

GRADO EN INTELIGENCIA ARTIFICIAL (MOSTOLES)

2361 - ESTRUCTURAS DE DATOS II - TARDE A - 2Q

 > [Evaluación](#) > Prueba Parcial - Introducción

Comenzado el	miércoles, 14 de febrero de 2024, 16:17
Estado	Finalizado
Finalizado en	miércoles, 14 de febrero de 2024, 16:37
Tiempo empleado	20 minutos
Calificación	5,32 de 10,00 (53,21%)

Pregunta 1

Finalizado

Se puntúa 0,50 sobre 1,00

[1 punto] Dado el siguiente código en C, escribe un código equivalente en C++ usando todas las **herramientas vistas en clase**.

```
#define N 5

int a[N];

a[0] = 42;
a[1] = 17;
a[2] = 3;
a[3] = 9;
a[4] = 37;

for(int i = 0; i < N; i++) {
    printf("%d ", a[i]);
}
printf("\n");
```

```
#define N 5
auto v<int>{42, 17, 3, 9, 37};
for (const auto& e:v){
    std::cout << e <<;
}
std::cout << "\n";
```

Comentario:

En c++ debes usar un const int en lugar de la macro.

No es aplicable auto en esa inicialización.

Solución:

```
const int N{5};

std::array<int, 5> a{42, 17, 3, 9, 37};

for(const auto& c: a)
    std::cout << c << " ";
}
std::cout << "\n";
```

Pregunta 2

Finalizado

Se puntúa 1,00 sobre 2,00

[2 puntos] Dado el siguiente código y asumiendo que **N** ya está definida, re-implementalo en C++ usando las **herramientas vistas en clase**:

```
int* a = 0;
int* b = (int*)malloc(sizeof(int)*N);

for(int i = 0; i < N; i++) {
    b[i] = i;
}

a = b;

for(int i = 0; i < N; i++) {
    printf("%d ", a[i]);
}
printf("\n");

free(a);
```

```
auto a{0};
auto b = new()

for (const auto& i: N)

for (const auto& i: a){
    std::cout << i <<;
}
std::cout << "\n" <<

delete(a);
```

Comentario:

Si usas Shared_ptr no necesitas liberar memoria.

El for each no funciona ni sobre new ni sobre el shared_ptr.

Solución:

Por error, la solución correcta es un poco más difícil de lo esperado (por no haberlo visto en clase) en la creación del shared_ptr así que la corrección de este ejercicio se ignora esta parte.

```
std::shared_ptr<int[]> a;  
std::shared_ptr<int[]> b(new int[10]);  
  
for(int i = 0; i < N; i++) {  
    b[i] = i;  
}  
  
a = b;  
  
for(int i = 0; i < N; i++) {  
    std::cout << a[i] << " ";  
}  
std::cout << "\n";
```

Pregunta 3

Finalizado

Se puntúa 0,50 sobre 1,00

[1 punto] Dadas las siguientes declaraciones de variables, indicar cuales se pueden sustituir por "auto". **Justifica la respuesta.**

```
int i = 3;  
unsigned u;  
float f = 2.4;  
double d = 5.1;  
std::vector<int> v{1, 2, 3};
```

int i = 3 si se puede sustituir por auto.
unsigned u no se puede sustituir por auto.
float f = 2.4 si se puede sustituir por auto.
double d = 5.1 si se puede sustituir por auto.
std::vector<int> v{1, 2, 3}; si se puede sustituir por auto.

Las que si se pueden sustituir por auto es porque son variables que ya están definidas. Por lo que C++ puede sacar de ellas de que tipo son.

Comentario:

Solución:

```
auto i = 3; // porque 3 es int
unsigned u; // auto necesita que se le asigne un valor
float f = 2.4; // no porque 2.4 es double. Podría ser auto f = 2.4f;
auto d = 5.1; // porque 5.1 es double
std::vector<int> v{1, 2, 3}; // {1, 2, 3} es una lista de inicializadores, hay
que decir en que contenedor se van a guardar.
```

Pregunta 4

Finalizado

Se puntúa 0,00 sobre 1,00

[1 punto] ¿Que diferencia hay entre struct y class?¿en qué situaciones se ha de declarar uno u otro?

La diferencia entre struct y class es que por ejemplo, class tiene una parte pública y otra privada, en cambio struct no, es todo público. Otra diferencia es que en class puedes crear métodos, pero en struct no.

Comentario:

Solución:

En una "class" los atributos y métodos son privados por defecto, en un "struct" son públicos por defecto. Si los atributos son una agrupación de variables a las que se les da un nombre y, por lo tanto, se va a acceder a sus campos, se debería usar un struct. En cualquier otro caso se debería usar una clase en la que siempre los atributos son privados.

