<!--Estudio Shonos-->

GUIA PARA NO MORIR CON C++ P2{

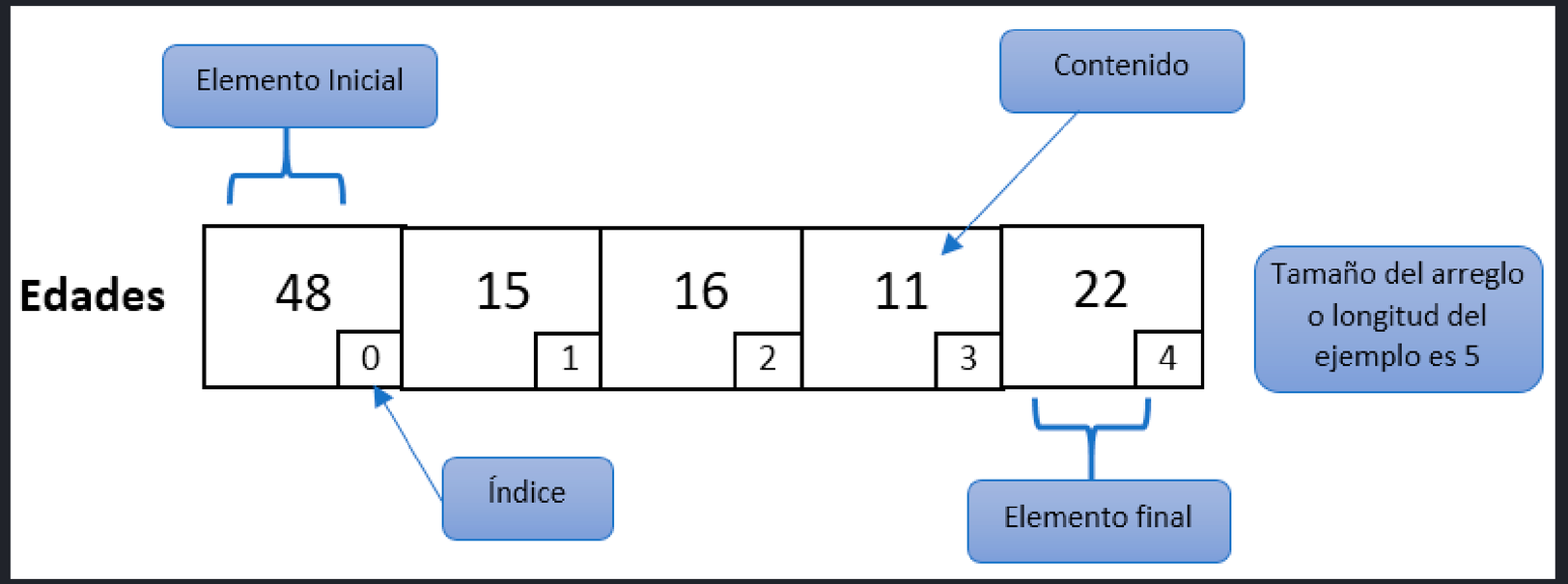
<Por="Marjorie Reyes"/>

}

Contenidos

Arreglos {

Se utilizan para almacenar múltiples valores dentro de una sola variable, en lugar de crear muchas variables.



}

Arreglos {

Se utilizan para almacenar múltiples valores dentro de una sola variable, en lugar de crear muchas variables.

F0 c0

F1 c0

F0 c1

F1 c1

}

Arreglos (Declaración) {

Para crear un arreglo se debe indicar el tipo de dato de los valores que se almacenarán así como la cantidad de ellos.

ejemplo.cpp

```
// Estructura
TipoDeDato NombreArreglo[Tamaño];

double prices[5];
string users[8];
```

#Quote #Programming #Selfcare

}

Arreglos (Acceso) {

Se obtiene uno de los valores dentro del arreglo por medio de su posición.

