光标位置时，光标的 X 坐标应该是：

```
coord.X = 2*y;
```

四. 随机生成食物

食物的生成是贪吃蛇游戏的难点，因为食物只能在绿色背景(■)部分生成，它不能占用钻红色边框(□)和贪吃蛇本身(★)的位置。

最容易想到的思路是：随机生成一个坐标，然后检测该坐标是不是绿色背景，如果是，那么成功生成，如果不是，继续生成随机数，继续检测。幸运的话，可以一次生成；不幸的话，可能要循环好几次甚至上百次才能生成，这样带来的后果就是程序卡死一段时间，贪吃蛇不能移动。

这种方案的优点就是思路简单，容易实现，缺点就是贪吃蛇移动不流畅，经常会卡顿。

改进的方案

最好的方案是生成的随机数一定会在绿色背景的范围内，这样一次就能成功生成食物。该如何实现呢？

这里我们提供了一种看起来不容易理解却行之有效的方案。

我们不妨将贪吃蛇的活动范围称为“贪吃蛇地图”，而加上边框就称为“全局地图”。首先定义一个二维的结构体数组，用来保存所有的点（也即全局地图）：

```
1.  struct{
2.      char type;
```

```
    int index;
}globalMap[MAXWIDTH][MAXHEIGHT];
```

MAXWIDTH 为宽度，也即列数；MAXHEIGHT 为高度，也即行数。成员 type 表示点的类型，它可以是食物、绿色背景、边框和贪吃蛇节点。

直观上讲，应该将 type 定义为 int 类型，不过 int 占用四个字节，而节点类型的取值范围非常有限，一个字节就足够了，所以为了节省内存才定义为 char 类型。

然后再定义一个一维的结构体数组，用来保存贪吃蛇的有效活动范围：

```
struct{
    int x;
    int y;
} snakeMap[ (MAXWIDTH-2)*(MAXHEIGHT-2) ];
```

x、y 表示行和列，也就是 globalMap 数组的两个下标。globalMap 数组中的 index 成员就是 snakeMap 数组的下标。

globalMap 表示了所有节点的信息，而 snakeMap 只表示了贪吃蛇的活动区域。通过 snakeMap 可以定位 globalMap 中的元素，反过来通过 globalMap 也可以找到 snakeMap 中的元素。它们之间的对应关系请看下图：

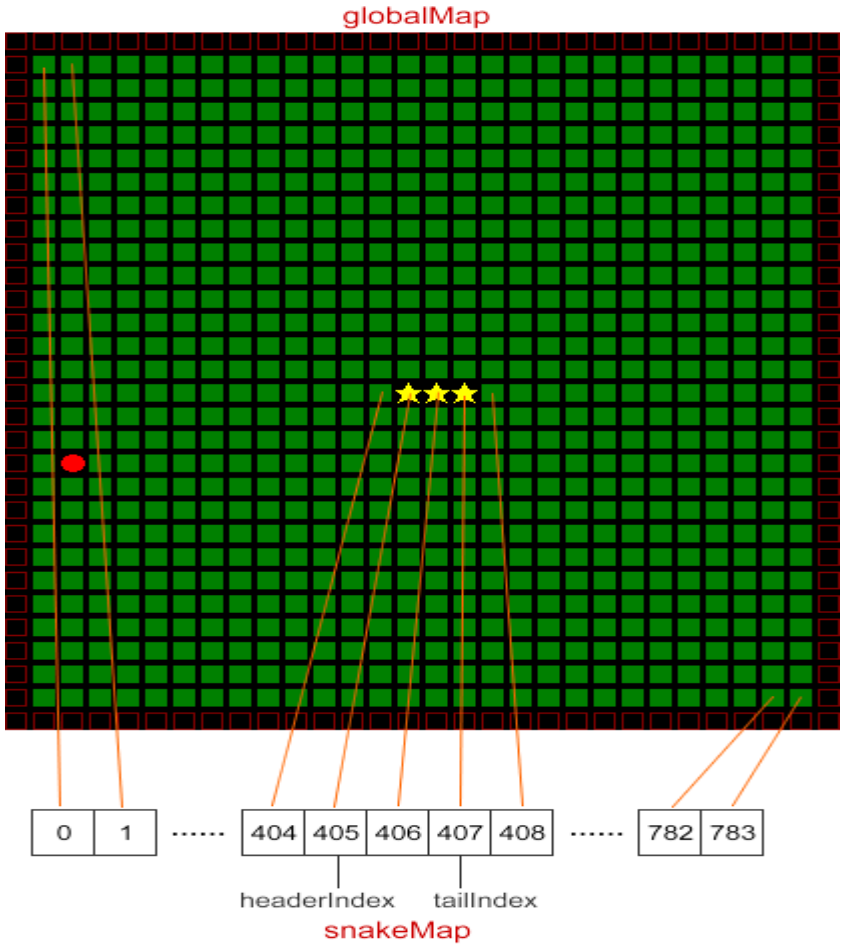


图 1：globalMap 和 snakeMap 的初始对应关系

贪吃蛇向左移动时，headerIndex 指向 404，tailIndex 指向 406。

在 snakeMap 数组中，贪吃蛇占用一部分元素，剩下的元素都是绿色的背景，可以随机选取这些元素中的一个作为食物，然后通过 x、y 确定食物的坐标。而这个坐标，一定在绿色背景范围内。

