### **1. Diferencia entre funciones y procedimientos en C++**

**Funciones:**

* **Definición:** Son bloques de código que realizan una tarea específica y pueden devolver un valor.
* **Sintaxis:**

**tipo\_de\_dato nombre\_de\_funcion(parametros) {**

**// cuerpo de la función**

**return valor;**

**}**

* **Uso:** Se utilizan cuando se necesita devolver un resultado y utilizar el valor en otra parte del programa.

**Procedimientos:**

* **Definición:** Son bloques de código que realizan una tarea específica pero no devuelven un valor.
* **Sintaxis:**

**void nombre\_de\_procedimiento(parametros) {**

**// cuerpo del procedimiento**

**}**

* **Uso: Se utilizan para ejecutar acciones sin necesidad de devolver un valor.**

**Cuándo usar cada uno:**

* **Función: Cuando necesitas un resultado que influencie el flujo de tu programa.**
* **Procedimiento: Cuando solo necesitas ejecutar una serie de instrucciones sin necesidad de un valor de retorno.**

### **2. Componentes de una función**

1. **Tipo de retorno:** Especifica el tipo de valor que la función devolverá. Si no devuelve ningún valor, se utiliza void.
2. **Nombre de la función:** Identificador que se usa para invocar la función.
3. **Parámetros (opcional):** Lista de variables de entrada que la función puede recibir. Se definen en los paréntesis después del nombre de la función.
4. **Cuerpo de la función:** Bloque de código que define lo que la función hace. Se encuentra entre llaves {}.

### **3. Diferencias entre un arreglo de caracteres y el tipo de dato string en C++**

**Arreglo de caracteres (char[]):**

* **Definición:** Es una secuencia de caracteres almacenados en una matriz.
* **Tamaño fijo:** Se debe definir el tamaño al momento de la declaración.
* **Termina en '\0':** Los arreglos de caracteres en C++ terminan con un carácter nulo ('\0').
* **Ejemplo:**

**char arr [ ] = "Hola";**

**Tipo de dato string:**

* **Definición: Es una clase de la biblioteca estándar de C++ que maneja cadenas de caracteres de manera más flexible.**
* **Tamaño dinámico: Puede crecer o reducirse según sea necesario.**
* **Métodos integrados: Ofrece métodos y operadores para manipular cadenas de manera más conveniente.**
* **Ejemplo:**

**std::string str = "Hola";**

**4. Funcionalidad de operadores y métodos de la clase string**

| **Método/Operador** | **Descripción** | **Ejemplo de uso** |
| --- | --- | --- |
| **max\_size()** | **Devuelve el tamaño máximo posible de la cadena.** | **str.max\_size()** |
| **compare()** | **Compara la cadena con otra cadena o substring.** | **str.compare("Hola")** |
| **copy()** | **Copia una parte de la cadena a un buffer.** | **str.copy(buffer, 4)** |
| **data()** | **Similar a c\_str(), pero puede devolver un puntero no constante.** | **str.data()** |
| **empty()** | **Verifica si la cadena está vacía.** | **str.empty()** |
| **erase()** | **Elimina caracteres de la cadena.** | **str.erase(0, 4)** |
| **find()** | **Busca una subcadena y devuelve la posición de la primera aparición.** | **str.find("Hola")** |
| **find\_last\_of()** | **Busca la última aparición de cualquier carácter de un conjunto.** | **str.find\_last\_of("o")** |
| **front()** | **Devuelve el primer carácter de la cadena.** | **str.front()** |
| **insert()** | **Inserta caracteres en la cadena en una posición específica.** | **str.insert(5, " Mundo")** |
| **replace()** | **Reemplaza parte de la cadena con otra subcadena.** | **str.replace(0, 4, "Hola")** |
| **reserve()** | **Reserva espacio en memoria para la cadena.** | **str.reserve(50)** |
| **resize()** | **Cambia el tamaño de la cadena.** | **str.resize(10)** |
| **substr()** | **Devuelve una subcadena de la cadena.** | **str.substr(0, 4)** |
| **swap()** | **Intercambia el contenido de la cadena con otra cadena.** | **str.swap(otra\_cadena)** |

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### **5. Vector en C++**

**Definición:** Un vector es una estructura de datos de la biblioteca estándar de C++ que representa una colección dinámica de elementos del mismo tipo.

**Características principales:**

* **Tamaño dinámico:** Puede crecer o reducirse según sea necesario.
* **Acceso aleatorio:** Permite acceso directo a cualquier elemento.
* **Contiguo en memoria:** Los elementos están almacenados en un bloque de memoria contiguo.

