第14天作业-生命游戏

提前布置这个作业，是想让大家知道完成两周的培训后，自己的编程能力能到达什么水平，并增加趣味性，相信大家能从这次的作业中收获更多（笑）。 - 李优

本次作业要构建一个名为“康威生命游戏”（Conway’s Game of Life）的模拟器，并使用它来模拟人类的繁衍过程，因为模拟需要在一个布满细胞的二维网络上进行，所以这次实验建议使用面向对象思想编写，CPP建议使用std::string 或 std::vector<char> 模拟地图，Python建议使用切片（slice）。

网格中的每个细胞在水平、垂直和对角线方向上总共有8个相邻细胞。在每一世代，单个细胞的生死存亡将取决于相邻细胞的存活数量。

# 开天辟地

在初次实现生命游戏时，我们需要将世界限制在固定的大小之内。具体来说，我们需要决定网格的尺寸并定义相应的常量（建议长宽为 80 \* 15，也可自行定义）。

接着还需要Universe 类用于持有二维细胞网络，并通过单个字符 ’\*’（星号） 和 ’ ’（空格）分别表示细胞的存活和死亡。

通过使用不定长的数组来表示世界，可以更方便地共享和修改世界（当然你也可以通过指针实现）

在此之后，我们还需要在 Universe 类的构造函数中给对象分配一个 height 行 width 列的 Universe，并将每个元素设置为 ‘ ‘ 或表示死亡的细胞。

# 观察世界

给 Universe 内添加一个Show函数，它能够使用 print（python）或 std::printf, std::cout （CPP）来将世界目前的状态打印至屏幕，其中存活的细胞用星号表示，而死亡的细胞用空格表示。此外它还需要在每次打印完一行细胞之后，将光标移动至新的输出行。

请在 main 函数中调用 Universe 的构造函数创造出新世界，然后调用 Show 函数把这个世界打印出来，在继续作业之前，请确保你的程序能正常运行，即使整个世界还没有任何细胞。

# 激活细胞

给 Universe 内添加一个 Seed 函数，它可以随机激活世界中约 25% 的细胞 (将对应元素的值设为存活)。实现时可以使用各自语言自带的 rand 库生成随机。

在此之后，请修改 main 函数并使用 Seed 函数对世界进行激活，然后使用 Show 函数将激活后的世界打印出来。

# 适者生存

以下是康威生命游戏的具体规则：

1. 当一个存活细胞邻近的存活细胞少于 2 个时，该细胞死亡。
2. 当一个存活细胞邻近有 2 个或 3 个存活细胞时，该细胞将延续至下一世代。
3. 当一个存活细胞邻近有多于 3 个存活细胞时，该细胞死亡。
4. 当一个死亡细胞邻近正好有 3 个存活细胞时，该细胞存活。

为了实现这些规则，，我们需要将它们分解成三个步骤，，并为每个步骤实现相应的函数：

1. 判断细胞是否存活的函数。
2. 统计邻近存活细胞数量的函数。
3. 判断细胞在下一世代存活或死亡的函数。

To be or not to be，判断细胞是否存活可以通过检测 Universe 内的对应元素的字符值来实现，只要该值为 ‘\*’，那么细胞就是存活的。请按以上要求为 Universe类编写一个 Alive 函数。

实现Alive函数最困难的就是处理边界情况，这个交给大家自由发挥（难点）。

统计邻近细胞的Neighbors函数，为了处理边界情况，请使用 Alive 函数而不是直接访问世界数据。

游戏逻辑实现，在实现了统计邻近存活细胞数量的函数后，我们就可以正式在 Next函数里实现游戏规则了。这个函数不会直接修改世界，而是会返回一个 bool 值，并以此来表示给定细胞在下一世代的存活或死亡。

# 平行世界

为了完成模拟操作，程序需要遍历世界中的每个细胞，并使用 Next 函数判断它们在下一世代中的状态。

这里有一个严重的问题，那就是统计邻近细胞必须基于世界先前的状态，如果程序在执行统计的同时直接修改世界，那么这样必定会对邻近细胞的统计结果造成影响。

最简单的办法就是创建两个世界，一个保存前一状态，一个保存当前状态，在读取之前状态世界同时修改当前状态世界。编写Step函数来实现迭代，当所有细胞迭代完成后，就可以交换两个世界，开始下一次迭代。（基础分）

请大家想想办法来节省内存消耗，用更短的时间和空间来实现迭代函数。（加分点）

在展示新世代的细胞之前，程序需要使用特殊的ANSI转义序列“\033[H”来清空屏幕（其他方法，只要能清空屏幕当前输出，也行）。在此之后程序就可以打印出整个世界，可以使用各自语言里关于 time 的库来实现每秒迭代次数（建议每秒5次，太快看不清）。

到这里，程序的编写就完成了，但是请仔细测试，想办法减少程序开销，经测试更优的程序得分更高。注意：只是按照上述操作编写只有基础分，请持续优化程序，减少其开销，增加程序最终得分。

那么恭喜你完成了本次作业以及本次培训，你现在已经可以完整编写一个程序模块了（笑）。请及时提交所有培训作业。

附加题（对应CPP 超级加倍部分，python和电控可不用做）：

Modern C++ 的一道题在生命游戏基础上进行，要求使用Modern C++ 的语法和思想对以上程序进行进一步优化，要求必须使用C++11的多线程分解任务计算。