需要注意的是，在贪吃蛇移动过程中需要维护 globalMap 和 snakeMap 的对应关系。

这种方案的另外一个优点就是，贪吃蛇移动时很容易知道下一个节点的类型，不用遍历数组就可以知道是否与自身相撞。

2. C 语言 2048 小游戏源码下载和思路解析

在《C 语言 2048 小游戏演示和说明》一节中，我们对 2048 游戏进行和简单的介绍和演示，这节就来分析一下它的源码。

2048 游戏的源码下载地址：

- 百度网盘：https://pan.baidu.com/s/1eTqoznk 密码：czns

各位读者不防先将源码下载下来浏览一遍，记住关键的几个函数，整理一下不了解的知识点，做到心中有数。

关键知识点

下面请各位读者先学习一下该游戏中涉及到的几个关键知识点，有了这些必备条件，我们才好讲解 2048 游戏的设计思路。

1) 改变文本颜色

2048 游戏设置的格子边框颜色为湖蓝色，其中的数字颜色设有淡绿色、绿色等，所有的这些设置都涉及到如何改变文本的输出颜色，请大家猛击《彩色版的 C 语言，让文字更漂亮》了解详情。

2) 键盘监听

在 2048 游戏中，玩家可通过 W\A\S\D 或者方向键来控制数字的移动方向，具体实现方法请大家猛击《C 语言非阻塞式键盘监听》了解详情。

3) 获取随机数

2048 游戏中，对于每次产生的新的数字所处位置，是随机的，这就需要通过产生随机数来完成，请大家猛击《C 语言获取随机数》了解详情。

2048 游戏整体设计思路

2048 小游戏的整体的设计思路是：

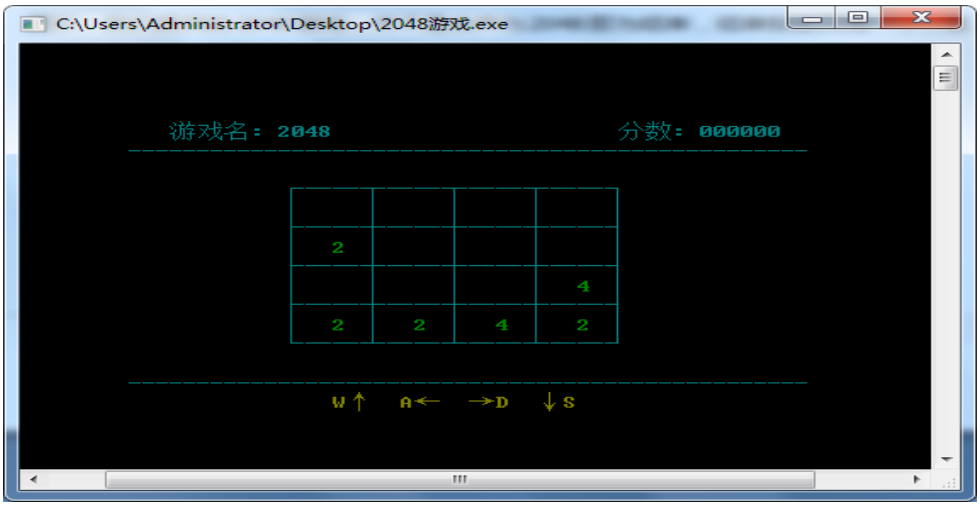
1. 游戏界面初始化，共有 4 行 4 列，总计 16 个位置，游戏开始时，在任意的两个位置上，随机产生数字 2 或 4（产生 2 的几率更大一些）；
2. 玩家可通过 W\A\S\D 或者键盘方向键来控制所有数字的移动，游戏过程中，要符合 2048 游戏的基本规则；
3. 当游戏中无空余位置，且相邻数字之间无法合并，则 game over；

提示：我们提供的 2048 游戏，并不是合并为 2048 即为结束，结束标志只有一个，就是第 3 条所说，直到游戏无法进行，才会结束。

数字移动和合并的算法实现

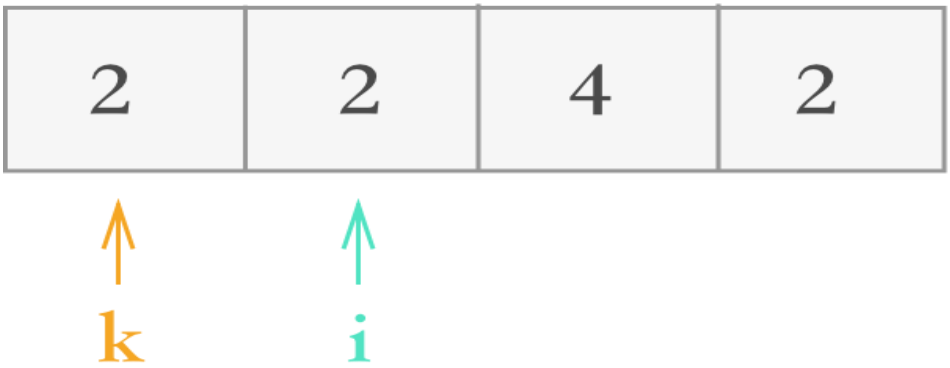
在 2048 游戏中，数字移动和合并为游戏的核心，在游戏过程中，无论数字向那个方向移动，其实现所用的算法都是相同的。

