

Trabajo de IC2

Daniel Rodríguez Alonso, Sofia Travieso García, Jaime Rivero Santana

05 mayo 2024

Índice

1	Introducción	3
2	Declaración del vector grupo	3
3	Funciones	4
3.1	inicializa_grupo	4
3.2	matricula_alumno	4
3.3	desmatricula_alumno	5
3.4	testea_alumnos	6
3.5	plazas_libres	6
3.6	plazas_ocupadas	6
4	Programa de prueba	7
4.1	matricula	7
4.2	dematricula	8
4.3	matricula_multiple	8
5	Main	9
6	Compilaciones y pruebas	9
7	División en módulos	15
8	Memoria dinámica	16
9	Conclusión	17

10 Actualización del Trabajo	17
10.1 Creación del test_grupo.c	17
10.2 Creando el test_grupo.h	20
10.3 Creando el grupo.c	21
10.4 Creamos el Makefile	21
10.5 Ejecuciones finales	22

1 Introducción

La práctica 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 participantes 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 del trabajo se divide de la siguiente forma entre los integrantes del grupo:

- **Daniel Rodríguez Alonso**

Se encargó del desarrollo de las funciones de test: “testea_matricula”, “testea_desmatricula” y “matricula multiple”. También de la definición del struct y compiló y ejecutó el programa cierto número de veces.

- **Sofía Travieso García**

Se encargó del desarrollo de las funciones: “matricula alumno”, “desmatricula alumno” y “testea_alumno”. Además, compiló y explicó las funcionalidades de las funciones en el PDF y desarrolló su maquetación.

- **Jaime Rivero Santana**

Participó en el desarrollo de las funciones: “inicializa grupo”, “plazas_libres” y “plazas_ocupadas”. Se encargó de cierta parte de la maquetación del trabajo, y compiló el programa ciertas veces para ver su funcionamiento.

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

2 Declaración del vector grupo

Para la declaración del vector “grupo”, que tiene que ser de MAX 50 personas, hemos creado directamente el vector utilizando el struct de Persona.

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

En la definición de este struct, declaramos: dni, nombre, apellido y número de matricula. Además, al final, definiremos un puntero global que va a tener acceso a todo el programa y que sólo va a poder apuntar a un struct de ese tipo.

Luego, en el main, declaramos un vector de grupo con máximo 50 alumnos.

```
#define MAX 50

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

Para definir el tamaño del vector utilizamos una constante que estará definida igualmente para todo el programa, por lo tanto, no podrá cambiar de valor. En esta parte del desarrollo, también le indicaremos al puntero “ptr” que apunte a la primera posición del vector clase.

Este código se puede representar gráficamente a través de PythonTutor:

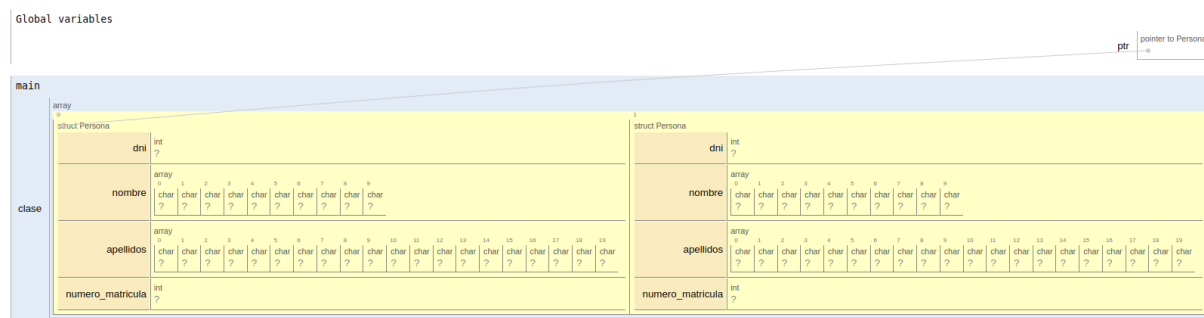


Figura 1: PythonTutor

Ahora pasamos al desarrollo de las funciones.

3 Funciones

3.1 inicializa_grupo

La primera función que desarrollamos es la de inicializa_grupo:

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

Esta función se utilizará para ejecutar el programa, poniendo 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 alumno ya esta matriculado
    for (int i = 0; i < MAX; i++)
    {
        if (ptr[i].dni == dni)
        {
            return -2;
        }
    }
    // Matriculando al alumno
    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, es decir, matricularlo. Primero se comprueba que la persona no está previamente matriculada que, de ser así, la función devolverá -2. Si el alumno no está matriculado, se buscará una posición del vector con dni = -1, lo que significa que esa posición está libre y el valor de dni cambiará a ser el dni del alumno. Además, devolverá su posición en el vector. En caso de que el vector se encuentre lleno, no se matricula el alumno y la función devuelve -1.

3.3 desmatricula_alumno

```

int desmatricular_alumno(int dni)
{
    // Buscando al alumno e iterando para desmatricularlo
    for (int i = 0; i < MAX; i++)
    {
        if (ptr[i].dni == dni)
        {
            ptr[i].dni = -1;
            return i;
        }
    }
    return -1;
}

```

Al contrario que la función “matricular_alumno”, esta función recibe por parámetro un dni que buscará usando el puntero que apunta al vector de clase. Cuando lo encuentre, pondrá el valor dni de esa posición a -1, lo que significa que queda libre.

3.4 testea_alumnos

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

Esta función testea una posición en el vector pasando por parámetro un número. Si el número es menor que 0 o mayor que 50, devuelve -2, ya que se está intentando acceder a una posición que no se encuentra disponible. En caso de que no se cumpla lo anterior, la función devolverá el dni de la persona que esté matriculada o un -1 si esa posición está 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 recibe ningún argumento por parámetro. Definimos un contador que recorre el vector de clase y se incrementa a medida que encuentra una posición 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 función tiene la misma lógica que la función “plazas_libres”. La diferencia es que, en cada iteración, va comprobando si el dni es distinto a -1. Esto significa que la plaza está ocupada y, por tanto, se incrementa el contador.

4 Programa de prueba

En esta sección del desarrollo, crearemos funciones específicas para poder poner a prueba el funcionamiento de las funciones definidas anteriormente. Las ejecutaremos con parámetros no esperados, observando su comportamiento.

4.1 matricula

