

Universidad Nacional Autónoma de México

Facultad de Ingeniería

Estructuras de Datos y Algoritmos I

Actividad #2 | Acordeón

Ramírez Pérez Daniela Itzel

26/Febrero/2021

Acordeón Lenguaje C

El lenguaje C es un lenguaje de programación creado el año 1972 por Dennis Ritchie para UNIX.

✓ Comentarios

Es una buena práctica en cualquier lenguaje de programación realizar comentarios para documentar el programa. En C existen dos tipos de comentarios:

- **Por línea:** inicia cuando se insertan los símbolos `//` y termina con el salto de línea
- **Por bloque:** inicia cuando se insertan los símbolos `/*` y termina cuando se encuentran los símbolos `*/` y puede abarcar varios renglones

```
// Comentario por línea
// Otro comentario por línea
/* Comentario por bloque */
/* Comentario por bloque
   que puede ocupar
   varios renglones */
```

✓ Declaración de variables

Para declarar variables en C se sigue la siguiente sintaxis:

```
[modificadores] tipoDeDato identificador [= valor];
```

✓ Tipos de datos

Los tipos de datos básicos en C son:

- **Caracteres:** codificación definida por la máquina.

Imprime	símbolo
variable de tipo carácter	%c
valor del código ASCII del carácter en base 10	%i o %d
valor del código ASCII del carácter en base 8	%o
valor del código ASCII del carácter en base 16	%x
manejar cadenas de caracteres	%s

- **Enteros:** números sin punto decimal.

<i>Tipo</i>	<i>Bits</i>	<i>Valor Mínimo</i>	<i>Valor Máximo</i>
<i>signed char</i>	8	-128	127
<i>unsigned char</i>	8	0	255
<i>signed short</i>	16	-32 768	32 767
<i>unsigned short</i>	16	0	65 535
<i>signed int</i>	32	-2,147,483,648	2 147 483 647
<i>unsigned int</i>	32	0	4 294 967 295
<i>signed long</i>	64	9 223 372 036 854 775 808	9 223 372 036 854 775 807
<i>unsigned long</i>	64	0	18 446 744 073 709 551 615
<i>enum</i>	16	-32 768	32 767

Imprime	símbolo
entero en base 10	%d o %i (%ld ó %li para enteros largos)
entero en base 8	%o
entero en base 16	%x

- **Flotantes:** números reales de precisión normal.

<i>Tipo</i>	<i>Bits</i>	<i>Valor Mínimo</i>	<i>Valor Máximo</i>
<i>float</i>	32	3.4 E-38	3.4 E38
<i>double</i>	64	1.7 E-308	1.7 E308
<i>long double</i>	80	3.4 E-4932	3.4 E4932

Tipo Valor	símbolo
valor real	%f
valor real de doble precisión	%lf
notación científica	%e
redondea la parte fraccionaria a 3 dígitos significativos	%g

- **Dobles:** números reales de doble precisión.

Para poder acceder al valor de una variable se requiere especificar el tipo de dato. Los especificadores que tiene lenguaje C para los diferentes tipos de datos son:

<i>Tipo de dato</i>	<i>Especificador de formato</i>
<i>Entero</i>	%d, %i, %ld, %li, %o, %x
<i>Flotante</i>	%f, %lf, %e, %g
<i>Carácter</i>	%c, %d, %i, %o, %x
<i>Cadena de caracteres</i>	%s

✓ Lectura de datos

scanf es una función que sirve para leer datos de la entrada estándar (teclado), para ello únicamente se especifica el tipo de dato que se desea leer entre comillas y en qué variable se quiere almacenar. Al nombre de la variable le antecede un ampersand (&), esto indica que el dato recibido se guardará en la localidad de memoria asignada a esa variable.

```
scanf ("%i", &varEntera);
```

Para imprimir con formato también se utilizan algunas secuencias de caracteres de escape, C maneja los siguientes:

\a carácter de alarma	\r regreso de carro
\b retroceso	\t tabulador horizontal
\f avance de hoja	\v tabulador vertical
\n salto de línea	'\0' carácter nulo

Modificador de Alcance (Se agregan antes de declarar)	Funcion
const	impide que una variable cambie su valor durante la ejecución del programa\ para crear constantes
static	indica que la variable permanece en memoria desde su creación y durante toda la ejecución del programa, es decir, permanece estática en la memoria.

