Unidad 1: Tarea 1

Agustín Alejandro Mota Hinojosa

Universidad Politécnica de Cd. Victoria

Introducción a la Programación

Dr. Mario Alberto Gómez Rodríguez

15/1/2021



# 1 Capítulo 1°

## 1.1 Preguntas de revisión

### 1.1.1 Opción múltiple

1. Un **programa** es un set de instrucciones que la computadora sigue para realizar una tarea.
2. Los dispositivos físicos que conforman a una computadora se les llaman: **Hardware**.
3. La parte de la computadora que corre los programas se le llama: **CPU**.
4. Hoy en día, los CPUs son chips pequeños que se les conoce como: **Microprocesadores**.
5. La computadora almacena un programa mientras que el programa está ejecutándose, así como los datos con el que esta trabajando el programa en: **Memoria secundaria**.
6. Este es un tipo de memoria volátil que se usa solo para almacenamiento temporal mientras el programa se está ejecutando: **RAM**.
7. Un tipo de memoria que puede almacenar datos durante largos períodos de tiempo, incluso cuando hay no hay energía en la computadora se llama: **Memoria principal**.
8. Un componente que recopila datos de personas u otros dispositivos y los envía a la computadora se llama: **Dispositivo de entrada**.
9. Un video se muestra en un: **Dispositivo de salida**.
10. Un **bit** es suficiente memoria para almacenar una letra del alfabeto o un número pequeño.
11. Un byte está hecho de 8: **bits**
12. En un sistema numérico **binario**, todos los valores numéricos son escritos en secuencias de 0s y 1s.
13. Un bit que está apagado representa el siguiente valor: **0**.
14. Un conjunto de 128 códigos numéricos que representan las letras en inglés, varios signos de puntuación, marcas y otros caracteres es: **ASCII**.
15. Un esquema de codificación extenso que puede representar los caracteres de muchos de los idiomas en el mundo es: **Unicode**.
16. Los números negativos se codifican mediante la técnica: **complemento a dos**.
17. Los números reales se codifican mediante la técnica: **de números flotantes**.
18. Los pequeños puntos de color que componen las imágenes digitales se llaman: **Pixles**.
19. Si mirara un programa en lenguaje de máquina, vería: **una cadena de números binarios**.
20. En la parte **ejecución** del ciclo de *fetch-decode-execute*, la CPU determina que operación debe realizar
21. Las computadoras solo pueden ejecutar programas escritos en **Lenguaje máquina**.
22. El **compilador** traduce un programa en lenguaje ensamblador a un lenguaje de máquina.
23. Las palabras que componen un lenguaje de programación de alto nivel se denominan **Comandos**.
24. Las reglas que se deben seguir al escribir un programa se llaman **sintaxis**.
25. Un programa **traductor** traduce un programa de lenguaje de alto nivel en un programa de lenguaje de máquina.

### 1.1.2 Verdadero o Falso

Aclaración: - [ ] Falso - [X] Verdadero

* ☐ 1. Hoy en día, las CPU son dispositivos enormes hechos de componentes eléctricos y mecánicos como tubos de vacío e interruptores.
* ☒ 2. La memoria principal también se conoce como RAM.
* ☒ 3. Cualquier dato que se almacene en la memoria de una computadora debe almacenarse como número binario.
* ☐ 4. Las imágenes, como las que crea con su cámara digital, no se pueden almacenar como números binarios.
* ☒ 5. El lenguaje de máquina es el único lenguaje que entiende una CPU.
* ☐ 6. El lenguaje ensamblador se considera un lenguaje de alto nivel.
* ☒ 7. Un intérprete es un programa que traduce y ejecuta las instrucciones en un programa de idiomas de alto nivel.
* ☐ 8. Un error de sintaxis no impide que se compile y ejecute un programa.
* ☐ 9. Windows Vista, Linux, UNIX® y Mac OS X son ejemplos de aplicaciones software.
* ☐ 10. Programas de procesamiento de texto, programas de hojas de cálculo, programas de correo electrónico, navegadores web y los juegos son todos ejemplos de programas de utilidad.

