**1. (单选题)**

**system()函数在C中使用，需要包含哪个头文件？**

 A. stdio.h

 B. stdlib.h

 C. iostream.h

 D. assert.h

*正确答案:* B

### 2. (单选题)

**system(“pause”)的作用是什么？**

* A. 程序暂停，冻结屏幕;
* B. 操作系统暂停;
* C. 程序结束退出;
* D. 操作系统退出

*正确答案:* A

**3. (单选题)以下哪个指令是错误或无效的？**

* A. **system**("CLS");
* B. **system**("DEL");
* C. **system**("mode con cols=48 lines=25")
* D. **system**("color 0B");

*正确答案:* B

**5. (单选题)**

**Console界面字符输出换行，以下哪个选项不是常见的操作？**

* A. /r
* B. /n
* C. Enter
* D. endl

*正确答案:* A

**6. (判断题)**

**在有语言包的情况下，Console界面可以直接输出中文？**

* A. 对
* B. 错

*正确答案:* 对

### 1. (简答题)

**在游戏开发中，system("CLS")清屏操作，可能会作为什么用途而被使用？**

*正确答案：*

system("CLS")清屏，是对整个游戏窗口进行清除。游戏画面的更新绘制，通常都是先更新背景图，然后绘制其他的游戏元素。为避免疏漏，大多采用先清屏，再更新背景图的操作。所以，需要用到清屏操作。

如果是采用局部更新的游戏画面绘制策略，则不需要清屏，只需对局部区域的内容进行重绘。

### 2. (简答题)用自己的语言描述下什么叫逻辑错误，它与编译错误有什么区别？

*正确答案：*

逻辑错误主要表现在程序运行后,得到的结果与预期设想的不一致,这就有可能是出现了逻辑错误。

    逻辑错误是相对于编译错误来说的，例如程序代码的语法有问题，编译时就会出错。而逻辑错误在语法上是正确的，程序能够编译并运行，但是运行的结果与预想的不同。

编译错误由于会被编译器发现，所以并不会带来严重的后果，相反很多逻辑错误很难发现，从而导致程序出现BUG。

**1. (单选题)**

**如何通过按下键盘的指定按键，对程序的循环进行中断**

* A. 按Esc键
* B. 用kbhit()函数检查键盘输入，并作出中断的逻辑判断；
* C. 耐心等待循环条件结束，避免程序崩溃；
* D. ALT+F4键，强制关闭程序

*正确答案:* B

### 2. (单选题)

**下方哪个函数可以直接侦测键盘的硬件中断，并返回键盘指定按键的down/up状态？**

* A. kbhit()
* B. GetAsyncKeyState( );
* C. GetKeyState( int vKey );
* D. GetKeyboardState( );

*正确答案:* B

**3. (单选题)**

**以SetConsoleTextAttribut()函数对Console窗口的文字与背景颜色进行设置，最多能得到多少种颜色组合？**

* A. 2的6次幂,64种组合;
* B. 2的8次幂，256种组合
* C. 2的4次幂,16种组合;
* D.

2的7次幂，128种组合

*正确答案:* B

### 1. (简答题) 颜色数值0XF0表达的是哪种颜色？

*正确答案：*

白底黑字

### 2. (简答题)

**如何通过指定的按键对循环进行中断？已经被中断的循环如何重新开始循环？**

*正确答案：*

①通过kbhit()等键盘接收函数，接收指定的键盘指令，对循环进行中断跳出；

②在该循环的外部再套一个循环，以确保在满足条件的情况下，被中断的循环能够重新开始

### 1. (简答题)

**游戏的框架由哪几部分构成？**

*正确答案：*

游戏的初始化、过程循环、输入处理、游戏逻辑、图形绘制。

或有相近的表述也可以

### 2. (简答题)游戏循环中“绘制”、“输入”、“逻辑”3个模块执行的先后顺序如果发生调整，会有怎样的影响？

*正确答案：*

“输入”会影响“逻辑”计算，进而影响“绘制”结果。但游戏中游戏循环的周期通常快于玩家的反应速度，例如60FPS，即是1秒针绘制60帧，游戏在0.0167秒内完成一次游戏循环。在游戏循环中，3个模块循环运行，因此对游戏体验的影响通常较小。但对于游戏代码调试则会有影响。比如循环里面的运行顺序是“绘制”、“输入”、“逻辑”，则当前“输入”的结果，需要在下次循环的“绘制”时候才会显现。

### 3. (简答题)

**贪吃蛇的方向表示为什么采用枚举类型？它有什么优点？**

*正确答案：*

 enum枚举类型的使用，是制定一个名字列表，使其中每个名字对应一个默认的数值。从另一种角度来说是令每个数值拥有一个类似宏定义的命名，赋予数值一个具有语义意义的名字，便于后续代码的维护和管理。

### 4. (简答题)

**游戏场景中，如果绘制出来的舞台边界(如贪吃蛇游戏中的4道墙体)不能闭合。可能的原因是什么？**

*正确答案：*

4道墙的坐标值给得不合理

### 5. (简答题)

**《贪吃蛇》游戏中，游戏舞台的尺寸设定为高度height，宽度width，但本节案例的实际执行稍有偏差。以舞台的第i行为例，左墙的坐标为（0，i），右墙坐标为（width-1，i），实际舞台的宽度为width-2。该情况可能会为后续蛇的移动判定带来困扰。请尝试设计一种舞台的游戏数值表达方案，使得舞台左右墙体和贪吃蛇舞台坐标的表示，不会出现混乱。**

*正确答案：*

答案：

//光标位置设定函数，让舞台坐标XY与屏幕坐标xy关联，让用户不用去考虑屏幕的物理坐标

void setPos(int X, int Y)

{

COORD pos;

pos.X = CORNER\_X + X + THICHKNESS;

pos.Y = CORNER\_Y + Y + THICHKNESS;

SetConsoleCursorPosition(h, pos);

}

////////////////////////

for (int i = 0; i < height; i++)

{

setPos(-THICHKNESS, i);//绘制左右的墙

for (int j = 0; j < width+THICHKNESS\*2; j++)

{

if (j == 0)

                            cout << "#";

else if (j == width + THICHKNESS)

      cout << "#";

else

      cout << " ";

}

cout << endl;

}

### 1. (简答题)

**为什么本节输入模块中不采用直接侦测键盘硬件中断的GetAsyncKeyState( )函数？**

*正确答案：*

该案例输入模块中涉及多个按键的响应，如单纯使用GetAsyncKeyState( )函数进行按键状态问询，会延伸出较多的按键代码判断，进而影响该模块的代码执行效率。

本节采用kbhit()函数进行按键响应的两级管理，即先判定是否有按键按下，再判定是哪个按键被按下，进而提高代码的执行效率。

另外，GetAsyncKeyState( )函数过于强大，如果需要用到组合按键，例如方向键“右”、“上”同时按下，则GetAsyncKeyState( )函数首选。但《贪吃蛇》的输入操作较简单。用大炮去打蚊子显得有点浪费。

### 2. (简答题)

**《贪吃蛇》游戏的输入模块中采用switch()语句进行按键筛选，可能会存在怎样的弊端？**

*正确答案：*

当玩家同时按下多个按键的时候，switch()语句的筛选方式，

只要检测到其他键的存在，switch就会往default分支走；

可能无法进行准确识别按键。

就所谓的，总有刁民想谋害朕！你不知道用户会怎么去折腾使用你的程序

### 3. (简答题)

**《贪吃蛇》游戏的输入模块，如果想采用键盘上的“←”、“→”、“↑”、“↓” 左右上下四个方向键进行控制，代码需要如何改？如果“w”、“a”、“s”、“d”按键和方向键均能控制，又如何改？**

*正确答案：*

在用\_getch()函数来读取按键信息时候需要注意：

1、使用getch读取字符时，读取一次就行

2、而读取方向键时，需要读取两次（第一次的返回值为224）

\_getch()函数在读取一个方向键时，函数会返回两次，第一次调用返回0xE0(224)，第二次调用返回实际的键值。

因此想在console版贪吃蛇中使用方向键，需在Input()函数中添加如下代码：

if (\_kbhit())

{

               ...

case 224:                    //方向键区的ASCII码

switch (\_getch()) {

case 72: //上

dir = UP;

break;

case 80: //下

dir = DOWN;

break;

case 75: //左

dir = LEFT;

break;

case 77: //右

dir = RIGHT;

break;

}

default:

break;

}

}

### 4. (简答题)

**全局变量的使用弊端请至少罗列3种.**

*正确答案：*

①可以被任意访问和随意更改，增加维护成本；②容易与局部变量重名，产生混淆冲突；③生命周期长，长期占用资源；④在资源回收时候容易被遗忘，导致内容泄露等问题；⑤容易引入了隐藏的依赖关系，干扰代码的模块化。

### 5. (简答题)

**游戏设计有哪4个组成要素？**

*正确答案：*

任何类型的游戏设计，都应该包含以下四种元素：行为模式、条件规则、娱乐身心和输赢胜负。

### 6. (简答题)

**《贪吃蛇》游戏中，如何让蛇身能够穿墙。即蛇头进入右(上)墙，会从左(下)墙出来？条件应该怎样设定？**

*正确答案：*

对蛇头位置和墙的位置进行距离判断，如果相撞，则将蛇头位置跳转，代码如下所示。

if (x >= width) x = 0; else if (x < 0) x = width - 1;

if (y >= height) y = 0; else if (y < 0) y = height - 1;

由于蛇身是跟随蛇头的坐标移动，蛇身跳转，后续的蛇身也会跟随着挨个跳转。给人一种蛇会穿墙的感官

注：穿墙判定的临界点，不同人的代码可能会有不同的取值，此处大家酌情考虑

**1. (单选题)**

**以下哪个选项不属于游戏框架的主要模块？()**

* A. 游戏的初始化
* B. 过程循环
* C. 游戏逻辑
* D. 胜负判断

*正确答案:* D

**2. (单选题)**

**贪吃蛇游戏，在游戏画面上如何表现或示意“水果”被蛇吃没了？**

* A. 用背景图案将原来水果图案覆盖掉；
* B. 把水果图案直接拿掉；
* C. 逻辑中不产生水果信息，画面上自然就不会有水果；
* D.

