NOTAS DE CLASE CECS 2202
Por: Prof. Blanca Tallaj
TEMA: ARREGLOS

Concepto de Arreglo:
Un arreglo es un grupo consecutivo de localizaciones de memoria que tienen el mismo nombre y el mismo tipo de dato.

Para poder accesar una localización de memoria en particular (o un elemento del arreglo), se debe especificar el nombre del arreglo y el sub-índice (la posición de ese elemento en particular dentro del arreglo).

Declaración de Arreglos:
int c[4]; → Declaración de un arreglo de 4 elementos de tipo entero
char a[4]; → Declaración de un arreglo de 4 elementos de tipo caracter
double b[4] → Declaración de un arreglo de 4 elementos de tipo double (real)

Ilustración de los tres arreglos declarados anteriormente:

c[0]	-33
c[1]	18000
c[2]	0
c[3]	55

a[0]	'H'
a[1]	'a'
a[2]	'y'
a[3]	'\0'

b[0]	3.333
b[1]	2.24
b[2]	-1000.0
b[3]	12.45

1. Nombre del arreglo: c, a, b
2. Sub-índice o posición de un elemento dentro de un arreglo: [0] – [3]
3. Contenido de cada elemento del arreglo: Un dato del tipo especificado en la declaración del arreglo (int, char, double, etc.).

Ejemplo de acceso de un elemento de un arreglo:

x = c[2] / 2;	→ Divide el contenido de la celda 2 del arreglo c (0), entre 2 y el resultado de la división entera se guarda en la variable x.
y = c[0] + c[1] + c[2] + c[3];	→ Se están sumando los contenidos de las celdas 0 a 3 del arreglo c y el resultado se guarda en la variable y.
c[m + n] = 12;	→ Si m = 1 y n = 1, se almacena 12 en la celda 2 del arreglo c.

Se debe tener especial cuidado de almacenar en una celda de un arreglo, solamente información que se corresponda con el tipo de dato con que se declaró el arreglo.

Cómo se recorren los elementos de un arreglo:

Si necesitamos limpiar los elementos de un arreglo, o recorrerlo completo, celda por celda, para llenarlo o para usar el contenido, se usa un ciclo (usualmente la instrucción for). En el siguiente ejemplo el arreglo c se recorre para guardar 0 a cada una de sus 4 celdas:

```
for (int i = 0; i < 4; i++)
        c[i] = 0;
```

Estas instrucciones recorren el arreglo c, elemento por elemento, y lo limpian, es decir, le mueve un cero a cada celda de memoria del arreglo c. Otra forma de limpiar un arreglo, o de asignarle valores iniciales a un arreglo, es por medio de una *lista de valores separados por coma*, en la misma declaración del arreglo, como sigue:

`int c[4] = {0};` → declaración del arreglo c, inicialización explicita del primer elemento del arreglo a 0, e inicialización implícita de los demás elementos del arreglo a 0.

`double b[4] = { 3.333, 2.24, -1000.0, 12.45 };` → declaración del arreglo b, inicialización explicita de sus 4 elementos en los valores especificados entre las llaves.

`int c[4] = { 1, 2, 3, 4, 5 };` → Error de sintaxis porque el arreglo se esta definiendo de 4 elementos y se esta inicializando con 5 elementos.

`int n[ ] = { 5, 4, 3, 2, 1 };` → Si el tamaño del arreglo se omite de una declaración que tiene una lista de inicialización, el número de elementos del arreglo va a ser asumido como el número de elementos en la lista de inicialización (5, en el caso de nuestro ejemplo).

