
MAIOR LÆTITIA NOSTRA
UdeSantiago
de Chile

UdeSantiago de Chile

PROYECTO SEMESTRAL

CURSO DE MÉTODOS DE PROGRAMACIÓN

2-2021

CONTENIDO

FIGURAS


MAIOR LÆTITIA NOSTRA
UdeSantiago
de Chile

1. Introducción

El siguiente documento muestra un problema a solucionar, en el marco del desarrollo del proyecto semestral de curso, cuyo objetivo es que el/la estudiante pueda construir, desde el pensamiento crítico, la búsqueda de soluciones para un problema ficticio. Principalmente el/la estudiante debe implementar los métodos de programación necesarios, abordados en clases de teoría y laboratorio de la asignatura.

El documento está estructurado de la siguiente forma:

En primer lugar, se presenta el contexto y la situación que genera el problema, para el cual el grupo de estudiantes debe identificar e implementar la solución correspondiente.

Posteriormente se explican las reglas que dicha implementación debe seguir de acuerdo con la situación explicada, para la construcción de su solución. En este punto pueden existir ambigüedades, las cuales el grupo puede consultar en el foro del proyecto que se encuentra disponible en la plataforma Moodle del curso de UdeSantiagoVirtual. Las respuestas que el cuerpo académico otorgue serán consideradas parte del enunciado.

El software debe implementar ciertas funcionalidades específicas, las cuales son listadas en la sección 5 de este documento. Estas serán las bases de la evaluación que se realizará en la entrega final de su proyecto.

Para finalizar el documento se encuentra la descripción de los distintos hitos a elaborar por el grupo para la evaluación del proyecto, junto con las reglas que posee cada uno y sus distintas fechas de entrega.

2. Contexto

What does not kill me makes me stronger.

Friedrich Nietzsche

Ingrid se despertó sin saber quien es ni donde se encontraba. A su alrededor había un grupo de jóvenes de su misma edad que la miraban fijamente. Lo primero que se le vino a la mente fue pararse y salir corriendo. Como vió que nadie la seguía miró hacia el horizonte y descubrió que se encontraba en un campo cuadrado de aproximadamente 1Km x 1Km con árboles, pasto y varias cabañas. El detalle más extraño fue constatar que se encontraba rodeada de paredes verticales muy altas de cemento.

Más tarde, una vez acogida por el clan, ellos le contarían que a todos les había pasado lo mismo. Durante años, dicha comunidad de jóvenes había intentado sin éxito escalar las paredes, construir túneles, incluso usar globos aerostáticos. Lo único en lo que tenían algún grado de avance, era la exploración del laberinto al que se accedía por el costado sur del campo. Ellos tenían la esperanza de que algún día lograrían encontrar una salida a dicho encierro.

El laberinto no era un lugar seguro. Su estructura de paredes y pasillos se modificaban a cada hora que pasaba. Al ocaso, la entrada del laberinto se cerraba y aparecían dentro de él criaturas mecánicas que perseguían a aquellos que quedaban atrapados dentro de él durante la noche. Eran pocos los que lograban sobrevivir y volver al campo cuando se abría la entrada a la mañana siguiente. El grupo de jóvenes mantuvo durante años su convicción de proseguir las exploraciones logrando un mapa parcial del laberinto.

Con el tiempo Ingrid logró conquistar la confianza del clan y fue nombrada jefa. Fue un tiempo de grandes aventuras para lograr vencer los peligros y liberarse de la prisión en que estaban. Más tarde fundaría un nuevo pueblo, esta vez con una vida libre de peligros. Pero esa es otra historia, lo que les he narrado es la historia de mi madre y detallo a continuación cuáles fueron las campañas que realizó para lograr escapar del laberinto.

Se requiere construir un juego basado en el contexto anterior, el cual estará compuesto de las siguientes campañas.

1. Campaña 1: Buscar la mayor cantidad de cofres en los laberintos. Estos cofres constituyen hitos importantes para lograr salir.
2. Campaña 2: Encontrar la mayor cantidad de botes en los laberintos. Existen lugares que se encuentran inundados cuyo paso requiere de botes.
3. Campaña 3: Salir de los laberintos.

3. Descripción de la implementación

3.1. Campaña 1

Esta etapa consiste en encontrar cofres en el laberinto y resolver las pistas que estos tengan. Las pistas pueden ser ocupadas en el mismo laberinto u en otro cuando éste cambie.

Las pistas disponibles son expresadas mediante juegos (Sopa de letras y ahorcado) cuya configuración viene dada por archivos que el programa debe leer, estos archivos se llamarán: pista_1.txt, pista_2.txt,..., pista_i.txt donde i es la cantidad de pistas en el mapa. Para resolver las pistas, debe entregar un menú que le permita al usuario interactuar con el juego.