这里我们用 "左移" 操作来给大家讲解算法的实现过程。



如上图所示，若这种情况下玩家进行“左移”操作，我们首先拿最后一行来具体分析源码中的实现过程（左移执行 move_left（ ）函数）。

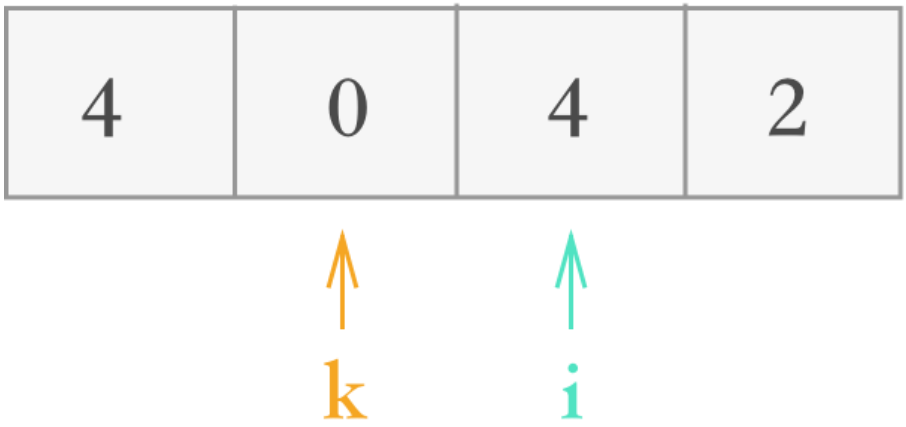
首先，我们用变量 k 表示坐标为 (4,1) 位置存放的数据，然后用变量 i 从（4,2）开始遍历，到有数字的位置就停止，一直到此行的最后位置（4,4），如下图所示：



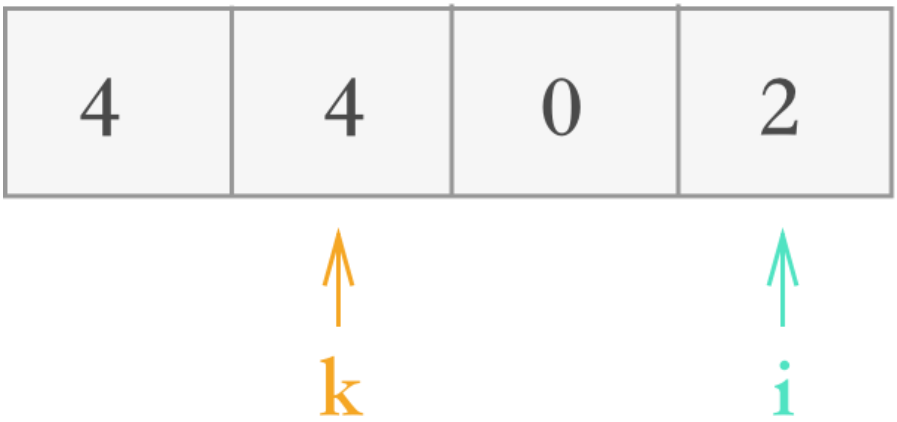
在 i 移动的过程中，我们统一做以下操作：

- 如果 k 指向位置的数字同 i 指向位置的数字相等：则消除 i 位置的数字，k 指向位置的数字*2，然后 k 本身 +1（即 k 指向下一个位置），i 向后移动；
- 如果 k 指向位置的数字同 i 指向位置的数字不相等：则将 i 位置的数字移动到 k+1 的位置，同时 k 本身要 +1（即 k 指向下一个位置），i 向后移动；
- 如果 k 指向位置没有数字：则将 i 位置的数字直接移动到 k 所指向的位置，同时消除 i 位置处的数字。k 位置不动，i 向后移动；

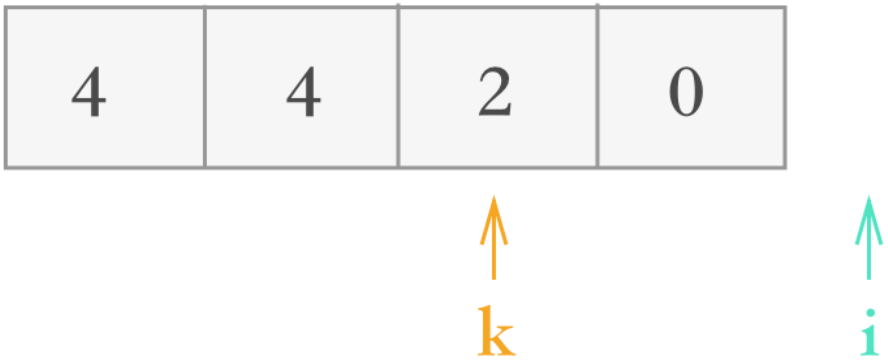
采用以上规则，第一次运行，k=2，同时 i=2，符合第 1 条，做相应改变之后，如下图所示：



继续运行，此时 k 位置无数字，i=4，符合第 3 条，改变之后，如下图所示：



继续运行，此时 k=4，i=2，符合第 2 条，做相应改变后，如下图所示：



此时，由于 i 超出了指定的遍历区间，所以此次移动任务结束。以上是第 4 行移动的过程。

同理可见，第 2 行中，i 可从坐标（2,2）位置向后移动，但是一直没有数字，所以不需任何移动；

第 3 行中，第一次运行，k 处无数字， i 指向的是最后的数字 4，按照第 3 条规则，直接将 4 移动到 k 处，此行移动就结束了。

所以，在实现左移操作的实现时，我们只需要遍历每一行，在每行的遍历过程中，嵌套遍历每一个非 0 数，根据以上规则，做相应移动即可。

举一反三，上移、下移、右移，都是如此实现。具体实现过程，可见源代码中 move_left()、move_right()、move_up()、move_down() 函数的代码实现。

游戏结束的判断标志

我们设计的 2048 游戏，由于其并不是产生 2048 即为结束标志，所以对于此游戏来说，玩家甚至可合并处 4096 、8192、16384 等。

游戏一旦开始，唯一的正常结束标志就是：16 个格子全部被数字占用，且无论做哪种方向的移动，数字之间都无法完成合并，此时，游戏无法继续，视为结束的标志。

所以，在游戏的运行过程中，我们只需要判断以下两种状态：

- 16 个格子是否全部沾满，如果有空格子，则游戏继续；
- 如果 16 个格子全部沾满，可使用嵌套循环判断：是否存在相邻数字之间相等，如果存在，游戏继续；如果不存在，游戏结束。

