Universidad Católica del Norte
Facultad de Ingeniería y Ciencias Geológicas
Departamento de Ingeniería de Sistemas y Computación

ESTRUCTURAS DE DATOS

☆ Profesora: Angélica Flores.

☆ Ayudantes: Francisco Llanos, Benjamín Miranda.

Taller N°1: C++ (II-2024)
Periodo: 26 de agosto al 29 de septiembre

I. Objetivo

Implementar arreglos estáticos/dinámicos y listas enlazadas utilizando punteros en el lenguaje de programación C++.

II. Enunciado

BuildTheEarth es un proyecto que busca recrear todo el planeta Tierra dentro del videojuego Minecraft. Debido a la extensión de la misión, los integrantes se han organizado de manera que cada país se encargue de recrear su territorio. BuildTheEarth Chile (ver Figura 1) es una subdivisión de BuildTheEarth que busca construir Chile en Minecraft. Iniciaron en abril del año 2020 y desde ese entonces no han dejado de construir localidades del país en el videojuego (ver Figura 2). Ellos mencionan que cualquier jugador puede contribuir construyendo alguna localidad que conozca del país y así, con la ayuda de todos, lograr finalizar el proyecto.

Figura 1. Logo BuildTheEarthChile

Figura 2. Costanera Center en Minecraft

Cuando un jugador desea contribuir, debe solicitar un área específica de terreno para que se le otorguen permisos de construcción. Esto es conocido como “Sistema de proyectos”, donde la administración tiene la facultad para aprobar o rechazar la solicitud.

Este último año, BuildTheEarth Chile ha querido mejorar los procesos de solicitudes de construcción y manejo de proyectos. Es por esto por lo que se han comunicado con el Departamento de Ingeniería de Sistemas y Computación de la UCN, para la elaboración de un programa que permita manejar de forma eficientes lo requerido.

Como entrada, BuildTheEarth Chile proporciona el archivo “solicitudes.csv” con todas las solicitudes hechas por usuarios para contribuir al proyecto (ver Figura 3). Posee el siguiente formato:

- Nickname: Nombre del jugador, es único para cada uno.
- Dificultad: Dificultad establecida para completar ese proyecto. Pueden ser: PRINCIPIANTE, FACIL, INTERMEDIO, DIFICIL y MUY_DIFICIL.

- Puntos: Cantidad de puntos que se le otorgaran al jugador luego de completar el proyecto.
- Fecha: La fecha en que se realizó la solicitud, en formato DD-MM-YYYY.
- Ciudad: La ciudad donde se realizará la construcción.
- Descripción: Una breve descripción de lo que se construirá.

```
Nachoo197;MUY_DFICIL;5000;23-06-2024;Santiago;Construcción del aeropuerto internacional
FranVG;PRINCIPIANTE;2500;07-07-2024;Santiago;Construcción de una casa pequeña en estación central
Angelo281;DIFICIL;2492;20-08-2024;Santiago;Mejoras en la construcción de la moneda
LlanosP;INTERMEDIO;5500;30-08-2024;Antofagasta;Construcción de la UCN
FernandChav;MUY_DIFICIL;9200;02-06-2021;Calama;Construcción del estadio de Calama
Francesca2000;DIFICIL;5400;05-06-2024;Iquique;Construcción del centro de Iquique
ManuelJara;FACIL;2800;22-08-2024;Osorno;Supermercado en osorno
```

Figura 3. Ejemplo archivo "solicitudes.csv"

También se tiene el archivo "proyectos.csv" con todos los proyectos registrados (ver Figura 4). Contiene los siguientes campos:

- Id: Identificador del proyecto, se compone de 6 letras mayúsculas.
- Nickname: Nombre del jugador propietario del proyecto, es único para cada jugador.
- Fecha: La fecha en que se aceptó la solicitud, en formato DD-MM-YYYY.
- Descripción: Una breve descripción de lo que se construirá.
- Dificultad: Dificultad establecida para completar ese proyecto. Las opciones son las mismas que en "solicitudes.csv".
- Finalizado: Indica "Si" si el proyecto está finalizado, y "No" en el caso contrario.

```
MMGZKM;JuanG;03-07-2024;Construcción de una casa en Arica;FACIL;No
RJSBHS;VX_s;05-05-2023;Construcción del aeropuerto de Antofagasta;DIFICIL;Si
LSHBCG;MartinUwU;29-01-2022;Construcción de calles en putaendo;INTERMEDIO;Si
DDDHFH;Marcelonko;21-03-2024;Construcción de avenida principal de Villarica;INTERMEDIO;No
```

Figura 4 Ejemplo archivo "proyectos.csv"

Cuando el programa inicia debe leer ambos archivos, guardar los datos de proyectos en un arreglo dinámico, y las solicitudes en una lista con nexo. Se deben ingresar a la lista con nexo considerando que las más antiguas son las que estarán al inicio y las más nuevas al final; la Figura 5 presenta un ejemplo visual.

