📝 ACTIVIDADES

Resolver las siguientes actividades utilizando el material adjunto, investigar en Internet en caso de ser necesario:1. Explica la diferencia entre funciones y procedimientos en C++. ¿Cuándo es más adecuado utilizar una u otra?
2. Describe los componentes de una función.
3. Explica las diferencias entre un arreglo de caracteres y el tipo de dato string en C++
4. Investiga la funcionalidad de los siguientes operadores y métodos de la clase string, organizarlos en una tabla con un ejemplo de uso:
max_size / compare /copy / c_str / data / empty / erase / find find_last_of / front / insert replace / reserve / resize / substr / swap
5. ¿Qué es un vector en C++ y cuáles son sus principales características?¿Cómo se declara y se inicializa un vector en C++?
6. ¿Cómo se modifica el tamaño de un vector en C++?
7. ¿Qué sucede cuando se intenta acceder a un elemento fuera del rango del vector?
8. ¿Qué es la clase vector en C++ y qué funcionalidades proporciona?
9. Para los siguientes métodos de la clase vector busca un ejemplo de uso, organizando la información en una tabla:
size(): Obtiene el tamaño del vector.
capacity(): Obtiene la capacidad de almacenamiento actual del vector.
empty(): Comprueba si el vector está vacío.
push_back(valor): Añade un elemento al final del vector.
pop_back(): Elimina el último elemento del vector.
front(): Devuelve una referencia al primer elemento del vector.
back(): Devuelve una referencia al último elemento del vector.
at(posición): Devuelve una referencia al elemento en la posición especificada.
assign(inicio, fin): Remplaza el contenido del vector con un rango de elementos.
erase(posición): Elimina el elemento en la posición especificada.
erase(inicio, fin): Elimina un rango de elementos del vector.
insert(posición, valor): Inserta un elemento en la posición especificada.
insert(posición, inicio, fin): Inserta un rango de elementos en la posición especificada.
clear(): Elimina todos los elementos del vector.
swap(otro_vector): Intercambia el contenido del vector con otro vector.
10. ¿Qué es una clase? ¿cuál es su estructura? ¿Qué es un objeto?¿Cual es su proposito?

1. Diferencia entre funciones y procedimientos en C++

- **Funciones**: devuelven un valor. Se usan cuando necesitas que el bloque de código te entregue un resultado.

```
int sumar(int a, int b) {
    return a + b;
}
```

- **Procedimientos (funciones void)**: no devuelven ningún valor. Se usan cuando solo querés ejecutar instrucciones.

```
void saludar() {
    cout << "Hola Nyaa!" << endl;
```

```
}
```

¿Cuándo usar cada una?

- Usás una **función** cuando necesitás un resultado.
- Usás un **procedimiento (void)** cuando solo hacés acciones.

---

2. Componentes de una función

1. **Tipo de retorno**: el tipo de dato que devuelve (ej: `int`, `float`, `string`, `void`)
2. **Nombre**: cómo se llama la función.
3. **Parámetros**: entre paréntesis, los datos que recibe.
4. **Cuerpo**: el bloque de instrucciones `{ ... }`.

```
int sumar(int a, int b) { // Tipo: int | Nombre: sumar | Parámetros: (int a, int b)
    return a + b;         // Cuerpo
}
```

---

3. Arreglo de caracteres vs tipo string

Arreglo de caracteres	Tipo string
Usás `char nombre[20]`	Usás `string nombre`
Tamaño fijo	Tamaño dinámico
No tiene funciones	Tiene muchos métodos útiles
Más bajo nivel	Más fácil de usar

---

4. Métodos y operadores de `string` en C++

Método	Funcionalidad	Ejemplo
`max_size()`	Tamaño máximo permitido	`str.max_size();`
`compare()`	Compara strings	`str1.compare(str2);`
`copy()`	Copia contenido a arreglo char	`str.copy(arr, 5);`
`c_str()`	Devuelve puntero tipo `char*`	`puts(str.c_str());`
`data()`	Similar a `c_str()`, acceso al contenido	`str.data();`
`empty()`	Devuelve `true` si está vacía	`str.empty();`

Método	Funcionalidad	Ejemplo
`erase()`	Borra parte del string	`str.erase(2,3);`
`find()`	Busca una subcadena	`str.find("cat");`
`find_last_of()`	Última aparición de un char de un grupo	`str.find_last_of("aeiou");`
`front()`	Primer carácter	`str.front();`
`insert()`	Inserta en una posición	`str.insert(2, "owo");`
`replace()`	Reemplaza parte del string	`str.replace(1,3,"nya");`
`reserve()`	Reserva espacio	`str.reserve(100);`
`resize()`	Cambia el tamaño	`str.resize(10);`
`substr()`	Devuelve una subcadena	`str.substr(2,4);`
`swap()`	Intercambia contenidos	`str1.swap(str2);`

---

5. ¿Qué es un vector en C++?

- Es una clase que representa un arreglo dinámico.
- Puede crecer o reducir su tamaño.
- Se incluye con `#include <vector>`

```
vector<int> numeros = {1, 2, 3};
```

---

6. ¿Cómo se modifica el tamaño de un vector?

Con el método `resize()`:

```
numeros.resize(10);
```

---

7. ¿Qué pasa si accedés fuera del rango?

Provoca **error en tiempo de ejecución**. Puede crashear el programa.

---

8. ¿Qué es la clase vector y qué funcionalidades tiene?

- Clase de la STL que permite manejar colecciones de datos dinámicos.
- Métodos: `push_back`, `pop_back`, `resize`, `insert`, `erase`, etc.
- Se maneja como un arreglo pero con más poder y seguridad.

---

9. Métodos del vector (tabla)

Método	Descripción	Ejemplo
`size()`	Tamaño actual	`v.size();`
`capacity()`	Capacidad reservada	`v.capacity();`
`empty()`	Si está vacío	`v.empty();`
`push_back(valor)`	Agrega al final	`v.push_back(5);`
`pop_back()`	Elimina el último	`v.pop_back();`
`front()`	Primer elemento	`v.front();`
`back()`	Último elemento	`v.back();`
`at(posición)`	Accede con control de rango	`v.at(2);`
`assign(inicio, fin)`	Reemplaza contenido	`v.assign(arr, arr + 3);`
`erase(posición)`	Borra un elemento	`v.erase(v.begin() + 1);`
`erase(inicio, fin)`	Borra un rango	`v.erase(v.begin(), v.begin()+2);`
`insert(posición, valor)`	Inserta elemento	`v.insert(v.begin() + 1, 10);`
`insert(pos, inicio, fin)`	Inserta rango	`v.insert(v.begin(), arr, arr+3);`
`clear()`	Elimina todos los elementos	`v.clear();`
`swap(otro_vector)`	Intercambia contenido	`v1.swap(v2);`

---

10. ¿Qué es una clase? ¿Y un objeto?

- **Clase**: una plantilla que define atributos y métodos.

```
class Gato {
public:
    string nombre;
    void maullar() {
        cout << "Myaa~" << endl;
    }
};
```

- **Objeto**: una instancia concreta de una clase.

```
Gato miNeko;
miNeko.nombre = "Nyaa-chan";
```

Propósito: Representar cosas del mundo real y organizar mejor tu código.