Programación I

Arreglos

Prof. Lisbeth Pérez Rivas
lisbethpe@ula.ve

Arreglos

- Secuencia de datos del mismo tipo que pueden ser de cualquier tipo de dato (entero, real, carácter, entre otros). Cada valor es referenciado utilizando uno o mas subíndices (enteros).

Ejemplo:

$$\text{Vector} = \begin{pmatrix} 4 \\ 5 \\ 6 \\ 7 \end{pmatrix}$$

$$\text{Matriz} = \begin{pmatrix} 1 & 3 & 5 \\ 2 & 1 & 4 \\ 0 & 1 & 3 \end{pmatrix}$$

Tipos de Arreglos

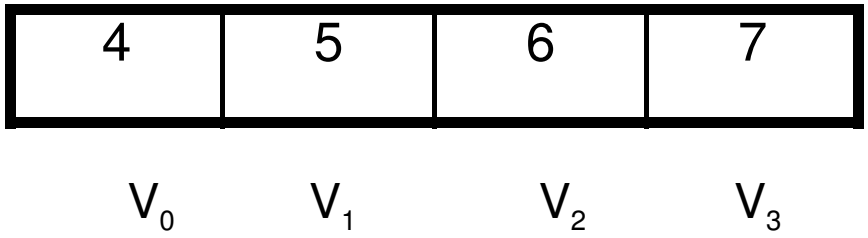
- Según el número de dimensiones
  - Vector (Unidimensionales)
  - Matriz (Bidimensionales)
  - Multidimensional (3 o más dimensiones)

Vectores (Arreglo Unidimensional)

Grupo de localidades consecutivas de memoria relacionadas por el hecho de que tienen el mismo nombre y tipo. Cada celda tiene el mismo tamaño y almacena un elemento del vector.

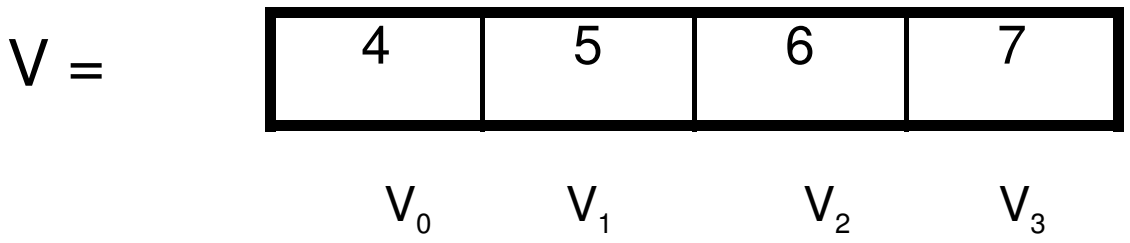
Vector = $\begin{pmatrix} 4 \\ 5 \\ 6 \\ 7 \end{pmatrix}$

V =



Vectores (Arreglo Unidimensional)

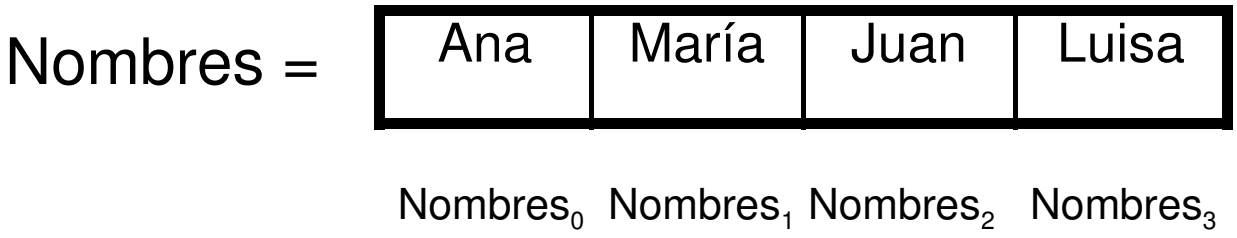
- Cada elemento del vector se puede acceder mediante un subíndice que representa la posición numérica (entero no negativo) de dicho elemento dentro del vector.



Vector de 4 elementos cuyo nombre es V

Subíndices

- El subíndice denota la posición del elemento dentro del vector



Nombres[1] ⟶ María

Declaración de un Vector

Tipo_de_dato nombre_arreglo[numero_de_elementos];

Donde,

Nombre_Arreglo = Nombre del Arreglo.