En el siguiente ejemplo, el arreglo c se recorre para guardar, en cada celda, una información entera que se solicita al usuario:

```
for(int i = 0; i < 4; i++)
{
        cout << ”Entre el número entero ” << i+1 << “: ”;
        cin >> c[i];
}
```

Note que i es la variable que se usa para representar el subíndice del arreglo, y es un valor que el for inicia en 0 y se va incrementando de 1 en 1 hasta llegar a 4. Para el valor 4 el ciclo se corta y el programa no entra a ejecutar lo que el ciclo tiene

adentro. Es decir, se van a solicitar valores al usuario cuando i es 0, 1, 2 y 3 (4 valores en total, y 4 es la capacidad de almacenamiento que tiene el arreglo c).

Supongamos ahora que declaramos una constante entera as que representará el número de elementos totales del arreglo (número de celdas). Entonces la definición del arreglo c anterior puede hacerse de la siguiente manera:

```
const int as = 4;
int c[as];
```

y el ciclo que recorre el arreglo c, elemento por elemento, ahora puede expresarse como:

```
for (int i = 0; i < as; i++)
{
        //Coloque aquí el código con aquella operación que desea realizar en o con
        //cada elemento c[i] del arreglo c.
}
```

A continuación se muestra el código que suma los elementos del arreglo c:

```
int sum = 0;
for (int i = 0; i < as; i++)
{
        sum = sum + c[i];
}
cout << "La suma de los elementos del arreglo c es = " << sum << '\n';
```

Y a continuación se presenta el código que multiplica los elementos del arreglo c:

```
int mult = 1;
for (int i = 0; i < as; i++)
{
        mult = mult * c[i];
}
cout << "La multiplicacion de los elementos del arreglo c es = " << mult << '\n';
```

ARREGLO DE CARACTERES:

Hay un tipo de arreglo para el cual C++ permite hacer un manejo especial: los arreglos de caracteres que se usan para almacenar palabras, frases u oraciones. Estos arreglos que contienen un conjunto de caracteres adentro se les suele denominar **strings**. Entonces llamamos strings a un arreglo caracteres que se usa para almacenar palabras, frases u oraciones. El string "Hola Lia", por ejemplo, es un arreglo que contiene los siguientes caracteres: 'H', 'o', 'l', 'a', ' ', 'L', 'i', 'a', '\0'

Los arreglos de caracteres que almacenan strings (palabras, frases u oraciones) tienen ciertas características especiales que se expresan a continuación:

1. Un arreglo de caracteres puede ser inicializado usando un string literal ("Hola Lia"), en lugar de una lista de inicialización ({'H', 'o', 'l', 'a', ' ', 'L', 'i', 'a', '\0'}). Por ejemplo:
   char str[ ] = "Hola Lia"; → inicializa los elementos del arreglo str con los caracteres individuales del string literal "Hola Lia". **OJO**: La longitud de este arreglo será de 9 elementos (1 más que el numero de caracteres en el string literal), ya que incluye el caracter especial de terminación de un string conocido como el caracter nulo ('\0') (el carácter nulo es un "backslash" seguido de un cero).
2. La declaración anterior es equivalente a:
   char str[ ] = { 'H', 'o', 'l', 'a', ' ', 'L', 'i', 'a', '\0' };
3. Los arreglos de caracteres deben ser definidos lo suficientemente grandes como para poder guardar todos los caracteres en el string **Y** el caracter de fin de string o caracter nulo ('\0').
4. Como un string es un arreglo de caracteres, podemos accesar caracteres individuales de un string accesando ese caracter por medio de su índice. Del ejemplo anterior, str[0] es 'H', y str[2] es 'l', str[4] es un espacio en blanco (' '), y str[8] es el carácter de fin de string, o sea '\0'.
5. Se puede recibir un arreglo de caracteres directamente en un string por medio de una instrucción cin, como sigue:

   cin >> str;