```
void test_matricula(int dni)
{
    for (int i = 0; i < MAX + 2; 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);
        }
    }
    // Comprobando que el alumno ya esta matriculado
    for (int i = 0; i < 5; i++)
    {
        int dni1;
        dni1 = dni + i;
        int resultado = matricular_alumno(dni1);
        if (resultado == -2)
        {
            printf("El alumno ya esta matriculado\n");
        }
    }
}
```

El uso de esta función es poder testear la función “matricula_alumno” en donde realizamos una iteración matriculando diversos dnis. Dependiendo del resultado, imprimimos por pantalla diferentes mensajes.

4.2 dematricula

```
void test_desmatricula(int dni)
{
    for (int i = 0; i < MAX - 10; i++)
    {
        int dni1;
        dni1 = dni + i;
        int resultado = desmatricular_alumno(dni1);
        if (resultado == -1)
        {
            printf("El alumno no existe\n");
        } else {
            printf("El alumno con DNI %d ha sido desmatriculado de la posicion %d\n",
                dni1, resultado);
        }
    }
    for (int i = 0; i < 3; i++)
    {
        int dni1;
        dni1 = dni + i;
        int resultado = desmatricular_alumno(dni1);
        if (resultado == -1)
        {
            printf("El alumno no existe\n");
        }
    }
}
```

Esta función testea la función “desmatricula_alumno” llamándola un número MAX de veces y pasándole un dni diferente en cada iteración. Cuando se llama la función 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 \n");
}
```


“matricula_multiple” es una función que trata de matricular a un conjunto de alumnos que se pasan por parámetro. Sólo se matriculan en caso de que exista la misma cantidad de espacios libres que de alumnos que se quieren matricular. En caso contrario, no se matriculará a ninguno.

5 Main

Aquí se mostrará la parte del código que contiene 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);
    int resultado = plazas_libres();
    printf("Hay %d plazas libres\n", resultado);
    int resultado2 = plazas_ocupadas();
    printf("Hay %d plazas ocupadas\n", resultado2);
    printf("El alumno en la posición %d, tiene el dni = %d\n", 1, testea_alumnos(1));
    printf("El alumno en la posición %d, tiene el dni = %d\n", 5, testea_alumnos(5));
    printf("El alumno en la posición %d, tiene el dni = %d\n", 48, testea_alumnos(48));
    return 0;
}
```

6 Compilaciones y pruebas

En la compilación del código ejecutamos los siguientes comandos:

- Compilamos

```
gcc main.c -o programa
```

- Ejecutamos el programa

```
./programa
```

- Y visualizamos el resultado

```
daniel@daniel-msi:~/Escritorio/IC2_Practicas/trabajo$ gcc main.c -o programa
daniel@daniel-msi:~/Escritorio/IC2_Practicas/trabajo$ ./programa
El alumno con DNI 79072763 ha sido matriculado en la posicion 0
El alumno con DNI 79072764 ha sido matriculado en la posicion 1
El alumno con DNI 79072765 ha sido matriculado en la posicion 2
El alumno con DNI 79072766 ha sido matriculado en la posicion 3
El alumno con DNI 79072767 ha sido matriculado en la posicion 4
El alumno con DNI 79072768 ha sido matriculado en la posicion 5
El alumno con DNI 79072769 ha sido matriculado en la posicion 6
El alumno con DNI 79072770 ha sido matriculado en la posicion 7
El alumno con DNI 79072771 ha sido matriculado en la posicion 8
El alumno con DNI 79072772 ha sido matriculado en la posicion 9
El alumno con DNI 79072773 ha sido matriculado en la posicion 10
El alumno con DNI 79072774 ha sido matriculado en la posicion 11
El alumno con DNI 79072775 ha sido matriculado en la posicion 12
El alumno con DNI 79072776 ha sido matriculado en la posicion 13
El alumno con DNI 79072777 ha sido matriculado en la posicion 14
El alumno con DNI 79072778 ha sido matriculado en la posicion 15
El alumno con DNI 79072779 ha sido matriculado en la posicion 16
El alumno con DNI 79072780 ha sido matriculado en la posicion 17
El alumno con DNI 79072781 ha sido matriculado en la posicion 18
El alumno con DNI 79072782 ha sido matriculado en la posicion 19
El alumno con DNI 79072783 ha sido matriculado en la posicion 20
El alumno con DNI 79072784 ha sido matriculado en la posicion 21
El alumno con DNI 79072785 ha sido matriculado en la posicion 22
El alumno con DNI 79072786 ha sido matriculado en la posicion 23
El alumno con DNI 79072787 ha sido matriculado en la posicion 24
El alumno con DNI 79072788 ha sido matriculado en la posicion 25
El alumno con DNI 79072789 ha sido matriculado en la posicion 26
El alumno con DNI 79072790 ha sido matriculado en la posicion 27
El alumno con DNI 79072791 ha sido matriculado en la posicion 28
El alumno con DNI 79072792 ha sido matriculado en la posicion 29
El alumno con DNI 79072793 ha sido matriculado en la posicion 30
El alumno con DNI 79072794 ha sido matriculado en la posicion 31
El alumno con DNI 79072795 ha sido matriculado en la posicion 32
El alumno con DNI 79072796 ha sido matriculado en la posicion 33
El alumno con DNI 79072797 ha sido matriculado en la posicion 34
El alumno con DNI 79072798 ha sido matriculado en la posicion 35
El alumno con DNI 79072799 ha sido matriculado en la posicion 36
El alumno con DNI 79072800 ha sido matriculado en la posicion 37
El alumno con DNI 79072801 ha sido matriculado en la posicion 38
El alumno con DNI 79072802 ha sido matriculado en la posicion 39
El alumno con DNI 79072803 ha sido matriculado en la posicion 40
El alumno con DNI 79072804 ha sido matriculado en la posicion 41
El alumno con DNI 79072805 ha sido matriculado en la posicion 42
El alumno con DNI 79072806 ha sido matriculado en la posicion 43
El alumno con DNI 79072807 ha sido matriculado en la posicion 44
El alumno con DNI 79072808 ha sido matriculado en la posicion 45
El alumno con DNI 79072809 ha sido matriculado en la posicion 46
El alumno con DNI 79072810 ha sido matriculado en la posicion 47
El alumno con DNI 79072811 ha sido matriculado en la posicion 48
El alumno con DNI 79072812 ha sido matriculado en la posicion 49
El alumno con DNI 79072763 ha sido desmatriculado de la posicion 0
El alumno con DNI 79072764 ha sido desmatriculado de la posicion 1
El alumno con DNI 79072765 ha sido desmatriculado de la posicion 2
El alumno con DNI 79072766 ha sido desmatriculado de la posicion 3
El alumno con DNI 79072767 ha sido desmatriculado de la posicion 4
El alumno con DNI 79072768 ha sido desmatriculado de la posicion 5
El alumno con DNI 79072769 ha sido desmatriculado de la posicion 6
El alumno con DNI 79072770 ha sido desmatriculado de la posicion 7
```

Figura 2: Terminal

Vemos que al probar las funciones “testea_martrricula” y “testea_desmatricula”, funcionan perfectamente, rellena el vector clase y luego los desmatricula.

Ahora vamos a intentar compilar el programa matriculando más de 50 alumnos y no desmatriculándolos todos para ver si funciona el código y las funciones “plazas_libres” y “plazas_ocupadas”.

