2023 秋季程序设计 Project 文档

生命游戏

一、 背景介绍

约翰·康威 (John Conway) 于 1970 年发明的“生命游戏”(Game of Life) 是一个零玩家“游戏”，也是冯诺依曼提出的元胞自动机的其中一种。游戏由一个无限延伸的二维矩形世界组成，这个世界中的每个方格居住着一个活着的或死了的细胞。游戏描述了细胞如何一代一代进化的规则，它根据当前一代中相邻细胞的状态，计算出下一代细胞的状态。在二维世界中，单元格的邻居是与该单元格垂直、水平或对角线相邻的 8 个单元格（若边缘格点没有 8 个邻居也按如下规则进行）。康威的一套规则可以概括为:

1. 任何活格点，若邻居活格点数 <2，则该格点在下一世代死亡
2. 任何活格点，若邻居活格点数 $=2$ 或 $=3$，则该格点在下一世代存活
3. 任何活格点，若邻居活格点数 >3，则该格点在下一世代死亡
4. 任何空格点，若邻居活格点数 $=3$，则该格点在下一世代变为活格点

在这个 project 中，我们将实现康威的生命游戏，有一个小小的限制，即我们的二维世界是有限的。你可以在维基百科 Conway’s Game of Life 上阅读更多关于康威的生命游戏的信息。

演化示例：

二、 项目目标

本项目中，你需要使用 C 编程语言，结合课堂知识、lab 内容，实现一个简单的、可以在命令行下交互的生命游戏，并按要求编写一份开发文档，介绍你的实现方式、开发思路等内容。

你需要实现的功能可分为两大类：

1. 用 C 代码表示生命游戏，实现其核心规则
2. 附加功能

此外，我们还会给出你在代码实现过程中需要遵循和注意的业务逻辑，帮助你更好地设计你的 project。

三、 项目实现

3.1 生命游戏核心规则

3.1.1 欢迎界面

展示基本信息和操作说明，你可以尽量设计得好看一些，下图是一个基本版本，也是你需要实现的基本功能。

```
---------------------------------------------------------------------
i <row> <col>    initialize an empty grid with <row> rows and <col>
                 columns
r                randomly set alive/dead states for all grids
n                evolve into next generation
c                automatically evolve, until receiving 'b' command
b                pause evolution
s <path>         save grid states to file <path>
l <path>         load grid staees from file <path>
q                quit
h                help
---------------------------------------------------------------------
```

3.1.2 初始化格点

运行生命游戏，程序需要提供细胞图初始化选项。玩家可以随意设计二维空间的大小。例如输入 i 20 20 回车：

```
| | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | |
------------------------------------------------------------------
i <row> <col>    initialize an empty grid with <row> rows and <col>
                 columns
r                randomly set alive/dead states for all grids
n                evolve into next generation
c                automatically evolve, until receiving 'b' command
b                pause evolution
s <path>         save grid states to file <path>
l <path>         load grid staees from file <path>
q                quit
h                help
------------------------------------------------------------------
```

