Gonzalez Rhonal Final Lab. Programación

/\*

El centro vacunatorio.

Se nos solicita que realicemos un sistema que utilizará el personal de admisión a un centro

vacunatorio.

Cada vez que ingresa una persona, se registrará:

- Apellidos

- Nombres

- DNI

- fecha de nacimiento.

- Si ya está vacunado.

- Si es alérgico.

- Si está acompañado por un mayor.

A su vez, se cuenta con un conjunto de dosis de vacunas. Cada dosis contiene los siguientes

datos y son de única aplicación (al iniciar estan todas sin aplicar):

- Número de dosis.

- Número de lote.

- Laboratorio.

- Aplicada

Debemos realizar un programa que

a) vaya dando ingreso a cada persona al centro vacunatorio, permitiendo ingresar sus datos, y en

caso de ser admitida se le asigne una dosis. Los carga termina con un DNI igual a 0 y como

máximo se vacunan 500 personas por día. De cada vacunado se debe registrar:

- DNI

- Apellido

- Nombres

- Laboratorio

- Lote

- Dosis

b) al finalizar la carga de la jornada el sistema debe informar:

i) Cantidad de gente recibida.

ii) Cantidad de vacunados

iii) Cantidad de no admitidos.

iv) Promedio de edad de personas vacunadas.

c) A continuación de los datos del punto b) se debe imprimir el listado de vacunados ordenado por

DNI:

DNI APELLIDO NOMBRES

d) El sistema debe permitir (también al finalizar el ingreso de personas) la búsqueda por DNI,

informando si fue vacunado, y los datos de su dosis.

e) Realizar un procedimiento que reciba el conjunto con la información de vacunas aplicadas y

emita el siguiente listado:

LABORATORIO: XXXXX

LOTE DOSIS

9999 9

CANT APLICACIONES: 999

LABORATORIO: XXXXX

LOTE DOSIS

9999 9

CANT APLICACIONES: 999

Restricciones:

● Nuestro programa deberá admitir o denegar el ingreso según los siguientes criterios:

i.Las personas alérgicas con más de 70 años no pueden ser vacunadas.

ii.Ninguna persona ya vacunada debe ser vuelta a vacunar.

iii.Cuando la persona sea menor de edad debe estar acompañada por un mayor.

La mayoría de edad es de 18 años.

● En caso de acabarse las dosis, debe finalizar y pasar en forma automática al punto b.

● El menú para buscar, debe permitir realizar tantas búsquedas como el usuario quiera

antes de terminar el programa.

● El vector de vacunas solo puede ser recorrido secuencialmente una vez

\*/

#include <iostream>

using namespace std;

#define CANT\_VACUNAS 500

typedef struct

{

string apellido;

string nombre;

int dni;

int fechaNacimiento; //AAAAMMDD

bool vacunado;

bool alergico;

bool acompañadoPorMayor;

string laboratorio;

int nroLote;

int nroDosis;

} Persona;

typedef struct

{

int nroDosis;

int nroLote;

string laboratorio;

bool aplicada; //false inicialmente

} Vacuna;

typedef struct

{

int cantRecibidos;

int cantVacunados;

int cantNoAdmitidos;

float promedioEdadVacunados;

int fechaActual; //AAAAMMDD

} Registro;

void leer(string mensaje, int &var)

{

cout << mensaje;

cin >> var;

cout << endl;

return;

}

void leer(string mensaje, bool &var)

{

cout << mensaje;

cin >> var;

cout << endl;

return;

}

void leer(string mensaje, string &var)

{

cout << mensaje;

cin >> var;

cout << endl;

return;

}

/\*\*

\* Recibe una fecha AAAAMMDD y devuelve el año, mes y día

\*/

void convertirFecha(int fecha, int &anio, int &mes, int &dia)

{

anio = fecha / 10000;

mes = (fecha % 10000) / 100;

dia = fecha % 100;

return;

}

/\*\*

\* Calcula la edad de una persona a partir de la fechaNaci y la fecha actual

\* Utiliza la funcion convertirFecha()

\*/

void calcularEdad(int fechaActual, int fechaNacimiento, int &edad)

{

int diaActual, mesActual, anioActual;

int diaNac, mesNac, anioNac;

convertirFecha(fechaActual, anioActual, mesActual, diaActual);

convertirFecha(fechaNacimiento, anioNac, mesNac, diaNac);

edad = anioActual - anioNac;

if (mesActual < mesNac)