Es importante saber que la primera posición es 0 y la última es (tamaño del arreglo - 1)

ejemplo.cpp

```
// Estructura
NombreArreglo[Posición] = Valor;

double prices[8];
prices[3] = 29.99; //Válido
prices[8] = 29.99; //No es válido
prices[0] = 29.99; //Válido
```

#Quote #Programming #Selfcare

}

Arreglos (Acceso) {

Se obtiene uno de los valores dentro del arreglo por medio de su posición.

Es importante saber que la primera posición es 0 y la última es (tamaño del arreglo - 1)

ejemplo.cpp

```
// Estructura
cout << NombreArreglo[Posición];
Variable = NombreArreglo[Posición];

double prices[8];
cout << prices[3];
double price3 = prices[3];
cout << price3;
```

#Quote #Programming #Selfcare

}

Arreglos (declaración múltiple) {

En lugar de asignar los valores uno por uno, es posible asignar varios a la vez. Esto se hace colocándolos dentro de llaves separados por comas.

Ya no es necesario colocar el tamaño, este tomará el de la cantidad de valores colocados.

ejemplo.cpp

```
// Estructura
TipoDato NombreArray[] = {Valor1,
Valor2, Valor3, Valor4};

double prices[] = {5.99, 3.2, 9.99,
29.99};

string names[] = {”Sara”, “Jenny”,
“Blair”};
```

#Quote #Programming #Selfcare

}

Arreglos (Iteración) {

Es posible iterar arreglos (pasar posición por posición) por medio de ciclos.

ejemplo.cpp

```
double prices[] = {5.99, 3.2, 9.99, 29.99};

for(int x=0;x<4;x++) {
  cout << prices[x] << endl;
}
```

#Quote #Programming #Selfcare

}

for-each loop {

for-each es un ciclo que se utiliza para iterar arreglos de una manera más sencilla.

“VariableTemporal” toma el valor del elemento de la posición que se está iterando.

ejemplo.cpp

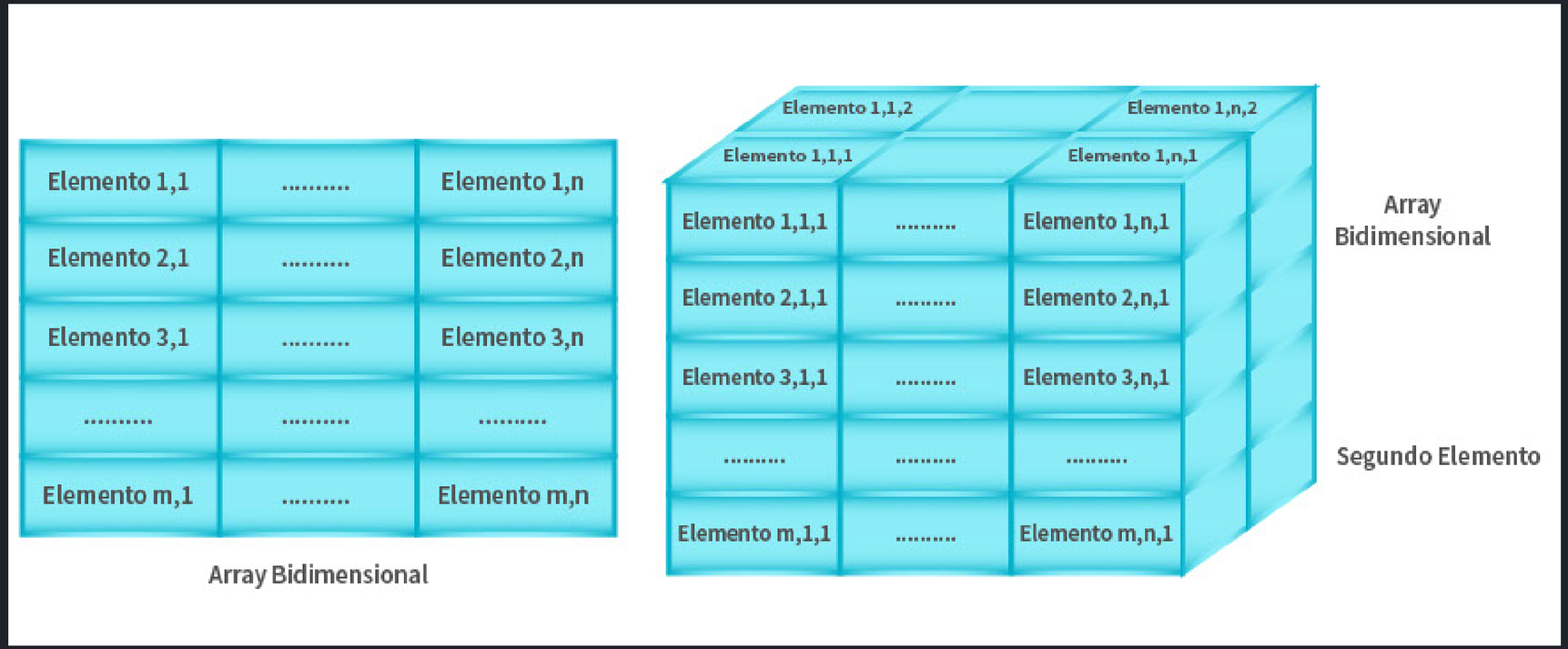
```
// Estructura

for(TipoDato VariableTemporal:
NombreArreglo) {
  //bloque de código
}
```

