

## Transición al uso de “structs”:

Durante la construcción del sub-menú registro, la idea original era utilizar punteros para dos variables e incluir dos vectores “nombre[]” y “apellido[]” y asimismo una matriz “carreras[][]” tal que el usuario modificara lo que se encontraba en las direcciones de memoria de las variables “padrón” y “num\_carrera”. El sub-menú registro funcionaba adecuadamente para esta situación, aunque se encontraba en un estado poco desarrollado respecto al desplazamiento por el menú.

En este momento, por decisión de la mayoría del grupo y tras consultar si era un método viable, pasamos a utilizar una estructura de datos “usuario\_t” tal que el sub-menú Asignaturas fuera más fácil, concluimos, de pasar y recibir datos tal como lo requiere el programa.

Aunque una solución utilizando punteros y arreglos era igual de viable como en el caso del sub-menú Registro, los miembros del grupo concluyeron que utilizar una estructura sería mas sencillo de escribir y la legibilidad del código sería mayor.

Sin Structs (Únicamente referente al Sub-menú Registro):

```
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
#include "main.h"

int menu_registro(int *ptr_padron, int *ptr_num_carrera, char carrera[][30], char apellido[], char nombre[]);

void imprimir_carrera(int fila, char carrera[][30]);

int main(void)
{
    /*Falta Eliminar Hardcodeo*/
    estado main estado = MAIN_MENU;
    int input_i = 0;
    char apellido[50];
    char nombre[50];
    int padron, num_carrera;
    char carrera[12][30] = {{ING_0}, {ING_1}, {ING_2}, {ING_3}, {ING_4}, {ING_5}, {ING_6}, {ING_7}, {ING_8}, {ING_9}, {ING_10}, {ING_11}};
```

```
case MENU_REGISTRO:
{
    input_i = menu_registro(&padron, &num_carrera, carrera, apellido, nombre);

    if(input_i == EXIT_SUCCESS)
        estado = MAIN_MENU;
    else
    {
        return EXIT_FAILURE;
    }

    printf("PRUEBA: PADRON: %i CARRERA: %i\n", padron, num_carrera);

    break;
}
```

```

Archivo  Editar  Ver  Buscar  Terminal  Ayuda
void imprimir_carrera(int fila, char carrera[][30])
{
    printf("%s\n", carrera[fila]);
}

int menu_registro(int *ptr_padron, int *ptr_num_carrera, char carrera[][30], char apellido[], char nombre[])
{
    int input_i = 0;
    puts(MSG_REGISTRO);
    while(1)
    {
        printf("1) %s\n2) %s\n3) %s\n0) %s\n", REGISTRO_OPCION_NOMBRE, REGISTRO_OPCION_PADRON, REGISTRO_OPCION_CARRERA, OPCION_VOLVER);

        if(scanf("%i", &input_i) != 1)
        {
            fprintf(stderr, "%s: %s\n", ERR_PREFIJO, ERR_OPCIONES);
            return EXIT_FAILURE;
        }
        while(getchar() != '\n');

        if(input_i == 1)
        {
            printf("%s: ", REGISTRO_ING_APELLIDO);
            if(scanf("%s", apellido) != 1)
            {
                fprintf(stderr, "%s: %s\n", ERR_PREFIJO, ERR_REG_NOMBRE);
                return EXIT_FAILURE;
            }
            while(getchar() != '\n');

            printf("%s: ", REGISTRO_ING_NOMBRE);

            if(scanf("%s", nombre) != 1)
            {
                fprintf(stderr, "%s: %s\n", ERR_PREFIJO, ERR_REG_NOMBRE);
                return EXIT_FAILURE;
            }
            while(getchar() != '\n');

            printf("%s: %s, %s\n", REGISTRO_ING_AVISO, apellido, nombre);
        }
        else if(input_i == 2)
        {
            printf("%s: ", REGISTRO_ING_PADRON);
            if(scanf("%i", ptr_padron) != 1)
            {
                fprintf(stderr, "%s: %s\n", ERR_PREFIJO, ERR_REG_PADRON);
                return EXIT_FAILURE;
            }
            while(getchar() != '\n');
            printf("%s: %i\n", REGISTRO_ING_AVISO, *ptr_padron);
        }
        else if(input_i == 3)
        {
            printf("%s: ", REGISTRO_ING_CARRERA);
            if(scanf("%i", ptr_num_carrera) != 1)
            {
                fprintf(stderr, "%s: %s\n", ERR_PREFIJO, ERR_REG_CARRERA);
                return EXIT_FAILURE;
            }
            while(getchar() != '\n');

            printf("%s: ", REGISTRO_ING_AVISO);
            imprimir_carrera(*ptr_num_carrera, carrera);
        }
        else if(input_i == 0)
        {
            break;
        }
        else
        {
            fprintf(stderr, "%s: %s\n", ERR_PREFIJO, ERR_OPCIONES);
            return EXIT_FAILURE;
        }
    }

    return EXIT_SUCCESS; /* Solo se sale exitosamente si se activa el break; */
}

