

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 relevante 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;
    }
}
```

Trabajo Práctico N° 1 - Problemas y Soluciones

```
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_AVIS0, 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_AVIS0, *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_AVIS0);
            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

Trabajo Práctico N° 1 - Problemas y Soluciones

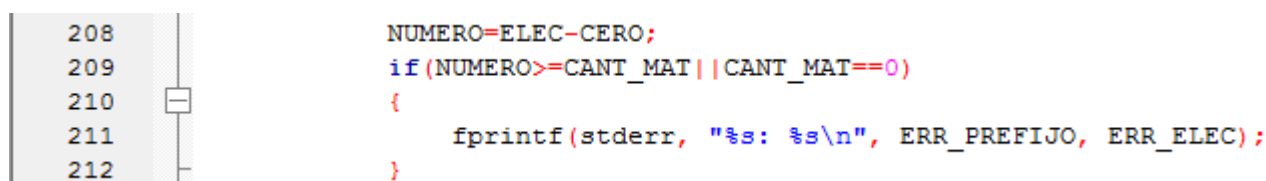
accidentalmente otro compañero lo sobre-escribiera con su version. Para esto, nos decidimos a usar un repositorio de GitHub y a aprender algunos comandos Git.

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

En el submenú de 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 elegir sobrescribir una asignatura tenían que ser números y que a su vez las otras opciones (agregar asignatura, eliminar asignatura y salir del submenú) tenían que tener un distintivo ya que esos casos se separan de lo que son las asignaturas, por este motivo usamos variables tipo char ('+', '-', '!') para identificar estas opciones, por lo que la decisión tomada trajo un problema. El programa tenía que tomar una variable tipo char o tipo int según lo que decida el usuario.

La primera solución que se nos ocurrió en esto fue guardar lo escrito en un tipo char, por lo que la selección de las opciones distintivas no era un problema, aunque surgía otro problema ya que al ingresar un número, el programa no lo reconocería como tal sino que se tomaría como un carácter, y 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 es igual a 48) y se guardaba ese valor 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 hasta 10 materias sin ningún problema, pero con si se quería agregar más materias después de eso, podía ocurrir que algún carácter no seleccionado, se tome como una variable cualquiera, y para esto encontramos la solución con la función "atoi" la cual traduce una cadena de caracteres en una variable entera. 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 ejemplo "a", la función atoi devuelve un cero, por lo que solamente hay que crear una verificación para cuando el usuario ingrese un 0 en la consola o si ingresa cualquier otro carácter.

Trabajo Práctico N° 1 - Problemas y Soluciones

```
258     NUMERO = atoi(ELEC); /*Modificación 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

A la hora de probar el programa, todo estaba funcionando, entonces probando los idiomas descubrimos que había un error de compilación con el idioma inglés. Al revisarlo mejor descubrimos que señalaba el submenú “métricas” donde en un switch un case había sido declarado dos veces, lo cual no tenía mucho sentido que el error aparezca solamente cuando cambiamos a ese idioma ya que lo único que hay son definiciones en el idioma, hasta que nos dimos cuenta en la definición de las variables donde poníamos cual es la letra que hay que ingresar, según el idioma, había una letra que se estaba repitiendo, por lo que para dos casos distintos estabas usando la misma letra, esto se pudo arreglar cambiándolo y ya no se notaba ese problema.



```
elias@Urquiza-Debian: ~/fiuba/algol/tp1/algol-tp1-electronica-master
Archivo Editar Ver Buscar Terminal Ayuda
elias@Urquiza-Debian:~/fiuba/algol/tp1/algol-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/algol/tp1/algol-tp1-electronica-master$
```

(Lectura del error en la consola)

Trabajo Práctico N° 1 - Problemas y Soluciones



```
english.h (~/.fiuba/algo1...electronica-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%
```

(Imagen de cual era el error en el código)