用蛇的图案将原来水果图案覆盖掉；

*正确答案:* A

**3. (单选题)**

**贪吃蛇游戏中，蛇每吃一个果子，蛇身会增长一节。为让游戏场景中的蛇身变长，以下哪种操作和说法比较合理**

* A. 碰撞函数中，当蛇头与果子相遇，则多绘制一节蛇身；
* B. 绘图函数中，每增加一节蛇身绘图函数就多绘制一节；
* C. 逻辑函数中，每增加一节蛇身，逻辑函数就给蛇身链表增加一节蛇身数据；
* D. 循环函数中，每增加一节蛇身，蛇身链表的循环次数加1。

*正确答案:* C

### 1. (简答题)

**为什么游戏窗口会出现闪屏？请讲述原因。**

*正确答案：*

答：当屏幕缓冲区内的显示字符在持续输入时，容易导致屏幕画面各局部区域的驻留时长不一致，会让观察者感觉窗口画面在闪烁。

或者：游戏窗口画面出现闪烁的原因，本质上来说是因为在一个游戏周期内，游戏画面的分步输出导致屏幕各区域内容驻留时长不一致。

### 2. (简答题)

**一个窗口可以有几个屏幕缓冲区？**

*正确答案：*

多个

### 3. (简答题)

**如何将变量的数值写入到屏幕缓冲区的显示字符数组中？sprintf()函数的作用是什么？**

*正确答案：*

将数值转换为字符串进行显示。sprintf()函数可以将数值以指定的格式转存到字符数组中。

### 4. (简答题)

**请简述什么是双缓冲显示。**

*正确答案：*

答：双缓冲技术的核心是启用两个显示缓冲区。当其中一个缓冲区的内容在显示的时候，对另一个缓冲区进行新内容的绘制，等后台缓冲区绘制完毕后，将两个缓冲区进行互换，实现画面的整体切换，避免屏幕出现闪烁的情况。

### 5. (简答题)

**双缓冲显示和单缓冲显示比较起来，游戏的刷新速度是变快了？还是变慢了？可能的原因是什么？**

**注：这里的单缓冲，不考虑缓冲里局部更新的情况**

*正确答案：*

双缓冲显示比全局更新的单缓冲显示的游戏刷新速度快。深层次的原因在于，单缓冲显示的清屏操作可能是通过system（"cls"）实现。在调用system函数的时候，相当于游戏中断，并将控制权交给操作系统。而操作系统往往不止游戏这一个程序在运行，导致system函数的响应会出现不及时的情况，因此才造成单缓冲显示的运行速度变慢。

### 6. (简答题)

**双缓冲显示通常是先将被显示的字符存入缓冲区对应的二维数组中，然后再由二维数组转存到屏幕缓冲区。在这种由显示字符数组直接转存缓冲区的双缓冲显示的模式下，如何实现多种字体颜色的显示？**

*正确答案：*

屏幕缓冲区是用于存放在控制台窗口中输出的字符和颜色数据的一段存储区域。每个屏幕缓冲区字符单元的CHAR\_INFO结构的Attributes成员是用于存储字符的前景及背景颜色。WriteConsoleOutputAttribute()函数可对屏幕缓冲区指定区域单元的字体属性进行颜色设置。

### 1. (简答题)

**游戏局部绘制（更新）的操作思路是什么？**

*正确答案：*

答：①绘制游戏场景；②用背景色或背景图案去擦除前帧的游戏对象；③绘制当前帧的游戏对象

### 2. (简答题)

**局部重绘需要先擦除原来的游戏对象，再绘制新的游戏对象。但从游戏循环的角度观察，新绘制的对象在下次循环的时候，其实也是很快就被擦除的。请问为什么局部重绘的方式可以解决游戏画面闪烁的问题？**

*正确答案：*

答：①局部绘制比全局绘制的速度更快，耗时更小；②局部绘制通常会在绘图模块进行类似sleep操作，以控制游戏的帧频，该操作使得更新的内容会有更多的屏幕驻留时间。总之，耗时更小的绘制时间和较长的画面驻留时间有助于抑制屏幕的闪烁。

### 3. (简答题)

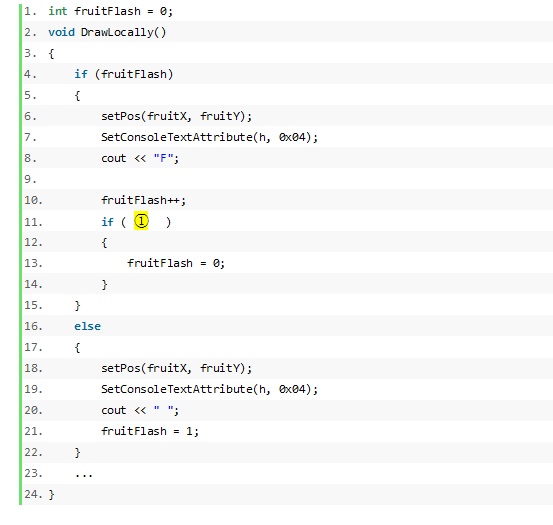
**上一章节《贪吃蛇》游戏绘制模块的cout输出，多输出一个空格符或少输出一个空格符都会影响到舞台右墙的排列。同样是cout输出，为什么局部绘制方案中的cout输出空格符却是覆盖舞台原位置的字符？**

*正确答案：*

答：每次的cout都是在指定的光标位置处进行输出。之前案例的cout时候，光标会根据输出的字符数自动右移，每次新的cout输出时候，光标的位置与之前已经不一样，所以会对舞台的右墙位置造成影响。后一个案例的cout的位置光标是被指定的，而原来屏幕上的输出又不会自行挪位，所以cout的输出会覆盖原来位置上的输出。

### 4. (简答题)贪吃蛇游戏中，如何实现水果的闪烁，并使得闪烁的频率是亮4下、灭1下？请说明编程实现的思路。

**阅读以下代码，并对第11行代码①处进行完善。**



*正确答案：*

正确答案：通过bool变量的切换，来决定当前时刻是否要绘制水果，以此实现水果的闪烁效果。

 要实现亮4下、灭1的闪烁频率，①处的代码应填写为fruitFlash % 5 == 0（或fruitFlash == 5或fruitFlash > 4 都给分）

(只简单给出①处的答案扣10分)

### 5. (简答题)

**游戏开发在进行框架搭建的时候，轻易不去变更未涉及的模块的内容。当进行贪吃蛇的局部重绘的时候，如果遇到新增的蛇身图形，首先出现在游戏场景的左上角，再下一帧才会跟随在蛇身之后。请问原因是什么？应该如何更改？**

*正确答案：*

答：因为eraseSnake()和DrawLocally()分处Logic()的前后端，使得原来逻辑上的一个问题被暴露出来：当nTail++后，该什么刷新蛇身链表的数据？蛇头的坐标应该什么时候刷新？

    原来逻辑执行的先后顺序是先刷新蛇身链表数据，后再触发蛇身增长事件，新增的蛇身就不能及时的附在原蛇身后面。至于新蛇身出现在舞台坐上角，那是因为蛇身链表的坐标，初始化的时候赋值为0。如果初始值不为零，还不一定会在左上角。

**1. (单选题)**

**Console界面的游戏，将游戏舞台的宽高比设置为1:1的时候，为什么实际画面的宽高比并不是1:1而呈现为长方形？可能的原因是哪个？**

* A.

屏幕的宽高比是16:9，渲染时候游，戏画面被拉伸了；

* B.

Console下单字节字符尺寸的宽高比是1:2

* C.

Console界面下字符的行间距大于字间距

* D.

观察者自己的视觉误差

*正确答案:* B

**2. (判断题)**

**程序的源代码要使用半角标点进行书写，但是字符串内部的数据可以使用全角字符或者全角标点。**

* A. 对
* B. 错

*正确答案:* 对

### 3. (简答题)

**如何在编程中为程序的输出显示更换字符图案？有什么注意事项？**

*正确答案：*

可以从charmap（字符映射表）中查找字符。不同平台对不同字符集的支持程度会有差异（实际上是程序工程项目的参数设置问题，比如是否支持双字节、多字节等）。

注意字符是单字节字符还是双字节字符，如果弄错会导致乱码的产生

### 4. (简答题)当目标计算机不支持中文等语言，目标计算机上只能使用半角字符。那在该计算机上运行界面中含有全角字符的游戏，会出现什么样的情况？请给出一个可行的解决方案。

*正确答案：*

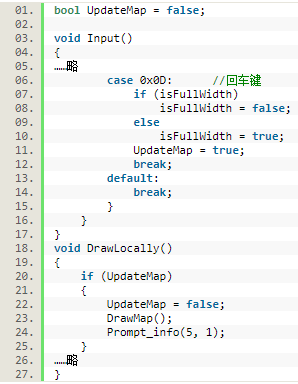
游戏界面上全角字符的位置上会出现乱码或异常的问号。

游戏程序中自行携带中文等字体文件，并以相应的字节数（例如双字节）进行全角字符显示。

### 5. (简答题)在游戏代码的输入函数中直接加入图形绘制代码，似乎代码开发的时候很高效，其实已严重破坏程序代码的模块结构，会给后续的代码更新维护工作带来很多隐患。下方输入函数中的第25、26行代码，能不能将之移到绘图函数Draw()中进行实现？请论述你的观点，并给出你的解决方案。



*正确答案：*

思路相近，逻辑正确，则给分。绘图函数根据Input函数的处理结果，决定是否绘制相应图形内容

**1. (单选题)**

**以下按键的键码是双字节的是？**

* A.

