Innovación en Formación Profesional

Módulo Profesional 03: Programación

UF2-Act1

Vectores y funciones – Parte3

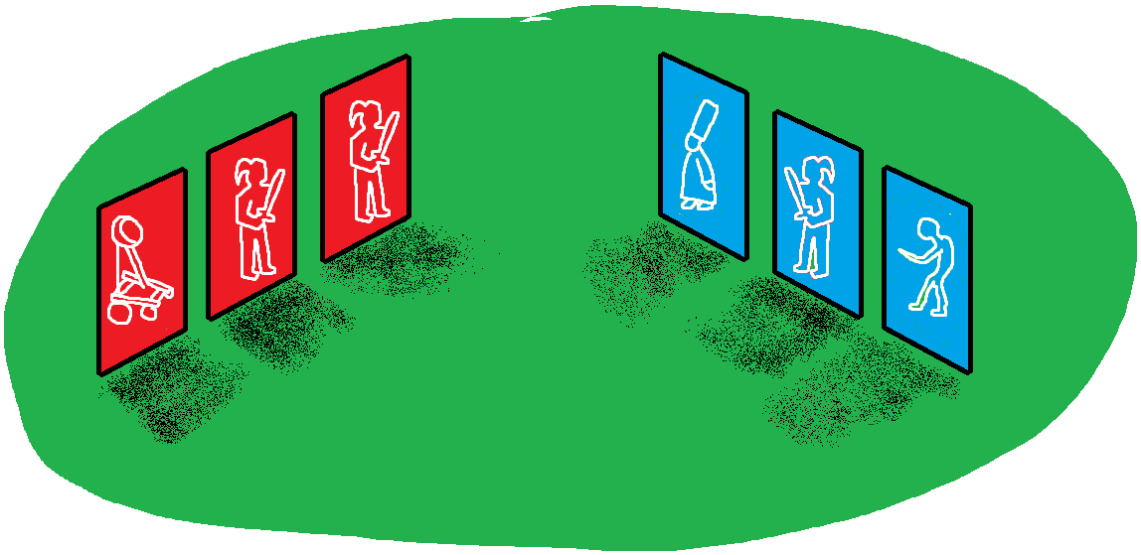
CICLO FORMATIVO DE GRADO SUPERIOR EN

Videojuegos y ocio digital

MODALIDAD PRESENCIAL

Nombre y apellidos del alumno

Introducción



En esta actividad en varias partes aprenderás a utilizar estructuras en C.

Objetivos
Aprender a utilizar estructuras en C

Metodología de evaluación
100% Versión 4

Entrega
A través del campus con nombre UF2-ACT1-P2-NombreApellidos.zip incluyendo enunciado cumplimentado, y una carpeta por versión con los ficheros de código y ejecutables de los programas.

Documentos de referencia
Curso de programación C/C++ Fco. Javier Ceballos

Situación

Para acompañar el lanzamiento de SmartSouls nos han pedido que creemos un juego de cartas.

Diseño de juego (Versión 1)

En esta versión todavía desconocemos las normas del juego, pero sí se ha establecido algunas reglas que nos permiten ir preparando algunos algoritmos que necesitaremos más adelante.

Cartas

Cada jugador tendrá diez cartas, cada una de las cuales tendrá una vida de entre 0 y 100 puntos, admitiéndose números con decimales.

Operaciones

Existirán diferentes acciones en el juego, pero se basarán en las siguientes operaciones:

Operación	Nombre	Descripción
1	IntercambiarPosiciones	Pide al jugador la posición de la primera carta y de la segunda y las intercambia.
2	CalcularVidaMaxima	Calcula y muestra por pantalla el valor máximo de la vida.
3	CalcularVidaMinima	Calcula y muestra por pantalla el valor mínimo de la vida
4	CalcularVidaPromedio	Calcula y muestra por pantalla el valor promedio de la vida
5	DanyarCartasPorDebajoDe	Pide al jugador una cantidad de vida y una cantidad de daño y lo aplica a todas las cartas cuya vida esté por debajo de la cantidad indicada
6	DanyarCartasPorEncimaDe	Pide al jugador una cantidad de vida y una cantidad de daño y lo aplica a todas las cartas cuya vida esté por encima de la cantidad indicada