```
daniel@daniel-msi:~/Escritorio/IC2_Practicas/trabajo$ gcc main.c -o programa
main.c: In function 'main':
main.c:160:9: error: redefinition of 'resultado'
160 |     int resultado = plazas_ocupadas();
    |         ^~~~~~
main.c:158:9: note: previous definition of 'resultado' with type 'int'
158 |     int resultado = plazas_libres();
    |         ^~~~~~
void multiple(int npersonas, int dias[])
```

Figura 3: Error

Al intentar compilar este programa vemos que nos salta un error que es bastante sencillo de solucionar. Se trata de que, en el apartado de main() de nuestro trabajo, duplicamos la variable de resultado 2 veces. Cambiando el nombre a esta variable solucionamos el problema y podremos compilar el programa.

- Antes

```
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);
    int resultado = plazas_libres();
    printf("Hay %d plazas libres\n", resultado);
    int resultado = plazas_ocupadas();
    printf("Hay %d plazas ocupadas\n", resultado);
    return 0;
}
```

- Despues

```
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);
    int resultado = plazas_libres();
    printf("Hay %d plazas libres\n", resultado);
    int resultado2 = plazas_ocupadas(); // Cambio de resultado -> resultado2
    printf("Hay %d plazas ocupadas\n", resultado);
    return 0;
}
```

