



## **Algoritmos y Programación 1**

—  
**Cátedra: Kuhn**  
—

***Trabajo Práctico N° 1 | 20/09/2017 | 1<sup>er</sup> Entrega***  
—

### **Alumnos:**

Mauricio Cabrera	101334	maurihuergo@hotmail.com
Aurelien Floutard	101999	aure.flout@hotmail.fr
Elías Urquiza	100714	eliasurquiza07@gmail.com

# Calculador de desempeño académico

95.11/75.02 Algoritmos y Programación I

6 de septiembre de 2017

## 1. Objetivo del TP

El objetivo del presente trabajo consiste en la realización de un conjunto de aplicaciones en modo consola, escritos en lenguaje ANSI-C89, que permitan implementar un sistema de cálculo de métricas sobre el desempeño académico del usuario.

## 2. Alcance del TP

Mediante el presente TP se busca que el estudiante adquiera y aplique conocimientos sobre los siguientes temas:

- Programas en modo consola
- Directivas al preprocesador C
- Juego de caracteres ASCII
- Tipos enumerativos
- Control de flujo
- Salida de datos con formato
- Funciones
- Modularización
- Arreglos
- Cadenas de caracteres

**Fecha de entrega: 20 de septiembre de 2017**

### 3. Introducción

Es común entre los alumnos llevar una cierta medida de cómo el mismo se va desempeñando a lo largo de la carrera. Para esta tarea, se pide desarrollar una aplicación que, cumpliendo con las especificaciones de la sección 4, pueda mostrar una serie de datos según la información provista por el alumno.

### 4. Especificaciones

El programa a desarrollar debe ser interactivo. Debe obtener sus datos a través del flujo de entrada estándar, `stdin`. Deberá consultar al usuario por los siguientes datos: nombre, apellido, padrón, carrera y asignaturas. La cantidad de materias es desconocida y serán ingresadas de a una, con su nombre y nota. Además, debe ser posible compilarlo en diferentes idiomas.

Ante cada error en el ingreso de los datos, el programa debe indicar al usuario, con un mensaje *claro*, la falla.

#### 4.1. Menú principal

Para la toma de datos, el programa se valdrá de un menú, similar al siguiente ejemplo:

```
Elija qué desea hacer:
  R) Modificar registro
      personal
  A) Modificar asignatura
  M) Calcular métrica
      individual (opcional)
5  !) Finalizar
    0) Salir
?
```

Ejemplo 1: Español

```
Velg hva du skal gjøre:
  P) Endre
      personopplysninger
  F) Endre fag
  B) Beregn individuelle
      beregninger
5  !) Slutt
    0) Avslutt
?
```

Ejemplo 2: Idioma 2

Cada uno de los ítems del menú puede o no tener un submenú en el que debe actuar. Nótese que los caracteres de selección pueden ser variados: alfabéticos, símbolos y/o numéricos. A continuación se detallan las operaciones de cada uno de estos ítems.

#### 4.2. Submenú de registro personal

En este menú se pide información para completar lo que llamaremos “registro personal”. Estos datos abarcan: nombre, apellido, padrón y carrera.

La selección del ítem a completar depende de un nuevo menú, a saber:

```
Elija qué desea modificar:
  A) Apellido y nombre
  #) Padron
  C) Carrera
```

```

5      0) Volver
      ?

```

Al seleccionar una opción, se mostrará el dato cargado, se solicitará el nuevo dato, y se registrará el mismo, sin pedido de confirmación.

### 4.3. Submenú de asignaturas

En este menú se presenta la información correspondiente a las asignaturas y se registran cambios y nuevos datos. Un ejemplo de ejecución de este menú es:

```

Usted no tiene asignaturas cargadas.
+ ) Ingresar nueva asignatura
0 ) Volver
?

```

ó bien

```

Opciones:
1 ) AMII (9)
2 ) Química (8)
3 ) Inglés (10)
5  4 ) Probabilidad y estadística (9)
    5 ) Sociología (10)
    + ) Ingresar nueva asignatura
    - ) Eliminar una asignatura (opcional)
10 0 ) Volver
    ?

```

Nótese la particularidad que tiene este menú de modificarse según el estado de ejecución del programa.

### 4.4. Submenú de métricas (opcional)

En este menú se da la opción al usuario de calcular alguna métrica con el estado actual del programa y presenta un menú similar al siguiente:

```

Métricas:
P ) Promedio
M ) Máximo
m ) Mínimo
5  # ) Cantidad de materias
    2 ) Aplazos
    0 ) Volver
?

```

Seleccionada una opción, el programa indica al usuario dicho resultado.

### 4.5. Submenú de finalización

Este submenú no mostrará un nuevo menú. En el mismo se computarán todos las métricas y se imprimirá un mensaje, por una salida estándar diferente a la utilizada en los menús. El mensaje debe respetar el siguiente formato:

```
Nombre y Apellido , Padrón , Carrera , Materias , Promedio , Aplazos
```

Todo el mensaje contenido en una única línea. Luego se volverá al inicio de la aplicación, con todos los datos eliminados.

#### 4.6. Submenú “Salir”

Termina el ciclo de ejecución, sin mostrar mensajes.

### 5. Opcionales

Se consideran puntos opcionales, la posibilidad de eliminar asignaturas ya cargadas de la lista, y la posibilidad de computar cada una de las diferentes métricas a pedido del usuario.

