Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC2133 — Estructuras de Datos y Algoritmos
2023 - 2

Tarea 0

Fecha de entrega código: 30 de agosto del 2023 23:59 UTC-4
Link a generador de repos: Generar repo base

Objetivos

- Modelar y diseñar programa para ejecución secuencial de eventos
- Implementar variaciones de distintas estructuras de datos
- Familiarizar al estudiante con el lenguaje de programación C

Material Útil (Cápsulas)

- Repositorio Talleres (Cápsulas)
- Cápsula memoria en C
- Cápsula estructuras básicas en C
- DCCentral de Apuntes - Intro a C

Recomendación inicial

La Tarea 0 del curso tiene como principal objetivo la familiarización con el lenguaje C. Es de mucha importancia realizar esta tarea con cierto grado de dedicación ya que en las próximas tareas se asume que el lenguaje es comprendido.

Además es importante mencionar que la principal dificultad de esta tarea reside en la modelación, por lo que es importante leer el enunciado con calma y es recomendable partir desde un principio con el **documento de diseño**.

Tarea 0
Battle of Data Structures

El pacífico y tecnológicamente avanzado reino del DCChueck era renombrado por su espíritu de innovación y colaboración. Sin embargo, tiempos turbulentos habían caído sobre esta distinguida sociedad.

Lord Farquaad, el gobernante de Duloc, ha llegado a este reino con su ejército de caballeros con la intención de desterrar a todos sus habitantes. Esto se debe a que recientes revelaciones indican que algunos seres y materiales tecnológicos de origen incierto se encuentran entre los residentes del reino, lo que alimenta el profundo odio que Farquaad siente hacia aquellos que no pertenecen a su mundo y lo lleva a tomar medidas tan drásticas.

No obstante, el reino se resiste valientemente y contraataca con sus mejores guerreros: Los profesores de Estructuras de Datos y Algoritmos. Sin embargo, en una maniobriga defensiva que resultó ser una trampa, todo el escuadrón es capturado. Ante esta situación, los ayudantes deciden convocar refuerzos para recuperar a los cautivos y ganar la batalla.

Figura 1: Nuestro valiente equipo de profesores y ayudante de bienestar

Figura 2: El malvado Carlos Paredes

En ese momento llega un antiguo enemigo de Farquaad, nada menos que **Shrek**, acompañado por el increíble **Equipo Alfa Buena Maravilla Onda Dinamita Escuadrón Lobo**. Unidos por un objetivo común, se embarcan en una misión para enfrentar a Farquaad y su ejército, y así liberar a los profesores y proteger la paz y la innovación en el DCChueck.

Problema:

Para comenzar, se tendrán listas con los guerreros de cada ejército, el del **DCChueck** y **Lord Farquaad**, a partir de las cuales se simulará una guerra entre los ejércitos. En un principio, debes procesar el enlistamiento inicial de los guerreros en sus respectivos ejércitos, para luego simular la guerra entre estos hasta que se pida un resultado final.
Para el enlistamiento, la simulación de la batalla y el fin de esta, deberás trabajar a base de los eventos que te entregaremos y que serán explicados de forma oportuna.

Ejércitos

En este conflicto dramático existen dos ejércitos, el ejército del reino **DCChueck** (que tendrá **ID 0**) y el ejército de **Farquaad** (que tendrá **ID 1**). Ambos ejércitos tienen una cantidad definida (y no necesariamente equivalente) de **escuadrones de batalla**.

Escuadrones

Los escuadrones, como todo buen equipo, dividen responsabilidades para lograr conquistar el campo de batalla. Cada **ejército** se compone de una cantidad fija de **escuadrones** diferentes, compuestos a su vez por una cantidad variable de **guerreros**. Los escuadrones poseen un ID relativo a su ejercito numerado entre 0 a $N-1$, con N siendo la cantidad de escuadrones del ejercito correspondiente.

Facciones

Los guerreros dentro de un escuadrón pueden tener distintas especialidades que denominamos **facciones**, estas tendrán un impacto en el desarrollo de la batalla, puesto que afectarán el resultado de los ataques y eventos especiales.

