	<b>Manual de prácticas del Laboratorio de Programación básica</b>	Código:	MADO-18
		Versión:	01
		Página	148/184
		Sección ISO	8.3
		Fecha de emisión	20 de enero de 2017
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

## Guía práctica de estudio 10: Arreglos unidimensionales y apuntadores

---



### *Elaborado por:*


Ing. Jorge A. Solano Gálvez  
Guadalupe Lizeth Parrales Romay

### *Revisado por:*

M.C. Edgar E. García Cano

### *Autorizado por:*

M.C. Alejandro Velázquez Mena

	<b>Manual de prácticas del Laboratorio de Programación básica</b>	Código:	MADO-18
		Versión:	01
		Página	149/184
		Sección ISO	8.3
		Fecha de emisión	20 de enero de 2017
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

## Guía práctica de estudio 10: Arreglos unidimensionales y apuntadores

### Objetivo:

Elaborar programas en lenguaje FORTRAN para resolver problemas que requieran agrupar conjuntos de datos del mismo tipo en arreglos unidimensionales.

### Actividades:

- Crear arreglos unidimensionales.
- Crear apuntadores.

### Introducción

Un arreglo es un conjunto de datos contiguos del mismo tipo con un tamaño fijo, definido al momento de crearse. A cada elemento (dato) del arreglo se le asocia una posición particular. Para acceder a los elementos de un arreglo es necesario utilizar un índice.


### Licencia GPL de GNU

El software presente en esta guía práctica es libre bajo la licencia GPL de GNU, es decir, se puede modificar y distribuir mientras se mantenga la licencia GPL.

```

/*
 *
 * This program is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU General Public License for more details.
 *

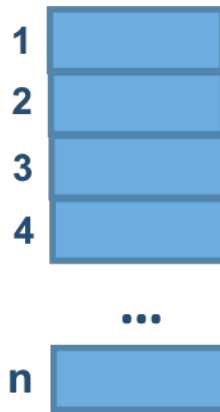
```

	<b>Manual de prácticas del Laboratorio de Programación básica</b>	Código:	MADO-18
		Versión:	01
		Página	150/184
		Sección ISO	8.3
		Fecha de emisión	20 de enero de 2017
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

```
* You should have received a copy of the GNU General Public License
* along with this program. If not, see <http://www.gnu.org/licenses/>.
*
* Author: Jorge A. Solano
*
*/
```

## Arreglos unidimensionales

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




**Figura 1.** Representación de los  $n$  elementos de un arreglo.

Por defecto, la primera localidad del arreglo corresponde al índice 1 y la última corresponde al índice  $n$ , donde  $n$  es el tamaño del arreglo.

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

tipoDeDato nombre(tamaño)

Donde 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 cualquier tipo de dato.

	<b>Manual de prácticas del Laboratorio de Programación básica</b>	Código:	MADO-18
		Versión:	01
		Página	151/184
		Sección ISO	8.3
		Fecha de emisión	20 de enero de 2017
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

Código (arreglo unidimensional)

```

program whileArreglo

c Este programa genera un arreglo unidimensional de
c 5 elementos y accede a cada elemento del arreglo
c a través de un ciclo do while.

integer indice, lista(5)

indice = 1


lista(1) = 10
lista(2) = 8
lista(3) = 5
lista(4) = 8
lista(5) = 7

write (*,*) 'Lista'

do while (indice .LE. 5)
  write (*,*) 'Calificación del alumno',indice,'es',lista(indice)
  indice = indice + 1
enddo

stop
end

```

	<b>Manual de prácticas del Laboratorio de Programación básica</b>	Código:	MADO-18
		Versión:	01
		Página	152/184
		Sección ISO	8.3
		Fecha de emisión	20 de enero de 2017
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

## Código (estructura de repetición do)

```

program doArreglo

c Este programa genera un arreglo unidimensional de
c 5 elementos y accede a cada elemento del arreglo
c a través de un ciclo do.

integer indice, lista(5)

indice = 1

lista(1) = 10
lista(2) = 8
lista(3) = 5
lista(4) = 8
lista(5) = 7

write (*,*) 'Lista'

do indice = 1, 5
  write (*,*) 'Calificación del alumno',indice,'es',lista(indice)
enddo

stop
end

```

## 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.

La sintaxis para declarar un apuntador y para asignarle la dirección de memoria de otra variable es, respectivamente:

```

TipoDeDato, pointer :: apuntador
TipoDeDato, target :: variable
apuntador => variable

```

	<b>Manual de prácticas del Laboratorio de Programación básica</b>	Código:	MADO-18
		Versión:	01
		Página	153/184
		Sección ISO	8.3
		Fecha de emisión	20 de enero de 2017
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

La declaración de una variable apuntador inicia con la palabra reservada `pointer`, seguida de `::` y el identificador del apuntador.

Los apuntadores solo pueden apuntar a direcciones de memoria del mismo tipo de dato con el que fueron declarados.