Se puede agregar, como opcional, que el cada uno de los caracteres que se utilizan en las selecciones no varíe con el idioma. De hacer este opcional, se considerará válido si la implementación es correcta, ya que no es correcto mantenerlos a través de hardcodes o modularizaciones pobres.

### 6. Ejemplo de ejecución

```
$ ./mi_aplicacion
Bienvenidx
Elija qué desea hacer:
  R) Modificar registro personal
5  A) Modificar asignatura
  M) Calcular métrica individual
  !) Finalizar
  O) Salir
? R
10 Elija qué desea modificar:
   A) Apellido y nombre
   #) Padron
   C) Carrera
   O) Volver
15 ? N
   No tiene nombre cargado.
   ¿Apellido y nombre? Fulanito Cosme
   Elija qué desea modificar:
20   A) Apellido y nombre
   #) Padron
   C) Carrera
   O) Volver
   ? #
   ¿Padron? 123456
25 Elija qué desea modificar:
   A) Apellido y nombre
   #) Padron
   C) Carrera
   O) Volver
30 ? C
   ¿Codigo de la carrera? 7
   Elija qué desea modificar:
35   A) Apellido y nombre
   #) Padron
   C) Carrera
   O) Volver
   ? O
   Elija qué desea hacer:
```

```

40      R) Modificar registro personal
      A) Modificar asignatura
      M) Calcular métrica individual
      !) Finalizar
      O) Salir
? A
45 Usted no tiene asignaturas cargadas.
  Opciones:
    +) Ingresar nueva asignatura
    O) Volver
? +
50 ¿Asignatura (Nota Nombre)? 9 AMII
  Opciones:
    1) AMII (9)
    +) Ingresar nueva asignatura
    O) Volver
55 ? 0
  Elija qué desea hacer:
    R) Modificar registro personal
    A) Modificar asignatura
    M) Calcular métrica individual
60    !) Finalizar
    O) Salir
? !
  Fulanito Cosme,123456,7,1,9,0
  Elija qué desea hacer:
65    R) Modificar registro personal
    A) Modificar asignatura
    M) Calcular métrica individual
    !) Finalizar
    O) Salir
70 ? 0
$

```

## 7. Comentarios

Si bien no es necesario utilizar variables globales en este trabajo, con su debida **justificación**, *algunas* pueden ser aceptadas.

Es recomendable tener un diccionario construido con las carreras como cadenas de caracteres y un tipo enumerado; lo mismo que los mensajes de error y el enumerando `status_t`. Este enumerando, en español representa una *condición*.

El nombre del enumerando que representa *estado* en el que se encuentra el programa (menú principal, algún submenú, error, inicio, reinicio, etc.) en inglés es `state_t`.

## 8. Restricciones

La realización del trabajo se encuentra sujeta a las siguientes restricciones:

- Debe realizarse en grupos de 3 (**tres**) integrantes.
- No está permitida la utilización de memoria dinámica.

- No está permitida la utilización de funciones específicas para manejo de archivos. Como funciones de E/S sólo pueden ser utilizadas las funciones de biblioteca `fgetc()`, `getchar()`, `scanf()`, `printf()`, `fprintf()`, `putc()`, `fputc()`, `puts()`, `fputs()` o similares.
- No está permitida la utilización de `gets()`<sup>1</sup>, `fflush(stdin)`<sup>2</sup>, la biblioteca `conio.h`<sup>3</sup> (`#include <conio.h>`), etc.
- Debe recurrirse a la utilización de funciones mediante una adecuada parametrización.
- No está permitido en absoluto tener hard-codings:

```

1 ...
2 char c;
3 ...
4 if (c == 'R')                               /* ; i hard-coded!! */
5     printf("%s", "xxxxxxxx");               /* ; i hard-coded!! */
6 else if (c == 'A')                           /* ; i hard-coded!! */
7     printf("%s", "yyyyyyyy");               /* ; i hard-coded!! */
8 ...

```

sino que debe recurrirse al uso de ETIQUETAS, CONSTANTES SIMBÓLICAS, MACROS, etc.

Los ejemplos no son exhaustivos, sino que existen otros hard-codings y tampoco son aceptados.

- Hay ciertas cuestiones que no han sido especificadas intencionalmente en este Requerimiento, para darle al/la desarrollador/a la libertad de elegir implementaciones que, según su criterio, resulten más convenientes en determinadas situaciones. Por lo tanto, se debe explicitar cada una de las decisiones adoptadas, y el o los fundamentos considerados para las mismas.

## 9. Entrega del Trabajo Práctico

La fecha de entrega del trabajo práctico es: 20 de septiembre de 2017 o antes.

No se requiere entrega en papel. Deberá realizarse una entrega digital, a través del campus de la materia, de un único archivo cuyo nombre debe seguir el siguiente formato:

YYYYMMDD\_apellido1-apellido2-apellido3\_entrega-N.tar.gz

donde YYYY es el año (2017), MM el mes y DD el día en que uno de los integrantes sube el archivo, apellido-1a3 son los apellidos de los integrantes ordenados alfabéticamente, entrega-N indica el número de vez que se envía el trabajo (entrega-1, entrega-2, etc.), y .tar.gz es la extensión, que no necesariamente es .tar.gz.

