项目准备活动 3 [30 分]

观看简报视频！ **不是自动评分评估**

您必须在实验 3 的截止日期之前将所有源代码提交到 Git 存储库。

重要的： **您提交的内容必须经过编译且没有语法错误才能获得成绩。不可编译的解决方案将不会被评分。**

关于 PPA 3

- 使用以下链接接受 PPA3 邀请：https://classroom.github.com/a/RZhN7P8Y PPA 3 代码提
- 交截止日期为**2023 年 11 月 10 日星期五晚上 11:59。**
- PPA 3 现场演示应在代码到期日后您指定的实验室日进行。

关于一般项目准备活动

- 非自动评分的编程活动有助于积累完成 COMPENG 2SH4 课程项目所需的知识和编程经验。

-评估形式

哦 讲座中将提供更多说明。
哦 代码提交截止日期后，在您指定的实验室会议中进行 5 分钟的基于功能的演示。知识评估将在指定日
哦 期的现场交叉检查中进行（关注公告）。

随机数生成简介

在 PPA3 中，我们将使用迄今为止在 2SH4 中学到的知识来创建我们的第一个可玩游戏。尽管如此，如果没有不可预测性，游戏就不会具有娱乐性。因此，让我们学习如何在 C 中生成随机整数来为我们的程序增添趣味。

1. 伪随机整数生成函数 –整数兰特（**空白**）

源库　　标准库文件

功能性　　预编程的非连续整数表存储在标准库，值范围从 0 到兰德_最大值（通常> 32767）。每次调用兰特（）返回表中的下一个整数。由于这些整数是非连续的，因此返回值兰特（）将要*似乎是随机的*。 兰特（）每次程序启动时都会在伪随机数表中返回相同的条目，因为

局限性

表中的初始读出位置始终为 0th。换句话说，兰特（）仍然是可以预见的。

用法：　　int myRandomNumber = rand();
// 从表中获取一个伪随机数

2. 播种随机数生成 –空白斯兰德（**无符号整数**数）

源库　　标准库文件

功能性　　改变初始读出位置兰特（）与种子号编号，从而使伪随机数的生成变得更不可预测。该函数不返回任何内容。

用法：　　兰德(100);　　// 为随机数生成器播种

int myRandomNumber = rand();
// 从表中获取*更好的*伪随机数

3. 伪随机播种

对于 srand()，我们仍然需要克服另一个限制——可预测的播种，从而可预测生成的数字。为了增加另一层不可预测性，让我们播种**当前时间**， 使用时间（）函数来自时间.h.time() 返回自 1970 年 1 月 1 日以来的秒数。

改进的伪随机数生成

```
srand(时间(NULL)); // 带时间的种子()

int myRandomNumber = rand();
// 获得一个更加随机的伪随机整数
```

您可以在程序中重复使用不同的时间戳进行播种，以获得更加随机的伪随机数生成。

最后，如果我们要生成范围之间的随机整数（**0、INTMAX]**，我们可以调用兰特（）这边走：

```
int myRandomNumber = rand() % INTMAX;
```

仔细考虑模数运算符如何提供我们所需的范围限制功能。

基本碰撞检测简介

在PPA3中，我们需要判断玩家可控制的物体是否与游戏板上的某个物品重叠，然后在发生碰撞时确定相应的动作。好消息是我们已经在 PPA2 中制定了一个非常有效的框架来简化碰撞检测。

回想一下 PPA2 中定义的对象位置结构。

```
结构体对象位置
{
    整数X;                  // 对象的 x 坐标 // 对象的 y 坐标
    整数y;
    字符象征;               // 要在屏幕上绘制的对象的 ASCII 符号
};
```

假设我们现在有这个结构的两个实例：player 和 item。然后，确定它们是否发生碰撞就像在每个程序循环中检查它们的 x 坐标和 y 坐标是否相同一样简单***后***玩家移动完成。

```
if(item.x ==player.x && item.y ==player.y) {

        // 碰撞发生了！采取一些相应的行动...
}
```

PPA3 要求您跟踪游戏板上至少 5 个随机生成的项目。您必须考虑一种巧妙的方法来存储可收集物品对象的 5 个实例，并在每个程序循环迭代中检查它们与玩家对象的碰撞。