# 2 Capítulo 2°

## 2.1 Preguntas de revisión

### 2.1.1 Opción múltiple

1. Un error **lógico** no impide que el programa se ejecute, pero hace que producir resultados incorrectos.
2. Un(a) **tarea** es una función única que el programa debe realizar para poder satisfacer al cliente
3. Un(a) **algoritmo** es un conjunto de pasos lógicos bien definidos que se deben seguir para realizar una tarea.
4. Un lenguaje informal que no tiene reglas de sintaxis y no está destinado a ser compilado o ejecutado se llama **pseudocódigo**.
5. Un **diagrama de flujo** es un diagrama que describe gráficamente los pasos que tienen lugar en un programa.
6. Un(a) **estructura de secuencia** es un conjunto de declaraciones que se ejecutan en el orden en que aparecen
7. Un(a) **cadena** es una secuencia de caracteres que se usa como datos.
8. Un(a) **variable** es una ubicación de almacenamiento en la memoria que está representada por un nombre.
9. Un **usuario** es cualquier persona hipotética que utiliza un programa y proporciona entrada para ello
10. Un(a) **declaración de entrada** es un mensaje que le dice (o pide) al usuario que ingrese un valor específico.
11. Un(a) **declaración de inicialización** establece una variable en un valor especificado.
12. En la expresión 12 + 7, los valores a la derecha e izquierda del símbolo + se llaman: **argumentos**
13. Un operador **exponente** eleva un número a una potencia.
14. Un operador **módulo** realiza la división, pero en lugar de devolver el cociente devuelve el resto.
15. Un(a) **declaración de variable** especifica el nombre y el tipo de datos de una variable.
16. La asignación de un valor a una variable en una declaración de declaración se llama **inicialización**.
17. Una variable **indefinida** es aquella que se ha declarado, pero no se ha inicializado o asignado un valor.
18. Un(a) **Constante** es una variable cuyo contenido tiene un valor que es de solo lectura y no se puede cambiar durante la ejecución del programa.
19. Un proceso de depuración en el que imaginas que tu eres la computadora ejecutando el programa se llama **Hand tracing**.
20. Notas breves colocadas en diferentes partes de un programa, explicando cómo esas partes de el programa de trabajo, se llaman: **Comentarios**.

### 2.1.2 Verdadero o Falso

Aclaración: - [ ] Falso - [X] Verdadero

* ☐ 1. Los programadores deben tener cuidado de no cometer errores de sintaxis al escribir pseudocódigo programas.
* ☒ 2. En una expresión matemática, la multiplicación y la división tienen lugar antes que la suma y sustracción.
* ☐ 3. Los nombres variables pueden tener espacios en ellas.
* ☒ 4. En la mayoría de los lenguajes, el primer carácter de un nombre de variable no puede ser un número.
* ☐ 5. El nombre Gross\_pay está escrito en la convención camelCase.
* ☒ 6. En lenguajes que requieren declaraciones de variables, la declaración de una variable debe aparecen antes de cualquier otra declaración que use la variable.
* ☒ 7. Las variables no inicializadas son una causa común de errores.
* ☒ 8. El valor de una constante nombrada no se puede cambiar durante la ejecución del programa.
* ☒ 9. El *Hand Tracing* es el proceso de traducir un programa de pseudocódigo a una máquina idioma a mano.
* ☐ 10. La documentación interna se refiere a libros y manuales que documentan un programa, y están diseñados para su uso dentro del departamento de programación de una empresa.

### 2.1.3 Respuesta corta

1. ¿Qué suele hacer un programador profesional primero para comprender de un problema?

**Entender y estudiar el problema, después diseñarlo.**

1. ¿Qué es el pseudocódigo?

**Un lenguaje informal**

1. ¿Cuáles son los tres pasos suelen realizar los programas informáticos?

**Fetch, decode, execute**

1. ¿Qué significa el término *user-friendly*?

**Que es fácil de usar para el usuario**

1. ¿Qué son dos cosas se deben especificar normalmente en una declaración de variable?

**Tipo y el nombre**

1. ¿Qué valor se almacena en las variables no inicializadas?

**Normalmente 0**

### 2.1.4 Algorithm Workbench

1. Diseñe un algoritmo que solicite al usuario que ingrese su altura y almacene la la entrada del usuario en una variable llamada altura.

* Estructura del algoritmo:
  + Inicio
    - Declaración de la variable *altura*
    - Solicitar la altura del usuario
    - Almacenar el input del usuario en la variable *altura*
  + Fin

1. Diseñe un algoritmo que solicite al usuario que ingrese su color favorito y almacena la entrada del usuario en una variable denominada color.

* Estructura del algoritmo:
  + Inicio
    - Declaración de la variable *color*
    - Solicitar el color del usuario
    - Almacenar el input del usuario en la variable *color*
  + Fin

1. Escriba declaraciones de asignación que realicen las siguientes operaciones con el variables: *a, b, c*.

* Agrega 2 a *a* y almacena el resultado en *b*.
* Multiplica *b* por 4 y almacena el resultado en un *c*.
* Divide *a* por y almacena el resultado en *b*.
* Resta 8 de *b* y almacena el resultado en *a*.
  + Definir a = 0
  + Definir b = 0
  + Definir c = 0
  + b = a + 2
  + c = b \* 4
  + b = a / 3.14
  + a = b - 8

1. Suponga que el resultado de las variables, *w, x, y, z* son números enteros, y que , , , . ¿Qué valor se almacenará en el resultado en cada uno de los siguientes declaraciones:

* Establecer resultado segundo.

**Resultado: 12**

* Establecer resultado: .

**Resultado: 4**

* Establecer resultado: .