<sup>1</sup>obsoleta en C99 [3], eliminada en C11 [4] por fallas de seguridad en su uso.

<sup>2</sup>comportamiento indefinido para flujos de entrada ([3],[4]). Definida en estándar POSIX.

<sup>3</sup>biblioteca no estándar, con diferentes implementaciones y licencias, y no siempre disponible.

El archivo comprimido debe contener los siguientes elementos:

- La correspondiente documentación de desarrollo del TP (en formato pdf), siguiendo la numeración siguiente, incluyendo:
  1. Carátula del TP. Incluir una dirección de correo electrónico.
  2. Enunciado del TP.
  3. Estructura funcional de los programas desarrollados.
  4. Explicación de cada una de las alternativas consideradas y las estrategias adoptadas.
  5. Resultados de la ejecución (corridas) de los programas, captura de las pantallas, bajo condiciones normales e inesperadas de entrada.
  6. Reseña sobre los problemas encontrados en el desarrollo de los programas y las soluciones implementadas para subsanarlos.
  7. Bibliografía (ver aparte).
  8. Indicaciones sobre la compilación de lo entregado para generar la aplicación.

**NOTA:** Si la compilación del código fuente presenta mensajes de aviso (warning), notas o errores, los mismos deben ser comentados en un apartado del informe.

**NOTA:** El Informe deberá ser redactado en *correcto* idioma castellano.

- Códigos fuentes en formato de texto plano (.c y .h), *debidamente documentados*.

**NOTA:** Todos los integrantes del grupo deben subir el *mismo* archivo.

**NOTA:** Se debe generar y subir un único archivo (comprimido) con todos los elementos de la entrega digital. **NO usar RAR.** La compresión RAR no es un formato libre, en tanto sí se puede utilizar *ZIP*, *GUNZIP*, u otros (soportados, por ejemplo, por la aplicación de archivo *TAR*).

Si no se presenta cada uno de estos ítems, será rechazado el TP.

## 10. Bibliografía

Debe incluirse la referencia a toda bibliografía consultada para la realización del presente TP: libros, artículos, URLs, etc., citando:

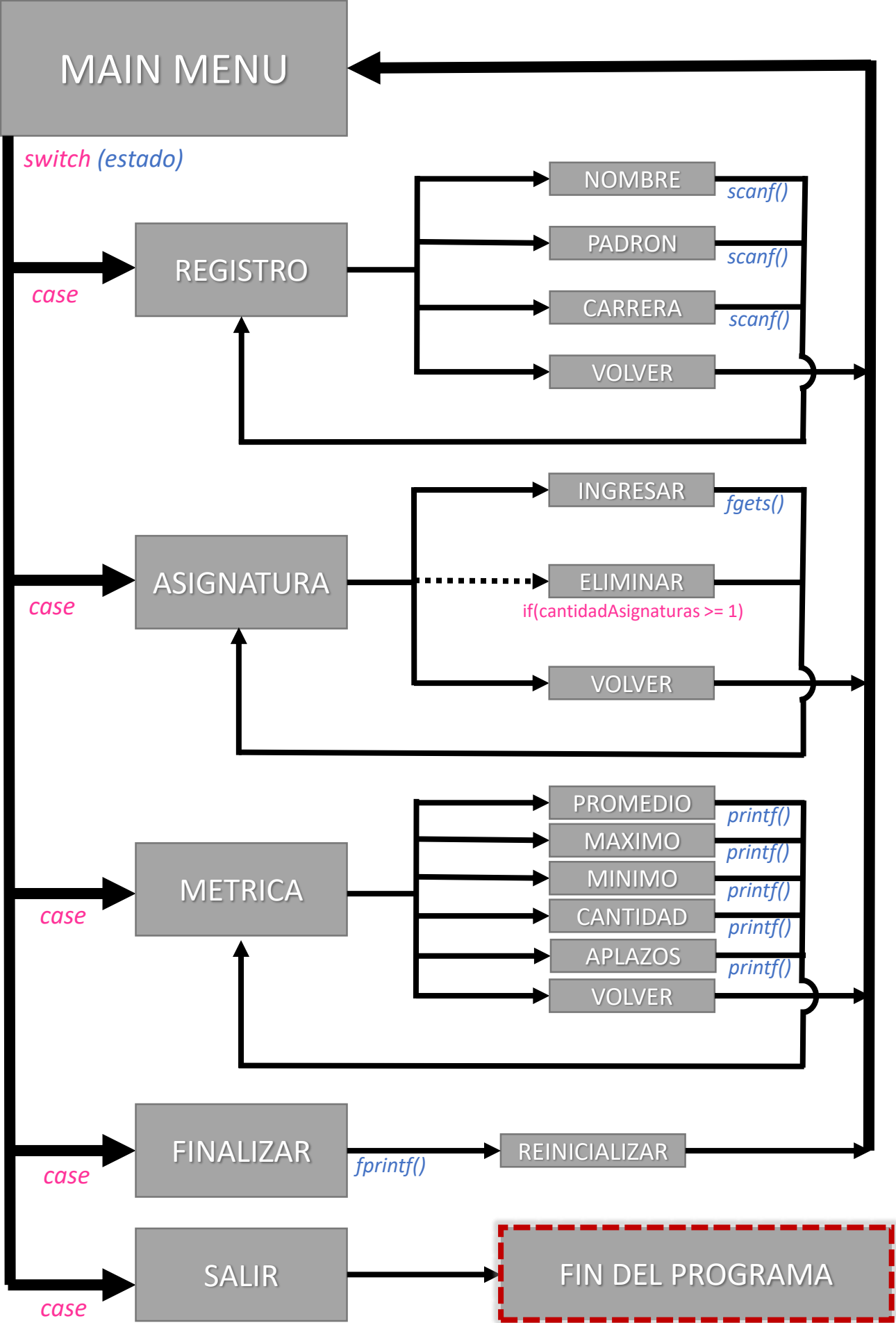
- Denominación completa del material (Título, Autores, Edición, Volumen, etc.).
- Código ISBN del libro (opcional: código interbibliotecario).
- URL del sitio consultado. No poner [Wikipedia.org](http://Wikipedia.org) o [stackexchange.com](http://stackexchange.com), sino que debe incluirse un enlace al artículo, hilo, etc. consultado.