项目准备活动 3 – 创建寻宝游戏“猜单词”

PPA2 中完成的工作为 PPA3 奠定了基础。现在，我们可以向现有的 PPA2 工作添加一些有趣的功能，并制作一个简单的寻宝游戏 - 请参阅示例可执行文件。

基本的游戏机制很简单：

- 游戏有一个内置的目标字符串，包含 15-25 个 ASCII 字符，隐藏在远离玩家的地方。启动时
-
    哦 从目标串中随机挑选 5 个角色并随机放置在游戏板上。
    哦 由未揭示的字符“？”组成的神秘字符串 显示在游戏板的底部，指示要收集/显示以匹配目标字符串的字符数量。
    哦 移动计数也显示在游戏板的底部，指示自游戏开始以来玩家在游戏板上走过的距离（游戏板块的数量）。
- 玩家必须使用 WASD 控制来控制玩家对象收集棋盘上的 ASCII 字符。
    哦对于每个收集的角色
        - 它的出现将在神秘弦中揭晓；未收集的字符将保留为“？”。
        - 棋盘上剩余的物品将被移除，另外 5 个从目标字符串中随机选择的角色将被生成并随机放置在棋盘上。
- 游戏的目标是通过揭示神秘字符串中与目标字符串匹配的所有字符，消耗最少的移动次数来赢得游戏。

添加到您现有 PPA2 工作中的附加功能包括：

- 部署一个恒定的目标字符串（隐藏），
- 部署神秘弦（如图所示）**在堆上实例化。**
- 部署一个 Move Count（或者，只是程序循环迭代的次数），在任何地方实例化。一组 5 个
- 随机生成的收藏品，**在堆上实例化。**
    哦 使用先前讨论的伪随机整数生成方法。
    哦 生成的项目***不得重叠***游戏板边界或玩家对象。
- 部署各自的集合检测逻辑和角色揭示逻辑。
    哦 当玩家与棋盘上的物品碰撞时，显示神秘字符串中相应的 ASCII 字符（如果尚未显示）。

    哦 然后，删除板上现有的项目并在板上生成另外 5 个项目。
- 部署游戏结束条件。
    哦 当神秘绳完全显露并与目标绳匹配时，结束游戏并显示获胜消息。

    哦 当按下退出键时，结束游戏而不显示任何获胜消息。
- 确保在关闭程序之前清理堆上的所有内容。
    哦内存泄漏将会受到惩罚。

-高级功能（超越活动） 哦让游戏更具挑战性！

- 生成的 ASCII 字符符号不限于目标字符串中的符号，而是 ASCII 表中的所有字母数字字符。

- 必须保证生成的 5 个项目中，至少有两个来自 Goal String。

使用示例可执行文件**PPA3.exe**供可交付参考。

必需：为您的系统配置 MacUILib 库和 Makefile

- 根据您的操作系统修改MacUILib.h如下：
  - 哦 **视窗**
    - 取消注释第 4 行（#定义 WINDOWS)
    - 注释掉第 5 行（#定义 POSIX)
  - **哦 Mac/Linux/Chrome**
    - 注释掉第 4 行（#定义 WINDOWS)
    - 取消注释第 5 行（#定义 POSIX)
- 根据您的操作系统修改Makefile如下：
  - 哦 **视窗**
    - 注释掉第 5 行（帖子链接器=…)
  - **哦 Mac/Linux/Chrome**
    - 取消注释第 5 行（帖子链接器=…)

