AMERIKE INSTITUTO UNIVERSITARIO

Nombre: Luis Fernando González Chávez **No. de Matrícula.:** ZAP408

Materia: Fundamentos de la Programación Grupo**:** Dev 22-1 Turno**:** Matutino

Carrera: Ingeniería en Desarrollo de Software Interactivo y Videojuegos

Tema: Class **No: R.1** 18

Fecha propuesta: 04/Nov/2021 **Fecha de Entrega:** 03/Nov/2021

Escuela: Instituto Universitario Amerike **Plantel** Guadalajara

Calle: C. Montemorelos **No:** 3503 **Colonia:** Rinconada de la Calma **C.P.:** 45080

Teléfono: 3336326100 **Ciudad:** Zapopan

Ferchus

Firma del alumno (a) **Firma de revisión fecha**

Qué se evalúa:	10 pts.	7 pts.	4 pts.	
Entrega electrónica	Es en tiempo y forma al iniciar la clase. (1 pts.)	Después de 30 minutos de iniciada la clase. (.7 pts.)	Al minuto 40. (Posteriormente ya no se reciben). (.4pts.)	
Del formato.	Cumple con todos los elementos solicitados. (1 pts.)	No cumple con dos elementos solicitados. (.7 pts.)	No cumple con tres o más elementos solicitados. (.4pts.)	
La ortografía.	Tiene dos errores ortográficos. (1 pts.)	Tiene de tres a cuatro errores ortográficos. (.7 pts.)	Tiene cinco o más errores ortográficos. (.4pts.)	
Del tema y objetivo.	La teoría y ejemplos corresponden al tema tratado. (1 pts.)	La teoría o ejemplos no corresponden al tema tratado. (.7 pts.)	La teoría y ejemplos no corresponden al tema tratado. (.4pts.)	
El programa y los cálculos.	Los parámetros y componentes corresponden al 100% de lo planeado. (1 pts.)	El programa arroja un error o componente no corresponden al 100% de lo planeado. (7 pts.)	El programa arroja dos errores o componentes no corresponden al 100% de lo calculado. (.4pts.)	
Diagramas.	Los diagramas a bloques, de flujo y esquemáticos son acorde al de la práctica y siguen una secuencia lógica. (1 pts.)	Los diagramas a bloques, o de flujo o esquemáticos no son acorde al de la práctica y o no siguen una secuencia lógica. (.7 pts.)	Los diagramas a bloques, de flujo y esquemáticos no son acorde al de la práctica y o no siguen una secuencia lógica. (.4pts.)	
La tabla de valores.	Los valores calculados y medidos presentan una desviación máxima del 10%. (1 pts.)	Los valores calculados y medidos presentan una desviación máxima del 15%. (. 7 pts.)	Los valores calculados y medidos presentan una desviación máxima del 20%. (.4pts.)	
Las observaciones y conclusiones.	Son específicas y congruentes con la práctica. (1 pts.)	Las observaciones o conclusiones son específicas y congruentes con la práctica. (.7 pts.)	Las observaciones y las conclusiones no son específicas y congruentes con la práctica. (.4pts.)	
Bibliografía.	Es acorde al (los) tema (s) tratado (s) y está completa (1 pts.)	Es acorde a algún (os) tema (s) tratado (s), le falta algún elemento que la conforman (.7 pts.)	No es acorde al (los) tema (s) tratado (s), le faltan 2 elementos que la conforma (.4pts.)	
Fuentes de consulta.	Es acorde al (los) tema (s) tratado (s) (1 pts.)	Es acorde a algún (os) tema (s) tratado (s) (.7 pts.)	Es acorde a algún (los) tema (s) tratado (s) (.4pts.)	

Índice

Teoría

Clases: "Una clase es en general un modelo, receta o plantilla que define el estado y comportamiento de cierto tipo de objetos. Una clase puede pensarse como una colección de variables (atributos o propiedades) y funciones (métodos) que permiten representar un conjunto de datos y especificar las operaciones o procedimientos que permiten manipular tales datos." (Camilo Correa, 2019).

Objetos: "Un objeto es una instancia de una clase, es decir una entidad que se construye a partir de las descripciones consignadas en una clase (datos y funciones). Por tanto, un objeto se puede entender como una "variable" que se declara del tipo de dato de cierta clase. Un objeto es como tal la entidad tangible que permite acceder a los datos y funciones modeladas al interior de la clase." (Camilo Correa, 2019).

Estructuras: "Técnicamente, una estructura es lo mismo que una clase; la única diferencia es que una clase es privada por defecto, mientras que una estructura es pública. La razón por la que siguen existiendo es porque en el lenguaje C no existían las clases, pero sí las estructuras. Si quitáramos la palabra "struct" por completo, le estaríamos quitando la compatibilidad entre C y C++, pues el compilador no sabría qué significa.