Operación	Nombre	Descripción
7	DanyarCartasIgualesA	Pide al jugador una cantidad de vida y una cantidad de daño y lo aplica a todas las cartas cuya vida sea exactamente la cantidad indicada
8	ContarCartasPorDebajoDe	Pide al jugador una cantidad de vida, calcula cuántas cartas tienen una vida menor a la indicada y muestra la cifra por pantalla.
9	ContarCartasPorEncimaDe	Pide al jugador una cantidad de vida, calcula cuántas cartas tienen una vida mayor a la indicada y muestra la cifra por pantalla.
10	ContarCartasIgualesA	Pide al jugador una cantidad de vida, calcula cuántas cartas tienen una vida igual a la indicada y muestra la cifra por pantalla.
11	DanyarCartasCercanas	Pide al jugador una posición, una cantidad de daño y una distancia y aplica el daño a todas las cartas cuya posición esté a una distancia menor o igual a la indicada.
12	BuscarTrioMaximo	Busca las tres cartas contiguas que sumen mayor vida y muestra por pantalla el índice de la primera del trío.

Diseño de programa (Versión 1)

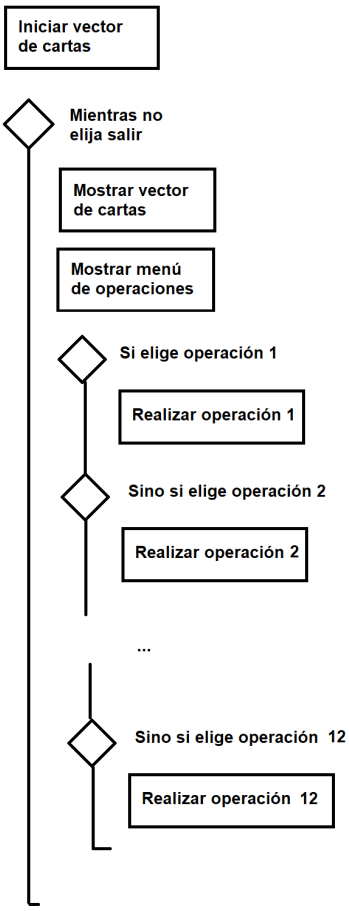
En esta versión el programa mostrará un menú que permitirá al jugador probar las diferentes operaciones.

Variables globales

Nombre	Tipo	Descripción
vidaCartas	float[10]	Vidas de cada carta
opcion	int	Opción elegida en el menú

Programa principal

El programa principal seguirá la siguiente estructura.



Diseño de juego (Versión 2)

En esta versión mantendremos el mismo diseño de juego.

Cartas

Cada jugador tendrá diez cartas, cada una de las cuales tendrá una vida de entre 0 y 100 puntos, admitiéndose números con decimales.

Operaciones

Existirán diferentes acciones en el juego, pero se basarán en las siguientes operaciones:

Operación	Nombre	Descripción
1	IntercambiarPosiciones	Pide al jugador la posición de la primera carta y de la segunda y las intercambia.
2	CalcularVidaMaxima	Calcula y muestra por pantalla el valor máximo de la vida.
3	CalcularVidaMinima	Calcula y muestra por pantalla el valor mínimo de la vida
4	CalcularVidaPromedio	Calcula y muestra por pantalla el valor promedio de la vida
5	DanyarCartasPorDebajoDe	Pide al jugador una cantidad de vida y una cantidad de daño y lo aplica a todas las cartas cuya vida esté por debajo de la cantidad indicada
6	DanyarCartasPorEncimaDe	Pide al jugador una cantidad de vida y una cantidad de daño y lo aplica a todas las cartas cuya vida esté por encima de la cantidad indicada
7	DanyarCartasIgualesA	Pide al jugador una cantidad de vida y una cantidad de daño y lo aplica a todas las cartas cuya vida sea exactamente la cantidad indicada
8	ContarCartasPorDebajoDe	Pide al jugador una cantidad de vida, calcula cuántas cartas tienen una vida menor a la indicada y muestra la cifra por pantalla.
9	ContarCartasPorEncimaDe	Pide al jugador una cantidad de vida, calcula cuántas cartas tienen una vida mayor a la indicada y muestra la cifra por pantalla.
10	ContarCartasIgualesA	Pide al jugador una cantidad de vida, calcula cuántas cartas tienen una vida igual a la indicada y muestra la cifra por pantalla.