{

edad--;

}

else

{

if (mesActual == mesNac)

{

if (diaActual < diaNac)

{

edad--;

}

}

}

return;

}

//PARTE A

/\*\*

\* Registra una vez que la persona haya sido admitida

\*/

void registroVacuna(Persona personas[], int cantPersonas, Vacuna conjuntoVacunas[], int cantVacunas, Registro &registroDelDía, Persona &aux)

{

leer("Ingrese el apellido: ", aux.apellido);

leer("Ingrese el nombre: ", aux.nombre);

aux.laboratorio = conjuntoVacunas[registroDelDía.cantVacunados].laboratorio;

aux.nroLote = conjuntoVacunas[registroDelDía.cantVacunados].nroLote;

aux.nroDosis = conjuntoVacunas[registroDelDía.cantVacunados].nroDosis;

personas[registroDelDía.cantVacunados] = aux; //carga el vector de personas;

conjuntoVacunas[registroDelDía.cantVacunados].aplicada = true; //vacuna aplicada

registroDelDía.cantVacunados++; //contador de vacunados

cout << endl;

return;

}

/\*\*

\* Registra en el vector únicamente a las personas que pueden vacunarse

\* El resto no las admite ni registra.

\*/

void registrarPersonas(Persona personas[], int cantPersonas, Vacuna conjuntoVacunas[], int cantVacunas, Registro &registroDelDía)

{

Persona aux;

registroDelDía.cantVacunados = 0; //indice de los vectores

registroDelDía.cantNoAdmitidos = 0;

registroDelDía.cantRecibidos = 0;

registroDelDía.promedioEdadVacunados = 0;

int edad;

leer("Ingrese el DNI de la persona: ", aux.dni);

while (aux.dni != 0 && registroDelDía.cantVacunados < CANT\_VACUNAS)

{

registroDelDía.cantRecibidos++;

leer("¿Ya está vacunado? 1 para SI, 0 para NO", aux.vacunado);

if (!aux.vacunado)

{ //si no está ya vacunado

leer("Ingrese la edad en formato AAAAMMDD: ", aux.fechaNacimiento);

calcularEdad(registroDelDía.fechaActual, aux.fechaNacimiento, edad);

if (edad < 18)

{

leer("¿Está acompañado de un mayor? 1 para SI, 0 para NO", aux.acompañadoPorMayor);

if (!aux.acompañadoPorMayor)

{

cout << "No puede ingresar." << endl;

registroDelDía.cantNoAdmitidos++;

}

else

{ //Menor de edad pero acompañado

registroVacuna(personas, CANT\_VACUNAS, conjuntoVacunas, CANT\_VACUNAS, registroDelDía, aux);

registroDelDía.promedioEdadVacunados += edad;

}

}

else

{

if (edad >= 70)

{

leer("¿Es alérgico/a? 1 para SI, 0 para NO", aux.alergico);

if (aux.alergico)

{

cout << "No puede ingresar." << endl;

registroDelDía.cantNoAdmitidos++;

}

else

{ //Mayor de 70 y NO alergico

registroVacuna(personas, CANT\_VACUNAS, conjuntoVacunas, CANT\_VACUNAS, registroDelDía, aux);

registroDelDía.promedioEdadVacunados += edad;

}

}

else

{ //edad entre 18 y 70 años

registroVacuna(personas, CANT\_VACUNAS, conjuntoVacunas, CANT\_VACUNAS, registroDelDía, aux);

registroDelDía.promedioEdadVacunados += edad;

}

}

}

else

{

cout << "No puede ingresar." << endl;

registroDelDía.cantNoAdmitidos++;

}

leer("Ingrese el DNI de la siguiente persona: ", aux.dni);

}

if (registroDelDía.cantVacunados != 0)

{

registroDelDía.promedioEdadVacunados /= registroDelDía.cantVacunados;

}

else

{

cout << "No se puede calcular el promedio de edad de los vacunados." << endl

<< endl;

}

return;

}

//PARTE b

void imprimirJornada(Registro registroDelDía)

{

cout << "Cantidad de gente recibida: " << registroDelDía.cantRecibidos << endl;

cout << "Cantidad de vacunados: " << registroDelDía.cantVacunados << endl;

cout << "Cantidad de no admitidos: " << registroDelDía.cantNoAdmitidos << endl;

cout << "Promedio de edad de personas vacunadas: " << registroDelDía.promedioEdadVacunados << endl;

cout << endl;

