# Dungeons & Salle

Data: 19-5-2013
Nom: Àlex Cordón Vila
Login: ls25707

# Index

**Resum de l'enunciat..... 2**

**Estructuració de la pràctica..... 3**

**Problemes observats..... 6**

**Conclusions..... 7**

## Resum de l'enunciat:

La practica 3 és un programa en C que simula un joc de "Role" semblant al Dungeons & Dragons. Per fer la pràctica s'ha usat la programació modular, les estructures de dades lineals i la lectura/escriptura de fitxers.

El programa rep externament dos fitxers inicials. En un hi ha totes les classes disponibles amb els valors d'atac, defensa i vida i a l'altre hi ha tots els enemics amb els valors d'atac, defensa, vida i experiència.
El programa comença amb un menú principal on el jugador pot escollir entre sortir del programa, crear un personatge nou o bé carregar-ne un de guardat. Per carregar un personatge cal utilitzar la lectura de fitxers per extreure'n la informació necessària i així definir el personatge. Per a poder carregar un personatge s'ha d'escollir l'opció 2 del menú i introduir el nom del fitxer on hi ha guardat el personatge, si el nom és incorrecte s'informa amb un missatge d'error. Per crear un personatge nou s'ha d'escollir l'opció 1 i introduir el nom i seguidament la classe que es vulgui, sempre i quant aquesta existeixi al fitxer de classes. Per mostrar les classes disponibles per pantalla abans s'ha hagut de fer una lectura de fitxer, en aquest cas el fitxer de classes.

Un cop escollida una de les dues opcions apareix el menú de personatge on podem escollir d'entre 4 opcions. La primera opció ens permet jugar al " mode aventura" on a l'inici es mostra la llista d'enemics disponibles i seguidament es demana que es vagi introduint el numero dels enemics amb es vol lluitar; per finalitzar la selecció cal introduir un -1. Per acumular tots els enemics escollits s'ha utilitzat una cua dinàmica.

Un cop escollida la cua d'enemics s'ha de lluitar fins que l'heroi mori o bé fins que s'hagin derrotat tots els enemics. En acabar sempre es retorna al menú de personatge. La segona opció condueix a una pantalla d'informació sobre l'heroi on s'hi mostra el nom, la classe , el nivell, la experiència, l'atac, la defensa i la vida; en prémer "enter" retorna al menú de personatge. La tercera opció permet guardar la informació de l'heroi dins d'un fitxer i, per fer-ho, hem utilitzat la tècnica d'escriptura de fitxers; al fitxer s'hi guarda el nom, la classe, el nivell i l'experiència. Finalment, la quarta opció permet retornar al menú principal d'inici del programa.

## Estructuració de la pràctica:

1-Estructuració de la pràctica de mòduls:

S'ha estructurat la pràctica en cinc mòduls , lògica, classes, enemics, herois i cua.

El mòdul de lògica és el mòdul més important perquè inclou tots els ".h" de la resta de mòduls. El mòdul lògica engloba totes les funcions de menús i les funcions o procediments que calculen i modifiquen les variables per l'opció "anar d'aventura" del menú de personatge.
Capçaleres del mòdul lògica:

1- Permet mostrar i fer interactiu el primer menú del programa:
   void LOGICA_CallMenu (int *nOp, Classe *pastClasse, Enemic *pastEnemic, int nNumClasse, int nNumEnemics);

2- Permet mostrar i fer interactiu el segon menú del programa:
   void LOGICA_heroMenu( int *nOp, Hero stPlayer, Enemic *pastEnemic,int nNumEnemics);

3- Inicia el sistema de selecció d'enemics, la fase de realització, la fase de generació i la fi de l'aventura:
   Hero LOGICA_iniciaAventura(Enemic *pastEnemic, int nNumEnemics, Hero stPlayer);