Operación	Nombre	Descripción
11	DanyarCartasCercanas	Pide al jugador una posición, una cantidad de daño y una distancia y aplica el daño a todas las cartas cuya posición esté a una distancia menor o igual a la indicada.
12	BuscarTrioMaximo	Busca las tres cartas contiguas que sumen mayor vida y muestra por pantalla el índice de la primera del trío.

Diseño de programa (Versión 2)

En esta versión moveremos el código de las operaciones que hemos creado en la versión anterior a diferentes funciones.

Variables globales

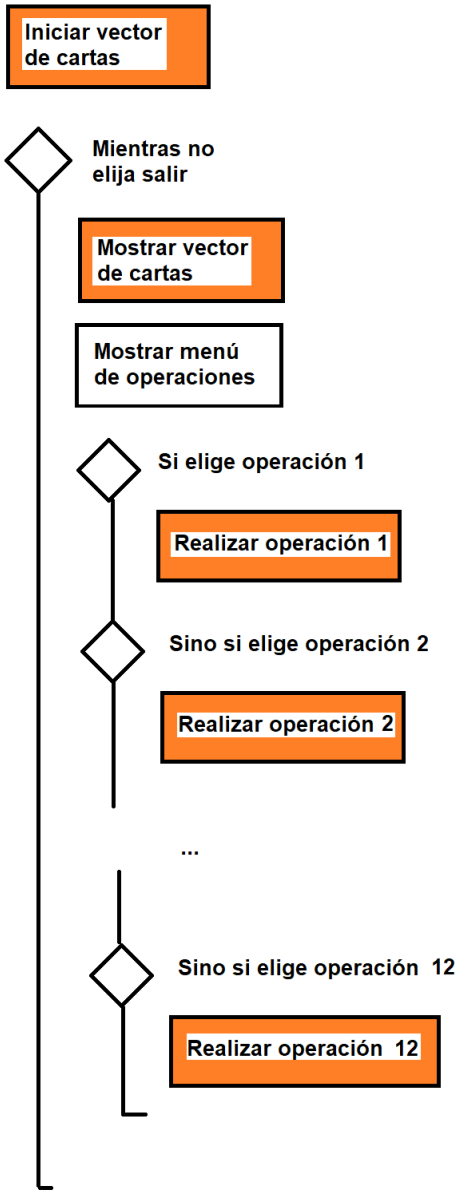
Nombre	Tipo	Descripción
vidaCartas	float[10]	Vidas de cada carta
opcion	int	Opción elegida en el menú

Funciones

Nombre	Descripción
void IniciarVidas(float vidas[])	Inicia a 100 todos los elementos del vector vidas.
void MostrarVidas(float vidas[])	Imprime por pantalla todos los elementos del vector.
float CalculaVidaMaxima(float vidas[])	Devuelve el valor maximo de entre los elementos del vector vidas.
float CalculaVidaMinima(float vidas[])	Devuelve el valor mínimo de entre los elementos del vector vidas.
float CalculaVidaPromedio(float vidas[])	Calcula el valor promedio de entre los elementos del vector vidas.
void DanyarPorDebajoDe(float vidas[], float valor, float danyo)	Aplica el daño indicado a todos los elementos del vector vidas que estén por debajo del valor.
void DanyarPorEncimaDe(float vidas[], float valor, float danyo)	Aplica el daño indicado a todos los elementos del vector vidas que estén por encima del valor.
void DanyarIgualesA(float vidas[], float valor, float danyo)	Aplica el daño indicado a todos los elementos del vector vidas que sean iguales al valor.
int ContarPorDebajoDe(float vidas[], float valor)	Devuelve la cantidad de elementos del vector vidas que son menores que el valor.
int ContarPorEncimaDe(float vidas[], float valor)	Devuelve la cantidad de elementos del vector vidas que son mayores que el valor.
int ContarIgualesA(float vidas[], float valor)	Devuelve la cantidad de elementos del vector vidas que son iguales al valor.
void DanyarCercanas(float vidas[], int posicion, int distancia, float danyo)	Aplica el daño a todos los elementos del vector cuyo índice está a una distancia de la posición menor que la indicada.
int BuscarTrioMaximo(float vidas[])	Encuentra el trío de elementos contiguos con mayor valor y devuelve la posición del primero

