**UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS**

**(Universidad del Perú, Decana de América)**

**FACULTAD DE INGENIERÍA DE SISTEMAS E INFORMÁTICA**

**ESCUELA ACADÉMICA PROFESIONAL DE INGENIERÍA DE SOFTWARE**

****

“La evolución de los microcontroladores”

**Curso**:  Estructura de datos

**Grupo 5:**

Bedia Gonzales Anthony Alonso

Collado Jimenez, Jhair Alejandro Ovenar

Alejandro Santaria, Gian Franco  
Sebastián Alonso Landeo Cuenta

2023

# **ÍNDICE**

[**ÍNDICE 2**](#_5el4ferv6r2g)

[**1. FUNCIONALIDAD: REGISTRO DE DNI 2**](#_dp0srdueq4au)

[1.1 ENTRADAS 3](#_x32b1xatgr15)

[1.2 RESULTADOS 3](#_20hhyf1om6sf)

[**1.3 CÓDIGO 3**](#_a7nda0k8v5h1)

[**2. GENERACIÓN DE NÚMERO DE DNI 4**](#_2nkemppgm63p)

[2.1 ENTRADAS 4](#_4ytaszu6v2za)

[2.2 RESULTADOS 5](#_tpyxlakhy653)

[**2.3 CÓDIGO 5**](#_3tymecdljh2c)

[**3. ACTUALIZACIÓN DE INFORMACIÓN 6**](#_vjvbhcpm2iiw)

[3.1 ENTRADAS 6](#_rphdfa9jlf7b)

[3.2 RESULTADOS 6](#_m1uegnvvyyib)

[**3.3 CÓDIGO 6**](#_mbpzldqxytj4)

[**4. CONSULTA DE INFORMACIÓN 6**](#_5wp631e1ldh2)

[4.1 ENTRADAS 6](#_8zb2cle59y3d)

[4.2 RESULTADOS 6](#_nw98o3zha8bz)

[**4.3 CÓDIGO 6**](#_v87v2w7gvqup)

[**5. ESTRUCTURAS DE DATOS 7**](#_wt0ku7ze9tba)

[**6. RESPONSABILIDADES 7**](#_le7l7djbju8d)

[**7. MENÚ DEL PROYECTO 8**](#_owg3uf7wku6)

[**8. BASE DE DATOS 8**](#_5lxyqc9azsju)

# **1. FUNCIONALIDAD: REGISTRO DE DNI**

El usuario al querer registrar su información en el DNI tendrá que ingresar ciertos datos que se indican a continuación:

1. Primer nombre:

2. Segundo nombre:

3. Primer apellido:

4. Segundo apellido:

5. Lugar nacimiento:

6. Nacionalidad:

7. Dirección de su vivienda:

8. Indique su sexo:

9. Indique su estado civil:

Al ingresar estos 9 datos se le aparecerá una resumen mostrando la información que ingresó.

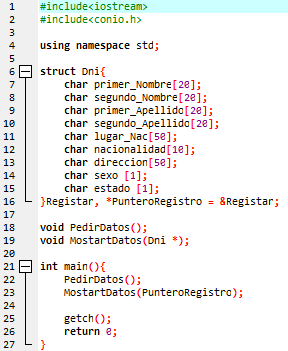
## **1.1 ENTRADAS**

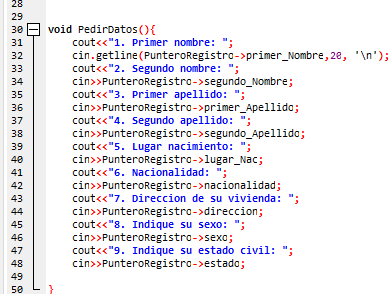
El usuario tendrá que ingresar sus nombres, apellidos, lugar de nacimiento, nacionalidad deireccion, su sexo y su esatdo civil actual.

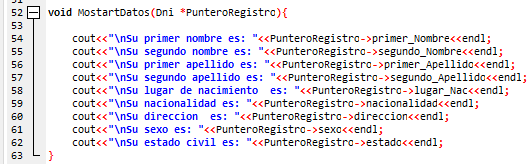
## **1.2 RESULTADOS**

Se mostrará al usuario los datos que ingresó.

