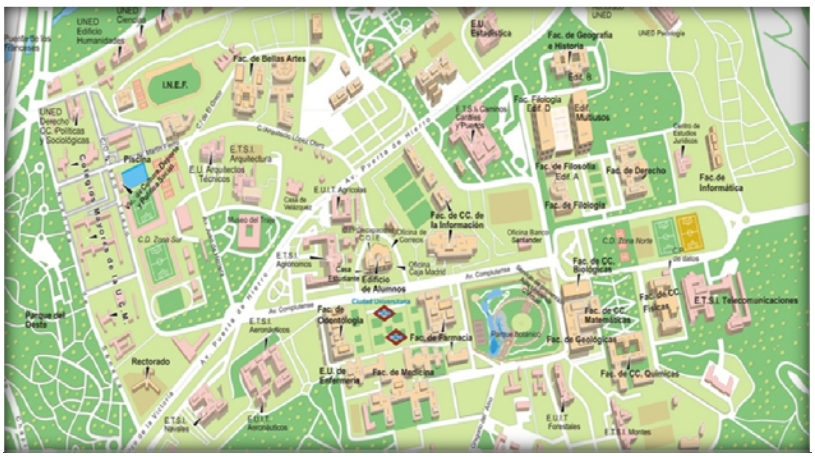
Práctica 4

Campus Ville

Fecha de entrega: 19 de mayo de 2012



1. Descripción de la práctica

Se trata de desarrollar un juego multijugador sobre la gestión de edificios en un campus universitario. Cada jugador juega como una empresa distinta del campus y puede comprar edificios de la lista de edificios disponibles en el campus. Los jugadores acceden al juego con un usuario y contraseña y, a través de una interfaz de consola, van comprando edificios del campus, de acuerdo con su presupuesto. En cada turno, cada edificio genera dinero (ganancias o pérdidas) y prestigio (positivo o negativo). El objetivo es liderar la clasificación de jugadores, ya sea por dinero o por prestigio.

En el juego existe un usuario especial, el **administrador**, que es el que puede añadir nuevos edificios al juego. Todos los datos de los edificios se guardan en una única lista de tamaño variable para todo el programa. El administrador también gestiona la lista de usuarios, pudiendo añadir nuevos usuarios a través de su interfaz de consola.

Cada cierto tiempo, el administrador **ejecuta un turno**, y se recorren todas las listas de edificios de todos los jugadores, aplicando el incremento o decremento de dinero y prestigio para cada uno.

La práctica consiste en desarrollar esta aplicación, en versión de consola, con la funcionalidad necesaria para gestionar la lista de edificios, la lista de jugadores y los turnos de juego.

Para ello, hay que crear un programa que empiece intentando cargar los contenidos iniciales de las listas de edificios y jugadores a partir de los archivos `edificios.txt` y `jugadores.txt`. En caso de que alguno de los archivos no exista, el programa empezará con una lista vacía. Después de cargar los archivos, se pedirá al usuario que indique su nombre y contraseña. Si el usuario es un jugador normal, se mostrará el **menú de juego**. Si el usuario es el administrador, se mostrará el **menú de administración**. Se dispondrá de tres intentos para acertar la contraseña.

> El administrador es un usuario especial que no se guarda en la lista de jugadores. Su nombre de usuario es **admin** y su contraseña **12345**.

Una vez que el usuario termina de jugar o administrar cerrará la sesión, de forma que se volverá a preguntar por un nombre de usuario y contraseña. La ejecución terminará cuando el usuario escriba como nombre de usuario "salir", y tras haber guardado los datos de las listas en los archivos `edificios.txt` y `jugadores.txt`.

1.1. Información del programa

A continuación se describe la información que se ha de mantener.

Edificios: Los edificios se guardan en una lista de tamaño variable, de acuerdo con la estructura vista en clase (estructura con array y contador). **La lista se debe mantener ordenada por código en todo momento.** El tamaño máximo de la lista es 50, pero tiene que ser sencillo cambiar el programa para admitir listas de mayor tamaño.

Para cada edificio, se guardará la siguiente información:

- `Código`: Número entero positivo, único para cada edificio.
- `Nombre`: Nombre del edificio (cadena que puede contener espacios).
- `Precio`: Precio de compra del edificio (entero positivo).
- `Dinero`: Dinero que genera (o pierde) el edificio en cada turno. Un número entero positivo (genera beneficios) o negativo (cuesta dinero).
- `Prestigio`: Prestigio que genera (o pierde) el edificio en cada turno. Un número entero positivo (genera prestigio) o negativo (daña la imagen).
- `Estado`: El edificio está disponible o ha sido comprado por un jugador o ha sido borrado del sistema (disponible, comprado o borrado: un enumerado).