具体游戏结束的算法实现，可参考源代码的 refresh_show() 函数中，对游戏结束的判断。

至于该游戏中界面的搭建，实则是使用特殊字符组成，即以 printf() 输出特殊字符的形式组成，具体实现可见 refresh_show() 函数，并无什么技术含量，这里不再详细介绍，大家可以参照源码。

3. C 语言推箱子小游戏源码下载和思路解析

在《C 语言推箱子游戏演示和说明》一节中，我们对推箱子游戏的玩法进行了介绍和演示，这节就来分析一下它的源码。

推箱子源代码百度网盘下载地址：https://pan.baidu.com/s/1jJRZEQE 密码：fpyy

各位读者不妨先将源码下载下来浏览一遍，记住关键的几个函数，整理一下不了解的知识点，做到心中有数。

关键知识点

下面请各位读者先学习一下该游戏中涉及到的几个关键知识点，有了这些必备条件，我们才好讲解贪吃蛇的设计思路。

1) 改变输出文本的颜色

推箱子游戏中的边框是灰色的，小人是湖蓝色的，箱子是淡红色的，星星是黄色的，等等这些涉及到如何改变文本的输出颜色，请大家猛击《彩色版的 C 语言，让文字更漂亮》了解详情。

2) 键盘监听

推箱子游戏中，人在移动过程中，必须能够及时捕获用户按下的方向键，并改变移动方向，具体是如何做到的，请大家猛击《C 语言非阻塞式键盘监听》了解详情。

设置游戏地图

推箱子游戏的地图，由于是二维结构，所以采用二维数组进行存储，再合适不过。

具体方案是采用整形数组设计地图，地图中不同的道具，用不同的数字表示：

- 箱子所在的位置，用数字 2 表示；
- 人所在的位置，用数字 3 表示；
- 边框的位置，用数字 1 表示；
- 箱子最终需到达的位置，用数字 4 表示；
- 地图中可行的道路，可用数字 0 表示；

如下的 map_1 这个二维数组，记录的就是第一关卡地图所对应的初始数据：

```
int map_1[10][10] = {
{ 0,0,1,1,1,0,0,0 },
{ 0,0,1,4,1,0,0,0 },
{ 0,0,1,0,1,1,1,1 },
{ 1,1,1,0,0,2,4,1 },
```

```
    { 1, 4, 2, 2, 0, 1, 1, 1 },
    { 1, 1, 1, 3, 2, 1, 0, 0 },
    { 0, 0, 0, 1, 4, 1, 0, 0 },
    { 0, 0, 0, 1, 1, 1, 0, 0 }
};
```

地图中的道具分别用不同的数字表示完成后，当输出到窗口上时，可对二维数组进行加工，即用适当的字符代替不同的数字：

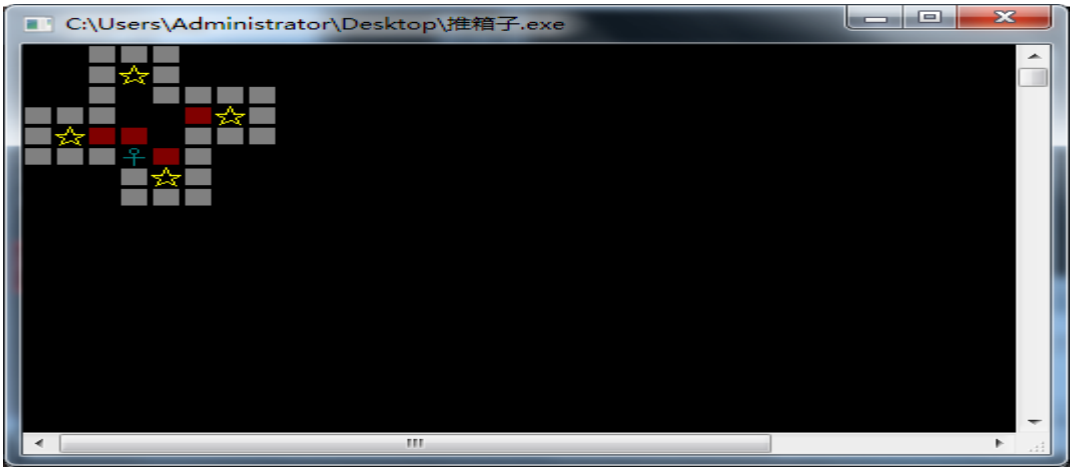
- 1 表示边框，可用 “■” 代替；
- 3 表示人物，可用 ”♀” 代替；
- 2 表示箱子，可用 “■” 代替；
- 4 表示箱子要去的目标位置，可用 ” ☆” 代替；
- 0 表示道路，可用 “空格” 代替；

需要说明的是：所有的字符都不是普通字符，都为特殊字符，每个特殊字符占用 2 个普通字符的位置（一个特殊“空格”可用两个普通空格来表示）。
除了以上几种情况外，在游戏过程中，还有可能出现以下几种情况，仍需要使用不同的数字表示：

1. 箱子到达目标位置 ” ☆” 时，需要用不同的标志显示出来，证明箱子已经达到目标位置，程序中用“★”表示，对应的数字为 5；
2. 人物到达有” ☆” 的位置时，此时需要特殊标记，程序中用数字 6 表示这个特殊位置（图标没变）。人物移动后，此处再恢复成 ” ☆” ；

综上所述，实现代码可见分享给大家的源码中的 render() 函数。

代码中，设定地图的最大规格为 10*10，在显示时，只需遍历整个二维数组，不同的数字采用不同的字符表示，同时赋予每个字符不同的颜色（color() 函数的作用就是赋予字符颜色），最终的效果如图所示：



