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

IIC2133 — Estructuras de Datos y Algoritmos
2022 - 1

Tarea 0

Fecha de entrega código e informe: Lunes 28 de Marzo del 2022.

Información General

La siguiente tarea contempla como objetivo principal la introducción al lenguaje C. En el cual se espera que apliquen los conceptos de *Structs*, *Punteros*, *Arrays* y *Listas ligadas*.
Recordar que existe el discord del curso y las issues del repositorio. Donde el equipo de ayudante estará atento para resolver dudas con respecto a la tarea.

Objetivos

- Comprender las diferencias entre arreglos y listas ligadas e implementar estas estructuras.
- Habituarse al uso de punteros, junto al manejo de memoria.
- Familiarizarse con el modelado de problemas.
- Aplicar y analizar conceptos con respecto a complejidad.

Introducción

Tu gran amigo de la infancia, DCChef Remy, decidió hace un par de años abrir su propio restaurant (La Ratatouille) a metros del campus Juan Soaquín, esperando tener una clientela gigantesca considerando el tamaño del estudiantado de la universidad. Lo que menos podía esperar Remy, fue la repentina llegada de la pandemia, situación que no le permitió tener mucho éxito a su local. Sin embargo, con la ansiada vuelta a la presencialidad, llegó un gigantesco boom de clientes, disparando el restaurant al estrellato. Tanto así, que estudiantes de otros campus empezaron a cambiarse de carrera solo para poder estudiar en el campus al lado de La Ratatouille, y disfrutar de sus prestigiosas (y baratas) comidas.

Viendo esta situación, la mentalidad de tiburón de DCChef Remy vió la oportunidad de negocio (a pesar de ser una rata), y decidió abrir un restaurante al lado de cada campus de la universidad, creando así una cadena de restaurantes y cumpliendo el sueño de su vida.

Para llevar a cabo la apertura de todos los restaurantes, necesita de más personal que lo ayude a manejar los restaurantes y su enorme clientela. Es por esto que DCChef Remy te contacta, y como eres estudiante y te encanta la comida de La Ratatouille, decides ayudarlo.

Figura 1: Logo del restaurant

Problema

El problema consiste en atender a los clientes que asistan a los restaurantes. Para esto deberás modelar el funcionamiento de todos los restaurantes según los eventos mencionados en la sección de EVENTOS.
Sobre el modelamiento base del problema debes considerar lo siguiente. La cadena de restaurantes consta de N ubicaciones, donde cada ubicación tiene una cantidad arbitraria m_i de mesas. Cada mesa puede estar abierta o cerrada, donde al momento de abrir una mesa se le asignará una capacidad c_i de clientes que puede acomodar. Ten en consideración que una mesa abierta puede no tener todos sus puestos ocupados.
Además, al ser una cadena de restaurantes, la carta es compartida. Por lo que basta solo con una carta que servirá para todas las ubicaciones.
Finalmente es importante notar que el sistema debe manejar las cuentas por persona y no por mesa, esto se explica más adelante en la sección de pedidos.

Eventos

Para simplificar los procesos el DCChef Remy separó los procesos en tres etapas de dificultad incremental.

A. Apertura de Restaurant (20%)

Esta etapa contempla todo lo que es la apertura de mesas y el flujo de clientes en un restaurante.
Los eventos a considerar para esta etapa son los siguientes:

1. `OPEN-TABLE locationId tableId capacity`

El evento abre una nueva mesa en la ubicación *locationId* que puede recibir *capacity* clientes. Se les deberá asignar un *id* que permitirá realizar consultas. Además todos los asientos de la mesa comienzan vacíos.

2. `MENU-ITEM itemId price`

Este evento registra un plato en el sistema de carta, con un *id* y un precio. Ambos son números enteros.

3. `CUSTOMER locationId tableId customerId`

Este evento indica que un cliente de id *customerId*, ha llegado a la mesa *tableId* ubicado en la ubicación *locationId* de la cadena de restaurantes. El cliente ha de ser sentado en el primer asiento disponible en dicha mesa. Y en caso de que no existan espacios, se ha de imprimir el mensaje `TABLE locationId:tableId FULL` y se debe eliminar al cliente en caso de que este fuera inicializado.
El ID del cliente es Universal para la aplicación.