Utilizando  $\text{\LaTeX}$ , la inclusión de citas/referencias es trivial. Los editores de texto gráficos de las suites de ofimática, como LibreOffice Write o MS Word, admiten plugins que facilitan la inclusión.



## Ejemplo de referencias

- [1] B.W. Kernighan y D.M. Ritchie. *The C Programming Language*. 2.<sup>a</sup> ed. Prentice-Hall software series. Prentice Hall, 1988. ISBN: 9780131103627.
- [2] P. Deitel y H. Deitel. *C How to Program*. 7.<sup>a</sup> ed. Pearson Education, 2012. ISBN: 9780133061567.
- [3] ISO/IEC. *Programming Languages – C*. ISO/IEC 9899:1999(E). ANSI, dic. de 1999, págs. 270-271.
- [4] ISO/IEC. *Programming Languages – C*. INCITS/ISO/IEC 9899:2011. INCITS/ISO/IEC, 2012, pág. 305.



## Alternativas Consideradas y Estrategias Adoptadas

### Uso de “estado\_main”

Para poder controlar el flujo del programa de manera eficiente entre menús, declaramos con typedef un tipo de variable a la cual denominamos “estado\_main” cuyos valores determinan el estadio del programa con la utilización de un “switch();”.

De esta manera, el programa únicamente debe tener en consideración en qué estado se encuentra para así desplegar el menú deseado. Además, permite no tener que incluir muchos estados de “if”.

### Uso de Estructura “usuario\_t”

Con la idea de poder pasar datos a (y desde) funciones por referencia en vez de por valor, originalmente planteamos un modelo del programa basado en punteros para simular este tipo de pasaje. Sin embargo, gracias al uso de una estructura a la cual denominamos “usuario\_t”, conseguimos simular el mismo efecto sin tener que usar una notación que en su momento consideramos engorrosa para programar. Además, facilita la re-inicialización de los datos del usuario puesto que solo tenemos que pasar la estructura a una función en vez de tener que pasar la dirección de memoria de cada uno de los datos. Esto se destaca al notar que la mayor parte de las funciones requieren únicamente la estructura como input para poder realizar su trabajo.

Igualmente, el uso de una estructura no viene sin inconvenientes, puesto que el uso de la misma ocupa más memoria. No consideramos que sea un método eficiente para todos los casos, pero sí que lo es para el alcance de este programa, al menos en cuanto a tiempo de programación y depurado.

### Motivos y usos de funciones:

#### La Función “menu();”

Puesto que originalmente la estructura de nuestro programa tenía muchos “ifs” anidados, con sus verificaciones necesarias y la capacidad de darle al usuario 3 oportunidades para ingresar una opción válida, hicimos la transición al uso de “switch()” para cada opción del menú. Sin embargo, esto trajo consigo la necesidad de encontrar una alternativa para aun así permitirle al usuario 3 intentos en caso de que cometiera un error (además de verificar que el usuario ingrese un solo valor). Originalmente desarrollamos los distintos casos del “switch()” e hicimos que en el caso por defecto (default), se re-imprimiera el menú y se fuera sumando la cantidad de intentos realizados por el usuario. De esta forma, si el usuario ingresaba algún dato no-válido muchas veces, al entrar más de 3 veces seguidas al caso default el programa cambiaba su estado al de salida del menú. La implementación del mismo no era muy elegante, pero nos permitió descubrir una mejor forma de implementar la impresión de los menús: con su propia función encargada de imprimirlos y buscar

por errores. La misma, llamada “menu();”, únicamente requiere el ingreso del estado de main y la cantidad de intentos que se ha llevado a cabo. Gracias a esto, pudimos realizar la corrección del programa más rápido. El único menú que requirió más trabajo, sin embargo, fue el sub-menú Asignaturas. La explicación del mismo se desarrolla más adelante en esta misma sección.

### **Función “reinit();”**

Con la intención de que la misma fuera utilizada por la función “finalizar();”, nuestro objetivo era condensar la re-inicialización de los datos del usuario en una función. Para ello, declaramos la función “reinit();”, la cual requiere únicamente el ingreso de la estructura a reinicializar y uno por uno reemplaza sus valores guardados por valores nulos. El beneficio de crear esta función en vez de realizar estas acciones dentro de “finalizar();” es que también nos permite utilizarla al principio del programa, antes de que el usuario ingrese algún dato, para así asegurarnos que las variables tengan valores nulos por defecto.