人物和箱子的移动

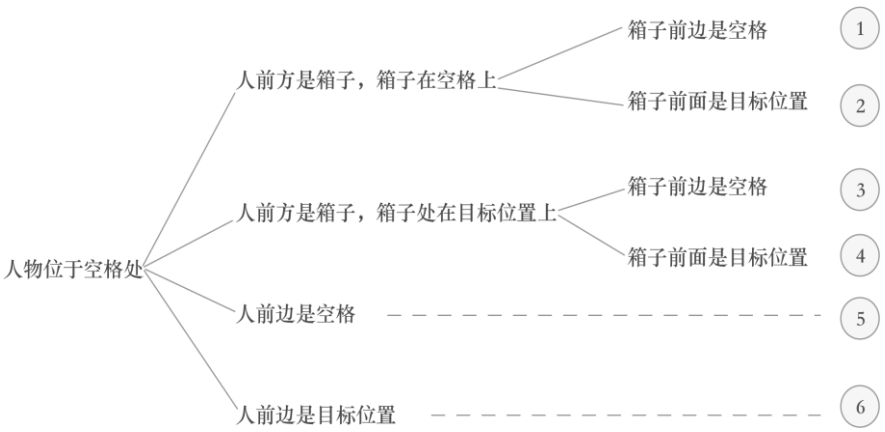
整个游戏过程中，玩家时刻都是在控制人物的移动，所以，程序中应为人物为出发点，采用穷举法罗列出所有由于人物移动而导致的情况。

另外，人物在移动过程中，无论人物是向哪个方向移动，程序中都可归为一种情况，那就是：人物在向前移动。

当人物向前移动之前，首先根据人物当前所在的位置不同，分为两种情况：

1. 当前人物位于道路上（即空格上），则当人物移动后，此位置应恢复为空格；
2. 当前人物位于箱子要去的目标位置上，则当人物移动后，此位置要恢复为目标位置，而不是空格；

如下图所示，即使是在第 1 种情况下，由于人物前方道路状况不同，处理方案也不一样：



- 第 1 种情况处理方案：人物移动后，原位置恢复为空格（即数字 0），人物移动后位于空格上，所以该位置对应的数字为 3，箱子也向前一步，此时箱子位于空格上，所有该位置对应的数字为 2。
- 第 2 种情况处理方案：人物移动后，原位置恢复为空格，人物站在空格上，该位置对应的数字为 3，箱子向前一步，恰好对于目标位置处，对应的数字为 5。
- 第 3 种情况处理方案：人物移动后，原位置恢复为空格，人物位于箱子的目标位置上，此位置对应的数字为 6，箱子向前一步，位于空格上，该位置对应的数字为 2。
- 第 4 种情况处理方案：人物移动后，原位置恢复为空格，人物位于箱子的目标位置上，此位置对应的数字为 6，箱子向前一步，还是位于目标位置处，对应的数字还是 5。
- 第 5 种情况处理方案：人物移动后，原位置恢复为空格，移动后的人物位于空格处，该位置对应的数字为 3。
- 第 6 种情况处理方案：人物移动后，原位置恢复为空格，移动后的人物位于箱子的目标位置上，此位置对应的数字为 6。

以上仅是第一种情况下，面临的 6 种可能。当人物位于箱子的目标位置时，也同样面临以上 6 种情况。每种情况的处理方案中，同第一种情况不同的是，人物移动后，原位置应恢复为箱子的目标位置，对应的数字为 4。

以上 12 种情况，及各自的处理方案的具体实现，可见源码中的 move() 函数，其中附带有详细的代码注释。

4. C 语言扫雷小游戏源码下载和思路解析

在《C 语言扫雷游戏演示和说明》一节中，我们对扫雷游戏的玩法进行了介绍和演示，这节就来分析一下它的源代码。

扫雷游戏源代码百度网盘下载地址：https://pan.baidu.com/s/1sncBcnz 密码: wcfg

各位读者不防先将源代码下载浏览一遍，记住关键的几个函数，整理一下不了解的知识点，做到心中有数。

一、关键知识点

下面请各位读者先学习一下该游戏中涉及到的几个关键知识点，有了这些必备条件，我们才好讲解扫雷的设计思路。

1) 改变输出文本的颜色

扫雷游戏的整个界面，都是由特殊字符组成，小方块 ■ 是灰色的，方框 □ 是白色的，炸弹 ● 是黄色的，等等。这就涉及到如何改变文本的输出颜色，请大家猛击《彩色版的 C 语言，让文字更漂亮》了解详情。

2) 键盘监听

在扫雷游戏中，当用户输入是否开始游戏的 Y/N 时，程序能够自动监听，当用户输入完成后，不用回车，程序立即做出反应，这就用到了键盘监听。请大家猛击《C 语言非阻塞式键盘监听》了解详情。

3) 获取随机数

扫雷游戏中，雷区会随机出现在整个区域的任意位置，没有任何规律，这就要求程序能够生成随机数值，由随机数设置雷区的位置。有关随机数的产生请大家猛击《C 语言获取随机数》了解详情。

二、扫雷游戏初始化

扫雷游戏初始化过程中，我们使用到了 3 个二维数组：mine、show、mineDow：

- mine 数组用于初始化扫雷游戏。具体做法是：首先默认整个二维数组中没有雷区，全部设为安全区域（用 0 表示），然后在二维数组中随机安插一定数量的雷区；
- show 数组用于每次将结果输出给用户。当用户输入完成后，show 数组会根据用户的输入对存储的数据做适当的更新，然后输出给用户；
- mineDow 数组中每个数据表示的，该位置相邻的周围雷区的数量，此数组的建立为的是实现“点击一个位置，开出一片安全区域”的效果