Estas 3 facciones (con sus IDS) son las siguientes:

- `ID 0: CAPASDECEBOLLAS`
- `ID 1: DRAGONERIA`
- `ID 2: YAMERITO`

Guerreros

Los guerreros son la unidad interactiva del campo de batalla, cada uno posee características particulares que determinarán en su conjunto al ejército vencedor. Poseen los siguientes atributos:

- ID: número que identifica a cada guerrero.
- Vida: cantidad de vida que posee.
- Daño Base: poder de ataque base que posee.
- Arsenal: espacio donde guarda artefactos de guerra para aumentar su ataque, este arsenal posee capacidad definida y está inicialmente vacío (El arsenal solamente contiene la cantidad de bonus al ataque).
- Facción: corresponde al ID de la facción a la que pertenece el guerrero: `CAPASDECEBOLLAS (ID 0)`, `DRAGONERIA (ID 1)`, `YAMERITO (ID 2)`.

Adicionalmente, cada guerrero cumple lo siguiente:

- Su ID es único con respecto a los otros guerreros.

- No pertenece a más de una facción.
- No pertenece a más de un escuadrón a la vez.
- No pertenece a más de un ejército a la vez.
- Los atributos ID, Vida y Ataque son números enteros no negativos.

Eventos

Para facilitar la implementación, los eventos estarán divididos en 3 partes. Además, se utilizará un escenario de ejemplo, el cual quedará ilustrado en la siguiente figura.

Disclaimer

La imagen presentada es solo una visualización gráfica del problema y no corresponde necesariamente al **struct** esperado en la implementación de su solución.

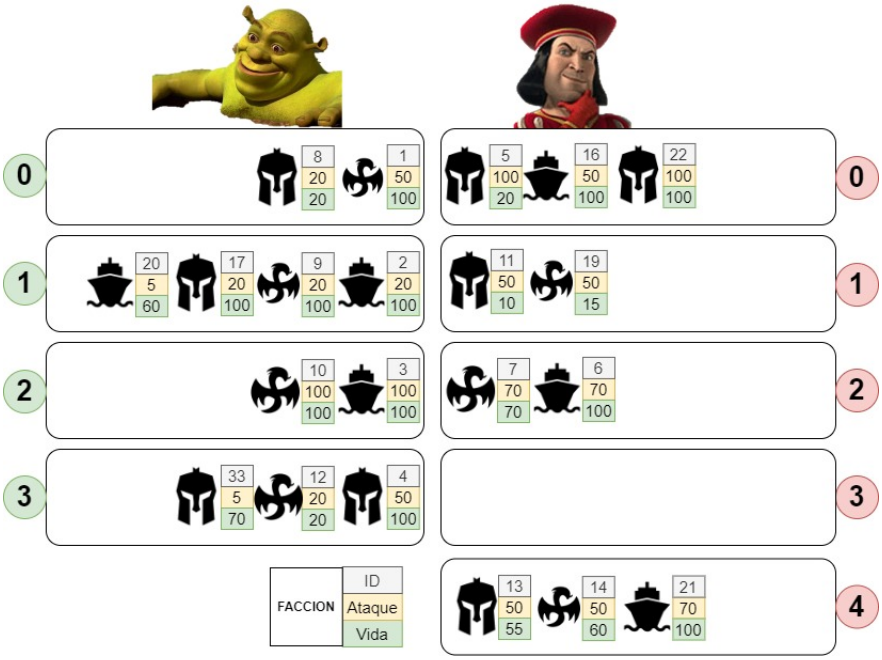

Figura 3: Ejemplo de disposición de las tropas

Para los ejemplos a continuación, supondremos que los cascos representan a la facción `CAPASDECEBOLLAS`, los dragones a las `DRAGONERIA` y los barcos a los `YAMERITO`

Parte A: Creación de Ejércitos

1. `ENLISTAR ejercitoID escuadronID guerreroID dañoBase faccionID capacidadArsenal`

El evento registra un nuevo guerrero en un **ejército** específico. El enlistamiento se debe realizar al final del escuadrón en posición `escuadronID`.
Luego de generar el enlistamiento, se deberá imprimir (en un archivo output) el ID del guerrero agregado en el siguiente formato:

```
ENLISTADO guerreroID escuadronID
```

Notar que **el orden de los escuadrones está dado por el orden de los enlistamientos**, es decir, el primer enlistado es la cabecilla del escuadrón y el último enlistado en ese escuadrón corresponde a la cola de este.