## **1.3 CÓDIGO**







# **2. GENERACIÓN DE NÚMERO DE DNI**

La función generar DNI() se encarga de generar un número de 8 cifras con ciertas condiciones. La cifra más significativa de este número solo puede ser 1, 2, 3 o 4, mientras que las demás cifras no tienen restricciones. El algoritmo utiliza la función srand(time(0)) de C++ para obtener un número aleatorio, al cual se le calcula el resto de su división entre 9000000. A este resultado se le suma 1000000, obteniendo así un número de 7 cifras. Luego se genera otro número aleatorio, se lo divide por 4 y se calcula el resto, al cual se le suma 1. Este resultado es un número que puede ser 1, 2, 3 o 4, y se concatena como cifra más significativa al número de 7 cifras obtenido anteriormente.

## **2.1 ENTRADAS**

El programa se ejecuta 10 veces

## **2.2 RESULTADOS**

37632869

24156317

19765372

36153821

18127169

25203858

44994445

27451344

23262035

27709908

## **2.3 CÓDIGO**

#include <iostream>

#include <cstdlib>

#include <ctime>

#include <string>

using namespace std;

int generarDNI(){

// Semilla para generar números aleatorios

srand(time(0));

// Generar número aleatorio de 7 cifras

int numero = rand() % 9000000 + 1000000;

int numeroAleatorio = rand() % 4 + 1;

string numeroString = to\_string(numero);

string numeroDNI = to\_string(numeroAleatorio) + numeroString ;

return stoi(numeroDNI);

}

int main() {

cout << "Numero" << generarDNI() << endl;

return 0;