提示：在源代码的 game 函数中，在 mine_sweep() 函数之前的所有工作，都是初始化工作。

三、扫雷功能的实现

扫雷功能的实现，整体思路是：

1. 判断用户输入坐标处是否是雷区；
2. 更新 show 数组：如果是雷区，将 show 数组中该坐标位置上由字符 ‘*’ 改为表示雷区的字符 ‘o’ ；如果不是雷区，借助 mineDow 数组中存储的信息，使用递归的方式，找到其他符合条件的非雷区区域（show_deal 函数的作用），将找到的所有区域一并更新到 show 数组中；
3. 将新的 show 数组以一定的格式输出，反馈给用户；

提示：show_deal 函数找的是该区域既不是雷区，其周围也没有雷区的区域，将其全部更新到 show 数组中。一旦遇到周围有雷区的区域（这部分区域也会被更新到 show 数组中），则递归结束。

扫雷功能的具体实现，可见源代码中的 mine_sweep 函数。

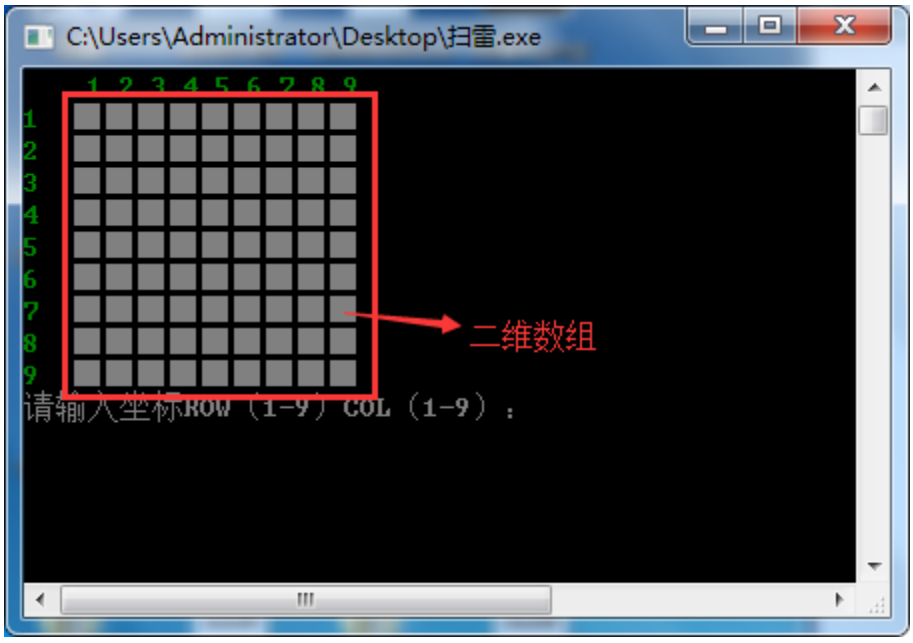
四 扫雷界面的优化

扫雷界面，实际上是将 show 数组中存储的数据换了一种方式输出出来。

例如，show 数组中存储有字符 ‘*’ ，在输出时，统一换为“■”；表示雷区的字符 ‘o’ ，统一换为 ‘●’ ，等等。

注意：用于替换的字符，并不是普通的字符，它们并不在 ASCII 码范围内，是宽字符，占用两个字节。

采用此种方式，再配以合适的颜色（采用 Windows API），可以将二维数组以如下的这种形式反馈给用户：



扫雷界面的详细优化代码，可见原代码中的 display_board() 函数的实现。

5. C 语言学生信息管理系统源码下载和思路解析（文件版）

在《C 语言学生信息管理系统演示和说明（文件版）》一节中，我们对学生信息管理系统进行了介绍和演示，这节就来分析一下它的源码。

学生信息管理系统源码下载地址：http://pan.baidu.com/s/1hqZRiDY　密码：ty9f

各位读者不妨先将源码下载下来浏览一遍，记住关键的几个函数，整理一下不了解的知识点，做到心中有数。

一. 整体设计思路

要想持久化地保存数据，必须要将数据写入磁盘中；本程序也不例外，我们会将学生信息最终都保存到文件中。当增加、删除、修改学生信息时，我们也应该对文件做出同样的操作。这就是本节的重点：

- 如何检索文件；
- 如何在文件中插入数据；
- 如何删除文件中的部分数据；
- 如何修改文件中的数据。

二. 关键知识点

大家需要先学习一下该程序中涉及到的几个关键知识点，有了这些必备条件，我们才能更容易理解代码。

1) 模块化编程

本程序的代码比较多，总共有 700 多行，需要分门别类、有规划地放到不同的源文件中，这就是所谓的模块化编程（也即多文件编程）。在模块化编程中，需要在 .c 文件中定义函数，在 .h 中声明函数、变量、自定义类型、结构体、宏等，请大家猛击《C 语言多文件编程》一章了解详情。

2) 文件操作

我们的程序将学生信息保存到文件中，并根据学号进行排序，这样在查询和定位时就比较方便。那么，问题来了，如何在文件的中间插入、删除、修改数据呢？如何在文件中定位某个学生的信息呢？请大家阅读《C 语言文件操作》一章学习。

常见的文件大都是顺序文件，也就是文件内容是依次存储在硬盘上的。顺序文件检索速度快，但是不利于数据的插入、删除和修改。例如，在文件中间插入数据时，理论上要将后面的数据整体后移，但是这会带来风险，很有可能覆盖后面的数据，导致其他文件出错，所以这种方式是绝对禁止的。

