

Estructuras de Datos Estáticas

Rodolfo Christian Catunta Uturunco (Elemental Bolivia)



28 de febrero de 2025

Contenido

1 Arreglos Unidimensionales

2 Arreglos Bidimensionales

3 Cadenas (*)

4 memset

Contenido

1 Arreglos Unidimensionales

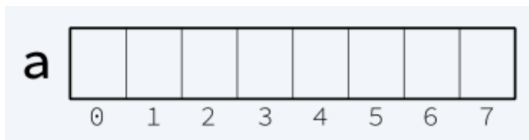
2 Arreglos Bidimensionales

3 Cadenas (*)

4 memset

Arreglos Unidimensionales

Los arreglos son una colección de variables del mismo tipo, accesibles a través de un índice. Llamaremos casilla a cualquiera de las variables individuales que componen un arreglo. Un arreglo se puede ver como



Para crearlos se utiliza:



Declaracion

```
1 int main(){  
2     int enteros[10]; //Declara un arreglo de tamaño 10 de tipo  
    entero  
3     float numeros[4]; //Declara un arreglo de tamaño 4 de tipo  
    flotante  
4     int tam;  
5     cin>>tam;  
6     int datos[tam]; //Declarara un arreglo de tamaño tam de  
    tipo entero  
7 }
```

```
1 int main(){
2     int enteros[10]; //Declara un arreglo de tamaño 10 de tipo
        entero
3     // Al inicio los arreglos comienzan sin valores en sus
        casillas
4     //Para acceder a una casilla hacemos
5     enteros[0] = 8; // Ahora la casilla 0 del arreglo tendrá el
        valor de 8
6     enteros[1] = -7; // Ahora la casilla 1 del arreglo tendrá
        el valor de -7
7     cout<<enteros[0]<<endl;
8 }
```

Impresión y recorrido

```
1 int main(){
2     int N[7]; // Arreglo de tamaño 7
3     //Rellenamos el arreglo, esta es una forma no optima de
        realizar el relleno, tambien toma en cuenta que puedes
        utilizar un for para llenarlo.
4     N[0] = 4;
5     N[1] = 6;
6     N[2] = -1;
7     N[3] = 0;
8     N[4] = 9;
9     N[5] = -14;
10    N[6] = 10;
11    //Recorremos e imprimimos el arreglo
12    for(int i=0;i<7;i++){//El final siempre debe ser el tamaño
        del arreglo
13        cout<<"N["<<i<<" ] = "<<N[i]<<endl;
14    }
15 }
```

Ordenamiento

```
1 #include <iostream>
2 #include <algorithm> // Libreria necesaria para ordenar
3
4 using namespace std;
5
6 int main(){
7     // Creamos el arreglo
8     int arr[10] = {3,5,1,0,8,9,2,6,4,7};
9     // Funcion para ordenar el arreglo
10    sort(arr, arr+10);
11
12    return 0;
13 }
```


Contenido

1 Arreglos Unidimensionales

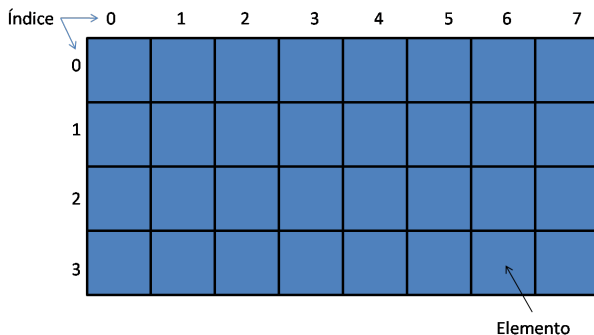
2 Arreglos Bidimensionales

3 Cadenas (*)

4 memset

Arreglos Bidimensionales

Una matriz es un arreglo de 2 dimensiones. Lo cual genera una estructura en forma de tabla como en la imagen.



The diagram shows a 4x8 matrix. The columns are indexed from 0 to 7 at the top. The rows are indexed from 0 to 3 on the left. An arrow labeled 'Índice' points to the row index '0'. Another arrow labeled 'Elemento' points to the cell at row 3, column 6.

