

Arreglos

# Arreglos

- Un computador no sirve para sacar promedios de 3 notas.
  - ¡Pero si de 10.000!
  - ¿Definir 10.000 variables?
  - ¿Definir funciones con 10.000 parámetros?

# Arreglos

```
void main() {  
    int nota1, nota2, nota3,.....nota10000;  
}  
  
double promedio(int n1, int n2, ...,n10000);
```

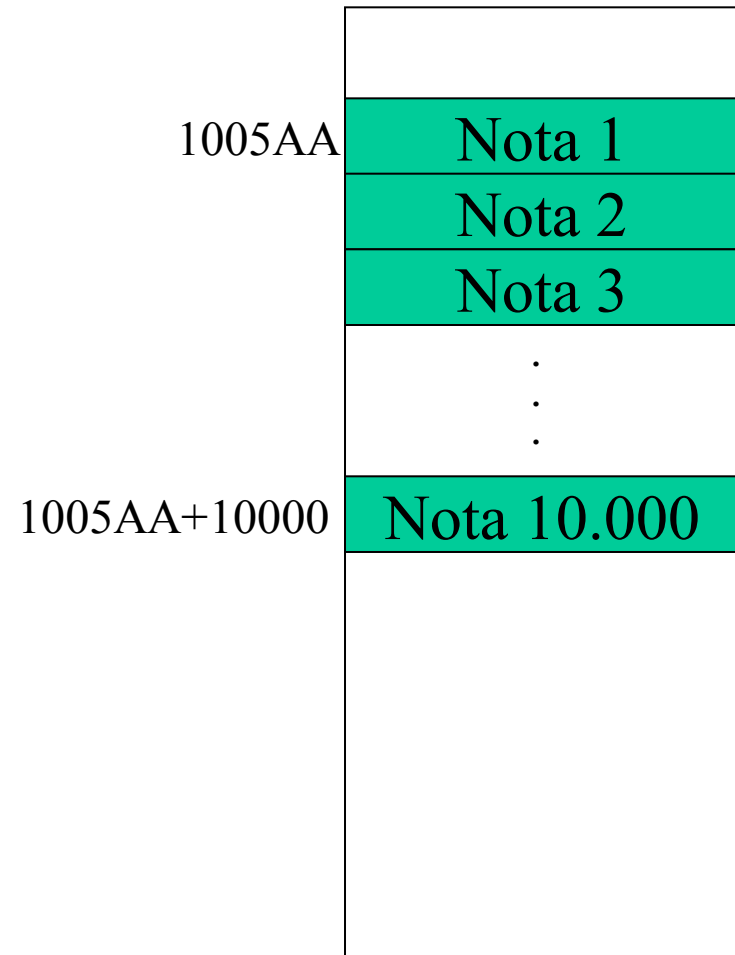
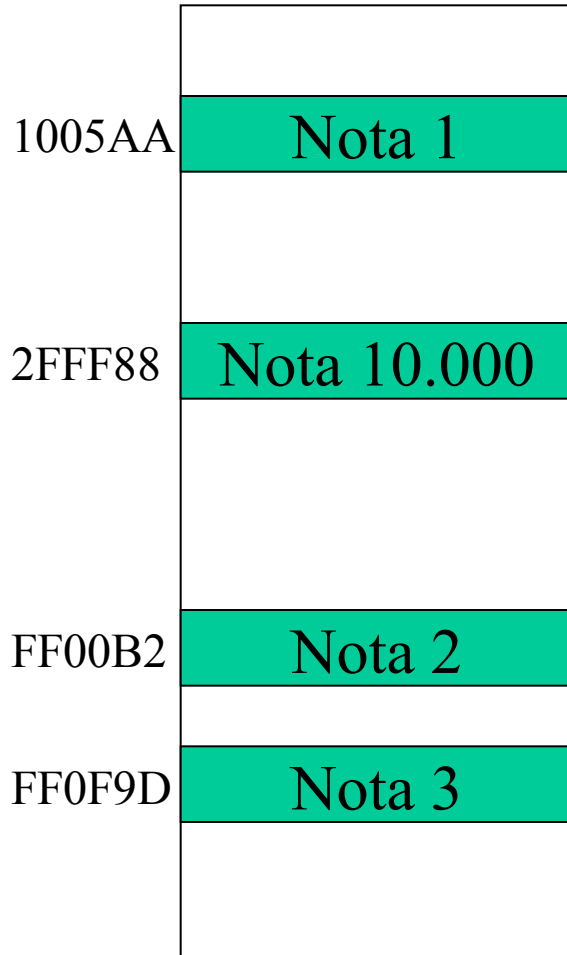
# Arreglos

- ¿Como decirle al computador que nota1..nota10000 son todas notas?
- ¿Cómo acceder a todas con un ciclo for?
- Se hace necesario especificar “grupos” de variables
  - Todas del mismo tipo.
  - Todas con un nombre parecido.
  - Definidas con una secuencia especifica.