### **Funciones “imprimir carrera aviso();” e “imprimir carrera fin();”**

Puesto que el programa, dentro del menú registro, imprime un aviso de lo que ha ingresado el usuario en cada opción. Se escribió una función que permitía imprimir la carrera seleccionada según el numero elegido. Sin embargo, la función imprime por stdout. Por ello, no se podía reutilizar en el menú finalizar ya que en el mismo se imprimen los datos por stderr. En vista de esto, se optó por armar dos funciones esencialmente iguales, pero una de ellas imprime por stdout y la otra por stderr.

Otra posible alternativa habría sido armar la función de tal forma que actué según el estado del programa. Si está en el menú registro, que imprima por stdout. Mientras que, si está en el menú finalizar, imprima por stderr.

Por cuestiones de tiempo, al darnos cuenta de esto durante la corrección, preferimos dejar las dos funciones separadas y mantener la funcionalidad del programa.

### **Función “clear buffer();”**

La misma fue diseñada con el propósito de incrementar la legibilidad del código así no teníamos que escribir un “while(getchar() != '\n');” cada vez que lo necesitábamos.

### **Funciones de impresión**

Además de las mencionadas, también desarrollamos otras funciones cuyo único propósito es imprimir algún dato o algún menú. Las mismas siempre siguen el formato “imprimir\_[nombre]();” Y consideramos redundante explicar cada una

puesto que cuentan con una estructura muy básica. Su propósito es facilitar la lectura del código fuente.

### **Función Finalizar:**

La misma es utilizada cuando el usuario desea dejar de ingresar datos y los ingresados se vean impresos por stderr. Lo que necesita para funcionar es la estructura utilizada y el arreglo que contiene el nombre de las materias según su número. Su estructura es muy sencilla pero está dividida en 3 funciones, 2 fprintf y la función “imprimir\_carrera\_fin();” puesto que esta última es un proceso y no retorna nada.

Al finalizar, retorna la estructura reinicializada con “reinit();” y limpia el buffer.

### **Menú Registro y sus Funciones:**

El menú registro está diseñado de tal forma que permita el ingreso de datos del usuario según lo que se le pida. Los datos posibles son: Apellido y Nombre, Padrón y Carrera. El mismo necesita recibir la estructura que va a modificar y trabaja en ella para finalmente devolverla. Además, también recibe el arreglo “carrera” que incluye el nombre de cada carrera ordenada según su número correspondiente. Su uso es para poder imprimir la carrera seleccionada dentro del submenú utilizando la otra función, explicada previamente, llamada “imprimir\_carrera\_aviso();”.

A causa de que el ingreso de los mismos no siempre es fácil de verificar su validez (por ejemplo, si el usuario ingresa un numero como parte de su nombre, ya sea accidentalmente o a propósito), el programa imprime los datos ingresados antes de volver al menú en sí mismo. De esta forma, el usuario puede corroborar si no cometió algún error. Esto fue implementado, originalmente, para permitirle al usuario saber si el código de materia ingresado es el correcto o no. Pero, para mantener cierta consistencia dentro del menú, se extendió a los otros dos datos posibles.

### **Menú Asignaturas y sus Funciones:**

En este menú el usuario podía agregar las materias que curso y las notas correspondientes, a su vez el usuario puede reescribir la función si es que escribió mal una materia o la nota ingresada no se corresponde, o simplemente podía borrar la materia seleccionando en que numero esta.

Al ser un menú de temática dinámica, ya que al menú va creciendo conforme a la cantidad de materias que se agregan (hasta llegar a un tope predefinido por nosotros) el mismo no se podía escribir de la misma forma que los demás, para esto al cambiar de usar ifs anidados a switchs recurrí al caso default para para dirigir a la opción de sobre escritura de materias, en la misma se comprueba que la materia

seleccionada exista y que tenga sentido lo ya que todo lo que se escriba y no sea un caso definido termina yendo al caso de default.

### Función Agregar:

En esta función agrego la matrices de asignaturas y al vector de notas los valores correspondientes, comprobando al entrar a esta función que la cantidad de materias que hay en el la matriz no haya alcanzado su valor máximo si es así, imprime un mensaje de error y sale de la misma, ahora si todavía no se alcanzó el máximo de las materias entonces comprueba de que de nota ingresada este dentro de los valores esperados (entero con un valor de 1 a 10), y si todo está bien sale y vuelve al menú asignaturas.

### Función Modificar:

La función permitía, después de haber ingresado al menos una materia con su nota, que el usuario modificara las asignaturas ya escritas y sus notas, la misma funciona como la función agregar solamente cambiando que no se le asigna en espacio nuevo, agrandando las opciones, sino que primero se elimina lo que estaba en el espacio seleccionado, para evitar que quedaran caracteres de otras materias. Esta función, a diferencia de las demás, necesito un proceso previo para eliminar los posibles errores en el caso de que se ingrese cualquier otra opción incorrecta, ya sea un carácter, opción fuera del rango disponible.

### Función Eliminar:

Esta función, como lo indica su nombre, se usa para eliminar asignaturas con su nota y, como ocurre en modificar, para modificar una asignatura primero se comprueba de que exista alguna previamente para poder seleccionarla, y de la cual se elimina completamente y se sobrescribe con la materia que tiene debajo suyo.