3.1.3 随机化格点

按照演化规则，空旷的格点不会发生任何事情。为了方便我们玩这个生命游戏，你需要提供一个随机化格点的方法，像女娲一样随机地创造一些生命。你需要随机地将每个格点上的状态设置为生或死，这样，你的格点才有可能按照规则动起来。基本规则下，每个格点有 0.5 的概率为生，0.5 的概率为死（当然概率的数值不强求，设置一个你喜欢的值也可以）。例如在上图基础上再输入 r 回车：

```
| |*| |*|*| | |*|*|*| |*|*|*| | | | | | |
|*| | | | | |*|*|*|*| |*| | | |*|*|*|*| |
|*|*|*|*|*| | | |*| |*|*|*|*|*| | | |*| |
| | |*| | |*| | |*|*|*|*| | |*| | |*|*| |
|*| |*|*|*| | | | |*|*|*|*| | |*| | |*|*|
| |*| |*|*| | | |*|*| |*|*|*| | |*| | |*|
|*| | |*|*|*|*| | | |*| |*|*| | |*|*|*| |
| | |*| | | |*| |*|*| |*|*| |*|*| |*|*| |
|*| |*|*|*| |*| | |*| |*| | | |*|*|*|*|*|
| | | | |*|*| | |*| | | |*|*|*| | | |*| |
| | |*|*|*| |*|*|*|*| | | |*| |*| |*| | |
|*| |*| | |*| | |*|*| | |*|*|*|*|*|*|*| |
| |*|*|*| | |*|*|*| | | |*|*|*|*|*| |*|*|
|*|*|*| |*|*| |*| | | |*|*|*| | | | |*|*|
|*| | | | |*| | |*| |*|*|*|*| |*| | |*|*|
|*|*| |*|*| | |*| | |*|*|*| | | | |*|*| |
|*|*|*| | |*| |*|*| |*| | |*|*| |*|*| | |
| |*|*|*| | | |*| |*|*|*|*|*| | |*|*| | |
| |*| | | |*|*|*|*|*|*|*| |*| | | |*| | |
|*| |*| | | |*| | |*|*|*|*|*| |*| |*| | |
------------------------------------------------------------------
i <row> <col>    initialize an empty grid with <row> rows and <col>
                 columns
r                randomly set alive/dead states for all grids
n                evolve into next generation
c                automatically evolve, until receiving 'b' command
b                pause evolution
s <path>         save grid states to file <path>
l <path>         load grid staees from file <path>
q                quit
h                help
------------------------------------------------------------------
```

3.1.4 演化

你需要根据演化规则，将格点的状态更新到下一个世代，你需要计算所有格点在下一个世代的状态，并更新当前格点的状态。按下 n 回车：

```
| | | | | | |*| | |*| |*|*| | | |*|*| | |
|*| | | | |*|*| | | | | | | |*|*|*|*| |
|*| |*|*|*|*|*| | | | | |*|*| | | | |*|
|*| | | | |*| | |*| | | | | |*|*| |*| | |
| | | | | |*| | | | | | | |*|*|*| | |*|
|*|*| | | | | | |*| | | | | |*|*|*| | |*|
| |*| | | | |*| | | | | | | | | | | |*|
| | |*| | | |*| |*|*| | | | |*| | | | | |
| |*|*| |*| |*| | |*| |*| | | | | | | |*|
| |*| | | | | | | | |*| |*|*| | | | | |*|
| |*|*| | | |*| | | | | | | | | | | | | |
| | | | | |*| | | | | | | | | | | | |*|
| | | | | | | | | |*| | | | | | | | | |
|*| | | |*|*| | | |*|*| | | | | |*| | | |
| | | | | |*| |*|*|*| | | | |*| | | | | |
| | | |*|*|*| |*| | | | | | |*| | | |*|
| | | | | | | |*| | | | | | |*|*| | | | |
| | | |*|*|*| | | | | | | | | |*| | |*| |
|*| | |*| |*| | | | | | | | | | | |*|*| |
| |*| | | |*|*| | | | | | |*|*| |*| | | |
```