2. `EQUIPAR bonusArtefacto escuadronID guerreroID`

Este evento simula la acción de equipar un artefacto en el arsenal de un guerrero. Esto asignando el valor del bonus en la primera posición de arsenal disponible. Se deberá imprimir (en un archivo output) lo siguiente en caso de que el guerrero haya podido agregar de forma correcta su artefacto al arsenal:

```
ARTEFACTO artefacto ASIGNADO A guerreroID
```

En caso de que el arsenal ya esté completo, no se debe realizar el equipamiento del artefacto y se debe imprimir (en un archivo output) lo siguiente:

```
ARSENAL LLENO guerreroID
```

Hint: Notar que el arsenal aloja una cantidad definida de bonus de ataque, no se debe sobreescribir la propiedad del guerrero de ataque base.

3. `CONTEO-CAMPAL`

Este evento entrega la información actual del campo de batalla, de forma más específica, entrega la cantidad de guerreros que tiene cada facción de cada ejército. Por ejemplo, un resultado es el siguiente: (*para el print se deben considerar los mismos espacios de indentación o tabs (\t)*)[1]

```
CONTEO-CAMPAL
    EJERCITO 0
        FACCION 0: 4
        FACCION 1: 4
        FACCION 2: 3
    EJERCITO 1
        FACCION 0: 4
        FACCION 1: 3
        FACCION 2: 3
TOTAL DE GUERREROS : 21
```

[1] Por ejemplo `EJERCITO 0` posee 1 tab de identación

Parte B: Orden y Batalla

1. `ORDEN-EJERCITO ejercitoID`

Esta acción permite **IMPRIMIR** en orden los escuadrones con mayor cantidad de guerreros (en orden decreciente) de un ejército específico. El resultado se debe imprimir (en un archivo output) de la siguiente manera:

```
EJERCITO ejercitoID
    ESCUADRON escuadronID cantidadGuerreros
    ...
```

En caso de que dos escuadrones tengan la misma cantidad de guerreros, deberá ir primero aquel que tenga **menor ID**. Por ejemplo, dado el siguiente comando:

```
ORDEN-EJERCITO 1
```

y dada la situación ejemplificada en la figura 1 se entregaría lo siguiente:

```
EJERCITO 1
    ESCUADRON 0 3
    ESCUADRON 4 3
    ESCUADRON 1 2
    ESCUADRON 2 2
    ESCUADRON 3 0
```

El `1` de `EJERCITO 1` indica que debemos revisar el **ejercito de Farquaad**. El escuadrón 0 y 4 tienen misma cantidad de guerreros, pero como el escuadrón 0 tiene menor ID, se escribe este primero. La misma lógica aplica para el empate entre los escuadrones 1 y 2. Finalmente, el escuadrón 3 queda para el final al no tener guerreros.

Importante: Este evento no deberá afectar el resto del funcionamiento del programa, es decir, no reordena los escuadrones, solo imprime según el criterio de orden.

2. `DESERTAR ejercitoID escuadronID guerreroID`

Este evento genera que un guerrero deserte la batalla. Considerar que siempre se preguntará por un guerrero que está activo (existe) en la batalla.
Como respuesta a este evento se debe imprimir (en un archivo output):

```
ABANDONA guerreroID ejercitoID
```

Siendo el **ejercitoID**, el identificador del ejercito al cual pertenecía en ese momento el guerrero, y subsecuentemente **debe ser eliminado**.