Programa principal

El programa principal seguirá la siguiente estructura, apoyándose en las funciones creadas.



Diseño de juego (Versión 3)

Cartas

Cada jugador tendrá diez cartas, cada una de las cuales tendrá una vida de entre 0 y 100 puntos, admitiéndose números con decimales.

Cada carta repreenta un tipo de personaje del juego. Algunas cartas sólo pueden atacar a las cartas del contrario que están frente a ellas, otras pueden atacar a cualquier carta que cumpla unas ciertas condiciones.

Los tipos de cartas son.

Tipo	Objetivo	Daño
SOLDIER	Sólo ataca a la carta que tenga enfrente.	1-5
KINGSLAYER	Ataca sólo a la carta (o cartas) que tengan más vida.	20-30
SCAVENGER	Ataca sólo a la carta (o cartas) que tengan menos vida.	20-30
PREDATOR	Ataca sólo a las cartas que tengan una vida por encima del promedio.	5-10
DARWIN	Ataca sólo a las cartas que tengan una vida por debajo del promedio.	5-10
PLAGUE	Ataca al grupo de cartas más numeroso, eligiéndolo entre los que tienen más de cincuenta puntos de vida y los que tienen menos.	5-10
DOOM	Ataca al trío de cartas contiguas que combinadas tengan más vida	10-20
WILDFIRE	Ataca a la carta que tenga enfrente y a las que estén en un radio de 2	5-10

Desarrollo de una partida

Los dos jugadores reciben 10 cartas, cinco de tipo SOLDIER y cinco de cualquiera de los otros tipos, al azar.

Empieza uno de los dos jugadores al azar.

En su turno, el jugador elige entre los siguientes movimientos.

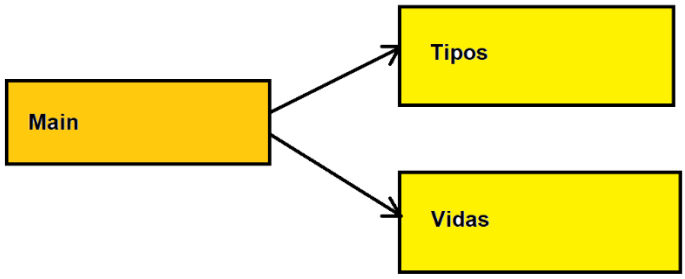
Movimiento	Descripción
Intercambiar cartas	El jugador intercambia las posiciones de dos de sus cartas, o la de una carta y un hueco vacío.
Atacar	El jugador elige la carta con que quiere atacar y ejecuta el ataque.

Cuando mueren todas las cartas de un jugador, el juego acaba.

Diseño de programa (Versión 3)

En esta versión crearemos funciones para gestionar los tipos de carta y moveremos las funciones a diferentes módulos.

