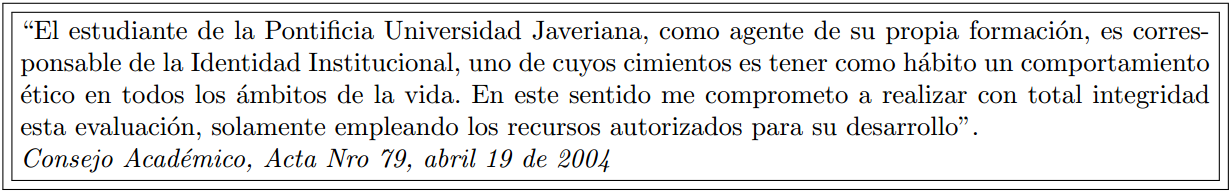
**INTRODUCCIÓN A LA PROGRAMACIÓN – GRUPO C**

**CONSIDERACIONES DE ENTREGA**

* **El proyecto debe ser entregado por Bb el día 24 de mayo hasta las 23:59, de manera que las sustentaciones se realicen el día 25 y 27 de mayo.**
* El proyecto debe ser subido a la plataforma Blackboard como único medio de entrega.
* **En Blackboard debe ser subido el o los archivos .py (archivo de Python) que desarrollen su proyecto. Su código no puede ser enviado por ningún otro formato.**
* El proyecto es completamente individual y deberá ser sustentado de forma virtual, por medio de cámara web donde se validará la propiedad intelectual de su trabajo.
* El fraude puede ocasionar la apertura de un proceso disciplinario.



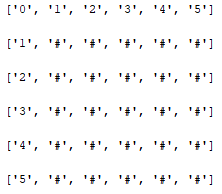
**Objetivo** **general:**

Practicar los temas aprendidos a lo largo del curso en un proyecto que integre los conocimientos adquiridos.

**Especificaciones Juego BuscaminasLab**

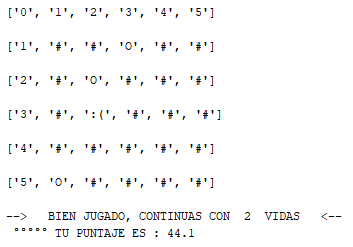
**Inicio de juego**

* 1. Al iniciar el juego debe permitir al usuario escoger un nivel de dificultad: bajo, medio, alto
  + Bajo, con tablero de 5x5 y 6 bombas
  + Medio, con tablero de 7x7 y 10 bombas
  + Alto, con tablero de 9x9 y 14 bombas
  1. El jugador debe suministrar el nombre.
  2. Al iniciar el juego, el tablero del buscaminas se debe ver como una matriz. Todas las casillas de la matriz deben tener un símbolo que indique no se ha jugado en ellas. Por ejemplo, la siguiente imagen usa el símbolo (#).



**Juego**

* 1. Para jugar el usuario dará una una coordenada de fila y columna. Esta coordina indicaría donde querrá destapar la casilla. En caso de que el jugador no cuente con suerte, en esa casilla habrá una bomba, esa bomba será representada por una carita triste **:(** , pero cuando el jugador sí haya tenido suerte, en la casilla deberá aparecer el símbolo (O). Un ejemplo de cómo se vería el tablero con algunas jugadas hechas está en la siguiente imagen.



* 1. Su buscaminas debe informar al usuario el puntaje y la cantidad de vidas con las que cuenta considerando las siguientes reglas:

**Vidas**

* El jugador iniciará el juego con **3 vidas** para descubrir todo el tablero. C
* Cuando la coordinada dada por el usuario coincide con una mina, perderá una vida
* Las minas del tablero deberán generarse de forma aleatoria, con la intención de que cada que se ejecute el juego la forma de solucionarlo sea diferente.

**Puntaje**

* Con la intención de hacerlo más entretenido, los puntos irán incrementando a medida que el usuario va acertando casillas sin bomba, para ser más específico, la primera jugada correcta se le dará 10 puntos, y a partir de esa jugada, cada siguiente jugada le dará un puntaje bajo la siguiente fórmula (puntos = puntos+(puntos\*1.1) ).
* Las jugadas en que el usuario coincida con una bomba no se le otorgarán puntos y si le restarán una vida como se explicó anteriormente.
  1. **Fin de juego** El juego terminará en 2 condiciones, porque se le acabaron las vidas al jugador, o porque logró descubrir todas las casillas que no tenían bombas antes de que sus vidas se le hubieran acabado.
  2. **Historial:** El juego debe permitir guardar los datos que sean necesarios mientras el usuario está jugando, de manera que, si el programa es cerrado abruptamente o el jugador quiso cerrar la ventana, al ejecutar el juego nuestro programa le pregunte si desea continuar el juego que dejó sin concluir o desea empezar un nuevo juego. Para lograr lo anterior, usted deberá utilizar sus conocimientos en el manejo de archivos que le permitan guardar y cargar los datos del juego cuando sea necesario. Se aconseja hacer uso de archivos serializados o binarios para guardas sus datos ya que se simplifica la escritura y lectura de los datos.
  3. **Sorpréndeme**. [ incorpore funcionalidades extras a las operaciones que solicito. Según la sustentación, creatividad e interés que tengan, le daré entre 0 y 5 décimas adicionales en la nota del proyecto]

En resumen, su juego cuenta con las siguientes consideraciones:

1. Juego de un solo jugador
2. Debe pedir el nombre al usuario
3. Tiene 3 vidas para jugar
4. Se resta una vida cada que da a una bomba
5. Se maneja un puntaje que incrementa cada que se da en una casilla sin bomba
6. En cada jugada debe mostrar el estado del tablero, las vidas que le quedan al jugador y su puntaje.
7. Las bombas del tablero inicial deben ser creadas aleatoriamente.
8. Si el usuario pierde, se debe mostrar el tablero hasta donde llegó el usuario y un mensaje indicando que se han acabado las vidas y con ello el juego.
9. Cuando el usuario gane se debe mostrar un mensaje de felicitaciones y el tablero resuelto.
10. El programa debe guardar la información del juego para que el usuario tenga la posibilidad de continuar su juego, aunque haya cerrado el programa, para ello se deben usar archivos.

**Reglas generales.**

* Cumplir con el estándar de codificación lowerCamelCase, en el que las operaciones inician con un verbo en infinitivo. Las palabras compuestas inician en minúsculas y la segunda palabra tiene la primera letra en mayúsculas Ejm: iniciarJuego*.* Las variables tienen nombres que semánticamente se relacionan con la función que cumplen. El cumplimiento de este estándar es obligatorio.
* Tenga en cuenta que, si usted comparte su código con alguien, ese alguien podría entregar su código como propio, o algo muy similar a su código, y la probabilidad de ser valorado como copia es muy alta. No se arriesgue.
* En las sesiones del 19 y 20 de mayo estaré conectad en el canal de clase para responder dudas y ayudarles con los problemas que tengan.

**Rúbrica de evaluación**

Este proyecto puedo traerle retos en el proceso de su construcción, pero tenga en cuenta que más allá de tener un producto operando a la perfección lo más importante es que consolide y ponga en práctica los conocimientos que el curso debe haberle entregado. Usted debe dar cuenta de esos conocimientos durante la sustentación.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **5** | **4** | **3** | **2** | **1** | **0** |
| Funcionalidad  (60%) | Excedió las expectativas | Cumplió con todos los requisitos. | Fueron desarrollados mínimo el 75% de los requisitos | Fueron desarrollados mínimo el 50% de los requisitos | Fueron desarrollados mínimo el 25% de los requisitos | Fueron desarrollados menos del 25% de los requisitos |
| Estilo de codificación  (15%) | El código se encuentra correctamente indentado, los nombres de los atributos y las funciones cumplen con el estándar de nombramiento. El código tiene documentación interna para facilitar la revisión. | La mayoría del código se encuentra correctamente indentado-  La mayoría de los nombres de los atributos y las funciones cumplen con el estándar de nombramiento. La mayoría del código tiene documentación interna para facilitar la revisión | Falta una de las cosas del estilo de codificación o alguna se cumple con mala calidad | Faltan dos de las cosas del estilo de codificación o se cumplen con mala calidad: | Faltan tres de las cosas del estilo de codificación o se cumplen con mala calidad: | No cumple con el estándar de nombramiento  No se encuentra correctamente indentado  No está divido adecuadamente |
| Mejores prácticas  (25%) | El código muestra mejores prácticas de desarrollo siempre. Reúso, separación de operaciones, buen manejo de ciclos, simplicidad,  programación defensiva  validaciones. | El código muestra mejores prácticas de desarrollo en la mayoría de los casos, pero falta mejorar algunos de los siguientes aspectos Reúso, separación de operaciones, buen manejo de ciclos, simplicidad,  programación defensiva  validaciones | El código muestra buenas prácticas de desarrollo pero falta mejorar  dos de los siguientes aspectos  Reúso, separación de operaciones, buen manejo de ciclos, simplicidad,  programación defensiva  validaciones | El código aplica pocas buenas prácticas de desarrollo. Falta mejorar tres de los siguientes aspectos:  Reúso, separación de operaciones, buen manejo de ciclos, simplicidad,  programación defensiva  validaciones. | El código aplica muy pocas buenas prácticas de desarrollo.  Falta mejorar cuatro de los siguientes aspectos:  Reúso, separación de operaciones, buen manejo de ciclos, simplicidad,  programación defensiva  validaciones. | El código no considera buenas prácticas de desarrollo.  Falta mejorar cinco o más de los siguientes aspectos:  Reúso, separación de operaciones, buen manejo de ciclos, simplicidad,  programación defensiva  validaciones. |

**Rubrica de propiedad intelectual**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 0.8 | 0.6 | 0.4 | 0.2 | 0 |
| **Sustentación** | Es evidente que el estudiante entiende el código que desarrolló lo explica con claridad y responde correctamente a las preguntas. | La sustentación es buena pero se evidenció inseguridad del estudiante para explicar algunas partes del trabajo desarrollado o para responder algunas preguntas. | La sustentación es aceptable se evidencia que el estudiante desarrolló el código pero le cuesta trabajo explicar aspectos del código. | La sustentación es regular se evidenció inseguridad del estudiante para explicar gran parte del trabajo desarrollado o para responder muchas de las preguntas. Parece que el código no hubiera sido desarrollado por el estudiante. | El estudiante demuestra que entiende partes del código, pero no tiene claro cómo se relacionan con la funcionalidad solicitada. | Se evidencia que el estudiante no entiende el código desarrollado, no es capaz de responder a las preguntas formuladas de manera correcta. |