UNT – FACEyT - DEEC

Programación I

Año 2016

Unidad 3: Arreglos en Lenguaje C

Concepto

- Dato estructurado ***estático***, compuesto por un conjunto ***ordenado*** de valores ***homogéneos***, que se alojan en domicilios contiguos de memoria.
- Todos los elementos están referenciados por una única variable.
- A cada elemento se accede por medio de uno o más índices que indican su posición relativa dentro del conjunto.

Unidad 3: Arreglos en Lenguaje C
Declaración

Sintaxis de la declaración de un arreglo

Unidimensional (vector o lista):

`tipo_de_dato` *nombre*[cantidad_de_elementos];

Bidimensional (matriz o tabla):

`tipo_de_dato` *nombre*[nro_filas][nro_columnas];

La cantidad de elementos, filas y/o columnas debe ser una variable o constante entera. Si es variable debe tener un valor antes de la declaración del arreglo.

Unidad 3: Arreglos en Lenguaje C
Declaración

Ejemplos:

Unidimensionales:

```
char    vectChar [18];
int     lista [15], vecInt[25];
short   edades[7];
float   vector_1 [10], vector_2 [9];
```

Bidimensionales:

```
char    vecStr [8][18],  matChar [5][15];
int     tablaActual [12][12];
float   tablaMes [5][7];
```

Unidad 3: Arreglos en Lenguaje C

Características

1. El índice del primer elemento es **siempre** cero. En consecuencia el índice de un arreglo de **n** elementos varían de **0** a **n-1.**
2. El nombre del arreglo es una **constante puntero** que contiene la dirección de memoria del primer elemento.

`vector ≡ &vector[0]`

3. El compilador no hace comprobación de límites.

Unidad 3: Arreglos en Lenguaje C
Inicialización

Los arreglos pueden inicializarse:

1. en la declaración,
2. asignando valores a los elementos en forma aleatoria, o
3. asignando valores a los elementos mediante un ciclo repetitivo.

Unidad 3: Arreglos en Lenguaje C

Inicialización en la declaración

Ejemplos :

```
char   vectChar [18] = {'a', 'e', 'i', 'o', 'u'};
int    lista [15] = {}, vecInt[] = {1, 2, 3, 4, 5, 6, 7, 8, 9};
short  edades[7] = {25, 19, 18, 21, 19, 17, 20, 22, 16};
float  vector_1 [] = {}, vector_2 [9] = {1.1, 2.2, 3.3, 4.4};

char   vecStr [8][18] = {{'p','q'},{'f','g'}};
char   matChar [5][15] = {{},{}};
int    tablaActual [12][12] = {{1, 2, 3, 4},{1,2,3},{1,2}};
float  tablaMes [5][7] = {{},{},{},{0.1, 0.2, 0.3}};
```

Unidad-3-Ej-03.c

Unidad-3-Ej-04.c

Unidad-3-Ej-05.c

Unidad 3: Arreglos en Lenguaje C

Inicialización aleatoria

Ejemplos :

vectChar [1] = 'e';

lista [15] = 8, *vecInt*[0] = 10;

edades[9] = 17;

vector_1 [2] = 3.33;

vector_2 [3] = 0.707;

vecStr [3][1] = 'z';

matChar [3][3] = '3';

tablaActual [1][2] = 12;

tablaMes [2][0] = 0.12;

Unidad-3-Ej-06.c

Unidad-3-Ej-07.c

Unidad 3: Arreglos en Lenguaje C

Inicialización en un ciclo repetitivo

Ejemplos :

```
for ( i = 0 ; i < sizeof(vectChar)/sizeof(char) ; i++)
      scanf("%c", &vectChar[i]);

for ( i = 0 ; i < sizeof(lista)/sizeof(int) ; i++)
      scanf("%i", &lista[i]);

for ( i = 0 ; i < sizeof(edades)/sizeof(short) ; i++)
      scanf("%i", &edades[i]);

for ( i = 0 ; i < sizeof(vector1)/sizeof(float) ; i++)
      scanf("%f", &float[i]);
```

Unidad-3-Ej-08.c

Unidad-3-Ej-09.c

Unidad 3: Arreglos en Lenguaje C
Asignación con valores aleatorios

Para cargar arreglos con números aleatorios se puede utilizar la función estándar de C **`int rand()`**. Esta función devuelve un entero positivo (pseudo aleatorio) entre 0 y el valor máximo de entero que usa el compilador.