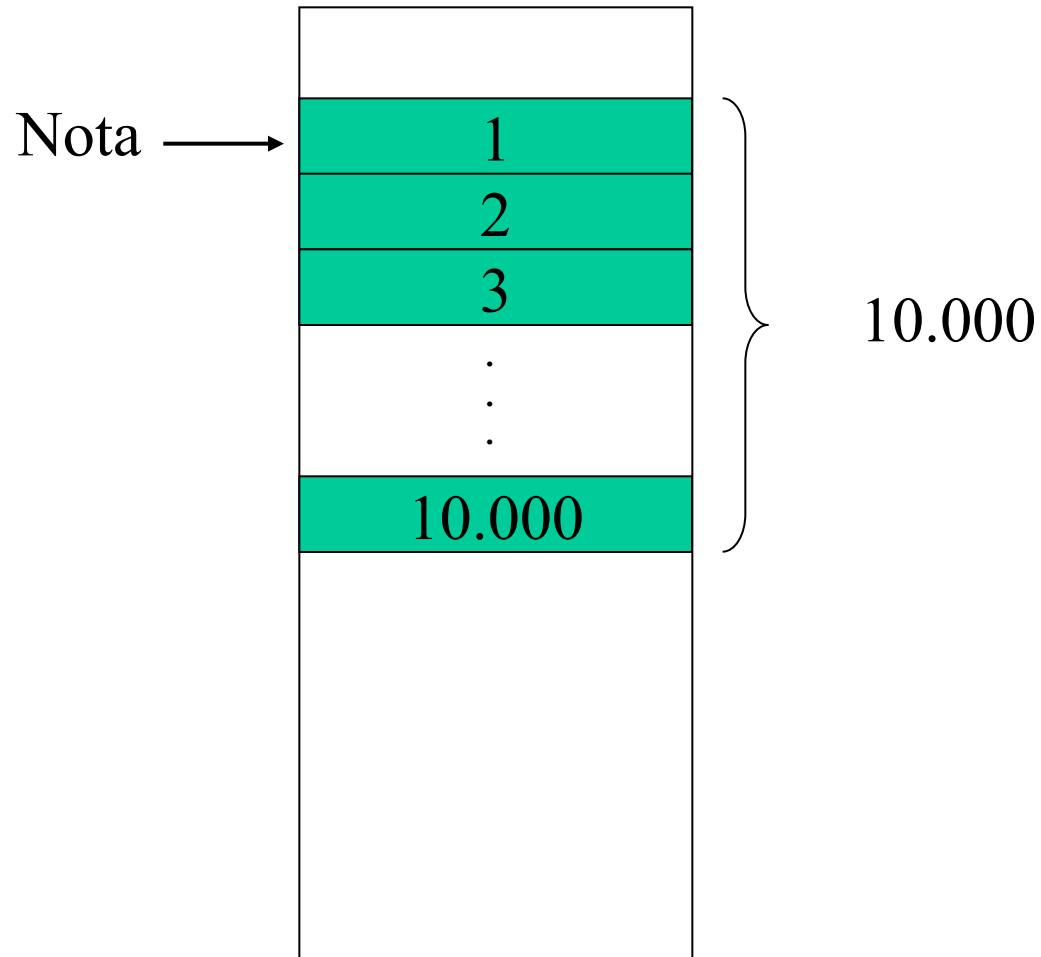
# Arreglos

- Un arreglo es:
  - Una variable que define un “grupo” de variables.
  - Define un nombre
  - Define un tipo
  - Define el tamaño del “grupo”.
  - Las variables dentro del “grupo” estan ordenadas

# Arreglos



# Arreglos



# Definición de arreglos


- Se definen como variables comunes y corrientes
  - Pueden ser locales, globales y parámetros.
  - Tienen un nombre que debe seguir las restricciones de toda variable.
  - El nombre debe ser único dentro del contexto
  - Tienen un tipo específico.
- Además
  - Definen un tamaño.
  - Definen una forma de acceder a cada una de las variables del “grupo”.