ESC键;

* B.

Delete键

* C.

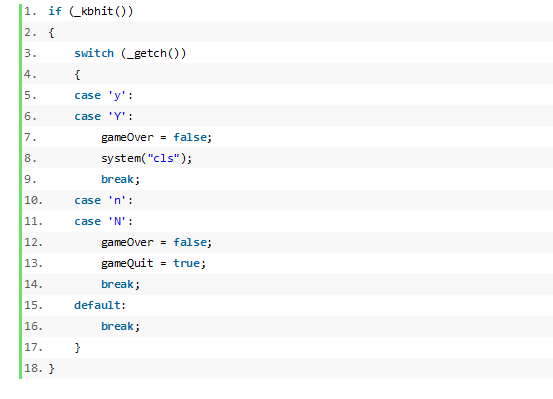
Backspace键

* D.

Enter键.

*正确答案:* B

### 2. (简答题)阅读以下代码，请回答第5、6、10、11行代码中 case 'y': case 'Y':以及case 'n': case 'N':的设计目的是什么？

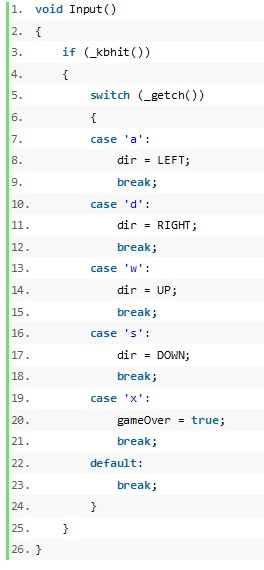
****

*正确答案：*

正确答案：使得按键Y和N，在大小写状态下能有相同的响应。用户不需关注当前按键的大小写状态，只要相应按键被按下，则执行物理按键对应的内容。

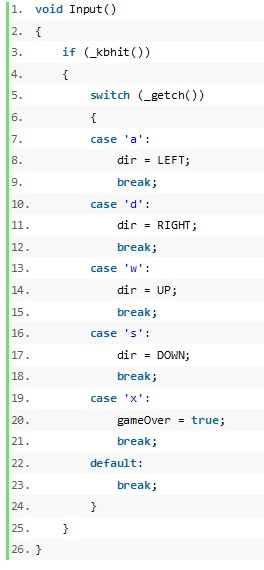
文字表述意思相近既给分

### 3. (简答题)贪吃蛇游戏，当蛇头与蛇身相撞时候，游戏即结束。但当蛇头向后退也会触发该条件。请阅读以下代码，通过对以下代码进行完善，避免蛇头后退，从而导致游戏结束的情况出现。在代码完善的过程中，请注明在哪一行代码的前面，插入哪些代码内容。例如：在第1行前，插入dir = STOP; 也可把代码自行写全。

****

### 3. (简答题)贪吃蛇游戏，当蛇头与蛇身相撞时候，游戏即结束。但当蛇头向后退也会触发该条件。请阅读以下代码，通过对以下代码进行完善，避免蛇头后退，从而导致游戏结束的情况出现。在代码完善的过程中，请注明在哪一行代码的前面，插入哪些代码内容。例如：在第1行前，插入dir = STOP; 也可把代码自行写全。

### *正确答案：*

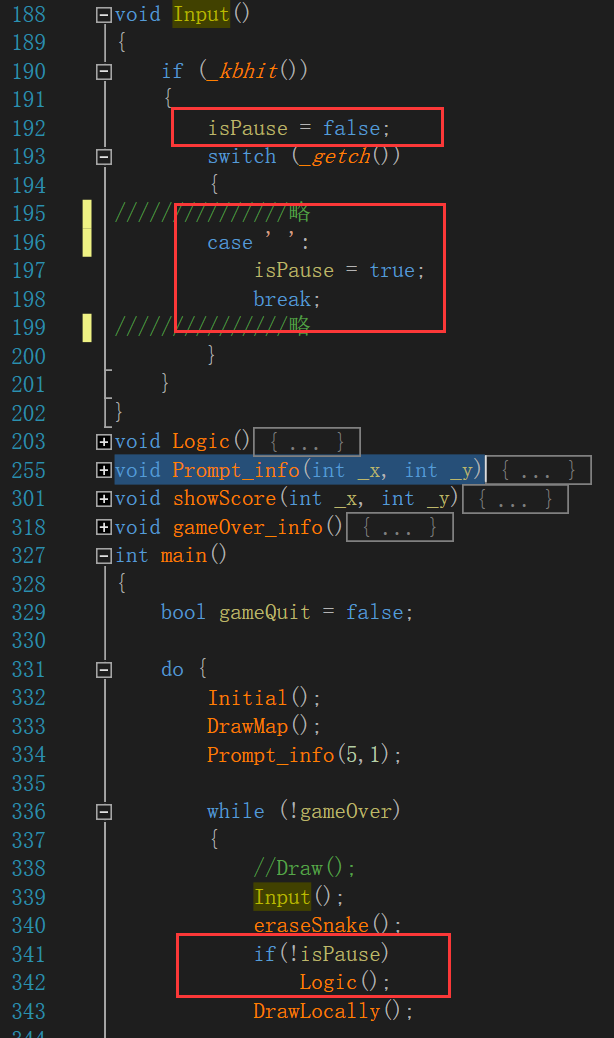
****

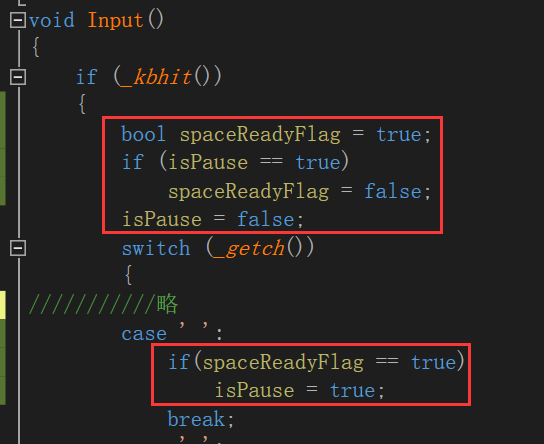
### 4. (简答题)

**请设计一种在游戏进行过程中暂停游戏的方法？并设计方案，使得在游戏运行的时候，按下空格键则游戏暂停，再按任意键，游戏继续。**

*正确答案：*

以下，第一种的答案给10分，因为如果第二下按空格键的话，无法继续游戏；第二种方案给20分

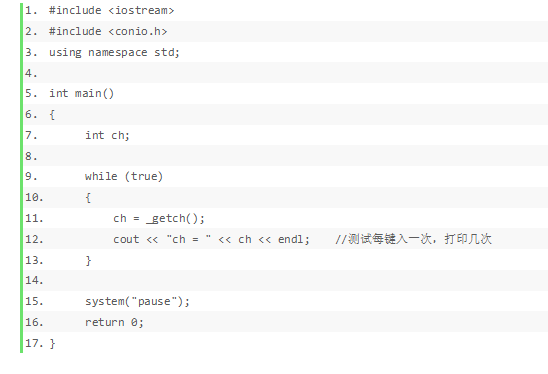




### 5. (简答题)

**请设计一段代码用于检测键盘按键的键码值。**

*正确答案：*



运行上方测试代码会有两种结果：

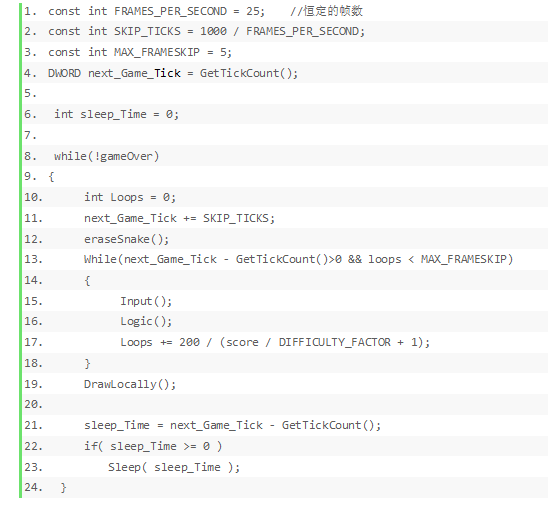
①1键1键码。英文字母、数字键、以及Tab、Space、ESC、Backspace、Enter等常用键，每按一次相应按键，\_getch()函数会立即返回真实的键码值。

②1键2键码。键盘的”↑←↓→”方向键，F1~F12、Delete等功能键，每按一次相应按键，\_getch()函数会返回两个键码值。

### 2. (简答题)

**本节案例中，一次游戏循环周期里面，游戏逻辑和图形绘制各执行一次，改变游戏循环周期的时长，会使得游戏逻辑和图形绘制的频率均发生变化。请问能否在保持游戏绘制频率恒定的情况下，单方面的改变游戏逻辑的执行频率，使得游戏的节奏发生变化？请给出你的理由或方案？**

*正确答案：*



### 3. (简答题)

**对于游戏框架的模块大家应该已有初步认识。贪吃蛇console版的开发，我们前前后后添加了不少功能。请大家回忆下，我们前后各个功能代码的添加，是否有破坏原先的模块框架？**

*正确答案：*

无标准答案，请自行判断

### 4. (简答题)

**已知程序窗口尺寸、游戏舞台尺寸、提示信息窗的位置坐标等变量。其中窗口、舞台和信息窗可以分别用Window、Stage、Info表示。舞台的宽度和顶点坐标可以分别用StageWidth、StagePos.x、StagePos.y表示，如此类推。请设计一段伪代码，使得当游戏舞台尺寸变化时候，提示信息窗的位置坐标能动态变化。**

*正确答案：*

要点在于“提示信息窗的位置坐标”与游戏舞台尺寸的变量关联。比如

Info.x = StageWidth/5\*6;Info.y = StageHeight/5;//这样信息窗的左顶点就会处于舞台右边的StageWidth/5处。

Info.x = WindowWidth-InfoWidth;Info.y = 0;//这样信息窗就会固定在窗口的右上角。

### 5. (简答题)

**贪吃蛇游戏，本章节课的代码与第一次课的代码相比，差异在哪里？最早建立的几个函数Initial()；Input(); Logic(); Draw ();哪些一直没更改过？哪些中间被更改了？更改的原因是什么？**

*正确答案：*

无标准答案，请自行判断给分

**1. (单选题)**

**SFML一共有5个模块（库），以下哪个不是？**

* A.

system

* B.

window

* C.

render

* D.

network

*正确答案:* C

**2. (单选题)**

**以下关于SFML库的说法错误的是？**

* A.