## **Menú Métricas y sus Funciones:**

Lo que permite este menú es calcular el promedio, la cantidad de materias cursadas, la nota máxima obtenida, al igual que la nota mínima, y la cantidad de aplazos. Para ello, requiere únicamente el ingreso de la estructura a modificar para así pasársela a sus funciones y finalmente devolverla.

Las funciones involucradas requieren únicamente que se les pase la estructura y la cantidad de asignaturas cursadas, la cual es calculada antes de darle elección al usuario, para evitar que haya un error si el usuario pide cualquier otra opción antes de pedir la cantidad de materias.

**Resultados de la Ejecución**

```

¡Bienvenido!
Elija que desea hacer:
    R) Registro Personal
    A) Asignaturas
    M) Calcular métrica individual
    !) Finalizar
    0) Salir

?d
ERROR: Opción ingresada no válida, intenta de nuevo:
Elija que desea hacer:
    R) Registro Personal
    A) Asignaturas
    M) Calcular métrica individual
    !) Finalizar
    0) Salir

?-1
ERROR: Opción ingresada no válida, intenta de nuevo:
Elija que desea hacer:
    R) Registro Personal
    A) Asignaturas
    M) Calcular métrica individual
    !) Finalizar
    0) Salir

?a
Elija que desea hacer:
    +) Ingreso de asignatura.
    !) Salir.

? █

```

Si el usuario ingresa una opción inválida, el programa pone un mensaje de error y pide al usuario de intentar de nuevo. Además podemos ver que si el usuario ingresa una opción en minúscula el programa funciona bien.

```

Elija que desea hacer:
    +) Ingreso de asignatura.
    !) Salir.

? +

Ingrese la asignatura: ingles
Ingrese la nota: 11
ERROR: La nota fue mal ingresada, por favor vuelva a intentar: -1
ERROR: La nota fue mal ingresada, por favor vuelva a intentar: aze
ERROR: La nota fue mal ingresada, por favor vuelva a intentar: 56
ERROR: La asignatura fue mal ingresada.
Elija que desea hacer:
    +) Ingreso de asignatura.
    !) Salir.

? █

```

La nota ingresada debe ser entre 0 y 10, sino el programa pide al usuario de intentar de nuevo tres veces antes de volver al menú asignaturas.

```
Menu Registro. Elija una opción para modificar los datos ingresados.
  A) Apellido y Nombre
  #) Padrón
  C) Carrera
  0) Volver
?c
Ingrese el número correspondiente a su carrera: 45
Queda registrado lo siguiente: Número de Carrera Inválido.

Menu Registro. Elija una opción para modificar los datos ingresados.
  A) Apellido y Nombre
  #) Padrón
  C) Carrera
  0) Volver
?-3
ERROR: Opción ingresada no válida, intenta de nuevo:
Menu Registro. Elija una opción para modificar los datos ingresados.
  A) Apellido y Nombre
  #) Padrón
  C) Carrera
  0) Volver
?67
ERROR: Opción ingresada no válida, intenta de nuevo:
Menu Registro. Elija una opción para modificar los datos ingresados.
  A) Apellido y Nombre
  #) Padrón
  C) Carrera
  0) Volver
?fdsc
ERROR: Opción ingresada no válida, intenta de nuevo:
Menu Registro. Elija una opción para modificar los datos ingresados.
  A) Apellido y Nombre
  #) Padrón
  C) Carrera
  0) Volver
?qfs
ERROR
Elija que desea hacer:
  R) Registro Personal
  A) Asignaturas
  M) Calcular métrica individual
  !) Finalizar
  0) Salir
2
```

Después de tres errores, el programa sale de submenú para volver al menú principal.



```

Elija que desea hacer:
    0) communication de datos (6)
    1) français (9)
    2) analisis (4)
    3) historia (7)
    4) geographia (6)
    5) economia (5)
    -) Eliminar una asignatura.
    +) Ingreso de asignatura.
    !) Salir.
? -

Ingrese el numero de asignatura que desea borrar: 6
ERROR: La eleccion realizada no esta permitida.
Elija que desea hacer:
    0) communication de datos (6)
    1) français (9)
    2) analisis (4)
    3) historia (7)
    4) geographia (6)
    5) economia (5)
    -) Eliminar una asignatura.
    +) Ingreso de asignatura.
    !) Salir.
? █

```

No podemos seleccionar asignaturas que no existen

```

Menu Registro. Elija una opción para modificar los datos ingresados.
    A) Apellido y Nombre
    #) Padrón
    C) Carrera
    0) Volver
?a
Ingrese Apellido: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
Ingrese Nombre: aurelien
Queda registrado lo siguiente: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaa, aurelien
Menu Registro. Elija una opción para modificar los datos ingresados.
    A) Apellido y Nombre
    #) Padrón
    C) Carrera
    0) Volver
?0
Elija que desea hacer:
    R) Registro Personal
    A) Asignaturas
    M) Calcular métrica individual
    !) Finalizar
    0) Salir
?!
aurelien aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa, 1633771
873, Número de Carrera Inválido., 1, 0.0, 1

Elija que desea hacer:
    R) Registro Personal
    A) Asignaturas
    M) Calcular métrica individual
    !) Finalizar
    0) Salir
? █