# Definición de arreglos

<tipo> <nombre> [**<tamaño>**] ;

Cualquier  
tipo válido



Cualquier  
nombre válido

Corchetes  
obligatorios

Constante que  
especifica el  
tamaño

# Definición de arreglos

```
#define TAMAGNO 32
```

```
...
```

```
int nota1, nota2, ..., nota10000;
```

```
int notas[10000];
```

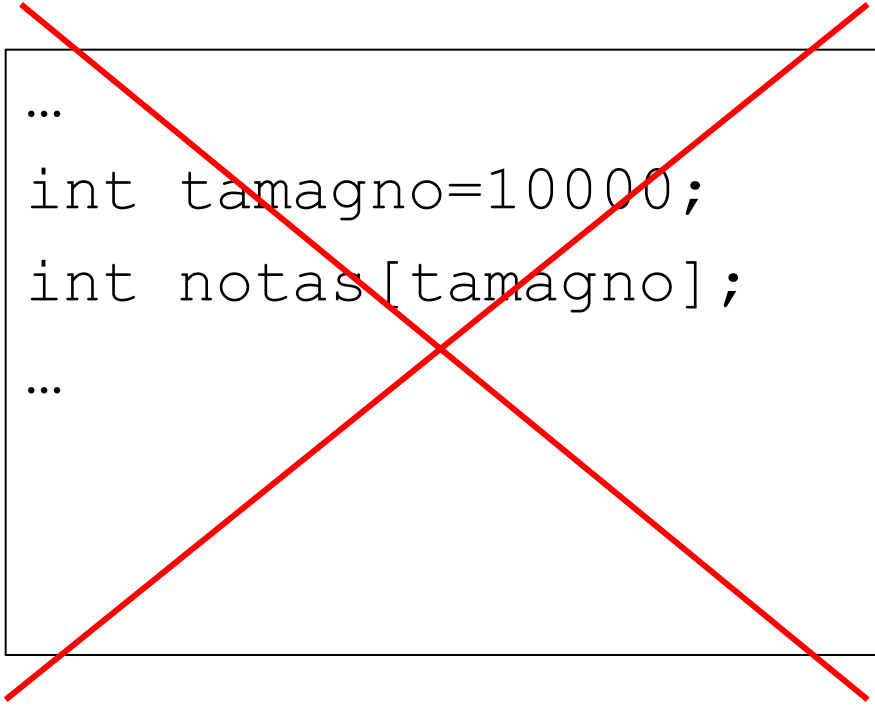
```
Float otro[TAMAGNO];
```

```
Char muchos[80];
```

# Definición de arreglos

- El tamaño debe ser constante
- No se pueden utilizar variables para especificar el tamaño.
- Se suelen utilizar constantes de preprocesador.

# Definición de arreglos



```
...  
int tamagno=10000;  
int notas[tamagno];  
...
```

```
#define tamagno 10000  
...  
int notas[tamagno];  
...
```

# Acceso a arreglos

- Como los arreglos definen “grupos”, es importante poder acceder a los “integrantes” del arreglo.
- Cada “integrante” se representa por un índice secuencial. Se les llama elementos.
- El índice varia entre 0 y (tamaño-1).
- Cada elemento es una variable común y silvestre.
- Para acceder a una de estas variables (lectura y escritura) se utiliza la notación “[<índice>]”

# Acceso a arreglos

`<arreglo>[<indice>] = <sentencia>;`

Modificación el elemento correspondiente al índice

`<variable> = <arreglo>[<indice>;`

Recuperación el elemento correspondiente al índice

# Acceso a arreglos

```
...  
int notas[10000];  
notas[0]=0;  
printf("%d\n", notas[0]);  
notas[1]=notas[0]+1;  
printf("%d\n", notas[1]);  
...  
notas[10000]=notas[9999]+1;  
printf("%d\n", notas[10000]);  
...
```

Primer elemento del arreglo "notas"

Recupera el valor del primer elemento

Modifica el valor de la segunda variable

# Ejemplo 1

```
int main() {  
  
    int numeros[10];  
    int i;  
  
    numeros[0]=0;  
    for(i=1;i<10;i++)  
        numeros[i]=numeros[i-1]+1;  
  
    for(i=0;i<10;i++)  
        printf("%d\n",numeros[i]);  
  
    return 1;  
}
```

0  
1  
2  
3  
4  
5  
6  
7  
8  
9



# Ejemplo 2

```
int main() {  
  
    int numeros[10];  
    int i;  
  
    for(i=0;i<10;i++)  
        scanf("%d", &numeros[i]);  
  
    for(i=0;i<10;i++)  
        printf("%d\n", numeros[i]);  
  
    return 1;  
}
```