3. `ATACA ejercitoID`

Esta acción permite que la cabecilla[2] de un escuadrón de un ejército ataque a la cabecilla de un escuadrón del ejército enemigo. El escuadrón atacante será el que tenga el **mayor valor de ataque base sumado entre sus guerreros**. Se realizará el ataque sobre el escuadrón enemigo que tenga **el menor número de miembros, distinto de 0**. En caso de empate, ya sea buscando al escuadrón atacante o defensor, siempre se debe priorizar al escuadrón de **menor ID**.
Luego de elegir al atacante, se debe **sumar el mejor bonus de ataque de su arsenal**, en caso de empate se escoge el primero, y **el bonus asociado a la facción que pertenece** (Ver tabla). Entonces, el daño recibido por el defensor es equivalente a la suma:

[2]Cabecilla == Primer miembro del escuadrón

```
DANO-BASE-GUERRERO + DANO-MEJOR-ARTEFACTO + DANO-FACCION
```

Al finalizar el ataque **se debe eliminar el artefacto** utilizado del arsenal del guerrero (Dejando el espacio disponible) y el guerrero **se moverá al final del escuadrón**, siendo el segundo guerrero del escuadrón la nueva cabecilla (se omite este paso si es el único guerrero en el escuadrón).

Tablas de Bonus de ataque por Facción

Facción	Bonus
(ID 0) Capas de Cebollas	5 ptos.
(ID 1) Dragonería	10 ptos.
(ID 2) Ya Merito	15 ptos.

Por ejemplo, dado la situación en la figura 1, si se utiliza el siguiente evento:

```
ATACA 0
```

El ejército atacante sería el de Shrek y el defensor el ejército de Farquaad. Además, se tiene que el escuadrón atacante sería el número 2, dado que la suma de ataque total es la mayor en comparación a los demás escuadrones, mientras que el escuadrón defensor sería el número 1 de Farquaad, dado que este tiene el menor número de miembros distinto de 0 (y tiene menor ID en comparación al escuadrón 2).

Luego, para la ejecución del ataque, los guerreros que se encuentran al principio de cada escuadrón son los responsables de luchar. Es decir, al primer guerrero del escuadrón defensor se le restan *<Daño Total Atacante>* puntos de vida, según el cálculo mencionado anteriormente. **En caso de que el ataque sea mayor o igual a la vida, el guerrero defensor muere y es eliminado** (Equivalente a la operación desertar) . En caso contrario, se debe actualizar su vida.

Finalmente, luego del ataque se debe actualizar la formación del escuadrón atacante, moviendo al primer guerrero a la posición última del escuadrón, generando que el anterior segundo sea la nueva cabecilla.

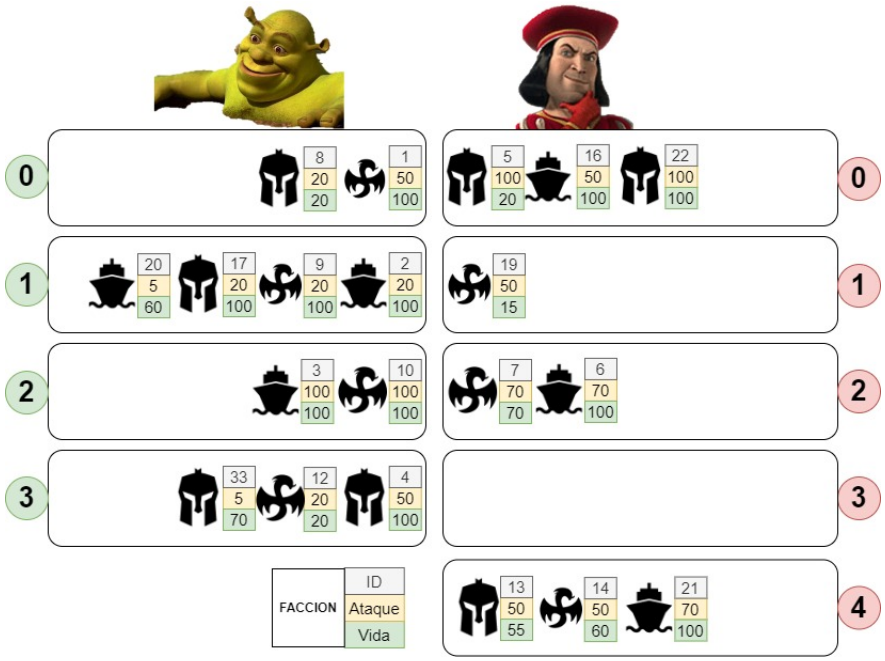

Figura 4: Ejemplo de Post ataque: Notar que muere el guerrero 11 y que el guerrero 3 se va al final de su escuadrón.