Tipo_de_dato= Tipo de datos almacenados en el arreglo.

Numero_de_elementos=Valor entero CONSTANTE mayor que cero.

Declaración de un Vector

Notación Algorítmica

entero numeros[4]; vector numeros de 4 posiciones de enteros
Real v1[10]; Vector v1 de 10 posiciones de reales
Carácter v2[6]; Vector v2 de 6 posiciones de caracteres

Declaración de un Vector

Notación en C

int numeros[4]; vector numeros de 4 posiciones de enteros
float v1[10]; Vector v1 de 10 posiciones de reales
char v2[6]; Vector v2 de 6 posiciones de caracteres

Es posible declarar múltiples vectores:

Int x[5],y[10],z[3];
float m[8],i[40];

Acceso a los elementos de un Vector

Nombre_vector[subíndice]

El subíndice o posición numérica de un elemento dentro del vector se expresa como:

- Constante Nombre_vector[3]
- Variable Nombre_vector[x]
- Expresión Nombre_vector[2*i+z]

El valor del subíndice puede variar de *0 a n-1, donde n* es el número de elementos del vector.

En general, para tener acceso al i-ésimo elemento del vector se escribe nombre_vector[i-1].

Acceso a los elementos de un Vector

Int Rango1[10];

Vector de enteros de 10 posiciones

Para acceder a:
- Posición 0 → Rango1[0]
- Posición 4 → Rango1[4]
- Posición 9→ Rango1[9]

Float rango2[10];

Para acceder a:
- Posición 1 → Rango2[1]
- Posición 5 → Rango2[5]
- Posición 9→ Rango2[9]

Instrucciones Válidas

Int rango1[10];

Asignación: Rango1[5]=200;
Lectura: scanf(“%d”,&Rango1[5]);
Escritura: printf(“%d”,Rango1[5]);
Comparación: Rango1[2]!=Rango1[0]
Rango1[2]>4
Otras operaciones

B=Rango1[2*2]; Cada elemento del vector puede utilizarse como una variable cualquiera

Ejemplos con vectores

Leer un vector de 10 números enteros y escribirlos por pantalla.

Análisis E-P-S.

Entrada: Los 10 elementos del vector, por tanto necesitamos un vector de 10 posiciones (numero) de tipo entero.

Proceso. Leer cada uno de los elementos y escribirlos por pantalla.

....

Salida. Los 10 elementos escritos por pantalla.

Ejemplos con vectores

Algoritmo

1. Inicio
2. Para (i=0;i<10;i=i+1)
   2.1 Leer (numeros[i])
   Fin_RP
1. Para (i=0;i<10;i=i+1)
   3.1. Escribir (numeros[i])
   Fin_RP
1. Fin

Código

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

        for (i=0;i<10;i++)
                scanf("%d",&numeros[i]);

        for (i=0;i<10;i++)
                printf("\n%d",numeros[i]);

        return 0;
}
```

Inicializar arreglos

Método 1: Recorrer los elementos del vector y asignar el valor deseado

Algoritmo

1. Inicio
2. Para (i=0;i<10;i=i+1)
   2.1 datos[i]=0
   Fin_RP
3. Fin

Código

```
#include<stdio.h>
int main(){
        int i,datos[10];
        /*Vector inicializado en 0*/
        for (i=0;i<10;i++)
                datos[i]=0;
        return 0;
}
```

Inicializar arreglos

Método 2: Inicializar directamente en la declaración

```
#include<stdio.h>
int main(){
        Int datos[10]={0};

        return 0;
}
```

```
#include<stdio.h>
int main(){
        Int datos[10]={5,10,6,1,2,3,4,4,9,100};
        Float x[5]={4.5,3.2,5.6,100.3,6.5};
        Char z[5]={'m','a','r','i','a'};
        return 0;
}
```

Ordenar los elementos de un vector (De menor a mayor)

Leer un vector de 10 números reales, ordenar el vector e imprimirlo por pantalla.

Análisis E-P-S.

Entrada: Los 10 elementos del vector, por tanto necesitamos un vector de 10 posiciones de tipo real.

Proceso: Comparar cada una de las posiciones del vector con las posiciones siguientes y verificar que si la posición inicial es mayor que la comparada se deben intercambiar los valores.

Ejemplo. Comparamos la posicion 0 del vector con la posicion 1, si posicion0>posicion 1, intercambiamos los datos. Luego comparamos la posición 0 y la 2 y evaluamos el mismo criterio.

Salida: Los 10 elementos ordenados escritos por pantalla.

Ordenar los elementos de un vector (De menor a mayor)

Algoritmo