# Ejemplo 3

```
#include <math.h>
#define N 4

int main() {

    int numeros[N];
    int i;
    double sum;
    for(i=0; i<N; i++)
        scanf("%d", &numeros[i]);

    sum=0;
    for(i=0; i<N; i++) {
        sum = sum + numeros[i];
    }
    return 1;
}
```

# Ejemplo 4

```
#include <math.h>
#define N 4

int main(){

    int numeros[N];
    int i;
    double sum, parit, pgeom;
    sum=0;
    pgeom=1;
    for(i=0;i<N;i++){
        sum = sum + numeros[i];
        pgeom = pgeom * numeros[i];
    }
    parit = 1.0*sum / N;
    pgeom = pow(pgeom, 1.0/N);

    return 1;
}
```

# Ejemplo 5

```
#include <math.h>
#define N 4

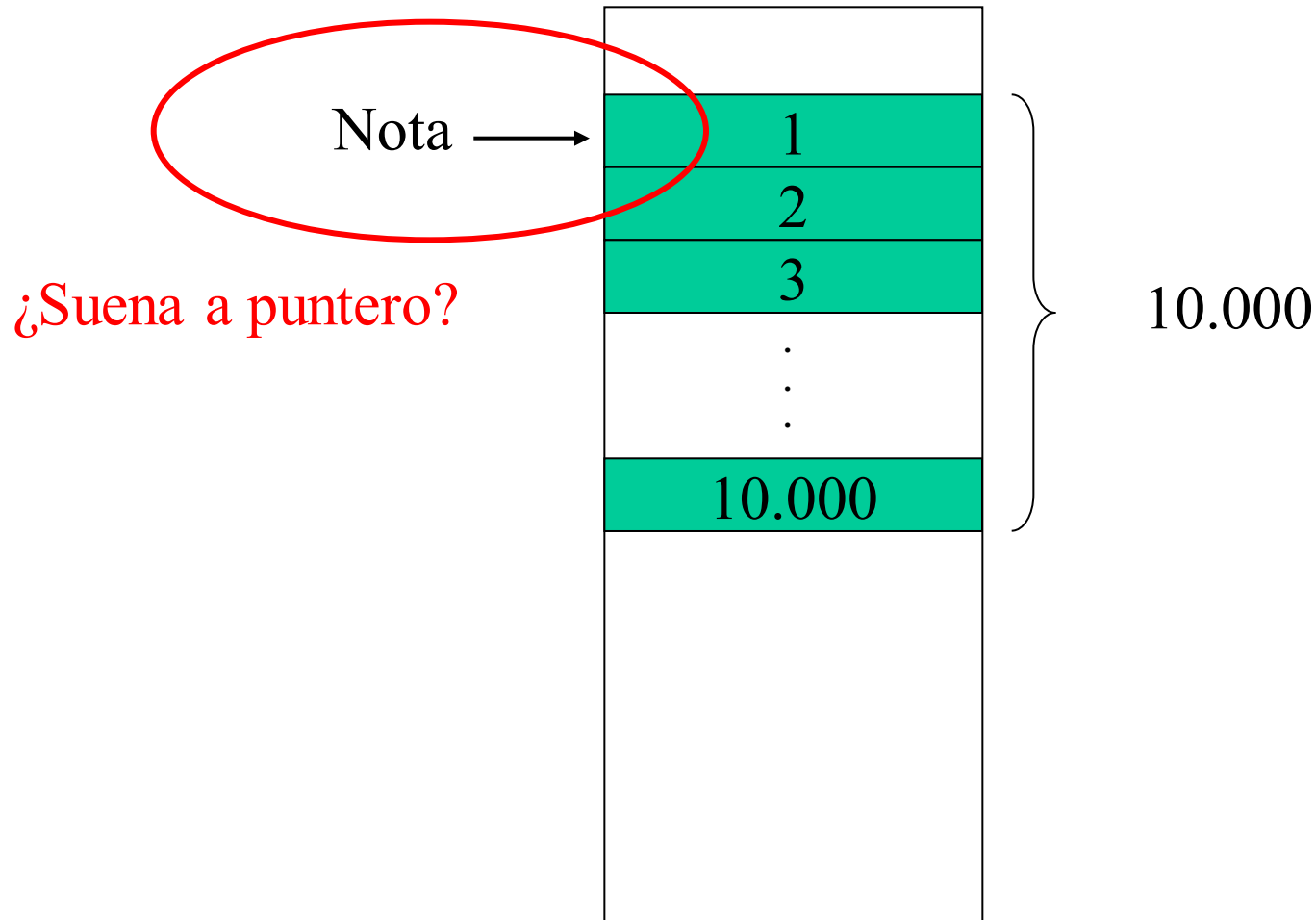
int main() {
    int numeros[N];
    int i;
    int maximo, minimo;

    minimo=1000;
    maximo=0;
    for(i=0;i<N;i++) {
        if(numeros[i]<minimo)
            minimo=numeros[i];
        if(numeros[i]>maximo)
            maximo=numeros[i];
    }
    return 1;
}
```

# Acceso a arreglos

- Solo se puede acceder a los índices entre el cero y tamaño-1.
- Sin embargo, C no realiza un chequeo de acceso a índices inválidos.
- ¿Qué pasa si accedemos a un índice inválido?
- Veremos el análisis de arreglos como punteros.

