

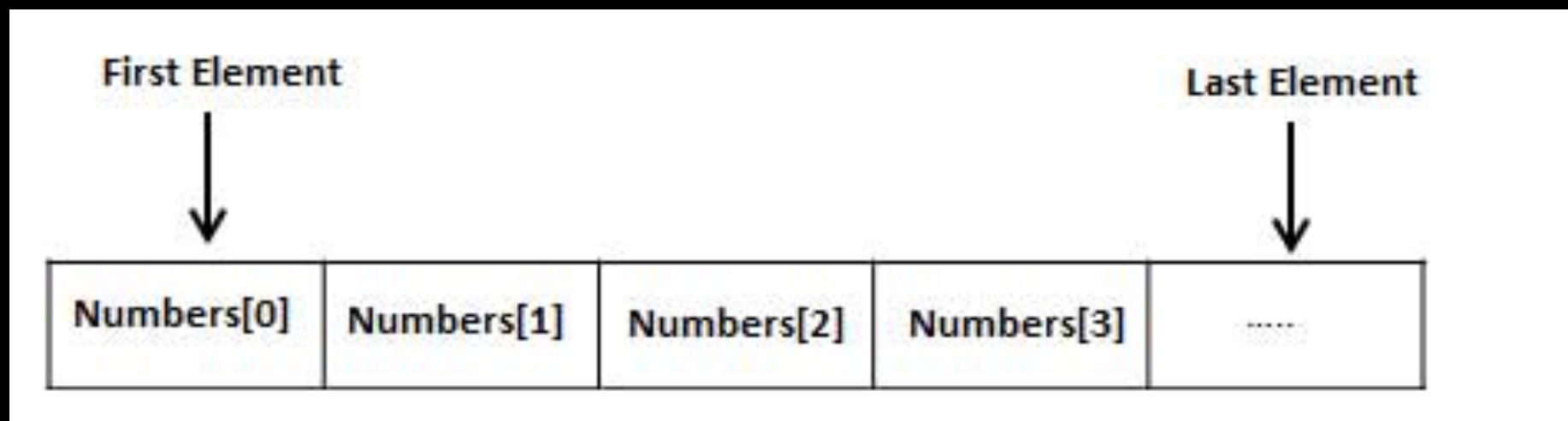
PROGRAMACIÓN BÁSICA

PROF. ALMA GONZÁLEZ



ARREGLOS (ARRAYS)

- Los arreglos son tipos de estructuras que permiten almacenar una colección de elementos de forma secuencial.
- En lugar de declarar variables separadas como numero0, numero1, numero2, etc... es posible declarar un arreglo que contiene estos elementos, a los cuales podemos acceder usando un índice.



ARREGLOS (ARRAYS)

- Debemos declarar los arreglos incluyendo el tipo de variables que lo integran. Los arreglos pueden ser del tipo entero, float, char, etc.
- `float numeros[10];` //Es la declaración de una variable tipo arreglo que tiene 10 elementos.
- Puede darse el tamaño a través de una variable previamente definida:

```
int N=10;
```

```
float numeros[N];
```

- Podemos asignar valores a cada uno de los elementos del arreglo, elemento por elemento, por ejemplo:

```
numeros[4]=7; //se le asigna valor 7 al 5o elemento.
```

- O bien usando ciclos, definiendo una regla de asignación.
- Ejemplo 1:

```
for ( i = 0; i < N; i++ ) {
```

```
    numeros[ i ] = 0.; //Asigna 0 a cada elemento del arreglo.
```

```
}
```

• Ejemplo 2:

```
for ( i = 0; i < N; i++ ) {  
    scanf("%f",&numeros[ i ]); //Asigna el valor a partir de lo  
    que proporciona el usuario, linea por linea.  
}
```

• Ejemplo 3:

```
for ( i = 0; i < N; i++ ) {  
    printf("%f \n",numeros[i]); //Imprime a la pantalla el valor  
    del elemento i-esimo del arreglo.  
}
```

• Ejemplo 4:

```
for ( i = 0; i < N; i++ ) {  
    scanf("%f",&numeros[ i ]);  
    numeros[i]*=2; //Multiplica el numero dado por 2.  
    printf("%f \n",numeros[i]); //Imprime el valor del arreglo  
    modificado.  
}
```