   (esto es algo que NO se puede hacer con los arreglos de enteros y reales, los cuales se deben recorrer siempre con un ciclo para llenar una a una sus celdas).
   Note que en la instrucción cin >> str; se observa lo siguiente:
   - Solo se coloca el nombre del arreglo, str.
   - La acción que se realiza es la de almacenar el string que se entra por el teclado en str, y automáticamente se le anexa el carácter nulo ('\0') al final del string.
   - Es responsabilidad del programador que el string que se entra no sobrepase la longitud del arreglo.
   - ***cin solo lee los caracteres que se entran en el teclado HASTA encontrar el primer carácter de "whitespace" (espacio en blanco), lo cual NO significa que un string no pueda tener espacios en blanco, sino que con una instrucción cin sola NO se puede almacenar un espacio en blanco en un string (para leer una frase completa, incluidos los espacios en blanco, estudiar la instrucción getline).***
6. Se puede imprimir un string con la instrucción cout. **La instrucción cout imprime todos los caracteres que encuentra HASTA el carácter de fin de string ('\0').** La forma de imprimir el string str con cout es: cout << str;

Vea Ej. 1 obtenido de FIG. 4.12, PAG. 255 DE DEITEL – Enfoque en el uso del for y en la condición de fin de ciclo que trabaja con el carácter de fin de string ('\0').

```
// Fig. 4_12: fig04_12.cpp
// Treating character arrays as strings

#include <iostream>

using std::cout;
using std::cin;
using std::endl;

int main()
{
   char string1[ 20 ], string2[] = "string literal";

   cout << "Enter a string: ";
   cin >> string1;
   cout << "string1 is: " << string1
        << "\nstring2 is: " << string2
        << "\nstring1 with spaces between characters is:\n";

   for ( int i = 0; string1[ i ] != '\0'; i++ )
      cout << string1[ i ] << ' ';

   cin >> string1;
   cout << "\nstring1 is: " << string1 << endl;

   cout << endl;
   return 0;
}
```

Ejecucion del Programa:

Enter a string: Hello World
string1 is: Hello
string2 is: string literal
string1 with spaces between characters is:
H e l l o
string1 is: World

Press any key to continue

COMO PASAR ARREGLOS A FUNCIONES:

C++ automáticamente pasa los arreglos a las funciones usando un caso especial de paso por referencia (lo que se pasa es la dirección de memoria del primer elemento del arreglo). La función llamada puede modificar los elementos individuales del arreglo, y dicha modificación se refleja de regreso en la función que hizo el llamado.

1. Para pasar un arreglo a una función, coloque el nombre del arreglo sin los corchetes en el llamado, como sigue:

   int Temp[10]; → Definición de un arreglo de temperaturas de 10 elementos.

   LlenarArreglo (Temp, 10); → Se está llamando al módulo LlenarArreglo, y se envía como argumento el arreglo Temp y el número de elementos del arreglo.

2. Prototipo de la función LlenarArreglo:

   void LlenarArreglo (int [ ], int);

3. Encabezado de la función LlenarArreglo:

   void LlenarArreglo (int T[ ], int NumElementos)

COMO PASAR ELEMENTOS INDIVIDUALES DE UN ARREGLO A FUNCIONES:

Paso Por Valor

1. Para pasar por valor (en una sola dirección), un elemento de un arreglo a un módulo, siga las mismas reglas que usa para pasar variables sencillas. Si el elemento que se quiere enviar es el elemento 3 del arreglo, se envía como parámetro: Temp[3] (paso por valor), como sigue:

   int Temp[10]; → Definición de un arreglo de temperaturas de 10 elementos.

   EnviaElementoPorValor (Temp [3]); → Se está llamando al módulo EnviaElementoPorValor, y se envía como argumento el elemento 3 del arreglo Temp (por valor).

2. Prototipo de la función EnviaElementoPorValor:

   void EnviaElementoPorValor (int);

3. Encabezado de la función EnviaElementoPorValor:

   void EnviaElementoPorValor (int e) → Note que se recibe el elemento del arreglo que se envió en una variable sencilla a la que se le pone el nombre con que se va a reconocer esa información dentro de la función.