```

Si ponemos un apellido más largo superior a 50, el programa funciona mal y completa los otros datos automáticamente de manera incorrecta. Lo mismo ocurre en asignaturas como puede ver a continuación.

```

Bienvenido!
Elija que desea hacer:
    R> Registro Personal
    A> Asignaturas
    M> Calcular metrica individual
    ?> Finalizar
    0> Salir
?A
Elija que desea hacer:
    +> Ingreso de asignatura
    ?> Volver.
? +
Ingrese la asignatura: Intruduccion a la ingenieria electrinca orientada a telec
omunicaciones
Ingrese la nota: ERROR: La nota fue mal ingresada, por favor vuelva a intentar:
2
Elija que desea hacer:
    0> Intruduccion a la ingenieria <2>
    -> Eliminar una asignatura
    +> Ingreso de asignatura
    ?> Volver.
? -

```

Aunque en este caso, el gracias al sistema de reintentos, se puede ingresar la nota correcta, aunque el nombre no estará completo.

```

Elija que desea hacer:
    R> Registro Personal
    A> Asignaturas
    M> Calcular metrica individual
    ?> Finalizar
    0> Salir
??
, 0, Ninguna Carrera Seleccionada, 0, -1.5, 0

```

Esto sucede al no ingresar ningún valor en el sistema antes de salir.

```

29> 6 (8)
30> f (6)
31> h (5)
32> h (6)
33> 5 (4)
34> 4 (6)
35> h (5)
36> 56 (7)
37> h (6)
38> d (5)
39> 56 (4)
40> f (4)
41> f (4)
42> 4 (6)
43> d (5)
44> d (3)
45> g (5)
46> h (6)
47> d (4)
48> h (6)
49> d (4)
-> Eliminar una asignatura
+> Ingreso de asignatura
!> Volver.
? +
ERROR: Se a llegado a la maxima cantidad de asignaturas ingresada.Elija que dese
a hacer:
0) Analisis Matematico (10)
1) f (4)
2) h (6)
3) 5 (6)
4) h (5)
5) d (3)
6) + (5)
7) + (4)
8) d (3)
9) g (4)
10) h (5)
11) 6 (4)
12) d (3)
13) j (5)
14) d (3)
15) h (5)
16) d (3)
17) f (4)
18) f (4)
19) f (4)
20) f (3)
21) h (5)
22) f (3)
23) f (3)
24) g (4)
25) d (3)
26) f (3)
27) h (4)
28) d (3)
29> 6 (8)
30> f (6)
31> h (5)
32> h (6)
33> 5 (4)
34> 4 (6)
35> h (5)
36> 56 (7)
37> h (6)
38> d (5)
39> 56 (4)
40> f (4)
41> f (4)
42> 4 (6)
43> d (5)
44> d (3)
45> g (5)
46> h (6)
47> d (4)
48> h (6)
49> d (4)
-> Eliminar una asignatura
+> Ingreso de asignatura
!> Volver.
?
```

Si alguien quiere superar el máximo de materias alcanzadas aparece un mensaje aclarando la situación al usuario.

```

Bienvenido!
Elija que desea hacer:
    R> Registro Personal
    A> Asignaturas
    M> Calcular metrica individual
    ?> Finalizar
    0> Salir
?m
ERROR: No se puede calcular ninguna metrica sin al menos una asignatura ingresad
a
Elija que desea hacer:
    R> Registro Personal
    A> Asignaturas
    M> Calcular metrica individual
    ?> Finalizar
    0> Salir
?_
```

Al intentar calcular métricas sin tener ninguna materia en el sistema aparece un cartel impidiendo la entrada a este menú a menos que se ingrese una asignatura previamente.

## 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 más sencillo de escribir y la legibilidad del código sería mayor.

Sin Structs (Únicamente en el 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_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]

## Lectura de un tipo char e int

En el submenú de asignaturas, como el menú debe cambiar conforme a la cantidad de asignaturas ingresadas, no se podía preseleccionar una casilla para cada asignatura ya que las mismas son reguladas por el usuario. Por lo tanto, se decidió que las opciones para 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 carácter distintivo ya que esos casos se separan de lo que son las asignaturas. Por este motivo, usamos variables tipo char ('+', '-', '!') para identificar estas opciones. Pero esta decisión trajo un problema puesto que el programa tenía que tomar una variable del 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 una variable del tipo char. Por ello, la selección de las opciones con caracteres distintivos no era un problema. Pero surgía otro problema cuando se ingresaba un número, puesto que el programa no lo reconocía como tal. En cambio, lo reconocía como un carácter. Por lo cual, al comparar la opción ingresada, el programa tenía en cuenta el valor que representa sino su equivalente en el código ASCII. Para solucionar 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 si se quería agregar más materias después de eso, podía ocurrir que algún carácter no seleccionado se tome como una variable cualquiera. Para esto encontramos la solución con la función “atoi();” que nos permitió cambiar la variable de ingreso de un solo carácter a una cadena de caracteres.

```

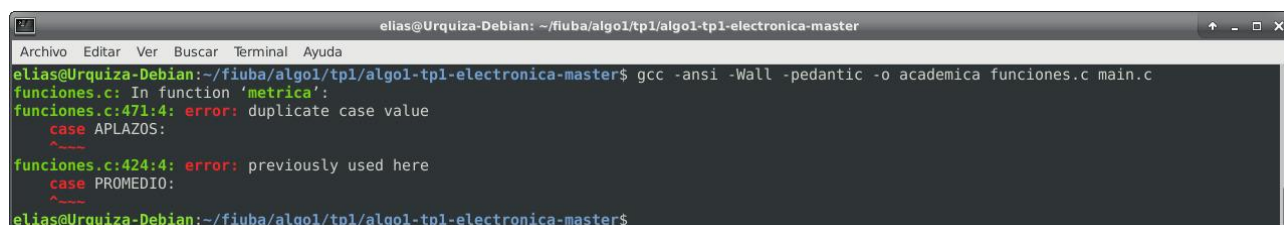
258 NUMERO = atoi(ELEC); /*Modificacion para que se guarde el numero ingresado en vez de su equivalente en ASCII*/
259 if(NUMERO >= CANT_MAT || CANT_MAT == 0 || NUMERO == 0 && (ELEC[0] != ASCII_CERO))
260 {
261     fprintf(stderr, \"%s: %s\\n\", ERR_PREFIJO, ERR_ELEC);
262 }

