UNIVERSIDAD DE MURCIA

Fundamentos de Informática

8. Arrays y cadenas:
grandes volúmenes de datos

Fundamentos de Informática
Grado en Ingeniería Química

2

Contenidos

- ***Arrays***
- ***Arrays e iteraciones***
- ***Cadenas de caracteres***

3

Arrays

4

Arrays *(definición)*

Array (lista, tabla, vector, matriz) :
estructura de datos compuesta para almacenar una secuencia de datos del mismo tipo

```
main()
{
      int v[5];

      v [ 0 ] = 0;
      v [ 1 ] = 9;
      v [ 2 ] = 29;
      v [ 3 ] = 17;
      v [ 4 ] = 33;
}
```

Define un array formado por 5 números enteros.

Los arrays en C siempre tienen índices desde 0 hasta n-1

v[0], v[1], … v[5]

Mete los valores en v

V_0	0
V_1	9
V_2	29
V_3	17
V_4	33

Arrays *(acceso a datos)*

Se puede acceder a los datos así

```
int v[10],a,b;

a=4; b=8;

v [ a ] = b;
v [ ++a ]= 4*b+2;
v [ 2*a-b ] = 12;
v [ 6 ] = v [ a-1 ];
v [ a ] = v [ a+1];
b = v [ a ];

if (v [ a ] > v [ a+1])
    v [ a ]=0;
```


Arrays *(control de rango)*

```
#define Tam 5
main()
{
      int a, v [ Tam ];

      v [ 0 ] = 0;
      v [ 1 ] = 9;
      v [ 2 ] = 29;
      v [ 3 ] = 17;
      v [ 4 ] = 33;
}
```

Establecer con un ***define*** la dimensión del array es una buena práctica.

```
#define Tam 5
main()
{
      int v [Tam ],a;
      …
      a=7;
      v [ a ] = 10;
}
```

¡Ojo! C no comprueba indices de arrays fuera de rango. ¡ escribe en otra zona de memoria !

7

Arrays y cadenas

Arrays *(bidimensionales)*

```
#define M 3
#define N 2
main()
{
      int a [ M ] [ N ];

      a[0][0] = 1; a[0][1] = 7;
      a[1][0] = 8; a[1][1] = 8;
      a[2][0] = 2; a[2][1] = 9;
}
```

Esto es un ***array bidimensional*** (una tabla, una matriz)

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn} \end{pmatrix}$$

Se puede usar para guardar desde una matriz matemática hasta una imagen.

8

Arrays y cadenas

Arrays *(3D, 4D)*

```
#define M 10
#define N 12
#define P 5
#define L 20

float a [M] [N] [P], coste;
int b [M] [N] [P] [L], i, j, k;

. . .

a [ i ] [ j ] [ k ] = coste*12.0;
b [7] [10] [9] [0] = 20;
```

Y de 3 dimensiones, y de 4, y de…

9

Arrays *(inicializaciones)*

```
main()
{
      int a[5];
      int b[5] = {2,7,12,0,1};

      . . .
      a[0]=2;   a[1]=7;
      a[2]=12; a[3]=0;
      a[4]=1;
      . . .
}
```

Se puede inicializar un array de esta forma

Pero esta es mejor

10

Arrays *(inicialización 2D)*

```
main()
{
      int a[2][3] = {0,1,2,10,11,12};
      int b[2][3] = { {0,1,2} , {10,11,12} };

      . . .
      sentencias;
      . . .
}
```

Los de dos dimensiones se inicializan de cualquiera de estas dos formas; mejor la segunda.

0	1	2
10	11	12

11

Arrays *(scanf)*

```
#define Tam 5
main()
{
      int  v [Tam ];

      . . .

      printf (“Introduce 5 datos separados por espacios : ”);
      scanf(“%d %d %d %d %d”, &v[0], &v[1],&v[2],&v[3],&v[4]);

}
```

De este modo puedo cargar el array ***v*** de datos del teclado

12

Arrays e iteraciones

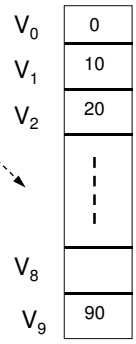
13

Iteraciones (*for* básico)

```
#define Tam 10
main()
{
      int v[Tam];

      for (i=0; i<Tam; i++)
            v [ i ] = i * 10;
}
```

Recorre el array y mete los valores:



14

Iteraciones (*for* anidado 2D)

```
#define M 1000
#define N 2000

main()
{
      int a [ M ] [ N ];

      for (i=0; i< M; i++)
            for (j=0; j< N; j++)
                  a [ i ] [ j ] = i* j* 10;
}
```

Recorre un array 2D

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn} \end{pmatrix}$$

Se puede usar para guardar desde una matriz matemática hasta una imagen.

