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 originalmente 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 oportunidades, 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. [ESTO ULTIMO NO ME CONVENCE HASTA SABER QUE ESTA TODO CORREGIDO]

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:

{Mauricio Here}

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.