Si a esta función se la utiliza en forma reiterada devolverá la misma secuencia de valores. Una forma de evitar esta reiteración de valores es reiniciar la "semilla" de los números aleatorios con la función **`srand(int)`** utilizando como parámetro el reloj del sistema:

```
time_t t;
time(&t);
srand(t);
```

Unidad 3: Arreglos en Lenguaje C

Matrices típicas

1. Matriz cuadrada: tiene la misma cantidad de filas que de columnas, **`M x M`**.

   Unidad-3-Ej-11.c

2. Diagonal principal: pertenecen a la diagonal principal todos los elementos **`a[i][j]`**, de una matriz cuadrada, tal que:

   `j = i`

   Unidad-3-Ej-12.c

3. Diagonal secundaria: pertenecen a la diagonal principal todos los elementos **`a[i][j]`**, de una matriz cuadrada, tal que:

   `j = M – i - 1`

   Unidad-3-Ej-13.c

4. Matriz diagonal: es aquella matriz cuadrada tal que:

   `a[i][j] <> 0 si j = i`

   `a[i][j] = 0 si j <> i`

   Unidad-3-Ej-14.c

Unidad 3: Arreglos en Lenguaje C

Matrices típicas

5. Matriz identidad o unitaria: es aquella matriz cuadrada tal que:

Unidad-3-Ej-15.c

```
a[i][j] = 1 si j = i
a[i][j] = 0 si j <> i
```

6. Matriz escalar: es aquella matriz cuadrada tal que:

```
a[i][j] = k si j = i
a[i][j] = 0 si j <> i
```

Unidad-3-Ej-16.c

7. Matriz triangular superior: es aquella matriz cuadrada tal que:

```
a[i][j] <> 0 si j >= i
a[i][j] = 0 si j < i
```

Unidad-3-Ej-17.c

func_mat.h

Unidad 3: Arreglos en Lenguaje C

Matrices típicas

8. Matriz triangular inferior: es aquella matriz cuadrada tal que:

```
a[i][j] <> 0 si j <= i
a[i][j] = 0 si j > i
```

Unidad-3-Ej-18.c

9. Matriz simétrica: es aquella matriz cuadrada tal que:

```
a[i][j] = a[j][i]
```

Unidad-3-Ej-19.c

10. Matriz antisimétrica: es aquella matriz cuadrada tal que:

```
a[i][j] = -a[j][i]
```

Unidad-3-Ej-20.c

11. Matriz transpuesta: la transpuesta de un matriz **A** de orden **M x N** es una matriz **B** de orden **N x M** tal que:

```
B[i][j] = A[j][i]
```

Unidad-3-Ej-21.c

Unidad 3: Arreglos en Lenguaje C
Operaciones con arreglos

Suma de vectores

Para que la suma entre dos o más vectores sea posible los vectores deben tener el mismo tamaño

Sean V1, V2, ... y Vk vectores de orden n, el vector suma será:

```
S = (( V1(0) + V2(0) + ... + Vk(0) ),
     ( V1(1) + V2(1) + ... + Vk(1) ),
     . . . . . . . . . . . . . . . . ,
     ( V1(n-1) + V2(n-1) + ... + Vk(n-1)))
```

Unidad 3: Arreglos en Lenguaje C

Operaciones con arreglos

Resta de vectores

Para que la resta entre dos vectores sea posible, los vectores deben tener el mismo tamaño.

Sean V1 y V2 vectores de orden n, el vector resta

`R = V1-V2` será:

```
R =  ((V1(0)-V2(0)), (V1(1)-V2(1)), . . .,
     (V1(n-1)-V2(n-1)))
```

Unidad 3: Arreglos en Lenguaje C

Operaciones con arreglos

Producto de un vector por un escalar

Para realizar esta operación se debe multiplicar cada elemento del vector por el valor del escalar.

Sea:

V =(V(0), V(2), ..., V(n-1))