```
1.  Inicio
2.  Para (i=0;i<10;i=i+1)
        2.1   Leer (numeros[i])
     Fin_RP
1.  Para (i=0;i<10;i=i+1)
      3.1. Para ( j=i+1;j<10;j=j+1)
              3.2  Si numeros[i]>numeros[j] entonces
                      3.2.1 aux=numeros[i]
                      3.2.2 numeros[i]=numeros[j]
                      3.2.3 numeros[j]=aux
                   Fin_si
             Fin_RP
      Fin_RP

1.  Para (i=0;i<10;i=i+1)
          4.1 Escribir(numeros[i])
    Fin_RP
1.  Fin
```

Código

```
#include<stdio.h>
int main(){
        float numeros[10];
        int i,j,aux;



        for (i=0;i<10;i++)
                scanf("%f",&numeros[i]);

        for (i=0;i<10;i++)
                for (j=i+1;j<10;j++)
                        if (numeros[i]>numeros[j]){
                                aux=numeros[i];
                                numeros[i]=numeros[j];
                                numeros[j]=aux;
                        }
        for (i=0;i<10;i++)
                printf("\t%f",numeros[i]);

        return 0;
}
```

Ejemplo

- Diseñar un programa que lea un vector de caracteres y lo escriba al revés.
  - ACDFE → EFDCA

Análisis E-P-S.

Entrada: vectores de caracteres.

Proceso: Intercambiar las posiciones del vector de caracteres.

Salida: La cadena invertida.

Ejemplo

Algoritmo

1. Inicio
2. Para (i=0;i<5;i=i+1)
   2.1 Leer (cadena[i])
   Fin_RP
   Para (i=0;i<2;i=i+1)
   3.1. aux=cadena[i]
   3.2 cadena[i]=cadena[4-i]
   3.3 cadena[4-i]=aux
   Fin_RP

   Para (i=0;i<5;i=i+1)
   4.1 Escribir(cadena[i])
   Fin_RP
   Fin

Código

```
#include<stdio.h>
int main(){
        char cadena[5];
        int i,aux;


        for (i=0;i<5;i++)
                scanf("%c",&cadena[i]);

        for (i=0;i<2;i++){
                                aux=cadena[i];
                                cadena[i]=cadena[4-i];
                                cadena[4-i]=aux;
                        }
        for (i=0;i<5;i++)
                printf("\t%c",cadena[i]);