SFML是基于OpenGL开发的;

* B.

SFML是用C/C++编写的；

* C.

SFML的使用没有任何限制，除了不能把SFML的源码说成是自己开发的源码；

* D.

SFML库有依赖外部的类库，但自己内部模块没有相互依赖性。

*正确答案:* D

**3. (单选题)**

**SFML做场景更新的时候，每次都先用sf::window的clear()函数进行窗口清除，SFML的画面显示采用的是什么机制？**

* A.

局部刷新

* B.

实时绘制

* C.

单缓冲

* D.

双缓冲

*正确答案:* D

**4. (单选题)**

**在用SFML进行开发时候，为什么有的人代码里面可以直接使用Sprite类及它的成员函数，不需要增加前缀sf::，而有的人不用sf::前缀程序就会报错？**

* A.

不同开发环境对SFML函数的调用要求不同

* B.

开发环境配置不同

* C.

没有进行命名空间声明的，使用SFML库函数等，需要在函数前追加sf::

* D.

不同SFML版本的函数调用方式存在差异

*正确答案:* C

**1. (单选题)在SFML开发的窗口界面显示中，中文输出变成乱码。请问最有可能的原因是什么？**

* A. 没有采用宽字符的设置或表示
* B. SFML是目前还不支持汉字显示
* C. 程序的字体库丢失
* D. 没有采用UTF-8编码

*正确答案:* A

**2. (单选题)**

**函数static bool sf::Keyboard::isKeyPressed(Key key)的功能与下列哪个函数类似？**

* A.

kbhit();

* B.

GetAsyncKeyState( );

* C.

GetKeyState( int vKey );

* D.

GetKeyboardState( );

*正确答案:* B

**3. (单选题)**

**以下说法正确的是？**

* A.

SFML的显示绘制可以采用画面局部更新的策略；

* B.

SFML通常用窗口的draw()函数将精灵Sprite，文本Text和形状Shape等对象直接绘制在屏幕上；

* C.

SFML在绘制对象的时候，采用的是“清除-绘制-显示”模式；

* D.

SFML的精灵Sprite，文本Text和形状Shape等对象是通过窗口的display()函数绘制在屏幕上。

*正确答案:* C

**4. (多选题)**

**在SFML体系下，如何进行游戏窗口更新速度的设定：（多选题）**

* A.

使用计时器或时钟相关函数进行更新速度的管理；

* B.

使用setFramerateLimit()函数设定游戏的帧频；

* C.

对图形绘制函数进行延时设定；

* D.

对游戏的逻辑数值计算进行延时设定；

*我的答案:* AB *正确答案:* ABCD

### 5. (简答题)

**SFML多媒体库中的窗口和控制台程序的窗口有什么不同？**

*正确答案：*

SFML中有专门的窗口类，游戏窗口被作为窗口类的实例对象管理。

### 6. (简答题)

**控制台程序在绘制游戏舞台的时候并没有栅格尺寸GRIDSIZE的概念，游戏舞台的尺寸直接就由舞台的宽、高数值确定。在SFML等游戏引擎开发的过程中需要考虑栅格尺寸。请问这是什么原因？**

*正确答案：*

控制台程序绘制的游戏舞台，实际上是有字符组成的，每个字符在屏幕上所占的面积，实际上就是栅格的面积。控制台程序中栅格的尺寸就是一个全角字符或者一个半角字符的尺寸。字符的字体调大之后，栅格的尺寸和面积就跟着变大。栅格可以认为是绘图的基本单元，通常它的尺寸是游戏对象在舞台上移动的最小刻度。它在绘制游戏场景的时候一直都存在的。

### 7. (简答题)

**当您成功加载了纹理，并且正确构造了精灵，但在屏幕上看到的只是一个白色正方形。请问可能发生了什么事？**

*正确答案：*

这是一个常见的错误。设置精灵的纹理时，它在内部所做的全部工作是存储于*指向*纹理实例的*指针*。因此，如果纹理被破坏或移动到内存中的其他位置，则精灵将以无效的纹理指针结束。

当您编写这种函数时，会发生此问题：

sf::Sprite loadSprite(std::string filename)

{

     sf::Texture texture;

     texture.loadFromFile(filename);

**return** sf::Sprite(texture);

} *// error: the texture is destroyed here*

您必须正确管理纹理的寿命，并确保它们在精灵使用的时候一直有效。

### 8. (简答题)

**SFML用函数pollEvent(Event & event)获取窗口消息队列中的消息。这个函数没有采用阻塞，如果没有挂起的事件消息，那它将返回false。为什么关于该函数的调用，通常是采用循环调用的方式进行？例如while(window. pollEvent(event)).**

*正确答案：*

因为窗口的消息队列中可能存在多个事件，需要循环调用此函数，以确保每个挂起的事件消息都能得到处理。

**1. (单选题)以下关于精灵和精灵表单的说法错误的是（）。**

* A. 精灵是游戏世界中的一个2D视觉对象，可以通过单张图像对其进行绘制
* B. 精灵只能用来表示运动的游戏对象
* C. 精灵对象有一个“绘制顺序”属性，其作用是保证按照前后顺序对场景中的精灵进行正确绘制
* D. 使用精灵表单的主要目的是节省精灵图像的存储空间

*我的答案:* B *正确答案:* B

**2. (单选题)精灵对象要做旋转时候需要指定旋转的中心点。但为什么对一个精灵对象指定旋转作用点的时候，其他的精灵对象通常也需要一起指定作用点？原因是什么？**

* A. 让各素材的坐标原点保持一致，便于统一素材管理
* B. 其实不指定坐标原点，影响也不大
* C. 精灵类的中心点设置，要求这样操作
* D. 避免游戏的逻辑发生混乱。

*我的答案:* A *正确答案:* A

**3. (单选题)SFML中，进行文字显示的操作顺序哪个正确？**

* A. ①选定文字内容、②设定字体大小、颜色、位置等属性、③添加到显存、④文字绘制并显示
* B.

①选定文字内容、②指定字体、③设定字体大小、颜色、位置等属性、④文字绘制并显示

* C.

①指定字体、②设定字体大小、颜色、位置等属性、③文字做光栅化处理、④文字绘制并显示

* D. ①指定字体、②设定字体大小、颜色、位置等属性、③添加到显存、④文字绘制并显示

*我的答案:* B *正确答案:* D

**4. (单选题)**

**SFML中，游戏舞台的内容和边框文字等界面内容，每次都需要在draw()函数中进行绘制么？以下说法正确的是（）**

* A.

不需要，除非边框文字等界面内容发生变化了，才需要重新绘制；

* B.

不需要，SFML所采用的缓冲机制，对于未更新的内容进行了自动保留；

* C.

绘制是发生在display()函数调用环节，需要将绘制的内容每次都添加到display()中，内容才会显示

* D.

需要，SFML采用的是双缓冲机制，如果在某次draw()函数绘制时未去刷新边框文字等界面内容，则屏幕上就不会有对应内容显示。

*我的答案:* D *正确答案:* D

### 5. (简答题)

**当精灵、图形等实例对象的原点origin位置发生变更后，即使其position属性没有更改，它在屏幕上的绘制位置也会发生变化。请问为什么？**

*正确答案：*

精灵、图形等实例对象的position坐标实际上指得是它们的原点origin在舞台窗口上的坐标。以舞台坐标为参照系，实例对象上的原点origin位置变更，等同于原先origin点在舞台上的坐标发生偏移，所以在屏幕上的绘制位置发生了变化。

### 6. (简答题)

**游戏开发中，当有精灵对象需要重设旋转中心点时，通常会将所有的精灵对象的原点进行统一调整。需要做旋转操作的精灵对象需要调整坐标原点，这个无可厚非。但不需要做旋转的精灵对象也重设坐标原点，原因是什么？如果不对全体精灵对象进行统一的坐标原点调整可能会发生什么事情？**

*正确答案：*

答案：精灵对象的原点发生变更后，它在舞台上的位置会发生相应的偏移。与其它未做原点变更的精灵在使用上时需要做区分，会给游戏对象的管理造成不便。

不然会出现个别精灵对象的图形在舞台上存在错位的情况，影响游戏画面的效果。

### 7. (简答题)

**SFML程序中，当游戏循环绘图模块的window.draw()调用的次数多了之后，发现游戏的FPS下降了。请问可能的原因是什么？**

*正确答案：*

首先，请检查FPS值是否未一直徘徊在某些特殊值（60、75、80、144等）附近。如果不管您做什么，它都会徘徊在这些值附近，则很可能是人为地将其限制在该数值。在继续下一步之前，请仔细检查是否有此类限制。