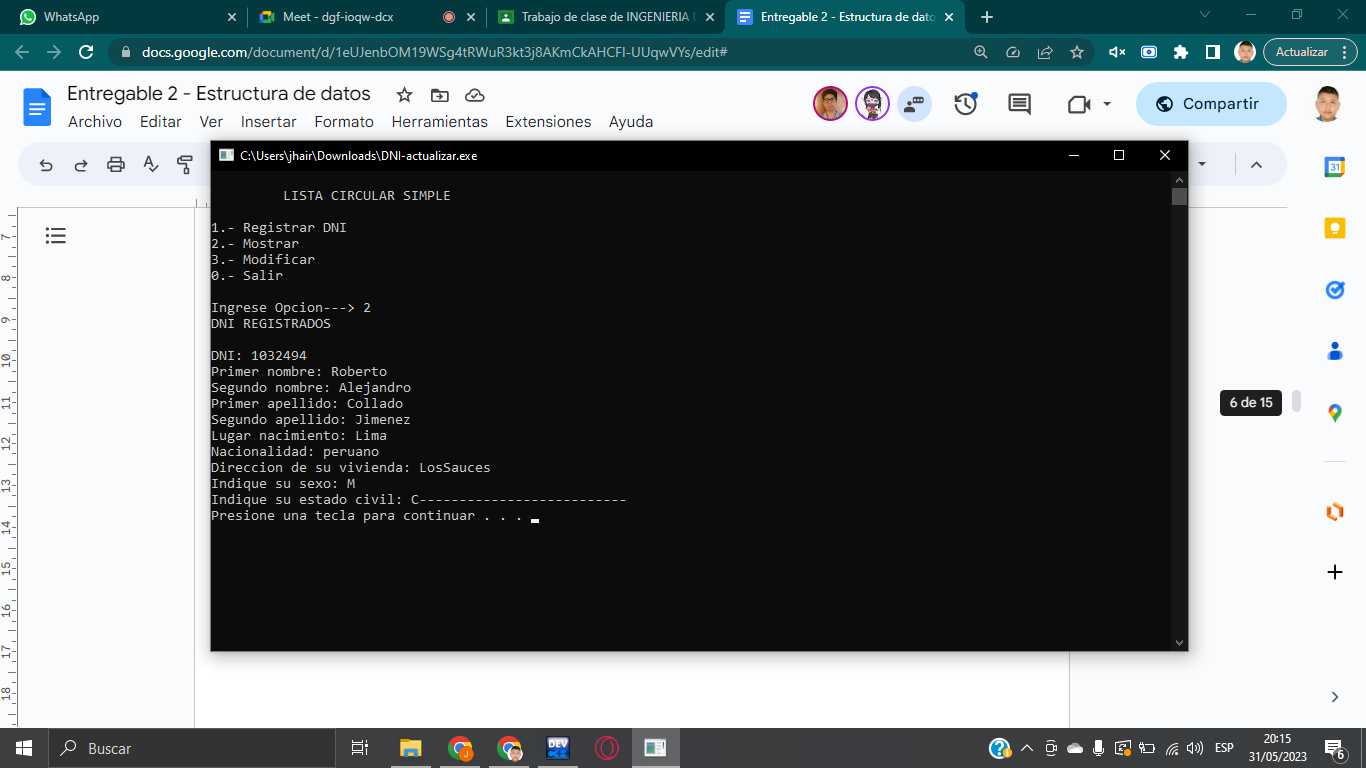
}

# **3. ACTUALIZACIÓN DE INFORMACIÓN**

## **3.1 ENTRADAS**

Ingresa el DNI que fue previamente registrado para la búsqueda y para su posterior actualización de los datos.

## **3.2 RESULTADOS**



## **3.3 CÓDIGO**

#include <iostream>

#include <conio.h>

#include <cstdlib>

#include <ctime>

#include <string>

using namespace std;

struct Dni{

string primer\_Nombre;

string segundo\_Nombre;

string primer\_Apellido;

string segundo\_Apellido;

string lugar\_Nac;

string nacionalidad;

string direccion;

string sexo;

string estado;

int numeroDNI;

struct Dni \*sgte;

};typedef struct Dni \*TpLista;

TpLista Registrar(){

TpLista nuevo=NULL;

nuevo=new(struct Dni);

srand(time(0));

int numero = rand() % 9000000 + 1000000;

int numeroAleatorio = rand() % 4 + 1;

nuevo->numeroDNI = numeroAleatorio + numero;

cout<<"\n\t Ingrese los datos solicitados\n\n";

cout<<"DNI: "<<nuevo->numeroDNI<<endl;

cout<<"1. Primer nombre: ";

cin>>nuevo->primer\_Nombre;

cout<<"2. Segundo nombre: ";

cin>>nuevo->segundo\_Nombre;

cout<<"3. Primer apellido: ";

cin>>nuevo->primer\_Apellido;

cout<<"4. Segundo apellido: ";

cin>>nuevo->segundo\_Apellido;

cout<<"5. Lugar nacimiento: ";

cin>>nuevo->lugar\_Nac;

cout<<"6. Nacionalidad: ";

cin>>nuevo->nacionalidad;

cout<<"7. Direccion de su vivienda: ";

cin>>nuevo->direccion;

cout<<"8. Indique su sexo: ";

cin>>nuevo->sexo;

cout<<"9. Indique su estado civil: ";

cin>>nuevo->estado;

cout<<endl;

nuevo->sgte=NULL;

return nuevo;

}

void InsertarDNI(TpLista &lista){

TpLista n=NULL;

n = Registrar();

if(lista != NULL){

n->sgte = lista;

}

lista=n;

}

void Mostrar(TpLista lista){

cout<<"DNI REGISTRADOS "<<endl<<endl;

while( lista != NULL){

cout<<"DNI: "<<lista->numeroDNI<<endl;

cout<<"Primer nombre: ";

cout<<lista->primer\_Nombre;

cout<<endl;

cout<<"Segundo nombre: ";

cout<<lista->segundo\_Nombre;

cout<<endl;

cout<<"Primer apellido: ";

cout<<lista->primer\_Apellido;

cout<<endl;

cout<<"Segundo apellido: ";

cout<<lista->segundo\_Apellido;

cout<<endl;

cout<<"Lugar nacimiento: ";

cout<<lista->lugar\_Nac;

cout<<endl;

cout<<"Nacionalidad: ";

cout<<lista->nacionalidad;

cout<<endl;

cout<<"Direccion de su vivienda: ";

cout<<lista->direccion;

cout<<endl;

cout<<"Indique su sexo: ";

cout<<lista->sexo;

cout<<endl;

cout<<"Indique su estado civil: ";

cout<<lista->estado;

cout<<"--------------------------";

cout<<endl;

lista=lista->sgte;

}

}

bool busquedaDNI(TpLista lista, int b){

TpLista p=lista;

bool flag=false;

while(p != NULL){

if(p->numeroDNI == b){

flag=true;

