Ejercicio 2do Cuatrimestre 2022

Buscaminas

Objetivo

El objetivo del ejercicio consiste en agregar el recuento de minas en un tablero de *Buscaminas* completo.

Buscaminas es un juego popular en el que el usuario tiene que encontrar las minas usando pistas numéricas que indican cuántas minas existen adyacentes (horizontal, vertical, diagonal) a un cuadrado particular.

Se debe desarrollar un programa que cuente el número de minas adyacentes a cada cuadrado vacío y los reemplace con el resultado.

El tablero es un rectángulo compuesto por caracteres punto (‘.'). Una mina se representa con un asterisco ('*').

Si un espacio no contiene minas adyacentes, se lo deja en blanco.

Ejemplos

Por ejemplo, se puede recibir un tablero de 5 x 4 como el siguiente:

```
.*.*.
..*..
..*..
.....
```

El programa debe transformarlo en esto:

```
1*3*1
13*31
·2*2·
·111·
```

Debe recibirse como parámetro la ruta al archivo del tablero de entrada con ese formato.

Uso de la memoria

El archivo de tablero tendrá una entrada y salida en formato **ASCII**. Los **Strings** de Rust y los **&str** son **UTF-8**. Dado que esperamos que **&str** sea **ASCII**, se puede invocar **.as_bytes()** y referirnos a los datos interiores a través del slice **&[u8]**. Iterar sobre un slice de u8 es mucho más rápido porque no hay verificaciones de codificación de caracteres (cada ASCII tiene el tamaño de un u8).

Escribir el programa sin clonar (.clone()) el input.

Recibir el Input

El input es un archivo en el filesystem con el formato de entrada del tablero.
En la invocación del programa se debe proveer la ruta a ese archivo.

Requerimientos No Funcionales

Los siguientes son los requerimientos no funcionales para la resolución del proyecto:

- El proyecto deberá ser desarrollado en lenguaje **Rust**, usando las herramientas de la biblioteca estándar.
- Se deben implementar **tests unitarios y de integración** de las funcionalidades que se consideren más importantes.
- No se permite utilizar crates externos.
- El código fuente debe compilarse en la versión estable del compilador y no se permite utilizar bloques unsafe.
- El código deberá funcionar en ambiente Unix / Linux.
- El programa deberá ejecutarse en la línea de comandos.
- La compilación no debe arrojar warnings del compilador, ni del linter clippy.
- Las funciones y los tipos de datos (struct) deben estar documentados siguiendo el estándar de cargo doc.
- El código debe formatearse utilizando cargo fmt.
- Las funciones no deben tener una extensión mayor a 30 líneas. Si se requiriera una extensión mayor, se deberá particionarla en varias funciones.
- Cada tipo de dato implementado debe ser colocado en una unidad de compilación (archivo fuente) independiente.