

Nombre: Ian Axel Hernández Ortega **No. de Matrícula.:** Zap288

Materia: Fundamentos de Programación **Grupo:** 21-1 **Turno:** Matutino

Carrera: Licenciatura en Desarrollo de Software Interactivo y Videojuegos

Tema: Practica 9, Battle Royale **No:** P.8

Fecha propuesta: 25 – 11 – 2020 **Fecha de Entrega:** 30 – 11 – 2020

Escuela: Instituto Universitario Amerike

Plantel: Guadalajara

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

Teléfono: 33 3632 6100

Ciudad: Zapopan

Logotipo personal



Logotipo (de la escuela)



IanAxelHernándezOrtega

Firma del alumno (a)

Qué se evalúa:	10 pts.	7 pts.	4pts.	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.)	

Nombre: Ian Axel Hernández Ortega

Práctica: Practica9, Battle Royale

No. P-9

Página 1

Índice	pag 2
Teoría	pag 2
Cálculos	pag 2
Diagramas	pag 2 y 3
Tabla Comparativa	pag 3
Observaciones	pag 3
Conclusiones	pag 3
Bibliografía	pag 3
Fuentes de consulta	pag 4

Teoría

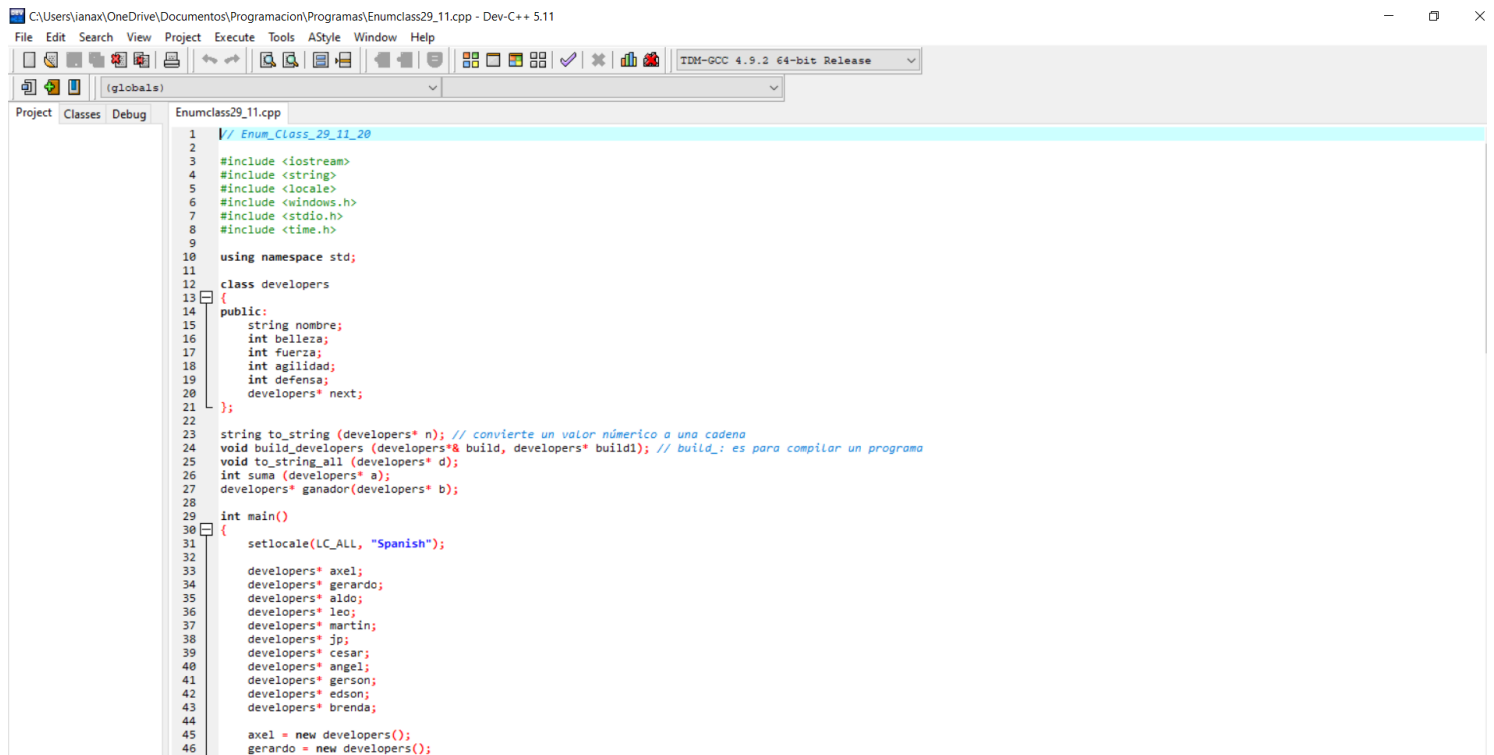
Una enumeración o enum es un tipo definido por el usuario que consta de un conjunto de constantes integrales con nombre que se conocen como enumeradores.