许多人在谈论应用程序中的低FPS时会犯一个错误，就是认为这是由显卡的图形绘制引起的。这可能是这样，但并不总是如此。通常，应用程序执行的CPU工作量要比实际图形处理工作多得多，并且应用程序受CPU约束，而不是GPU约束。估算应用程序是否受CPU约束的一种粗略方法是监视应用程序的CPU使用率。如果利用率接近100％，则很有可能受到CPU的限制。在这些情况下，减少每帧的CPU负载将导致每帧的时间缩短，因此在给定的时间段内将产生更多的帧。

一旦感觉到应用程序受CPU限制，查找热点（CPU花费时间最多的地方）的最佳工具是性能分析器。任何性能分析器都可以轻松找到热点。从那里，您可以尝试根据CPU（而非GPU）负载优化代码。

但是，如果您发现应用程序不受CPU限制并且FPS值仍然很低，则需要开始评估绘制方式。SFML利用OpenGL进行渲染。使用OpenGL进行绘制所花费的时间中，有不是很重要的一部分，花费在驱动程序中以及与图形硬件的通信中。自然，如果减少驱动程序的调用以减少与硬件的通讯沟通，将获得更好的图形性能。这就是为什么第一个优化策略是考虑将多个较小的对象打包成为较大的对象（例如sprite sheet），以便同时绘制的原因。这部分与SFML本身无关。您将必须评估什么是最佳的操作方案，并使用SFML为您提供的工具设计一个优化的解决方案。在这方面进行优化的一般经验法则是：每个帧中.draw()函数调用越少，运行速度就越快。

**1. (单选题)使用SFML做游戏开发，设定按下空格键进行游戏模式切换。但按键响应时灵时不灵。请问以下表述，哪种是对的？**

* A.

SFML的功能有缺陷，无法根除该问题；

* B.

按键按下，容易瞬间多次触发响应，建议改用按键释放（KeyReleased）响应的方式。

* C.

应提高游戏窗口消息队列的查询频率，以提升按键的响应速度；

* D.

按键状态判断建议采用Keyboard::isKeyPressed(Keyboard::key)进行状态问询，以节省消息队列的查询等待时间。

*我的答案:* B *正确答案:* B

**2. (单选题)《贪吃蛇》游戏在蛇转向的时候，如何确保蛇能够走完完整格后再转向？**

* A.

确定蛇走完完整格后，再给蛇传递新的方向指令；

* B.

对位置坐标点的小数取整进1，然后转向；

* C.

将最小步进距离设置为整数，避免蛇在非整数格位置移动；

* D.

以上都对；

*我的答案:* D *正确答案:* D

**3. (单选题)**

**《贪吃蛇》游戏中，如果要实现细腻的蛇行移动，蛇身的移动应该如何实现？**

* A.

分方向设细化的步进为stepX、stepY，每次绘制蛇身时候，都增加响应的stepX、stepY值。

* B.

设细化的步进为step，绘制蛇身时候，严格的让后一节蛇身跟随前一节进行移动，步长为step。

* C.

保持游戏地图屏幕尺寸不变的情况下，减少单元格的尺寸，增加单元格的数量，使得蛇身移动更加细腻；

* D.

以上都对

*我的答案:* D *正确答案:* D

**4. (单选题)**

**sf::Sound和sf::Music，哪个适合用来播放游戏中比较短小的声音，比如枪声、脚步声等。**

* A.

sf::Sound

* B.

sf::Music

* C. sf::SoundBuffer
* D. sf::SoundStream

*我的答案:* A *正确答案:* A

**5. (单选题)**

**SFML中，以下哪个函数可以让音频循环播放？**

* A.

play()

* B.

setPitch()

* C.

setLoop()

* D.

getLoop()

*我的答案:* C *正确答案:* C

### 6. (简答题)

**《贪吃蛇》游戏步进细化中，每一节蛇身的细化步长的前进方向是如何确定的？请描述一下解决思路。**

*正确答案：*

对于蛇身的位置则需分情况讨论。首先明确的是，后一节蛇身是跟随前一节进行移动。跟随的方式可以细分为2种：①水平方向跟随；②竖直方向跟随。

当进行水平方向跟随的时候，第i节蛇身与第i-1节蛇身的Y坐标相同，Y方向不发生坐标偏移；第i节蛇身的X坐标需要额外增加步进的细分步长stepLength，至于步进的方向，则由前后节蛇身x位移的差值来表示。

当进行竖直方向跟随的时候，第i节蛇身与第i-1节蛇身的X坐标相同，X方向不发生坐标偏移；第i节蛇身的Y坐标需要额外增加步进的细分步长stepLength，至于步进的方向，则由前后节蛇身y位移的差值来表示。

### 8. (简答题)

**游戏中如何确保精灵对象以恒定速度进行移动？**

*正确答案：*

精灵移动与帧速率脱钩方案的具体实现思路如下：

①指定精灵的速度，例如每秒100像素。

②将该速度转化为每一帧的移动距离。将速度与经过的时间相乘得到对象的行进距离。

具体代码示例如下所示：

*// speed的单位为像素/秒*

**float** speed = 100.f;

*// 创建用于计时的clock对象*

sf::Clock clock;

**while**( window.isOpen() )

{ *// 获取前1帧所经过的时间，单位为秒*

**float** delta = clock.restart().asSeconds();

*// 更新精灵的位置*

sprite.move( speed \* delta, 0.f );

*// 绘制对象*

window.clear();

window.draw( sprite );

window.display();

}

按照以上思路，不管帧速率如何，示例代码都会以每秒100像素的速度移动精灵对象。

### 9. (简答题)

**在用SFML进行音频控制的时候，我们可以用play()函数播放音频，用stop()函数停止音频，用pause()函数暂停音频。请问，当音频被暂停后，如何从暂停的位置继续播放音频？**

*正确答案：*

调用play()函数可以开始或继续播放音频

### 10. (简答题)

**如若计划采用按键的sf::Event::EventType::KeyPressed状态进行条件判定。请在游戏循环中设计一段代码实现声音开关的开启/关闭功能。**

*正确答案：*

参考代码如下

bool keyReady = true;

……

if (event.type == sf::Event::EventType::KeyPressed &&  event.key.code == sf::Keyboard::Enter)

{

if (keyReady == true)

{

keyReady = false;

if (MusicOn == true)

{

bkMusic.pause(); //bkMusic.stop();

MusicOn = false;

}

else

{

bkMusic.play();

MusicOn = true;

}

}

}

if (event.type == sf::Event::EventType::KeyReleased &&  event.key.code == sf::Keyboard::Enter)

{

keyReady = true;

}

或者直接关闭按键的重复响应设置

window.setKeyRepeatEnabled(false).

### 1、游戏进行过程中如何暂停游戏？如何恢复暂停的游戏进程？

*正确答案：*

通过bool变量gamepause来对Logic函数进行管理。当gamepause为true，则不执行Logic函数；为false则执行Logic函数。类似以下的操作也是对的。核心在于是否对Logic进行冻结，让游戏数值不发生变化

if (!gamepause || !gameover) {

if (delay % 5 == 0)

{

delay = 0;

Logic();

}

Draw();

}

else Draw();

### 2. (简答题)能否例举3种方式，将int或float型的变量数值转为字符串进行显示输入。

*正确答案：*

例1：stringstream

std::stringstream ss; int score;

ss << score;

text.setString(ss.str()); window.draw(text);

例2：sprintf函数

char str[20];

int a=20984,b=48090;

sprintf(str,"%3d%6d",a,b);

str[]="20984 48090"

例3：itoa() 等函数

itoa()  、\_itoa\_s()     将整数转化为字符串（非标准函数）

          char \* itoa ( int value, char \* str, int base );

          value： 待转整数

          str：指向以‘\0’结尾的结果字符串的内存

          base：结果字符串以哪种进制表示，可以选择2-36之间的所有值

### 3. (简答题)

**下方的输入模块设计，如果同时按多个键，存在蛇调头导致的可能，导致游戏立即结束。请设计一个改进方案，以避免这种情况。**

**if (Keyboard::isKeyPressed(Keyboard::Left)) //按键判定**

**if (dir != RIGHT)**

**dir = LEFT;**

**if (Keyboard::isKeyPressed(Keyboard::Right))**

**if (dir != LEFT)**

**dir = RIGHT;**

**if (Keyboard::isKeyPressed(Keyboard::Up))**

**if (dir != DOWN)**

**dir = UP;**

**if (Keyboard::isKeyPressed(Keyboard::Down))**

**if (dir != UP)**

**dir = DOWN;**

*正确答案：*

我们在课内前前后后应该讲过至少3种方案，然后各有利弊，请大家给出自己觉得最优的方案。

下方代码仅供参考

答案：设置一个bool变量dirChange，确保dir值在输入模块中只会被变更一次。例如：

bool dirChange = false;

if (Keyboard::isKeyPressed(Keyboard::Left)) //按键判定

if (dir != RIGHT && dirChange == false)

{dir = LEFT;

dirChange = true}

if (Keyboard::isKeyPressed(Keyboard::Right))

if (dir != LEFT&& dirChange == false)

{dir = RIGHT;  dirChange = true}

if (Keyboard::isKeyPressed(Keyboard::Up))

if (dir != DOWN&& dirChange == false)

{dir = UP;  dirChange = true}

if (Keyboard::isKeyPressed(Keyboard::Down))

if (dir != UP&& dirChange == false)