3.1.1 Sopa de letras

En caso de encontrar una pista “i”, se debe leer el archivo de texto correspondiente, en su primera fila se tendrá la letra correspondiente al juego, en este caso una letra s, para representar el juego de sopa de letras, la segunda fila tiene la cantidad de palabras verticales `"x"`, horizontales `"y"` y la cantidad de intentos fallidos respectivamente. La tercera fila tendrá la cantidad de filas y columnas de la sopa de letra mientras que las siguientes `x + y` filas corresponden a las palabras dentro de la sopa (ordenadas con las verticales primero y luego horizontales), a continuación, se tendrá la sopa de letras donde las siguientes 10 filas tendrán las letras y palabras de la matriz en su respectiva posición. El formato del archivo está en la figura 1.

pista_1.txt

```
s
3 6 5
10 20
melon
sandia
guinda
arandano
pera
frutilla
naranja
platano
frambuesa
kjofrutillagdslapera
uyhgtjtrdopuunndkswl
srtfcesawdftimanzana
rtfdeswctghtnwtfreer
strdmnaranjadredstrf
afdteframbuesaiolyht
nhytlhtfgrtfdrwsxzas
dcfvojurtdplatanoyhb
iwsanjuarandanoytgrd
akjgyloujmnxzcvbgtrd
```

Figura 1: ejemplo archivo de entrada pista_1.txt

Una vez leído el archivo, se deberá entregar un menú que le permita al usuario interactuar con el juego. Para ello, se pregunta si se desea usar la pista de inmediato o si prefiere guardarla. En caso de usar la pista, se le presenta la sopa de letra y se le pide que ingrese una palabra y las posiciones de la letra inicial y final, esto hasta completar el juego o utilizar todos los intentos. Un ejemplo de esto está en la figura 2, cabe destacar que puede hacer el menú con el estilo que usted prefiera, se agradece la creatividad.

