Explicación de

int main(int argc, char* argv[])

1. Significado de argc y argv

argc (Argument Count): Es un entero que representa el número total de argumentos pasados al programa desde la línea de comandos. Siempre incluye el nombre del propio programa, así que el valor mínimo de argc es 1.

argv (Argument Vector): Es un array de cadenas (strings) que contiene los argumentos pasados al programa. argv[0] es el nombre del programa, y argv[1] hasta argv[argc-1] son los argumentos adicionales proporcionados.

Ejemplo de Uso Básico

Si tienes un programa llamado mi_programa y lo ejecutas desde la línea de comandos de la siguiente manera:

bash

```
./mi_programa argumento1 argumento2 argumento3
argc será 4 (el nombre del programa más tres argumentos).
argv será un array de cadenas:
argv[0] será "./mi_programa"
argv[1] será "argumento1"
argv[2] será "argumento2"
argv[3] será "argumento3"
```

Ejemplo Práctico de Carga, Modificación y Eliminación de Datos

Vamos a crear un programa que simule un sistema de gestión de datos a partir de los argumentos de la línea de comandos. En este caso, el programa permitirá cargar, modificar y eliminar entradas.

Código de Ejemplo

```
#include <iostream>
#include <vector>
#include <string>

// Función para mostrar el menú de opciones

void mostrarMenu() {
        std::cout << "Opciones:\n";
        std::cout << "1. Cargar datos\n";
        std::cout << "2. Modificar datos\n";
        std::cout << "3. Eliminar datos\n";
        std::cout << "4. Mostrar datos\n";
        std::cout << "5. Salir\n";
}

int main(int argc, char* argv[]) {
        // Vector para almacenar los datos
        std::vector<std::string> datos;

        // Verifica si hay argumentos suficientes

if (argc < 2) {
                std::cerr << "Uso: " << argv[0] << " <opcion> [<dato> ...]" << std::endl;
                return 1;
        }

        // La primera opción es el tipo de operación

int opcion = std::stoi(argv[1]);

       switch (opcion) {
                case 1: { // Cargar datos
```

```
                        for (int i = 2; i < argc; ++i) {
                                datos.push_back(argv[i]);
                        }
                        std::cout << "Datos cargados:\n";
                        for (const auto& dato : datos) {
                                std::cout << dato << std::endl;
                        }
                        break;
                }
                case 2: { // Modificar datos
                        if (argc < 4) {
                                std::cerr << "Uso para modificar: " << argv[0] << " 2 <indice>
<nuevo_dato>" << std::endl;
                                return 1;
                        }
                        int indice = std::stoi(argv[2]);
                        if (indice >= 0 && indice < datos.size()) {
                                datos[indice] = argv[3];
                                std::cout << "Datos modificados:\n";
                                for (const auto& dato : datos) {
                                        std::cout << dato << std::endl;
                                }
                        } else {
                                std::cerr << "Índice fuera de rango.\n";
                        }
                        break;
                }
                case 3: { // Eliminar datos
                        if (argc < 3) {
                                std::cerr << "Uso para eliminar: " << argv[0] << " 3 <indice>" <<
std::endl;
                                return 1;
                        }
                        int indice = std::stoi(argv[2]);
                        if (indice >= 0 && indice < datos.size()) {
                                datos.erase(datos.begin() + indice);
                                std::cout << "Datos después de la eliminación:\n";
                                for (const auto& dato : datos) {
                                        std::cout << dato << std::endl;
                                }
                        } else {
                                std::cerr << "Índice fuera de rango.\n";
                        }
                        break;
                }
                case 4: { // Mostrar datos
                        std::cout << "Datos actuales:\n";
                        for (const auto& dato : datos) {
                                std::cout << dato << std::endl;
                        }
                        break;
                }
                case 5: { // Salir
                        std::cout << "Saliendo...\n";
                        break;
                }
                default:
                        std::cerr << "Opción no válida.\n";
                        mostrarMenu();
                        return 1;
        }

        return 0;
}
```

Cómo Ejecutar el Programa

Cargar Datos:

bash

```
./mi_programa 1 "dato1" "dato2" "dato3"
```

Salida esperada:

```
Datos cargados:
dato1
dato2
dato3
```

Modificar Datos:

bash

```
./mi_programa 2 1 "nuevo_dato2"
```

Salida esperada:

Datos modificados:

```
dato1
nuevo_dato2
dato3
```

Eliminar Datos:

bash

```
./mi_programa 3 1
```

Salida esperada:

Datos después de la eliminación:

```
dato1
dato3
```

Mostrar Datos:

bash

```
./mi_programa 4
```

Salida esperada:

```
Datos actuales:
dato1
dato3
```

Consideraciones Adicionales

Manejo de Errores: Asegúrate de manejar los posibles errores, como la conversión de argumentos a enteros y la verificación de índices válidos.

Validación de Argumentos: Verifica que los argumentos sean los esperados antes de usarlos para evitar errores en tiempo de ejecución.