Código (apuntadores)

```

program apuntador

c Este programa crea un apuntador de tipo caracter

c Definición de un apuntador de tipo caracter
c utilizando la palabra reservada pointer
character, pointer :: ap

c Se utiliza la palabra reservada target para indicar
c que la variable c puede ser apuntada por un pointer
character, target :: c


c = 'a'

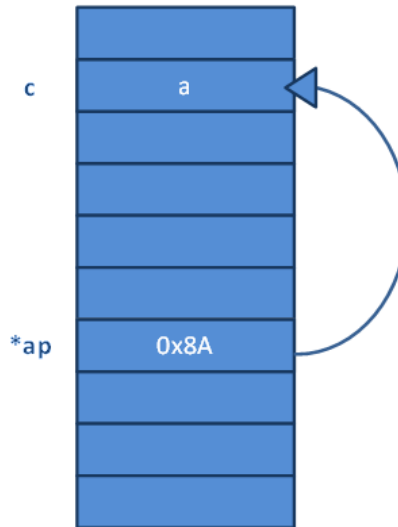
c se realiza la asignación, ap apunta a la localidad
c de memoria de c
ap => c

write (*,*) 'Caracter:', ap
write (*,*) 'Codigo ASCII:', ichar(ap)

stop
end

```

	<b>Manual de prácticas del Laboratorio de Programación básica</b>	Código:	MADO-18
		Versión:	01
		Página	154/184
		Sección ISO	8.3
		Fecha de emisión	20 de enero de 2017
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			



**Figura 2.** Representación de una variable apuntador en la memoria.

### Código (apuntadores)

```


program apuntador2

c Este programa accede a las localidades de memoria de
c distintas variables a través de un apuntador

c declaración de una variable apuntador
integer, pointer :: apEnt
c se declara que a, b y el arreglo c pueden ser apuntadas
integer, target :: a, b, c(10)

a = 5
b = 10
c(1) = 5
c(2) = 4
c(3) = 3
c(4) = 2
c(5) = 1
c(6) = 9
c(7) = 8
c(8) = 7
c(9) = 6
c(10) = 0

```

	<b>Manual de prácticas del Laboratorio de Programación básica</b>	Código:	MADO-18
		Versión:	01
		Página	155/184
		Sección ISO	8.3
		Fecha de emisión	20 de enero de 2017
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

c se asigna la localidad de memoria de 'a' a la variable apuntador

```

apEnt => a

write (*,*) 'a = ',a
write (*,*) 'b = ',b
write (*,*) 'c(10) = ',c
write (*,*) 'apEnt => a'

b = apEnt
write (*,*) 'b = apEnt -> b =', b

b = apEnt + 1
write (*,*) 'b = apEnt + 1 -> b =', b

apEnt = 0
write (*,*) 'apEnt = 0 -> a =', a


apEnt => c(1)
write (*,*) 'apEnt => c(1) -> apEnt =', apEnt

stop
end

```

Es posible saber si un apuntador está asociado a una variable a través de la función ASSOCIATED. Por otra parte, la función NULLIFY permite desasociar el apuntador de una variable.



	<b>Manual de prácticas del Laboratorio de Programación básica</b>	Código:	MADO-18
		Versión:	01
		Página	156/184
		Sección ISO	8.3
		Fecha de emisión	20 de enero de 2017
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

Código (associated y nullify)

```

program associatedNullify

c Este programa valida si un apuntador tiene referencia
c hacia una variable o no. También permite desasociar el
c valor de un apuntador.

integer, target :: arr(5)
integer, pointer :: apArr

arr = (/5, 4, 3, 2, 1/)
c apArr apunta a la primera localidad del arreglo
apArr => arr(1)


if (associated(apArr)) then
    write (*,*) 'apArr está apuntando a -> arr(1) = ', apArr
end if

c se elimina la asociación del apuntador hacia la variable.
nullify(apArr)
write (*,*) '¿apArr está asociado?', (associated(apArr))
write (*,*) 'apArr -> ', apArr

c apArr apunta a la tercera localidad del arreglo
apArr => arr(3)
if (associated(apArr)) then
    write (*,*) 'apArr está apuntando a -> arr(3) = ', apArr
endif

stop
end

```

	<b>Manual de prácticas del Laboratorio de Programación básica</b>	Código:	MADO-18
		Versión:	01
		Página	157/184
		Sección ISO	8.3
		Fecha de emisión	20 de enero de 2017
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

### Código (cadenas)

```

program cadena

c Este programa lee una palabra del teclado, de una longitud
c máxima de 20 caracteres
character (len = 20):: palabra

write (*,*) 'Ingrese una palabra: '
read (*,*) palabra
write (*,*) 'La palabra ingresada es: ', palabra

stop
end

```

### Código (arreglos como cadenas)

```

program recorrePalabra

c Este programa lee una palabra de 20 máximo caracteres
c Después imprime en la salida estándar (pantalla) la
c palabra y además imprime el arreglo carácter por
c carácter


character palabra(20)
integer i

write (*,*) 'Ingrese una palabra: '
read (*,*) palabra
write (*,*) 'La palabra ingresada es: ', palabra

do i = 1, 20, 1
write (*,*) palabra(i)
enddo

stop
end

```

	<b>Manual de prácticas del Laboratorio de Programación básica</b>	Código:	MADO-18
		Versión:	01
		Página	158/184
		Sección ISO	8.3
		Fecha de emisión	20 de enero de 2017
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

## Bibliografía

- Oracle (2010). Fortran 77 Lenguaje Reference. Consulta: Julio de 2015. Disponible en: <http://docs.oracle.com/cd/E19957-01/805-4939/>
- Stanford University (1995). Fortran 77 Tutorial. Consulta: Julio de 2015. Disponible en: [http://web.stanford.edu/class/me200c/tutorial\\_77/](http://web.stanford.edu/class/me200c/tutorial_77/)