```
El alumno con DNI 79072800 ha sido matriculado en la posicion 37
El alumno con DNI 79072801 ha sido matriculado en la posicion 38
El alumno con DNI 79072802 ha sido matriculado en la posicion 39
El alumno con DNI 79072803 ha sido matriculado en la posicion 40
El alumno con DNI 79072804 ha sido matriculado en la posicion 41
El alumno con DNI 79072805 ha sido matriculado en la posicion 42
El alumno con DNI 79072806 ha sido matriculado en la posicion 43
El alumno con DNI 79072807 ha sido matriculado en la posicion 44
El alumno con DNI 79072808 ha sido matriculado en la posicion 45
El alumno con DNI 79072809 ha sido matriculado en la posicion 46
El alumno con DNI 79072810 ha sido matriculado en la posicion 47
El alumno con DNI 79072811 ha sido matriculado en la posicion 48
El alumno con DNI 79072812 ha sido matriculado en la posicion 49
No hay plazas disponibles
No hay plazas disponibles
El alumno con DNI 79072763 ha sido desmatriculado de la posicion 0
El alumno con DNI 79072764 ha sido desmatriculado de la posicion 1
El alumno con DNI 79072765 ha sido desmatriculado de la posicion 2
El alumno con DNI 79072766 ha sido desmatriculado de la posicion 3
El alumno con DNI 79072767 ha sido desmatriculado de la posicion 4
El alumno con DNI 79072768 ha sido desmatriculado de la posicion 5
El alumno con DNI 79072769 ha sido desmatriculado de la posicion 6
El alumno con DNI 79072770 ha sido desmatriculado de la posicion 7
El alumno con DNI 79072771 ha sido desmatriculado de la posicion 8
El alumno con DNI 79072772 ha sido desmatriculado de la posicion 9
El alumno con DNI 79072773 ha sido desmatriculado de la posicion 10
El alumno con DNI 79072774 ha sido desmatriculado de la posicion 11
El alumno con DNI 79072775 ha sido desmatriculado de la posicion 12
El alumno con DNI 79072776 ha sido desmatriculado de la posicion 13
El alumno con DNI 79072777 ha sido desmatriculado de la posicion 14
El alumno con DNI 79072778 ha sido desmatriculado de la posicion 15
El alumno con DNI 79072779 ha sido desmatriculado de la posicion 16
El alumno con DNI 79072780 ha sido desmatriculado de la posicion 17
El alumno con DNI 79072781 ha sido desmatriculado de la posicion 18
El alumno con DNI 79072782 ha sido desmatriculado de la posicion 19
El alumno con DNI 79072783 ha sido desmatriculado de la posicion 20
El alumno con DNI 79072784 ha sido desmatriculado de la posicion 21
El alumno con DNI 79072785 ha sido desmatriculado de la posicion 22
El alumno con DNI 79072786 ha sido desmatriculado de la posicion 23
El alumno con DNI 79072787 ha sido desmatriculado de la posicion 24
El alumno con DNI 79072788 ha sido desmatriculado de la posicion 25
El alumno con DNI 79072789 ha sido desmatriculado de la posicion 26
El alumno con DNI 79072790 ha sido desmatriculado de la posicion 27
El alumno con DNI 79072791 ha sido desmatriculado de la posicion 28
El alumno con DNI 79072792 ha sido desmatriculado de la posicion 29
El alumno con DNI 79072793 ha sido desmatriculado de la posicion 30
El alumno con DNI 79072794 ha sido desmatriculado de la posicion 31
El alumno con DNI 79072795 ha sido desmatriculado de la posicion 32
El alumno con DNI 79072796 ha sido desmatriculado de la posicion 33
El alumno con DNI 79072797 ha sido desmatriculado de la posicion 34
El alumno con DNI 79072798 ha sido desmatriculado de la posicion 35
El alumno con DNI 79072799 ha sido desmatriculado de la posicion 36
El alumno con DNI 79072800 ha sido desmatriculado de la posicion 37
El alumno con DNI 79072801 ha sido desmatriculado de la posicion 38
El alumno con DNI 79072802 ha sido desmatriculado de la posicion 39
Se han matriculado todos los alumnos.
Hay 38 plazas libres
Hay 38 plazas ocupadas
daniel@danield-m2:~/Escritorio/IC2_Practicas/trabajo5$
```

Figura 4: Terminal

En esta compilación, el programa funciona pero nos aparece que hay el mismo número de plazas ocupadas que de plazas libres. Nos dimos cuenta del problema que se encontraba en el main():

- Antes

```
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);
    int resultado = plazas_libres();
    printf("Hay %d plazas libres\n", resultado);
    int resultado2 = plazas_ocupadas();
    printf("Hay %d plazas ocupadas\n", resultado);
    return 0;
}
```

- Después

```
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);
int resultado = plazas_libres();
printf("Hay %d plazas libres\n", resultado);
int resultado2 = plazas_ocupadas();
// Cambiamos resultado -> resultado2
printf("Hay %d plazas ocupadas\n", resultado2);
return 0;
}

```

El problema se encuentra en que, en el segundo printf, imprimimos la misma variable. Por eso, el resultado es dos veces el mismo. Cambiando eso, se solucionaría el problema.

Volvemos a compilar el programa y vemos cómo esta vez el resultado sí es correcto. Se matriculan 52 alumnos y, cuando llega a su límite, salta el mensaje: “No hay plazas disponibles”. Consecutivamente se desmatriculan 40 alumnos. Cuando terminan esos procesos, se matriculan dos alumnos con la función “matricula_multiple()” y salta el mensaje: “Se han matriculado todos los alumnos”. Por último, saltan los mensajes: “Hay 38 plazas libres” y “Hay 12 plazas ocupadas” teniendo todo el sentido ya que $38 + 12 = 50$.

