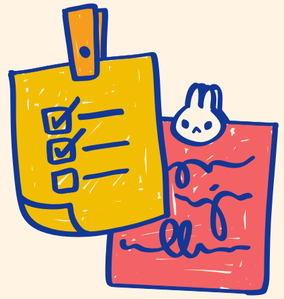
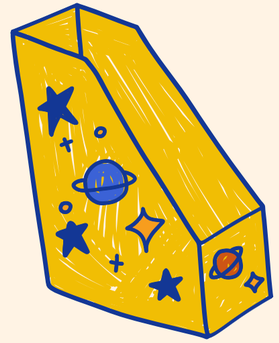






Curso de C++ básico

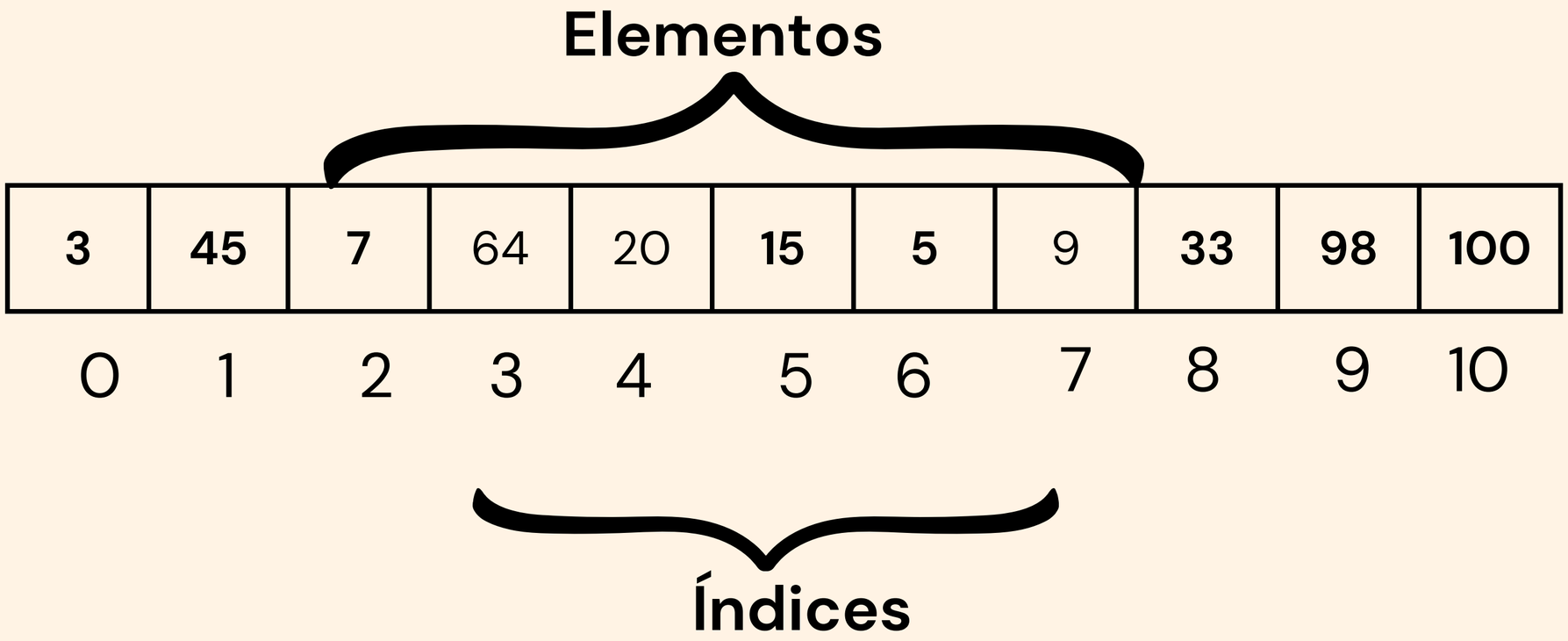
Programa de Tecnología en Cómputo (PROTECO)
Intersemestral 2023-1

PROTECO

Arreglos.

¿Qué es un arreglo?

Un arreglo es una variable compuesta por un conjunto de datos del mismo tipo. Tienen un tamaño fijo, que es la cantidad de objetos que pueden almacenar. Todos los elementos de un arreglo, sin importar su tamaño o dimensiones, se guardan de manera contigua en la RAM y tienen asignado un índice mediante el cual se puede acceder a ellos.