#Quote #Programming #Selfcare

}

Arreglos multidimensionales {

Son arreglos que tienen más de una dimensión. Por ejemplo, cuando se busca almacenar registros de asientos en un cine.



}

Arreglos multidimensionales (declaración){

Son arreglos dentro de arreglos.

La cantidad de corchetes indica las dimensiones:

2 dimnesiones [][]
3 dimensiones [][][]

ejemplo.cpp

```
// Estructura
TipoDato Nombre[Filas][Columnas];

TipoDato Nombre[Filas][Columnas] =
{{Valor00, Valor01, Valor02, ...},
{Valor10, Valor11, Valor12, ...},
...}
```

#Quote #Programming #Selfcare

}

Arreglos multidimensionales (acceso){

ejemplo.cpp

```
// Estructura

NombreArray[n][m];
NombreArray[x][y][z];
```

#Quote #Programming #Selfcare

}

Arreglos multidimensionales (iterar){

Es posible iterar arreglos de 2 dimensiones utilizando ciclos anidados.

}

Punteros {

Una **VARIABLE** es un valor almacenado en un espacio en la memoria de la computadora.

Un **PUNTERO** es una variable que almacena la dirección de memoria en la que está almacenada otra variable.

ejemplo.cpp

```
// Estructura declaración
TipoDato *NombrePuntero;

// Estructura asignación
TipoDato *NombrePuntero = &Variable;

// Acceder a la dirección guardada
cout << NombrePuntero;

// Acceder al valor en la dirección
cout << *NombrePuntero;
```

#Quote #Programming #Selfcare

}

Punteros {

& -> Accede a la dirección de memoria de una variable.

* -> Se utiliza para crear punteros y para acceder al valor de una dirección en memoria.

ejemplo.cpp

```
//Declarar puntero y acceder a la dirección en memoria
int *p = &num;

//Muestra el valor de la dirección
cout << *p;

//Muestra la dirección en memoria
cout << p;
```

#Quote #Programming #Selfcare

}

Punteros y arreglos {

Aunque al declarar un arreglo no se utilice “*”, un arreglo es un puntero que almacena la dirección en memoria del primer elemento.

ejemplo.cpp

```
char letras = {{‘A’,’B’},
{‘C’,’D’}}
char *p = letras;
cout << *p << endl; //Imprime el
primer valor del arreglo.
```