Cálculos

```
enum [identifier] [: type]
{enum-list};
```

```
// scoped enum:
enum [class|struct]
[identifier] [: type]
{enum-list};
```

Diagramas



```
1 // Enum_Class_29_11_20
2
3 #include <iostream>
4 #include <string>
5 #include <locale>
6 #include <windows.h>
7 #include <stdio.h>
8 #include <time.h>
9
10 using namespace std;
11
12 class developers
13 {
14 public:
15     string nombre;
16     int belleza;
17     int fuerza;
18     int agilidad;
19     int defensa;
20     developers* next;
21 };
22
23 string to_string (developers* n); // convierte un valor númerico a una cadena
24 void build_developers (developers*& build, developers* build1); // build_ es para compilar un programa
25 void to_string_all (developers* d);
26 int suma (developers* a);
27 developers* ganador(developers* b);
28
29 int main()
30 {
31     setlocale(LC_ALL, "Spanish");
32
33     developers* axel;
34     developers* gerardo;
35     developers* aldo;
36     developers* leo;
37     developers* martin;
38     developers* jp;
39     developers* cesar;
40     developers* angel;
41     developers* gerson;
42     developers* edson;
43     developers* brenda;
44
45     axel = new developers();
46     gerardo = new developers();
```

C:\Users\ianax\OneDrive\Documentos\Programacion\Programas\Enumclass29_11.cpp - Dev-C++ 5.11

File Edit Search View Project Execute Tools AStyle Window Help

IDM-GCC 4.9.2 64-bit Release

(globals)

```

Project Classes Debug Enumclass29_11.cpp
45 | axel = new developers();
46 | gerardo = new developers();
47 | aldo = new developers();
48 | leo = new developers();
49 | martin = new developers();
50 | jp = new developers();
51 | cesar = new developers();
52 | angel = new developers();
53 | gerson = new developers();
54 | edson = new developers();
55 | brenda = new developers();
56 |
57 | build_developers(axel, gerardo);
58 | build_developers(aldo, leo);
59 | build_developers(martin, jp);
60 | build_developers(cesar, angel);
61 | build_developers(gerson, edson);
62 | build_developers(brenda, NULL);
63 |
64 | to_string_all (axel);
65 |
66 | cout << "El ganador es: " << endl;
67 | cout << to_string (ganador(axel));
68 |
69 | }
70 |
71 |
72 | void build_developers (developers* & build, developers* build1)
73 | {
74 |     srand(time(NULL));
75 |     build->belleza = rand() %10;
76 |     build->fuerza = rand() %10;
77 |     build->agilidad = rand() %10;
78 |     build->defensa = rand() %10;
79 |     build->next = build1;
80 |
81 |     cout << "Ingresa un nombre: ";
82 |
83 |     getline(cin, build->nombre);
84 | }
85 |
86 | void to_string_all(developers* n)
87 | {
88 |     while (n != NULL)
89 |     {
90 |         cout << to_string(n);
91 |         n = n->next;
92 |     }
93 | }
94 |
95 | string to_string (developers* d)
96 | {
97 |     string resultado;
98 |
99 |     resultado += d->nombre + "\n";
100 |     resultado += "Obtiene: " + to_string(d->belleza) + "Score de belleza: " + "\n";
101 |     resultado += "Obtiene: " + to_string(d->fuerza) + "Score de fuerza: " + "\n";
102 |     resultado += "Obtiene: " + to_string(d->agilidad) + "Score de agilidad: " + "\n";
103 |     resultado += "Obtiene: " + to_string(d->defensa) + "Score de defensa: " + "\n";
104 |
105 |     return resultado;
106 | }
107 |
108 |
109 | int suma(developers* a)
110 | {
111 |     if (a==NULL)
112 |     {
113 |         return 0;
114 |     }
115 |     int resultado = s->belleza + s->fuerza + s->agilidad + s->defensa;
116 |     return resultado;
117 | }
118 |
119 | developers* ganador(developers* b)
120 | {
121 |     developers* ganador = b;
122 |     while (b != NULL)
123 |     {
124 |         if (!(suma(ganador)>suma(d->next)))
125 |         {
126 |             ganador = d->next;
127 |         }
128 |         d = d->next;
129 |     }
130 |     return ganador;
131 | }

