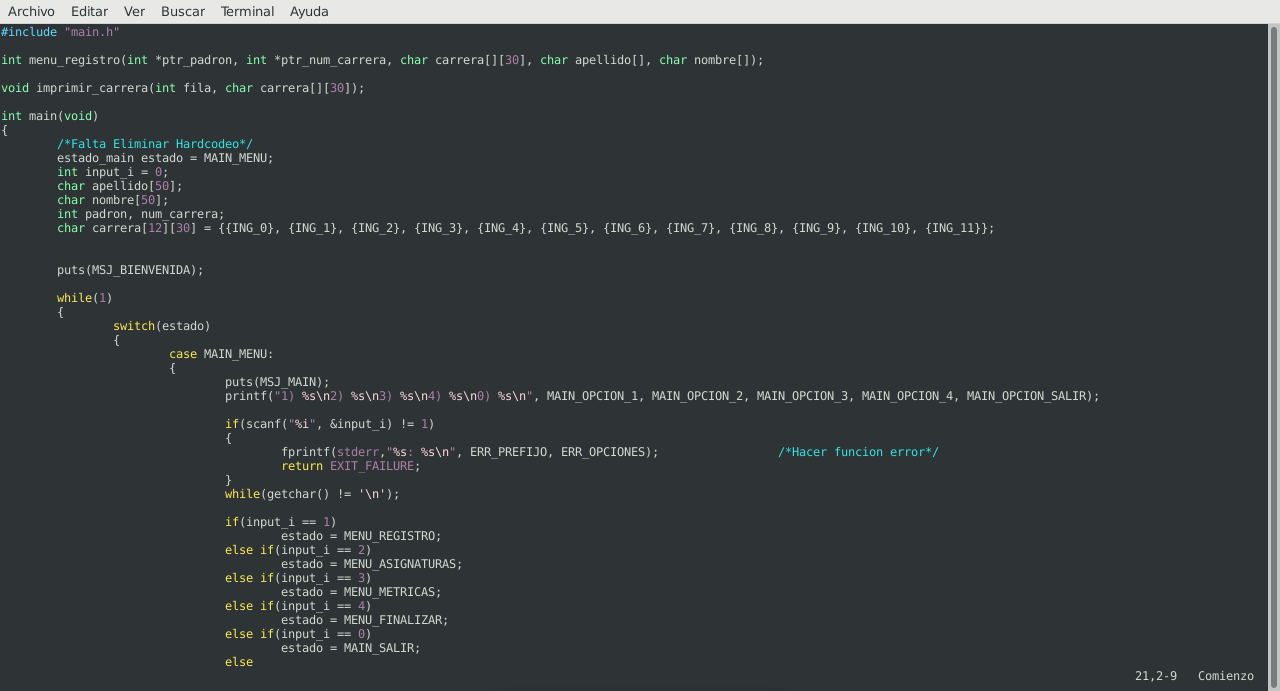
Transición al uso de “structs”:

