Trabajo de IC2

Daniel Rodríguez Alonso, Sofia, Jaime

23 marzo 2024

Índice

1 Introducción

La practica titulada “Caso práctico en el desarrollo en C”, constituye una oportunidad fundamental para profundizar en diversos aspectos del lenguaje C. A lo largo de este ejercicio, los paraticipantes explorarán conceptos esenciales como punteros, vectores, struct y funciones, todos ellos pilares fundamentales en el dominio de la programación en C.

Esta práctica se presenta como un desafío integral, donde los participantes tendrán la oportunidad de poner en práctica sus conocimientos teóricos y habilidades técnicas adquiridas hasta el momento.

El desarrollo de la practica se dividirá en las siguientes actividades:

2 Declaración del vector grupo

Para la delcaración del vector grupo que tiene que ser de MAX 50 persona hemos creado directamente el vector utilizando el struct de Persona

```c
// Definición de la estructura
struct Persona
{
    int dni;
    char nombre[10];
    char apellidos[20];
    int numero_matricula;
} *ptr;
```

En la definicion de esta struct le definimos un dni, nombre, apellido, numero_matricula. Y al final de la definicon de ese struct declaramos un puntero global que va a tener acceso todo el programa y que solo va a poder apuntar a un struct de ese tipo.

Luego en el main declaramos en vector de grupo con maximo 50 alumnos.

```c
#define MAX 50

int main()
{
    struct Persona clase[MAX];
    ptr = clase;
    return 0;
}
```

Para definir el tamaño del vector utilizamos una constante que estara definida igualmente para todo el programa pero que no podrá cambiar de valor. En esta parte del desarrollo tambien le decimos al puntero *ptr* que va a apuntar a la primera posicion del vector clase.

Este codigo se puede representar graficamente a traves de PythonTutor.

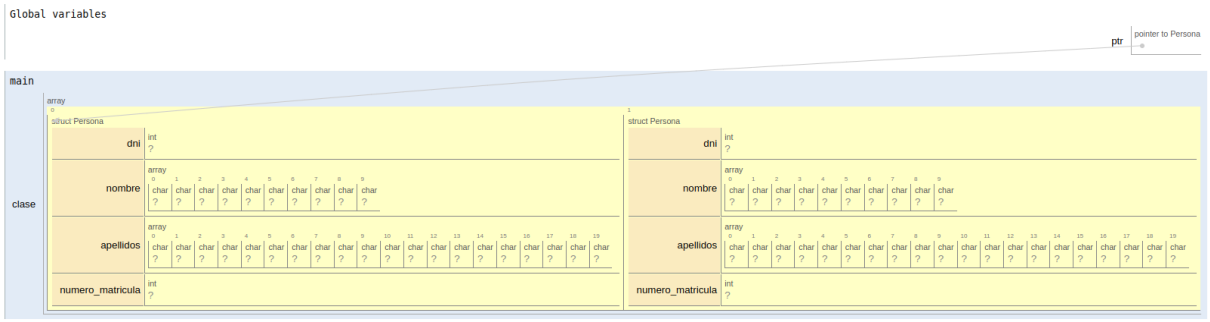

Figura 1: PythonTutor

Ahora pasamos al desarrollo de las funciones

3 Funciones

3.1 incializa_grupo

La primera funcion que desarrollamos es la de inicializa_grupo:

```
void inicializa_grupo()
{
    for (int i = 0; i < MAX; i++)
    {
        ptr[i].dni = -1;
    }
};
```

Esta funcion se utilizara nada mas ejecutar el prgrama para poder poner en todas las posiciones el dni = -1, esto nos indicará que no hay nadie matriculado en esa posición.

3.2 matricula_alumno

```
int matricular_alumno(int dni)
{
    // Comprobar si el ealumno ya esta matriculado
    for (int i = 0; i < MAX; i++)
    {
        if (ptr[i].dni == dni)
        {
            return -2;
        }
    }

    for (int i = 0; i < MAX; i++)
    {
```

```
        if (ptr[i].dni == -1)
        {
            ptr[i].dni = dni;
            return i;
        }
    }
    return -1;
}
```

Esta función servirá para poder asignarle una plaza a un alumno, en donde comprueba primero que no esta matriculado, que en caso de ser asi devolverá un -2, si el alumno no estaba matriculado buscará una posición donde el valor dni = -1 lo que significa que esa posición esta libre y pondrá el valor del dni que se quiere matricular en esa posicion, devolviendo la posición del dni guardado. En caso de que el vector se encuentre lleno y no se matriculará el alumno y la función devolverá un -1.