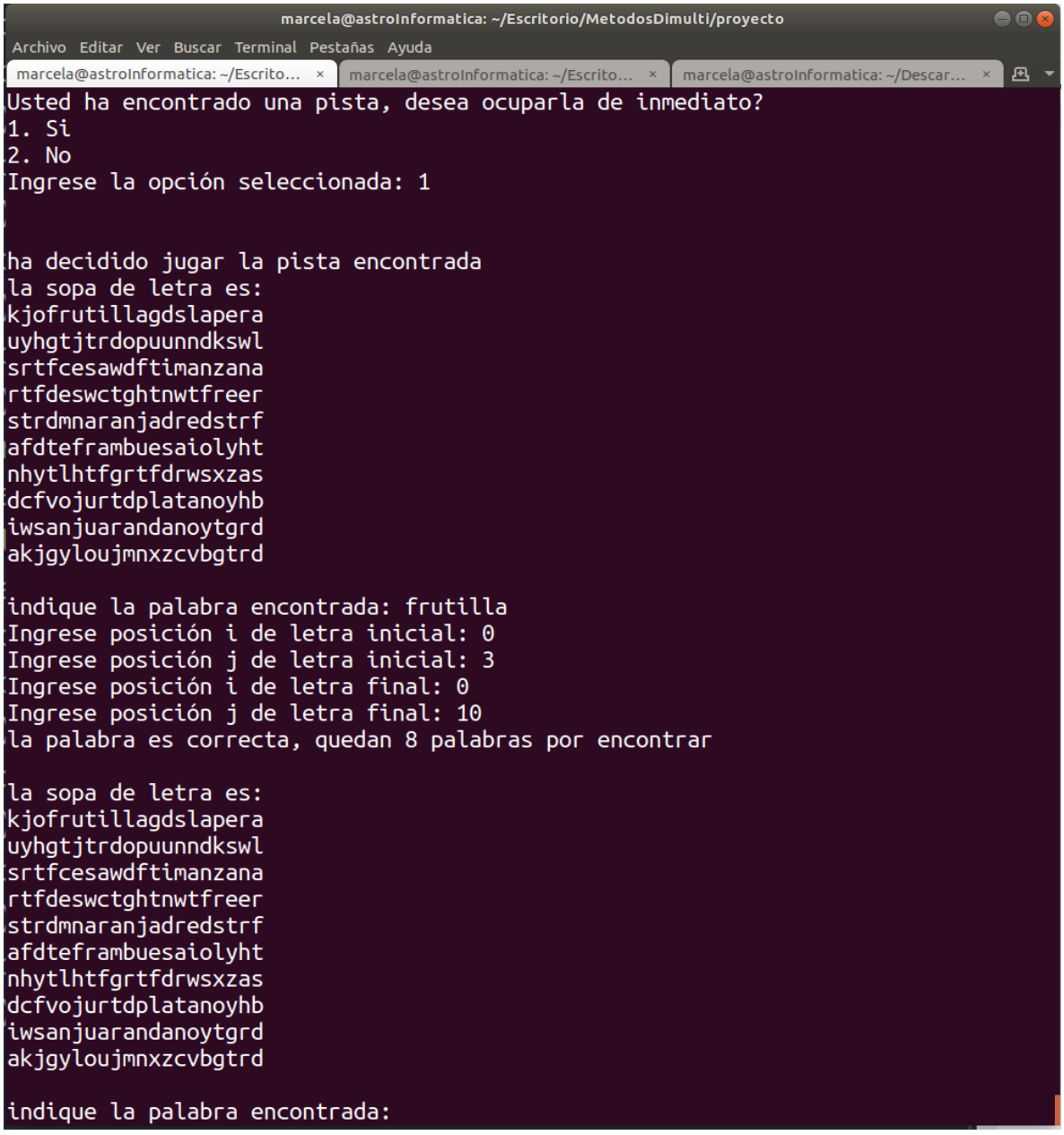

Figura 2: ejemplo de menú para juego sopa de letras

3.1.2 Ahorcado (opcional)

El ahorcado (también llamado colgado) es un juego de adivinanzas de lápiz y papel para dos o más jugadores. Un jugador piensa en una palabra, frase u oración y el otro trata de adivinar según lo que sugiere por letras o números. Para este laboratorio, el juego consistirá en leer un archivo de texto el cual contendrá en su primera fila la letra a, que indica que es un archivo de ahorcado. La segunda fila la cantidad de palabras a jugar y la cantidad de intentos fallidos por palabra respectivamente, a continuación la lista de palabras dentro del juego y finalmente las palabras incompletas, las cuales deben ser descubiertas. Un ejemplo de archivo se encuentra en la figura 3.

pista_2.txt

```
1  a
2  3 10
3  metodos
4  programacion
5  proyecto
6  m___do_s
7  __og__m_c_on
8  p__y___o
```

Figura 3: ejemplo archivo de entrada pista_2.txt

Cabe destacar que las palabras a utilizar serán de mínimo 3 letras y no ocurrirán casos extremos como por ejemplo que calcen dos palabras en una misma fila.

Una vez leído el archivo de entrada, para jugar al ahorcado, primero se le debe mostrar las palabras a completar y a continuación se le debe preguntar la palabra que desee completar, para luego consultar por la acción a realizar, y así permitir que el usuario juegue. Un ejemplo de menú se encuentra en la figura 4, recuerde que es solo un ejemplo ya que puede ser todo lo creativo que guste.