3.1.5 保存状态到文件

你需要实现将格点保存到文件的功能，在基本的生命游戏中，你需要按照以下命令保存格点状态：s <path> （将<path>替换为保存文件的路径）：

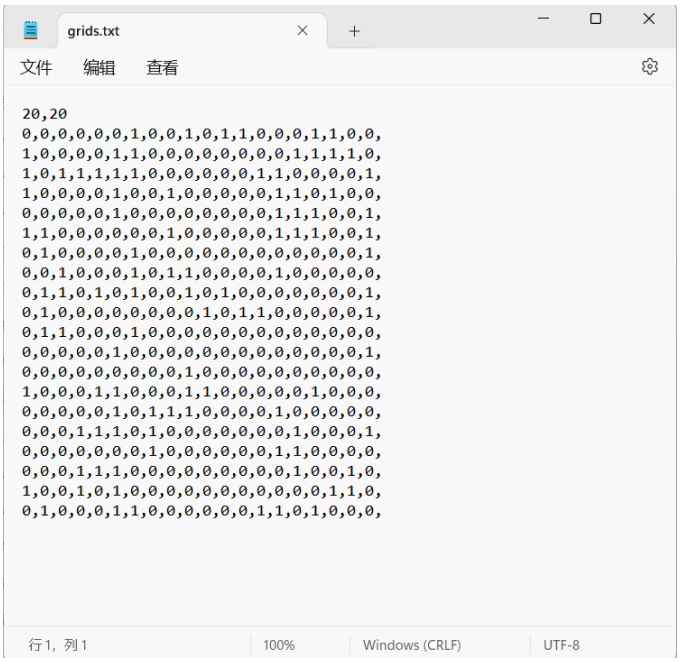
```
--------------------------------------------------------------------
i <row> <col>    initialize an empty grid with <row> rows and <col>
                 columns
r                randomly set alive/dead states for all grids
n                evolve into next generation
c                automatically evolve, until receiving 'b' command
b                pause evolution
s <path>         save grid states to file <path>
l <path>         load grid staees from file <path>
q                quit
h                help
--------------------------------------------------------------------
s grids.txt
```

提示：

文件中的保存格式参考如下：

1. 第一行保存行数和列数，用英文逗号分隔；
2. 余下若干行保存格点状态，一行之内每个格点的状态数据用英文逗号分隔

例如，上面所示的格点保存到文件 grids.txt 后应当有以下内容（但对于附加内容的状态保存我们不做格式上的要求）：



3.1.6 从存盘文件中初始化世界状态

前面要求的初始化功能只初始化为空旷格点。你还需要实现从文件中读取数据并初始化格点的功能。例如，根据导入文件 grids.txt 初始化，请设置你的输入格式如下图

```
--------------------------------------------------------------------------
i <row> <col>    initialize an empty grid with <row> rows and <col>
                 columns
r                randomly set alive/dead states for all grids
n                evolve into next generation
c                automatically evolve, until receiving 'b' command
b                pause evolution
s <path>         save grid states to file <path>
l <path>         load grid staees from file <path>
q                quit
h                help
--------------------------------------------------------------------------
l grids.txt
```