}

// PARTE C

// A continuación de los datos del punto b) se debe imprimir el listado de vacunados ordenado por

// DNI:

// DNI APELLIDO NOMBRES

void ordenarPorDNI(Persona arr[], int n)

{

int i = 0, j;

Persona aux;

bool ordenado = false;

while (i < n && !ordenado)

{

ordenado = true;

for (j = 0; j < n - i - 1; j++)

{

if (arr[j].dni > arr[j + 1].dni)

{

aux = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = aux;

ordenado = false;

}

}

i++;

}

}

void imprimirVacunados(Persona personasVacunadas[], int cantPersonas)

{

cout << "Listado de personas vacunadas: " << endl

<< endl;

cout << "DNI \t\t APELLIDO \t\t NOMBRES" << endl;

for (int i = 0; i < cantPersonas; i++)

{

cout << personasVacunadas[i].dni << "\t\t" << personasVacunadas[i].apellido << "\t\t" << personasVacunadas[i].nombre << endl;

}

cout << endl;

return;

}

//PARTE D

// d) El sistema debe permitir (también al finalizar el ingreso de personas) la búsqueda por DNI,

// informando si fue vacunado, y los datos de su dosis.

int buscarIndiceDNI(Persona arr[], int cantElementos, int valorBuscado)

{

//Cuando arranco evalúo todo el vector de 0 a cantElementos-1

int inicio = 0;

int fin = cantElementos - 1;

while (fin >= inicio)

{

int mitad = inicio + (fin - inicio) / 2;

//Si el elemento es el del medio, devolvemos la posición

if (arr[mitad].dni == valorBuscado)

{

return mitad;

}

//Si el elemento es menor, entonces sólo puede estar en la 1ra mitad del array

if (arr[mitad].dni > valorBuscado)

{

fin = mitad - 1; //Cambio el límite superior

}

else

{

inicio = mitad + 1; //Cambio el límite inferior

}

}

return -1; //Si no se encuentra

}

void buscarPorDNI(Persona personasVacunadas[], int cantPersonas)

{

int valorBuscado, posicion;

leer("Ingrese el número de DNI que desea buscar (0 para terminar): ", valorBuscado);

while (valorBuscado != 0)

{

posicion = buscarIndiceDNI(personasVacunadas, cantPersonas, valorBuscado);

cout << "Vacunado\t\tnroDosis\t\tnroLote\tLaboratorio" << endl;

cout << "Sí \t\t" << personasVacunadas[posicion].nroDosis << "\t\t" << personasVacunadas[posicion].nroLote << "\t\t" << personasVacunadas[posicion].laboratorio << endl

<< endl;

cout << endl;

leer("Ingrese el número de DNI que desea buscar (0 para terminar): ", valorBuscado);

}

return;

}

//e

void imprimirVacunasAplicadas(Vacuna conjuntoVacunas[], int cantVacunas)

{

for (int i = 0; i < cantVacunas; i++)

{

if (conjuntoVacunas[i].aplicada)

{

cout << "LABORATORIO: " << conjuntoVacunas[i].laboratorio << endl;

cout << "LOTE: " << conjuntoVacunas[i].nroLote << endl;

cout << "DOSIS: " << conjuntoVacunas[i].nroDosis << endl;

cout << "CANT APLICACIONES: "

<< "1" << endl;

}

}

}

int main()

{

Persona personasVacunadas[CANT\_VACUNAS];

Vacuna conjuntoVacunas[CANT\_VACUNAS];

Registro registroDelDía;

//a

leer("Ingrese la fecha de hoy: ", registroDelDía.fechaActual);

registrarPersonas(personasVacunadas, CANT\_VACUNAS, conjuntoVacunas, CANT\_VACUNAS, registroDelDía);

//b

imprimirJornada(registroDelDía);

//c

ordenarPorDNI(personasVacunadas, registroDelDía.cantVacunados);

imprimirVacunados(personasVacunadas, registroDelDía.cantVacunados);

//d

buscarPorDNI(personasVacunadas, CANT\_VACUNAS);

//e

imprimirVacunasAplicadas(conjuntoVacunas, CANT\_VACUNAS);

return 0;

}

