UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

ESTRUCTURAS DE DATOS Y ALGORITMOS 1

M. en I. Fco. Javier Rodríguez García

Cuestionario/Repaso

1 Tipos

Explique qué es una instrucción o comando.

Explique qué es una variable.

Explique qué es una expresión.

Explique qué es una constante. ¿Cuál es la ventaja de usar `const` en lugar de `#define` cuando se definen constantes?

Escriba el rango de valores que una variable tipo `char` con/sin signo puede almacenar.

Escriba el rango de valores que una variable tipo `int` de 32 bits con/sin signo puede almacenar.

Escriba el rango de valores que una variable tipo `int` de 64 bits con/sin signo puede almacenar.

¿Cuál es la palabra reservada de C para indicar una variable con signo? Explique porqué es importante marcar si una variable de tipo entero tiene signo o no. Tip: Tiene que ver con la aritmética.

Escriba el rango de valores que una variable tipo `float` puede almacenar.

Escriba el rango de valores que una variable tipo `double` puede almacenar.

Explique la representación de un tipo bool en C.

Explique la representación de cadenas de texto en C. ¿Qué biblioteca se utiliza para procesar dichas cadenas?

Explique la diferencia entre *declaración* y *definición*. Aunque ambas se aplican a variables y funciones, el efecto es el mismo.

Explique qué es una variable local.

Explique qué es una variable global.

Explique porqué ningún buen programador le recomendaría utilizar variables globales en sus programas.

Explique el hecho de declarar/definir variables dentro de un *bloque*. Un bloque en C es cualquier conjunto de instrucciones dentre de llaves, { y }. Tip: Tiene que ver con el tiempo de vida de la variable.

2 Operadores

Expliqué qué es un operador.

Liste TODOS los operadores del lenguaje C, agrupados por función: aritméticos, lógicos, a nivel de bit, etc.

Liste los operadores abreviados.

Explique para qué sirve la asociatividad de los operadores, y qué efecto produce utilizar paréntesis con dicha asociatividad.

Explique la evaluación de corto-circuito.

Explique qué es la conversión de tipos (casting).

3 Selección

Explique la sentencia **if**. Dibuje un ejemplo en un diagrama de flujo.

Explique la sentencia **if-else**. Dibuje un ejemplo en un diagrama de flujo.

Explique la sentencia **if-else if-else**. Dibuje un ejemplo en un diagrama de flujo.

Explique el operador condicional **?:**. Dibuje un ejemplo en un diagrama de flujo.

Explique las sentencias **if-else** anidadas.

Explique el efecto que tiene utilizar la palabra reservada **break** en sentencias **if-else** anidadas. Dibuje un ejemplo en un diagrama de flujo.

Explique la evaluación de corto-circuito en las expresiones lógicas anteriores. Tip: La explicación la podría encontrar como *eager evaluation*.

Explique la sentencia de control **switch**.

Explique la función de: **case**, **break**, **default**.

4 Ciclos

Explique el ciclo **while**. Dibuje un ejemplo en un diagrama de flujo. ¿Cuándo es utilizado este ciclo, generalmente?

Explique el ciclo **do-while**. Dibuje un ejemplo en un diagrama de flujo. ¿Cuándo es utilizado este ciclo, generalmente?

Explique el ciclo **for**. Dibuje un ejemplo en un diagrama de flujo. ¿Cuándo es utilizado este ciclo, generalmente?

Explique el efecto que tiene utilizar la palabra reservada **continue** en cualquiera de los ciclos anteriores. Dibuje un ejemplo en un diagrama de flujo.

Explique el efecto que tiene utilizar la palabra reservada **break** en cualquiera de los ciclos anteriores. Dibuje un ejemplo en un diagrama de flujo.

Explique qué es un ciclo infinito, cómo se implementa en los 3 tipos de ciclos mencionados, y cuándo se utilizan. Tip: Puede buscar acerca del *super-loop*.

5 Funciones

Expliqué qué es una función y porqué las funciones son tan importantes en el lenguaje C.

Explique qué es la abstracción funcional. Tip: Ud. la conoce como *descomposición funcional*.

Describa la estructura general de una función.

Describa la estructura general de una función en forma gráfica.

¿Qué es el cuerpo de una función?

Explique la diferencia entre *declaración* de una función, y su *definición*.

¿En C se pueden tener dos funciones con el mismo nombre? Explique.

De un ejemplo de funciones que no devuelven resultados. ¿Qué hace la palabra reservada **void**?

De un ejemplo de funciones que devuelven resultados. ¿Qué hace la palabra reservada **return**?

Explique los *tipos* de resultados que una función puede devolver. Es decir, ¿qué hay que hacer para avisarle al compilador que la función va a devolver un valor de tipo entero, o de tipo real de doble precisión, etc.?

¿Cuántos resultados puede devolver una función? Explique.

¿Una función marcada como **void** puede utilizar sentencias **return**? Explique.

Explique los argumentos de una función, y porqué son tan importantes.

Una función debe hacer una sóla cosa, y hacerla bien, ¿porqué?

Una misma función no debe mezclar dos o más intenciones diferentes, ¿porqué?

Explique la forma de llamar a una función que no devuelve resultados.

Explique la forma de llamar a una función que devuelve un resultado.

Explique el paso de parámetros por *copia*. Tip: Normalmente también encontrará la explicación del paso de parámetros por *referencia*.

6 Bibliografía

[JOYANES04]

Joyanes A., Luis; Zahonero M., Ignacio. **Algoritmos y estructuras de datos, una perspectiva en C**. SPAIN: McGraw-Hill, 2004.

[JOYANES05]

Joyanes A., Luis; Zahonero M., Ignacio. **Programación en C, Metodología, algoritmos y estructura de datos**. 2da. Ed. SPAIN: McGraw-Hill, 2005.

[FISCHER01]

Fischer, Alice E.; Eggert, David W.; Ross, Stephen M. **Applied C: An introduction and more**. USA: McGraw-Hill, 2001.

[SAVITCH07]

Savitch, Walter. **Problem solving with C++**. 6th ed. USA: Pearson Education, 2007.

[DALE10]

Dale, Nell; Weems, Chip. **Programming and problem solving with C++**. 5th ed. (Comprehensive edition.). USA: Jones and Bartlett Publishers, 2010.