Finalmente, como resultado de este evento deberás imprimir (en un archivo output) lo siguiente:

`ATACA guerreroID victimaID`

Parte C: Comandos especiales

1. `TRAICIONAR ejercitoID escuadronID guerreroID`

Este evento genera un cambio de bando para el guerrero, es decir, que abandona a su escuadrón y se pasa al escuadrón de **mayor número de miembros** del ejército oponente. En caso de empate de miembros, se traspasa al escuadrón de menor ID.
Adicionalmente se tiene que imprimir (en un archivo output) lo siguiente:

`TRAICION ejercitoID`

Donde el ejercitoID corresponde al ejército al cual se ha traicionado (abandonado).

2. `ALTA-TRAICION ejercitoID escuadronID guerreroID`

Este evento genera un cambio de bando para el guerrero y sus seguidores, es decir, **desde el guerrero señalado hacia atrás** pasan al escuadrón de mayor número de miembros del ejército oponente, **en el mismo orden en que estaban originalmente y enlazados en la cola del escuadrón que los recibe**. En caso de empate de miembros, se debe traspasar al escuadrón de menor ID.
Se debe imprimir lo siguiente:

`ALTA-TRAICION ejercitoID`

Donde el ejercitoID corresponde al ejército al cual se ha traicionado (abandonado).

Ejecución

Tu programa se debe compilar con el comando `make` y debe generar un ejecutable de nombre `executable` que se ejecuta con el siguiente comando:

```
./dcchuek input.txt output.txt
```

Donde input será un archivo con los eventos a simular y output el archivo donde se deben guardar los resultados. Tu tarea será ejecutada con archivos de creciente dificultad, asignando puntaje a cada una de estas ejecuciones en las que tenga un output igual al esperado. A continuación detallaremos los archivos de input y output.

Input

El archivo de input comenzará con el número de escuadrones del ejército de Shrek `S`, seguido en la siguiente línea por el número de escuadrones del ejercito de Lord Farquaad `F`. La siguiente línea corresponde a un número `E` que indica el número de eventos a recibir (esto incluye equipación de artefactos, enlistar, atacar, desertar, etc.). El input se vería con el siguiente formato:

```
S
F
E
ENLISTAR 0 0 12 0 CAPASDECEBOLLAS 3
CONTEO-CAMPAL
ENLISTAR 1 0 50 1 DRAGONERIA 2
ENLISTAR 5 1 100 2 CAPASDECEBOLLAS 9
ATACA 0
...
```

Para el ejemplo de las imágenes, podemos ver que $F = 5$ y que $S = 4$

Output

Tu output debe consistir de un archivo con la información solicitada por cada evento en el archivo de input.

Código Base

Se entregara un código base conteniendo lo necesario para leer los eventos y funcionar bajo el esquema del curso. Es importante recalcar que modificaciones mayores en el archivo `Makefile` podrían causar que el programa falle en la instancia de evaluación.
Puedes encontrar el generador de repositorio en el siguiente enlace: https://classroom.github.com/a/7feYdEVB

Documento de diseño

La tarea contempla un documento de diseño que posee ponderación en la nota. En particular el documento es evaluado de forma binaria. Asignando puntaje completo en caso de realizar las tres secciones, y ningún puntaje en caso contrario.
La idea del documento son varias. Primero es incentivar el análisis del enunciado y del problema en el inicio. Es recomendable realizar al menos el 50 % del documento antes de empezar a programar.

Por otro lado la idea es permitir realizar recorrecciones manuales de código. Donde se asigna puntaje parcial a la implementación realizada.

Importante: En caso de no realizar el documento de diseño NO se podrá optar a recorrección manual

Contenidos mínimos

El documento de diseño ha de ser llenado en el template `docs/design.md` ubicado en el repositorio base de la tarea.
No se espera que el documento sea formal, si no que se espera que queden como registros del proceso de diseño de su solución. Es por esto que ha de contener como minimo las siguientes secciones:

1. Análisis del enunciado: Breve análisis identificando las estructuras y sus relaciones.
2. Planificación de solución: una idea a rasgos generales sobre como sera abordado el problema, identificando que Estructura de datos se utilizara.
3. Justificación: Sección donde se justifican las decisiones de diseño consideradas para la implementación de la tarea

Entrega documento de diseño

El documento de diseño se entrega en conjunto con el repositorio de la tarea, el contenido principal ha de ir en el archivo `docs/design.md`. En caso de querer dejar anexos como imágenes o diagramas, han de ser añadidos en la misma carpeta.