```

209,1 Final

[La versión con Structs se puede encontrar en el código fuente funciones.c]

## Usar GitHub

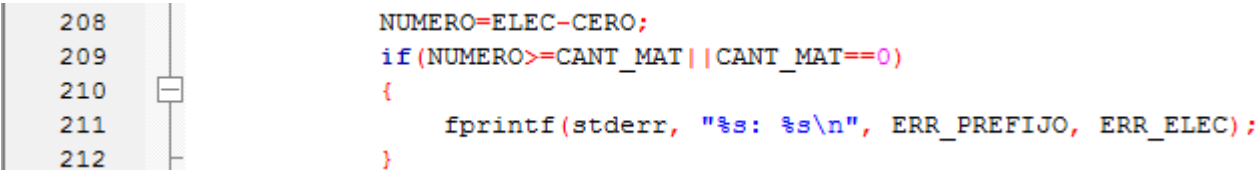
Puesto que queríamos trabajar en el proyecto a través de internet, tuvimos que buscar un servicio que nos permitiera realizar esto sin que hubiera problemas al editar el código entre nosotros. Por ejemplo, evitar que alguien trabajara en algo solo para que accidentalmente otro compañero lo sobre-escribiera con su versión. Para esto, nos decidimos a usar un repositorio de GitHub y a aprender algunos comandos Git.

Primeramente, nos pareció un tanto complicado aprender los comandos y hasta el momento seguimos sin tener el tema muy bien afianzado. Sin embargo, utilizando el navegador y arreglándonos, pudimos aprovechar la capacidad de Git para separar el trabajo en ramas (“Branches”) en las que cada uno trabajaba y mediante “Pull Requests” unir todo en la rama maestra.

## Lectura de una lectura tipo char o int (Sub-menú Asignaturas)

Como el menú debe cambiar conforme a la cantidad de asignaturas ingresadas, no se podía preseleccionar un caso para cada asignatura ya que las mismas son reguladas por el usuario, por lo que se decidió que las opciones para poder sobrescribir una asignatura tenían que ser números. A su vez, las otras opciones (agregar asignatura, eliminar asignatura y salir del sub-menú) tenían que poder distinguirse claramente. Por este motivo, usamos variables del tipo char (más específicamente: '+', '-', '!') para identificar estas opciones. Sin embargo, esto trajo un problema ya que ahora el programa tenía que tomar un input e interpretarlo como uno del tipo char o tipo int según lo que quisiera el usuario.

La primera solución que se nos ocurrió en esto fue guardar lo escrito en una variable de tipo char, por lo que la selección de las opciones no-numéricas no era un problema. Pero esto dio lugar a problema ya que, al ingresar un número, el programa no lo reconocería como tal sino que lo interpretaría como un carácter,. Por lo cual, al compararlo no se tiene en cuenta el valor que representa sino su equivalente en el código ASCII, para esto al valor seleccionado se le restaba el valor del '0' en ASCII (que se le asigna el valor 48) y se guardaba en una variable entera.



```

208 NUMERO=ELEC-CERO;
209 if(NUMERO>=CANT_MAT || CANT_MAT==0)
210 {
211     fprintf(stderr, "%s: %s\\n", ERR_PREFIJO, ERR_ELEC);
212 }
    
```

(Imagen del programa con la solución temporal)