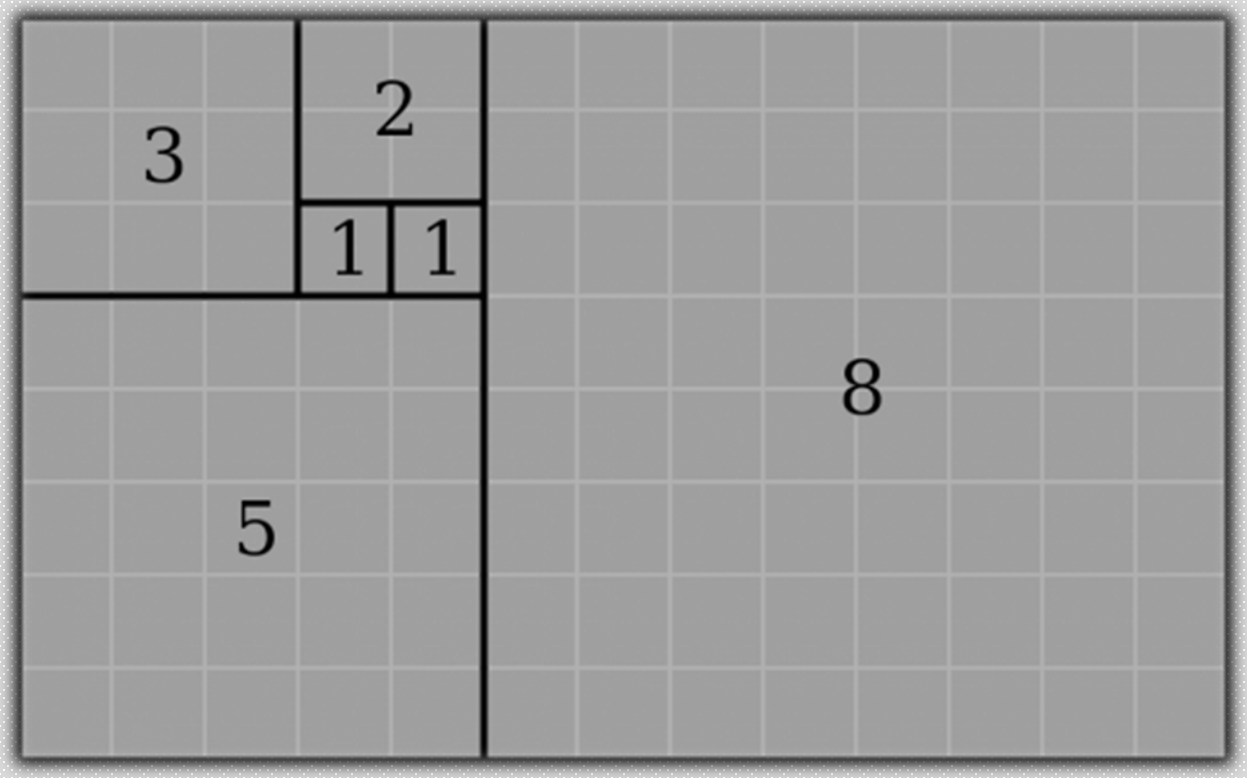
y **k** un escalar, el vector producto **P** será:

P = k*V =(k*V(0), k*V(2), ..., k*V(n-1))