Jugadores: La información de los jugadores se guarda en una lista de tamaño variable, de acuerdo con la estructura vista en clase (estructura con array y contador). El tamaño máximo de la lista es 20, pero tiene que ser sencillo cambiar el programa para admitir listas de mayor tamaño.

Para cada jugador, se guardará la siguiente información:

- `Usuario`: Nombre de usuario (cadena sin espacios) debe ser único para cada usuario. Ningún usuario puede llamarse "admin" o "salir".
- `Contraseña`: Contraseña del usuario (cadena sin espacios).
- `Universidad`: Nombre de la universidad (cadena con espacios).
- `Dinero`: Dinero que tiene el jugador en el banco (número entero). Al principio todos los jugadores empiezan con 3.000 créditos.
- `Prestigio`: Prestigio acumulado por el jugador (número entero). Al principio todos los jugadores empiezan con 0 puntos de créditos de prestigio.
- `Comprados`: Lista de tamaño variable con los códigos de los edificios que ha comprado el jugador (sólo los códigos).

Todas las cadenas de caracteres se implementarán como cadenas al estilo de C.

1.2. Menú de administración

Este menú permitirá al administrador gestionar las listas de edificios y de jugadores, así como ejecutar turnos en los que se actualiza toda la información:

```
1.- Ver el listado de edificios
2.- Nuevo edificio
3.- Borrar un edificio
4.- Ver el listado de jugadores
5.- Nuevo jugador
6.- Borrar un jugador
7.- Ejecutar un turno
8.- Ver la clasificación
0.- Cerrar la sesión
```

A continuación se describe cada opción del menú.

- **Ver el listado de edificios**: Muestra los datos completos de los edificios de la lista. Tanto los edificios disponibles como los ya comprados y los borrados.
- **Nuevo edificio**: Comprueba que la lista de edificios no esté llena. Si no lo está, pide al administrador el código (uno que no exista), el nombre, el precio, el

prestigio y el dinero (inicialmente los edificios están disponibles). Una vez creado, lo añade a la lista. **La lista debe mantenerse ordenada por código.**

- **Borrar un edificio**: Pide al administrador un código de edificio y, si existe, lo borra del sistema. En realidad los edificios no se quitan de la lista, sino que se marcan como borrados en ella. Sólo se puede borrar un edificio que no haya sido comprado todavía por ningún jugador.
- **Ver el listado de jugadores**: Muestra todos los datos de todos los jugadores de la lista. Se muestran también los datos básicos de todos los edificios que posee cada jugador (nombre, dinero por turno y prestigio por turno).
- **Nuevo jugador**: Comprueba que la lista de jugadores no esté llena. Si no lo está, pide al administrador los campos usuario (uno que no exista), contraseña y nombre de la empresa. Inicialmente el jugador tendrá 3.000 créditos y 0 puntos de prestigio. Una vez creado el jugador, lo añade **al final de la lista**.
- **Borrar un jugador**: Pide al administrador un nombre de usuario y elimina todos los datos del jugador de la lista. Si el jugador había comprado edificios, todos esos edificios vuelven a estar disponibles.
- **Ejecutar un turno**: Recorre la lista de jugadores y, para cada uno, recorre su lista de edificios comprados. Para cada edificio, añade (o resta) el dinero y puntos de prestigio que el edificio genera en cada turno.
- **Ver la clasificación**: Muestra la clasificación actual (ordenada por dinero).

1.3. Menú de juego

Cuando el jugador entra en el juego puede ver su lista edificios, la lista de edificios disponibles y comprar nuevos edificios. También puede comprobar el estado actual de la clasificación. Esto se hace por medio del siguiente menú:

```
1.- Ver mis edificios
2.- Ver los edificios disponibles
3.- Comprar un edificio
4.- Ver la clasificación (ordenada por dinero)
5.- Ver la clasificación (ordenada por prestigio)
0.- Cerrar la sesión
```

- **Ver mis edificios**: Muestra un listado con todos los edificios comprados por el jugador. Para cada edificio, se muestran los datos completos.
- **Ver los edificios disponibles**: Muestra un listado con todos los edificios de la lista, mostrando únicamente los edificios disponibles (los no comprados aún).

- **Comprar un edificio**: Pide al usuario un código de edificio y comprueba si el edificio existe, no ha sido comprado por ningún jugador y el jugador actual tiene dinero suficiente para comprar el edificio. Si se cumplen todas las condiciones, realiza la compra, resta el dinero al jugador y añade el código del edificio a la lista de edificios comprados por el jugador.
- **Ver la clasificación**: Muestra la clasificación actual de los jugadores, ordenada por dinero o por prestigio según la opción seleccionada.