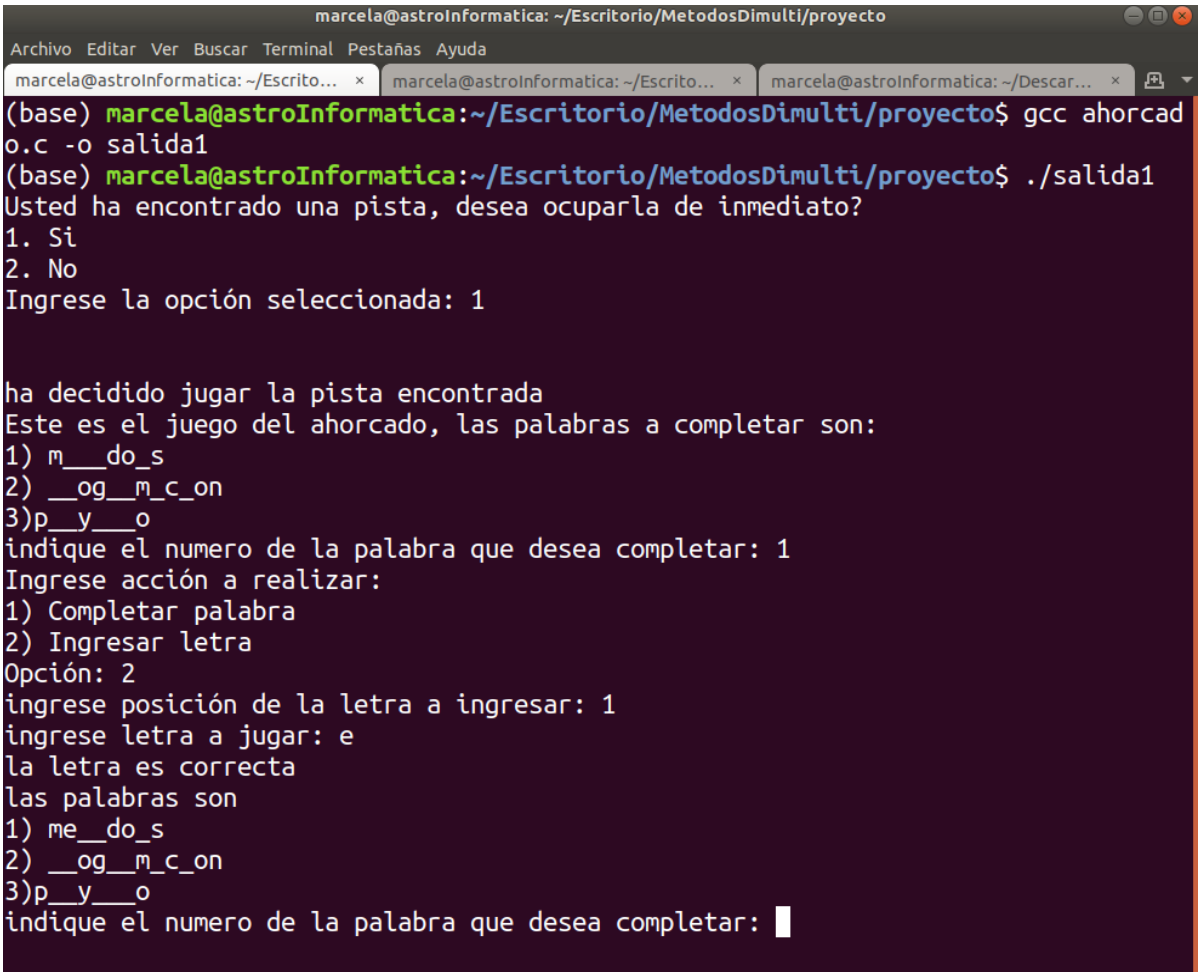

Figura 4: ejemplo de menú para juego ahorcado

En ambos casos las pistas resueltas entregarán un trozo del mapa, por ejemplo: revelará el camino de las siguientes dos posiciones alrededor. Es decir, si el jugador está en la posición [3,3], la pista le revelará el camino que hay desde la posición [i-2,j-2] hasta la [i+2,j+2].

3.2 Campaña 2

Esta etapa consiste en buscar y guardar botes existentes dentro de los laberintos porque según las investigaciones, se han encontrado botes inflables en distintas partes. Se sabe además que hay veces que no se han encontrado botes pero hay agua que impide el paso en el laberinto. Sería bueno guardar los que se pueda para ocuparlos en un futuro. Además estos cuentan con una capacidad máxima y solo un bote puede estar en el agua a la vez. Los botes deben ser manipulados por una persona como mínimo y cada uno tiene una capacidad limitada. Estos al ser inflados ya no se pueden volver a desinflar para guardarlos y así transportarlos por el mapa, es decir, solo se pueden usar en un sector con agua.

Para encontrar los botes deberá generar una estrategia de búsqueda. Tenga en consideración que pueden encontrarse botes en caminos no directos que tengan algún inconveniente para llegar como por ejemplo, un bote puede encontrarse en alguna isla en medio del agua y podría tener mayor capacidad que los que tienen actualmente.

La cantidad de botes a recolectar por lo menos debe ser suficiente para poder terminar el laberinto actual.

3.3 Campaña 3

Esta etapa consiste en salir del laberinto por el camino más corto, es decir, por la cantidad mínima de pasos para que todos salgan sanos y salvos. Debe considerar las condiciones actuales del jugador, es decir, botes disponibles y el camino conocido. El jugador a medida que realiza las campañas anteriores va generando un mapa conocido, por lo que recorre el mapa utilizando esta técnica para así encontrar todos los caminos posibles del mapa generando la oportunidad de encontrar botes o cofres que le permitan avanzar más rápido por el laberinto.

Una vez recorrido el ¾ del mapa, automáticamente su objetivo será encontrar la ruta más corta para llegar a la salida.

4. Acotaciones de Entradas y Salidas

En esta sección se habla ya específicamente de cómo su equipo de trabajo deberá desarrollar el proyecto en sí, en cuanto a la lectura de los archivos de entrada y la salida de la simulación.

4.1 Descripción del laberinto

Para realizar la simulación, el laberinto corresponde a un cuadrado desde los 10x10 hasta los 20x20, en donde cada casilla corresponde a un elemento del laberinto. Se indica en paréntesis el significado de cada letra en el archivo de entrada:

- Celda de ingreso al laberinto (i).
- Celda de salida del laberinto (e).
- Espacio en blanco en donde se puede transitar (x).
- Pared que no se puede traspasar (w).
- Cofre con una de las dos pistas. Al obtener la pista, el cofre se elimina (c).
- Piso con púas que inicialmente se encuentran levantadas en forma inicial (p).
- Interruptor en el piso para bajar o subir las púas de una casilla. No se puede pasar por encima de él (s).
- Agua en donde no se puede traspasar a menos que sea con un bote (a).
- Un bote inflable que al obtenerlo queda la casilla vacía (b).

Los archivos de entrada para los mapas se llamarán lab_i.txt y tendrán en la primera fila el tamaño de la matriz `n`, cantidad de interruptores, cantidad de cofres y cantidad de botes. Las siguientes `n` filas y columnas corresponden a la ubicación de los elementos antes mencionados del laberinto. Después seguirán las coordenadas correspondientes a dónde el interruptor desactiva las púas. El formato es `ii.jj aa.bb` en donde `ii` y `jj` corresponden a la ubicación del interruptor en el laberinto y `aa` y `bb` a la ubicación de las púas correspondiente a ese interruptor. Después de los interruptores irá el archivo correspondiente a cada cofre con las pistas, de tal modo que primero van las coordenadas `ii.jj` y luego el nombre del archivo a revisar. Al terminar los cofres, corresponde el turno a los botes, en donde irá la coordenada de la ubicación `ii.jj` y luego la capacidad en kilogramos. Un ejemplo del archivo de entrada está en la figura 5.