Unidad 3: Arreglos en Lenguaje C

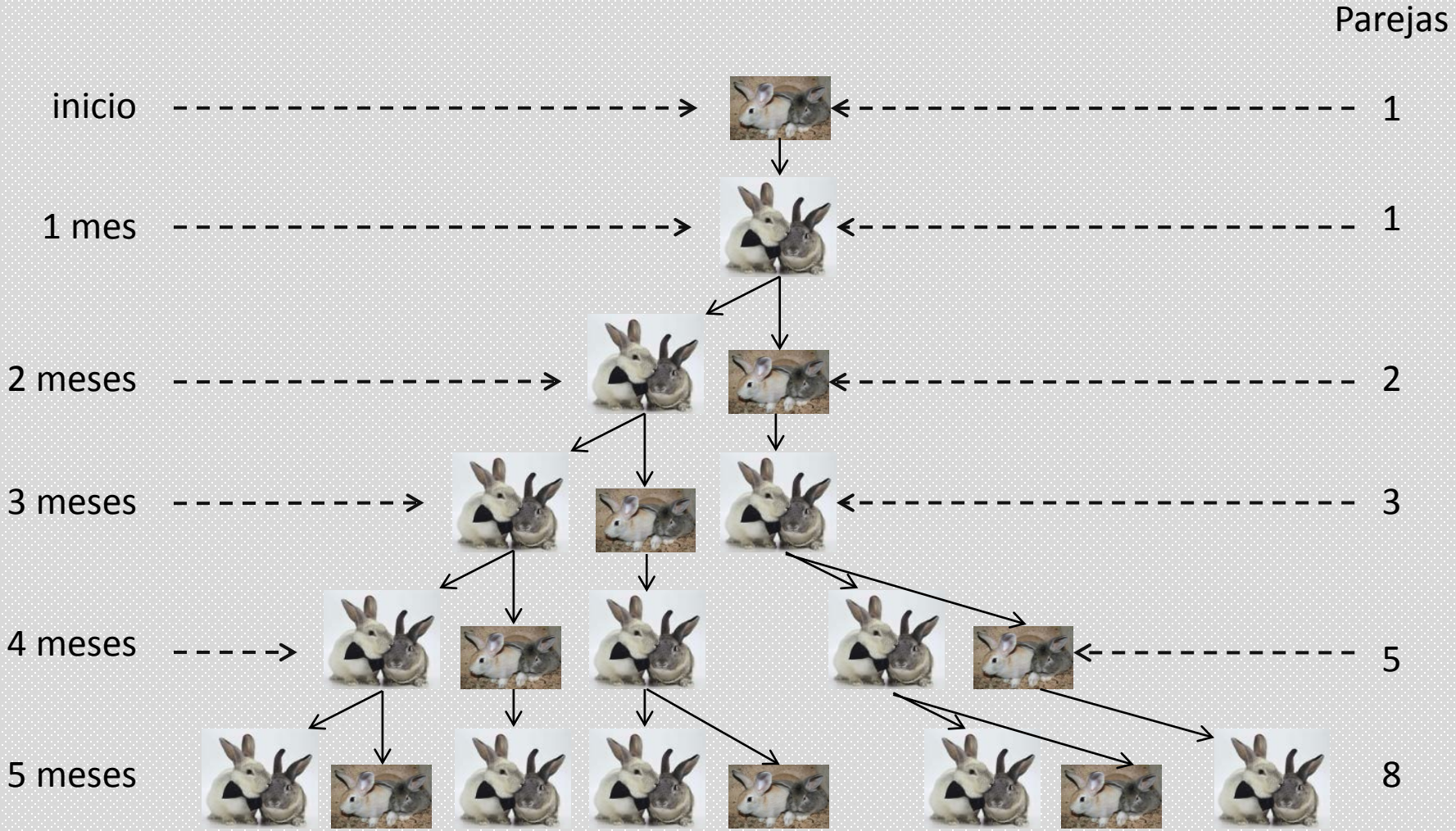
Operaciones con arreglos

Sucesión de Fibonacci



Unidad 3: Arreglos en Lenguaje C

Operaciones con arreglos



Unidad 3: Arreglos en Lenguaje C

Operaciones con arreglos

Generar un vector con la sucesión de Fibonacci

```
void generarFibo(int n, int v[])
   {
      int i;

      v[0] = v[1] = 1;
      for ( i=2 ; i<n ; i++)
           v[i] = v[i-1]+v[i-2];
   }
```

Unidad 3: Arreglos en Lenguaje C

Operaciones con arreglos

1. Suma de matrices

Sólo se pueden sumar matrices si son del mismo orden.

Sean **A** y **B** dos matrices de orden **M** x **N**, la matriz suma **S** estará compuesta por los elementos **S[i][j]** tal que:

$$\mathbf{S[i][j]} = \mathbf{A[i][j]} + \mathbf{B[i][j]}$$

Ejemplo:

$$\begin{pmatrix} 1 & -2 & 5 \\ 0 & 9 & 12 \\ -21 & -6 & 17 \\ 23 & -1 & 7 \end{pmatrix} + \begin{pmatrix} -3 & 12 & -1 \\ 1 & -4 & 3 \\ 12 & -3 & 11 \\ 3 & -10 & 17 \end{pmatrix} = \begin{pmatrix} -2 & 10 & 4 \\ 1 & 5 & 15 \\ -9 & -9 & 28 \\ 26 & -11 & 24 \end{pmatrix}$$

Unidad 3: Arreglos en Lenguaje C

Operaciones con arreglos

Suma de matrices

```
void sumaMatInt(int m,int n,int a[30][30],
          int b[30][30], int s[30][30])
{
   int i, j;
      for ( i = 0 ; i < m ; i++)
          for ( j = 0 ; j < n ; j++)
              s[i][j] = a[i][j] + b[i][j];
return;
}
```

Unidad 3: Arreglos en Lenguaje C

Operaciones con arreglos

2. Asignar cero a los elementos de la diagonal principal. Unidad-3-Ej-27.c
3. Asignar cero a los elementos de la diagonal secundaria. Unidad-3-Ej-28.c
4. Asignar cero a los elementos de la cruz de una matriz.
   - Caso 1: M y N impares Unidad-3-Ej-29.c
   - Caso 2: M impar y N par Unidad-3-Ej-30.c
   - Caso 3: M y N pares Unidad-3-Ej-31.c
   - Caso 4: M par y N impar Unidad-3-Ej-32.c

func_matrices2.h

Unidad 3: Arreglos en Lenguaje C
Operaciones con arreglos

3. Producto de dos matrices.

```
for ( i = 0; i < m ; i++ )
  for ( j = 0; j < q ; j++ )
    {
     c[i][j] = 0;
     for ( k = 0 ; k < n ; k++ )
        c[i][j] = c[i][j] + a[i][k] * b[k][j];
    }
```

func_matrices2.h

Unidad-3-Ej-33.c

Unidad 3: Arreglos en Lenguaje C
Ordenamiento y búsqueda

Cuando se dispone de un conjunto de datos las tareas que se realizan, con frecuencia, sobre ellas son:

- Ordenamiento
- Búsqueda

El **ordenamiento** consiste en poner los elementos de una estructura de datos en una secuencia dada por una relación de orden.