¿Cuándo debo usar clases y cuando estructuras? Muy simple: Si quieres que todos los miembros sean públicos pero no quieres escribir "public:", usa "struct"; así de simple. Usualmente uno no usa estructuras si es que va a haber una jerarquía de herencia, pues le añade otro nivel de complejidad y en las estructuras siempre quieres tener valores simples." (Fernando González, 2021).

Cálculos

```
for (int i = 0; i < númeroAlumnos; i++)
{
Alumno[i].swag = rand() % 100;
Alumno[i].drip = rand() % 100;
Alumno[i].flow = rand() % 100;
Alumno[i].facha = rand() % 100;
}
```

Debemos darle un valor aleatorio a las propiedades de cada uno de los alumnos. En este caso, es entre 0 y 99.

```
int alumnoPowSwag = rand() % númeroAlumnos;
int alumnoPowDrip = rand() % númeroAlumnos;
int alumnoPowFlow = rand() % númeroAlumnos;
int alumnoPowFacha = rand() % númeroAlumnos;
```

Asignamos una variable que almacene la posición del alumno al que le vamos a dar cada Power-Up. Un número aleatorio entre 0 y númeroAlumnos – 1.

```
Alumno[alumnoPowSwag].swag = Alumno[alumnoPowSwag].swag + rand() % 100;
Alumno[alumnoPowDrip].drip = Alumno[alumnoPowDrip].drip + rand() % 100;
Alumno[alumnoPowFlow].flow = Alumno[alumnoPowFlow].flow + rand() % 100;
Alumno[alumnoPowFacha].facha = Alumno[alumnoPowFacha].facha + rand() % 100;
```

Le añadimos el Power-Up a los alumnos en las posiciones que obtuvimos con el cálculo anterior.

```
int mayorSwag = 0;
int menorSwag = 0;
int mayorDrip = 0;
int menorDrip = 0;
int mayorFlow = 0;
int menorFlow = 0;
int mayorFacha = 0;
int menorFacha = 0;

for (int i = 1; i < númeroAlumnos; i++)
{
if (Alumno[i].swag > Alumno[mayorSwag].swag) mayorSwag = i;

if (Alumno[i].swag < Alumno[menorSwag].swag) menorSwag = i;

if (Alumno[i].drip > Alumno[mayorDrip].drip) mayorDrip = i;

if (Alumno[i].drip < Alumno[menorDrip].drip) menorDrip = i;

if (Alumno[i].flow > Alumno[mayorFlow].flow) mayorFlow = i;

if (Alumno[i].flow < Alumno[menorFlow].flow) menorFlow = i;

if (Alumno[i].facha > Alumno[mayorFacha].facha) mayorFacha = i;

if (Alumno[i].facha < Alumno[menorFacha].facha) menorFacha = i;
}
```

Esto lo hacemos para obtener al alumno con menor y mayor valor en cada propiedad. Empezamos declarando “0” como la posición en la que está el alumno con mayor y menor de todo. En el bucle, iremos comparando cada alumno con las propiedades del alumno en la posición de mayor y menor de cada propiedad. Si se cumplen las condiciones, la posición se actualiza y la siguiente vez se comparará con el alumno en esta nueva posición.