15

Iteraciones (arrays 3D, 4D…)

```
float a [M] [N] [P];

for (i=0; i< M; i++)
      for (j=0; j< N; j++)
            for (k=0; k< P; k++)
                  a [ i ] [ j ] [ k ] = i* j* 10;
```

Y de 3 dimensiones, y de 4, y de…

```
int v[80][50][100][60];
```

Este array ocupa:

80 x 50 x 100 x 60 x 4 =

96.000.000 bytes =

¡ 96 MB !

sizeof(v);

16

Iteraciones (carga de arrays)

```
#include <stdio.h>
#define Tam 10
main()
{
      int i, v [ Tam ];

      for (i=0; i< Tam; i++)
      {
            printf (“Introduce %d :”, i);
            scanf(“%d”, &v[ i ] );
      }
}
```

Esta iteración carga el array ***v*** de datos de usuario

17

Iteraciones (carga de arrays)

```
#include <stdio.h>
#include <stdlib.h>
#define Tam 1000
main()
{
        int i; float v[ Tam ];

        srand(time(NULL));
        for (i=0; i< Tam; i++)
          v[i]=rand()/(float)RAND_MAX;
}
```

¡ Y ésta de valores aleatorios entre 0 y 1 !

¡ Y ésta un entero entre 0 y 10 !

```
v[i] = rand() % 11;
```

Y ésta un real entre 0 y 10

```
v[i]= (rand()/(float)RAND_MAX)*10;
```

18

Iteraciones (while)

```
for (i=0; i< Tam; i++)
{
        printf ("Introduce %d :", i);
        scanf("%d", &v[ i ] );
}
```

Por supuesto puedo usar **`while`** en vez de **`for`**

```
i=0;
while (i< Tam)
{
        printf ("Introduce %d :", i);
        scanf("%d", &v[ i ] );
        i++
}
```

Y **`do while`**

```
i=0;
do
{
        printf ("Introduce %d :", i);
        scanf("%d", &v[ i ] );
        i++
} while (i< Tam)
```

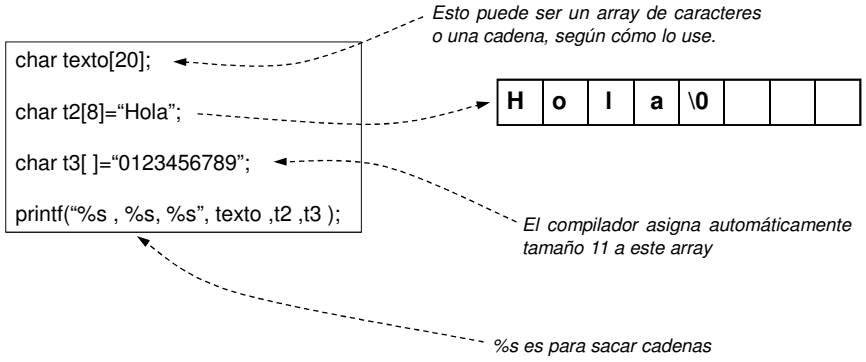
19

Cadenas de caracteres (*strings*)

20

Cadenas *(definición)*

Una cadena es un array de caracteres que acaba en **'\0'**. C lo maneja de forma especial.



