Exercici L5

Tamagotchi 2.0

Fonaments de Programació II

Objectiu

L'objectiu d'aquesta pràctica és que practiqueu la lectura i escriptura de fitxers binaris i, addicionalment, el pas d'arguments per línia de comandes.

Per aconseguir-ho, modificareu la pràctica del Laboratori L3 per afegir les funcionalitats de **Guardar partida** i **Carregar partida**, utilitzant fitxers binaris. També afegireu funcionalitats a través dels paràmetres de la línia de comandes. La pràctica es divideix en dues fases.

Descripció de l'exercici

Ja disposeu d'un joc de Tamagotchi funcional, desenvolupat en el laboratori L3. Les seves funcionalitats es mantindran intactes: no es modificaran ni les puntuacions ni la dinàmica del joc.

Si encara no heu completat el Tamagotchi del L3, assegureu-vos de fer-ho abans de continuar amb aquesta pràctica.

Tasques a fer:

- **Fase 1**: Implementareu un sistema per **guardar** i **carregar** una partida, de manera que l'estat del Tamagotchi es pugui desar en un fitxer i recuperar posteriorment.
- **Fase 2**: Modificareu el programa perquè es pugui carregar una partida directament des de la línia de comandes, sense necessitat d'entrar al menú.

Fase 1: Guardar i carregar partida

Per poder guardar una partida, cal desar totes les variables d'estat perquè, en començar una nova sessió, en comptes d'inicialitzar-les als valors predeterminats, es recuperin els valors guardats. Per exemple, en un joc com els escacs, l'estat de la partida correspondria a la disposició de les peces al tauler. En el cas del Tamagotchi, les variables d'estat són **son, gana,**

felicitat i higiene, que ja tenim convenientment encapsulades en una estructura de tipus registre.

Per guardar la partida, cal obrir un fitxer en **mode escriptura binària** i escriure-hi aquesta estructura. Per començar, assumirem que disposem d'una **constant simbòlica** (`#define`) que indica el nom del fitxer on es desarà la partida (per exemple `#define NOM_FIT_PARTIDA "partida_guardada.dat"`). Així doncs, quan haguem de guardar la partida ho farem a aquest fitxer.

Per carregar una partida, caldrà mostrar un **menú addicional** a l'inici del joc, on l'usuari podrà triar entre:

- **Començar una partida nova**, en què l'estat del Tamagotchi s'inicialitzarà com en la versió original.
- **Carregar una partida guardada**, en què es llegirà el fitxer `NOM_FIT_PARTIDA` per recuperar l'estat anterior.

Consells per a la implementació

- Implementeu primer la funcionalitat de "guardar partida" abans de "carregar partida", ja que sense una partida guardada no podreu comprovar si la càrrega funciona correctament.
- Perquè l'usuari pugui triar de guardar partida, només cal afegir una nova opció al menú que contingui l'opció de guardar i sortir.
- Encapsuleu el procés de guardar partida en una funció.
- Creeu una funció separada per al procés de demanar a l'usuari si vol començar una partida nova o carregar-ne una d'existent.
- Encapsuleu el procés de càrrega en una funció dedicada. Aquesta funció s'encarregarà d'obrir el fitxer, llegir-ne el contingut i restaurar l'estat del Tamagotchi.

Aquest enfocament facilitarà la implementació i permetrà mantenir un codi més clar i modular.

Compte amb els punters!

Compte, no feu servir els operadors `&` i `*` sense pensar. Quan utilitzeu `fread` i `fwrite`, heu de passar l'adreça de la variable o estructura que voleu llegir o escriure. Per tant, és d'esperar que el primer paràmetre serà `&t`, on `t` és una variable de tipus registre.

Però compte: si aquestes funcions es criden dins d'una altra funció que ja ha rebut l'estructura per referència, **què heu de passar-hi?** Si la funció ja rep un punter a l'estructura, no s'ha d'afegir cap `&`, perquè el que heu rebut **ja és una adreça**.

Reflexioneu bé sobre quins paràmetres esperen `fread` i `fwrite`, i assegureu-vos de no utilitzar els operadors de punters a l'atzar.

Fase 2: Arguments de línia de comandes

No continueu amb aquesta fase si no heu completat la primera, ja que depèn de la seva implementació.

En aquesta fase, afegirem la possibilitat de carregar una partida directament des de la línia de comandes, sense haver de passar pel menú que pregunta si es vol començar una nova partida o carregar-ne una de guardada.

Si el programa s'executa normalment, per exemple:

```
tamagotchi.exe
```

el joc funcionarà com a la fase anterior, preguntant a l'usuari si vol començar una partida nova o carregar-ne una de guardada.

Tanmateix, si l'usuari especifica l'opció `--load` seguida d'un nom de fitxer, com ara:

```
tamagotchi.exe --load partida_usuari.dat
```

aleshores el programa carregarà directament la partida des del fitxer `partida_usuari.dat`, sense mostrar el menú inicial.

Comprovació dels arguments

Per implementar aquesta funcionalitat, el programa ha de comprovar el nombre d'arguments rebuts a la línia de comandes:

- Si hi ha **només un argument** (el nom del programa), es procedeix com en la fase anterior.
- Si hi ha **tres arguments** i el segon és exactament `--load`, es carrega la partida des del fitxer indicat com a tercer argument.
- Si el nombre d'arguments no és cap d'aquests dos casos, es considera un error, es mostra un missatge explicatiu i el programa finalitza.

Compte amb la gestió de cadenes

Per comparar cadenes (`--load`) i treballar amb noms de fitxers passats com a argument, necessitareu les funcions `strcmp` i `strcpy`. Llegiu-ne la documentació abans d'utilitzar-les per assegurar-vos que les feu servir correctament.

Per practicar més...

A continuació, es presenten diverses idees, ajustades al nivell de la classe, que pots aplicar per practicar una mica més.

Nom de partida personalitzat

- Modifica el programa perquè, abans de desar una partida, demani a l'usuari el nom del fitxer on la vol guardar. Això permetrà tenir múltiples fitxers de partides guardades i evitar sobreescriure-les utilitzant sempre el mateix nom de fitxer.

Tractament d'errors més simpàtic

- En la versió actual, probablement has implementat que, en cas d'error durant la lectura o escriptura del fitxer, el programa es tanqui immediatament. Tot i que aquest enfocament simplificat és acceptable per a aquesta pràctica, en una aplicació real seria massa dràstic.
    - Si no es pot escriure el fitxer, el programa no hauria de finalitzar, sinó permetre continuar jugant.
    - Si el fitxer de la partida que volem carregar no existeix, el programa hauria de demanar a l'usuari que introdueixi un altre nom de fitxer en lloc de finalitzar la partida i perdre tot el progrés.
    - Implementa aquests canvis en aquesta pràctica.

Afegir un sistema d'autosave

- Modifica el programa perquè desi automàticament l'estat del joc cada cert nombre de torns.
- En cas que el programa es tanqui inesperadament, l'usuari podria reprendre la partida des d'aquest punt.