La **búsqueda** radica en localizar un elemento con ciertas propiedades dentro de una estructura de datos

Unidad 3: Arreglos en Lenguaje C
Ordenamiento - Clasificación

Los métodos de ordenamiento se clasifican:

- Según el lugar dónde se realice:
  - Interno: Se realiza sobre la memoria principal (RAM) del computador.
  - Externo: Se realiza sobre un dispositivo secundario de almacenamiento aunque, necesariamente, se debe hacer uso de la RAM.
- Según la forma de ordenar:
  - Intercambio
  - Selección
  - Inserción

Unidad 3: Arreglos en Lenguaje C
Ordenamiento - Clasificación

Intercambio

Se basa en comparar los elementos del arreglo e intercambiarlos si su posición actual o inicial es contraria o inversa a la deseada.

Selección

Trabaja bajo el principio básico de seleccionar el elemento más pequeño (o más grande) del arreglo y colocarlo en la posición más baja (o más alta) del arreglo.

Inserción

Consiste en insertar los elementos no ordenados del arreglo en sub-arreglos del mismo que ya estén ordenados.

Unidad 3: Arreglos en Lenguaje C

Ordenamiento – Método de la burbuja (intercambio)

Es el de algoritmo más simple. Consiste en comparar todos los elementos de la lista contra todos los elementos de la lista para encontrar el menor o mayor según se ordene en forma ascendente o descendente.

```
void ordenBurbujaAscen ( int n, int  v[])
{
     int i, j;
     register int temp;
        for (i = 0 ; i < n – 1 ; i++)
              for ( j = i + 1 ; j < n ; j++)
                    if ( v[j] < v[i] )
                          {
                          temp = v[i];
                          v[i] = v[j];
                          v[j] = temp;
                          }
}
```

Unidad 3: Arreglos en Lenguaje C
Búsqueda

Existen, básicamente, dos métodos:

- Lineal o secuencial
- Binaria

Secuencial o lineal: Este método se puede aplicar sobre una estructura de datos ordenada o desordenada. Comienza desde el primer elemento y recorre la estructura secuencialmente buscando la coincidencia entre la clave de búsqueda (elemento a buscar) y algún elemento de dicha estructura. La búsqueda finaliza con la primera ocurrencia o cuando se llega al final de la estructura y el elemento a buscar no se encuentra.

Unidad 3: Arreglos en Lenguaje C
Búsqueda lineal o secuencial

```
int busquedaLineal (int n, int v[], int cb)
{
     int i;
          for ( i = 0 ; i < n ; i++ )
               if ( cb == v[i])
                    break;
return i;
}
```

Unidad 3: Arreglos en Lenguaje C

Búsqueda binaria

- Este método sólo se puede utilizar cuando el arreglo está ordenado.
- Toma el elemento central del arreglo y lo compara con la clave de búsqueda si esta es mayor la búsqueda se realiza, de la misma manera, en la mitad "derecha" del vector, caso contrario la búsqueda se realiza en la mitad "izquierda".
- Así sucesivamente hasta encontrar o no la coincidencia. La búsqueda finaliza con la primera ocurrencia o cuando no se puede subdividir el arreglo y el elemento a buscar no se encuentra.

Unidad 3: Arreglos en Lenguaje C

Búsqueda binaria

```
int busqBinariaInt (int n, int v[], int cb)
{
  int central, izq, der;
  izq = 0;
  der = n-1;

  while (izq <= der)
  {
       central = (izq+der)/2;
       if (cb > v[central])
              izq = central + 1;
       else
              if (cb < v[central])
                     der = central - 1;
              else
                     return central;
  }
return -1;
}
```

Unidad 3: Arreglos en Lenguaje C
Cadena de caracteres – Datos tipo *char*

- Un dato de tipo caracter es aquel que puede tomar por valor un símbolo perteneciente al conjunto de los caracteres que puede representar el ordenador.
- El estándar ISO/IEC C11 utiliza el juego 1 de caracteres ASCII (American Standard Code for Information Interchange).
- En lenguaje C, el valor de un dato de tipo carácter se debe representar entre comillas simples (').

  Por ejemplo: `'m'`, `'W'`, `'7'`, etc.
- Las variable y constantes tipo **`char`** se declaran de la forma convencional

Unidad 3: Arreglos en Lenguaje C

Cadena de caracteres - Juego 1 de caracteres ASCII

	0	1	2	3	4	5	6	7	8	9
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT
1	LF	VT	FF	CR	SO	SI	DLE	DC1	DC2	DC3
2	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS
3	RS	US		!	"	#	$	%	&	'
4	(	)	*	+	,	-	.	/	0	1
5	2	3	4	5	6	7	8	9	:	;
6	<	=	>	?	@	A	B	C	E	D
7	F	G	H	I	J	K	L	M	N	O
8	P	Q	R	S	T	U	V	W	X	Y
9	Z	[	\	]	^	_	`	a	b	c
10	d	e	f	g	h	i	j	k	l	m
11	n	o	p	q	r	s	t	u	v	w
12	x	y	z	{	|	}	~	DEL		