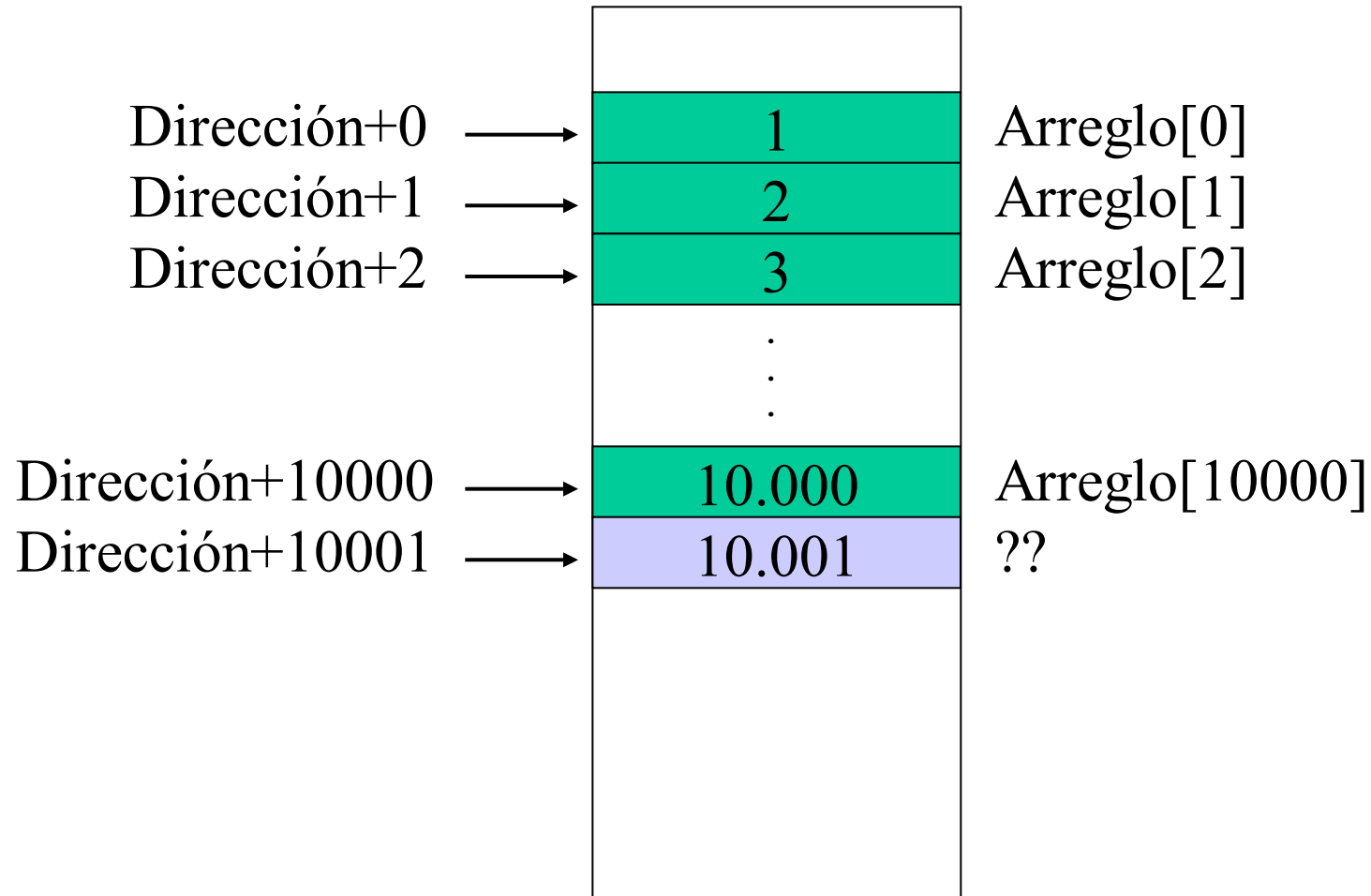
# Arreglos como punteros



# Arreglos como punteros

- Los arreglos son grupos de variables asignadas en zonas contiguas de memoria.
  - Una después de la otra.
- Define un area mayor de memoria
- ¿Como se definen intervalos?
  - a) Definir el inicio y el término
  - b) Definir el inicio y el largo
- Los arreglos se definen como un puntero al inicio del área de memoria y el tamaño que esta abarca.