21

Cadenas *(asignación de valores)*

*Esto **NO** se puede hacer*

Hay que hacerlo con la función **strcpy***, predefinida en* **string.h**

Para leer de teclado se usa **gets()** *(sin poner &).* **scanf()** *solo lee hasta el primer espacio en blanco.*

```
#include <stdio.h>
#include <string.h>

main()
{
 char texto[20];
 char t2[80]="  ";
 char t3[ ]="0123456789";

 texto="Prueba";
 texto=t2;
 strcpy(texto,"Prueba");
 strcpy(texto,t2);
 gets(t2);

 printf("%s , %s, %s", texto ,t2 ,t3 );
 getchar();
}
```

22

Cadenas *(desbordamiento)*

```
#include <stdio.h>
#include <string.h>

main()
{
 char t1[6];
 char t2[5]="0123456789";
 char t3[3];

 gets(t1);
 strcpy(t3,t2);

 printf("%s , %s, %s", texto ,t2 ,t3 );
 getchar();
}
```

¡Mucho ojo!

C no controla rangos, por lo que si no los controlamos nosotros, se mete en las zonas de memoria de otras variables.

¡el programa no funcionará y no sabremos por qué!

Cadenas *(Acceso a caracteres)*

Este programa lee una cadena completa con ***gets()***, y luego la recorre caracter a caracter hasta encontrar el final.

```
#include <stdio.h>
#include <string.h>
#define MAX 100

main()
{
 char t1[MAX]; int i;

 gets(t1); i=0;
 while ( (t1[ i ] != '\0') && ( i < MAX) )
 {
      printf("%c",t1[i]);
      i++;
 }
 printf("\n"); getchar();
}
```


Algunas funciones para cadenas y caracteres

Funciones para cadenas (stdio.h)

int getchar()	lee un caracter	int car; car=getchar();
gets(char[ ])	lee una cadena	char texto[20]; gets(texto);

putchar(int):	Escribe un caracter	int letra;
		letra=65;
		putchar(letra);

puts(char[ ]);	Escribe una cadena	char texto[20]=“hola”; puts(texto);

Funciones para cadenas (stdlib.h)

double atof(char[ ])	
Convierte una cadena a float	float num; char texto[10] = “23.45"; num=atof(texto); printf(“ %f ”, num);
int atoi(char[ ])	
Convierte una cadena a entero	int num; char texto[10] = “2345"; num=atoi(texto); printf(“ %d ”, num);

27

Cadenas *(string.h)*

int strlen(char[])	int longi; char texto[100];	
Longitud de una cadena	gets(texto);	(“Hola”)
	longi=strlen(texto);	(4)
int strcmp(char[],char[])	char t1[20]=“Hola”, t2[20]=“abcd”;	
Compara dos cadenas	if(strcmp(t1,t2)==0)	
(0 si iguales)	printf(“Iguales”);	
strcat(char[], char[], int)	char t1[30]=“Hola ”;	
Añade a la primera cadena	char t2[10]=“amigo”;	
hasta n caracteres de la segunda	strcat(t1,t2,3);	(Hola ami)
	strcat(t1,t2,30)	(Hola amigo)

28

Cadenas *(string.h)*

strupr(char[])	char texto[10]=“Hola”;	
Pasa una cadena a mayúsculas	strupr(texto);	(HOLA)
strlwr(char[])	char texto[10]=“HOLa”;	
Pasa una cadena a minúsculas	strlwr(texto);	(hola)
char[] strchr(char[], char)	char t1[100]=“Hola P”, t2[100], c = ‘o’;	
Devuelve la cadena a partir de la	printf(“%s”, strchr (t1,c));	(ola P)
primera aparición del caracter	strcpy(t2, strchr(t1,’l’));	(la P)
char[] strstr(char[], char[])	char t1[100]=“ohola P”, t2[10] = “ol”;	
Devuelve la cadena a partir de la	printf(“%s”, strstr (t1,t2));	(ola P)
primera aparición de la segunda cadena		

Cadenas *(ctype.h)*

char tolower(char)	Pasa a minúscula un caracter	letra=tolower(c);
char toupper(char)	Pasa a mayúsculas un caracter	letra=toupper(c);
int isalpha (char)	Devuelve !=0 si es alfabético	if (isalpha (c) != 0) printf(“Alfabetico”);
int isdigit(char)	Devuelve !=0 si es un dígito	if (isdigit (c) != 0) printf(“Digito”);
int islower(char)	Devuelve !=0 si es minúscula	If (islower (c) != 0) printf(“Minus”);

Arrays y cadenas

- - - - - -