**Nombre del alumno:** José Carlos Toledo Pérez

**Nombre del tema:** Unidad 1 y 2

**Parcial:** 1

**Nombre de la materia**: Fundamentos y lógica de programación

**Nombre del profesor:** Juan José Ojeda Trujillo

**Nombre de la Licenciatura:** Ingeniería en Sistemas Computacionales

**Cuatrimestre:** 3

# Unidad III. Estructura general de un programa

3.1 Concepto de programa

Un programa de computadora es un conjunto de instrucciones órdenes dadas a la máquina que producirán la ejecución de una determinada tarea. En esencia, un programa es un medio para conseguir un fin.

3.2 Partes constitutivas de un programa

Tras la decisión de desarrollar un programa, el programador debe establecer el conjunto de especificaciones que debe contener el programa: entrada, salida y algoritmos de resolución, que incluirán las técnicas para obtener las salidas a partir de las entradas.

3.3 Instrucciones y tipos de instrucciones

Las instrucciones disponibles en un lenguaje de programación dependen del tipo de lenguaje. 1. instrucciones de inicio/fin, 2. instrucciones de asignación, 3. instrucciones de lectura, 4. instrucciones de escritura, 5. instrucciones de bifurcación.

3.4 Elementos básicos de un programa Los elementos básicos constitutivos de un programa o algoritmo son:

• palabras reservadas • identificadores • caracteres especiales • constantes • variables • expresiones • instrucciones

3.5 Datos, tipos de datos y operaciones primitivas

Un dato es la expresión general que describe los objetos con los cuales opera una computadora. Existen dos tipos de datos: básicos, incorporados o integrados (estándar) que se incluyen en los lenguajes de programación; definidos por el programador o por el usuario.

3.5.1 Datos numéricos • tipo numérico entero (integer). • tipo numérico real (real). Enteros: Los enteros son números completos, no tienen componentes fraccionarios o decimales y pueden ser negativos o positivos. Reales: Los números reales siempre tienen un punto decimal y pueden ser positivos o negativos.

3.5.2 Datos lógicos (booleanos) Este tipo de datos se utiliza para representar las alternativas (sí/no) a determinadas condiciones. Por ejemplo, cuando se pide si un valor entero es par, la respuesta será verdadera o falsa, según sea par o impar.

3.5.3 Datos tipo carácter y tipo cadena

El tipo carácter es el conjunto finito y ordenado de caracteres que la computadora reconoce. Un dato tipo carácter contiene un solo carácter. • caracteres alfabéticos (A, B, C, ..., Z) (a, b, c, ..., z), • caracteres numéricos (1, 2, ..., 9, 0), • caracteres especiales (+, -, *, /, ^, ., ;, , $, ...)

3.6 Constantes y variables

Una constante es un dato que permanece sin cambios durante todo el desarrollo del algoritmo o durante la ejecución del programa. Una variable es un objeto o tipo de datos cuyo valor puede cambiar durante el desarrollo del algoritmo o ejecución del programa.

3.7 Expresiones Las expresiones son combinaciones de constantes, variables, símbolos de operación, paréntesis y nombres de funciones especiales. Las mismas ideas son utilizadas en notación matemática tradicional; por ejemplo, a + (b + 3) + √c

3.7.1 Expresiones aritmética

Las expresiones aritméticas son análogas a las fórmulas matemáticas. Las variables y constantes son numéricas (real o entera) y las operaciones son las aritméticas. + suma - resta * multiplicación / división ↑, **, ^ exponenciación div, / división entera mod, % módulo (resto)

3.7.2 Reglas de prioridad

En los lenguajes que soportan la operación de exponenciación, este operador tiene la mayor prioridad. En caso de coincidir varios operadores de igual prioridad en una expresión o su expresión encerrada entre paréntesis, el orden de prioridad en este caso es de izquierda a derecha, y a esta propiedad se denomina asociatividad

3.7.3 Expresiones lógicas (booleanas)

Un segundo tipo de expresiones es la expresión lógica o booleana, cuyo valor es siempre verdadero o falso. Recuerde que existen dos constantes lógicas, verdadera (true) y falsa (false) y que las variables lógicas pueden tomar sólo estos dos valores.

3.8 La operación de asignación

La operación de asignación es el modo de almacenar valores a una variable. La operación de asignación se conoce como instrucción o sentencia de asignación cuando se refiere a un lenguaje de programación. ← expresión es igual a expresión, variable o constante.

3.9 Entrada y salida de información

Los cálculos que realizan las computadoras requieren para ser útiles la entrada de los datos necesarios para ejecutar las operaciones que posteriormente se convertirán en resultados, es decir, salida. Las operaciones de entrada permiten leer determinados valores y asignarlos a determinadas variables.