4- Crea la cua d'enemics i es mostren per pantalla per a poder ser seleccionats un a un:
   Cua LOGICA_faseGeneracio(Enemic *pastEnemic, int nNumEnemics);

5- Genera tot el sistema de lluites i ho mostra per pantalla:
   int LOGICA_faseRealitzacio(Cua *pCuaEnemics, Hero stPlayer);

6- Rep l'experiència guanyada i assigna les puntuacions corresponents a tots els camps del jugador (EXP,VIDA,ATAC,NIVELL/S,DEFENSA):
   void LOGICA_fiAventura(int nExpTotal, Hero *stPlayer);
7- És utilitzada dins de la 5 i en funció del valor de dau escull quin atac genera l'heroi:
   void LOGICA_atacaHeroi(Hero atacant, Enemic *defensor);
8- És utilitzada dins de la 5 i en funció del valor de dau escull quin atac genera l'enemic:
   void LOGICA_atacaEnemic(Hero defensor, Enemic *atacant, int *nVida);
9- Simula una tirada dw'un dau de 20 cares i en retorna el resultat
   int LOGICA_dau( );

El mòdul classe s'encarrega de la lectura del fitxer anomenat fitxer_classes.txt i conté l'estructura del tipus Classe.
Capçaleres del mòdul classe:

1- Llegeix el fitxer de classes i retorna el nombre de classes llegides i la informació del fitxer en una variable Classe*:
   Classe* CLASSE_lecturaFitxer (FILE *f, int *nClasse);

El mòdul d'enemics s'encarrega de la lectura del fitxer d'enemics i conté l'estructura del tipus Enemic.
Capçaleres del mòdul enemic:

1- Llegeix el fitxer d'enemics i retorna el nombre d'enemics llegits i la informació del fitxer dins una variable de tipus Enemic*:
   Enemic* ENEMIC_lecturaFitxer(FILE *g, int *nNumEnemics);

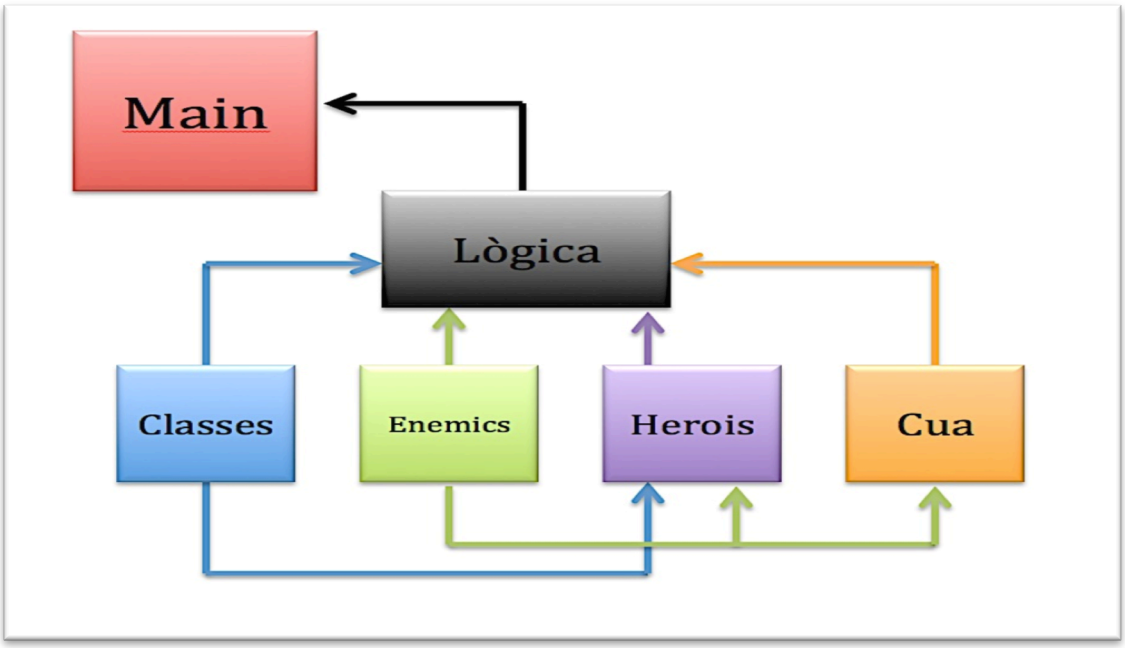
El mòdul herois conté les funcions encarregades de les opcions del menú principal, tant la de carregar i crear jugador com també la de guardar personatge del segon menú. Hi ha també la estructura del tipus Hero.
Capçaleres del mòdul herois:

1- Aquesta funció s'encarrega del procés de creació del personatge, retorna una variable inicialitzada amb la informació necessària per jugar:
   Hero HERO_heroCreation (Classe *pastClasse, int nNumClasse, Hero stPlayer);
2- Permet carregar les dades d'un jugador guardat en una partida anterior:
   Hero HERO_loadHero(Classe *pastClasse, int nNumClasse,Hero stPlayer, int *nError);
3- Permet guardar totes les dades del personatge de la partida dins d'un fitxer que més tard es pot recuperar:
   void HERO_guardarHeroi (Hero stPlayer);
4- Mostra per pantalla tota la informació referent al personatge que ha creat o carregat el jugador:
   void HERO_heroInfo (Hero stPlayer);

Finalment el mòdul cua conté totes les funcions i procediments necessaris per tractar la informació d'una cua dinàmica.
Capçaleres del mòdul cua:

1- Demana memòria de forma dinàmica per poder treballar amb la cua:
   Cua CUA_crea();
2- Afegeix un element a la cua:
   void CUA_encua(Cua*c, Enemic e);
3- Elimina el primer element introduït a la cua:
   void CUA_desencua(Cua *c);
4- Allibera la memòria demanada al crear la cua:
   Cua CUA_destrueix(Cua c);
5- Consulta el primer element de la cua:
   Enemic CUA_primer (Cua c);
6- Retorna cert si la cua és buida:
   int CUA_buida(Cua c);
7- Retorna cert si la cua és plena
   int CUA_plena(Cua c);

Representació en blocs dels mòduls:




Explicació de l'ús i funcionament de la cua d'Enemics:

La cua d'Enemics s'ha utilitzat per acumular la informació de cada enemic escollit per l'usuari i, un cop acabat, generar tots els combats en ordre de selecció.

Fase de generació:
Dins el codi, la cua es crea sempre que l'usuari entra a l'opció 1 del menú de personatge fent us de la funció (Cua CUA_crea ();). Tot seguit, cada vegada que el jugador escull un enemic correcte la cua afegeix un nou enemic a la cua usant la funció (Cua CUA_encua (Cua*c, Enemic e);). Com que la cua utilitza memòria dinàmica no hi ha cap límit per afegir enemics.

Fase de realització:
La fase de realització repeteix sempre la mateixa seqüencia mentre la cua no sigui buida o la vida de l'heroi sigui més gran que zero. Per saber si la cua és buida s'ha utilitzat la funció (int CUA_buida(Cua c);) que ens retorna cert si és buida o fals si és plena.
Un cop l'heroi ha derrotat un enemic aquest es "desencua" i el següent enemic es selecciona usant la funció (Enemic CUA_primer (Cua c);).

Un cop s'han derrotat tots els enemics de la cua, el programa utilitza la funció (Cua CUA_destrueix(Cua c);) per alliberar tota la memòria dedicada a guardar la cua creada.

Altres consideracions:

Per millorar l'estètica dels menús s'ha utilitzat la funció "gotoxy(x,y)", la funció per netejar la pantalla "clrscr()" i getchar() per fer pauses controlades del programa. Aquestes funcions ens ha permès mostrar els menús centrats a la pantalla i mostrar taules de dades en columnes paral·leles. Tot això ens ha

millorat l'estètica però ens ha augmentat considerablement el nombre de línies de codi i ha fet més difícil la implementació d'algunes funcións.
Amb tot, a la fase de realització s'ha afegit la capacitat de mostrar per pantalla els noms dels enemics que l'usuari va escollint.