        return 0;
}
```

Ejercicios

- Escribir un programa que genere un vector con los valores que resultan de la ecuación

  Y= 2*sin(0.5t)

  Donde t varía entre 0 y 20 con un incremento de t de 0.5.
- De un vector de números enteros de 20 posiciones, determinar el máximo, minimo y promedio de los elementos.
- Leer una línea de texto en minúsculas y convertirla a mayúsculas.
- Determinar si una palabra es palíndromo (se escribe igual al derecho y al revés.

Matrices (Arreglos Bidimensionales)

Grupo de localidades consecutivas de memoria relacionadas por el hecho de que tienen el mismo nombre y tipo. Cada celda tiene el mismo tamaño y almacena un elemento de la matriz.

$$M1 = \begin{pmatrix} 4 & 3 & 1 \\ 5 & 4 & 8 \\ 6 & 4 & 1 \end{pmatrix}$$

M1=

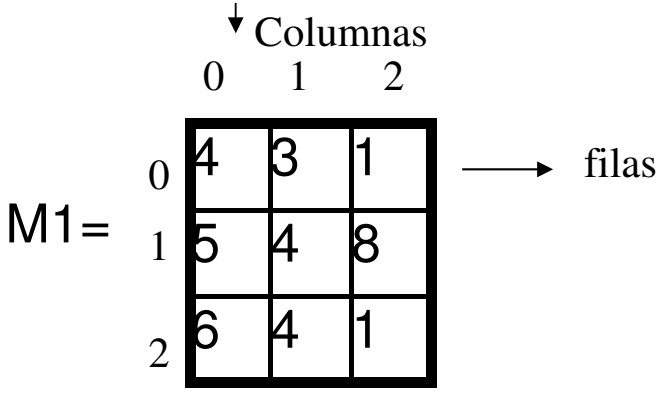
4	3	1
5	4	8
6	4	1

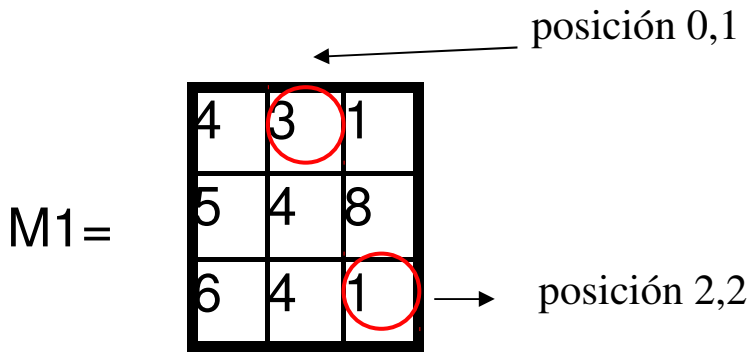
$$M2 = \begin{pmatrix} a & c & d \\ r & x & e \\ x & t & w \end{pmatrix}$$

M2=

a	c	d
r	x	e
x	t	w

Matrices (Arreglos Bidimensionales)





Matrices (arreglos Bidimensionales)

- Cada elemento del vector se puede acceder mediante dos subíndices correspondientes a la fila y la columna del elemento.

M =

$M_{0,0}$	$M_{0,1}$	$M_{0,2}$
$M_{1,0}$	$M_{1,1}$	$M_{1,2}$
$M_{2,0}$	$M_{2,1}$	$M_{2,2}$

Para acceder al elemento de la segunda fila y tercera columna ($M_{2,1}$), escribimos:

M[2][1] → Posición [fila] [columna] entre corchetes

↓ Nombre de la matriz

Subíndices

- El subíndice denota la posición del elemento dentro de la matriz.

MatrizN =

1	3	2	0
4	7	9	1
11	34	7	3
6	8	0	10

MatrizN[1][1]→ elemento de la fila 1 columna 1 → 7

MatrizN[1][3]→ elemento de la fila 1 columna 3 → 1

MatrizN[3][1]→ elemento de la fila 3 columna 1 → 8

MatrizN[3][2]→ elemento de la fila 3 columna 2 → 0

MatrizN[3][3]→ elemento de la fila 3 columna 3 → 10

Declaración de una matriz

Filas Columnas

tipo_dato nombre_matriz[n_filas][n_columnas];

Donde,

tipo_dato= tipo de dato de la matriz.

Nombre_matriz = Nombre de la matriz.

n_filas,n_columnas= Numero entero CONSTANTE de filas y columnas que tiene la matriz

Declaración de una matriz

Int matriz[4][4]; → matriz 4x4 de números enteros.

Float M1[10][4]; → matriz 10x4 de números reales.

Char M2[6][3]; → matriz 6x3 de caracteres.

Aspectos Importantes

(Se cumplen los mismos que para vectores)

- Los indices de una matriz no pueden contener variables.
  - Int X[N][2]; Erróneo a menos que N sea una constante previamente declarada.
- Los subíndices son manejados desde el elemento 0 hasta el número de elementos -1 y deben respetar los rangos de la matriz.

Acceso a los elementos de una Matriz

Nombre_matriz[subíndice_fila][subindice_columna]

El subíndice o posición numérica de un elemento dentro de la matriz se expresa como:

- Constante Nombre_matriz[3][5]
- Variable Nombre_matriz[n][i]
- Expresión Nombre_matriz[2*i][j+1]

Instrucciones Válidas

Int rango1[10][5];

Asignación: Rango1[5][1]=200;
Lectura: scanf(“%d”,&Rango1[5][1]);
Escritura: printf(“%d”,Rango1[5][1]);
Comparación: Rango1[2][3]!=Rango1[1][4]
Rango1[2][4]>4
Otras operaciones

B=Rango1[5*2-1][2+1]; Cada elemento de la matriz puede utilizarse como una variable cualquiera

Inicializar matrices