```

El alumno con DNI 79072800 ha sido matriculado en la posicion 37
El alumno con DNI 79072801 ha sido matriculado en la posicion 38
El alumno con DNI 79072802 ha sido matriculado en la posicion 39
El alumno con DNI 79072803 ha sido matriculado en la posicion 40
El alumno con DNI 79072804 ha sido matriculado en la posicion 41
El alumno con DNI 79072805 ha sido matriculado en la posicion 42
El alumno con DNI 79072806 ha sido matriculado en la posicion 43
El alumno con DNI 79072807 ha sido matriculado en la posicion 44
El alumno con DNI 79072808 ha sido matriculado en la posicion 45
El alumno con DNI 79072809 ha sido matriculado en la posicion 46
El alumno con DNI 79072810 ha sido matriculado en la posicion 47
El alumno con DNI 79072811 ha sido matriculado en la posicion 48
El alumno con DNI 79072812 ha sido matriculado en la posicion 49
No hay plazas disponibles
No hay plazas disponibles
El alumno con DNI 79072763 ha sido desmatriculado de la posicion 0
El alumno con DNI 79072764 ha sido desmatriculado de la posicion 1
El alumno con DNI 79072765 ha sido desmatriculado de la posicion 2
El alumno con DNI 79072766 ha sido desmatriculado de la posicion 3
El alumno con DNI 79072767 ha sido desmatriculado de la posicion 4
El alumno con DNI 79072768 ha sido desmatriculado de la posicion 5
El alumno con DNI 79072769 ha sido desmatriculado de la posicion 6
El alumno con DNI 79072770 ha sido desmatriculado de la posicion 7
El alumno con DNI 79072771 ha sido desmatriculado de la posicion 8
El alumno con DNI 79072772 ha sido desmatriculado de la posicion 9
El alumno con DNI 79072773 ha sido desmatriculado de la posicion 10
El alumno con DNI 79072774 ha sido desmatriculado de la posicion 11
El alumno con DNI 79072775 ha sido desmatriculado de la posicion 12
El alumno con DNI 79072776 ha sido desmatriculado de la posicion 13
El alumno con DNI 79072777 ha sido desmatriculado de la posicion 14
El alumno con DNI 79072778 ha sido desmatriculado de la posicion 15
El alumno con DNI 79072779 ha sido desmatriculado de la posicion 16
El alumno con DNI 79072780 ha sido desmatriculado de la posicion 17
El alumno con DNI 79072781 ha sido desmatriculado de la posicion 18
El alumno con DNI 79072782 ha sido desmatriculado de la posicion 19
El alumno con DNI 79072783 ha sido desmatriculado de la posicion 20
El alumno con DNI 79072784 ha sido desmatriculado de la posicion 21
El alumno con DNI 79072785 ha sido desmatriculado de la posicion 22
El alumno con DNI 79072786 ha sido desmatriculado de la posicion 23
El alumno con DNI 79072787 ha sido desmatriculado de la posicion 24
El alumno con DNI 79072788 ha sido desmatriculado de la posicion 25
El alumno con DNI 79072789 ha sido desmatriculado de la posicion 26
El alumno con DNI 79072790 ha sido desmatriculado de la posicion 27
El alumno con DNI 79072791 ha sido desmatriculado de la posicion 28
El alumno con DNI 79072792 ha sido desmatriculado de la posicion 29
El alumno con DNI 79072793 ha sido desmatriculado de la posicion 30
El alumno con DNI 79072794 ha sido desmatriculado de la posicion 31
El alumno con DNI 79072795 ha sido desmatriculado de la posicion 32
El alumno con DNI 79072796 ha sido desmatriculado de la posicion 33
El alumno con DNI 79072797 ha sido desmatriculado de la posicion 34
El alumno con DNI 79072798 ha sido desmatriculado de la posicion 35
El alumno con DNI 79072799 ha sido desmatriculado de la posicion 36
El alumno con DNI 79072800 ha sido desmatriculado de la posicion 37
El alumno con DNI 79072801 ha sido desmatriculado de la posicion 38
El alumno con DNI 79072802 ha sido desmatriculado de la posicion 39
Se han matriculado todos los alumnos
Hay 38 plazas libres
Hay 12 plazas ocupadas
daniel@daniel-wsi:~/Escritorio/IC2 Practicas/trabajo$

```

Figura 5: Terminal

En la última compilación vamos a probar la función “testea_alumno” en la que le pasas una posición y te devuelve el dni de la persona en esa posición.

Mostramos las líneas de código que ejecutaran la función varias veces:

```

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);
    int resultado = plazas_libres();
    printf("Hay %d plazas libres\n", resultado);
    int resultado2 = plazas_ocupadas();
    //----- testea_alumnos-----
    printf("Hay %d plazas ocupadas\n", resultado2);
    printf("El alumno en la posición %d, tiene el dni = %d\n", 1, testea_alumnos(1));
    printf("El alumno en la posición %d, tiene el dni = %d\n", 5, testea_alumnos(5));
    printf("El alumno en la posición %d, tiene el dni = %d\n", 48, testea_alumnos(48));
    //----- fin testea_alumnos-----
    return 0;
}

```

```

El alumno con DNI 79072786 ha sido desmatriculado de la posicion 23
El alumno con DNI 79072787 ha sido desmatriculado de la posicion 24
El alumno con DNI 79072788 ha sido desmatriculado de la posicion 25
El alumno con DNI 79072789 ha sido desmatriculado de la posicion 26
El alumno con DNI 79072790 ha sido desmatriculado de la posicion 27
El alumno con DNI 79072791 ha sido desmatriculado de la posicion 28
El alumno con DNI 79072792 ha sido desmatriculado de la posicion 29
El alumno con DNI 79072793 ha sido desmatriculado de la posicion 30
El alumno con DNI 79072794 ha sido desmatriculado de la posicion 31
El alumno con DNI 79072795 ha sido desmatriculado de la posicion 32
El alumno con DNI 79072796 ha sido desmatriculado de la posicion 33
El alumno con DNI 79072797 ha sido desmatriculado de la posicion 34
El alumno con DNI 79072798 ha sido desmatriculado de la posicion 35
El alumno con DNI 79072799 ha sido desmatriculado de la posicion 36
El alumno con DNI 79072800 ha sido desmatriculado de la posicion 37
El alumno con DNI 79072801 ha sido desmatriculado de la posicion 38
El alumno con DNI 79072802 ha sido desmatriculado de la posicion 39
Se han matriculado todos los alumnos
Hay 38 plazas libres ("Se han matriculado todos los alumnos \n");
Hay 12 plazas ocupadas
El alumno en la posición 1, tiene el dni = 1324567
El alumno en la posición 5, tiene el dni = -1
El alumno en la posición 48, tiene el dni = 79072811

```

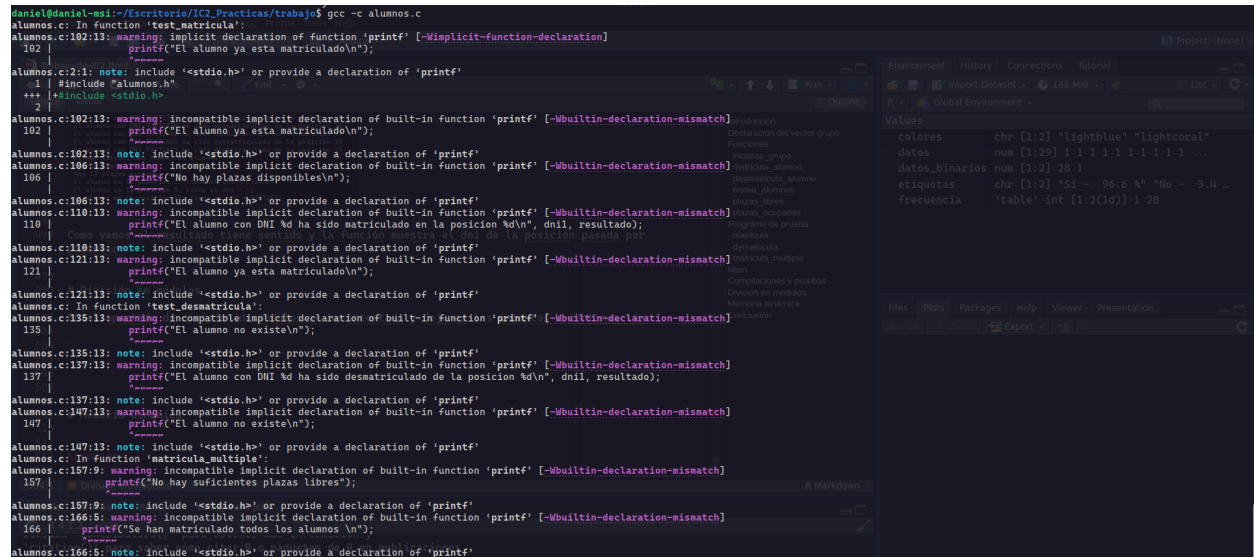
Figura 6: Terminal

Como vemos, el resultado tiene sentido y la función muestra el dni de la posición pasada por parámetro.

7 División en módulos

El apartado de división de módulos se ha intentado y dejamos los intentos a continuación:

```
gcc -c alumnos.c
```



```
daniel@daniel-msi:~/Escritorio/IC2_Practicas/trabajo$ gcc -c alumnos.c
alumnos.c: In function 'test_matricula':
alumnos.c:102:13: warning: implicit declaration of function 'printf' [-Wimplicit-function-declaration]
102 |         printf("El alumno ya esta matriculado\n");
    |         ^~~~~~
alumnos.c:2:1: note: include '<stdio.h>' or provide a declaration of 'printf'
1 | #include "alumnos.h"
+++ | #include "<stdio.h>"
2 |
alumnos.c:102:13: warning: incompatible implicit declaration of built-in function 'printf' [-Wbuiltin-declaration-mismatch]
102 |         printf("El alumno ya esta matriculado\n");
    |         ^~~~~~
alumnos.c:102:13: note: include '<stdio.h>' or provide a declaration of 'printf'
alumnos.c:106:13: warning: incompatible implicit declaration of built-in function 'printf' [-Wbuiltin-declaration-mismatch]
106 |         printf("No hay plazas disponibles\n");
    |         ^~~~~~
alumnos.c:106:13: note: include '<stdio.h>' or provide a declaration of 'printf'
alumnos.c:110:13: warning: incompatible implicit declaration of built-in function 'printf' [-Wbuiltin-declaration-mismatch]
110 |         printf("El alumno con DNI %d ha sido matriculado en la posicion %d\n", dni1, resultado);
    |         ^~~~~~
alumnos.c:110:13: note: include '<stdio.h>' or provide a declaration of 'printf'
alumnos.c:121:13: warning: incompatible implicit declaration of built-in function 'printf' [-Wbuiltin-declaration-mismatch]
121 |         printf("El alumno ya esta matriculado\n");
    |         ^~~~~~
alumnos.c:121:13: note: include '<stdio.h>' or provide a declaration of 'printf'
alumnos.c:135:13: warning: incompatible implicit declaration of built-in function 'printf' [-Wbuiltin-declaration-mismatch]
135 |         printf("El alumno no existe\n");
    |         ^~~~~~
alumnos.c:135:13: note: include '<stdio.h>' or provide a declaration of 'printf'
alumnos.c:137:13: warning: incompatible implicit declaration of built-in function 'printf' [-Wbuiltin-declaration-mismatch]
137 |         printf("El alumno con DNI %d ha sido desmatriculado de la posicion %d\n", dni1, resultado);
    |         ^~~~~~
alumnos.c:137:13: note: include '<stdio.h>' or provide a declaration of 'printf'
alumnos.c:147:13: warning: incompatible implicit declaration of built-in function 'printf' [-Wbuiltin-declaration-mismatch]
147 |         printf("El alumno no existe\n");
    |         ^~~~~~
alumnos.c:147:13: note: include '<stdio.h>' or provide a declaration of 'printf'
alumnos.c: In function 'matricula_multiple':
alumnos.c:157:9: warning: incompatible implicit declaration of built-in function 'printf' [-Wbuiltin-declaration-mismatch]
157 |         printf("No hay suficientes plazas libres");
    |         ^~~~~~
alumnos.c:157:9: note: include '<stdio.h>' or provide a declaration of 'printf'
alumnos.c:166:5: warning: incompatible implicit declaration of built-in function 'printf' [-Wbuiltin-declaration-mismatch]
166 |         printf("Se han matriculado todos los alumnos\n");
    |         ^~~~~~
alumnos.c:166:5: note: include '<stdio.h>' or provide a declaration of 'printf'
```

Figura 7: Terminal

```
gcc -c main.c
```

Este comando se ejecuta sin ningún error.

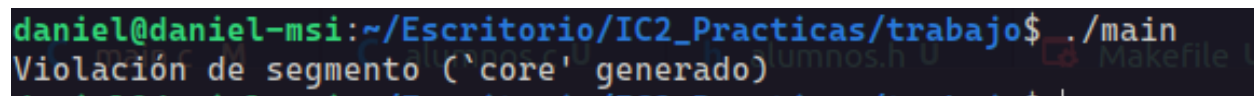
```
gcc -o main main.o alumnos.o
```

Este comando tampoco da ningún error al ejecutarse.

- Ejecutamos el programa

```
./main
```

Al intentar ejecutar el programa nos salta el siguiente error:



```
daniel@daniel-msi:~/Escritorio/IC2_Practicas/trabajo$ ./main
Violación de segmento (core generado)
```

Figura 8: Terminal

8 Memoria dinámica

En este apartado intentamos la memoria dinámica. El código quedaría de la siguiente manera:

Declaramos el puntero con memoria dinámica.