```

(Imagen de la función con la solución definitiva)

## Problema con los idiomas

A la hora de probar el programa, todo parecía funcionar. Fue entonces que, probando los idiomas, descubrimos que había un error de compilación con el idioma inglés. Al revisarlo mejor, descubrimos que el error provenía del submenú “métricas” donde en un case de un switch se utilizaba el mismo caracter dos veces para representar dos estados distintos. Esto quiere decir que el programa no podría reconocer que opción debía tomar. Esto se pudo arreglar fácilmente al corregir el error y elegir un mejor carácter para las opciones. En la segunda imagen inferior puede notarse que el carácter ‘A’ es utilizado dos veces.

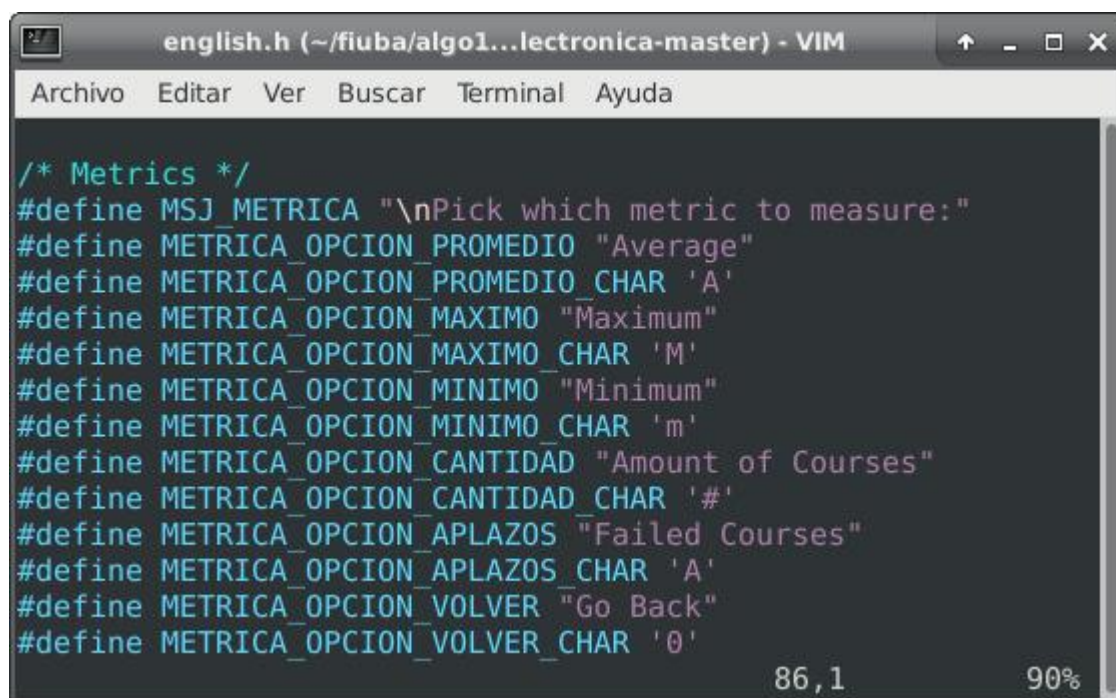


```

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)



```

english.h (~/fiuba/algol...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 del error en el código.)



## **Bibliografía**

- <https://stackoverflow.com/questions/2001626/fgets-function-in-c>
- <https://stackoverflow.com/questions/22608160/atom-change-indentation-mode>
- H. M. Deitel y P. J. Deitel. Como programar en C/C++ y Java. 4ª Edición. Pearson Educación, 2004, ISBN 970-26-0531-8, pág 295 y 296.
- [Repositorio del Grupo](#)

## **Indicaciones de Compilación**

Para compilar el programa se puede escribir lo siguiente en la consola:

```
$ gcc -ansi -Wall -pedantic -o nombrePrograma main.c funciones.c
```

O bien:

```
$ gcc -ansi -Wall -pedantic -c -o funciones.o funciones.c
```

```
$ gcc -ansi -Wall -pedantic -c -o main.o main.c
```

```
$ gcc -ansi -Wall -pedantic -o nombrePrograma main.o funciones.o
```

Posteriormente, se puede correr el programa utilizando `./nombrePrograma`, siendo “nombrePrograma” el nombre que usted desee asignarle al mismo.