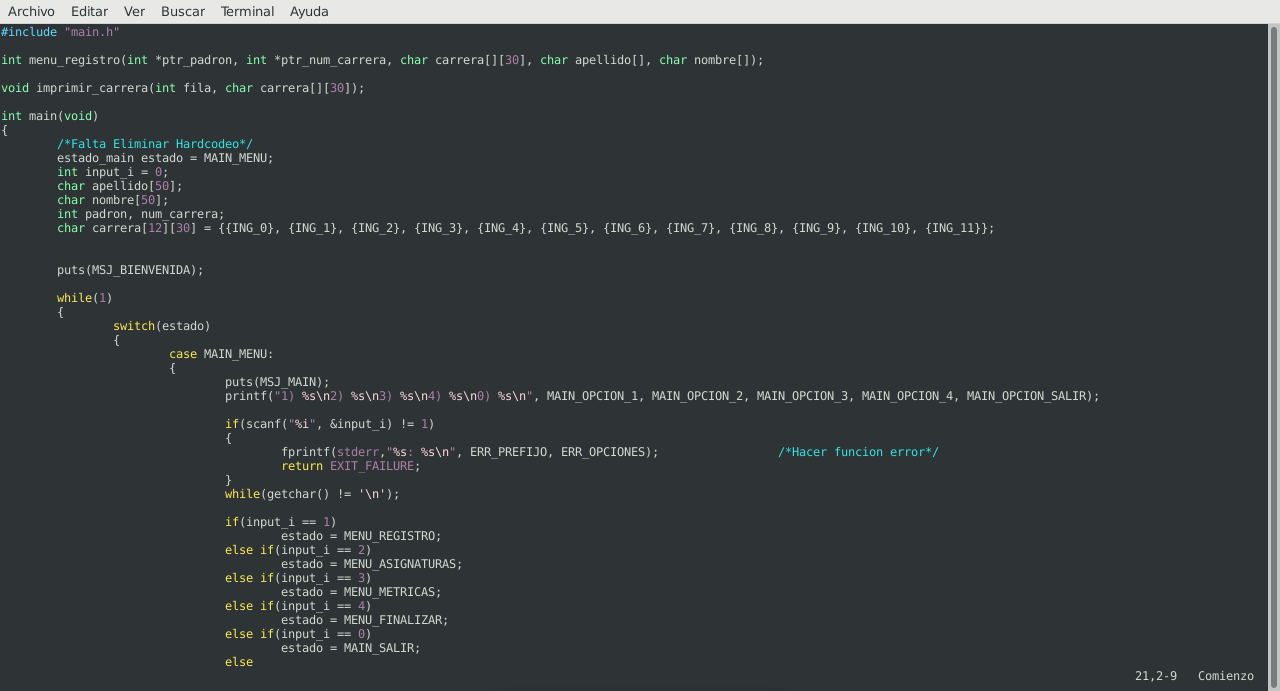
Transición al uso de “structs”:

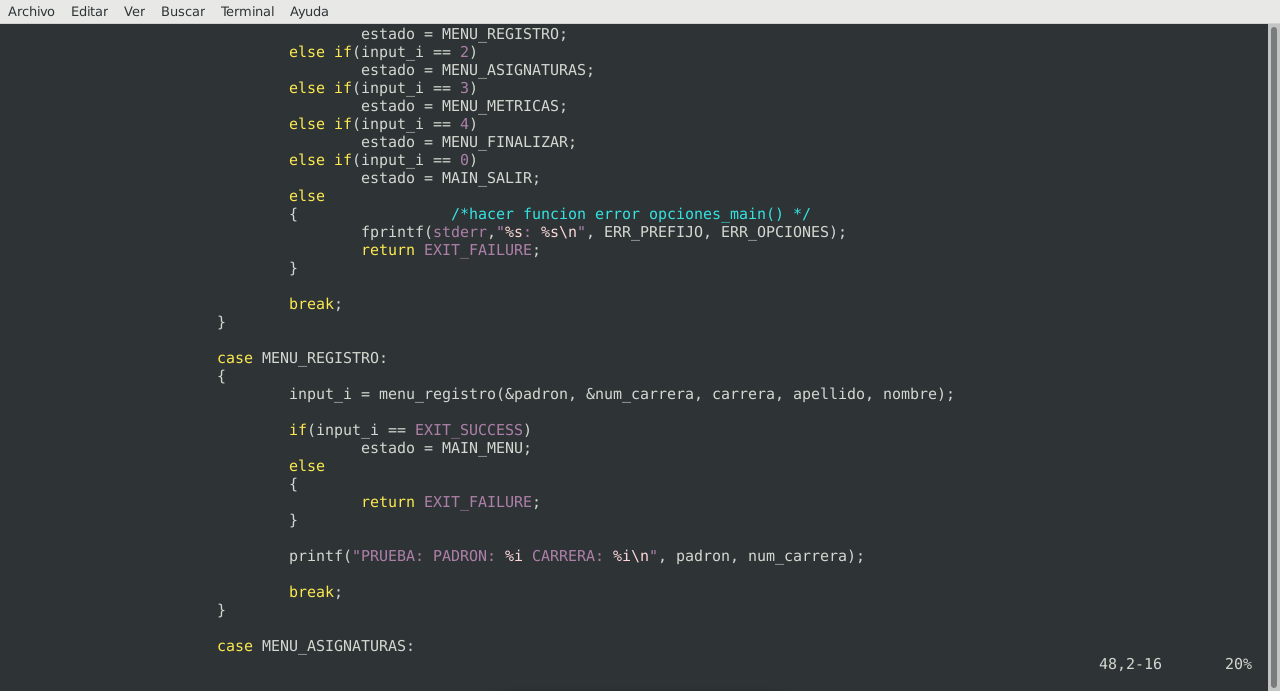
Durante la construcción del sub-menú registro, la idea original era utilizar punteros para dos variables e incluir dos vectores “nombre[]” y “apellido[]” y asimismo una matriz “carreras[][]” tal que el usuario modificara lo que se encontraba en las direcciones de memoria de las variables “padrón” y “num\_carrera”.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