✓ Operadores Matemáticos

Operador	Operación	Uso	Resultado
+	Suma	125.78 + 62.5	188.28
-	Resta	65.3 - 32.33	32.97
*	Multipliación	8.27 * 7	57.75
/	División	15 / 4	3.75
%	Módulo	4 % 2	0

✓ Operadores lógicos

Operador	Operación	Uso	Resultado
>>	Corrimiento a la derecha	8 >> 2	2
<<	Corrimiento a la izquierda	8 << 1	16
&	Operador AND	5 & 4	4
	Operador OR	3 2	3
~	Complemento ar-1	~2	1

✓ Expresiones lógicas

Una expresión lógica puede tomar únicamente los valores verdadero o falso. Los operadores de relación permiten comparar elementos numéricos, alfanuméricos, constantes o variables.

<i>Operador</i>	<i>Operación</i>	<i>Uso</i>	<i>Resultado</i>
<code>==</code>	Igual que	<code>'h' == 'H'</code>	Falso
<code>!=</code>	Diferente a	<code>'a' != 'b'</code>	Verdadero
<code><</code>	Menor que	<code>7 < 15</code>	Verdadero
<code>></code>	Mayor que	<code>11 > 22</code>	Falso
<code><=</code>	Menor o igual	<code>15 <= 22</code>	Verdadero
<code>>=</code>	Mayor o igual	<code>20 >= 35</code>	Falso

✓ Condiciones simples

<i>Operador</i>	<i>Operación</i>	<i>Uso</i>
<code>!</code>	No	<code>! p</code>
<code>&&</code>	Y	<code>a > 0 && a < 11</code>
<code> </code>	O	<code>opc == 1 salir != 0</code>

✓ Incrementos y decrementos

El operador ++ Agrega una unidad (1) a su operando	preincrementos (++n) posincrementos (n++).
El operador – Resta una unidad (1) a su operando	predecrementos (--n) o posdecrementos (n--).

✓ Estructura de control selectiva if

La estructura de control de flujo más simple es la estructura condicional if, su sintaxis es la siguiente:

```
if (expresión_lógica) {  
    // bloque de código a  
    ejecutar  
}
```

En esta estructura se evalúa la expresión lógica y, si se cumple se ejecutan las instrucciones del bloque que se encuentra entre las llaves de la estructura.

✓ Estructura de control selectiva if-else

La sintaxis de la estructura de control de flujo if-else es la siguiente:

```
if (expresión_lógica) {  
    // bloque de código a ejecutar  
    // si la condición es  
verdadera  
} else {  
    // bloque de código a ejecutar  
    // si la condición es falsa  
}
```

Esta estructura evalúa la expresión lógica y si la condición es verdadera se ejecutan las instrucciones del bloque que se encuentra entre las primeras llaves, si la condición es falsa se ejecuta el bloque de código que está entre las llaves después de la palabra reservada 'else'.

✓ Estructura de control selectiva switch-case

La sintaxis de la estructura switch-case es la siguiente:

```
switch (opcion_a_evaluar){  
    case valor1:  
        /* Código a ejecutar*/  
        break;  
    case valor2:  
        /* Código a ejecutar*/  
        break;  
    ...  
    case valorN:  
        /* Código a ejecutar*/  
        break;  
    default:  
        /* Código a ejecutar*/  
}
```

La estructura switch-case evalúa la variable que se encuentra entre paréntesis después de la palabra reservada switch y la compara con los valores constantes que posee.

Si la opción a evaluar no coincide dentro de algún caso, entonces se ejecuta el bloque por defecto (default).

✓ Enumeración

Existe otro tipo de dato constante conocido como enumeración. Una variable enumerador se puede crear de la siguiente manera:

```
enum identificador {VALOR1, VALOR2, ... , VALORN};
```

Los valores son elementos enteros y constantes.

✓ Estructura de control repetitiva while

```
while (expresión_lógica) {  
    // Bloque de código a  
    repetir  
  
    // mientras que la  
    expresión  
  
    // lógica sea verdadera.  
}
```

✓ Estructura de control repetitiva do-while

```
do {  
    /*  
    Bloque de código que se ejecuta  
    por lo menos una vez y se repite  
    mientras la expresión lógica sea  
    verdadera.  
    */  
} while (expresión_lógica);
```

✓ Estructura de control de repetición for

```
for (inicialización ; expresión_lógica ;  
    operaciones por iteración) {  
    /*  
    Bloque de código  
    a ejecutar  
    */  
}
```

- **Inicialización** Se definen las variables e inicializan sus valores; esta parte solo se ejecuta una vez cuando se ingresa al ciclo (opcional).
- **Expresión lógica** Se evalúa y, si ésta es verdadera, ejecuta el bloque de código, si no se cumple se continúa la ejecución del programa (opcional).
- **Conjunto de operaciones** que se realizan cada vez que termina de ejecutarse el bloque de código y antes de volver a validar la expresión lógica (opcional).