大家在阅读教程时尤其要注意对顺序文件的插入、删除、修改操作是如何实现的，这是程序得以实现的关键。

3) 循环菜单

程序运行时，会不停地显示主菜单和子菜单，而不是执行完一次操作就退出，这是如何实现的呢？请大家猛击《C 语言循环菜单的设计，让程序一直运行》了解详情。

三. 程序的整体架构

程序由 6 个文件构成，其中包括 3 个头文件（.h）和 3 个源文件（.c）。

1) main.c 是主文件，包含了主函数 main() 以及两个打印菜单的函数 printMainMenu()、printSubMain()。

2) common.h 是程序的配置文件，每个文件都应该将它包含进去。配置文件中主要是宏定义，每一个宏都是一个配置选项，用户可以更改。例如：

- FILENAME 宏定义了数据文件的路径，也就是将学生信息保存到何处，默认是当前目录下的 stu.data。如果你希望将文件放在其他目录下，完全可以改成诸如 D:\\Demo\stu.data、C:\\data.stu 的形式。
- MAX_STU_AGE 宏定义了学生的最大年龄，如果用户输入的年龄大于该值，就会给出提示。

3) tools.c 和 tools.h 主要提供了工具类函数。所谓工具类函数，也就是通用函数，它们不针对具体程序编写，可以用在当前程序中，也可以用在其他程序中，在移植的过程中一般不需要修改代码。

4) stu.c 和 stu.h 是主要的两个文件，包含了对学生信息进行增删改查的函数。

四. 索引的建立

为了方便检索，我们将学生信息按照学号从小到大依次保存到文件中。但即便这样，直接检索文件数据也是非常低效的，所以我们将所有学生的学号保存到一个 int 数组中，这样：
1) 新增学生信息时，如果学号和数组中某个元素的值相等，就说明该学生存在，不能插入。

2) 新增、删除、修改、查询学生信息时，可以很容易地获取当前学号在数组中的下标，进而计算出该学生信息在文件中的位置。例如，学号 10 在数组中的下标为 5，现在想删除该学生信息，那么将文件内部的位置指针调整到 5*sizeof(STU) 处就可以（STU 是保存学生信息的结构体）。

这个数组和文件中学生信息的位置直接关联，我们不妨将它称为索引。需要注意的是，每次插入、删除学生信息时都要更新索引，这就是 updateIndex() 函数的作用。

总结起来，索引的作用是更高效地检索和定位学生信息。

5. C 语言学生信息管理系统源码下载和思路解析（数据结构版）

在《C 语言学生信息管理系统演示和说明（数据结构版）》一节中，我们对学生信息管理系统进行了介绍和演示，这节就来分析一下它的源码。

学生信息管理系统源码下载地址：http://pan.baidu.com/s/1kUsIcuz　提取密码：fpd3

各位读者不妨先将源码下载下来浏览一遍，记住关键的几个函数，整理一下不了解的知识点，做到心中有数。

一. 整体设计思路

要想持久化地保存数据，必须要将数据写入磁盘中；本程序也不例外，我们会将学生信息最终都保存到文件中。

在《C 语言学生信息管理系统源码下载和思路解析（文件版）》一节中我们看到，直接对文件数据进行增删改查操作不仅麻烦，而且低效，所以我们引入了数据结构中的单链表。

当对学生信息进行插入、删除、修改操作时，我们需要先更新单链表，再将单链表中的所有数据（包括未修改的数据）更新到文件。如此，对文件进行的就是整体操作，就简单很多，可以先将文件内容清除，再写入新的数据。

需要注意的是：只能将节点中的数据写入文件，不能将 next 节点的指针写入。例如：

```
typedef struct _NODE{
    STU data;
    struct _NODE *next;
}NODE;
```

只需要将成员 data 写入文件，而不能写入 next。

这是因为，next 是指针，指向内存中的某个位置，程序重新启动时，它的内存布局和上次肯定不同，next 的指向也会发生改变，所以每次启动程序时都必须读取文件中的数据新建单链表，存储 next 是没有意义的。

总结：对单链表的增删改查是本程序的重点。

二. 关键知识点

大家需要先学习一下该程序中涉及到的几个关键知识点，有了这些必备条件，我们才能更容易理解代码。

1) 模块化编程

本程序的代码比较多，总共有 670 多行，需要分门别类、有规划地放到不同的源文件中，这就是所谓的模块化编程（也即多文件编程）。在模块化编程中，需要在 .c 文件中定义函数，在 .h 中声明函数、变量、自定义类型、结构体、宏等，请大家猛击《C 语言多文件编程》一章了解详情。

2) 循环菜单

程序运行时，会不停地显示主菜单和子菜单，而不是执行完一次操作就退出，这是如何实现的呢？请大家猛击《C 语言循环菜单的设计，让程序一直运行》了解详情。

3) 单链表

大家在阅读代码之前，需要对数据结构中的单链表有所了解，知道如何对节点信息进行增删改查。

三. 程序的整体架构

程序由 6 个文件构成，其中包括 3 个头文件（.h）和 3 个源文件（.c）。

1) main.c 是主文件，包含了主函数 main() 以及两个打印菜单的函数 printMainMenu()、printSubMain()。

2) common.h 是程序的配置文件，每个文件都应该将它包含进去。配置文件中主要是宏定义，每一个宏都是一个配置选项，用户可以更改。例如：

- FILENAME 宏定义了数据文件的路径，也就是将学生信息保存到何处，默认是当前目录下的 stu.data。如果你希望将文件放在其他目录下，完全可以改成诸如 `D:\\Demo\stu.data`、`C:\\data.stu` 的形式。
- MAX_STU_AGE 宏定义了学生的最大年龄，如果用户输入的年龄大于该值，就会给出提示。

3) tools.c 和 tools.h 主要提供了工具类函数。所谓工具类函数，也就是通用函数，它们不针对具体程序编写，可以用在当前程序中，也可以用在其他程序中，在移植的过程中一般不需要修改代码。

