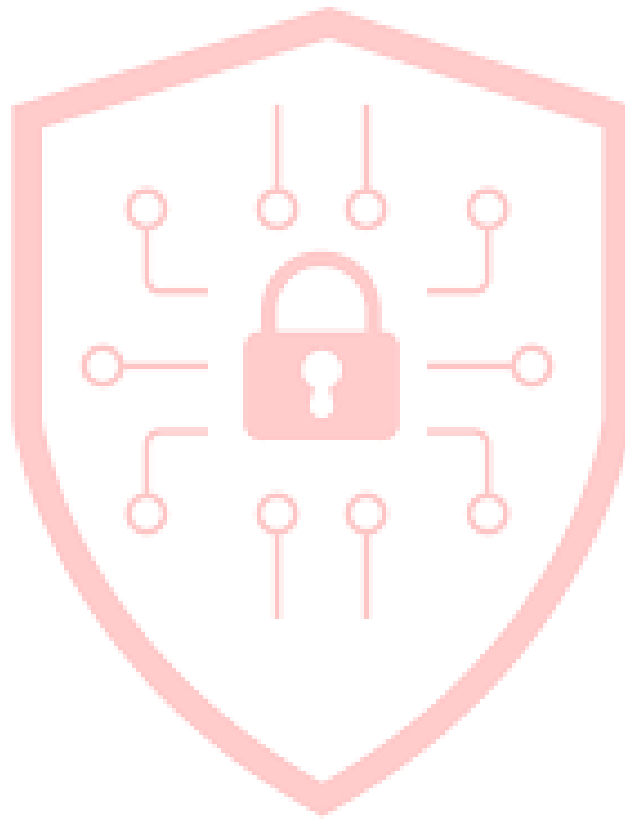




TECNOLÓGICO
NACIONAL DE MÉXICO



TECNOLÓGICO NACIONAL DE MEXICO
PLANTEL I "IZTAPALAPA"
INGENIERÍA EN SISTEMAS COMPUTACIONALES



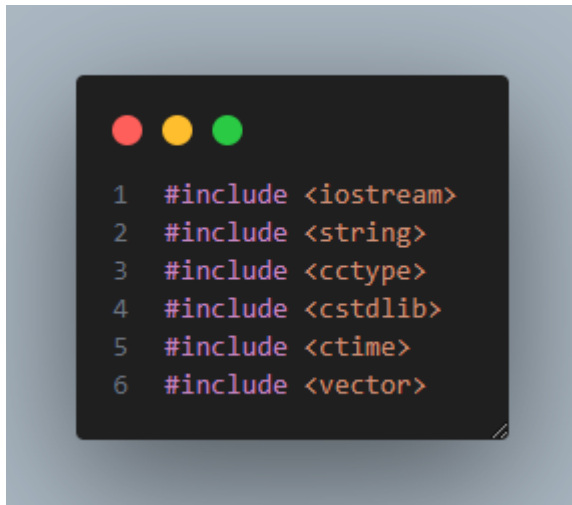
Materia > Estructura de Datos

Tutor > Juan Carlos Sotelo Olivera

Alumno > Ricardo Kalid Cabrera Castillo

Matricula > 221080451

Documentación



#include <iostream>: Importa la librería que permite realizar operaciones de entrada y salida en la consola (como `std::cout` y `std::cin`).

#include <string>: Importa la librería que permite el uso de la clase `std::string` para manejar cadenas de caracteres.

#include <cctype>: Importa la librería que proporciona funciones para la manipulación de caracteres, como `std::toupper` y `std::tolower`.

#include <cstdlib>: Importa la librería que permite el uso de funciones de generación de números aleatorios, como `rand()` y `srand()`.

#include <ctime>: Importa la librería que se utiliza para inicializar la semilla de números aleatorios con la función `time(0)`.