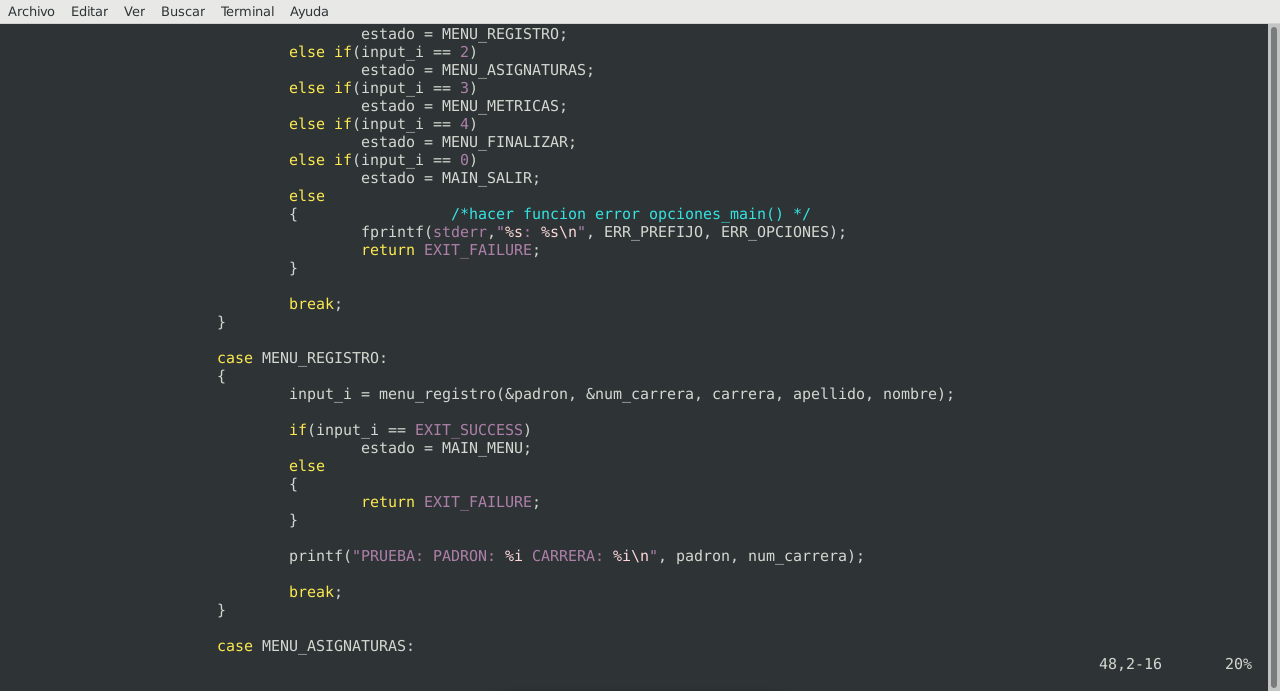
Durante la construcción del sub-menú registro, la idea original era utilizar punteros para dos variables e incluir dos vectores “nombre[]” y “apellido[]” y asimismo una matriz “carreras[][]” tal que el usuario modificara lo que se encontraba en las direcciones de memoria de las variables “padrón” y “num\_carrera”.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