**Declaración e inicialización:**

**#include <vector>**

**std::vector<int> vec; // Vector vacío**

**std::vector<int> vec\_inicializado(5, 10); // Vector con 5 elementos inicializados a 10**

### **6. Modificación del tamaño de un vector**

**Para modificar el tamaño de un vector en C++ se pueden utilizar los siguientes métodos:**

* **resize(): Cambia el tamaño del vector. Si el nuevo tamaño es mayor, se añaden elementos con el valor predeterminado.**

**vec.resize(10); // Aumenta el tamaño a 10 elementos**

**vec.resize(5, 1); // Cambia el tamaño a 5 y llena con el valor 1**

**reserve(): Reserva espacio en la memoria para el vector, pero no cambia su tamaño actual.**

**vec.reserve(20); // Reserva espacio para 20 elementos**

### **7. Acceso a elementos fuera del rango del vector**

**Cuando se intenta acceder a un elemento fuera del rango del vector, se produce un comportamiento indefinido o se lanza una excepción std::out\_of\_range si se usa el método at(). Ejemplo:**

**std::vector<int> vec = {1, 2, 3};**

**std::cout << vec[10]; // Comportamiento indefinido**

**std::cout << vec.at(10); // Lanza std::out\_of\_range**

### **8. Clase vector en C++**

**Definición: La clase vector es una plantilla en la biblioteca estándar de C++ que representa un contenedor dinámico de elementos.**

**Funcionalidades:**

* **Almacenamiento dinámico: Permite redimensionar el contenedor.**
* **Acceso rápido: Proporciona acceso aleatorio a los elementos.**
* **Métodos de manipulación: Incluye métodos para añadir, eliminar, e insertar elementos.**

**9. Métodos de la clase vector**

| **Método** | **Descripción** | **Ejemplo de uso** |
| --- | --- | --- |
| **size()** | **Obtiene el tamaño actual del vector.** | **vec.size()** |
| **capacity()** | **Obtiene la capacidad de almacenamiento del vector.** | **vec.capacity()** |
| **empty()** | **Comprueba si el vector está vacío.** | **vec.empty()** |
| **push\_back(valor)** | **Añade un elemento al final del vector.** | **vec.push\_back(10)** |
| **pop\_back()** | **Elimina el último elemento del vector.** | **vec.pop\_back()** |
| **front()** | **Devuelve una referencia al primer elemento del vector.** | **vec.front()** |
| **back()** | **Devuelve una referencia al último elemento del vector.** | **vec.back()** |
| **at(posición)** | **Devuelve una referencia al elemento en la posición dada.** | **vec.at(2)** |
| **assign(inicio, fin)** | **Rellena el vector con un rango de elementos.** | **vec.assign(5, 1)** |
| **erase(posición)** | **Elimina el elemento en la posición especificada.** | **vec.erase(vec.begin() + 2)** |
| **erase(inicio, fin)** | **Elimina un rango de elementos del vector.** | **vec.erase(vec.begin() + 1, vec.begin() + 3)** |
| **insert(posición, valor)** | **Inserta un elemento en la posición dada.** | **vec.insert(vec.begin() + 2, 5)** |
| **insert(posición, inicio, fin)** | **Inserta un rango de elementos en la posición dada.** | **vec.insert(vec.begin() + 1, otro\_vec.begin(), otro\_vec.end())** |
| **clear()** | **Elimina todos los elementos del vector.** | **vec.clear()** |
| **swap(otro\_vector)** | **Intercambia el contenido del vector con otro vector.** | **vec.swap(otro\_vec)** |

### **10. Clase en C++**

**Definición:**

* **Una clase en C++ es una plantilla que define un tipo de dato compuesto. Permite encapsular datos y funciones en una única unidad, lo que facilita la modelización de entidades y el diseño orientado a objetos.**

**Estructura de una clase:**

* **Definición: Utiliza la palabra clave class seguida del nombre de la clase.**

**class NombreDeLaClase {**

**private:**

**// Miembros privados**

**TipoDeDato atributo;**

**public:**

**// Miembros públicos**

**void metodo();**

**};**

**Componentes:**

1. **Miembros de datos (atributos): Variables que almacenan el estado del objeto.**
2. **Métodos (funciones miembro): Funciones que definen el comportamiento del objeto.**
3. **Constructores y destructores: Métodos especiales que inicializan y limpian los objetos.**
4. **Modificadores de acceso: public, private, y protected determinan la visibilidad de los miembros.**

**Ejemplo:**

**class Coche {**

**private:**

**std::string marca;**

**int año;**

**public:**

**// Constructor**

**Coche(std::string m, int a) : marca(m), año(a) {}**

**// Método para mostrar información**

**void mostrarInfo() {**

**std::cout << "Marca: " << marca << ", Año: " << año << std::endl;**

**}**

**};**

**Objeto:**

* **Un objeto es una instancia de una clase. Es una entidad que encapsula tanto datos (atributos) como comportamiento (métodos). Los objetos se crean a partir de una clase y utilizan sus métodos y atributos para interactuar con el programa.**

**Propósito:**

* **Encapsulamiento: Agrupa datos y funciones que operan sobre esos datos, protegiendo el estado interno del objeto de modificaciones no autorizadas.**
* **Reutilización: Permite crear múltiples instancias de una clase sin necesidad de redefinir atributos y métodos.**
* **Modularidad: Facilita la organización y el mantenimiento del código al dividirlo en unidades independientes.**

**Ejemplo de uso de una clase y objetos:**

**int main() {**

**// Crear un objeto de la clase Coche**

**Coche miCoche("Toyota", 2020);**

**// Llamar a un método del objeto**

**miCoche.mostrarInfo();**

**return 0;**

**}**