加载后：

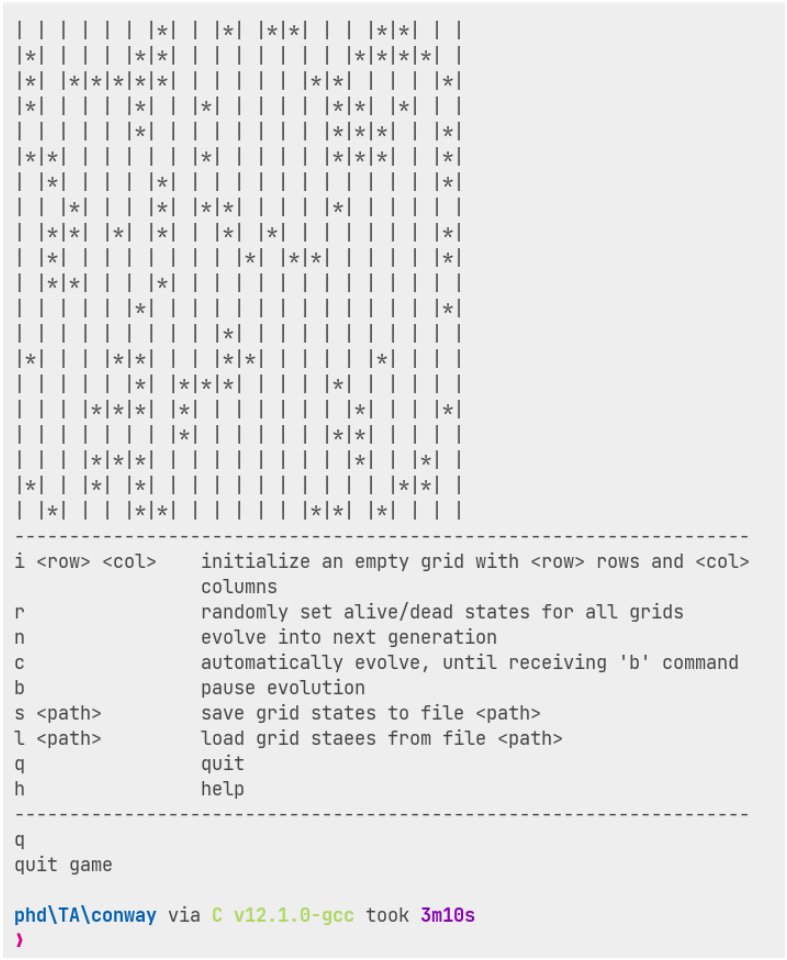
```
| | | | | | |*| | |*| |*|*| | | |*|*| | |
|*| | | | |*|*| | | | | | | |*|*|*|*| |
|*| |*|*|*|*|*| | | | | |*|*| | | | |*|
|*| | | | |*| | |*| | | | | |*|*| |*| | |
| | | | | |*| | | | | | | | |*|*|*| | |*|
|*|*| | | | | |*| | | | | |*|*|*| | |*|
| |*| | | | |*| | | | | | | | | | | |*|
| | |*| | | |*| |*|*| | | | |*| | | | | |
| |*|*| |*| |*| | |*| |*| | | | | | | |*|
| |*| | | | | | | |*| |*|*| | | | | |*|
| |*|*| | | |*| | | | | | | | | | | | |
| | | | | |*| | | | | | | | | | | | |*|
| | | | | | | | | |*| | | | | | | | | |
|*| | | |*|*| | | |*|*| | | | | |*| | | |
| | | | | |*| |*|*|*| | | | |*| | | | | |
| | | | |*|*|*| |*| | | | | | |*| | | |*|
| | | | | | | |*| | | | | | |*|*| | | | |
| | | |*|*|*| | | | | | | | | |*| | |*| |
|*| | |*| |*| | | | | | | | | | |*|*| |
| |*| | | |*|*| | | | | |*|*| |*| | | |
----------------------------------------------------------------------
i <row> <col>    initialize an empty grid with <row> rows and <col>
                 columns
r                randomly set alive/dead states for all grids
n                evolve into next generation
c                automatically evolve, until receiving 'b' command
b                pause evolution
s <path>         save grid states to file <path>
l <path>         load grid staees from file <path>
q                quit
h                help
----------------------------------------------------------------------
```

3.1.7 其它

你还需要实现打印帮助列表的功能，输入 h 应该列出如下内容

```
----------------------------------------------------------------------
i <row> <col>    initialize an empty grid with <row> rows and <col>
                 columns
r                randomly set alive/dead states for all grids
n                evolve into next generation
c                automatically evolve, until receiving 'b' command
b                pause evolution
s <path>         save grid states to file <path>
l <path>         load grid staees from file <path>
q                quit
h                help
----------------------------------------------------------------------
```

你需要实现退出功能，输入 q 应该退出程序：




```
------------------------------------------------------------------------
i <row> <col>    initialize an empty grid with <row> rows and <col>
                 columns
r                randomly set alive/dead states for all grids
n                evolve into next generation
c                automatically evolve, until receiving 'b' command
b                pause evolution
s <path>         save grid states to file <path>
l <path>         load grid staees from file <path>
q                quit
h                help
------------------------------------------------------------------------
q
quit game

phd\TA\conway via C v12.1.0-gcc took 3m10s
❯
```

图中列出的自动演化和暂停功能（c 和 b）不需要你去实现，脚手架代码里面已经实现了这两个功能。

3.2 附加规则

以下规则你可以选择性的完成一部分或全部，当然进一步地，你也可以有其他的创新功能，但需要在你的产品文档中描述清楚。如果你不确定你的创新是否合理以及符合课程项目要求，建议与老师和助教交流确定。