Índice	0	1	2	3	4	5	6	7
0								
1								
2								
3								

Una matriz a diferencia de un arreglo tiene 2 índices en lugar de 1, como se muestra en la figura. Cabe también recalcar que cada una de las casillas es un elemento de la matriz.

Arreglos Bidimensionales

Las posiciones de una matriz se acceden primero por la fila y luego por la columna. Las filas son horizontales y las columnas verticales.

	Column 1	Column 2	Column 3	Column 4
Row 1	<code>x[0][0]</code>	<code>x[0][1]</code>	<code>x[0][2]</code>	<code>x[0][3]</code>
Row 2	<code>x[1][0]</code>	<code>x[1][1]</code>	<code>x[1][2]</code>	<code>x[1][3]</code>
Row 3	<code>x[2][0]</code>	<code>x[2][1]</code>	<code>x[2][2]</code>	<code>x[2][3]</code>

Arreglos Multidimensionales

Es posible realizar arreglos de la cantidad de dimensiones que se desee, en la imagen inferior se muestra como se vería una matriz de 3 dimensiones.



Recorrido de una Matriz

```
1 //Llena a la matriz de 0s
2 int main(){
3     int matriz[4][8]; // 4 filas y 8 columnas
4     //Recorremos la matriz
5     for(int i=0;i<4;i++){//Primer ciclo recorre las filas
6         for(int j=0;j<8;j++){//Segundo ciclo recorre las columnas
7             matriz[i][j]=0;
8         }
9     }
10 }
```

Recorrer una matriz desde el teclado

```
1 int main(){
2     int num_filas , num_columnas;
3     cin>>num_filas>>num_columnas; // Pedimos el numero de filas
        y columnas.
4     int matriz[num_filas][num_columnas]; // Creamos la matriz
        de numeros
5
6     // Peticion de los numeros que tendra la matriz
7     for(int i=0;i<num_filas;i++){// Filas
8         for(int j=0;j<num_columnas;j++){// Columnas
9             cin>>matriz[i][j];
10        }
11    }
12 }
```

Contenido

1 Arreglos Unidimensionales

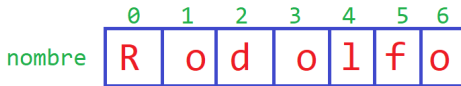
2 Arreglos Bidimensionales

3 Cadenas (*)

4 memset

Cadenas de caracteres

Las cadenas, también conocidas como strings son llamadas así, porque almacenan una cadena de caracteres en su interior.



Un string es similar a un arreglo de caracteres, solo que tiene aún mas características, las cuales serán analizadas en un módulo especializado en Strings.

Recorrido de un string

```
1 int main() {  
2     string curso="CreadoresDigitales2021";  
3     for(int i=0;i<curso.size();i++){  
4         char letra=curso[i];  
5         cout<<letra<<" ";  
6     }  
7 }
```

Código ASCII

En programación resulta de gran utilidad el uso de código ASCII para manipular cadenas. La Tabla ASCII nos muestra todos los caracteres imprimibles que se pueden colocar en un string.

