ALGORITMOS Y ESTRUCTURAS DE DATOS 2023/1

PROFESOR: Manuel Alejandro Moscoso Domínguez
manuel.moscoso.d@gmail.com

Laboratorio Semana 3

En esta oportunidad realizaremos actividades que refuercen el trabajo con variables, punteros y estructuras en C.

Objetivos

- Resolver ejercicios que involucran el uso de punteros y estructuras en C.
- Desarrollar algoritmos que permitan entregar una solución a los problemas entregados.

Ejercicios

Nomenclatura para nombre de archivos fuentes

El nombre del archivo en el cual se almacena el código fuente debe considerar el siguiente formato: labsemanaX_ejercicioY.EXT donde; X corresponde a la semana, la Y al número de ejercicios y EXT a la extensión del lenguaje de programación utilizado.

Ir a la ayuda memoría

Ejercicio número 1

Estamos trabajando en el diseño de un video juego en el cual nos sumaremos al desarrollo y almacenamiento de las estadísticas de los jugadores. Para cada jugador es fundamental contar con su **nombre, nivel, salud y puntaje**. Necesitamos que el trabajo con esta información sea "eficiente" por lo cual se debe implementar en C.

La idea es poder tener una estructura que nos permita almacenar esta información por cada jugador y ver un ejemplo de a lo menos tres jugadores. El ingreso de la información debe ser solicitada al usuario.

Ejercicio número 2

En la modalidad de arena de batalla existirán dos equipos con un máximo de 5 jugadores con el propósito que se puedan enfrentar entre sí. Como es un enfrentamiento creemos que nos falta información dentro de nuestra estructura debido a que no podemos generar "peleas entre ellos".

El objetivo es poder actualizar la estructura diseñada además de trabajar con el o los tipos de datos que nos permitan representar los equipos señalados. Puede utilizar las funciones ya desarrolladas para ingresar la información o generar la información de manera automática, exceptuando los nombres de los jugadores.

La explicación de cómo funciona cada miembro de la estructura debe ser detallada en los comentarios del código.

Ejercicio número 3

Necesitamos que, para complementar la arena de batalla, realizar las acciones que nos permitan simular peleas por turnos. En este caso necesitamos que se pueden desarrollar a través de funciones:

- Ver las estadísticas de un jugador.
- Actualizar las estadísticas de un jugador.
- Determinar el equipo ganador según puntos de experiencia.

La explicación de cómo funciona cada miembro de la estructura debe ser detallada en los comentarios del código.

Ejercicio número 4

Estamos interesados en poder validar completamente la implementación del estado actual de las estadísticas del arena de batalla a través de un “demo” para lo cual nos gustaría poder incorporar en C#, lo desarrollado además de:

- Inicio del juego. El usuario debe definir el número de jugadores por equipo (máximo 5).
- Cantidad de turnos para terminar la partida. Se define un número máximo de turnos para realizar el cálculo de experiencia experiencia.
- Incorporar un dado de acción. El lanzamiento del dato determina si el usuario realiza un ataque o no.
- Selección de rival a atacar. Durante cada turno y cuando se ataca, se debe seleccionar un rival para atacar.
- Los jugadores que llegan a tener 0 puntos de vida no pueden seguir jugando.

Es fundamental que realice todos los cálculos y validaciones para que se pueda simular la jugabilidad. Es importante además considerar umbrales mínimos para ciertos valores además de su explicación a través de mensajes al usuario.

Ayuda memoria

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct {
    char name[50];
    int level;
    float avg;
} Student;
//typedef en C permite definir tipos de datos abstractos con un alias.
//Ayuda a evitar el uso de struct a lo largo de todo el código al trabajar con
//el TDA.

struct patient {
    char name[50];
    int room;
    int age;
};

int main() {

    int integer = 42;
    float floating = 3.14;
    char character = 'A';
    char string[] = "Hello, world!";
    void* pointer = &integer;

    struct patient patient1 = {"Alejandro", 54 , 30};
    Student studentTemp = {"Rocio", 2, 6.6};

    //Uso de malloc
    Student *studentTemp2 = malloc(sizeof(Student));
    studentTemp2->level = 3;
    strcpy(studentTemp2->name, "Manuel");
    studentTemp2->avg = 3.8;

    printf("Entero/Integer: %d\n", integer);
    printf("Flotante/Floating point number: %f\n", floating);
    printf("Caracter/Character: %c\n", character);
    printf("Cadena/String: %s\n", string);
    printf("Puntero/Pointer: %p\n", pointer);
    printf("Precision de 2 decimales: %7.2f\n", floating);
    printf("Entero con simbolo / Integer with plus sign: %+d\n", integer);
    printf("Entero con cero de relleno / Integer with zero padding: %05d\n", integer);

    //Struct
    printf("El paciente de la habitacion %d, tiene %d de edad y su nombre es
%s\n",patient1.room,patient1.age,patient1.name);
    printf("El estudiante %s pertenece al %d° Basico y tiene promedio
%.1f\n",studentTemp.name,studentTemp.level,studentTemp.avg);
    printf("El estudiante %s pertenece al %d° Basico y tiene promedio
%.1f\n",studentTemp2->name,studentTemp2->level,studentTemp2->avg);
```

```
    free(studentTemp2);
    printf("El estudiante %s pertenece al %d° Basico y tiene promedio
%1.1f\n",studentTemp2->name,studentTemp2->level,studentTemp2->avg);

    system("pause");
    return 0;
}
```