细胞们发现这个神秘的 2D 世界中似乎存在着一种神奇的能量，被称为食物资源。食物资源以某种不可思议的方式在地图上随机出现，每个位置最多只能有一个食物资源。

食物资源是细胞们生存和繁衍的关键。当细胞与食物资源在同一位置时，它可以通过吸收和利用食物资源来获取能量，从而延续自己的生命。当细胞发现周

围有食物资源时则会争夺它们的控制权。然而，在多个细胞争夺同一个食物资源的情况下，只有一个细胞能够获胜并成功地消耗它。

具体规则如下：

1. 每个回合开始时，食物资源随机出现在地图的某些位置，每个位置最多只能有一个食物资源。食物资源总量不变，为地图的边长/2(向上取整)。
2. 如果一个细胞与食物资源在同一位置，该细胞就消耗了这个食物资源。
3. 如果一个细胞的邻居为食物资源，则该细胞可以移动至食物资源位置并消耗该食物资源。
4. 多个细胞争夺同一个位置的食物资源时，随机选择一个细胞消耗食物资源。
5. 消耗了食物资源的细胞获取特殊能力，在下一轮进化中不受 Conway 规则的影响

示例：

1. 情况一：每个回合开始时，食物资源随机出现在地图上的某些位置。例如，地图边长为 10，那么食物资源总量为 5， 随机位置(2,3)，(3,1)，(4,4)，(5,7)，(8,9)出现食物资源。
2. 情况二：一个细胞和一个食物资源在同一位置。在这种情况下，细胞直接消耗食物资源。例如，如果细胞 A 在位置(2,2)，食物资源也在位置(2,2)，那么细胞 A 会消耗这个食物资源。
3. 情况三：如果一个细胞的邻居为食物资源。在这种情况下，如果细胞在下一次演化中不会死亡，那么细胞将随机选择移动至食物资源位置并消耗资源或留在原地。如果细胞在下一次演化中会死亡，则一定会移动到食物资源位置并消耗资源。例如，如果细胞 A，B，C 分别在位置(2,1),(2,2),(3,2)，食物资源在位置(1,1),(1,3)，细胞 B 在下一轮进化中不会死亡，因此细胞 B 随机选择留在原地或向位置(1,3)移动并消耗资源，而细胞 A 一定会向位置(1,1)移动并消耗资源，否则在下一次演化中 A 一定会死亡。

4. 情况四：多个细胞争夺同一个位置的食物资源。在这种情况下，随机选择一个细胞移动到食物所在位置消耗食物资源。例如，如果细胞 A、细胞 B 分别在位置(2,1)，(2,3)食物资源位置为(2,2)，那么我们随机选择细胞 A 或细胞 B 来消耗这个食物资源。
5. 情况五：消耗了食物的细胞获得特殊能力，如情况三中的细胞 A，在下一轮进化中，即使周围邻居的环境不满足 Conway 规则，细胞 A 也会存活。

3.3 项目实现中的提示

本次 project 中，助教会提供一个代码脚手架，你需要使用给定的函数实现以上功能，可以做适当修改，但基本框架要和给出的代码一致。除此之外，我们也会对一些逻辑实现进行提示，请你仔细阅读，并保持良好的程序设计习惯，助教将会对代码逻辑进行评分。

3.3.1 格点的表示

基本的生命游戏是在二维场景下的格点模型，你需要用二维数组来表示格点。每个格点有生或死两种状态，在代码中你可以用数字 1 和 0 来分别表示格点的生或死。

如果要进一步添加附加功能，一个格点的状态可以不单单只有生或死，可以有**食物，空，普通活细胞，强化活细胞**等状态，这个时候可以将表示一个格点状态的类型从 0, 1 值变成枚举类型(enum)或者宏定义的形式。

3.3.2 初始化格点

程序需要能够根据指定的格点行数和列数初始化特定大小的格点，表示格点的二维数组大小将由创建格点的使用者决定，在运行程序之前，我们不知道使用者想要创建多大的格点。为此，你需要能够动态地根据指定的行列大小创建二维数组。一般的二维数组不能动态指定大小，你需要使用二级指针代替二维数组，并使用 **malloc()函数**分配内存