# Arreglos como punteros





# Arreglos como punteros

- Entonces, los arreglos son punteros
- ¿Qué pasa si accedemos a un índice inválido?
  - Estamos accediendo a una zona de memoria que puede estar asignada para otro fin.
  - Puede resultar en un error grave y el término del programa.

# Ejemplo 1

```
int main() {  
  
    int numeros[10];  
    int i;  
  
    numeros[0]=0;  
    for(i=1;i<10;i++)  
        numeros[i]=numeros[i-1]+1;  
  
    for(i=0;i<10;i++)  
        printf("%d\n",numeros[i]);  
  
    return 1;  
}
```

0  
1  
2  
3  
4  
5  
6  
7  
8  
9

# Ejemplo 2

```
int main() {  
  
    int numeros[10];  
    int i;  
  
    for(i=0;i<10;i++)  
        scanf("%d", &numeros[i]);  
  
    for(i=0;i<10;i++)  
        printf("%d\n", numeros[i]);  
  
    return 1;  
}
```

# Ejemplo 3

```
int main() {  
  
    int numeros[10];  
    int i;  
    int*p;  
  
    for (p=numeros;p<(numeros+10);p++)  
        scanf("%d",p);  
  
    for(i=0;i<10;i++)  
        printf("%d\n",numeros[i]);  
  
    return 1;  
}
```

# Ejemplo 4

```
int main() {  
  
    int numeros[10];  
    int i;  
    int*p;  
  
    numeros[0]=0;  
    for(i=1;i<10;i++)  
        numeros[i]=numeros[i-1]+1;  
  
    for(p=numeros;p<(numeros+10);p++)  
        printf("%d\n", *p);  
  
    return 1;  
}
```

# Arreglos de caracteres

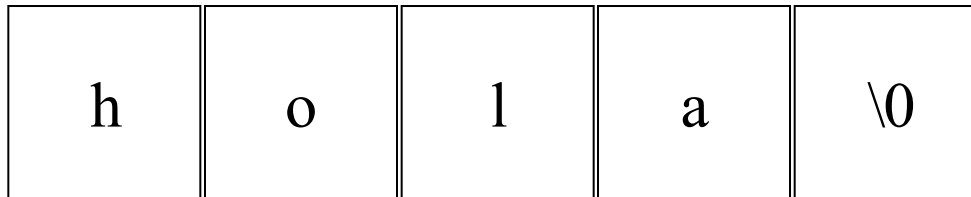
- Las palabras se pueden ver como “grupos” de caracteres en una secuencia.
  - ¡Si definimos el tipo como “char”, un arreglo es un palabra!
- Existen manejos específicos para arreglos de caracteres.
- Los arreglos de caracteres se denominan “**cadena de caracteres**” o “**strings**”.
- El carácter ‘\0’ (representado por el número cero) indica el termino de la palabra.

# Arreglos de caracteres

- Las cadenas de caracteres tienen un largo máximo.
- El tamaño del arreglo limita el largo de las palabras/frases.
- Cuando no se sabe el largo maximo se usan punteros a char
  - `char*`
  - hay que asignar memoria en tiempo de ejecución.
  - En otra ocasión...

# Arreglos de caracteres

“hola”



```
char palabra[5]  
char[0]='h';  
char[1]='o';  
char[2]='l';  
char[3]='a';  
char[4]='\0';
```



# Arreglos de caracteres

```
int main() {  
  
    int i;  
    char palabra[32];  
  
    for(i=0;i<5;i++)  
        palabra[i]=65+i;  
  
    palabra[4]=0;  
  
    for(i=0;i<5;i++)  
        printf("%c",palabra[i]);  
    printf("\n");  
    printf("%s\n",palabra);  
    return 1;  
}
```

# Arreglos de caracteres

```
int main() {  
  
    int i;  
    char palabra[32];  
  
    for(i=0; i<10; i++)  
        palabra[i]=65+i;  
    printf("%s\n", palabra);  
  
    palabra[4]=0;  
    printf("%s\n", palabra);  
    return 1;  
}
```