4. `TABLE-STATUS locationId tableId`

Se ha de imprimir el estado de la mesa solicitada en el siguiente formato

```
ESTADO MESA locationId:tableId
    CAPACIDAD TOTAL
    CAPACIDAD RESTANTE
    CLIENTES
        ID_CLIENTE
        ID_CLIENTE
        -
        -
        ...
```

```
FIN ESTADO
```

Se ha de imprimir _ en caso de que algún asiento no este utilizado.

Parte B. Manejar Pedidos (40%)

Esta etapa contempla todo lo que son pedidos y las cuentas por mesa. Como recomendación analiza bien como modelarás el siguiente funcionamiento.

1. `ORDER-CREATE locationId tableId customerId itemId`

Se agrega un plato a la cuenta de una **persona**. El plato se agrega a la persona, no a la mesa en la que se encuentra (Hint: Un usuario puede pedir todos los platos que quiera)

2. `ORDER-CANCEL locationId tableId customerId`

Se elimina el último pedido desde la cuenta de la persona. Y se ha de imprimir el ID del elemento eliminado de la siguiente forma

```
DEVOLUCION itemID
```

3. `BILL-CREATE locationId tableId`

Se genera una boleta para la mesa, de la siguiente forma:
Por cada usuario en la mesa en el orden de los asientos

```
BOLETA MESA locationId:tableId
    PEDIDOS
        PERSONA customerId
            itemId price
            ...
        gastoTotalPersona
        FIN PERSONA
        PERSONA customerId
            ...
        FIN PERSONA
    FIN PEDIDOS
    EL CLIENTE customerId PIDIO MAS PLATOS
    EL CLIENTE customerId GASTO MAS
    EL CLIENTE customerId GASTO MENOS
    TOTAL precioTotal
FIN BOLETA
```

En caso que exista un empate entre los cliente se escoge el cliente con menor `customerId`.

Parte C. Manejar Restaurant (40%)

A veces los clientes que van al restaurant realizan peticiones muy particulares y ponen a prueba el sistema para ver cómo es su atención al cliente, por esto tienes que realizar un programa robusto para poder cumplir con sus locas ideas y dar la mejor atención al cliente. En esta sección te pueden pedir intercambiar asientos entre mesas o entregar platos de una mesa a otra.

1. `CHANGE-SEATS locationId, tableIdX, customerIdX, tableIdY, customerIdY`

En este evento tendrás que poner al customerIdX en la posición del customerIdY, por su parte el customerIdY irá en la posición del customerIdX.

2. `PERROU-MUERTO locationId tableId customerId`

Uno de los integrantes de la mesa estaba apurado y se retiró antes de pagar la cuenta. Como el restaurante no puede permitir regalar comida, la cuenta y los platos pedidos de dicho cliente se anexarán al primer cliente de dicha mesa.

Condiciones/Supuestos

Algunas de las condiciones se podrían tomar como casos borde para los test

- Todas los numeros presentados seran enteros
- Para eliminar un plato DEBE existir uno antes
- Todas las operaciones seran validas. (A excepcion de la llegada de un cliente que posee dicho caso)

Ejecución

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

```
./ratatouille input.txt output.txt
```

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

Archivo Input

El archivo comenzará indicando el número de restaurantes N. Seguido por N líneas indicando el número de mesas en cada uno $m_1, m_2, \ldots, m_n$. Posteriormente se iniciará la sección de apertura de mesas. Se indicará el número S de mesas a abrir seguido por S líneas, con las instrucciones de apertura para cada una.
Luego se indicará el M número de platos a existir, seguido por el evento de creación para cada plato.
Finalmente hay un número arbitrario de líneas con eventos a realizar, escritos tal como se especificaron en la sección Eventos del enunciado.
El archivo termina con la secuencia END que indica el final del programa. En ese momento debes imprimir el estado de todas las mesas (Ejecutar evento ESTADO MESA para todas)

```
N
n_1
n_2
n_3
...
S
OPEN-TABLE locationId tableId capacity
OPEN-TABLE locationId tableId capacity
...
M
MENU-ITEM itemId price
MENU-ITEM itemId price
...
CUSTOMER locationId tableId customerId
CUSTOMER locationId tableId customerId
ORDER-CREATE locationId mesaId customerId itemId
TABLE-STATUS locationId tableId
...
END
```