初始化的格点应当是完全空旷的，即所有格点都是空着的。虽然初始化一个什么都没有的格点看上去没有任何意义，但其实这一步为舞台筹备了内存资源，可谓盘古开天辟地。

3.3.3 演化

你需要根据演化规则，将格点的状态更新到下一个世代，就如背景当中的示例一样。你需要计算所有格点在下一个世代的状态，并更新当前格点的状态。请注意：

1. 不要边计算边更新，要计算完之后再一起更新；
2. 你需要再分配一个二维数组，以存储下一个世代的格点；
3. 原来格点的内存需要被回收掉，否则会发生内存泄露。

3.3.4 获取格点当前状态

不论是演化还是打印，我们都需要频繁地根据格点坐标获取格点的状态，为此你需要实现一个简单的访问格点状态的功能。

你可能会觉得这个功能有些多余，毕竟我们当然可以用数组下标访问的形式来获取状态。但请注意，你的下标可能会超出数组的范围，而且你表示格点的指针可能还是没有经过初始化过的空指针，因此随意地访问不光会导致未知的行为，还会有重大的安全隐患。封装好的安全访问状态的函数在软件工程中是非常必要的。

3.3.5 获取格点下一个状态

与获取格点当前状态类似，在演化更新格点时，我们需要大量地计算某个坐标的格点在下一个世代的状态。你需要实现一个简单的计算格点下一个状态的功能。

3.3.6 删除格点

我们创建格点时分配的内存需要回收掉，为此你需要实现专门的删除格点功能，以在我们不需要格点的时候收回分配的内存。你需要使用 free()函数。

以上几点是数据逻辑上的注意事项，如果你在开发过程中有更多的体会，请记录到你的开发文档中，作为你的开发过程。

四、 项目评分

本项目满分 100 分，包括三个部分：基础功能部分、综合评价部分和附加分值。其中，基础部分 55 分，综合评价部分 30 分，附加分值为 15 分，实现 2 个附加功能并正确描述在开发文档中即可获得相应分值。其中自己设计的创新功能通过老师或助教认可后会在原得分的基础上附加不超过 5 分，且总分不超过 100 分。

基础功能部分	
初始化格点	5 分
随机化格点	10 分
演化	10 分
保存状态到文件	7 分
从存盘文件中初始化世界状态	8 分
其他（如帮助和退出程序）	5 分
游戏运行总体正常，符合相关描述，不会异常退出	10 分
综合评价部分	
设计文档（包括但不限于程序结构设计与分析，主要函数的功能，简要描述如何使用你的程序，编程中遇到的问题和解决策略）	8 分
代码结构（根据脚手架实现的情况）	5 分
代码风格（包括但不限于命名规范，缩进与换行，代码可读性）	4 分
程序鲁棒性（对于玩家的非法输入，能够正确地处理和给予提示。比如玩家输入错误的初始化细胞图名称，不按照程序的选项引导进行输入等）	3 分
面试情况（能否清晰地解释程序结构，能否回答助教的问题等。如果面试过程中发现抄袭，将会直接判定不及格甚至 0 分）	10 分

附加功能	
实现 1 个附加功能	6 分
实现 2 个附加功能	6 分
附加创新功能（酌情加分，但总分不会超过 100 分）	5 分（附加）
附加功能的设计说明	3 分
总分	100 分

五、 提交

1. 提交物：将源代码与设计文档打包成压缩文件（格式为 zip 或 rar），命名为学号_姓名_pj（如 20302010000_王明.zip_pj），作为提交物。
2. 提交：提交至超星学习通对应的作业中。
3. 截止时间：12 月 28 日 23:59

六、 注意事项

1. 注意自己的代码风格。
2. 合理安排时间，尽早动手，不要拖到最后。
3. 认真做好每个功能点，特别是基础功能的实现。
4. 超时处理：无特殊情况，超时后助教会进行适当扣分。每超过一天扣掉课程项目分数的 10%，直到扣完为止，或者到课程成绩提交截止日完全扣完。
5. 欢迎同学们相互讨论，但抄袭是严格禁止的。一旦发现抄袭行为，我们将会根据抄袭量对抄袭者和被抄袭者都酌情扣分（甚至直接给予 0 分）。