{dir = DOWN;  dirChange = true}

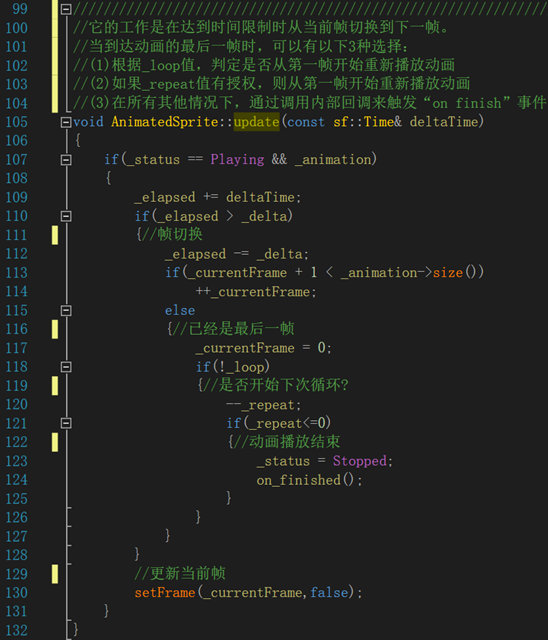
### 1. (简答题)

**游戏有帧频，精灵动画也存在帧频。游戏运行时候，有时会根据硬件平台的处理器性能对游戏的帧频进行动态调整。例如，如果硬件性能不足的时候，可以采用降低游戏帧频的方法，确保游戏运行的流畅度。请设计一个方案，确保在游戏帧频发生变化的时候，精灵动画的帧频保持稳定。**

**注：精灵动画帧频指精灵对象的动画一秒播放多少帧**

*正确答案：*

下方代码的第110行，确保精灵动画每\_delta时间播放一帧



### 1. (简答题)

**如何切换游戏难度（以《扫雷》游戏为例）？**

*正确答案：*

答案：①定义变量gameLvL对游戏的难度进行描述（分别取值 1 2 3），

     ②用switch语句，根据gameLvL取值进行游戏初始条件的重置，比如：舞台尺寸、雷数等等

     ③根据最新的游戏设定，调用 Initial()函数完成新的初始化工作

     ④进入新的一轮游戏循环，执行新的游戏逻辑、绘制新的舞台，实现游戏难度切换；

### 2. (简答题)

**如何绘制界面上的按钮？**

*正确答案：*

答案：SFML库本身并没有提供按钮相关的API函数或控件。不仅仅SFML没提供按钮，我们之前使用的C2引擎也没提供官方的按钮控件。究其原因，在游戏开发领域，对于游戏窗口上的图形，采用了统一的绘制更新机制（尤其是双缓冲），将任何的对象（包括按钮）都一视同仁对待。这些对象，通常都是以游戏精灵实例的形式出现。需要多个按钮，则实例化多个具有按钮图形的精灵对象，将各按钮素材加载到纹理区，读取按钮素材的像素尺寸， 录入各个按钮的显示坐标， 用 draw()函数进行绘制。

### 3. (简答题)

**如何实现游戏界面上的计时和计分显示？**

*正确答案：*

答案：本质上是将各个数字贴图显示在屏幕的指定位置

     ①加载带有数字图形的贴图，

     ②将计时和计分的数字，分解为百位、十位及个位数字。

     ③根据具体数字，从数字图形贴图中获取对应的数字贴图纹理

     ④将正确的数字贴图纹理绘制在正确的屏幕坐标上。

### 4. (简答题)

**如何布雷及确保玩家第一次点击的不是雷?**

*正确答案：*

答案：①布雷：通过随机函数生成随机坐标，在随机坐标上布置雷，当有效的雷数达到预设值时，则布雷结束。

 ②在玩家进行第一次鼠标左键点击之后再开始布雷，布雷时要回避第一次点击的位置坐标。以此确保玩家第一次点击的不是雷。

### 5. (简答题)

**如何使得玩家第一次鼠标点击处及其周围⼋个点都没有雷？**

*正确答案：*

答：玩家第一次鼠标点击之后再进行布雷。布雷时候，如果随机到的雷坐标位于点击处及周边8领域，则需重新布雷，以此杜绝第一次鼠标点击处及周边出现雷。

### 6. (简答题)

**如何实现鼠标双击之后自动揭开周边的未揭开方格？**

*正确答案：*

答案：对鼠标双击位置的周边8领域进行遍历检测，对于未揭开的方格，如果不是雷的，将之置为揭开状态。如果其中有“空”状态的格，则采用“空”操作。如果其中有未被正确判定的“雷”，则在方格揭开的同时，游戏结束。

### 1. (简答题)

**Windows如何区分鼠标双击和两次单击？**

*正确答案：*

Windows根据两个条件来做这个区分：

（1）双击的时间间隔

两次单击会产生四个鼠标点击消息，如果第三个消息（第二次按下）和第二个消息（第一次弹起）间隔短于指定值，则把第三个消息为鼠标双击消息；第四个消息为鼠标按键弹起。

这个指定的时间间隔，在Windows XP SP2上缺省是0.5秒，其他操作系统可能相同。在系统中，这个间隔值是可以通过API函数读取和重新设置的。例如通过::GetDoubleClickTime调用可以得到这个值。通过::SetDoubleClickTime进行设置。

（2）两次鼠标击点的空间距离

在第一次点击时，Windows以击点为中心，检测一个矩形区域，如果第二次点击不落在这个区域内，那就不把第三个消息算作WM\_LBUTTONDBLCLK消息。

这个矩形区域的缺省大小，在Windows XP SP2上缺省是4pt×4pt。可以以SM\_CXDOUBLECLK或SM\_CYDOUBLECLK为参数调用::GetSystemMetrics得到。这个值也是可以设置的。设置的方法是通过SPI\_SETDOUBLECLKWIDTH或SPI\_SETDOUBLECLKHEIGHT为第一个参数来调用::SystemParametersInfo。设置的结果对系统中其他的应用程序也起作用。

### 2. (简答题)Animate中的按钮元件有3个显示状态，①普通状态；②鼠标滑过状态；三鼠标点击状态。请结合本节的SFML鼠标事件，写一段伪代码实现按钮元件的功能。

*我的答案：*

*正确答案：*

例：

void Game::Input()

{

sf::Event event;

while (window.pollEvent(event))

{

if (event.type == sf::Event::MouseButtonPressed && event.mouseButton.button == sf::Mouse::Left)

{

if (ButtonRectRestart.contains(event.mouseButton.x, event.mouseButton.y))

{

ButtonRestartState = PRESSED;

}

}

if (event.type == sf::Event::MouseMoved)

{

if (ButtonRectRestart.contains(event.mouseMove.x, event.mouseMove.y))

ButtonRestartState = MOVE\_IN;

else

ButtonRestartState = MOVE\_OUT;

}

}

### 3. (简答题)

**鼠标双击操作，通常会先触发一次单击，再构成一次双击。如果鼠标双击中的第一次单击操作不执行单个单击功能，那么鼠标处理逻辑可能会变得非常复杂。请设计一种鼠标点击方案，使得双击时的第一次单击操作不被执行。即鼠标单击则出现单击响应，鼠标双击则只出现双击响应。**

*正确答案：*

鼠标的左键双击，通常需要经历“左键按下”、“左键松开”、“左键再按下”、“左键再松开”4个过程。只想对双击事件进行响应，而不处理第一次的“左键按下”事件，有下面的方法：

在收到“左键按下”事件后，不立即处理单击事件，而是引入计时器，动态的为每次鼠标单击计时。例如，如果500ms内鼠标发生再次点击，则执行双击响应，否则则执行单击响应。

或

       当进行某次单击的时候, 如果过了500个毫秒没发生第二次单击, 就认为是单击操作, 执行单击操作的内容。

        当进行某次单击操作的时候，如果这次单击操作与上次的时间差在500毫秒内. 就认为是双击操作。

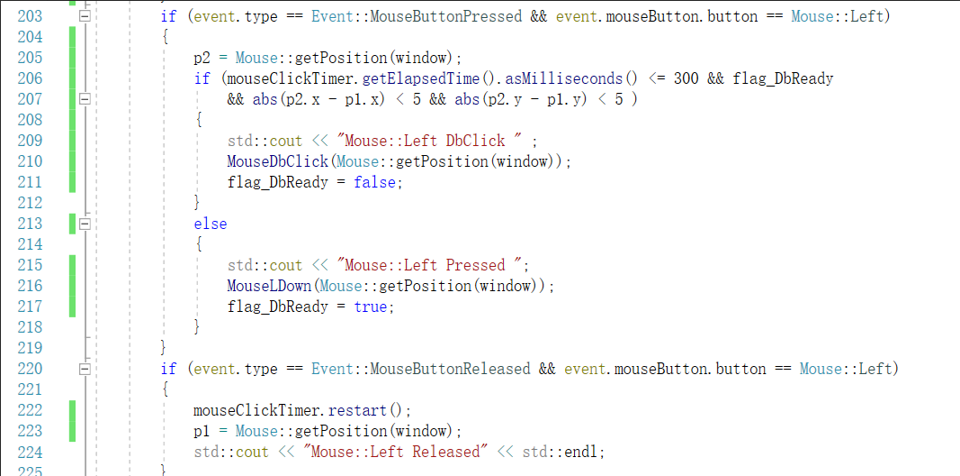
### 4. (简答题)

**基于前面的鼠标左键双击方案，当鼠标左键快速连击3下，可能激活两次左键双击事件。若希望只在左键连击偶数次的时候才激活左键双击事件，请问怎么改？请给出思路。**

*正确答案：*

设置一个bool变量，放置在“单击”响应的后面，设为true。在进行“双击”响应前判断bool变量是否为true，为true则执行双击响应，然后将bool变量设置为false；若为false，表示前面刚双击过，本次暂不执行双击响应，改为单击响应。

例如



### 1. (简答题)

**在IniData()函数中，所有的栅格的状态都被初始化为ncUNDOWN。但本节DrawGrid()函数在绘制栅格ncUNDOWN的状态，并不是依据栅格的mState值为ncUNDOWN。问题一：请问可能的原因是什么？问题二：既然如此，那在IniData()函数中将所有栅格的状态都初始化为ncUNDOWN的作用是什么？**