Caracteres no imprimibles

Caracteres imprimibles

Unidad 3: Arreglos en Lenguaje C
Funciones de caracteres (**ctype.h**)

int isdigit(char c);	Devuelve un valor distinto que **0** si el caracter **c** es un dígito (**0-9**)
int isalpha(char c);	Devuelve un valor distinto que **0** si el caracter **c** es un caracter alfabético (**A-Z** , **a-z**)
int isalnum(char c);	Devuelve un valor distinto que **0** si el caracter **c** es un caracter alfanumérico (**A-Z** , **a-z** , **0-9**)
int islower(char c);	Devuelve un valor distinto que **0** si el caracter **c** es un caracter alfabético en minúsculas (**a-z**)
int isupper(char c)	Devuelve un valor distinto que **0** si el caracter **c** es un caracter alfabético en mayúsculas (**A-Z**)
toupper(char c);	Devuelve el carácter **c** en mayúscula
tolower(char c);	Devuelve el carácter **c** en minúscula

Unidad 3: Arreglos en Lenguaje C

Cadena de caracteres

- Una cadena de caracteres (string) es una sucesión finita de caracteres alfanuméricos.
- En lenguaje C no existe un tipo primitivo de datos que pueda contener una cadena de caracteres.
- Una cadena de caracteres se declara como un vector (arreglo unidimensional) de caracteres, pero puede tratarse como arreglo o como cadena propiamente dicha.
- El lenguaje C tiene una biblioteca específica de funciones definidas para tratar un arreglo de caracteres como cadena de caracteres: **`string.h`**

Unidad 3: Arreglos en Lenguaje C

Cadena de caracteres

- Los literales **string** se indica entre comillas dobles:

"cadena", "Programacion I", "12345", etc.

Declaración de una cadena de caracteres

char nombre_de_la_cadena [tamaño];

Ejemplos:

char cadena1 [10], cadena2[12];

char calle [10];

Unidad 3: Arreglos en Lenguaje C

Cadena de caracteres

Inicialización en la declaración

Ejemplos:

```
char cad1[10];
char cad2[10] = {};
char cad3[] = {'C', 'u', 'r', 's', 'o'};
char cad4[] = "Curso";
char cad5[5] = "Universidad"; // ????
char cad6[8] = "clase";
char cad7[] = {};
char cad8[7] = {'a', 'e', 'i', 'o', 'u'};
```

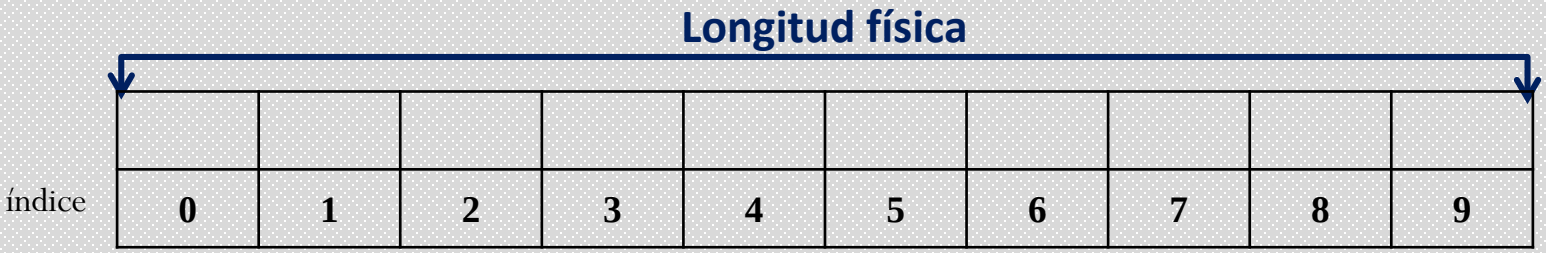
Unidad 3: Arreglos en Lenguaje C

Cadena de caracteres

Longitud física de una cadena de caracteres

Es el espacio de memoria que se reserva en el momento de la declaración de la cadena.

Ejemplo:



Se reservaron 10 bytes para alojar 10 caracteres en total.

La longitud física se obtiene con el operador **`sizeof`**`( )`.

Unidad 3: Arreglos en Lenguaje C
Cadena de caracteres

Longitud lógica de una cadena de caracteres

Es el espacio que efectivamente ocupan los caracteres de la cadena.

Cuando se realiza la asignación de una cadena (en la declaración, por lectura o por copia), automáticamente se inserta el terminador nulo (**'\0'**) en el espacio contiguo al último caracter de dicha cadena.