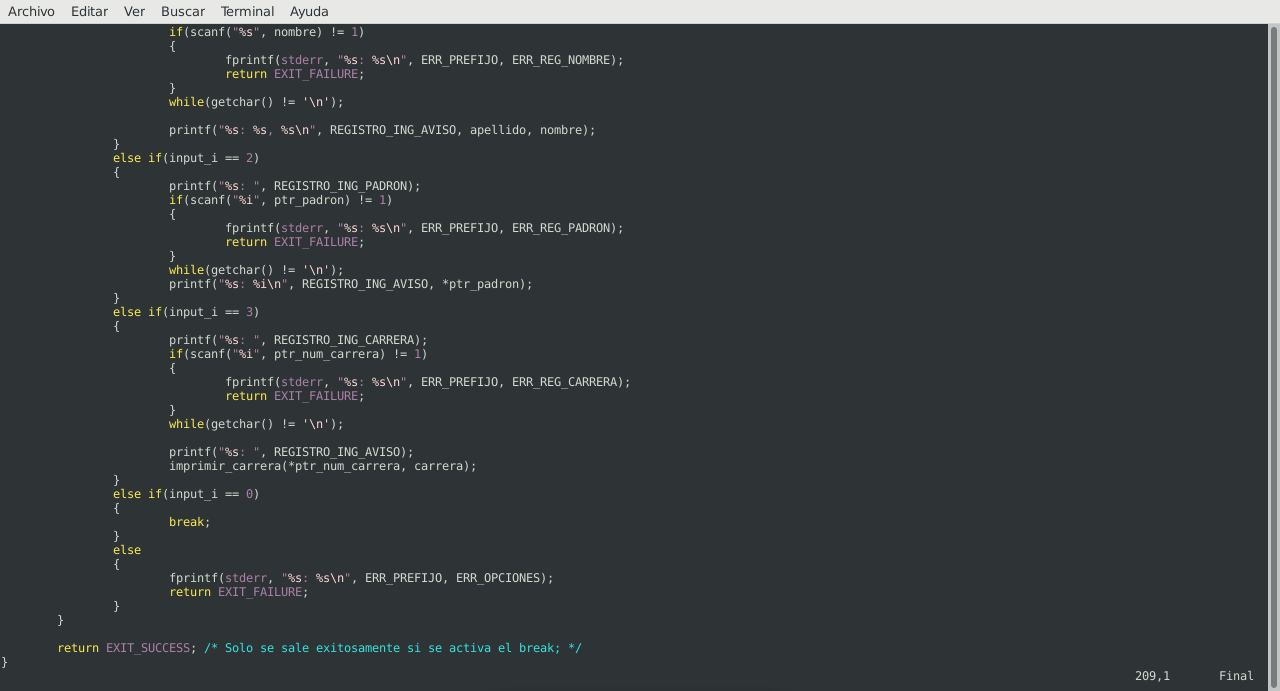
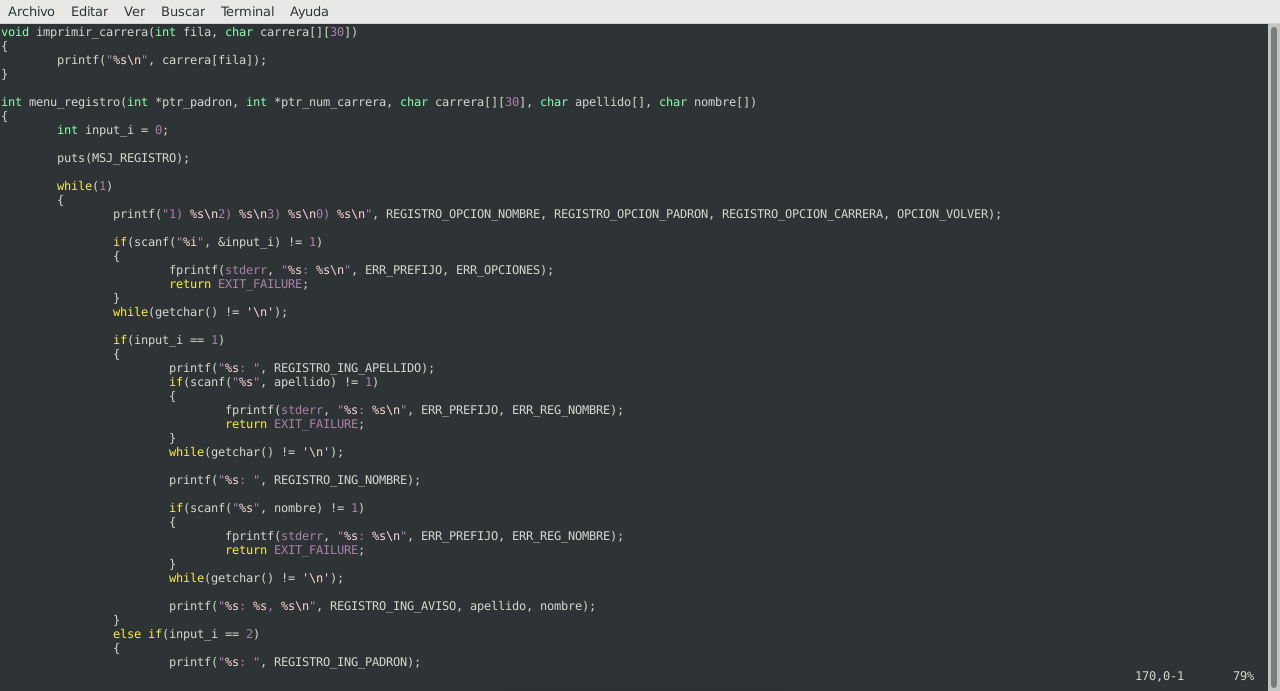
En este momento, por decisión de la mayoría del grupo y tras consultar si era un método viable, pasamos a utilizar una estructura de datos “usuario\_t” tal que el sub-menú Asignaturas fuera más fácil, concluimos, de pasar y recibir datos tal como lo requiere el programa.