UdeSantiago de Chile

lab1.txt

```
10 2 2 2
xxxxpxbwex
cwxxwwwaaw
wswxwaaaax
xxwpwaaawx
xwwxwbwwxx
cxwxwwxxxw
wxxxxxxxwx
xxwwwwwxww
xwwxxxwxws
xxxxixwxxx
02.01 00.04
08.9 03.03
01.00 cofre1
05.00 cofre2
00.06 50
04.05 100
```

Figura 5: ejemplo de archivo para el laberinto

La Figura 6 muestra un ejemplo de forma gráfica cómo sería el ejemplo indicado anteriormente.

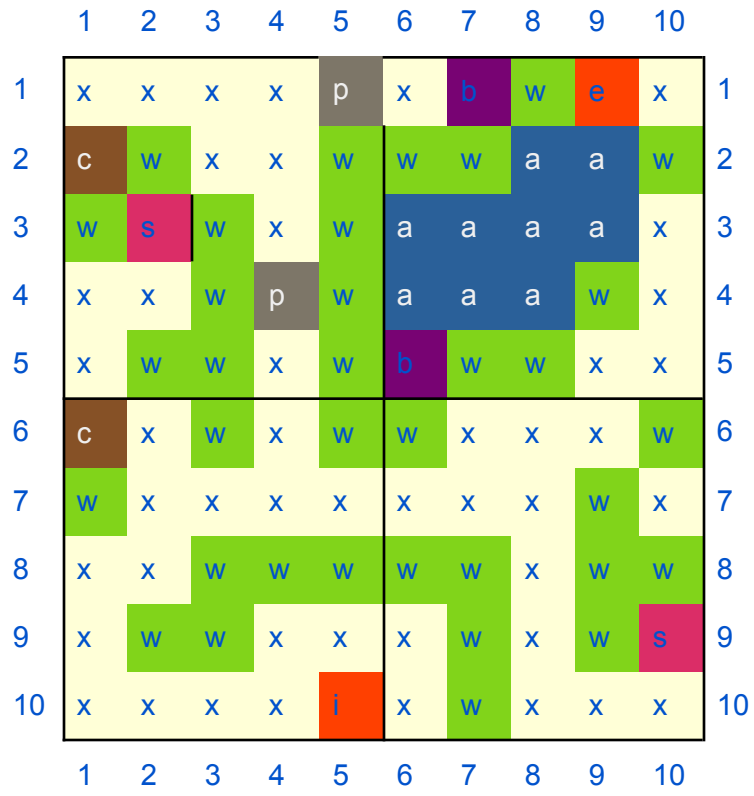

Figura 6: ejemplo de mapa inicial

Los avances por medio del laberinto solamente pueden realizarse hacia arriba, abajo, derecha o izquierda. Inicialmente el laberinto se encontrará oculto y mientras se avance quedará descubierto y se podrá ir visualizando el mapa completo poco a poco. Las personas solo podrán descubrir un cuadrado alrededor de ellas de 1 para cada lado cada vez que avance, es decir, si se encuentra en la posición x,y podrá descubrir las posiciones `x-1, y-1; x-1, y; x-1, y+1; x, y+1; x+1, y+1; x+1, y; x+1, y-1` y `x, y-1`.

4.2 Distribución de personas a transportar (opcional)

Para la realización de la distribución, se debe considerar la afinidad entre las personas a transportar, su peso y la resistencia de los botes. La cantidad de personas que desean salir del mapa estará definido en la primera fila de un archivo de texto, en las siguientes filas se indicará la persona y su respectivo peso, a continuación siguen las afinidades entre las personas a transportar.

Considere que la resistencia del bote indicará cuántas personas se pueden transportar, por lo que al momento de desplazar personas deberá considerar las afinidades y el peso máximo que puede soportar un bote.

En cuanto a la afinidad, esto indica cuáles integrantes están dispuestos a viajar juntos. Dado que hay un estrés por el largo tiempo que llevan intentando salir, además de muchas misiones fallidas, hay integrantes que preferirían no trabajar en equipo con cualquiera. Para ellas/os, remar es un trabajo que requiere colaboración y esfuerzo así que es mejor realizarlo con buenos compañeros, donde se sienta una mayor afinidad.