Pregunta **5**

Finalizado

Se puntúa 1,75 sobre 2,00

[2 puntos] La declaración de la siguiente clase tiene algunos errores (**no en las operaciones**) y además podría ser un clase genérica. Re-implementa la clase corrigiendo los **errores** y haciéndola **genérica**.

```
class Complejo {  
    float r; // real  
    float i; // imaginaria  
  
    void suma(Complejo c) {  
        r = r + c.r;  
        i = i + c.i;  
    }  
  
    void multiplicacion(Complejo c) {  
        r = r * c.r - i * c.i;  
        i = r * c.i + i * c.r;  
    }  
  
    float real() {return r;}  
    float imaginaria() {return i;}  
};
```

```
typename <template T>
public:
class Complejo{
    <T> r; // real
    <T> i; // imaginaria

    void suma(Complejo c){
        r = r + c.r;
        i = i + c.i;
    }

    void multiplicacion(Complejo c){
        r = r * c.r - i * c.i;
        i = r * c.i + i * c.r;
    }

private:
    <T> real() {return r;}
    <T> imaginaria() {return i;}
}
```

Comentario:

La sintaxis no es correcta.

Solución:

```
template <typename T>
class Complejo {
public:
    void suma(Complejo c) {
        r = r + c.r;
        i = i + c.i;
    }

    void multiplicacion(Complejo c) {
        r = r * c.r - i * c.i;
        i = r * c.i + i * c.r;
    }

    T real() {return r;}
    T imaginaria() {return i;}
private:
    T r; // real
    T i; // imaginaria
}
```

Pregunta 6

Parcialmente correcta

Se puntúa 0,57 sobre 1,00

[1 punto] Dadas las siguientes declaraciones de variables con auto, indica el tipo de la variable:

auto a; // tipo	unsigned	✗
auto b{42}; // tipo	int	✓
auto c{}; // tipo	No válido	✓
auto d{4.2f}; // tipo	float	✓
auto e{42u}; // tipo	unsigned	✓
auto f{4.2}; // tipo	float	✗
auto g{42l}; // tipo	int	✗

unsigned	long	char	No válido	float	double	int
----------	------	------	-----------	-------	--------	-----

Respuesta parcialmente correcta.

Ha seleccionado correctamente 4.

La respuesta correcta es:

[1 punto] Dadas las siguientes declaraciones de variables con auto, indica el tipo de la variable:

auto a; // tipo	[No válido]
auto b{42}; // tipo	[int]
auto c{}; // tipo	[No válido]
auto d{4.2f}; // tipo	[float]
auto e{42u}; // tipo	[unsigned]
auto f{4.2}; // tipo	[double]
auto g{42l}; // tipo	[long]

Pregunta 7

Parcialmente correcta

Se puntúa 1,00 sobre 2,00

[2 puntos] Dadas las siguientes estructuras de C++ indicar, para cada operación si tiene **complejidad constante (O(1))** y si la estructura es una implementación **válida para colas**.

Operación	std::array	std::deque
push_front	Si <input type="checkbox"/> No <input checked="" type="checkbox"/>	No <input checked="" type="checkbox"/> Si <input type="checkbox"/>
push_back	Si <input checked="" type="checkbox"/> No <input type="checkbox"/>	Si <input checked="" type="checkbox"/> No <input type="checkbox"/>
insert	Si <input checked="" type="checkbox"/> No <input type="checkbox"/>	No <input checked="" type="checkbox"/> Si <input type="checkbox"/>
front	Si <input checked="" type="checkbox"/> No <input type="checkbox"/>	Si <input checked="" type="checkbox"/> No <input type="checkbox"/>
back	Si <input checked="" type="checkbox"/> No <input type="checkbox"/>	Si <input checked="" type="checkbox"/> No <input type="checkbox"/>
operator[]	No <input checked="" type="checkbox"/> Si <input type="checkbox"/>	No <input checked="" type="checkbox"/> Si <input type="checkbox"/>
data()	No <input checked="" type="checkbox"/> Si <input type="checkbox"/>	No <input checked="" type="checkbox"/> Si <input type="checkbox"/>
Cola Válida	No <input checked="" type="checkbox"/> Si <input type="checkbox"/>	Si <input checked="" type="checkbox"/> No <input type="checkbox"/>

No ☐ Si ☐

Respuesta parcialmente correcta.

Ha seleccionado correctamente 8.

La respuesta correcta es:

[2 puntos] Dadas las siguientes estructuras de C++ indicar, para cada operación si tiene **complejidad constante (O(1))** y si la estructura es una implementación **válida para colas**.

Operación	std::array	std::deque
push_front	[No]	[Si]
push_back	[No]	[Si]
insert	[No]	[Si]
front	[Si]	[Si]
back	[Si]	[Si]
operator[]	[Si]	[Si]
data()	[Si]	[No]
Cola Válida	[No]	[Si]

Actividad previa

Examen EDA Junio 2023

Ir a...

Siguiente actividad

[Parcial 2: Árboles](#)