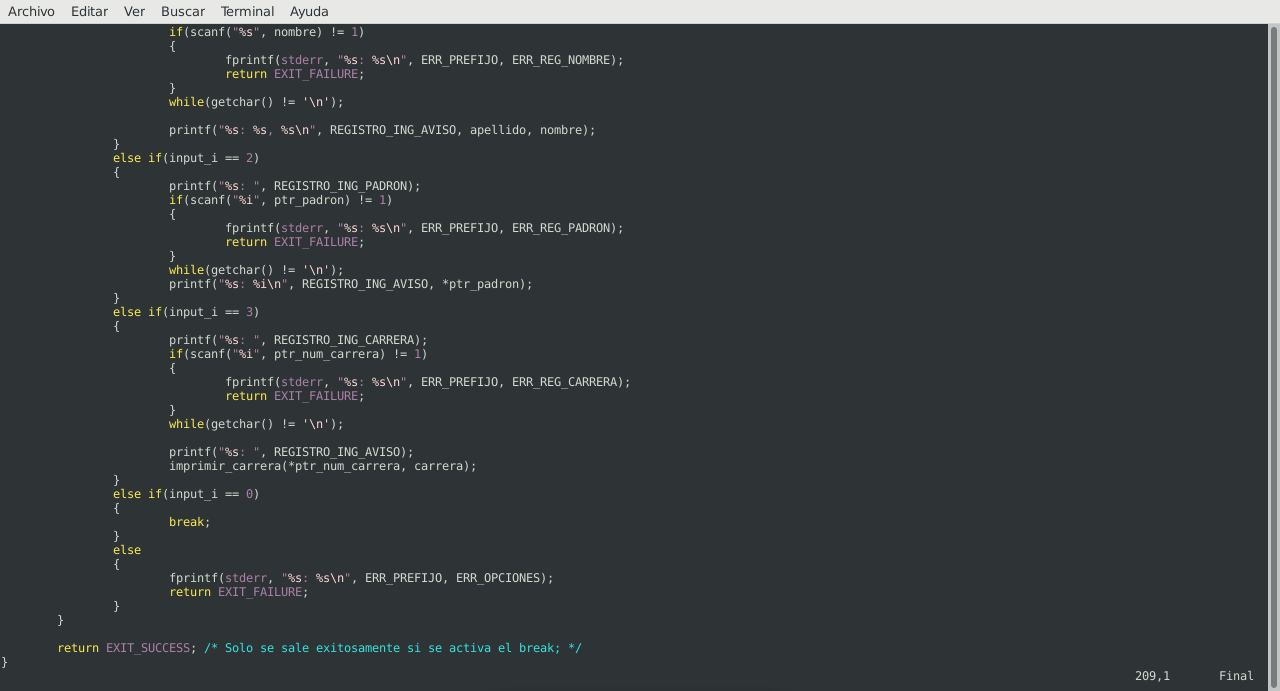
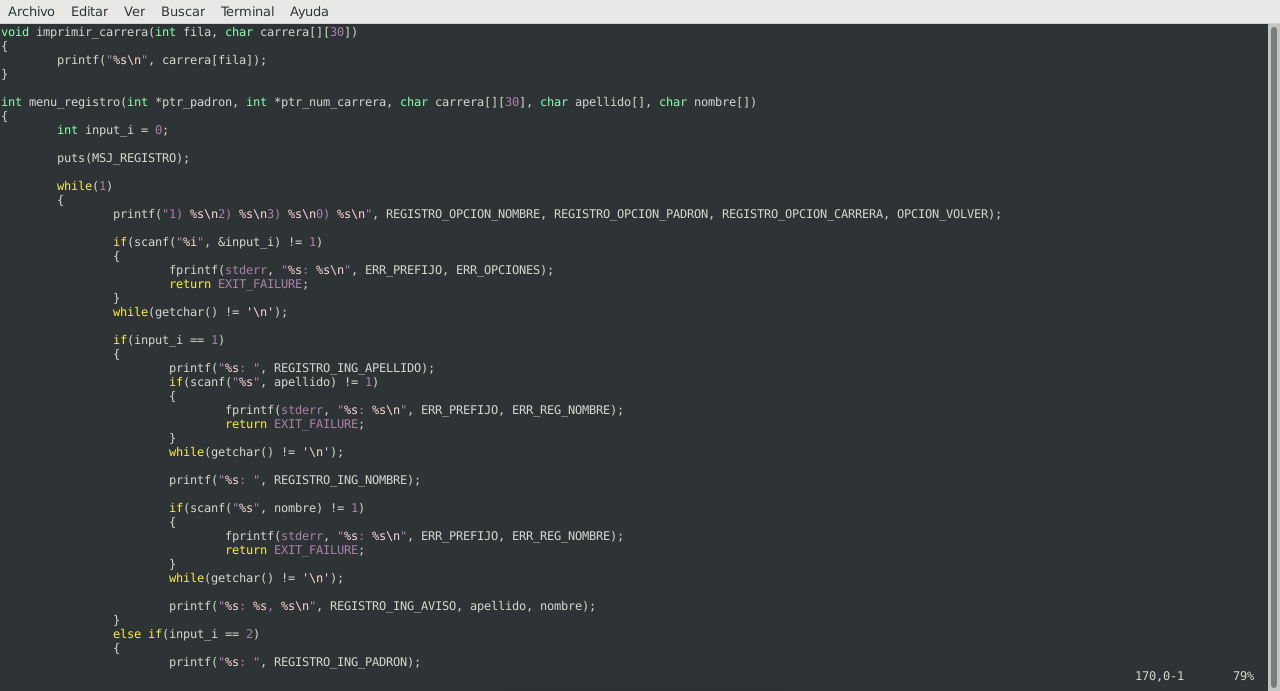
En este momento, por decisión de la mayoría del grupo y tras consultar si era un método viable, pasamos a utilizar una estructura de datos “usuario\_t” tal que el sub-menú Asignaturas fuera más fácil, concluimos, de pasar y recibir datos tal como lo requiere el programa.