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):





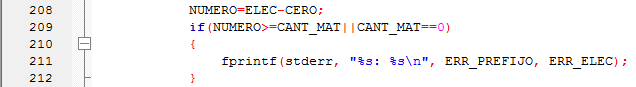


[La versión con Structs se puede encontrar en el código fuente funciones.c]

Lectura de un tipo char e int

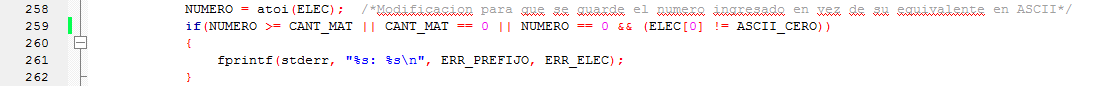
En el submenú de asignaturas, como el menú debe cambiar conforme a la cantidad de asignaturas ingresadas, no se podía preseleccionar una casilla para cada asignatura ya que las mismas son reguladas por el usuario. Por lo tanto, se decidió que las opciones para sobrescribir una asignatura tenían que ser 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 caracter. 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.



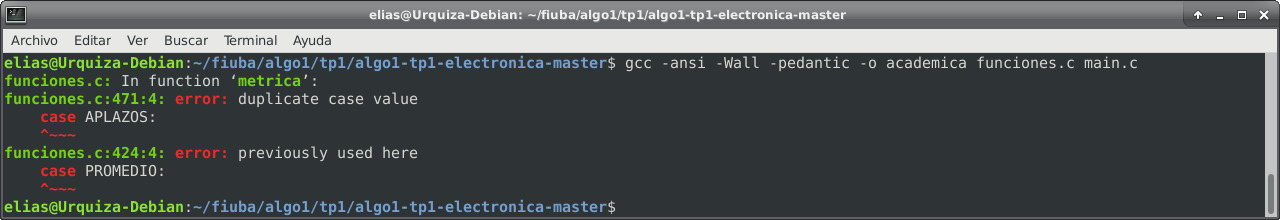
(Imagen del programa con la solución temporal)

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.

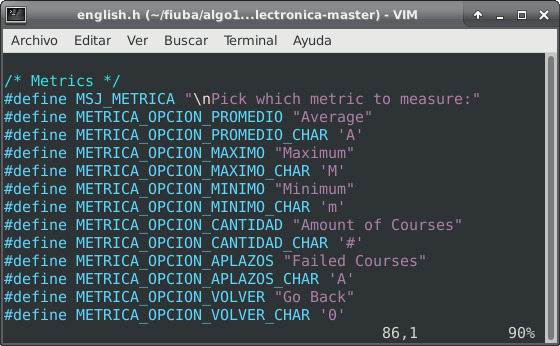
(Imagen de la función con la solución definitiva)

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.



(Lectura del error en la consola)



(Imagen del error en el código.)