```
// Asignación de memoria dinámica
ptr = (struct Persona *)malloc(MAX * sizeof(struct Persona));
```

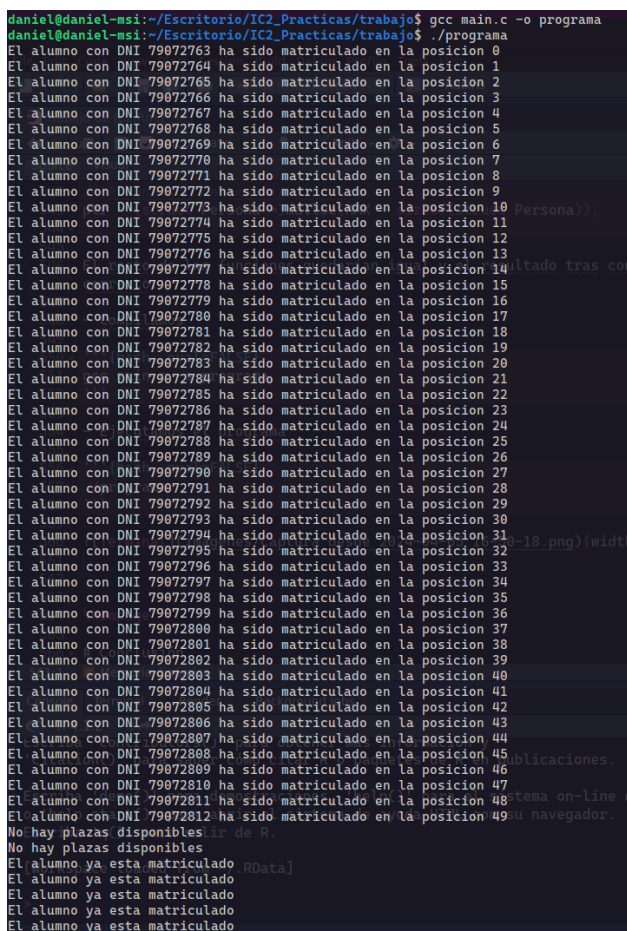
El resto de las funciones quedarían igual y el resultado tras compilar y ejecutar el código sería correcto.

- Compilamos

```
gcc main.c -o programa
```

- Ejecutamos el programa

```
./programa
```



```
daniel@daniel-msi:~/Escritorio/IC2_Practicas/trabajo$ gcc main.c -o programa
daniel@daniel-msi:~/Escritorio/IC2_Practicas/trabajo$ ./programa
El alumno con DNI 79072763 ha sido matriculado en la posicion 0
El alumno con DNI 79072764 ha sido matriculado en la posicion 1
El alumno con DNI 79072765 ha sido matriculado en la posicion 2
El alumno con DNI 79072766 ha sido matriculado en la posicion 3
El alumno con DNI 79072767 ha sido matriculado en la posicion 4
El alumno con DNI 79072768 ha sido matriculado en la posicion 5
El alumno con DNI 79072769 ha sido matriculado en la posicion 6
El alumno con DNI 79072770 ha sido matriculado en la posicion 7
El alumno con DNI 79072771 ha sido matriculado en la posicion 8
El alumno con DNI 79072772 ha sido matriculado en la posicion 9
El alumno con DNI 79072773 ha sido matriculado en la posicion 10
El alumno con DNI 79072774 ha sido matriculado en la posicion 11
El alumno con DNI 79072775 ha sido matriculado en la posicion 12
El alumno con DNI 79072776 ha sido matriculado en la posicion 13
El alumno con DNI 79072777 ha sido matriculado en la posicion 14
El alumno con DNI 79072778 ha sido matriculado en la posicion 15
El alumno con DNI 79072779 ha sido matriculado en la posicion 16
El alumno con DNI 79072780 ha sido matriculado en la posicion 17
El alumno con DNI 79072781 ha sido matriculado en la posicion 18
El alumno con DNI 79072782 ha sido matriculado en la posicion 19
El alumno con DNI 79072783 ha sido matriculado en la posicion 20
El alumno con DNI 79072784 ha sido matriculado en la posicion 21
El alumno con DNI 79072785 ha sido matriculado en la posicion 22
El alumno con DNI 79072786 ha sido matriculado en la posicion 23
El alumno con DNI 79072787 ha sido matriculado en la posicion 24
El alumno con DNI 79072788 ha sido matriculado en la posicion 25
El alumno con DNI 79072789 ha sido matriculado en la posicion 26
El alumno con DNI 79072790 ha sido matriculado en la posicion 27
El alumno con DNI 79072791 ha sido matriculado en la posicion 28
El alumno con DNI 79072792 ha sido matriculado en la posicion 29
El alumno con DNI 79072793 ha sido matriculado en la posicion 30
El alumno con DNI 79072794 ha sido matriculado en la posicion 31
El alumno con DNI 79072795 ha sido matriculado en la posicion 32
El alumno con DNI 79072796 ha sido matriculado en la posicion 33
El alumno con DNI 79072797 ha sido matriculado en la posicion 34
El alumno con DNI 79072798 ha sido matriculado en la posicion 35
El alumno con DNI 79072799 ha sido matriculado en la posicion 36
El alumno con DNI 79072800 ha sido matriculado en la posicion 37
El alumno con DNI 79072801 ha sido matriculado en la posicion 38
El alumno con DNI 79072802 ha sido matriculado en la posicion 39
El alumno con DNI 79072803 ha sido matriculado en la posicion 40
El alumno con DNI 79072804 ha sido matriculado en la posicion 41
El alumno con DNI 79072805 ha sido matriculado en la posicion 42
El alumno con DNI 79072806 ha sido matriculado en la posicion 43
El alumno con DNI 79072807 ha sido matriculado en la posicion 44
El alumno con DNI 79072808 ha sido matriculado en la posicion 45
El alumno con DNI 79072809 ha sido matriculado en la posicion 46
El alumno con DNI 79072810 ha sido matriculado en la posicion 47
El alumno con DNI 79072811 ha sido matriculado en la posicion 48
El alumno con DNI 79072812 ha sido matriculado en la posicion 49
No hay plazas disponibles
No hay plazas disponibles
El alumno ya esta matriculado
El alumno ya esta matriculado
El alumno ya esta matriculado
El alumno ya esta matriculado
El alumno ya esta matriculado
```

Figura 9: Terminal

9 Conclusión

El desarrollo de este trabajo nos ha dado la oportunidad de profundizar en diversos aspectos del lenguaje de C. Hemos visto elementos esenciales como punteros, vectores, struct y funciones, siendo conscientes de la posibilidad de mejora de este trabajo. Dejamos por escrito las ganas que tenemos de mejorar y añadir mejores funcionalidades como memoria dinámica para siguientes entregas. Adjuntamos un link de [GitHub](#) donde está el repositorio del trabajo con todas las versiones.

10 Actualización del Trabajo

En este apartado se actualiza el trabajo añadiéndole la división en módulos que no funcionaba anteriormente

10.1 Creación del test_grupo.c