## Problemes observats:

1- Al començar la pràctica em va ser difícil crear per primer cop un makefile i veure la seva utilitat perquè mai abans havíem hagut de crear-lo de zero i a classe no s'havia vist.
2- La programació modular inicialment era confusa tant en idea com a l'hora d'escriure i organitzar el codi.
3- Al generar valors aleatoris usant la funció rand s'ens retornava dos valors aleatoris iguals per a l'enemic i el personatge. Vam resoldre el problema col·locant al main srand(time(NULL));
4- Un dels punts més complicats ha estat el funcionament de cua i l'enllaç entre diversos mòduls quan ens veiem obligats a incloure's mútuament.
5- Al mostrar els menús o els llistats d'informació dels fitxers moltes vegades no quedaven alineats per culpa de la dimensió variable dels noms de les classes o dels enemics. Per solucionar-ho s'ha utilitzat la funció gotoxy(x,y).
6- Al fer la funció de guardat de personatge vam veure que no s'especificava l'extensió que havia de portar el fitxer. Es va solucionar preguntant als professors de pràctiques.
7- Programar en un editor com el VIM ha estat difícil per treballar amb més d'un fitxer a l'hora.
8- Tot i que es va organitzar el codi en mòduls i es va distribuir el codi en subprogrames, la programació en equip no ha estat possible i el grup de treball s'ha dissolt.
9- Durant la pràctica no hem seguit la guia d'estil i això ha provocat que al final hàgi hagut de mirar línia per línia buscant errors d'estil.

## Conclusions:

En aquesta pràctica he vist la aplicació directe de la programació modular. M'ha permès organitzar el programa en blocs que he anat connectant a mesura que els anava necessitant. Ha estat molt útil a l'hora de veure l'execució del codi perquè l'he pogut compilar de forma controlada el nostre codi a mesura que l'anava creant.

He pogut treballar per primer cop creant de zero un makefile i això m'ha portat a entendre la funció dels fitxers ".o".

Per començar a programar ha estat molt útil la representació de tots els mòduls en forma d'esquema. També es va representar en forma de màquina d'estats les diferents fases del menú 1 i el menú 2 amb totes les condicions de retorns endavant i endarrere.

És molt important saber molt bé quin és l'objectiu del programa i conèixer totes les capacitats que ofereix el codi C per implementar-lo. Malgrat a l'enunciat ja se'ns deia que havíem d'utilitzar una cua, hem entès que cal conèixer tots els mòduls bàsics que s'utilitzen per a organitzar de forma eficient tota la informació que mou el nostre programa. Cal que al començar a escriure codi es segueixi al màxim la guia d'estil per no haver-ho de fer al final perquè en primer lloc, suposa invertir-hi més temps i, en segon lloc, ajuda a mantenir el codi net i ben estructurat.

Les eines utilitzades han estat l'editor de text VIM i DEVC++ que ens han permès escriure codi i compilar-lo per veure els resultats per pantalla. Hem utilitzat el programa FileZilla per moure fitxers i carpetes del servidor Cygnus al nostre ordinador. A l'inici es va utilitzar Dropbox per intercanviar codi amb el company fins que el grup es va dividir.

La majoria de dubtes i problemes que he anat trobant al llarg de la pràctica els he pogut resoldre a classe i als horaris de dubtes.

Un cop finalitzada la pràctica puc dir que, per una banda, la part més complicada ha estat començar la pràctica utilitzant la programació en mòduls i, per l'altre, la implementació de l'opció "anar d'aventura" del menú de personatge perquè és la que genera un major nombre d'informació a tractar, guardar i mostrar per pantalla.