Las personas una vez desembarcadas del bote deberán esperar al resto para continuar su viaje todas juntas por tierra. Si fuese necesario explorar las aguas no es necesario que se llene el bote a su máxima capacidad, basta con una para ello mientras el resto espera en la orilla. Un ejemplo de archivo está en la figura .

archDist.txt

```
5
Lider, 60
hab_1, 58
hab_2 ,40
hab_3, 75
hab_4, 68
lider, hab_1, hab_2, hab_3, hab_4
hab_1, hab_2
hab_2, hab_4
hab_3,  hab4
```

Figura 7: ejemplo de archivo de entrada archDist.txt con las afinidades

4.3 Elementos necesarios para el desarrollo del algoritmo de supervivencia (opcional)

Para el cálculo de la supervivencia de las personas atrapadas en el laberinto, se entregará un informe detallado con el cálculo de lo que ocurre durante el día, específicamente es necesario entregar un archivo con nombre “ResultadoParte3_XX.out”; donde XX es un número entero que indica la versión del día analizado. Además, el archivo debe tener:

- La cantidad de habitantes actuales que hay.
- La cantidad de habitantes que han muerto en ese día. Mueren cuando excedan peso de balsa o pisan las púas.
- La cantidad de habitantes atrapados en el mapa.
- La cantidad de cada uno de los mapas recorridos en ese día.
- El día correspondiente analizado, comenzando como día 0 aquel día en que comienzan a realizar las campañas.

Todos los datos deben ser mostrados al terminar el día, es decir, son los cálculos finales de cada uno de ellos.

4.4 Registros de las simulaciones realizadas

Es necesario realizar un registro de las acciones del usuario (log de acciones) por completo (indicando cada acción e instrucción dada por él; principalmente cada vez que el usuario escriba algo en la entrada estándar, se debe registrar) Además, se deben registrar las respuestas del sistema en cuanto al resultado de la acción del usuario. Este registro debe tener el formato de día, mes, año, hora, minuto, segundo en donde se realizó la acción, luego dos puntos, y luego la acción realizada por el usuario, esto debe ser escrito en un archivo llamado Log<fecha de la simulación>.log, donde la fecha de simulación está compuesta por el año, mes y día de la simulación. Ejemplos de Log son los siguientes:

1 Archivo: Log20190923.log:

1.1 [23:00:34]: El usuario ha indicado realizar una simulación manual de la parte uno del …..

2 Archivo: Log20180624.log:

2.1 [21:22:54]: El usuario ha indicado.

3 Archivo: Log20171230.log:

3.1 [04:35:01]: El sistema ha realizado la simulación de XXXX, la solución final ha quedado almacenada en el archivo “SalidaParte2_XX.out” y se ha demorado 15 minutos y 43 segundos en realizar el proceso.

4.5 Ingreso de datos al programa

Para el ingreso de datos, existen dos formas, la primera por parámetros al programa y la segunda solicitando los datos al usuario. Ambas deben realizarse según el tipo de datos que se requiera.

4.5.1 Parámetros al programa

Para ejecutar el programa se requieren los siguientes parámetros, los cuales puede solicitar mediante consola o utilizando la función getopt. En caso de utilizar getopt, se tiene los siguientes parámetros de entrada:

- **`-l:`** cantidad de archivos de laberintos que deberán recorrer uno tras otro en orden ascendente.
- **`-a 1|0:`** Si la simulación debe realizarse de forma automática (computador realiza la simulación completa). 1 indica que es automática y 0 que es manual solicitando datos al usuario.
- **-d** `distribucion`: Indica el archivo de distribución a usar.
- -D: Corresponde a debug, es decir, muestra todo el laberinto al usuario pero las personas que se encuentran en el laberinto siguen las mismas reglas descritas en este enunciado y/o alguna otra ayuda que considere necesario para ejecutar el programa y como saltarse pasos o cambiar de mapa, etc. Esta forma es opcional al realizar la simulación. Considere incluir ayuda si corresponde para poder ejecutar con esta opción.

Ejemplo:

```
./proyecto  -D -a 1 -d archDist.txt -l 3  (Linux)

proyecto.exe -D -a 1 -d archDist.txt -l 3 (windows)
```

En donde `proyecto` es el nombre del ejecutable, -D indica que se va a mostrar todo el laberinto al usuario y alguna otra ayuda que usted incluya. -a 1 indica que la simulación debe realizarse de forma automática (opcional), en cambio si -a 0 significa que

no se desea ver mapa. El archivo de distribución de las personas es archDist y finalmente entrega dos laberintos que primero se debe recorrer el lab1 y después el lab2.

En caso de no utilizar getopt, simplemente solicite al usuario la misma información y guarde con scanf lo ingresado.

4.5.1 Solicitud de datos manual

Si el programa debe ejecutarse de forma manual, se deberá solicitar al usuario por pantalla los movimientos y las acciones a tomar como por ejemplo en qué momento ocupar alguna pista descubierta, activar un interruptor, inflar un bote, etc. Las opciones son las siguientes:

- w: mover arriba
- s: mover abajo
- a: mover izquierda
- d: mover derecha
- p: desplegar pista
- e `x.y`: activar/desactivar interruptor en la coordenada `x,y`. Debe encontrarse adyacente a éste (opcional).
- B: muestra el listado de botes disponibles sin inflar, y los que se encuentren inflados con su coordenada correspondiente.
- b `n x.y`: inflar bote `n` de la lista de disponibles anterior (opción B) en la coordenada `x,y`. Este debe ser adyacente en donde se encuentre parado actualmente
- f `x.y nombre_1 nombre_2 … nombre_n`: subir al bote en la coordenada `x,y` las personas indicadas.
- g `x.y nombre_1 nombre_2 … nombre_n`: bajar del bote las personas indicadas en la coordenada `x,y`.