*正确答案：*

答：①每个栅格的状态首先分pressed和unpressed，在pressed以后，才是正常显示它的mState值对应的图案。因此DrawGrid()函数在绘制栅格ncUNDOWN的状态的依据，是该栅格是否pressed。

②在IniData()函数中将所有栅格的状态都初始化为ncUNDOWN的作用是，在后面布雷的时候，用来区分该栅格是否已经布雷。当布雷结束后，所有未布雷的栅格都会被重新赋值。不信，你回去再看一遍代码。

当然，布雷方法的构建，并不是唯一的，可以有其他方案的存在。

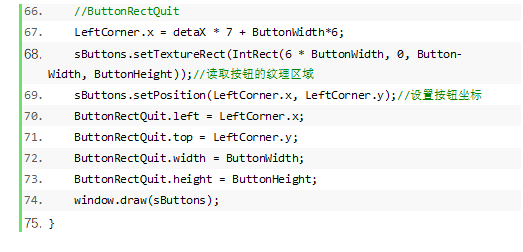
### 2. (简答题)

**2、在游戏绘制中，将多种素材以精灵表单的形式整合为一个纹理文件导入到纹理区，再由各个精灵对象在纹理图像的各个局部区域进行纹理提取、绘制。请问此种纹理管理方式有什么优势？**

*正确答案：*

答：SFML在绘制对象的时候，纹理需通过setTexture()函数与精灵对象产生关联，但精灵对象只获取纹理的地址并没做纹理的备份。精灵对象的每次绘制通过window.draw()实现，每次精灵绘制均会访问一次纹理。window.draw()的时间开销有部分即是来自纹理数据的传输。单个大纹理区的访问，纹理数据传输的频次，要低于多个小纹理区的纹理数据传输的频次。因此，单个大纹理区的访问效率要高于多个小纹理区的访问效率，有助于渲染效率的提升。

**3、下方代码中第67、68、72行代码中共出现4次ButtonWidth变量。请问每次该变量值起到的作用是什么？**

****

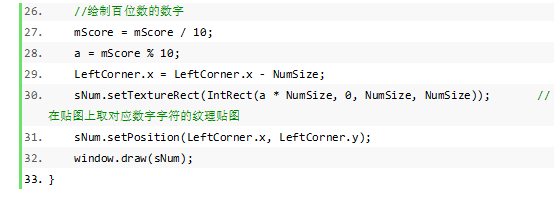
*正确答案：*

答：第67行的ButtonWidth变量是用于辅助定位按钮的左顶点x坐标值，比如此处表明按钮的左边存在7个空隙、6个按钮宽度值，暗示该按钮的左边存在有水平排列的其他6个按钮。

第68行的2处ButtonWidth变量值，第1处表示该按钮纹理在大纹理表单上是第7个纹理图案，左边需要偏移6个按钮纹理的宽度值，左顶点坐标取（6\*ButtonWidth，0）。第2处表示该按钮纹理在大纹理表单上提取的局部纹理的宽度为ButtonWidth。

第72行的ButtonWidth变量值，表示按钮在游戏窗口上呈现的宽度尺寸为ButtonWidth个像素，用于鼠标点击的交互区设置。

**4、下方代码中第27、28、29行代码的作用，请分别进行描述。第30行代码中，变量a数值的对应数字贴图能够准确获取的前提条件是什么？**

****

*正确答案：*

答：第27行代码，让mScore变量的数值以十进制的形式进行右移1位的操作；

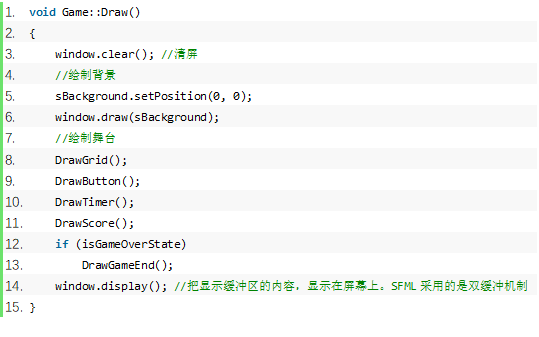
第28行代码，提取mScore变量数值当前个位上的数值；

第29行代码，左顶点坐标左移一个数字纹理宽度的位移，用于绘制高位数字的纹理贴图。

第30行代码起作用的前提是：①所有数字纹理贴图的尺寸一致，此处为NumSizexNumSize

②纹理贴图上的数字图案排放规则是按照0123456789的顺序依次水平排列

**5、游戏结束画面的绘制，可通过如下方式进行实现。如第12行代码所示，当变量isGameOverState为true则对结束画面进行绘制。但如果现在游戏分3个场景关卡，关卡变量为StageLVL，请用伪代码重新实现Draw()，使得程序能对指定StageLVL的内容进行绘制。**

****

*正确答案：*



### 1. (简答题)

**游戏过程中，如果游戏窗口发生尺寸变化，可能导致获取到的鼠标点击坐标出现偏差。请列举两种禁止游戏窗口大小缩放的方式。**

*正确答案：*

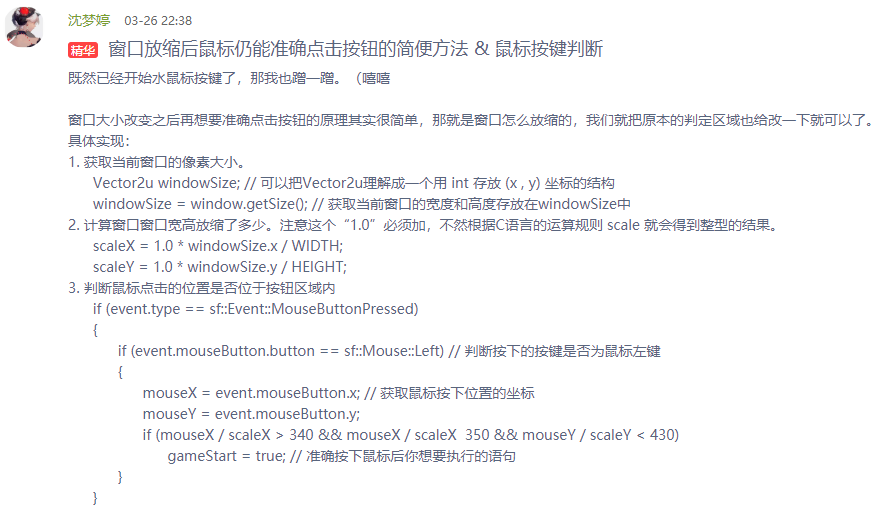
①通过sf::Event::Resized事件响应，将游戏窗口修改为原来尺寸；

②修改游戏窗口属性，禁用Resize设定

### 2. (简答题)

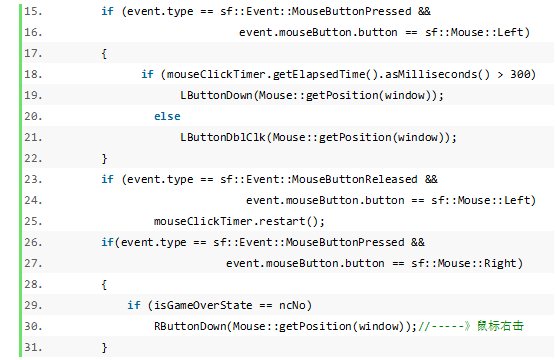
**游戏过程中，如果游戏窗口发生尺寸变化，可能导致获取到的鼠标点击坐标出现偏差。请设计一种解决方案使得游戏窗口发生缩放的时候，鼠标点击能够得到正常响应。**

*正确答案：*

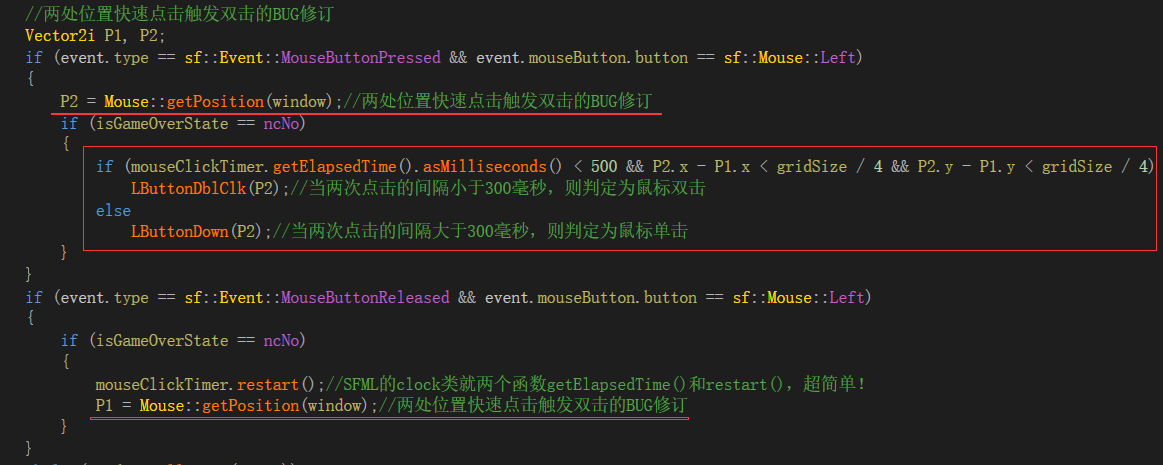


### 3. (简答题)

**下方代码，当快速在两处不同位置进行单击操作，会被误判为双击操作。请给出一种解决方案，以规避该种误判。**

****

*确答案：*



解决方案的核心是对于双击的条件进行更加严格的限定。

### 4. (简答题)

**在《扫雷》游戏中，如果第一下鼠标点击为右键单击，请问游戏能否正常进行？如果能请说明理由。如果不能请尝试给出修正方案。**

*正确答案：*

答：如果第一下鼠标点击为右键单击，则出现的情况是先插旗后布雷。因此在后续布雷的时候要注意到在布雷之前已经显示为旗的格子的状态。两个原则：①布雷时，不能改变已经插旗格子的当前显示状态；②插旗是玩家的一种判断，布雷的时候，要对当前格子的真实状态进行正常赋值，即旗的显示，不能影响当前格子真实状态产生，不然游戏会出BUG。

