TEMA Nº 2. CONCEPTOS BÁSICOS, LIBRERÍA, VARIABLES, ENUNCIADOS, PROGRAMAS SIMPLES

CONCEPTOS BÁSICOS. ALGORITMOS

Un algoritmo es un conjunto definido, finito y ordenado de operaciones que se realizan para dar solución a un problema determinado. Cabe destacar que el mismo posee un estado inicial o entrada, pasos sucesivos y bien definidos denominados cuerpo y un estado final en el que se obtiene la solución.

Existen dos tipos de algoritmos:

Los naturales o sencillos que son todas las situaciones que enfrentamos cotidianamente, ejemplo: hacer una lista del mercado de la semana o el procedimiento que se hace para dirigirse del hogar al trabajo.

Los algoritmos que requieren una base lógica para ser resueltos, debido a que forman parte de programas de computadoras y arrojan resultados.

Las etapas de desarrollo de un algoritmo, con base en la lógica, son las siguientes:

1. Definición. En esta etapa se especifica el propósito del algoritmo y se ofrece una definición clara del problema por resolver. Además, aquí también se establece lo que se pretende lograr con su solución.

2. Análisis. En este punto se analiza el problema y sus características, y se determinan las entradas y salidas del problema. De igual modo, también se realiza una investigación sobre si ya se conoce alguna o varias soluciones de este. En el caso de que ya se conozcan varias soluciones, entonces se determina cuál es la más conveniente para el problema que estamos tratando. Si no se conoce ninguna, o no nos satisfacen las soluciones existentes, se propone una nueva.

3. Diseño. Aquí es donde se plasma la solución del problema. Con ese fi n, se emplea una herramienta de diseño, que consiste en el diagrama de flujo y el pseudocódigo.

4. Implementación. En este último paso es donde se realiza o se ve concretado el programa y, por ende, se hacen varias pruebas.

En cada una de las etapas especificadas antes, se utiliza un tipo de descripción conveniente e inteligible para cada uno de los participantes en el proceso de concepción y realización del algoritmo.

PROGRAMA

Se define Programa, a un conjunto de instrucciones que, una vez ejecutado, realiza una o varias tareas en una computadora. Está definido por su sintaxis, que establece e indica las reglas de escritura (la gramática), y por la semántica de los tipos de datos, instrucciones y definiciones que lo constituyen.

PARADIGMAS DEL PROGRAMADOR

La visión y los métodos de un programador en la construcción de un programa o subprograma. Existen diferentes paradigmas que derivan en múltiples y variados estilos de programación y en diferentes formas de solución de problemas:

1. Paradigma Imperativo: En este paradigma se impone que cualquier programa es una secuencia de instrucciones o comandos que se ejecutan siguiendo un orden de arriba hacia abajo; este único enlace del programa se interrumpe exclusivamente para ejecutar otros subprogramas o funciones, después de lo cual se regresa al punto de interrupción.

2. Paradigma Estructurado: Este paradigma es un caso particular de paradigma imperativo, por lo que se imponen únicamente algunas estructuras de código, prohibiendo una continuación del cálculo de manera caótica. Por ejemplo, se impone que las instrucciones sean agrupadas en bloques (procedimientos y funciones) que comunican; por tanto, el código que se repite tiene la forma de un ciclo (*loop*, en inglés), gobernado por una condición lógica.

3. Paradigma Declarativo: Un programa describe el problema a solucionar y la manera de resolverlo, pero no indica el orden de las acciones u operaciones que se deben seguir. En este caso, hay dos paradigmas principales:
   - 3.1. Paradigma Funcional: Conforme a este, todo se describe como una función.
   - 3.2. Paradigma Lógico: De acuerdo con este, todo se describe como un predicado lógico.

VARIABLES. TIPOS Y EXPRESIONES

Una variable es la opción que elige el programador para representar los datos dentro de un programa.

El nombre de una variable debe ser único y no ambiguo. La unicidad del nombre de la variable durante su ciclo de vida, asegura una semántica correcta de las operaciones (expresiones, órdenes o proposiciones) que implican a la variable.

De esta forma, el nombre de una variable es un identificador diferente de cualquier palabra clave utilizada en el lenguaje o nombre de una función externa. Generalmente, los nombres de las variables inician con una letra y son sucesiones de letras y cifras y el símbolo _ (guión bajo). Para la cualidad del programa, es preferible que el nombre de una variable sea sugestivo al tratamiento y de un largo de tamaño aceptable, ya que un nombre de variable muy largo puede generar errores de tecleo al momento de la edición del programa, lo que produce pérdidas de tiempo para su corrección.

En la determinación del nombre de la variable, también se sugiere utilizar únicamente letras sin acento, para una mejor portabilidad del código o porque la sintaxis del lenguaje no lo permite.

Algunos ejemplos de nombres de variables son los siguientes: a, a1, area, suma. También lo son: a1b159 y a2b158; sin embargo, la lectura de un programa con nombres de este tipo sería difícil. Por lo que respecta a la extensión del nombre, una variable llamada nueva_suma_valores_cantidades, tomaría mucho más tiempo escribirla.

Se considera que variables de nombre i, j, k, indican variables enteras usadas para índices; en tanto, las variables de nombre a, b y c, por lo general se utilizan para valores numéricos reales (punto flotante)

Las variables llamadas p y q se emplean para apuntadores; las variables llamadas n y m son variables que contienen valores de tamaños de arreglos.