```
int m1[3][3]={1,1,2,4,5,6,0,6,3};

char m2[2][3]={'a','b','d','z','t','h'};

char m3[2][3]={"abc","gef"};
```

Inicializar matrices

Recorriendo todos los elementos

```
int m1[2][3];

for (i=0;i<2;i++)
        for (j=0;j<3;j++)
                m1[i][j]=0;
```

Ejemplos con matrices

Leer una matriz real de 2 filas y 3 columnas y escribir cada componente por pantalla.

Análisis E-P-S.

Entrada: Los 6 elementos de la matriz, por tanto necesitamos una matriz de 2 filas y 3 columnas de tipo real.

Proceso. Leer cada uno de los elementos y escribirlos por pantalla. Necesitamos dos estructuras de repetición anidadas para la lectura y lo mismo para la escritura

Salida. Los 6 elementos escritos por pantalla.

Ejemplos con matrices

Algoritmo

1. Inicio
2. Para (i=0;i<2;i++)
   2.1 Para (j=0;j<3;j++)
   2.1.1 Leer (numeros[i][j])
   Fin_RP
   Fin_RP
1. Para (i=0;i<2;i++)
   3.1 Para (j=0;j<3;j++)
   3.1.1 Escribir (numeros[i][j])
   Fin_RP
   Fin_RP
   Fin

Código

```
#include<stdio.h>
int main(){
        int numeros[2][3],i,j;

        for (i=0;i<2;i++)
                for (j=0;j<3;j++)
                        scanf("%d",&numeros[i][j]);

        for (i=0;i<2;i++)
                for (j=0;j<3;j++)
                        printf("%d",numeros[i][j]);

        return 0;

}
```

Ejemplos con matrices

Código 2

```
#include<stdio.h>
int main(){
        int numeros[2][3],i,j;

        for (i=0;i<2;i++)
                for (j=0;j<3;j++)
                        scanf("%d",&numeros[i]
        [j]);

        for (i=0;i<2;i++){
                printf(" \n");
                for (j=0;j<3;j++)
                        printf("%d\t",numeros[i]
        [j]);
        }
        return 0;
}
```

Ejemplos con matrices

Leer una matriz entera de 2 filas y 3 columnas. Calcular la suma de los elementos de cada fila y la suma de los elementos de cada columna y almacenar los resultados en 2 vectores.

Matriz

E0,0	E0,1	E0,2
E1,0	E1,1	E1,2

Vector suma_fila

E0,0+E0,1+E0,2
E1,0+E1,1+E1,2

Vector suma_columna

E0,0+E1,0	E0,1+E1,1	E0,2+E1,2

Análisis E-P-S.

Entrada: Los 6 elementos de la matriz, por tanto necesitamos una matriz de 2 filas y 3 columnas de tipo entero.

Proceso. Leer cada uno de los elementos. Necesitamos dos estructuras de repetición anidadas para la lectura. Luego dos estructuras de repetición anidadas para la suma de las filas y 2 más para la suma de las columnas

Salida. Los dos vectores de sumas.

Ejemplos con matrices

Cada semana el gerente de una tienda local de artefactos domésticos registra las ventas de los artículos individuales que hay en existencia. A final del mes, estos resúmenes semana se envían a la oficina central, donde se analizan. Un ejemplo de un mes típico se muestra en la siguiente tabla.

Semana	Artefactos: Lavadoras	Secadoras	Cocinas
1	1	2	3
2	12	6	8
3	10	12	14
4	6	8	6

Calcular: Número total de artefactos vendidos por semana.
Número total de cada artefacto vendido por mes.

Ejercicios

1. Hacer un programa que lea dos matrices y las sume.
2. Hacer un programa que lea dos matrices y las multiplique.
3. Hacer un programa que calcule la transpuesta de una matriz.
4. Considere la siguiente lista de estados y sus capitales:

   Mérida Mérida
   Anzoategui Barcelona
   Monagas Maturín
   Nueva Esparta La Asunción
   Zulia Maracaibo
   Carabobo Valencia

   Escribir un programa interactivo que acepte el nombre de un estado como entrada y escriba su capital. Diseñar el programa de modo que se ejecute repetidamente, hasta que se introduzca la palabra FIN.