4) stu.c 和 stu.h 是主要的两个文件，包含了对学生信息进行增删改查的函数。

7. C 语言学生信息管理系统源码下载和思路解析（密码版）

在《C 语言学生信息管理系统演示和说明（密码版）》一节中，我们对学生信息管理系统进行了介绍和演示，这节就来分析一下它的源码。

学生信息管理系统源码下载地址：http://pan.baidu.com/s/1pKjMVij　提取密码：s2ki

各位读者不妨先将源码下载下来浏览一遍，记住关键的几个函数，整理一下不了解的知识点，做到心中有数。

一. 整体设计思路

要想持久化地保存数据，必须要将数据写入磁盘中；本程序也不例外，我们会将学生信息最终都保存到文件中。

在《C 语言学生信息管理系统源码下载和思路解析（文件版）》一节中我们看到，直接对文件数据进行增删改查操作不仅麻烦，而且低效，所以我们引入了数据结构中的单链表。

当对学生信息进行插入、删除、修改操作时，我们需要先更新单链表，再将单链表中的所有数据（包括未修改的数据）更新到文件。如此，对文件进行的就是整体操作，就简单很多，可以先将文件内容清除，再写入新的数据。

需要注意的是：只能将节点中的数据写入文件，不能将 next 节点的指针写入。例如：

```
typedef struct _NODE{
    STU data;
    struct _NODE *next;
}NODE;
```

只需要将成员 data 写入文件，而不能写入 next。

这是因为，next 是指针，指向内存中的某个位置，程序重新启动时，它的内存布局和上次肯定不同，next 的指向也会发生改变，所以每次启动程序时都必须读取文件中的数据新建单链表，存储 next 是没有意义的。

注意：数据在写入文件前要先进行加密，读取后也要先进行解密才能使用。

二. 关键知识点

大家需要先学习一下该程序中涉及到的几个关键知识点，有了这些必备条件，我们才能更容易理解代码。

1) 模块化编程

本程序的代码比较多，总共有 670 多行，需要分门别类、有规划地放到不同的源文件中，这就是所谓的模块化编程（也即多文件编程）。在模块化编程中，需要在 .c 文件中定义函数，在 .h 中声明函数、变量、自定义类型、结构体、宏等，请大家猛击《C 语言多文件编程》一章了解详情。

2) 循环菜单

程序运行时，会不停地显示主菜单和子菜单，而不是执行完一次操作就退出，这是如何实现的呢？请大家猛击《C 语言循环菜单的设计，让程序一直运行》了解详情。

3) 单链表

大家在阅读代码之前，需要对数据结构中的单链表有所了解，知道如何对节点信息进行增删改查。

4) 数据的加密解密

数据的加密方式有很多，这里我们使用一种相对简单的方式，就是异或运算，这在《C 语言文件加密的原理--异或运算》一节中进行了详细讲解。

5) MD5

用于对数据加密解密的密码也得保存在文件中，否则下次使用时就无法判断用户输入的密码是否正确。但是，直接将密码保存到文件中是非常危险的，这无疑是将密码告诉了他人。为了避免这种情况，我们还需要对密码本身进行加密。

用于对密码加密的算法有多种，MD5 算法是常用的一种。本程序也使用 MD5 算法。

不管密码有多长，包含什么字符，经过 MD5 算法处理后都会变成一个由数字和字母组成的字符串，并且这个字符串的长度始终是 32。例如，123456 经 MD5 加密后的结果是：

```
e10adc3949ba59abbe56e057f20f883e
```

MD5 算法具有唯一性和不可逆性，也就是说：

- 同一个密码加密后的字符串始终是相同的；
- 通过加密后的字符串不能逆向推算出原来的密码。

对密码进行 MD5 运算后，将得到的长度为 32 的字符串保存到文件中就非常安全了，即使被他人看到，也不知道密码是什么。

三. 程序的整体架构

程序由 8 个文件构成，其中包括 4 个头文件（.h）和 4 个源文件（.c）。

1) main.c 是主文件，包含了主函数 main() 以及两个打印菜单的函数 printMainMenu()、printSubMain()。

2) common.h 是程序的配置文件，每个文件都应该将它包含进去。配置文件中主要是宏定义，每一个宏都是一个配置选项，用户可以更改。例如：

- FILENAME 宏定义了数据文件的路径，也就是将学生信息保存到何处，默认是当前目录下的 stu.data。如果你希望将文件放在其他目录下，完全可以改成诸如 `D:\\Demo\stu.data`、`C:\\data.stu` 的形式。
- MAX_STU_AGE 宏定义了学生的最大年龄，如果用户输入的年龄大于该值，就会给出提示。

3) tools.c 和 tools.h 主要提供了工具类函数。所谓工具类函数，也就是通用函数，它们不针对具体程序编写，可以用在当前程序中，也可以用在其他程序中，在移植的过程中一般不需要修改代码。

4) stu.c 和 stu.h 是主要的两个文件，包含了对学生信息进行增删改查的函数。

5) md5.c 和 md5.h 是 MD5 算法的源文件和头文件。大家不必理解 MD5 算法是如何实现的，阅读源代码时可以将这两个文件跳过，只要知道 MD5() 函数如何使用就可以。下面是 MD5()函数的原型：

```
void MD5(unsigned char *original, unsigned char cipher[33]);
```

original 是要加密的数据，cipher 是加密后的数据。cipher 的长度之所以为 33，而不是 32，是因为在最后要添加字符串结束标志 \0。

四. 密码的保存以及数据的加密

本程序中，我们将 MD5 加密后的密码保存在文件的开头，占用 32 个字节。也就是说，从第 33 个字节才开始，存储的才是学生信息。

不妨将原始密码称为 pwd，将一次 MD5 加密后的密码称为 pwdMD5。我们可以把 pwdMD5 保存在文件开头，并用 pwd 对文件数据（学生信息）进行加密。

程序运行后，读取用户输入的密码 pwd，然后进行 MD5 运算得到 pwdMD5，将 pwdMD5 与文件前 32 字节进行比较就知道用户输入的密码是否正确。如果密码正确，就用 pwd 对学生信息进行解密，否则提示用户密码错误。