#Quote #Programming #Selfcare

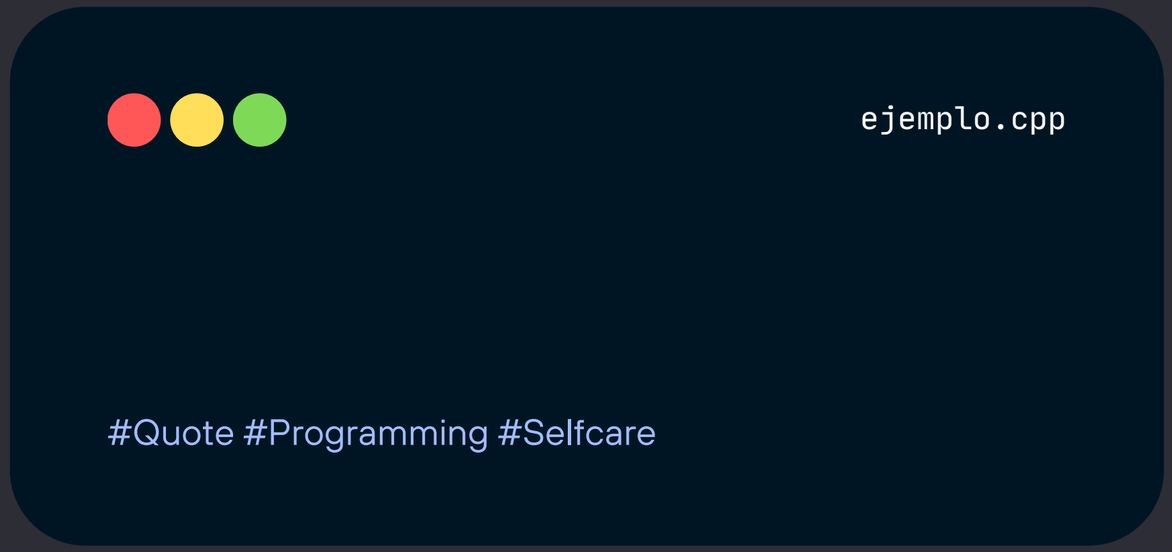
}

Memoria dinámica {

Se utiliza cuando se quiere almacenar varios datos pero no se tiene certeza de cuántos son.

Se utiliza la palabra “new” para asignar en un puntero.

Es posible eliminarlo por medio de delete.



}

Funciones {

Son bloques de código que ejecutan una tarea.

La idea es crearlos cuando se realizará esa tarea varias veces a lo largo del programa.

Las funciones **void** no retornan valores.
Las funciones **no void** retornan un valor del tipo de dato indicado.

ejemplo.cpp

```
//Estructura
void NombreFuncion(){
  //Bloque de código
}


TipoDato NombreFuncion(){
  //Bloque de código
  return valor;
}
```

#Quote #Programming #Selfcare

}

Llamar funciones {

Es la forma en la que la función ejecutará el bloque de código.

ejemplo.cpp

```
// Estructura
// Función void
NombreFuncion();

//Función no void
Variable = NombreFuncion();
```

#Quote #Programming #Selfcare

}

Parámetros de funciones {

Se utiliza para utilizar una misma función con diferentes valores, lo que hace que sea reusable.

ejemplo.cpp

```
// Estructura

void NombreFuncion(TipoDato
NombreParametro, TipoDato
NombreParametro, ...){
  //Bloque de código
}
```

#Quote #Programming #Selfcare

}

Archivos {

Se utiliza el header fstream para manejar archivos.

ofstream se usa para abrir un archivo.

is_open() se usa para verificar si se pudo abrir/crear el archivo.

close() se utiliza para cerrar el archivo.

ejemplo.cpp

```
// Estructuras

#include <fstream>

using namespace std;

ofstream =
VariableArchivo(NombreArchivo.Extensión);

VariableArchivo << TextoEscribir << endl;

VariableArchivo.is_open();

VariableArchivo.close()
```

#Quote #Programming #Selfcare

}

<!--Estudio Shonos-->

Gracias {

}