5. Funcionalidades

Para el desarrollo de la simulación a realizar por su equipo de trabajo, se le solicita cumplir con las siguientes características de la implementación y funcionalidades, las cuales son:

1. Se debe construir una solución utilizando el lenguaje de programación C. **(Obligatorio)**

2. La solución debe poder ser ejecutada en sistemas operativos Windows y Linux. **(Obligatorio)**
3. La implementación debe leer los archivos, acorde a lo señalado en el enunciado con los mismos nombres, acorde a lo señalado en el capítulo 4. **(Obligatorio)**
4. La implementación debe entregar un archivo de salida acorde a lo señalado en el punto 4.4. **(Obligatorio)**
5. La implementación debe ser capaz de permitir al usuario realizar las campañas 1,2,y 3 de forma manual **(Obligatorio)** mientras que el automático será **opcional.**
6. El programa debe ser capaz de mostrar por la salida estándar las salidas de las simulaciones de las partes 1, 2 y 3. Es decir debe mostrar cómo avanza el jugador, el menú de las pistas, entre otros. **(Obligatorio)**

Opcionales para optar a bonificación:

1-. El programa en la salida estándar debe mostrar los distintos tipos de terreno con colores distintos.

2-. El programa debe ser capaz de leer archivos con errores de formato o contenido, e informar al usuario de que el archivo está corrupto. Es necesario que el programa siga funcionando.

3-. Todas las funcionalidades opcionales serán bonificadas siempre y cuando se cumpla con las funcionalidades obligatorias.

6. Entregas

La entrega final consta de una aplicación funcional y la documentación necesaria de esta. El sistema creado debe cumplir con todas las funcionalidades de forma completa (siendo un total de 12 funcionalidades) a excepción de aquellas que estén nombradas como opcionales, las cuales significarán una nota extra para las entregas 3.2 y 4.

La entrega del proyecto está dividida por 4 hitos, los cuales poseen una ponderación de 10, 20 30 y 40% respectivamente. La división de las entregas está basada en la estructura de la construcción de un software, las cuales son el análisis del problema, el diseño de la solución y la construcción. La última etapa consta de la mejora del software de las fallas que éste pueda tener.

Cada grupo deberá estar compuesto por **dos alumnos que pertenezcan a la misma sección del laboratorio**. No puede haber grupos compuestos por más

integrantes, a menos que el profesor lo estime conveniente. Esto último implica una mayor exigencia en cuanto a la calidad del trabajo presentado, pero no así en las funcionalidades a realizar. En caso de que un alumno no posea grupo, el profesor lo podrá designar a uno que él encuentre conveniente. En caso de que un alumno desee cambiar de grupo una vez iniciado el proyecto, deberá encontrar un alumno de otro grupo que esté de acuerdo con intercambiar de grupo, en dicho caso el profesor confirmará o rechazará el intercambio entre los grupos.

6.1 Entrega 1

La entrega N°1 consta de la etapa de análisis del problema y el inicio de la etapa de diseño, en donde el grupo de trabajo deberá hacer entrega de:

- **Informe**: El informe debe contener una portada, introducción, descripción del problema, descripción de la solución y conclusiones.
    - **Introducción**: Debe tener un acercamiento para el lector hacia el problema a resolver, además será necesario que realice un marco teórico en el cual se explique los conceptos utilizados para resolver la problemática a desarrollar, dentro de la introducción también deberá indicar la estructura del informe y su organización.
    - **Descripción del problema**: Se debe explicar en qué consiste el problema general y el contexto de éste.
    - **Descripción de la solución**: Se debe explicar cómo se afrontará el problema y la resolución de éste, indicando el método de resolución del problema, su justificación explicando por qué se escoge este método. En caso de existir sub problemas se debe hacer el mismo proceso.
    - **Conclusiones**: Se debe indicar cuales son los resultados del problema y que es lo que queda por realizar para resolver el problema general.
    - Anexo: Como anexo deben entregar el documento relleno correspondiente al desarrollo del trabajo dentro del grupo y los tiempos destinados a éste.
    - Además, el informe deberá contener un índice de contenidos, figuras y tablas.
    - La estructura del informe y el formato estará disponible en el sitio de UdeSantiagoVirtual del curso.

El principal punto que evaluar dentro de este hito es verificar el entendimiento del problema por parte del grupo.

Las entregas de los documentos correspondientes a esta entrega deben ser subidos en el sitio de UdeSantiagoVirtual el **5 de Noviembre del 2021 antes de las 23:55 horas**.

La evaluación de esta etapa está dividida en:

- **Informe**: Tiene un 100% de ponderación para esta entrega. Este tendrá un 10% en formato, 20% ortografía y redacción, y 70% fondo del informe. El informe no debe tener más de 15 páginas de extensión, sin contar los anexos.