Archivo Output

Como archivo de output se solicita que se vaya escribiendo el estado de la aplicación según se solicita. Además luego de finalizar la ejecución de los eventos, se ha de imprimir el estado general de todos los restaurantes, ejecutando `ESTADO MESA` para cada mesa de cada establecimiento (respetando el orden de id de restaurantes y mesas).

Informe

Deberás escribir un informe de análisis [1] donde menciones los siguientes puntos:

- Describe y justifica la estructura de datos usada para cada elemento (mesa, persona, cuenta, etc.).
- Calcula y justifica la complejidad en notación $\mathcal{O}$ para cada uno de los eventos del programa.
- Comentar al menos 3 ventajas y desventajas de tanto listas ligadas y de arrays.
- Investigar y comentar ventajas de C versus Python. (Principalmente orientado a la velocidad de cada uno)

Nota Importante: El informe es totalmente independiente del código, y por ende puedes entregarlo aunque no hagas entrega de código. Además de que todos los cálculos y/o análisis pueden ser teóricos sin ninguna relación a tu implementación.

Archivos iniciales

Para ayudarte a empezar a programar y que te centres en la creación de estructuras y sus funciones, los ayudantes han creado un código base que incluye la lectura de archivos e interpretación de cada evento. Este se encuentra en tu repositorio cuyo nombre es T0-2022-1-usuario-github. Recuerda aceptar la invitación cuando te llegue ya que esta vence luego de 7 días.

[1] De no más de 6 planas

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 :)

Uso de memoria

Parte de los objetivos de esta tarea, es que implementen en la práctica el *trade-off* entre memoria y tiempo, es por esto que independiente del test, tendrán como máximo `1.2 GB` de memoria `RAM` disponible. Pueden revisar la memoria que utiliza su programa con el comando `htop`. Además, el servidor avisará en caso de superar el máximo permitido.

Eficiencia

Se solicita que el programa sea eficiente. No podrá tomar más de 10 segundos `user + sys`, pueden utilizar `time` seguido del comando de ejecución de su programa para revisarlo.

Evaluación

La nota de tu tarea se descompone como se detalla a continuación:

- 70% a la nota de tu código, dividido en:
  - 90% que el output de tu programa sea correcto y eficiente. Para esto el puntaje se divide en partes iguales para los tests Easy, Medium y Hard.
  - 5% a que *valgrind* indique que tu programa no tiene errores de memoria.
  - 5% a que *valgrind* indique que tu programa no tiene *memory leaks*.

  Para cada test de evaluación, tu programa deberá entregar el output correcto en menos de 10 segundos. De lo contrario, recibirás 0 puntos en ese test.
- 30% a la nota del informe

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.

Informe: CANVAS - En el cuestionario correspondiente en formato .pdf Sigue las instrucciones del cuestionario. Se entrega a más tardar el día de entrega a las 23:59 hora de Chile continental. En caso de no seguir las instrucciones de formato, Se realizara descuento
Atraso: A lo largo del semestre tendrás 4 días de gracia, los cuales podrás utilizar en caso de que no alcances a entregar alguna tarea en el tiempo indicado. Para esto, puedes usar un máximo de 2 días en una tarea, sin importar si tienes más días disponibles, y tendrás que avisar si quieres que se revise un commit posterior a la fecha de entrega en el formulario que se mandará el día siguiente. Cabe destacar que si entregas a las 00:01 hrs perderás un día en caso de llenar el formulario, y no será revisado ese commit en caso de que decidas

no contestarlo. Por otro lado, si se te acaban los 4 días y entregas una tarea atrasada, entonces tendrás la calificación mínima **sin derecho a reclamo.** Cabe recalcar que el informe solo se puede realizar en un **máximo de 6 hojas**, de pasarse no podrán objetar por posibles descuentos.

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.

¡Éxito! :)