}

p=p->sgte;

}

if(flag==false){

cout<<"Valor buscado "<<b<<" no existe en la lista"<<endl;

system("pause");

}

return flag;

}

void Actualizar(TpLista lista, int b, string primer\_N, string segundo\_N, string primer\_A, string segundo\_A, string lugar\_Na, string nacion, string direc, string sex, string estad){

TpLista p=lista;

while(p != NULL){

if(p->numeroDNI == b){

p->primer\_Nombre=primer\_N;

p->segundo\_Nombre=segundo\_N;

p->primer\_Apellido=primer\_A;

p->segundo\_Apellido=segundo\_A;

p->lugar\_Nac=lugar\_Na;

p->nacionalidad=nacion;

p->direccion=direc;

p->sexo=sex;

p->estado=estad;

}

p=p->sgte;

}

}

void menu(){

system("cls");

cout<<"\n\t LISTA CIRCULAR SIMPLE\n\n";

cout<<"1.- Registrar DNI"<<endl;

cout<<"2.- Mostrar"<<endl;

cout<<"3.- Modificar"<<endl;

cout<<"0.- Salir"<<endl<<endl;

cout<<"Ingrese Opcion---> ";

}

int main(){

TpLista lista=NULL;

int opc, busca;

string reempNP, reempSN, reempPA, reempSA, reempLN, reempN, reempD, reempS, reempE;

do{

menu();

cin>>opc;

switch(opc){

case 1:{

InsertarDNI(lista);

system("pause");

break;

}

case 2:{

Mostrar(lista);

system("pause");

break;

}

case 3:{

if(lista == NULL){

cout<<"\nERROR.. La lista esta vacia.. no permite esta opcion "<<endl;

system("pause");

}

else{

cout<<"\nIngresar el DNI: ";cin>>busca;

if(busquedaDNI(lista, busca)==true){

cout<<"1. Primer nombre: ";

cin>>reempNP;

cout<<"2. Segundo nombre: ";

cin>>reempSN;

cout<<"3. Primer apellido: ";

cin>>reempPA;

cout<<"4. Segundo apellido: ";

cin>>reempSA;

cout<<"5. Lugar nacimiento: ";

cin>>reempLN;

cout<<"6. Nacionalidad: ";

cin>>reempN;

cout<<"7. Direccion de su vivienda: ";

cin>>reempD;

cout<<"8. Sexo: ";

cin>>reempS;

cout<<"9. Estado civil: ";

cin>> reempE;

Actualizar(lista, busca, reempNP, reempSN, reempPA, reempSA, reempLN, reempN, reempD, reempS, reempE);

}

else{

cout<<"\nERROR: No esta permitido ese valor"<<endl;

system("pause");

}

}

system("pause");

break;

}

}//switch

}while(opc != 0);

return 0;

}

# **4. CONSULTA DE INFORMACIÓN**

La función "consultar datos" en un sistema se refiere a la capacidad de acceder y obtener información específica almacenada en una base de datos o sistema de gestión. Esta función permite a los usuarios realizar consultas y obtener respuestas precisas y relevantes a partir de los datos disponibles.

## **4.1 ENTRADAS**

Número de DNI

## **4.2 RESULTADOS**

* Primer nombre
* Segundo nombre
* Primer apellido
* Segundo apellido
* Lugar nacimiento
* Nacionalidad
* Dirección de su vivienda
* Indique su sexo
* Indique su estado civil

## **4.3 CÓDIGO**

void ImprimirDNI(TpLista lista, int dni) {

TpLista p = lista;

bool encontrado = false;

while (p != NULL) {

if (p->numeroDNI == dni) {

cout << "DNI: " << p->numeroDNI << endl;

cout << "Primer nombre: " << p->primer\_Nombre << endl;

cout << "Segundo nombre: " << p->segundo\_Nombre << endl;

cout << "Primer apellido: " << p->primer\_Apellido << endl;

cout << "Segundo apellido: " << p->segundo\_Apellido << endl;

cout << "Lugar nacimiento: " << p->lugar\_Nac << endl;

cout << "Nacionalidad: " << p->nacionalidad << endl;

cout << "Direccion de su vivienda: " << p->direccion << endl;

cout << "Indique su sexo: " << p->sexo << endl;

cout << "Indique su estado civil: " << p->estado << endl;

encontrado = true;

break;

}

p = p->sgte;

}

if (!encontrado) {

cout << "El DNI ingresado no se encontró en la lista." << endl;

}

}

# **5. ESTRUCTURAS DE DATOS**

Para elaborar un sistema de registro de DNI (Documento Nacional de Identidad), una estructura dinámica lineal recomendada es la lista enlazada.