Diagramas

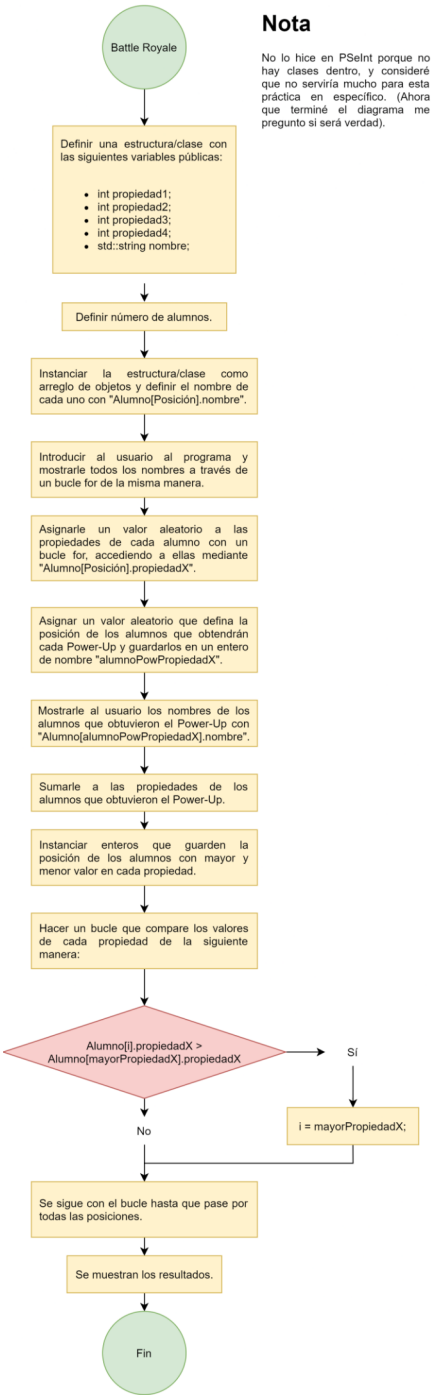
Battle Royale
Nota
No lo hice en PSeInt porque no hay clases dentro, y consideré que no serviría mucho para esta práctica en específico. (Ahora que terminé el diagrama me pregunto si será verdad).
Definir una estructura/clase con las siguientes variables públicas:
int propiedad1;
int propiedad2;
int propiedad3;
int propiedad4;
std::string nombre;
Definir número de alumnos.
Instanciar la estructura/clase como arreglo de objetos y definir el nombre de cada uno con "Alumno[Posición].nombre".
Introducir al usuario al programa y mostrarle todos los nombres a través de un bucle for de la misma manera.
Asignarle un valor aleatorio a las propiedades de cada alumno con un bucle for, accediendo a ellas mediante "Alumno[Posición].propiedadX".
Asignar un valor aleatorio que defina la posición de los alumnos que obtendrán cada Power-Up y guardarlos en un entero de nombre "alumnoPowPropiedadX".
Mostrarle al usuario los nombres de los alumnos que obtuvieron el Power-Up con "Alumno[alumnoPowPropiedadX].nombre".
Sumarle a las propiedades de los alumnos que obtuvieron el Power-Up.
Instanciar enteros que guarden la posición de los alumnos con mayor y menor valor en cada propiedad.
Hacer un bucle que compare los valores de cada propiedad de la siguiente manera:
Alumno[i].propiedadX > Alumno[mayorPropiedadX].propiedadX
Sí
i = mayorPropiedadX;
No
Se sigue con el bucle hasta que pase por todas las posiciones.
Se muestran los resultados.
Fin

Tabla (comparativa)

Acción	Expectativa	Resultado	Conclusión
Instanciación de la estructura de alumno.	Se podrán acceder a todas sus variables públicas dentro de main().	Se pueden acceder a todas sus variables públicas dentro de main(). Funciona correctamente, pero el compilador me da advertencias de que no inicié las propiedades.	Funciona correctamente pero probablemente está propensa a tener errores.
Bucle para mostrar nombres.	Mostrará todos los nombres en orden.	Muestra todos los nombres en orden.	Funciona correctamente.
Bucle para asignar valores.	Le dará valores aleatorios del 0 al 99 a las propiedades de cada alumno.	Le da valores aleatorios del 0 al 99 a las propiedades de cada alumno.	Funciona correctamente.
Asignación alumno Power-Up	Definirá qué alumno recibirá cada Power-Up.	Define qué alumno recibirá cada Power-Up.	Funciona correctamente.
Suma Power-Up	Le agregará un valor entre 0 y 99 a las propiedades de los alumnos que obtuvieron el Power-Up	Le agrega un valor entre 0 y 99 a las propiedades de los alumnos que obtuvieron el Power-Up	Funciona correctamente
Bucle comparar propiedades	Comprobará las propiedades de cada alumno con los valores que tiene el mayor y menor de cada una. Si la comprobación es cierta, se actualizará el valor de mayor y menor.	Comprueba las propiedades de cada alumno con los valores que tiene el mayor y menor de cada una. Si la comprobación es cierta, se actualiza el valor de mayor y menor.	Funciona correctamente.
Mostrar resultados	Se mostrarán los resultados de qué alumno fue el que tuvo mayor y menor valor de cada propiedad.	Se muestran los resultados de qué alumno fue el que tuvo mayor y menor valor de cada propiedad.	Funciona correctamente.

Bibliografía

Stroustrup, B. (2013). 16 Classes. En The C++ Programming Language (4th ed., p. 449). Addison-Wesley Professional.

Fuentes de consulta

Correa, C. (2019). *Clases y Objetos en C++ - Clases y Objetos en C++ (Práctica 1)*. CodinGame. Recuperado 3 de noviembre de 2021, de https://www.codingame.com/playgrounds/50557/clases-y-objetos-en-c-practica-1/clases-y-objetos-en-c

Chernikov, Y. (2017j, junio 25). CLASSES in C++. YouTube.

https://www.youtube.com/watch?v=2BP8NhxjrO0

Chernikov, Y. (2017k, julio 2). CLASSES vs STRUCTS in C++. YouTube.

https://www.youtube.com/watch?v=fLgTtaqqJp0