Con respecto a la **entrega mínima** de esta etapa corresponde a un informe que abarque todos los contenidos mencionados anteriormente.

6.2 Entrega 2

En cuanto a la entrega 3 se solicitarán esta vez dos elementos, los cuales son una presentación y la aplicación.

La presentación deberá tener como contenidos los siguientes puntos:

- **Introducción**: Debe tener un acercamiento para el lector hacia el problema a los temas tratados dentro de la presentación, haciendo hincapié en los métodos de resolución de problemas, dentro de la introducción también deberá indicar la estructura de la presentación y su organización.
- **Descripción del problema**: Se debe explicar en qué consiste el problema general y el contexto de éste.
- **Descripción de la solución**: Se debe explicar cómo se afrontará el problema y la resolución de éste, indicando el método de resolución del problema, su justificación explicando por qué se escoge este método. En caso de existir subproblemas se debe hacer el mismo proceso.
- **Diseño de la aplicación**: Se debe explicar cómo estará estructurada la aplicación de acuerdo con la modularidad que desee realizar el grupo para desarrollo final. En este punto, el equipo puede utilizar distintos tipos de diagramas para poder mostrar esta información.

- **Descripción de la aplicación**: Se debe mostrar cómo está estructurada la aplicación, además se deben establecer las comparaciones entre el diseño y la construcción final.
- **Conclusiones**: Se debe indicar cuáles han sido los resultados de esta etapa y cuáles son los problemas que posee la aplicación vistos por el grupo de trabajo, además de analizar la aplicación de las estrategias de resolución de problemas en la aplicación, mostrando sus ventajas y desventajas. Otro punto que tendrán que abordar será la ejecución del código de acuerdo con las pruebas realizadas, donde se recomienda analizar sus resultados.
- Las imágenes deben ser legibles dentro de la presentación.

El programa debe cumplir con lo siguiente:

7. Una aplicación desarrollada en C.
8. La aplicación debe estar escrita en el paradigma de Programación imperativo procedural.
9. La aplicación debe poseer un control de errores para las acciones del usuario.

En la presentación el grupo deberá mostrar la aplicación funcionando, descargada desde la plataforma de UdeSantiagoVirtual (Si es que el profesor así lo desea), deben mostrar la ejecución completa de dos funcionalidades, de los cuales uno será seleccionado por el grupo deberá escoger uno e informar antes de la presentación, mientras que el profesor, antes de la presentación escogerá otro. Al momento de la presentación de las funcionalidades el grupo debe mostrar cómo se comunica su aplicación internamente para cumplir con el objetivo. Esta presentación tendrá una duración determinada por cada uno de los profesores.

La entrega de los documentos correspondientes a esta entrega (presentación y código) deben ser subidos en el sitio de UdeSantiagoVirtual. **Con fecha de entrega máxima 7 de Enero del 2021**

La evaluación de esta etapa está dividida en:

- **Presentación**: Tiene un 40%, lo cual está dividida en un 30% en la forma y un 70% del fondo de la presentación.

- **Aplicación**: Tiene un 100% de ponderación, la cual estará compuesta en un 10% en la correcta aplicación de la programación, lenguaje y el orden del código, un 20% de la modularidad del código y un 70% con respecto al funcionamiento de este.

Con respecto a la entrega mínima de esta etapa corresponde a una presentación de la aplicación, la implementación en sí la cual debe al menos compilar y leer el archivo de entrada, mostrando su salida correspondiente.

6.3 Entrega 3

La entrega 3 corresponde a la corrección completa de la entrega 3 acuerdo con los errores indicados por el cuerpo docente.

La entrega de este Hito deberá hacerse por la plataforma de UdeSantiagoVirtual con al menos 24 horas de anticipación a su clase de laboratorio en la semana del 21 de Enero.

Esta entrega posee como entrega la presentación de la aplicación solamente, sin ser necesario un documento de una presentación por separado, es decir, se puede solo mostrar la ejecución de la simulación realizada.

Con respecto a la entrega mínima de esta etapa corresponde a la aplicación en sí la cual debe al menos compilar.

6. Entregas en general

La no entrega de uno de los hitos (evaluativos y formativos), significará la nota mínima directamente para el proyecto.

Todas las entregas poseen una entrega mínima, cuyo incumplimiento significará la nota mínima en el hito. Además de lo indicado en cada uno de los ítems, cada grupo deberá realizar la corrección de la entrega anterior, dejado de esta forma cada ítem con al menos una corrección por parte del profesor.

En caso de existir atraso en la entrega, se aplicará un descuento de 1 punto en la nota final del hito por cada hora de atraso, a contar de 30 minutos después de la hora límite de subida.

En casos de que los links no estén disponibles o la plataforma no esté en funcionamiento, se les solicita enviar los elementos a los profesores y ayudantes correspondientes mediante un correo y subiendo el material a la plataforma LOA correspondiente.