Una lista enlazada es una estructura de datos en la que cada elemento, llamado nodo, contiene tanto los datos como una referencia al siguiente nodo en la secuencia. En el caso del registro de DNI, cada nodo podría contener la información del individuo, como nombre, número de DNI, dirección, etc.

Las ventajas de utilizar una lista enlazada para este tipo de sistema son las siguientes:

Inserción eficiente: La lista enlazada permite una inserción eficiente de nuevos registros. Al ser una estructura dinámica, no es necesario tener un tamaño fijo predefinido, lo que facilita la incorporación de nuevos registros sin necesidad de reorganizar toda la estructura.

Eliminación flexible: Si se requiere eliminar un registro del sistema, la lista enlazada también permite una eliminación flexible. Solo es necesario actualizar las referencias de los nodos adyacentes, sin necesidad de desplazar elementos en la memoria.

Búsqueda y acceso secuencial: La lista enlazada es adecuada cuando se necesita acceder o buscar los registros de manera secuencial, ya que cada nodo contiene una referencia al siguiente nodo. Esto facilita recorrer todos los registros de forma ordenada.

Sin embargo, es importante tener en cuenta que la elección de la estructura de datos también depende de otros factores, como la cantidad de registros prevista, los requisitos de búsqueda y acceso, y las operaciones específicas que se realizan en el sistema de registro. En algunos casos, podría ser necesario considerar otras estructuras de datos, como árboles o tablas hash, para optimizar determinadas operaciones.

# **6. RESPONSABILIDADES**

Todo el grupo apoya en la parte del código, así que todos tienen por defecto el rol de programador como rol secundario.

**Sebastian Alonso Landeo Cuentas:** Analista de base de datos principal y programador.

**Gian Franco Alejandro Santaria:** Analista de base de datos secundario y programador.

**Anthony Bedia Gonzales:** Analista, diseñador y programador.

**Jhair Collado Jimenez:** Analista, diseñador y programador.

# **7. MENÚ DEL PROYECTO**

# **8. BASE DE DATOS**

-- MySQL Workbench Forward Engineering

SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0;

SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0;

SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION';

-- -----------------------------------------------------

-- Schema SITD

-- -----------------------------------------------------

-- -----------------------------------------------------

-- Schema SITD

-- -----------------------------------------------------

CREATE SCHEMA IF NOT EXISTS `SITD` DEFAULT CHARACTER SET utf8 ;

USE `SITD` ;

-- -----------------------------------------------------

-- Table `SITD`.`Usuario`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `SITD`.`Usuario` (

`ID\_Solictud` INT NOT NULL,

`primer\_nombre` VARCHAR(20) NOT NULL,

`segundo\_nombre` VARCHAR(20) NOT NULL,

`primer\_Apellido` VARCHAR(20) NOT NULL,

`segundo\_Apellido` VARCHAR(45) NOT NULL,

`fecha` DATE NOT NULL,

`lugar\_Nac` VARCHAR(50) NOT NULL,

`nacionalidad` VARCHAR(10) NOT NULL,

`direccion` VARCHAR(50) NOT NULL,

`sexo` VARCHAR(1) NOT NULL,

`estado` VARCHAR(1) NOT NULL,

PRIMARY KEY (`ID\_Solictud`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `SITD`.`DNI`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `SITD`.`DNI` (

`N\_dni` INT NOT NULL,

`ID\_Solictud\_2` INT NOT NULL,

PRIMARY KEY (`N\_dni`),

INDEX `fk\_DNI\_Usuario\_idx` (`ID\_Solictud\_2` ASC) VISIBLE,

CONSTRAINT `fk\_DNI\_Usuario`

FOREIGN KEY (`ID\_Solictud\_2`)

REFERENCES `SITD`.`Usuario` (`ID\_Solictud`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

SET SQL\_MODE=@OLD\_SQL\_MODE;

SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS;

SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS;