评分方案

-基本特点**[总分25分]** 哦

[10分，提供细分]随机非重复项生成算法实现。

- [6 分] 在正确范围内生成不重复的 xy 坐标。
- [4 分] 从目标字符串中选择非重复字符。

哦 **[2 分]**正确的玩家-物品碰撞检测

哦 **[3分]**正确的神秘字符串显示、角色揭示以及碰撞时新物品的生成 **[2 分]**正确的移动计数实施和显示

哦

哦 **[2 分]**正确的最终游戏实施 - 获胜与强制退出。

哦 **[6 分]**无内存泄漏。在堆上创建项目箱和神秘字符串。

-高级功能**[总分5分]** 哦

[1分]仅从目标字符串中正确选择 2 个字符

哦 **[4 分]**从目标字符串以外的字符中正确选择另外 3 个字符。

推荐工作流程

这是进入2SH4项目之前的最后一份购电协议。虽然完成 PPA3 所需添加的代码行数可能不会超过 100 行，但需要一系列批判性思维活动才能完成设计。凭借您在前两个 PPA 中积累的经验，您应该能够在没有太多编码指导的情况下完成大部分实现。下面推荐的工作流程将为您提供概念性提示和伪代码算法，以帮助您完成实施。

请记住，除非您是经验丰富的程序员，**不要**尝试一次性开发这个程序。总是，**总是**在添加更多功能之前，请确保当前功能按预期工作，以防止造成调试噩梦。

熟悉这个工作流程！ 当您继续学习更高级别的软件课程时，迭代设计策略是绝对必须的。假设您在软件开发实践方面更有经验，随着我们进入后续实验室/项目准备活动，我们还将逐步减少手持技巧的数量。

注意动态内存分配！ 这是第一个要求您将数据放置在堆上的活动，以实现更好的内存管理实践。您必须记住在使用完毕后将每一块堆内存返回给计算机！

迭代 0 – 将 PPA2 移植到 PPA3 Skeleton 中，并将 Lab 3 Q1 移植到 myStringLib 中

- 通读 PPA3 框架代码，并将 PPA2 实现中适用的所有内容复制到正确的部分。PPA3 框架代码包含足够的 [COPY AND PASTE] 说明供您遵循。
- **停止！** 复制完成后，编译并确保您可以运行具有与 PPA2 相同功能的 PPA3，而不会出现任何语法或语义错误。

- **在测试之前请勿继续**

- 然后，检查 myStringLib.h；它包含您在实验 3 Q1 中部署和测试的四个字符串函数的原型。将所有四个函数的实现复制到 PPA3 中的 myStringLib.c 中。这是我们自己的字符串库，将在 PPA3 中使用。
  哦这就是功能性 C 库的典型开发方式：原型化、单元测试，然后模块化。

迭代 1 – 实现游戏功能框架

- 在全局范围内声明常量字符串“McMaster-ECE”。这是我们的**目标字符串**。
- 在全局范围内声明一个字符数组指针，然后在该数组中分配足够的空间堆 保存与目标字符串相同数量的字符。这是我们的**神秘的弦**,并且它必须根据要求分配在堆上。
  哦 分配后，用“?”填充神秘字符串中的所有字符，除了最后一个字符为 NULL。考虑应该在哪个例程中进
  哦 行分配调用。
- 在全局范围内声明一个 struct objPos 指针，然后在堆中分配足够的空间以在数组中保存 5 个 struct objPos 实例。这是我们的**物品箱**,并且它必须根据要求分配在堆上。
  哦 再次考虑要为项目箱中的所有元素初始化什么，以及应在哪个例程中完成分配和初始化调用。

  哦 选择字母数字字符来初始化项目，并选择一些可预测的 xy 坐标以简化调试。

- 在全局范围内声明 moveCount 整数，以跟踪自程序启动以来程序循环迭代的次数。 **重要的！**部署这些变量后，转到
- CleanUp() 例程并确保释放所有堆分配的变量。您将因内存泄漏而受到惩罚！

- 部署完这些功能后，继续在游戏板上显示神秘字符串、移动计数和物品箱中的五个物品。

  哦考虑要修改主逻辑和绘制例程中的哪些内容才能实现这些功能。

哦 停止！ 在您可以在显示屏上正确绘制这些特征之前，请勿继续。
哦 停止！ 在内存分析器告诉您程序内存泄漏为零之前，请勿继续。

- **推荐的验证设置**
  哦 目标字符串 “麦克马斯特-ECE”
  哦 神秘的弦 “？？？？？？？？？？？？？”
  哦 移动次数 0,更新每个程序循环迭代 {10, 5, '@'}
  哦 玩家
  哦 物品箱 {1, 1, 'A'}, {2, 2, 'B'}, {3, 3, 'C'}, {4, 4, 'D'}, {5, 5, 'E'}