El caracter nulo no cuenta en la longitud lógica, sí en la física. Por lo tanto la longitud lógica de una cadena no debe ser mayor a la longitud física menos 1.

Máxima Longitud Lógica <= Longitud Física – 1

La longitud lógica se obtiene con la función **`strlen`**(s)

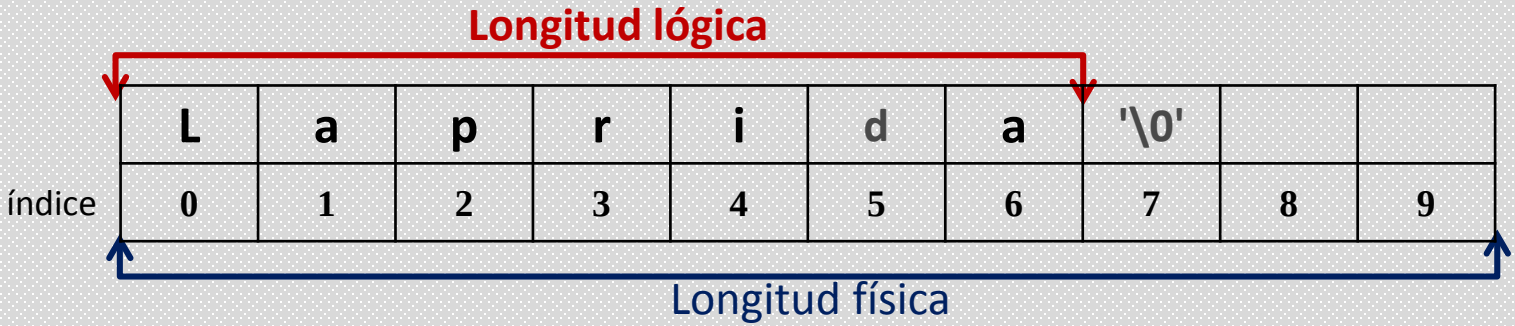
Unidad 3: Arreglos en Lenguaje C

Cadena de caracteres

Ejemplo:

char calle [10] = "Laprida";

En el momento de la asignación la posición de los caracteres es la siguiente:



La inserción del terminador nulo es automática cuando se asigna una cadena, no así cuando se asignan un conjunto de caracteres.

Unidad 3: Arreglos en Lenguaje C

Cadena de caracteres

Lectura de cadenas de caracteres (`strings`)

Para la lectura de cadena de caracteres se utilizan, entre otras, las siguientes funciones de la librería **`stdio.h`**:

- **`scanf`**("%s", `char` *s);
- **`fscanf`**(`stdin`, "%s", `char` *s);

  `stdin`: dispositivo estándar de entrada.
- **`fgets`** (`char` *s, num_char, `stdin`);

  num_char: es la cantidad de caracteres que se leerán.

Unidad 3: Arreglos en Lenguaje C

Cadena de caracteres

Las funciones `scanf()` y `fscanf()`admiten un modificador de longitud que limita la cantidad de caracteres que se asignarán a la cadena en el momento de la lectura. Este modificador es un entero que se coloca entre operador módulo (**%**) y el especificador de tipo de dato (s). Ejemplo:

```
char s1[15];
char s2[15];
scanf("%6s", s1);
scanf("%5s", s2);
```

Si se ingresa la cadena "**universitario**" para **s1**, sólo asignará "**univer**" el resto queda en el buffer del teclado y le asigna a **s2** "**sitar**". La cadena "**io**" quedará en el buffer.

Unidad 3: Arreglos en Lenguaje C

Cadena de caracteres

Con la función:

```
fgets(char *s, num_char, stdin);
```

se presenta la misma situación que en los casos anteriores. Si la cantidad de caracteres ingresados por teclado supera la cantidad *num_char*, los caracteres restantes quedarán en el buffer del teclado y se asignarán a cualquier variable tipo `char` o cadena que se quiera leer en forma inmediata posterior.

Si la cantidad de caracteres ingresados por teclado es menor que *num_char - 2* la función asignará a la cadena el caracter de salto de línea (ENTER) antes del terminador nulo. Para quitar este caracter se puede usar la siguiente instrucción:

```
s[strlen(calle)-1]='\0';
```

Unidad 3: Arreglos en Lenguaje C

Cadena de caracteres

Las funciones `scanf(...)` y `fscanf(...)`, también admiten un limitador de formato de propósito general llamado "*Juego de inspección"* que actúa como "filtro" para la lectura de una cadena de caracteres. Este juego de inspección tiene el siguiente formato:

`scanf(%[...]s, char*);`

Dentro de los corchetes se indican los caracteres que son admitidos o no dentro de la cadena. Estos filtros se comportan en forma diferente con los distintos sistemas operativos.