#include <vector>: Importa la librería que permite utilizar la estructura de datos `std::vector`, un contenedor dinámico para almacenar elementos.

```

1 // Funcion para obtener la primera vocal interna del apellido
2 std::string obtenerVocalInterna(const std::string& apellido) {
3     for (char c : apellido) {
4         if (std::tolower(c) != apellido[0] && (c == 'A' || c == 'E' || c == 'I' || c == 'O' || c == 'U')) {
5             return std::string(1, c);
6         }
7     }
8     return "X";
9 }

```

std::string obtenerVocalInterna(const std::string& apellido): Esta función toma como parámetro un apellido y devuelve la primera vocal interna (que no sea la primera letra) de ese apellido.

for (char c : apellido): Recorre cada carácter c en la cadena apellido.

if (std::tolower(c) != apellido[0] && (c == 'A' || c == 'E' || c == 'I' || c == 'O' || c == 'U')): Verifica si el carácter c es una vocal (mayúscula o minúscula) y si no es la primera letra del apellido.

return std::string(1, c);: Si encuentra una vocal interna, la devuelve como una cadena de un solo carácter.

return "X";: Si no encuentra ninguna vocal interna, devuelve "X".

```

1 char generarDigitoVerificador() {
2     int random = rand() % 36; // 0-9 + A-Z (10 números + 26 letras)
3     if (random < 10) {
4         return '0' + random; // Números
5     } else {
6         return 'A' + (random - 10); // Letras
7     }
8 }

```

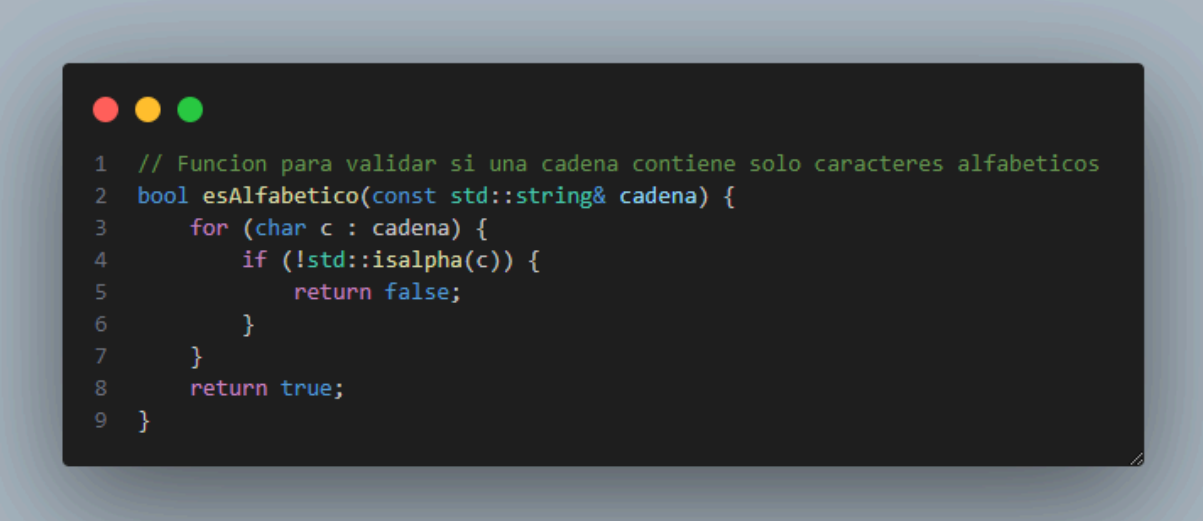
char generarDigitoVerificador(): Esta función genera un dígito verificador aleatorio (un número o una letra).

int random = rand() % 36;: Genera un número aleatorio entre 0 y 35.

if (random < 10): Si el número aleatorio es menor que 10, se trata de un dígito numérico.

`return '0' + random;`: Convierte `random` en un carácter numérico (del '0' al '9').

`return 'A' + (random - 10);`: Si `random` es 10 o mayor, convierte el valor en una letra (de 'A' a 'Z').