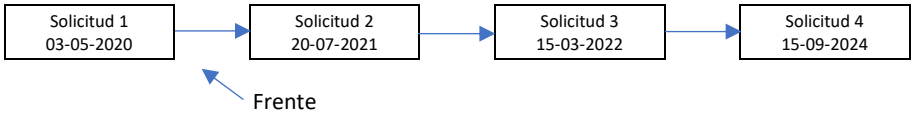

Figura 5. Ejemplo de lista con nexo

Luego, el programa debe mostrar un menú que permita interactuar con las siguientes opciones (ver Figura 6):

```
[<---_---- BuildTheEarth Chile ---_---->]
[A]. Revisar solicitudes
[B]. Búsqueda de proyectos
[C]. Volver a evaluar el proyecto
[D]. Estadísticas
[E]. Salir
[<---_---- -------------------- ---_---->]
5
```

Figura 6. Ejemplo del menú inicial

- **[A] Revisar solicitudes:** muestra información de la primera solicitud en la cola:
  - Nickname del usuario que envío la solicitud.
  - Dificultad.
  - Puntos.
  - Descripción.
  - Cantidad de días que han pasado entre que se envió la solicitud y el momento que se obtiene la información.

  Luego, se debe desplegar un submenú con las siguientes opciones:

  - Aceptar solicitud: Elimina la solicitud de la cola y se registra en un nuevo proyecto en el arreglo dinámico. Como información del nuevo proyecto, se debe considerar:
    - Id: Generado de forma aleatoria considerando que está compuesto por 6 letras mayúsculas.
    - Nickname: nombre del jugador autor de la solicitud.
    - Fecha: La fecha actual (DD-MM-YYYY).
    - Descripción: Descripción de lo que se construirá.
    - Dificultad: Dificultad establecida en la solicitud.
    - Finalizado: "No".

    Al finalizar, se debe mostrar la siguiente solicitud de la lista con nexo y abrir el submenú de acciones a realizar sobre ella, tal y como se hizo al inicio de [A].
  - Rechazar solicitud: Elimina la solicitud de la lista con nexo y se muestra la siguiente solicitud.
  - Siguiente la solicitud: Muestra la siguiente solicitud en la lista con nexo, con la misma información y opciones. Si no hay más elementos, debe llevar al menú principal.
  - Salir: Vuelve al menú inicial.

- **[B] Búsqueda de proyectos:** Para la administración es importante llevar un control de todos los proyectos existentes, es por eso que el sistema debe permitir realizar búsquedas avanzadas aplicando varios filtros. Los filtros existentes son los siguientes:
  - Filtro 1: Muestra los proyectos que han sido finalizados.
  - Filtro 2: Muestra los proyectos cuya solicitud se aceptó en cierto año ingresado.
  - Filtro 3: Muestra los proyectos cuya solicitud se aceptó en cierto mes ingresado.
  - Filtro 4: Muestra los proyectos asociados a un Nickname ingresado.
  - Filtro 5: Muestra los proyectos asociados a una dificultad ingresada.

  Cuando el usuario interactúa con esta opción se le pregunta qué filtros desea aplicar (puede no aplicar filtros o agregar más de uno simultáneamente), y, a partir de ello, despliega la siguiente información asociada a cada proyecto (ver Figura 7):
  - Nombre: Nickname del jugador propietario del proyecto.
  - Dificultad: Nivel asignado al proyecto en formato 'O':
    - Principiante: O
    - Fácil: OO

        - Intermedio: OOO
        - Difícil: OOOO
        - Muy Difícil: OOOOO
    - Semanas: Número de semanas que han pasado entre que se aceptó el proyecto y el día en que se realizó la consulta.
    - Descripción: Descripción del proyecto, solo se deben mostrar los primeros 50 caracteres de está.

```
[.......... Proyectos encontrados ..........]
-----------------------------------------
Nombre: iBxnjaDev
Dificultad: OO
Semanas: 3
Descripcion: Construccion en Antofagasta
-----------------------------------------
-----------------------------------------
Nombre: fabianV
Dificultad: OOOO
Semanas: 10
Descripcion: Construccion de puerto
-----------------------------------------
```

Figura 7. Resultado de búsquedas

Al finalizar, debe volver al menú principal.