En C existen cinco tipos de datos según puede verse en la tabla siguiente:

Tipo de dato	Descripción.
`char`	Carácter o entero pequeño (byte)
`int`	Entero
`float`	Punto flotante
`double`	Punto flotante (mayor rango que *float*)
`void`	Sin tipo (uso especial)

Algunos ejemplos de variables de C serían:
float a;
int b,c;
char caracter,otro_caracter;

OPERADORES ARITMÉTICOS

Los operadores aritméticos existentes en C son, ordenados de mayor a menor precedencia:

Operador		Operador		Operador	
++	Incremento	--	Decremento		
-	Menos unario				
*	Multiplicación.	/	División	%	Módulo
+	Suma	-	Resta		

Los operadores ++, -- y % solo pueden usarse con datos de tipo int o char. El operador incremento (++), incrementa en una unidad el valor de la variable sobre la que se aplica, el operador decremento (--), decrementa en una unidad el valor de la variable, y el operador módulo (%), calcula el resto de una división de dos variables de tipo entero o carácter.

Un aspecto que conviene explicar es el hecho de que los operadores incremento y decremento pueden preceder o posceder a su operando, lo cual permite escribir, si x es una variable de tipo int, las expresiones ++x o x++. Usado de forma aislada no presenta ninguna diferencia, sin embargo, cuando se usa en una expresión existe una diferencia en el orden de ejecución del mismo. Cuando el operador incremento (o decremento) precede al operando, C primero realiza el incremento (o decremento), y después usa el valor del operando, realizándose la operación al contrario si el operador poscede al operando.

OPERADORES RELACIÓNALES Y LÓGICOS.

Los operadores relaciónales y lógicos de C, ordenados de mayor a menor prioridad son:

Operador		Operador		Operador		Operador	
!	Not						
>	Mayor que	>=	Mayor o igual que	<	Menor que	<=	Menor o igual que
==	Igual	!=	No igual				
&&	And						
||	Or						

FUNCION main()

La función main() indica donde empieza el programa, cuyo cuerpo principal es un conjunto de sentencias delimitadas por dos llaves, una inmediatamente después de la declaración main() " { ", y otra que finaliza el listado " } ". Todos los programas C arrancan del mismo punto: la primer sentencia dentro de dicha función, en este caso printf (".....").

ENCABEZAMIENTO

Las líneas anteriores a la función main() se denominan ENCABEZAMIENTO (HEADER) y son informaciones que se le suministran al Compilador. La primera línea del programa está compuesta por una directiva: " #include " que implica la orden de leer un archivo de texto especificado en el nombre que sigue a la misma (<stdio.h>) y reemplazar esta línea por el contenido de dicho archivo.

En este archivo están incluidas declaraciones de las funciones luego llamadas por el programa (por ejemplo printf()) necesarias para que el compilador las procese. Hay dos formas distintas de invocar al archivo, a saber, si el archivo invocado está delimitado por comillas (por ejemplo "stdio.h") el compilador lo buscará en el directorio activo en el momento de compilar y si en cambio se lo delimita con los signos <.....> lo buscará en algún otro directorio, cuyo nombre habitualmente se le suministra en el momento de la instalación del compilador en el disco (por ejemplo C:\TC\INCLUDE). Por lo general estos archivos son guardados en un directorio llamado INCLUDE y el nombre de los mismos está terminado con la extensión .h. La razón de la existencia de estos archivos es la de evitar la repetición de la escritura de largas definiciones en cada programa. Nótese que la directiva "#include" no es una sentencia de programa sino una orden de que se copie literalmente un archivo de texto en el lugar en que ella está ubicada, por lo que no es necesario terminarla con "; ".

EJEMPLOS DE PROGRAMAS BÁSICOS EN LENGUAJE C

a)

```
#include <stdio.h>
main()
{
int multiplicador; /* defino multiplicador como un entero */
int multiplicando; /* defino multiplicando como un entero */
int resultado; /* defino resultado como un entero */
multiplicador = 1000 ; /* les asigno valores */
multiplicando = 2 ;
resultado = multiplicando * multiplicador ;
printf("Resultado = %d\n", resultado); /* muestro el resultado */
return 0;
}
```

b)

```
#include <stdio.h>
main()
{
int multiplicador=1000 , multiplicando=2 ;
printf("Resultado = %d\n", multiplicando * multiplicador);
return 0;
}
```

c)

```
#include <stdio.h>
int main(void)
{
puts("Hola, Mundo");
return 0;
}
```

d)

```
#include <stdio.h>
main()
{
printf("hola, Mundo\n");
}
```

e) Cada nombre de variable debe ir precedido por un ampersand (&). Los datos que se introducen deben corresponderse en tipo y orden con los argumentos de la función scanf.

Ejemplo:
scanf("%d", &a);
Lee un número entero introducido por el teclado y guarda el valor leído, en la variable a.

Ejemplo:
scanf("%f", &a);
Lee un número con decimales introducido por el teclado y guarda el valor leído, en la variable a.

Ejemplo:
scanf("%c", &a);
Lee un carácter introducido por el teclado y guarda el valor leído, en la variable a.

Esta sentencia, es equivalente a: a=getchar();

Ejemplo:
int i;
float j;
scanf("%d %f",&i,&j);
Lee un entero y un float introducidos por este orden por el teclado.

f)
Se pueden escribir datos en el dispositivo de salida estándar utilizando la función de biblioteca printf. Es análoga a la función scanf, con la diferencia que su propósito es visualizar datos en vez de introducirlos.

En general la función printf se escribe:

printf(cadena de control, arg1,arg2,...,argn);
Ejemplo:
printf("%d", 23);
imprime por pantalla: 23

Ejemplo:
printf("%f", 27.64);
imprime por pantalla: 27.64

Ejemplo:
printf("x=%f", 27.64);
imprime por pantalla: x=27.64

Ejemplo:
printf("hola", 'o');
imprime por pantalla: hola

Ejemplo:
printf("%c=%d", 'x',45);
imprime por pantalla: x=45