Declaración de arreglos unidimensionales

Para declarar un arreglo en C y C++, se utiliza la siguiente sintaxis:

<tipo_elementos> <nombre_arreglo>[<tamaño>];

Para declarar e inicializar un arreglo en C y C++, se puede utilizar cualquiera de las siguiente sintaxis:

<tipo> <nombre>[] = {<elemento_0>, ..., <elemento_n-1>};

<tipo> <nombre>[<n>] = {<elemento_0>, ..., <elemento_n-1>};

Uso de arreglos unidimensionales

En C y C++, una vez creado un arreglo, éste únicamente se puede manipular elemento por elemento, por lo que es necesario conocer el índice del elemento deseado.

Para acceder a un elemento de un arreglo en C y C++, se utiliza la siguiente sintaxis:

<nombre_arreglo>[<índice>];

O bien, se pueden recorrer utilizando ciclos for, como se muestra a continuación.

```
int i;
for (i = 0; i < <tamaño_del_arreglo>; i++) {
        <nombre_arreglo>[i];
 }
```

Arreglos bidimensionales

Los arreglos bidimensionales permiten representar matrices de datos.
Se pueden ver como arreglos unidimensionales, donde cada elemento a su vez es otro arreglo unidimensional.

Elementos[0] →	Elemento[0][0]	Elemento[0][1]	...	Elemento [0][n-1]
Elementos[1] →	Elemento[1][0]	Elemento[1][1]	...	Elemento[1][n-1]
	...	....	...	....
Elementos[m-1] →	Elemento [m-1][0]	Elemento [m-1][1]	...	Elemento[m-1][n-1]

Declaración de arreglos bidimensionales

Para declarar un arreglo bidimensional en C++, se utiliza la siguiente sintaxis:
<tipo_elementos> <nombre_arreglo>[<num_filas>][<num_columnas>];

Para declarar e inicializar un arreglo bidimensional en C y C++, se puede utilizar cualquiera de las siguiente sintaxis o combinaciones de ellas:

<tipo> <nombre>[<m>][<n>] = {<0,0>, ..., <0,n-1>, ..., <m-1,0>, ..., <m-1,n-1>};
<tipo> <nombre>[<m>][<n>] = {{<0,0>, ..., <0,n-1>},
...,
{<m-1,0>, ..., <m-1,n-1>}};
<tipo> <nombre>[][<n>] = {{<0,0>, ..., <0,n-1>}, ..., {<m-1,0>, ..., <m-1,n-1>}};

Uso de arreglos bidimensionales

En C++, una vez creado un arreglo bidimensional, éste únicamente se puede manipular elemento por elemento, por lo que es necesario conocer el índice de la fila y la columna del elemento deseado.

Para acceder a un elemento de un arreglo en C y C++, se utiliza la siguiente sintaxis:

<nombre_arreglo>[<índice_fila>] [<índice_columna>];

O bien, se pueden recorrer utilizando for anidados, como se muestra a continuación.

```
int fila;
int columna;
for (fila = 0; fila < <número_de_filas>; fila++) {
        for (columna = 0; columna < <número_de_columnas>; columna++){
            <nombre_arreglo>[fila][columna];
        }
}
```

Manejo de cadenas en C++.

Manejando *string*

En C, las cadenas se representan mediante arreglos de caracteres que terminan con el carácter nulo o de fin de cadena (\0).

C++ permite representar cadenas de la misma forma que C. Sin embargo, también implementa una clase llamada string para representar cadenas de caracteres, la cual incluye métodos y operadores para facilitar su manejo.

La clase ***string*** pertenece a la cabecera ***string*** y al *namespace std*.

Uso básico de cadenas.

La sintaxis para crear, leer e imprimir una cadena en C++ es la siguiente.

std::string <id>; //Declaración
std::getline(std::cin, <id>); //Lectura
std::cout << <id>; //Impresión

También se pueden manejar los elementos individuales de la cadena con la notación de arreglos.
La clase string contiene varios métodos y operadores que facilitan su manejo. A continuación se muestra una tabla con algunos de ellos.
En esta liga podrán encontrar todas las funciones y métodos de la clase string.

Método	**Descripción**
length()	Regresa la longitud de la cadena.
c_srt()	Regresa la cadena en formato de C (char *).
empty()	Regresa true si la cadena está vacía.
compare(str)	Compara dos cadenas y regresa true si son iguales. También se pueden usar los operadores == y !=.