- **[C] Volver a evaluar un proyecto:** Si un proyecto fue aceptado de forma apresurada, se puede revertir el proceso. Para eso, el sistema debe solicitar el ID del proyecto. Si es un código válido, se debe eliminar el proyecto y agregar nuevamente a la lista con nexo de solicitudes en la posición que corresponda (ver Figura 8). Al finalizar, debe volver al menú principal.

```
[.......... Revertir un proyecto ..........]
Ingrese la Id: D230EUHIOWD
 Error: proyecto no encontrado
Ingrese la Id: DHHSJS
 Proyecto borrado.
Se ha creado una nueva solicitud.
```

Figura 8. Menú y opciones para revertir un proyecto

- **[D] Estadísticas:** El sistema debe mostrar las siguientes estadísticas:
    - **Porcentaje de proyectos por dificultad:** Porcentaje de proyectos asociados a cada nivel de dificultad (ver Figura 9).

```
[Porcentaje proyectos principiantes]: 15%
[Porcentaje proyectos faciles]: 12%
[Porcentaje proyectos intermediarios]: 25%
[Porcentaje proyectos dificil]: 31%
[Porcentaje proyectos muy dificil]: 17%
```

Figura 9. Primera estadística

Nota: El sistema debe estar preparado y adaptarse si es que existen más o menos tipos de dificultades.

- **Usuarios que tienen más proyectos aceptados:** El sistema debe mostrar el usuario que más proyectos aceptado tiene. Del jugador se debe indicar lo siguiente:
  - Nickname
  - Promedio de la dificultad de todas las solicitudes enviadas por el usuario, para ello debe considerar el siguiente peso de las dificultades:
    - Principiante: 1.
    - Fácil: 2.
    - Intermedio: 3.
    - Difícil: 4.
    - Muy difícil: 5.

    Se debe mostrar el valor numérico resultante de la operación redondeado al segundo decimal.

  - Porcentaje de proyectos aceptados por este usuario con respecto al total de todos los proyectos.

- **Filtro más utilizado:** El sistema debe mostrar el filtro más utilizado.

- **[E] Salir:** Antes de finalizar el programa, el sistema debe:
  1. Actualizar los archivos “proyectos.csv” y “solicitudes.csv”.
  2. Destruir todas las estructuras de datos utilizando referencias a punteros.

III. Entrega

La implementación de las siguientes estructuras de datos es **obligatoria**. En caso de que alguna no sea implementada, se utilice otra estructura de datos en su reemplazo, o se recurra a una librería que contenga la implementación interna, el taller completo será evaluado con la nota mínima.

3.1. Arreglo dinámico

Debe recurrir al uso de un arreglo dinámico para poder almacenar la información de todos los proyectos existentes.

La ventaja de un arreglo dinámico es que puede aumentar o disminuir su tamaño al variar la cantidad de datos que contiene. Por ejemplo, si el arreglo tiene una capacidad inicial de 5 elementos, al insertar el número 6 deberá expandirse. El sistema debe permitirlo mediante la reasignación de memoria utilizando la función nativa “realloc”.

3.2. Lista con nexo

Las solicitudes de contribución deben almacenarse en una lista con nexo simple, donde deben estar ubicadas las solicitudes más antiguas al inicio de la lista y las más actuales al final.

IV. Entrega

4.1. Fecha

La fecha límite para la entrega del taller es el 29 septiembre hasta las 23:59 hrs. Por cada hora de atraso en la entrega, se descontará un punto a la nota final del taller, es decir: si la entrega llega entre 1 y 60 minutos tarde, implica un descuento de 10 décimas; si llega entre 61 y 120 minutos tarde, implica un descuento de 20 décimas, y así sucesivamente.

4.2. Entregables

Se debe subir:

1. Programa en C++. Proyecto completo en la IDE Clion o Visual Studio.
2. Informe en tamaño carta en formato PDF.

4.3. Integrantes

El taller se puede realizar de forma individual o en parejas.

4.4. Formato de Entrega

El taller se debe subir a GitHub utilizando el controlador de versiones Git. El repositorio debe estar en privado y contener contribuciones de ambos integrantes, si el taller se resuelve solo, solo debe haber contribuciones de ese estudiante. A la hora de enviar el proyecto, se debe agregar al repositorio como colaborador a los ayudantes:

- francisco.llanos@alumnos.ucn.cl
- benjamin.miranda02@alumnos.ucn.cl

El taller se revisará hasta el commit más reciente antes el último plazo de entrega del taller, 5:00 AM (con descuento). Además, como respaldo, se debe subir a Campus Virtual en un archivo de texto con el enlace del repositorio de Github.