- Ejercicio 5:

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

```
int main(){
```

```
    int i,j,n;
```

```
    printf("Dime el numero de elementos para trabajar\n");
```

```
    scanf("%i",&n);
```

```
    float numeros[n]; //No es la forma optima de hacer una  
    reserva de memoria dinamica.
```

```
    for(j=0;j<n;j++){
```

```
        scanf("%f",&numeros[j]);
```

```
        numeros[j]*=2;
```

```
        printf("%f\n",numeros[j]);
```

```
    }
```

```
    return 0;
```

```
}
```

EJERCICIO 1.

- Escribir un programa que solicite información de un numero pre-determinado de estudiantes. Ej. que pida la edad, el sexo (0:hombre, 1:Mujer), semestre(1-9) y promedio de la carrera de 10 alumnos.
- El programa debe almacenar esta información en arreglos.Finalmente se deberá reportar un resumen de la información capturada, ej:
- Ej. se capturaron: 10 estudiantes de los cuales 4 son mujeres y 6 hombres. El numero de estudiantes por semestre es:.... y el promedio de calificaciones de todos los estudiantes es:

ABRIR Y CERRAR ARCHIVOS.
PARA LEER SU CONTENIDO, O ESCRIBIR CONTENIDO

- Debemos declarar una variable que nos permita manipular el archivo, la sintaxis para declararla es:

`FILE *fp;` // fp es el nombre de la variable, podemos llamar de otra forma, pero fp es estándar

ABRIR Y CERRAR ARCHIVOS.

- Para abrir un archivo, una vez declarada la variable:

```
fp=fopen("nombre_archivo", "r"); //El argumento "r"
```

indica que es de solo lectura, otras opciones son:
"w" (write, escribir), "a" (append, añadir). También se
pueden usar "r+", "w+", "a+"

- Para cerrar el archivo, una vez que terminé de leerlo o de escribir en él:

```
fclose(fp);
```


LEER UN ARCHIVO

- Una vez que el archivo ha sido abierto en modo lectura, podemos leer su contenido de la siguiente forma:
 - `fscanf(fp, "%f %f", &var1, &var2);` //Por ejemplo, suponiendo que estamos leyendo la primera línea que tiene dos números y asignamos cada uno a una variable.
 - `fgets(var3, 255, (FILE*)fp);` //Suponiendo que la línea es una cadena de caracteres y que se guardara en una variable previamente definida como:
 - `char var3[255];` //el número 255 es el tamaño máximo para definir una cadena de caracteres (string).

- Una vez que el archivo ha sido abierto en modo escritura, podemos añadir contenido de la siguiente forma:
 - `fprintf(fp, "%f %f", var1, var2);` //Por ejemplo, suponiendo que escribimos una linea que tendrá dos numero dados por las variables `var1, var2`. Podemos usar `fprintf` con variables del tipo `char, string, int, float, etc...`
 - `fputs("Esta es una prueba de fputs...\n", fp);` //Nos permite escribir un string.
 - `fprintf(fp, "Esta es una prueba de fprintf...\n");`
 - `fputc("a", fp);` //Nos permite escribir un carácter.

EJEMPLO

```
FILE *archivo;  
float var1,var2;  
char var[255];  
//ESCRITURA  
archivo = fopen("test.txt", "w");  
fputs("Esta es una prueba de fputs...\n", archivo);  
fprintf(archivo, "fprintf...\n");  
fprintf(archivo, "%f %f \n", 0.15,100.8);  
fclose(archivo);
```

```
//LECTURA
```

```
archivo = fopen("test.txt", "r");  
fgets(var,255,(FILE*)archivo);  
printf("%s",var);  
fscanf(archivo,"%s",var);  
printf("%s\n",var);  
fscanf(archivo,"%f %f",&var1,&var2);  
printf("%f %f\n",var1,var2);  
fclose(archivo);
```

EJERCICIO 2.

- Modificar el ejercicio de la clase pasada para leer la información de los estudiantes a partir de un archivo, y a su vez escribir el resultado en otro archivo.