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):





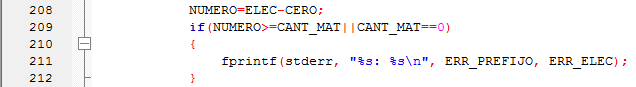


[La versión con Structs se puede encontrar en el código fuente funciones.c]

Lectura de un tipo char e int

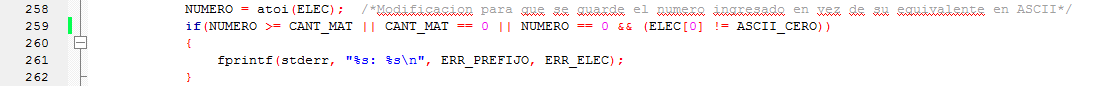
En el submenú de asignaturas como el menú debe cambiar conforme a la cantidad de asignaturas ingresadas no se podía preseleccionar un casa para cada asignatura ya que las mismas son reguladas por el usuario, por lo que se decidió que las opciones para elegir sobrescribir una asignatura tenían que ser 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 caso 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 ocurren en esto fue guardar lo escrito en un tipo char, por lo que la selección de las opciones distintivos 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.



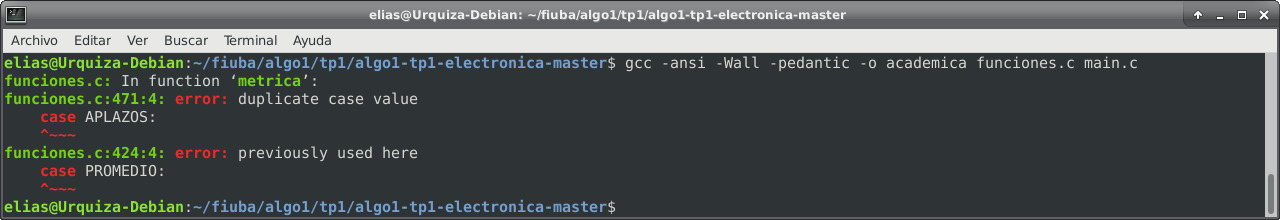
(Imagen del programa con la solución temporal)

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” y cambiando la variable de ingreso de un carácter a una cadena de caracteres.

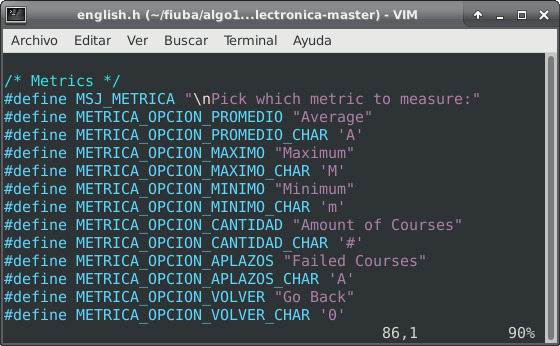
(Imagen de la función con la solución definitiva)

Problema con los idiomas

A la hora de probar el programa, todo 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 sigo 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 él idioma, había una letra que se estaba repitiendo, por lo que para dos casos distintos estabas usando la misma letras, esto se pudo arreglar cambiándolo y ya no se notaba ese problema.



(Lectura del error en la consola)



(Imagen de cual era el error en el codigo)