Módulos



Nombre	Descripción
Tipos	Gestiona los vectores de tipos de cartas
Vidas	Gestiona los vectores de vidas de cartas
Main	Gestiona el bucle de juego y la interacción con los jugadores

Módulo Vidas

Funciones

Nombre	Descripción
void IniciarVidas(float vidas[])	Inicia a 100 todos los elementos del vector vidas.
void MostrarVidas(float vidas[])	Imprime por pantalla todos los elementos del vector.
float CalculaVidaMaxima(float vidas[])	Devuelve el valor maximo de entre los elementos del vector vidas.
float CalculaVidaMinima(float vidas[])	Devuelve el valor mínimo de entre los elementos del vector vidas.
float CalculaVidaPromedio(float vidas[])	Calcula el valor promedio de entre los elementos del vector vidas.
void DanyarPorDebajoDe(float vidas[], float valor, float danyo)	Aplica el daño indicado a todos los elementos del vector vidas que estén por debajo del valor.
void DanyarPorEncimaDe(float vidas[], float valor, float danyo)	Aplica el daño indicado a todos los elementos del vector vidas que estén por encima del valor.
void DanyarIgualesA(float vidas[], float valor, float danyo)	Aplica el daño indicado a todos los elementos del vector vidas que sean iguales al valor.
int ContarPorDebajoDe(float vidas[], float valor)	Devuelve la cantidad de elementos del vector vidas que son menores que el valor.
int ContarPorEncimaDe(float vidas[], float valor)	Devuelve la cantidad de elementos del vector vidas que son mayores que el valor.
int ContarIgualesA(float vidas[], float valor)	Devuelve la cantidad de elementos del vector vidas que son iguales al valor.
void DanyarCercanas(float vidas[], int posicion, int distancia, int danyo)	Aplica el daño a todos los elementos del vector cuyo índice está a una distancia de la posición menor que la indicada.
int BuscarTrioMaximo(float vidas[])	Encuentra el trío de elementos contiguos con mayor valor y devuelve la posición del primero
void IntercambiarVidas(int vidas[], int indice1, int indice2)	Intercambia las vidas que están en el índice 1 el índice 2 del vector.

Módulo Tipos

En el módulo tipos utilizaremos la siguiente codificación para los tipos de carta

0 – Soldier.

1 – Kingslayer

2 – Scavenger

3 – Predator

4 – Darwin

5 – Plague

6 – Doom

7 - Wildfire

Funciones

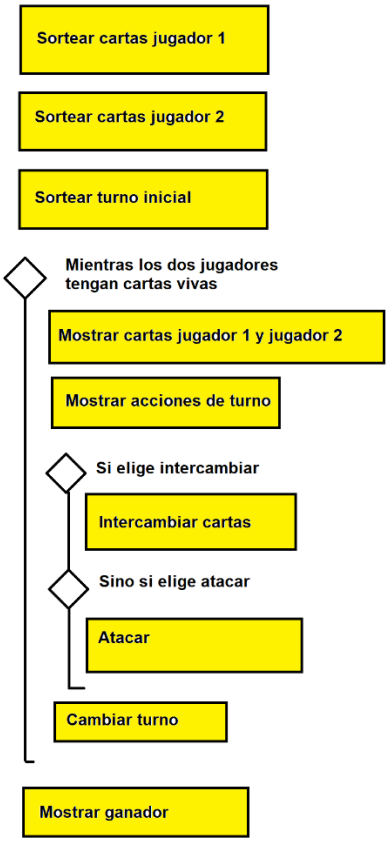
Nombre	Descripción
void IniciarTipos(int tipos[])	Inicia el vector con los tipos iniciales de las cartas.
void MostrarTipos(int tipos[])	Imprime por pantalla todos los tipos contenidos en el vector.
void IntercambiarTipos(int tipos[], int indice1, int indice2)	Intercambia los tiposque están en el índice 1 el índice 2 del vector.

Programa principal

Variables globales

Nombre	Tipo	Descripción
vidaCartas1	float[10]	Vida de cada carta del jugador 1
tipoCartas1	int[10]	Tipos de cartas del jugador 1
vidaCartas2	float[10]	Vida de cada carta del jugador 2
tipoCartas2	int[10]	Tipos de cartas del jugador 2
accion	int	Accion elegida en el turno.
turno	int	0 si es del jugador 1 y 1 si es del jugador 2

Estructura del programa



Ejercicio 1: Crea la versión 3 del programa y graba un vídeo corto (2-3 minutos) en que muestres su funcionamiento, incluido el código. Si lo alojas en un servicio externo, incluye el enlace a continuacion.