  哦 预期显示内容：

```
############################
# A                        #
#B                         #
# C                        #
#D                         #
#      乙     @            #
#                          #
#                          #
#                          #
##########################神
秘字符串：???????????? 移动次数：0
```

迭代 2 – 随机物品位置生成

在上一次迭代中部署了所有静态功能后，我们首先部署随机项目位置生成逻辑。在 PPA3 框架代码中，随机项生成函数的签名已经原型化为：

空白生成项目（**结构体**obj位置列表**[],常量整型**列表大小，**常量结构**对象位置*****玩家对象\
常量整型x范围，**常量整型**y 范围，**常量字符***）

在哪里

- 列表[] 指向项目箱的指针。所有随机生成的物品都应该放在这里。物品箱的大小。根据
- 列表大小 PPA3 规范，默认值为 5。
- 玩家对象 指向玩家结构体的指针，以确保随机生成的物品不会落在玩家位置上。

- x范围 随机生成的整数 x 坐标的独占上限 随机生成的整数 y 坐标的独占上限
- y范围
- 斯特 应为项目箱中的 5 个随机项目选择 5 个随机字符的目标字符串。

每当我们需要在游戏板上生成 5 个新的随机项目时，就应该调用此函数。

基于增量工程的原理，我们将分两个阶段来解决这个功能的实现：a）游戏板范围内随机不重复的xy坐标生成，b）从目标字符串中随机不重复的角色选择。迭代 2 涉及阶段 a)。

您并不局限于下面要介绍的推荐算法。您可以使用您提出的任何算法，只要它们符合 PPA3 软件规范即可。

停止！ 在实施之前，将物品箱中物品的初始坐标更改为游戏板外的某个位置。想想为什么这是必要的。

- **非重复xy坐标生成的通用算法**
  - **哦** 生成[0, xRange-1]范围内的候选整数x坐标值。（想想这个范围是否正确！如果不正确，你有责任改正）
  - **哦** 生成[0, yRange-1]范围内的候选整数y坐标值。再次思考一下它的正确性。使用候选 xy 坐标，检查它是否匹配
  - **哦** 1) 玩家位置，以及 2) 函数调用的此实例中项目箱中任何先前生成的项目的位置。
    - 如果是，则丢弃该候选并继续生成另一个 xy 候选。
    - 否则，将此 xy 坐标写入项目箱中的目标项目，然后移至下一个项目。
  - **哦** 当 list[] 中的所有项目都收到新生成的随机 xy 坐标时，算法停止。

  哦 作为良好的编程习惯，在开发该算法时，打印以下项目作为调试消息，以跟踪实现的正确性。我们鼓励您在需要时使用调试器。

    - 玩家位置
    - 物品箱中所有物品的位置
    - 新生成的 xy 坐标候选
    - 回顾讲座——先睹为快！

  哦 **陷阱**–如果 C 编译器抱怨无法使用变量（xRange、yRange）作为数组维度来创建数组，则可以使用全局预处理器常量而不是本地输入参数。这不是最正确的方法，但很好地达到了目的。

  哦 **停止！**在继续之前，请确保您的随机坐标生成工作正常。

- 正确部署算法后，用此函数替换 Item Bin 的手动初始化，以便程序可以在程序启动时将初始项目集放置在随机生成的坐标上。

- 例如，一旦算法正确开发并在启动时调用以初始化项目箱中的项目，屏幕上的初始输出可能如下所示，并且每次您启动时，这五个项目将出现在不同的非重复位置。启动该程序。

```
############################
#                          #
#          A               #
#                          #
#            乙            #
#            @     D       #
#                  C       #
#                          #
#B                         #
#########################神
秘字符串：??????????? 移动次数：0
```

- **停止！**在进行下一次迭代之前，您应该确保正确实现随机非重复坐标生成。

  哦 我们建议您保留与随机项生成相关的调试消息，直到下一次迭代结束。

迭代 3 – 从目标字符串中随机选择字符

接下来，我们需要将字符选择算法添加到GenerateItems函数中。该算法可以独立于 xy 坐标生成来完成（想想为什么）。

- **从参考字符串（目标字符串）中选择非重复字符的通用算法**
  - **哦** 生成适当范围内的随机整数索引，使得所有可能的索引在目标字符串中都有有效的对应字符***排除***NULL 终止符。为此，您可能需要使用您自己的字符串库中的 my_strlen() 。