**Resultado: 2**

* Establecer resultado: .

**Resultado: 6**

1. Escriba una declaración de pseudocódigo que declare el costo en una variable que almacene números reales.

**Set** cost **as Integer**

1. Escriba una declaración de pseudocódigo que declare la variable *total* para que pueda contener enteros. Inicializa la variable con el valor 0.

**Set** total = 0

1. Escriba una declaración de pseudocódigo que asigne el valor 27 a la variable *count*.

**Set** count = 27

1. Escriba una declaración de pseudocódigo que asigne la suma de 10 y 14 a la variable *total*.

**Set** total = 10 + 14

1. Escriba una declaración de pseudocódigo que resta la variable *downPayment* de la variable *total* y asigna el resultado a la variable *due*.

**Set** downPayment, total, due **as Real** due = total - downPayment

1. Escriba una declaración en pseudocódigo que multiplique la variable *subtotal* por y que asigne el resultado a la variable *totalfee*.

**Set** subtotal, totalfee **as Real**

totalfee = subtotal \* 0.15

1. Si el siguiente pseudocódigo fuera un programa real, ¿qué mostraría?

* Declarar entero a = 5
* Declarar entero b = 2
* Declarar entero c = 3
* Declarar resultado entero
* Establecer resultado = a + b \* c
* Mostrar resultado

**Resultado: 11**

1. Si el siguiente pseudocódigo fuera un programa real, ¿qué mostraría?

* Declarar número entero = 99
* Establecer num = 5
* Mostrar num

**Resultado: 5**

## 2.2 Ejercicios de programación

1. Predicción de ventas

Una empresa ha determinado que su beneficio anual normalmente es el del total de ventas. Diseñe un programa que le pida al usuario que ingrese la cantidad proyectada del total de ventas y luego muestra el beneficio que se obtendrá de esa cantidad. *Pista: use el valor 0.23 para representar el 23 por ciento.*

* Inicio
  1. **Set** percent, amount, benefit **as Integer**
  2. percent = 0.23
  3. **Display** *“Insert the amount of sales”*
  4. benefit = percent \* amount
  5. **Output** benefit
* Fin

1. Cálculo de tierras

Un acre de tierra equivale a 43,560 pies cuadrados. Diseñe un programa que pregunte el usuario debe ingresar el total de pies cuadrados en un terreno y calcular el número de acres del tramo.

* Inicio
  1. **Set** sqFeet, acre **as Integer**
  2. **Display** *“Insert the amount of Square Feet:”*
  3. **Input** sqFeet
  4. acre = sqFeet \* 0.000022957
  5. **Output** acre
* Fin

1. Compra total

Un cliente en una tienda está comprando cinco artículos. Diseñe un programa que solicite la precio de cada artículo, y luego muestra el subtotal de la venta, impuesto y el total. Suponga que el impuesto sobre las ventas es del 6 por ciento.

* Inicio
  1. **Set** subTotal, tax, total **as Integer**
* // Aquí defino una lista para juntar todas las variables y no tener que definirlas por separado
  1. **Set** articles[5]
  2. **Display** “Insert the price”
  3. **Input** “articles[0]”
  4. **Display** “Insert the price”
  5. **Input** “articles[1]”
  6. **Display** “Insert the price”
  7. **Input** “articles[2]”
  8. **Display** “Insert the price”
  9. **Input** “articles[3]”
  10. **Display** “Insert the price”
  11. **Input** “articles[5]”
  12. tax = (articles[0]+articles[1]+articles[2]+articles[3]+articles[4]) \* 0.06
  13. subTotal = articles[0]+articles[1]+articles[2]+articles[3]+articles[4]
  14. total = subtotal + tax
  15. **Display** “Total:”, **Output** total
  16. **Display** “Sub total;”, **Output** subTotal
  17. **Display** “Taxes:”, **Output** tax
* Fin

1. Distancia viajada

Asumiendo que no hay accidentes o retrasos, la distacia que un carro recorre en la interestatal puede ser calculada con la siguiente fórmula:

Un carro está viajando a 60 millas por hora. Diseña un programa que muestre lo siguiente:

* La distancia que el carro va viajar en 5 horas
* La distancia que el carro va a viajar en 8 horas
* La distancia que el carro va a viajar en 12 horas
* Inicio
  1. **Set** speed **as Integer**
  2. speed = 60
  3. **Set** time[3]
  4. **Set** distance[3]
  5. time[0] = 5; time[1] = 8; time[2] = 12; 6 distance[0] = speed \* time[0]
  6. distance[1] = speed \* time[1]
  7. distance[2] = speed \* time[2]
  8. **Display** “Distance of 5h going at 60mph:”, **Output** distance[0]
  9. **Display** “Distance of 8h going at 60mph:”, **Output** distance[1]
  10. **Display** “Distance of 12h going at 60mph:”, **Output** distance[2]
* Fin