Con este método el programa podía funcionar con un ingreso de hasta 10 materias sin ningún problema aparente. Pero si se superaba este límite, podía ocurrir que algún carácter no seleccionado se tome como una variable cualquiera. Para esto encontramos la solución con la función “atoi()”, la cual traduce una cadena de caracteres en una variable entera. Por ejemplo, cuando el usuario ingresa un carácter numérico como “4” la función “atoi()”, convierte ese ‘4’ de un carácter a un número y si se ingresa otro tipo de carácter como, por dar otro ejemplo, ‘a’, la función “atoi()”, devuelve un cero, por lo que solamente hay que verificar que, cuando el usuario ingrese un 0, el programa lo interprete como tal.

```

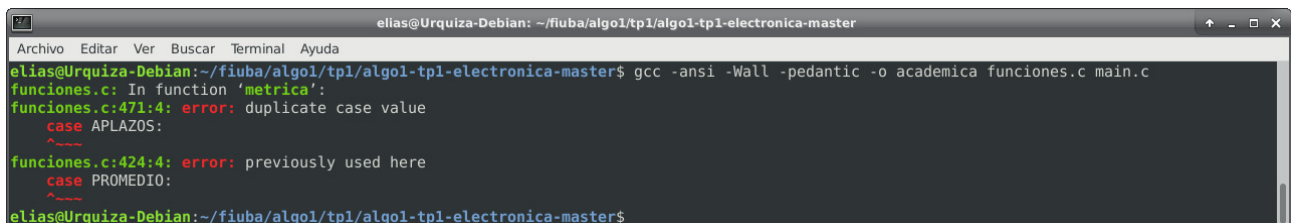
258     NUMERO = atoi(ELEC); /*Modificacion para que se guarde el numero ingresado en vez de su equivalente en ASCII*/
259     if(NUMERO >= CANT_MAT || CANT_MAT == 0 || NUMERO == 0 && (ELEC[0] != ASCII_CERO))
260     {
261         fprintf(stderr, "%s: %s\n", ERR_PREFIJO, ERR_ELEC);
262     }

```

(Imagen de la función con la solución definitiva)

## Problema con los idiomas

Al realizar los primeros test, el programa parecía funcionar correctamente. Sin embargo, al probar los distintos idiomas incluidos, descubrimos que había un error de compilación al utilizar el idioma inglés. Al revisarlo mejor descubrimos que en el sub-menú “métricas”, en la definición de las variables donde especificamos cual es el caracter que hay que ingresar para elegir una opción, la letra ‘A’ se utilizaba para acceder a dos opciones distintas. Gracias al mensaje de error de compilación del gcc pudimos identificarlo rápidamente y solucionarlo a tiempo.



```

elias@Urquiza-Debian: ~/fiuba/algo1/tp1/algo1-tp1-electronica-master
Archivo Editar Ver Buscar Terminal Ayuda
elias@Urquiza-Debian:~/fiuba/algo1/tp1/algo1-tp1-electronica-master$ gcc -ansi -Wall -pedantic -o academica funciones.c main.c
funciones.c: In function 'metrica':
funciones.c:471:4: error: duplicate case value
    case APLAZOS:
    ^~~~~
funciones.c:424:4: error: previously used here
    case PROMEDIO:
    ^~~~~
elias@Urquiza-Debian:~/fiuba/algo1/tp1/algo1-tp1-electronica-master$

```

[Lectura del error en la consola (Superior) y Motivo del error (Inferior)]



```

english.h (~/fiuba/algo1...lectronica-master) - VIM
Archivo Editar Ver Buscar Terminal Ayuda

/* Metrics */
#define MSJ_METRICA "\nPick which metric to measure:"
#define METRICA_OPCION_PROMEDIO "Average"
#define METRICA_OPCION_PROMEDIO_CHAR 'A'
#define METRICA_OPCION_MAXIMO "Maximum"
#define METRICA_OPCION_MAXIMO_CHAR 'M'
#define METRICA_OPCION_MINIMO "Minimum"
#define METRICA_OPCION_MINIMO_CHAR 'm'
#define METRICA_OPCION_CANTIDAD "Amount of Courses"
#define METRICA_OPCION_CANTIDAD_CHAR '#'
#define METRICA_OPCION_APLAZOS "Failed Courses"
#define METRICA_OPCION_APLAZOS_CHAR 'A'
#define METRICA_OPCION_VOLVER "Go Back"
#define METRICA_OPCION_VOLVER_CHAR '0'

86,1 90%

```