Cuestionario

La tarea además contemplara un cuestionario (De alternativas y desarrollo) donde se evaluarán los contenidos teóricos y la comprensión del problema. Por ejemplo para esta tarea (T0) algunos de los contenidos a abordar son:

- Estructuras basicas
- Diferencia entre Stack y Heap
- Conceptos del lenguaje C
- Principios de Sorting

El cuestionario será anunciado en los próximos días igualmente que su plazo de respuesta.

Evaluación y Calculo de Nota

La tarea contempla dos secciones principales que contribuyen a la nota. La sección practica, compuesta por resolver correctamente los tests y la parte teorica.

Parte Practica (70 %)

Para cada test de evaluación, tu programa deberá entregar el output correcto en menos de 10 segundos y utilizar menos de 1 GB de ram[3]. De lo contrario, recibirás 0 puntos en ese test.
En particular se realizaran **6** tests por sección (A, B, C). Considerando la siguiente ponderación

- Parte A: 25 %
- Parte B: 35 %
- Parte C: 30 %
- No Leaks de Memoria: 5 %
- No Errores de Memoria: 5 %

Además la nota individual de cada parte se evalúa con la cantidad de tests correctos más 1.
En el caso de leaks y errores. Puedes chequear que tu programa no los posea utilizando la herramienta `valgrind`

Parte Teorica (30 %)

La parte teórica contempla el documento de diseño y el cuestionario.
Documento de diseño (20 %): Su evaluación es binaria. Ósea, obtiene puntaje máximo en caso de realizarlo completo y puntaje mínimo en caso contrario. Solo se evalúa la presencia de las secciones, no es importante que sea detallado.

Cuestionario (80 %): El cuestionario posee un total de 25 puntos obtenibles, de los cuales se consideran 20 para la nota máxima. Su formula es la siguiente:

$$nota = 6 \cdot \frac{min(puntaje, 20)}{20} + 1$$

Resumen evaluación

Por ende la nota final de la tarea se calcula como

- 70 % de la nota obtenida en código. Considerando
  - Parte A: 25 %
  - Parte B: 35 %
  - Parte C: 30 %
  - No Leaks de Memoria: 5 %
  - No Errores de Memoria: 5 %
- 30 % de la nota de parte teórica considerando
  - Documento de diseño: 20 %
  - Cuestionario: 80 %

[3]Puedes revisarlo con el comando htop o con el servidor

Recomendación de tus ayudantes

Las tareas requieren de mucha dedicación de tiempo generalmente, por lo que **desde ya** te recomendamos distribuir tu tiempo considerando los plazos definidos. Así mismo, te recomendamos fuertemente que antes de empezar a programar tu tarea, leas el enunciado y te dediques a entender de manera profunda lo que te pedimos. Una vez que hayas comprendido el enunciado, dedica el tiempo que sea necesario para la planificación y modelación de tu solución, para posteriormente poder programar de manera eficiente. Estos son consejos de tus ayudantes que te pueden ayudar a pasar el ramo :)

Entrega

Código: GIT - Rama master del repositorio asignado. Se entrega a más tardar el día de entrega a las 23:59 hora de Chile continental.

Política de atraso: Para esta tarea no aplica la política de atrasos del curso

Integridad académica

Este curso se adscribe al Código de Honor establecido por la Escuela de Ingeniería. Todo trabajo evaluado en este curso debe ser hecho **individualmente** por el alumno y **sin apoyo de terceros**. Se espera que los alumnos mantengan altos estándares de honestidad académica, acorde al Código de Honor de la Universidad. Cualquier acto deshonesto o fraude académico está prohibido; los alumnos que incurran en este tipo de acciones se exponen a un Procedimiento Sumario. Es responsabilidad de cada alumno conocer y respetar el documento sobre Integridad Académica publicado por la Dirección de Pregrado de la Escuela de Ingeniería.