Diseño de juego (Versión 4)

En esta versión mantendremos el mismo diseño de juego.

Cartas

Cada jugador tendrá diez cartas, cada una de las cuales tendrá una vida de entre 0 y 100 puntos, admitiéndose números con decimales.

Operaciones

Existirán diferentes acciones en el juego, pero se basarán en las siguientes operaciones:

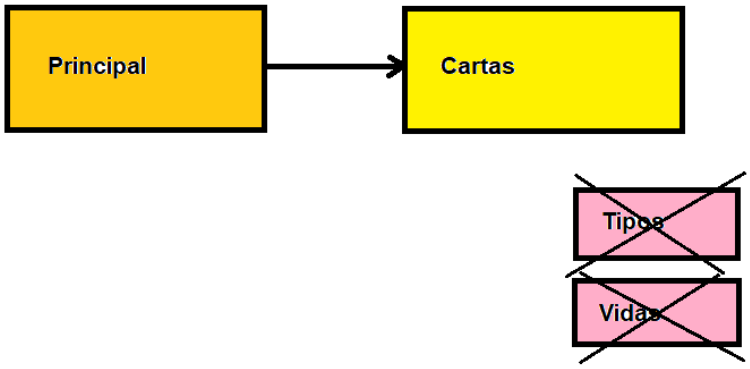
Operación	Nombre	Descripción
1	IntercambiarPosiciones	Pide al jugador la posición de la primera carta y de la segunda y las intercambia.
2	CalcularVidaMaxima	Calcula y muestra por pantalla el valor máximo de la vida.
3	CalcularVidaMinima	Calcula y muestra por pantalla el valor mínimo de la vida
4	CalcularVidaPromedio	Calcula y muestra por pantalla el valor promedio de la vida
5	DanyarCartasPorDebajoDe	Pide al jugador una cantidad de vida y una cantidad de daño y lo aplica a todas las cartas cuya vida esté por debajo de la cantidad indicada
6	DanyarCartasPorEncimaDe	Pide al jugador una cantidad de vida y una cantidad de daño y lo aplica a todas las cartas cuya vida esté por encima de la cantidad indicada
7	DanyarCartasIgualesA	Pide al jugador una cantidad de vida y una cantidad de daño y lo aplica a todas las cartas cuya vida sea exactamente la cantidad indicada
8	ContarCartasPorDebajoDe	Pide al jugador una cantidad de vida, calcula cuántas cartas tienen una vida menor a la indicada y muestra la cifra por pantalla.
9	ContarCartasPorEncimaDe	Pide al jugador una cantidad de vida, calcula cuántas cartas tienen una vida mayor a la indicada y muestra la cifra por pantalla.
10	ContarCartasIgualesA	Pide al jugador una cantidad de vida, calcula cuántas cartas tienen una vida igual a la indicada y muestra la cifra por pantalla.

Operación	Nombre	Descripción
11	DanyarCartasCercanas	Pide al jugador una posición, una cantidad de daño y una distancia y aplica el daño a todas las cartas cuya posición esté a una distancia menor o igual a la indicada.
12	BuscarTrioMaximo	Busca las tres cartas contiguas que sumen mayor vida y muestra por pantalla el índice de la primera del trío.

Diseño de programa (Versión 4)

En esta versión desaparecerán los módulos tipos y vidas y se sustituirán por un módulo llamado cartas que contará con una estructura que incluirá todas las propiedades.

Módulos



Nombre	Descripción
Principal	Gestiona el bucle de juego y la interacción con los jugadores
Cartas	Gestiona las cartas
~~Tipos~~	~~Gestiona los vectores de tipos de cartas~~
~~Vidas~~	~~Gestiona los vectores de vidas de cartas~~

Módulo Cartas

En el módulo carta utilizaremos la siguiente codificación para los tipos de carta

0 – Soldier.