Paso Por Referencia

1. Para pasar por referencia (en dos direcciones), un elemento de un arreglo a un módulo, siga las mismas reglas que usa para pasar variables sencillas. Si el elemento que se quiere enviar por referencia es el elemento 3 del arreglo, se envía como parámetro: Temp[3] (paso por referencia, modalidad parámetros de referencia), pero se recibe con &, como sigue:

   int Temp[10]; → Definición de un arreglo de temperaturas de 10 elementos.

   EnviaElementoPorRef (Temp [3]); → Se está llamando al módulo EnviaElementoPorRef, y se envía como el elemento 3 del arreglo Temp

2. Prototipo de la función EnviaElementoPorRef:

   void EnviaElementoPorRef (int &);

3. Encabezado de la función EnviaElementoPorRef:

   void EnviaElementoPorRef (int &e) → Note que se recibe el elemento del arreglo que se envió en un nombre de variable cualquiera, que es una referencia (u otro nombre) para la misma celda Temp[3] que se envió, en este caso.

```
// Fig. 4.14: fig04_14.cpp
// Passing arrays and individual array elements to functions

#include <iostream>

using std::cout;
using std::endl;

#include <iomanip>

using std::setw;

void modifyArray( int [], int );
void modifyElement( int );

int main()
{
   const int arraySize = 5;
   int i, a[ arraySize ] = { 0, 1, 2, 3, 4 };

   cout << "Effects of passing entire array call-by-reference:"
        << "\n\nThe values of the original array are:\n";

   for ( i = 0; i < arraySize; i++ )
      cout << setw( 3 ) << a[ i ];

   cout << endl;

   // array a passed call-by-reference
   modifyArray( a, arraySize );

   cout << "The values of the modified array are:\n";

   for ( i = 0; i < arraySize; i++ )
      cout << setw( 3 ) << a[ i ];

   cout << "\n\n\n"
        << "Effects of passing array element call-by-value:"
        << "\n\nThe value of a[3] is " << a[ 3 ] << '\n';

   modifyElement( a[ 3 ] );

   cout << "The value of a[3] is " << a[ 3 ] << endl;

   return 0;
}
```

```
// In function modifyArray, "b" points to the original
// array "a" in memory.
void modifyArray( int b[], int sizeOfArray )
{
   for ( int j = 0; j < sizeOfArray; j++ )
      b[ j ] *= 2;
}

// In function modifyElement, "e" is a local copy of
// array element a[ 3 ] passed from main.
void modifyElement( int e )
{
   cout << "Value in modifyElement is "
        << ( e *= 2 ) << endl;
}
```

Salida del Programa Anterior:

Effects of passing entire array call-by-reference:

The values of the original array are:
0 1 2 3 4
The values of the modified array are:
0 2 4 6 8

Effects of passing array element call-by-value:

The value of a[3] is 6
Value in modifyElement is 12
The value of a[3] is 6
Press any key to continue

```
//Ejemplo de pasos de parámetros por valor y por
//referencia cuando el argumento es un elemento de un arreglo.

#include <iostream.h>

void PasoPorValor (int);
void PasoPorRef (int &);

void main()
{
        int Temp[] = {1,2,3,4,5};

        cout << "Elemento 3 de arreglo Temp antes de PasoPorValor = "
                << Temp[2] << endl;

        PasoPorValor ( Temp[2] );

        cout << "Elemento 3 de arreglo Temp despues de PasoPorValor = "
                << Temp[2] << endl;

        PasoPorRef ( Temp[2] );

        cout << "Elemento 3 de arreglo Temp despues de PasoPorRef = "
                << Temp[2] << endl;

}

void PasoPorValor ( int e )
{
        e = e + 1;
}

void PasoPorRef ( int &e )
{
        e = e + 1;
}
```

Salida Del Programa Anterior:
Elemento 3 de arreglo Temp antes de PasoPorValor = 3
Elemento 3 de arreglo Temp despues de PasoPorValor = 3
Elemento 3 de arreglo Temp despues de PasoPorRef = 4
Press any key to continue