2. Guía de implementación

2.1. Organización en módulos

La práctica deberá organizarse en módulos, con un módulo independiente para cada tipo principal. Se propone la creación de, al menos, estos módulos:

Módulo de Edificio

Declarará un tipo de datos `tEdificio` para mantener la información de un edificio. Además, implementará las siguientes operaciones:

- `nuevoEdificio()`: No recibe ningún valor de entrada. Pide al usuario los datos de un edificio y devuelve como salida una estructura `tEdificio` debidamente inicializada.
- `mostrarEdificio()`: Dado un `tEdificio` muestra todos sus datos en la consola, debidamente formateados.

Módulo de Lista de edificios

Declarará un tipo de datos `tListaEdificios` para mantener la lista de edificios e implementará las siguientes operaciones:

- `listaEdificiosoLlena()`: Función que recibe como parámetro una lista de edificios y devuelve un booleano indicando si esa lista está llena o no.
- `buscarEdificio()`: Recibe una lista de edificios y un código. Indica si se ha encontrado o no un edificio con ese código y, en caso afirmativo, la posición en la que se encuentra. Debe estar implementado como una búsqueda binaria.
- `insertarEdificio()`: Recibe un `tEdificio` y una lista de edificios. Inserta el edificio en la lista en la posición que le corresponda por código y devuelve la lista actualizada.
- `bajaEdificio()`: Recibe la lista edificios y un código de edificio. Si el edificio existe y está disponible, lo marca como borrado en la lista.

Módulo de Jugador

Declarará un tipo de datos `tJugador` para mantener la información de un jugador e implementará las siguientes operaciones:

- `nuevoJugador()`: Lee los datos de un jugador, inicializa su lista de edificios comprados como vacía, el dinero a 3.000 créditos y los puntos de prestigio a 0. Devuelve la estructura `tJugador` debidamente inicializada.
- `mostrarJugador()`: Dado un `tJugador` y la lista de edificios, muestra en la consola los datos del jugador y los datos básicos de todos sus edificios (nombre, dinero y prestigio por turno).
- `listaCompradosLlena()`: Recibe como entrada un `tJugador` y devuelve un booleano indicando si su lista de edificios comprados está llena.
- `comprarEdificio()`: Recibe como entrada un `tJugador`, un código de edificio y la lista completa de edificios. Si se puede hacer la compra (el edificio existe, está disponible y hay dinero suficiente) realiza la compra y añade el código de edificio a la lista de edificios del jugador.

Módulo de Lista de jugadores

Declarará un tipo de datos `tListaJugadores` para mantener la lista de jugadores e implementará las siguientes operaciones:

- `listaJugadoresLlena()`: Función que recibe como parámetro una lista de jugadores y devuelve un booleano indicando si la lista está llena o no.
- `buscarJugador()`: Recibe como entrada una lista de jugadores y un nombre de usuario. Devuelve un booleano indicando si se ha encontrado el jugador o no y, en caso afirmativo, la posición en la que se encuentra el jugador.
- `insertarJugador()`: Recibe una variable de tipo `tJugador` y una lista de jugadores. Inserta el jugador al final de la lista de jugadores.
- `bajaJugador()`: Recibe un nombre de usuario, la lista de jugadores y la lista de edificios. Busca al jugador en la lista y lo elimina completamente.

2.2. Formato de los archivos de texto

`edificios.txt`: Datos de los edificios. Por cada edificio, esta serie de líneas:

Código del edificio (-1 para terminar) ↲

Nombre ↲

Precio de compra ↲

Dinero por turno ↲

Prestigio por turno (real) ↲

Estado actual (entero con el código del enumerado: 0, 1 ó 2) ↲

`jugadores.txt`: Datos de los jugadores. Por cada jugador esta serie de líneas, primero con sus datos y luego con los edificios comprados:

Nombre de usuario ("X" para terminar) ↲

Contraseña ↲

Nombre de la empresa ↲

Dinero ↲

Prestigio ↲

Por cada edificio comprado:

Código del edificio (-1 para terminar) ↲

2.3. Otras indicaciones

- Las listas de tamaño variable deberán implementarse con la estructura de array más contador vista en el Tema 6.
- Siempre que se pueda, las operaciones de búsqueda serán binarias.
- Todos los módulos se implementarán con archivos `.h` y `.cpp`. En el archivo `.h` se incluirán los prototipos de las funciones públicas con comentarios describiendo qué hace cada función, qué datos recibe y qué datos devuelve.
- Los módulos de `tListaEdificios` y `tListaJugadores` únicamente realizan operaciones con listas; nunca deben escribir nada en la consola. Pueden devolver valores al programa que los llame para que éste escriba en pantalla lo que sea necesario.

3. Entrega de la práctica

La práctica se entregará a través del Campus Virtual. Se habilitará una nueva tarea **Entrega de la Práctica 4** que permitirá subir un único archivo. Hay que subir un archivo comprimido `GrupoXX.zip` que contenga los archivos `.h` y `.cpp` del proyecto (y sólo esos).

Fin del plazo de entrega: **19 de mayo a las 23:55**.