ABCDEFGHIJ ABCD
--------------------

# Funciones de cadenas

- Definidas en <string.h>

strcpy(c1,c2)	Copia c1 en c2
strcat(c1,c2)	Concatena c2 al final de c1
strlen(c1)	Cálcula el largo de c1
strcmp(c1,c2)	Compara c1 con c2
strchr(c1,char)	Encuentra char dentro de c1
strstr(c1,c2)	Encuentra c2 dentro de c1

# Funciones de cadenas

```
int main(){

    int i;
    char palabra1[32], palabra2[32];

    scanf("%s",palabra1);
    scanf("%s",palabra2);

    printf("%s vs %s\n",palabra1,palabra2);
    printf("Iguales? %s\n", (strcmp(palabra1,palabra2)==0?"si":"no"));
    printf("Largos: %d y %d\n", strlen(palabra1), strlen(palabra2));
    printf("Concatenacion: %s\n", strcat(palabra1, palabra2));
    return 1;
}
```

# Arreglos multidimensionales

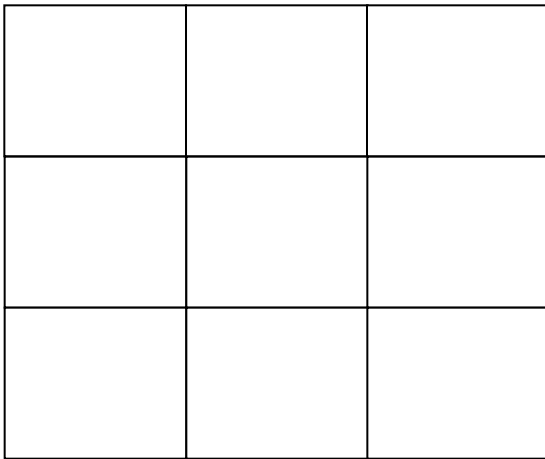
- Un arreglo de tamaño  $n$  puede verse como una matriz  $1 \times n$ .
- ¿Cómo definir matrices de  $m \times n$ ?
- ¿Cómo definir matrices de  $r \times m \times n$ ?
- ....

# Arreglos multidimensionales



1x3

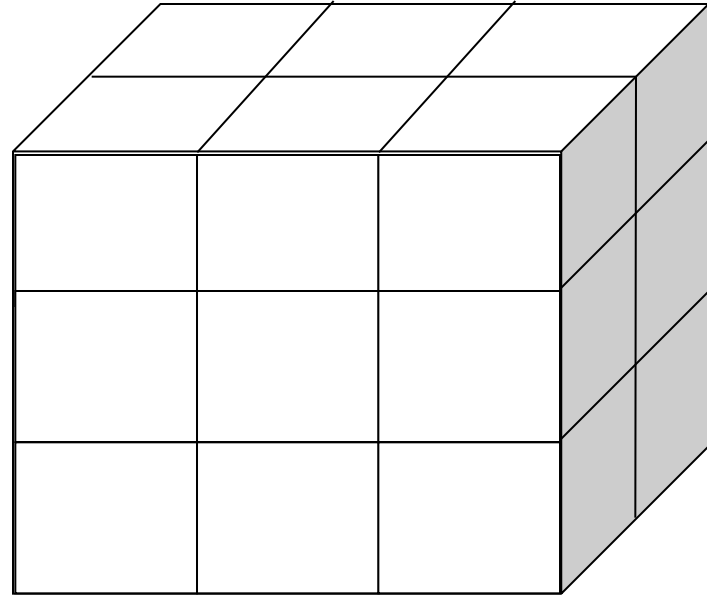
Arreglo de tres variables



3x3

Arreglo de 9 variables

¿O arreglo de 3 arreglos de 3 variables?



3x3x2

Arreglo de 18 variables

¿O arreglo de 2 arreglos de 3  
arreglos de 3 variables?

# Arreglos multidimensionales

- Las dimensiones en los arreglos se agregan con mas pares de corchetes.
- Nos limitaremos a ejemplos de matrices bidimensionales

# Arreglos multidimensionales

A[ fila ][ columna ]