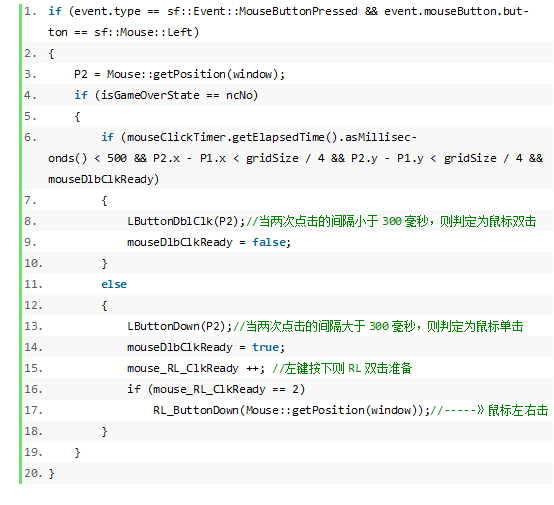
因此，需要三个状态变量，分别记录当前格的：①真实值；②显示值；③点击状态。②和③显示值和点击状态可以进行合并。

### 5. (简答题)

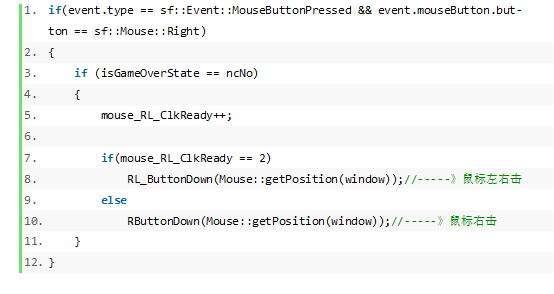
**经典扫雷游戏中，关于鼠标的双击方式有两种定义。一是鼠标左键快速点击两次，另一种是鼠标左右键同时按下。基于前面的内容，请思考下：鼠标左右键同时按下的双击，应该如何实现？**

*正确答案：*

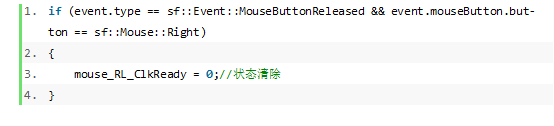
双键点击的状态管理，由于不确定左右两键哪个键会被先按下，所以它的管理比左键单键的双击管理稍微复杂一点。解决方案是将mouse\_RL\_ClkReady变量设置为int型，左右键的点击均会使得该变量+1，当该变量值为2则表示左右双键均以按下，可调用RL\_ButtonDown()函数进行双键响应操作。左键按下的RL双击准备及响应调用的示例如下所示



右键按下的RL双击准备及响应调用的示例如下所示。



通过前面两段代码，在Input函数中，不管左键先按，还是右键先按，只要按键累积次数达到2次（即mouse\_RL\_ClkReady=2），则调用RL\_ButtonDown()函数。当左右双键的任意键松开，则mouse\_RL\_ClkReady = 0，进行归零。下方代码为右键松开的代码示例。



### 1. (简答题)

**1、微软《扫雷》游戏的左右键双击操作是否允许点击在游戏舞台中未揭开的格子上？如果可以，请说明会发生的结果。若不可以，请说明理由。**

*正确答案：*

答：真的可以的。左右键同时按下后，点击格子的周围8领域中，未揭开的格子处于“待揭开状态”，当任意键松开之后，状态还原。

### 2. (简答题)

**2、阅读下方代码，并回答以下问题：**

**①请解释下第15行代码的判断条件是什么意思。**

**②微软《扫雷》游戏的左右键双击操作的规则是：如果左右双击点错了格子会将待揭开的格子状态还原，如果点对了则直接将对应格子揭开。第17-21行代码直接将相应格子状态还原，请问为什么？**

**③请解释下第24行代码的判断条件是什么意思。**

**④第24-25行代码直接调用鼠标左键的双击响应函数，请问会不会出现触雷的情况？**

****

*正确答案：*

答：①格子被标记为旗的同时，它本身是“雷”，同时满足条件的时候，才表明雷被找出来了。

②错了，会还原；对了则直接打开，那打开的时候，前面是否还原，其实无所谓。与其在后面再进行一次判断，不如这里顺手直接还原。让代码更精简一点。至少能省下2行for循环的代码。

此处，是为了引导大家思考一个问题？代码量优先？还是运算量优先？编译器里面其实是有对应参数设置，只是课上我们没说。

③在扫雷游戏案例格子值的枚举设置中数字1-8对应编号是3-10。也就是说枚举的值与实际真实数值存在偏差2.

④不会触雷。第24行代码的意思是，如果8领域中所有的雷都有找出来，那旗数flagNum+2 ，会与格子的真实值相等，此时调用双击函数快速打开未揭开的格子。雷没找对的情况下，是不会触发左键双击函数。

**3、鼠标左右键双击操作是由左键、右键的单击所构成。但微软《扫雷》游戏中鼠标左右键双击操作的优先级高于单击操作的优先级。即当判定为左右双键操作的时候，则不执行单键的响应。请结合下方代码，对左右键的按键设定进行梳理，以脑图的形式绘制左右键双击和各自单击的关系。（提示，要把按键的“按下”和“释放”两种状态纳入做统一考虑）**

****

*正确答案：*

答案：略，

左键的按下和释放基本就参照题中代码的逻辑执行。右键的响应请放在按键释放时候执行，左右双击的响应发生在“按下”状态，如果已发生，其实已经不存在右键点击的前提。

另外，再好好思考一下，左右键双击的执行逻辑，尤其其中的逻辑锁设定

**4、案例中undownOpen()函数的功能设定是什么？它与unCover()函数的功能有没重叠？**

*正确答案：*

答案：undownOpen()函数：游戏胜利后，将所有未揭开格子进行揭开

unCover()函数：游戏失败后，将所有未揭开的雷显示出来

**1、如果要将void RButtonDown(Vector2i mPoint); void LButtonDown(Vector2i mPoint); void LButtonDblClk(Vector2i mPoint)；3个函数放置在Logic()函数中进行响应，程序的框架结构应该进行如何改动？**

*正确答案：*

答：①首先创建变量mouseAction在Logic模块中管理当前要激活哪个鼠标响应函数，同时记录鼠标点位置；

②创建mouseFunction枚举类型

typedef enum MOUSEFUNCTION

{

RButtonDownFunc;

LButtonDownFunc;

LButtonDblClkFunc

};

③将原来Input模块中，调用鼠标响应函数的地方，全部改用mouseAction先记录需要响应的鼠标函数；例如用调用RButtonDown()的地方，用mouseAction=RButtonDownFunc;进行替代。

④在Logic中，构建鼠标响应的代码段，或进行函数封装。主体内容用switch(mouseAction)进行分类

switch(mouseAction)

{

case RButtonDownFunc:

RButtonDown(Vector2i mPoint);

break;

……

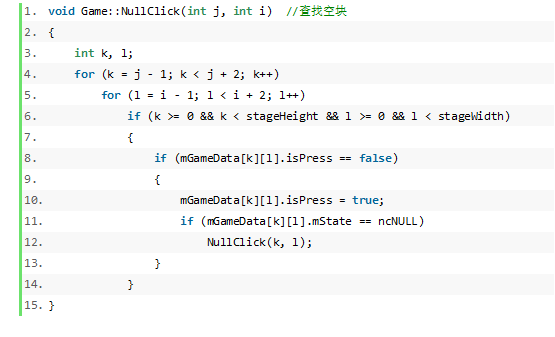
}

**2、为提升玩家的游戏体验，《扫雷》游戏采用了哪些提升游戏节奏的功能设置？**

*正确答案：*

答：鼠标双击响应，快速揭开周边8领域中未揭开的格子；击空NullClick响应，通过递归调用，快速揭开一片非雷的格子；左右键同时按下，允许玩家进行安全的双击测试。

**3、请对下方代码中第4-14行代码进行逐行注释。**

****

*正确答案：*

答：

第4-5行：对鼠标点击位置(j,i)的周边8领域进行遍历

第6行：判段(j,i)周边8领域是否都在舞台区域，只对舞台区域的格子进行响应

第8行：位置(k,l)的格子是否未揭开

第10行：将位置(k,l)的格子标记为揭开状态

第11行：位置(k,l)格子的状态是否为“空”

第12行：以位置(k,l)格子为中心，调用NullClick()函数。

**4、isWin()函数第21-22行代码的if判定条件有否存在漏洞，它能否判断剩下未点击的块是不是都是雷？请给出你的理由。**

****

*正确答案：*

答：可以的。在游戏未结束的情况下，只有雷全被找出，才会被判定游戏胜利。

采用反证法。设：

总雷数mMineNum

旗子数mFlagCalc

未揭开格子unPress

假设前面有旗子误标，也就是标为“旗”的格子实际不是“雷”，那该“雷”必然在剩下未揭开的格子里。那剩下未揭开的格子数必然大于总雷数减“旗”子数。

未揭开格子unPress>总雷数mMineNum - 旗子数mFlagCalc

统计的c值 = 未揭开格子unPress + 旗子数mFlagCalc > 旗子数mFlagCalc

由于C值必然大于总雷数，第26行代码条件无法满足，游戏无法结束。

因此，只有在Flag全部判定正确的情况下，才会可能使得第26行代码条件被满足。当然，这个时候需要非雷的格子全部被揭开。

综上所述，该判定能够判断剩下未点击的块是不是都是雷