3.10 Escritura de algoritmos/programas

Los algoritmos deben ser escritos en lenguajes similares a los programas. Un algoritmo constará de dos componentes: una cabecera de programa y un bloque algoritmo. La cabecera de programa es una acción simple que comienza con la palabra algoritmo. Esta palabra estará seguida por el nombre asignado al programa completo

3.11 Cabecera del programa

Todos los algoritmos y programas deben comenzar con una cabecera en la que se exprese el identificador o nombre correspondiente con la palabra reservada que señale el lenguaje. En los lenguajes de programación, la palabra reservada suele ser programa. En Algorítmica se denomina algoritmo. algoritmo DEMO1

3.12 Declaración de variables

En esta sección se declaran o describen todas las variables utilizadas en el algoritmo, listándose sus nombres y especificando sus tipos. Esta sección comienza con la palabra reservada var (abreviatura de variable) y tiene el formato Var tipo-1: lista de variables1. Tipo-2: lista de variables-2. Tipo-n : lista de variables-n

3.13 Declaración de constantes numéricas

En esta sección se declaran todas las constantes que tengan nombre. Su formato es Const pi = 3.141592 Tamaño = 43 horas = 6.50 Los valores de estas constantes ya no pueden variar en el transcurso del algoritmo

3.14 Declaración de constantes y variables carácter

Las constantes de carácter simple y cadenas de caracteres pueden ser declaradas en la sección del programa const, al igual que las constantes numéricas. Las variables de caracteres se declaran de dos modos: 1. Almacenar un solo carácter. 2. Almacenar múltiples caracteres (cadenas)

3.15 Comentarios Visual Basic 6 / VB .NET

1. Los comentarios utilizan un apóstrofe simple y el compilador ignora todo lo que viene después de ese carácter 2. También se admite por guardar compatibilidad con versiones antiguas de BASIC y Visual Basic la palabra reservada Rem

# Unidad IV. Estructuras de control de un programa

4.1 Estructura secuencial Una estructura secuencial es aquella en la que una acción (instrucción) sigue a otra en secuencia. Las tareasse suceden de tal modo que la salida de una es la entrada de la siguiente y así sucesivamente hasta el final del proceso

4.2 Estructuras selectivas

Las estructuras selectivas se utilizan para tomar decisiones lógicas; de ahí que se suelan denominar también estructuras de decisión o alternativas.

4.3 Alternativa simple (sientonces/if-then)

La estructura alternativa simple si-entonces (en inglés ifthen) ejecuta una determinada acción cuando se cumple una determinada condición. La selección si-entonces evalúa la condición y • si la condición es verdadera, entonces ejecuta la acción S1 • si la condición es falsa, entonces no hacer nada.

4.4 Alternativa doble (sientonces-sino/if-then-else).

La estructura anterior es muy limitada y normalmente se necesitará una estructura que permita elegir entre dos opciones o alternativas posibles, en función del cumplimiento o no de una determinada condición. Si la condición C es verdadera, se ejecuta la acción S1 y, si es falsa, se ejecuta la acción S2

4.5 Alternativa múltiple (según_sea, caso de/case)

Con frecuencia —en la práctica— es necesario que existan más de dos elecciones posibles (por ejemplo, en la resolución de la ecuación de segundo grado existen tres posibles alternativas o caminos a seguir,según que el discriminante sea negativo, nulo o positivo). La estructura de decisión múltiple evaluará una expresión que podrá tomar n valores distintos, 1, 2, 3, 4, …

4.6 Estructuras de decisión anidadas (en escalera) Es posible también utilizar la instrucción si para diseñar estructuras de selección que contengan más de dos alternativas. Por ejemplo, una estructura si-entonces puede contener otra estructura si-entonces, y esta estructura sientonces puede contener otra, y así sucesivamente cualquier número de veces; a su vez, dentro de cada estructura pueden existir diferentes acciones.

4.7 Estructura mientras (“while”) Cuando se ejecuta la instrucción mientras, la primera cosa que sucede es que se evalúa la condición (una expresión booleana). Si se evalúa falsa, no se toma ninguna acción y el programa prosigue en la siguiente instrucción del bucle. Si la expresión booleana es verdadera, entonces se ejecuta el cuerpo del bucle, después de lo cual se evalúa de nuevo la expresión booleana.

4.8 Estructura hacermientras("do-while") El bucle mientras al igual que el bucle desde que se verá con posterioridad evalúan la expresión al comienzo del bucle de repetición; siempre se utilizan para crear bucle pretest. Los bucles pre-test se denominan también bucles controlados por la entrada.

4.9 Estructura repetir ("repeat") En la estructura mientras si el valor de la expresión booleana es inicialmente falso, el cuerpo del bucle no se ejecutará; por ello,se necesitan otros tipos de estructuras repetitivas. La estructura repetir (repeat) se ejecuta hasta que se cumpla una condición determinada que se comprueba al final del bucle El bucle repetir-hasta_que se repite mientras el valor de la expresión booleana de la condición sea falsa.

4.11 Sentencias de salto interrumpir (break) y continuar (continue) La sentencia interrumpir se puede utilizar para terminar una sentencia de iteración y cuando se ejecuta produce que el flujo de control salte fuera a la siguiente sentencia inmediatamente a continuación de la sentencia de iteración

4.10 Estructura desde/para ("for") En muchas ocasiones se conoce de antemano el número de veces que se desean ejecutar las acciones de un bucle. En estos casos, en el que el número de iteraciones es fijo, se debe usar la estructura desde o para (for, en inglés). La estructura desde ejecuta las acciones del cuerpo del bucle un número especificado de veces y de modo automático controla el número de iteraciones o pasos a través del cuerpo del bucle.

# FUENTES DEW INFORMACION

https://plataformaeducativauds.com.mx/assets/docs/libro/ISC/3651f6209f73201727db65c8fc768c9c-LC-ISC304%20FUNDAMENTOS%20Y%20L%C3%93GICA%20DE%20PROGRAMACI%C3%93N.pdf