Facultad de Ingeniería Electrónica y Eléctrica

Universidad Nacional Mayor de San Marcos

Examen Parcial de P.O.O. – Grupo 01 (20/05/2019) - Dur. 90'

Alumno: Maguiña Noa Jesus Daniel 18190309

1) (2.0 puntos) Describir como mínimo 3 paradigmas de programación y dar sus características

- Paradigma estructurado.

Solo una secuencia, decisión y repetición.
El código más fácil de leer, pero aún difícil para sistemas grandes debido a la repetición de código.
Que hacer si es necesario repetir una secuencia de líneas de código en diferentes lugares.

- Paradigma Procedente

Sinónimo: paradigma procedural
Uso de subprogramas.
- Agrupamiento de código que permite la creación de acciones complejas.
- Asignación de un nombre para estas acciones complejas.
- Llamada a estas acciones complejas de cualquier punto de programa.

Estas acciones complejas se denominan procedimientos, subrutinas y funciones.

- Paradigma Orientada a Objetos

Clases de objetos: Agrupación de procedimientos y variables afines.
- Paquetes de clases: *Agrupación de clases afines.
  *Representan bibliotecas de apoyo.

2) (1.5puntos) Cual es la diferencia entre i++ e ++i, dar un ejemplo en código.

Ambos hacen lo mismo salvo que el de postincremento se evalúa al valor de la variable ANTES de que se almacene el valor incrementado y el de preincremento se evalúa al valor de la variable DESPUÉS de que se almacene el valor incrementado.

```
int x = 1;
System.out.println(++x);
System.out.println(x);
```

```
int x = 1;
System.out.println(x++); System.out.println(x);
```

3) (1.5puntos) Completar el cuadro con la información de PRIORIDAD, siendo 1 más prioritario que 5

Operador	Prioridad	Operador	Prioridad
/	4	(expr)	1
--var	3	+expr	3
*	4	+	5
%	4	-	5
Var--	2	&&	5

4) (3.0 puntos) Responder las siguientes preguntas y dar ejemplos:

A) ¿Qué significa un casting en programación?

Consiste transformar el tipo de variable de una expresión:
*Para ello, basta preceder la expresión por "(tipo)".
*Tipo casting tiene una prioridad superior a *,/e%

```
//pasar un real para entero
Float a = 5.1f;
Int x= int(a); // x toma el valor de 5 porque pasa de una variable flotante a variable entera
```

B) ¿Qué es una funcion y un procedimiento?

En programación a los procedimientos y funciones también se les conoce por el nombre de *rutinas*, *subrutinas* o *subprogramas*. Son bloques de instrucciones que realizan tareas específicas. Las rutinas se declaran una sóla vez pero pueden ser utilizadas, mediante *llamadas*, todas las veces que se quiera en un programa. Una rutina es independiente del resto del programa por lo que, en principio, facilita el diseño, el seguimiento y la corrección de un programa. Pueden además almacenarse independientemente en colecciones llamadas *librerías* o *unidades*, lo cual permite que sean utilizadas en cualquier programa. De hecho, existen funciones y procedimientos que vienen ya construidos para el lenguaje de programación TurboPascal (`Cos`, `Sin`, `Exp`, `ReadLn`, `WriteLn`, `ClrScr`...), que están almacenados en distintas unidades (`System`, `Crt`, `Graph`...) y que el programador puede emplear en sus programas. Además, el programador puede construir sus propias unidades con las constantes, tipos de datos, variables, funciones y procedimientos,... que desee incluir.

C) ¿Qué quiere decir sobre carga de operadores

La sobrecarga de operadores es la capacidad para transformar los operadores de un lenguaje como por ejemplo el +, -, etc, cuando se dice transformar se refiere a que los operandos que entran en juego no tienen que ser los que admite el lenguaje por defecto. Mediante esta técnica podemos sumar dos objetos creados por nosotros o un objeto y un entero, en vez de limitarnos a sumar números enteros o reales, por ejemplo.

Los operadores que podemos sobrecargar son los unarios, +, -, !, ~, ++, --; y los binarios +, -, *, /, %, &, |, ^, <<, >>. Es importante decir que los operadores de comparación, ==, !=, <, >, <=, >=, se pueden sobrecargar pero con la condición que siempre se sobrecargue el complementario, es decir, si sobrecargamos el == debemos sobrecargar el !=