```

C:\Users\ianax\OneDrive\Documentos\Programacion\Programas\Enumclass29_11.cpp - Dev-C++ 5.11

File Edit Search View Project Execute Tools AStyle Window Help

IDM-GCC 4.9.2 64-bit Release

(globals)

```

Project Classes Debug Enumclass29_11.cpp
86 | void to_string_all(developers* n)
87 | {
88 |     while (n != NULL)
89 |     {
90 |         cout << to_string(n);
91 |         n = n->next;
92 |     }
93 | }
94 |
95 | string to_string (developers* d)
96 | {
97 |     string resultado;
98 |
99 |     resultado += d->nombre + "\n";
100 |     resultado += "Obtiene: " + to_string(d->belleza) + "Score de belleza: " + "\n";
101 |     resultado += "Obtiene: " + to_string(d->fuerza) + "Score de fuerza: " + "\n";
102 |     resultado += "Obtiene: " + to_string(d->agilidad) + "Score de agilidad: " + "\n";
103 |     resultado += "Obtiene: " + to_string(d->defensa) + "Score de defensa: " + "\n";
104 |
105 |     return resultado;
106 | }
107 |
108 |
109 | int suma(developers* a)
110 | {
111 |     if (a==NULL)
112 |     {
113 |         return 0;
114 |     }
115 |     int resultado = s->belleza + s->fuerza + s->agilidad + s->defensa;
116 |     return resultado;
117 | }
118 |
119 | developers* ganador(developers* b)
120 | {
121 |     developers* ganador = b;
122 |     while (b != NULL)
123 |     {
124 |         if (!(suma(ganador)>suma(d->next)))
125 |         {
126 |             ganador = d->next;
127 |         }
128 |         d = d->next;
129 |     }
130 |     return ganador;
131 | }

```

Tabla comparativa

Datos ingresados	Datos Esperados	Datos Obtenidos
10 puntos	10 puntos	10 puntos
Axel	Axel	Axel
Agilidad	Agilidad	Agilidad

Observaciones

Al ingresar un nombre de los que vienen dentro del programa, el random hace que expulse los datos concretos de acuerdo al puntaje que se le había asignado, dando así resultados diferentes cada vez que se ejecuta el programa.

Conclusiones

Este es uno de los programas que están por terminarse, pero es verdad que la forma en como los datos ya pre-programados con el random, cada random que se reproduce en el programa nos da una forma distinta de quien es el mejor o el ganador de todos los devs. En este caso los devs y los contendientes estamos peleando por saber quién es el mejor de todos nosotros.

Bibliografía

D. (2088). *Enum c++* (1.^a ed., Vol. 1) [Libro electrónico]. eni.com. <https://www.ediciones-eni.com/open/mediabook.aspx?idR=1ce6ca88d795dcaaaca46a7a3871b09e>

Fuentes de consulta

D. (2018, 1 junio). *Enumeraciones c++*. Enum.com. <https://docs.microsoft.com/en-us/cpp/cpp/enumerations-cpp?view=msvc-160>