  - **哦** 使用新生成的索引，检查目标字符串中此索引处的 ASCII 字符是否与函数调用的此实例中任何先前生成的项目的字符匹配。
    - 如果是，则丢弃该候选并继续生成另一个索引。
    - 否则，将此 ASCII 字符写入当前项目的符号成员，然后移至 bin 中的下一个项目。

  - **哦** 当 list[] 中的所有项目都收到来自目标字符串的新生成的非重复字符的随机选择时，算法停止。

  - 哦 同样，您应该打印出所有相关的调试消息，以确保您的算法实现按预期工作。

  - **哦停止！**在继续之前，请确保您的随机角色选择实现运行良好。

- 正确部署算法后，该函数应在游戏板上的随机位置生成五个项目，并从目标字符串中随机选择字符。

- 每个程序启动时的示例输出都会有所不同。请注意，所有五个字符均来自示例目标字符串“McMaster-ECE”。

```
############################
#                          #
#          中号            #
#                          #
#           -              #
#           @     A        #
#                 C        #
#                          #
#E                         #
#########################神
秘字符串：??????????? 移动次数：0
```

- **停止！**在进行下一次迭代之前，您应该确保正确实现随机非重复项生成。

  哦完成后，您可以删除与随机项目生成相关的调试消息。

迭代 4 – 物品收集、角色启示、移动计数更新和游戏结束条件

这个迭代并不困难，但包含许多需要实现的小功能。下面提供了提示。完成后，您将拥有一个可运行的 PPA3 寻宝游戏，如果您选择完善游戏体验或追求超越功能，则可以在额外的迭代中对其进行完善。推荐的工作流程只能为您提供到目前为止的支持。

- **项目收集实施**
  - 哦 阅读碰撞检测部分。
  - 哦 当玩家对象与物品箱中的任何一项物品发生碰撞时
    - 检查该物品下的符号字符是否已被收集。如果没有，请在神秘字符串的正确索引中显示该字符的所有实例。否则，请勿执行任何操作。
    - 无论神秘字符串中是否显示了字符，始终在检测到的碰撞结束时重新生成项目箱中的所有 5 个项目。

- **人物启示**
  - 哦 要在正确位置揭示神秘字符串中的相应字符，请迭代目标字符串并找到其出现的所有索引。

- **移动计数更新**
  - 哦 实际上只是计算程序循环运行的迭代次数。
  - 哦 当播放器 FSM 处于 STOP 状态时，您可能希望停止计数。想想为什么。

- **游戏结束条件**
  - 哦 在每次检测到的碰撞结束时，使用您自己的 my_strcmp() 函数检查神秘字符串的内容是否与目标字符串中的内容相同。
  - 哦 如果是，则以获胜消息结束游戏。
  - 哦 您可能需要设置一些东西来区分获胜的最终游戏和命令强制的最终游戏。

- **不要忘记！！** 增量工程，亲眼目睹才相信，并使用所有调试技术来确保您的程序开发处于控制之中。

-不要忘记！！ 在每次开发迭代结束时检查内存泄漏。

额外的迭代 - 高级功能（超越活动）

- **对于超出范围的活动，仅提供概念性提示。**
  您需要应用先前设计迭代中的知识和技能来提出附加功能。

- **超越功能 – 不仅从目标字符串中生成随机选择的字符**
  - 哦 首先，在迭代 4 中使用非重复选择算法从目标字符串中选择两个字符。然后，从所有可能的字母数字和标点
  - 哦 字符中选择剩余的三个字符，并再次使用非重复选择算法。可能的 ASCII 字符范围可能是 (33, 126)。

    - 哦 为防止游戏体验混乱，应避免使用玩家对象符号、边界符号和空间字符。

  - 哦 如果有兴趣，您甚至可以研究用于非重复元素生成的位向量方法。直到 2SI3 中我们才会讨论这个问题。