La no entrega del proyecto en alguna plataforma implicará la no revisión del proyecto.

4.5. Interrogación

Todos los estudiantes que cursen la asignatura Estructura de Datos serán interrogados al menos una vez durante el semestre en relación con sus entregas de talleres. El propósito de esta interrogación es asegurarse de que los participantes puedan demostrar el conocimiento y la participación en el trabajo entregado.

En términos de programación, resulta vital señalar que la contribución de cada miembro del equipo al taller debe ser equitativa y esencial para la construcción de las estructuras de datos. En otras palabras, no se permitirá que un estudiante se encargue únicamente de implementar las estructuras de datos mientras que otro se dedica exclusivamente a crear métodos “get” y “set”, realizar despliegues por consola, o desarrollar la documentación. Es fundamental que ambos demuestren su dominio de las estructuras de datos.

En los días posteriores a la entrega del taller, el ayudante enviará un correo electrónico a todos los seleccionados para la interrogación con el fin de coordinar el lugar y el horario en el que se llevará

a cabo. Es importante destacar que esta reunión debe ser obligatoriamente presencial y, en caso de que sea en parejas, ambos integrantes deben asistir. Si alguno de ellos no asiste, ambos recibirán la calificación mínima de 1.0.

La interrogación de los estudiantes seleccionados implica mantener la nota obtenida o reducirla según lo estime el ayudante, en la medida que sea necesario. Por ejemplo, si se determina que el estudiante no participó efectivamente en el taller, la calificación podría reducirse de 7.0 a 1.0.

4.6. Ejecución

En el caso de que el proyecto no compile, el taller se evaluará con nota máxima 3.9.

Incluso si el estudiante tiene alguna bonificación, por ejemplo, décimas de ayudantía, no se puede superar esta nota. Sobre ella se deben aplicar también todos los descuentos en los que pueda haber incurrido el estudiante.

4.7. Usabilidad

La ejecución del programa debe ser amigable con el usuario, esto significa que debe cumplir lo siguiente:

- Todas las entradas de datos deben ser validadas; si algo falla se debe indicar por qué no pasó la validación.
- En el caso de que una validación falle, el programa debe mantener los datos ingresados y únicamente solicitar la corrección del dato erróneo, permitiendo al usuario corregir el error sin perder información ingresada al volver a un menú anterior.

El no cumplimiento de algún inciso provocará descuento en los apartados que asocien alguna de estas actividades.

4.8. Puntaje para la evaluación

Categoría	Apartado	Puntaje
Diseño	Se respeta el encapsulamiento de las clases.	5
	El programa principal es sencillo (método main).	5
	Posee documentación en el proyecto. Doxygen documentation para CLion y XML documentation para Visual Studio.	10
	Utiliza nombres nemotécnicos, tanto para variables, constantes, funciones, métodos, clases y paquetes.	5
	El programa se diseña utilizando componentes abstractos.	5
Ejecución	Lectura de archivos. Se lee el archivo “solicitudes.csv” y “proyectos.csv”.	10
	Almacenamiento de elementos en las estructuras de datos.	5
	Menú	5
	Opción A: Opción Aceptar.	15
	Opción A: Opción Rechazar.	10
	Opción A: Opción Siguiente.	10
	Opción B: Filtros.	20
	Opción C: Estadísticas.	10
	Opción D: Salir.	10

Informe	**Portada.** Contiene el nombre de la profesora y de los ayudantes de taller, fecha de entrega, nombres, correos electrónicos de los estudiantes, y nombre del taller. (Ejemplo: Taller 1).	5
	Introducción. Posee una breve introducción del problema a abordar junto a como este documento presentará la solución.	10
	Cuerpo. Posee la explicación y metodología para resolver el taller. Explica en detalle las estructuras de datos implementadas. Expone el diagrama de clases y realiza explicaciones algunas estrategias para abordar la solución. (Ejemplo: como se implemento la lectura de archivos).	20
	Conclusión. Posee un análisis final sobre lo aprendido, las dificultades enfrentadas y que se debería mejorar para el siguiente taller.	10
	Total (60% de exigencia)	170

Sea *P* el puntaje obtenido, la nota final se calcula mediante la siguiente expresión:

$$N = \begin{cases} Si\ P < 102, & \frac{3P}{102} + 1 \\ Si\ P \geq 102 & \frac{3P - 306}{68} + 4 \end{cases}$$

Donde *N,* se redondea a un solo decimal.

¡ÉXITO! ☺