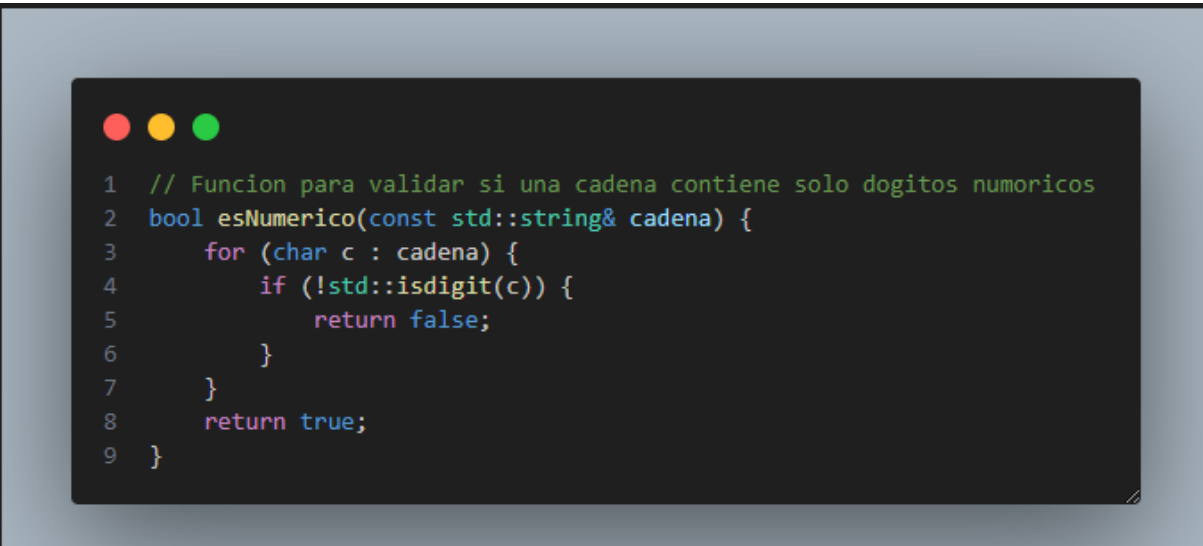


```
1 // Funcion para validar si una cadena contiene solo caracteres alfabeticos
2 bool esAlfabetico(const std::string& cadena) {
3     for (char c : cadena) {
4         if (!std::isalpha(c)) {
5             return false;
6         }
7     }
8     return true;
9 }
```

`bool esAlfabetico(const std::string& cadena)`: Esta función verifica si todos los caracteres de una cadena son alfabéticos.

`if (!std::isalpha(c))`: Comprueba si `c` no es un carácter alfabético. Si encuentra uno que no lo es, devuelve `false`.

`return true;`: Si todos los caracteres son alfabéticos, devuelve `true`.



```
1 // Funcion para validar si una cadena contiene solo digitos numericos
2 bool esNumerico(const std::string& cadena) {
3     for (char c : cadena) {
4         if (!std::isdigit(c)) {
5             return false;
6         }
7     }
8     return true;
9 }
```

`bool esNumerico(const std::string& cadena)`: Esta función verifica si todos los caracteres de una cadena son dígitos numéricos.

`if (!std::isdigit(c))`: Comprueba si `c` no es un dígito numérico. Si encuentra uno que no lo es, devuelve `false`.

`return true;`: Si todos los caracteres son numéricos, devuelve `true`.

```

1 // Funcion para desplegar el menu de seleccion de estado
2 std::string seleccionarEstado() {
3     std::vector<std::string> estados = {
4         "AS - Aguascalientes", "BC - Baja California", "BS - Baja California Sur", "CC - Campeche", "CL - Coahuila",
5         "CM - Colima", "CS - Chiapas", "CH - Chihuahua", "DF - Ciudad de Mexico", "DG - Durango", "GT - Guanajuato",
6         "GR - Guerrero", "HG - Hidalgo", "JC - Jalisco", "MC - Estado de Mexico", "MN - Michoacan", "MS - Morelos",
7         "NT - Nayarit", "NL - Nuevo Leon", "OC - Oaxaca", "PL - Puebla", "QT - Queretaro", "QR - Quintana Roo",
8         "SP - San Luis Potosi", "SL - Sinaloa", "SR - Sonora", "TC - Tabasco", "TS - Tamaulipas", "TL - Tlaxcala",
9         "VZ - Veracruz", "YN - Yucatan", "ZS - Zacatecas"
10    };

```

std::string seleccionarEstado(): Esta función despliega un menú para que el usuario seleccione su estado de nacimiento.

std::vector<std::string> estados = {...}; Se crea un vector de cadenas que contiene las claves y nombres de los estados de México.

for (size_t i = 0; i < estados.size(); ++i): Recorre el vector de estados y muestra cada opción numerada en pantalla.

```

1     int opcion = 0;
2     std::cout << "Seleccione su estado de nacimiento:\n";
3     for (size_t i = 0; i < estados.size(); ++i) {
4         std::cout << i + 1 << ". " << estados[i] << std::endl;
5     }
6
7     do {
8         std::cout << "Ingrese el numero correspondiente a su estado: ";
9         std::cin >> opcion;
10    } while (opcion < 1 || opcion > static_cast<int>(estados.size()));
11
12    return estados[opcion - 1].substr(0, 2); // Retorna la clave del estado (las primeras dos letras)
13 }

```

std::cin >> opcion; El usuario ingresa el número de la opción correspondiente al estado.