✓ Define	Las líneas de código que empiezan con # son directivas del preprocesador, el cual se encarga de realizar modificaciones en el texto del código fuente, como reemplazar un símbolo definido con #define por un parámetro o texto #define <nombre> <valor>
✓ Break	Un break provoca que el ciclo que lo encierra termine inmediatamente.
✓ Continue	La proposición continue provoca que inicie la siguiente iteración del ciclo de repetición que la contiene.

✓ Arreglos unidimensionales

Un arreglo unidimensional de n elementos en la memoria se almacena de la siguiente manera:



La primera localidad del arreglo corresponde al índice 0 y la última corresponde al índice n-1, donde n es el tamaño del arreglo.

La sintaxis para definir un arreglo en lenguaje C es la siguiente:

```
tipoDeDato nombre[tamaño]
```

- **nombre** se refiere al identificador del arreglo
- **tamaño** es un número entero y define el número máximo de elementos que puede contener el arreglo.

Un arreglo puede ser de los tipos de dato entero, real, carácter o estructura.

✓ Apuntadores

Un apuntador es una variable que contiene la dirección de una variable, es decir, hace referencia a la localidad de memoria de otra variable. Debido a que los apuntadores trabajan directamente con la memoria, a través de ellos se accede con rapidez a un dato.

```
TipoDeDato *apuntador, variable;
```

```
apuntador = &variable;
```

Los apuntadores solo pueden apuntar a direcciones de memoria del mismo tipo de dato con el que fueron declarados; para acceder al contenido de dicha dirección, a la variable apuntador se le antepone `*`.

✓ Arreglos multidimensionales

Lenguaje C permite crear arreglos de varias dimensiones con la siguiente sintaxis:

```
tipoDato nombre[ tamaño ][ tamaño ]...[tamaño];
```

- **nombre** se refiere al identificador del arreglo
- **tamaño** es un número entero y define el número máximo de elementos que puede contener el arreglo.

Acordeón Lenguaje Dart

Dart es un lenguaje open source desarrollado en Google con el objetivo de permitir a los desarrolladores utilizar un lenguaje orientado a objetos y con análisis estático de tipo.

✓ Definir y utilizar variables

Con las variables, se determinan los espacios del programa. Comenzamos con las cifras.

Aquí se ha determinado una nueva **variable** "mi espacio" a la que se le ha asignado el **valor** "174".

```
var mySize = 174;
```

Para **obtener el valor** de las variables, utilizamos el comando "**print**".

```
var mySize = 174;  
print(mySize);
```

Los tipos de variables que se pueden imprimir en la consola son:

- números enteros ("int")
- números decimales ("double")
- "dynamic" puede admitir diferentes valores y expresiones en la secuencia de programación

Cadena de caracteres

Mediante **cadenas de caracteres**, con el tipo de dato **"String"** se pueden procesar diferentes caracteres en el lenguaje Dart.

```
String text1 = 'this is a single line string';
String text2 = '''this is a multiline
    line string with a break''';
print(text1);
print(text2);
```

Mediante la función String, el texto se reproduce encerrando el contenido deseado entre comillas simples o dobles: ' o ".

Si el texto se abre y cierra **con el triple de cualquiera de los dos tipos de comillas** (''' o """"),

Malabarismos de cifras con Dart

Para calcular resultados, Dart emplea **operadores aritméticos**.

```
String product = 'calendar';

String curr = 'EUR';

String isFor = 'for'; // 3 strings for later print use

double singlePrice = 7.96; // floating comma for single price

int amount = 3; // number of ordered calendars

var totalPrice = (amount*singlePrice); // calculation of the total price with
multiplier *

var invTotal = '$totalPrice $curr'; /* Merging of two variables in a new one
by adding a $ sign before the old variables.*/

var result = '$amount $product\s'; //also plus adding a letter "s" for plural

print (invTotal); // creating the screen output

print (isFor);

print (result);
```