```
#include <stdio.h>
#include <stdlib.h>
#include "test_grupo.h"

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

struct Persona *ptr;

int inicializa_grupo()
{
    // Asignación de memoria dinámica
    ptr = (struct Persona *)malloc(MAX * sizeof(struct Persona));
    if (ptr == NULL)
    {
        printf("Error: No se pudo asignar memoria\n");
        return -1;
    }
    for (int i = 0; i < MAX; i++)
    {
        ptr[i].dni = -1;
    }
    return 0;
}

int matricular_alumno(int dni)
{
    // Comprobar si el alumno ya está matriculado
    for (int i = 0; i < MAX; i++)
    {
```

```

        if (ptr[i].dni == dni)
        {
            return -2;
        }
    }
    // Matricular al alumno
    for (int i = 0; i < MAX; i++)
    {
        if (ptr[i].dni == -1)
        {
            ptr[i].dni = dni;
            return i;
        }
    }
    return -1;
}

// Las demás funciones permanecen sin cambios...

int desmatricular_alumno(int dni)
{
    // Buscando al alumno e iterando para desmatricularlo
    for (int i = 0; i < MAX; i++)
    {
        if (ptr[i].dni == dni)
        {
            ptr[i].dni = -1;
            return i;
        }
    }
    return -1;
}

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

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

```

```

    }
    return count;
}

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

void test_matricula(int dni)
{
    for (int i = 0; i < MAX + 2; 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);
        }
    }
    // Comprobando que el alumno ya esta matriculado
    for (int i = 0; i < 5; i++)
    {
        int dni1;
        dni1 = dni + i;
        int resultado = matricular_alumno(dni1);
        if (resultado == -2)
        {
            printf("El alumno ya esta matriculado\n");
        }
    }
}

void test_desmatricula(int dni)

```

```

{
    for (int i = 0; i < MAX - 10; i++)
    {
        int dni1;
        dni1 = dni + i;
        int resultado = desmatricular_alumno(dni1);
        if (resultado == -1)
        {
            printf("El alumno no existe\n");
        } else {
            printf("El alumno con DNI %d ha sido desmatriculado de la posicion %d\n",
                dni1, resultado);
        }
    }
    for (int i = 0; i < 3; i++)
    {
        int dni1;
        dni1 = dni + i;
        int resultado = desmatricular_alumno(dni1);
        if (resultado == -1)
        {
            printf("El alumno no existe\n");
        }
    }
}

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 \n");
}

void finaliza_grupo() {
    free(ptr);
}

```

10.2 Creando el test_grupo.h

```

#define MAX 50

//struct Persona;

```

```

int inicializa_grupo();
int matricular_alumno(int dni);
int desmatricular_alumno(int dni);
int testea_alumnos(int posicion);
int plazas_libres();
int plazas_ocupadas();
void test_matricula(int dni);
void test_desmatricula(int dni);
void matricula_multiple(int npersonas, int dnis[]);
void finaliza_grupo();

```

10.3 Creando el grupo.c

```

#include <stdio.h>
#include <stdlib.h> // Para las funciones malloc y free
#include "test_grupo.h"

int main()
{
    // Crear el vector de estructuras
    int dnis[2] = {1234567, 1324567};
    inicializa_grupo();
    test_matricula(79072763);
    test_desmatricula(79072763);
    matricula_multiple(2, dnis);
    int resultado = plazas_libres();
    printf("Hay %d plazas libres\n", resultado);
    int resultado2 = plazas_ocupadas();
    printf("Hay %d plazas ocupadas\n", resultado2);
    printf("El alumno en la posición %d, tiene el dni = %d\n", 1, testea_alumnos(1));
    printf("El alumno en la posición %d, tiene el dni = %d\n", 5, testea_alumnos(5));
    printf("El alumno en la posición %d, tiene el dni = %d\n", 48, testea_alumnos(48));
    finaliza_grupo();
    return 0;
}

```

10.4 Creamos el Makefile

Creamos el makefile para poder ejecutar los comandos

```

grupo: grupo.o test_grupo.o
    gcc -o grupo grupo.o test_grupo.o
grupo.o: grupo.c
    gcc -c grupo.c
test_grupo.o: test_grupo.c
    gcc -c test_grupo.c
clean:
    rm *.o

```

10.5 Ejecuciones finales

Ahora ejecutaremos el comando `make` para poder compilar el programa y ejecutaremos el resultado