do { ... } while (opcion < 1 || opcion > static_cast<int>(estados.size())); Asegura que el usuario ingrese un número válido.

return estados[opcion - 1].substr(0, 2); Devuelve las dos primeras letras de la opción seleccionada, que es la clave del estado.

```

// Funcion para generar la CURP utilizando los datos del usuario
std::string generarCURP(const std::string& nombres, const std::string& primerApellido, const std::string& segundoApellido,
                        const std::string& fechaNacimiento, char sexo, const std::string& estado) {
    std::string curp;
    std::string primerNombre = nombres;

    // Si hay m s de un nombre, seleccionar el adecuado
    size_t espacio = nombres.find(' ');
    if (espacio != std::string::npos) {
        std::string primerNombreTemp = nombres.substr(0, espacio);
        std::string segundoNombre = nombres.substr(espacio + 1);

        // Si el primer nombre es "Jos " o "Mar a", usar el segundo nombre
        if (primerNombreTemp == "Jos " || primerNombreTemp == "Mar a") {
            primerNombre = segundoNombre;
        } else {
            primerNombre = primerNombreTemp;
        }
    }
}

```

std::string generarCURP(...): Esta funci n genera la CURP completa bas ndose en los datos proporcionados.

std::string primerNombre = nombres; Inicializa **primerNombre** con la cadena **nombres**.

size_t espacio = nombres.find(' '); Busca el primer espacio en la cadena de nombres para separar los nombres.

if (primerNombreTemp == "Jos " || primerNombreTemp == "Mar a" || primerNombreTemp == "MA" || primerNombreTemp == "J"): Si el primer nombre es com n (como Jos  o Mar a), toma el segundo nombre como el primero.

```

1  int main() {
2      srand(time(0)); // Inicializa la semilla para números aleatorios
3
4      std::string nombres, primerApellido, segundoApellido, fechaNacimiento;
5      char sexo;
6
7      // Menú para ingresar los datos con validación
8      do {
9          std::cout << "Ingrese su(s) nombre(s) completo(s) (solo letras): ";
10         std::getline(std::cin, nombres);
11     } while (!esAlfabetico(nombres));
12
13     do {
14         std::cout << "Ingrese su apellido paterno (solo letras): ";
15         std::getline(std::cin, primerApellido);
16     } while (!esAlfabetico(primerApellido));
17
18     do {
19         std::cout << "Ingrese su apellido materno (solo letras): ";
20         std::getline(std::cin, segundoApellido);
21     } while (!esAlfabetico(segundoApellido));
22
23     do {
24         std::cout << "Ingrese su fecha de nacimiento (AAMDD, solo números): ";
25         std::getline(std::cin, fechaNacimiento);
26     } while (!esNumerico(fechaNacimiento) || fechaNacimiento.length() != 6);
27
28     do {
29         std::cout << "Ingrese su sexo (H/M): ";
30         std::cin >> sexo;
31         sexo = std::toupper(sexo);
32     } while (sexo != 'H' && sexo != 'M');
33
34     // Seleccionar el estado de nacimiento
35     std::string estado = seleccionarEstado();
36
37     // Generar y mostrar la CURP
38     std::string curp = generarCURP(nombres, primerApellido, segundoApellido, fechaNacimiento, sexo, estado);
39     std::cout << "CURP generada: " << curp << std::endl;
40
41     return 0;
42 }

```

int main(): Función principal donde se ejecuta el programa.

std::getline(std::cin, nombres);: Captura el/los nombres ingresados por el usuario.

std::getline(std::cin, primerApellido);: Captura el primer apellido ingresado por el usuario.

std::getline(std::cin, segundoApellido);: Captura el segundo apellido ingresado por el usuario.

std::getline(std::cin, fechaNacimiento);: Captura la fecha de nacimiento en formato AAAA-MM-DD.

std::cin >> sexo;: Captura el sexo (Hombre/Mujer) del usuario.

std::cin.ignore();: Ignora el salto de línea pendiente en el búfer de entrada después de leer el sexo.

estado = seleccionarEstado();: Llama a la función **seleccionarEstado** para que el usuario seleccione su estado de nacimiento.

```
std::string curp = generarCURP(...);: Genera la CURP utilizando la
función generarCURP con los datos proporcionados.
std::cout << "Su CURP generada es: " << curp << std::endl;: Imprime
la CURP generada en la consola.
return 0;: Finaliza la ejecución del programa con un valor de retorno 0,
indicando que terminó correctamente.
```