Unidad 3: Arreglos en Lenguaje C

Cadena de caracteres

Filtro	Acción
[A-Z]	Permite sólo caracteres alfabéticos en mayúsculas.
[a-z]	Permite sólo caracteres alfabéticos en minúsculas
[A-Za-z]	Permite sólo caracteres alfabéticos
[0-9]	Permite sólo caracteres numéricos
[A-Za-z0-9]	Permite caracteres alfanuméricos
[A-Za-z0-9]	Permite caracteres alfanuméricos y espacio en blanco
[^0-9]	No permite caracteres numéricos.

Tener en cuenta que los caracteres que se ingresaron por teclado y fueron filtrados quedan en el buffer.

Unidad-3-Ej-44.c

Unidad-3-Ej-45.c

Cadena de caracteres

Salida por monitor de cadenas de caracteres (strings)

Para las cadenas de caracteres se utilizan, entre otras, las siguientes funciones de la librería `stdio.h`:

- `printf("%s", char*);`
- `fprintf(stdout, "%s", char*);`

  `stdout` : dispositivo estándar de salida (monitor)

- `fputs (char*, stdout);`

Unidad 3: Arreglos en Lenguaje C

Cadena de caracteres - Funciones más utilizadas

`double atof(char *s);`	Convierte a doble precisión la cadena **s** `stdlib.h`
`int atoi(char *s);`	Convierte a entero la cadena **s** `stdlib.h`
`long atol(char *s);`	Convierte a entero largo la cadena **s** `stdlib.h`
`strcat (char *s1, char *s2);`	Concatena la cadena **s2** al final de la cadena **s1**
`char *strchr (char *s, char c)`	Busca el primer elemento de la cadena **s** que coincida con el caracter **c**
`int strcmp (char *s1, char *s2);`	Compara las cadenas **s1** y **s2** y devuelve: **0** son iguales, **>0** si **s1>s2** y **<0** si **s1<s2**
`char *strcpy (char *s1,char *s2);`	Copia la cadena **s2** en la cadena **s1**
`strlen (char *s);`	Devuelve la longitud lógica de la cadena **s**

Unidad 3: Arreglos en Lenguaje C
Cadena de caracteres

Problemas de ejemplo:

1. Ingresar una cadena, poner en mayúscula la primera letra y el resto en minúsculas. Unidad-3-Ej-47.c

2. Leer cadena con caracteres numéricos y convertir a entero. Unidad-3-Ej-48.c

3. Leer cadena que representa un número real y convertir a punto decimal flotante. Unidad-3-Ej-49.c

Unidad 3: Arreglos en Lenguaje C
Cadena de caracteres

Arreglos de cadena de caracteres

Si una cadena de caracteres se declara:

```
char nombreCad[longFisica];
```

Un vector de cadenas de caracteres se declara como:

```
char nombreVect[cantCadenas][longFisica];
```

Ejemplo:

```
char  vectCad[15][20];
```

Este vector podrá contener hasta 15 cadenas de una longitud lógica máxima de 19 caracteres cada una.

Unidad 3: Arreglos en Lenguaje C

Cadena de caracteres

Parámetros de la función main()

La cabecera de la función main() es:

```
int main(int argc, char *argv[])
{
        ...
}
```

El primer prámetro, `argc`, es el contador de los argumentos que se escriben en la línea de órdenes, cuando se ejecuta el programa.

El segundo, `argv[]`, es el arreglo de cadenas de caracteres donde se guardan los argumentos de llamada.

Unidad 3: Arreglos en Lenguaje C

Cadena de caracteres

Ordenamiento y búsqueda en vectores de string

1. Ordenamiento:

   Método de la burbuja — Unidad-3-Ej-52.c

2. Búsqueda:

   Búsqueda lineal — Unidad-3-Ej-53.c

   Búsqueda binaria — Unidad-3-Ej-54.c

Cadena de caracteres

Punteros a cadenas de caracteres

Una cadena se declara:

```
char nombreCad[longFisica];
```

también se admite declararla como:

```
char *nombreCad;
```

en esta declaración nombreCad tomó por defecto una dirección cualquiera de memoria. Si se le asigna una cadena es probable que se produzca un error de segmentación. Para utilizar un puntero a cadena se debe asignar memoria dinámica.

Unidad 3: Arreglos en Lenguaje C

Cadena de caracteres

Ordenamiento y búsqueda en vectores de `string`

(implementados con punteros a `char`)

1. Ordenamiento:

   Método de la burbuja — Unidad-3-Ej-56.c

2. Búsqueda:

   Búsqueda lineal — Unidad-3-Ej-57.c

   Búsqueda binaria — Unidad-3-Ej-58.c