3.3 desmatricula_alumno

```
int desmatricular_alumno(int dni)
{
    for (int i = 0; i < MAX; i++)
    {
        if (ptr[i].dni == dni)
        {
            ptr[i].dni = -1;
            return i;
        }
    }
    return -1;
}
```

Al contrario de la funcion de matricular_alumno, esta función resive por parámetro un dni que buscara usando el puntero que apunta al vector de clase y cuando los encuentre pone el valor de dni de esa posicion a -1 lo que significa que queda libre esa posición

3.4 testea_alumnos

```
int testea_alumnos(int posicion)
{
    if (posicion < 0 || posicion >= MAX)
    {
        return -2;
    }
    return ptr[posicion].dni;
}
```

Esta función testea la posición de un alumno, esa posición se pasa por parametro que en caso de que esa posicion sea menor que 0 o mayor que el tamaño del vector devolvera un -2 ya que se esta intentando acceder a una posicion que no esta disponible, en caso que no se cumpla lo anterior la función devolverá el dni de la persona que este matriculada en esa posición o un -1 si esa posición esta libre.

3.5 plazas_libres

```
int plazas_libres()
{
    int count;
    count = 0;
    for (int i = 0; i < MAX; i++)
    {
        if (ptr[i].dni == -1)
        {
            count++;
        }
    }
    return count;
}
```

Esta función no resive ningun argumento por parametro, lo unico que hace es denifinir un contador y recorrer el vector de clase y va incrementando el contador cada vez que encuentre una posicion con el dni a -1.

3.6 plazas_ocupadas

```
int plazas_ocupadas()
{
    int count;
    count = 0;
    for (int i = 0; i < MAX; i++)
    {
        if (ptr[i].dni != -1)
        {
            count++;
        }
    }
    return count;
}
```

Esta funcion tiene la misma lógica que la función de plazas_libres lo unico que en cada iteración va comprobando si el dni es distinto a -1 lo que significará que la plaza esta ocupada y incrementaŕa el contador, cuando termine de iterar devuleve el contador.

4 Programa de prueba

En esta seccion del desarrollo crearemos funciones especificas para poder poner a prueba el funcionamiento de las funciones ya definidas y ejecutarlas con parametros no esperados por las funciones observando su comportamiento.

4.1 matricula

```
void test_matricula(int dni)
{
    for (int i = 0; i < MAX; i++)
    {
        int dni1;
        dni1 = dni + i;
        int resultado = matricular_alumno(dni1);
        if (resultado == -2)
        {
            printf("El alumno ya esta matriculado\n");
        }
        else if (resultado == -1)
        {
            printf("No hay plazas disponibles\n");
        }
        else
        {
            printf("El alumno con DNI %d ha sido matriculado en la posicion %d\n",
                   dni1, resultado);
        }
    }
}
```

El uso de esta funcion es para poder testear la funcion de matricula_alumno en donde realizamos una iteracion matriculando diversos dnis y dependiendo del resultado imprimimos por pantalla diferentes mensajes.

4.2 dematricula

```
void test_desmatricula(int dni)
{
    for (int i = 0; i < MAX; i++)
    {
        int dni1;
        dni1 = dni + i;
        int resultado = desmatricular_alumno(dni1);
        if (resultado == -1)
        {
            printf("El alumno no existe\n");
        }
        printf("El alumno con DNI %d ha sido desmatriculado de la posicion %d\n",
               dni1, resultado);
    }
}
```

Esta funcion testea la funcion de desmatricula_alumno llamandola un numero MAX de veces y pansandole un dni diferente en cada iteración. Cuando se llama la funcion test dependiendo del resultado imprime un mensaje u otro

4.3 matricula_multiple

```
void matricula_multiple(int npersonas, int dnis[])
{
    int plazas = plazas_libres();
    if (plazas < npersonas)
    {
        printf("No hay suficientes plazas libres");
        return;
    }

    // Matricular a cada persona de la lista
    for (int i = 0; i < npersonas; i++)
    {
        matricular_alumno(dnis[i]);
    }
    printf("Se han matriculado todos los alumnos");
}
```

matricula multiple es una funcion que trata de matricular a un conjunto de alumnos que se pasa por parametro y que solo los matriculara en caso de que hubiese le mismos de espacios libres que de alumnos que se quieren matricular en caso contrario no se matriculará a ninguno.

5 Main

Aqui mostraremos la parte del codigo que tiene el main()

```
int main()
{
    // Crear el vector de estructuras
    int dnis[2] = {1234567, 1324567};
    struct Persona clase[MAX];
    ptr = clase;
    inicializa_grupo();
    test_matricula(79072763);
    test_desmatricula(79072763);
    matricula_multiple(2, dnis);
    return 0;
}
```