Fila del arreglo



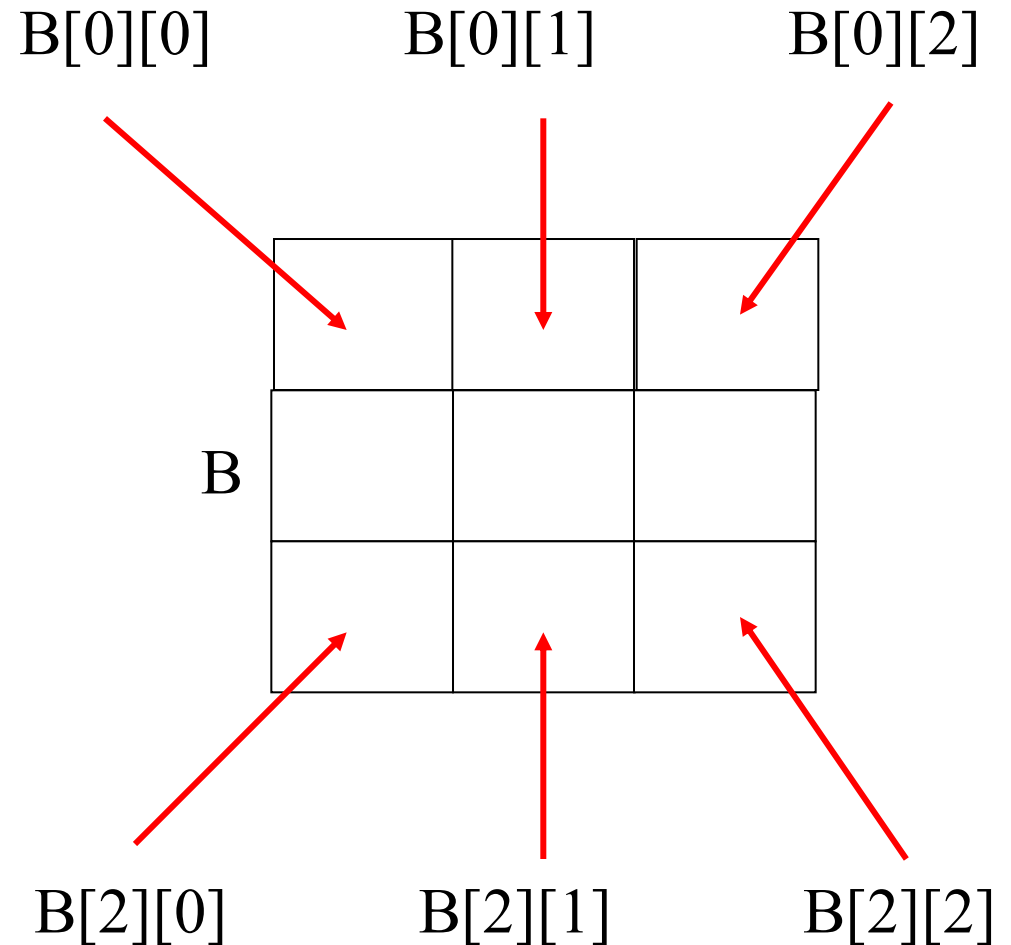
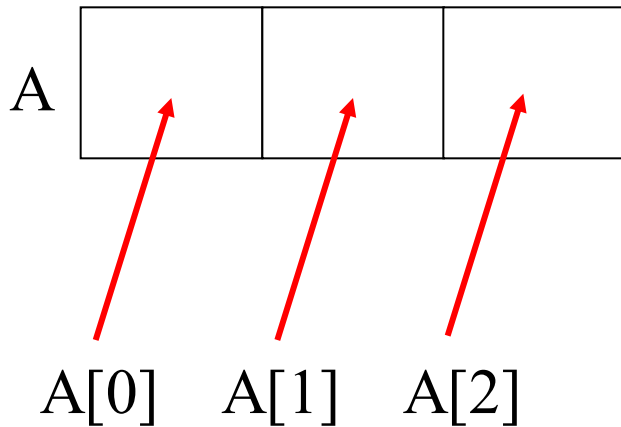
Columna del arreglo



B[ fila ][ columna ][ prof ]



# Arreglos multidimensionales



# Ejemplo 1

```
#include <stdio.h>
#define N 5

int main() {

    int B[N][N];
    int i, j;
    for(i=0; i<N; i++)
        for(j=0; j<N; j++)
            B[i][j]=0;
    return 1;
}
```

# Ejemplo 2

```
#include <stdio.h>

#define N 5

int main(){

    int maiz[N][N];
    for(i=0;i<N;i++){
        for(j=0;j<N;j++){
            printf("%10d ",maiz[i][j]);
            printf("\n");
        }

    return 1;
}
```

# Ejemplo 3

```
#include <stdio.h>

#define N 5

int main(){

    int maiz[N][N];
    int i, j;
    int cantidad=1;
    for(i=0;i<N;i++)
        for(j=0;j<N;j++){
            maiz[i][j]=cantidad;
            cantidad*=2;
        }
    ...

    return 1;
}
```

# Otros ejemplos

- Transponer una matriz
- ¿Es simétrica?
- ¿Es diagonal?
- Determinante
- Etc...