Caracteres ASCII de control			Caracteres ASCII imprimibles				ASCII extendido									
00	NULL	(carácter nulo)	32	espacio	64	@	96	.	128	Ç	160	à	192	Ł	224	Ó
01	SOH	(inicio encabezado)	33	!	65	A	97	a	129	ú	161	í	193	ł	225	ô
02	STX	(inicio texto)	34	"	66	B	98	b	130	ë	162	ô	194	—	226	ö
03	ETX	(fin de texto)	35	#	67	C	99	c	131	â	163	û	195	—	227	õ
04	EOT	(fin transmisión)	36	\$	68	D	100	d	132	ä	164	ñ	196	—	228	ö
05	ENQ	(consulta)	37	%	69	E	101	e	133	å	165	ñ	197	—	229	ö
06	ACK	(reconocimiento)	38	&	70	F	102	f	134	å	166	*	198	—	230	µ
07	BEL	(timbre)	39	'	71	G	103	g	135	ç	167	*	199	—	231	þ
08	BS	(retroceso)	40	(72	H	104	h	136	è	168	¿	200	—	232	ÿ
09	HT	(tab horizontal)	41)	73	I	105	i	137	é	169	¿	201	—	233	ÿ
10	LF	(nueva línea)	42	*	74	J	106	j	138	è	170	¿	202	—	234	ÿ
11	VT	(tab vertical)	43	+	75	K	107	k	139	í	171	¼	203	—	235	ÿ
12	FF	(nueva página)	44	,	76	L	108	l	140	î	172	½	204	—	236	ÿ
13	CR	(retorno de carro)	45	-	77	M	109	m	141	ï	173	¾	205	—	237	ÿ
14	SO	(desplaza afuera)	46	.	78	N	110	n	142	Ä	174	»	206	—	238	ÿ
15	SI	(desplaza adentro)	47	/	79	O	111	o	143	Å	175	»	207	—	239	ÿ
16	DLE	(esc.vínculo datos)	48	0	80	P	112	p	144	É	176	»	208	—	240	ÿ
17	DC1	(control disp. 1)	49	1	81	Q	113	q	145	æ	177	»	209	—	241	ÿ
18	DC2	(control disp. 2)	50	2	82	R	114	r	146	Æ	178	»	210	—	242	ÿ
19	DC3	(control disp. 3)	51	3	83	S	115	s	147	ø	179	»	211	—	243	ÿ
20	DC4	(control disp. 4)	52	4	84	T	116	t	148	ö	180	»	212	—	244	ÿ
21	NAK	(conf. negativa)	53	5	85	U	117	u	149	õ	181	»	213	—	245	ÿ
22	SYN	(inactividad sinc)	54	6	86	V	118	v	150	ù	182	»	214	—	246	ÿ
23	ETB	(fin bloque trans)	55	7	87	W	119	w	151	û	183	»	215	—	247	ÿ
24	CAN	(cancelar)	56	8	88	X	120	x	152	ý	184	»	216	—	248	ÿ
25	EM	(fin del medio)	57	9	89	Y	121	y	153	Û	185	»	217	—	249	ÿ
26	SUB	(sustitución)	58	:	90	Z	122	z	154	Ü	186	»	218	—	250	ÿ
27	ESC	(escape)	59	;	91	[123	{	155	ø	187	»	219	—	251	ÿ
28	FS	(sep. archivos)	60	<	92	\	124		156	£	188	»	220	—	252	ÿ
29	GS	(sep. grupos)	61	=	93]	125	}	157	ø	189	»	221	—	253	ÿ
30	RS	(sep. registros)	62	>	94	^	126	~	158	×	190	»	222	—	254	ÿ
31	US	(sep. unidades)	63	?	95	_			159	f	191	»	223	—	255	nbsp
127	DEL	(suprimir)														

Uso de código ASCII en C++

```
1 int main(){
2     for(int i=0;i<256;i++){
3         char caracter = char(i);
4         int ascii = int(caracter);
5         cout<<"El elemento "<<ascii<<" de la tabla ASCII es "<<
        caracter<<endl;
6     }
7 }
```

Contenido

1 Arreglos Unidimensionales

2 Arreglos Bidimensionales

3 Cadenas (*)

4 **memset**

memset

La función memset es muy útil para inicializar en ciertos valores a arreglos o cadenas. Sin embargo se debe ser precavido en su uso como se muestra en el siguiente ejemplo.

```
1 #include <bits/stdc++.h> // Para garantizar el uso de memset
   en cualquier compilador
2 using namespace std;
3 int main()
4 {
5     // memset(nombre de estructura, valor, tamaño en bytes)
6     int a[5];
7     // todos los elementos de 0
8     memset(a, 0, sizeof(a));
9     // todos los elementos de -1
10    memset(a, -1, sizeof(a));
11    // No funcionara
12    memset(a, 5, sizeof(a)); // INCORRECTO
13 }
```