Console

```
23.88 EUR
for
3 calendars
```

Establecer enunciados condicionales

Una condicional siempre lleva a una decisión en este sentido: si se da el caso A (**if**), aparece X en la pantalla; si se da el caso B (**elseif**), entonces se muestra Y; si no se da ninguno de los dos casos (**else**), entonces el resultado es Z. Con los comandos Dart entre paréntesis, se obtiene el siguiente código:

```
var tOol = 'Glove';
if (tOol == 'Pliers' || tOol == 'Ruler')
  { print('That is a tool.')}
else if (tOol == 'brush')
  { print('That is a tool.')}
else { print('That is not a tool.')}
}
```

Console

That is not a tool.

Uno más y uno menos también con Dart

Incremento y decremento, que agregan y sustraen gradualmente un resultado con los operadores “++” y “--”.

```
var upAndDown = 50;
print (upAndDown);
print('----');
++upAndDown;
print (upAndDown);
print('----');
upAndDown++;
print (upAndDown);
print('----');
--upAndDown;
print (upAndDown);
print('----');
upAndDown--;
print (upAndDown);
```

Console

50

51

52

51

50

Bucles

Nos referimos, por ejemplo, a las **comparaciones con espacios ya existentes**. Para ello, hay que servirse de la siguiente “expresión”: tenemos el valor A que se modifica continuamente hasta alcanzar el estado B.

```
String myLabel = ' pieces';  
var piece = 3;  
while (piece < 12) {  
  var allThisStuff = '$piece $myLabel';  
  print(allThisStuff);  
  piece++;  
}
```

Console

```
3 pieces  
4 pieces  
5 pieces  
6 pieces  
7 pieces  
8 pieces  
9 pieces  
10 pieces  
11 pieces
```

✓ Operadores

En la tabla se ha aplicado el valor 35 a la variable "muster". var muster = 35;

Tipo de operador	Denominación	Símbolo	Ejemplo	Resultado
Aritmético	Suma	+	var muster + 2;	37
	Resta	-	var muster - 2;	33
	Multiplicacion	*	var muster * 3;	105
	Division	/	var muster / 7;	5
	Divide y retorna un resultado entero	~/	var muster ~/ 7;	11
		+=	var muster += 6;	41
	Asigna suma	-=	var muster -= 7;	28
	Asigna resta	*=	var muster *= 2;	70
De igualdad y relacionales	Asigna producto	/=	var muster /= 7;	5
	Asigna división			
	Es igual	==	var muster == 35;	True
	No es igual	!=	var muster != 44;	True
	Menor que	<	var muster < 44;	True
	Menor o igual que	<=	var muster <= 33;	
Incremento y decremento	Mayor que	>	44 > var muster;	True
	Mayor o igual que	>=	var muster>=23;	
	Incremento	++	++var muster;	36
	Incrementa	++	var muster++;	36
	Decrementa	--	--var muster;	34
	Decrementa	--	var muster--;	34
Lógicos	Valor restante	%	%var muster%3;	2
	Y	&&	muster1 && muster2	... and
	O			muster1
	Negación	!	muster1 ! muster2	... not

Condicionales	If-else	? ... :	var y = muster < 34 ? 15 : 10;	10
	If-else	? ... :	var y = muster < 36 ? 15 : 10;	15
	Verificar si es Null	??	var y = muster ?? 9;	35
	Verificar si es Null	??	var z = 0 ?? muster;	35

✓ Referencias

- Nakayama Cervantes, A., Castañeda Perdomo, M., Solano Gálvez, J. A., García Cano, E. E., Sandoval Montaña, L., & Arteaga Ricci, T. I. (2018, 6 abril). Manual de prácticas del laboratorio de Fundamentos de programación. Laboratorio de Computación Salas A y B. http://odin.fib.unam.mx/salac/practicasp/MADO-17_FP.pdf
- 1&1. (2020, 6 noviembre). Tutorial de Dart: primeros pasos prácticos. IONOS Digitalguide. <https://www.ionos.mx/digitalguide/paginas-web/desarrollo-web/tutorial-de-dart/>
- Divi, V. (2020, 30 octubre). ¿Qué es el lenguaje de programación Dart? inLab FIB. <https://inlab.fib.upc.edu/es/blog/que-es-el-lenguaje-de-programacion-dart>