1 – Kingslayer

2 – Scavenger

3 – Predator

4 – Darwin

5 – Plague

6 – Doom

7 - Wildfire

Estructuras

Carta		
Campo	**Tipo**	**Descripción**
tipo	Int	Tipo de carta
vida	Float	Vida que tiene la carta

Funciones

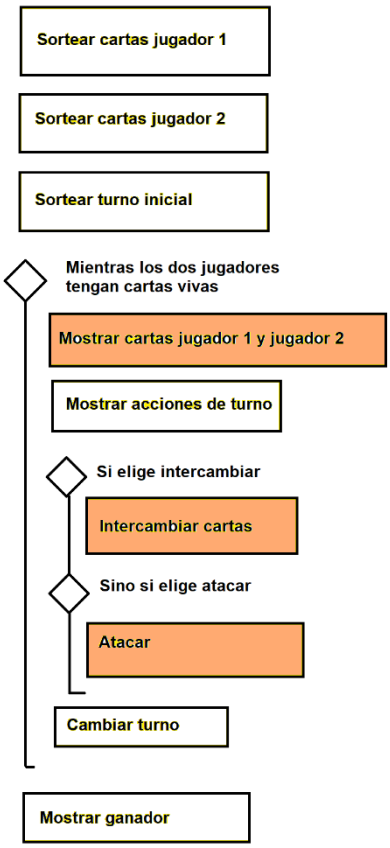
Nombre	Descripción
void IniciaCartas(Carta cartas[])	Inicia la vida de todos los elementos del vector cartas a 100 y el tipo a 0.
void MostrarCartas(Carta cartas[])	Imprime por pantalla todos los elementos del vector cartas.
float CalculaVidaMaxima(Carta cartas[])	Devuelve el valor maximo de vida de entre los elementos del vector cartas.
float CalculaVidaMinima(Carta carta[])	Devuelve el valor mínimo de vida de entre los elementos del vector cartas.
float CalculaVidaPromedio(Carta cartas[])	Calcula el valor promedio de vida de entre los elementos del vector cartas.
void DanyarPorDebajoDe(Carta cartas[], float valor, float danyo)	Aplica el daño indicado a todos los elementos del vector cartas cuya vida esté por debajo del valor.
void DanyarPorEncimaDe(Carta cartas[], float valor, float danyo)	Aplica el daño indicado a todos los elementos del vector cartas cuya vida esté por encima del valor.
void DanyarIgualesA(Carta cartas[], float valor, float danyo)	Aplica el daño indicado a todos los elementos del vector cartas cuya vida sea igual al valor.
int ContarPorDebajoDe(Carta cartas[], float valor)	Devuelve la cantidad de elementos del vector cartas cuya vida sea menor que el valor.
int ContarPorEncimaDe(Carta cartas[], float valor)	Devuelve la cantidad de elementos del vector cartas cuya vida sea mayor que el valor.
int ContarIgualesA(Carta cartas[], float valor)	Devuelve la cantidad de elementos del vector cartas cuya vida sea igual al valor.
void DanyarCercanas(Carta cartas[], int posicion, int distancia, int danyo)	Aplica el daño a todos los elementos del vector cartas cuyo índice está a una distancia de la posición menor que la indicada.
int BuscarTrioMaximo(Carta cartas[])	Encuentra el trío de elementos contiguos con mayor valor conjunto de vida y devuelve la posición del primero
void IntercambiarCartas(Carta cartas[], int indice1, int indice2)	Intercambia las cartas que están en el índice 1 el índice 2 del vector.

Programa principal

Variables globales

Nombre	Tipo	Descripción
~~vidaCartas1~~	~~float[10]~~	~~Vida de cada carta del jugador 1~~
~~tipoCartas1~~	~~int[10]~~	~~Tipos de cartas del jugador 1~~
~~vidaCartas2~~	~~float[10]~~	~~Vida de cada carta del jugador 2~~
~~tipoCartas2~~	~~int[10]~~	~~Tipos de cartas del jugador 2~~
cartas1	Carta[10]	Cartas del jugador 1
cartas2	Carta[10]	Cartas del jugador 2
accion	int	Accion elegida en el turno.
turno	int	0 si es del jugador 1 y 1 si es del jugador 2

Estructura del programa



Ejercicio 1: Crea la versión 4 del programa.