



UNLaM

Dto. Ingeniería e Investigaciones Tecnológicas

ATENCIÓN:

Este material es continuación del archivo: "03-UsoDePuntoFlotante.pdf", [Semana 3]
/[Uso De Punto Flotante].

Objetivos:

- Entender el problema (necesidad de una función de menú).
- Buscar estrategias de solución.
- Que sea de uso general (no reinventar la rueda).
- Seleccionar una estrategia (la más conveniente).
- Implementar y probar la solución.
- Replanteo de la solución implementada (que tal vez resulte en dos alternativas).
- Enriquecer la base de conocimiento del lenguaje (uso de funciones de biblioteca, macro reemplazos, estilo de programación, etc.).

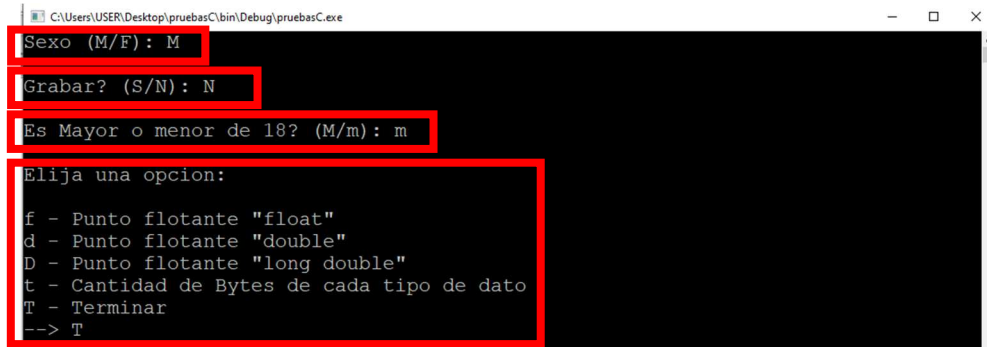
Una función de menú.**Entendiendo el problema:**

En más de una ocasión, en un programa, se puede llegar a requerir que el usuario elija una entre una variedad de opciones. Si en el programa se requiere más de una vez elegir una opción de distintos menús, será conveniente disponer de una función de menú de uso general. No sería muy apropiado hacer una función por cada menú a mostrar (¡no hay que *"reinventar la rueda"* todas las veces!).

El planteo más simple es elegir sólo opción de las mostradas.

Un menú de opciones puede ser tan simple como mostrar un mensaje para que se ingrese el sexo, preguntar si quiere grabar, si es mayor o no de 19, etcétera.

A continuación, se ven algunos modos de uso.



```
C:\Users\USER\Desktop\pruebas\bin\Debug\pruebasC.exe
Sexo (M/F): M
Grabar? (S/N): N
Es Mayor o menor de 18? (M/m): m
Elija una opcion:
f - Punto flotante "float"
d - Punto flotante "double"
D - Punto flotante "long double"
t - Cantidad de Bytes de cada tipo de dato
T - Terminar
--> T
```

Las opciones pueden ser tan simples como para ser respondidas con: ('M' / 'F'); ('S' / 'N'); ('M' / 'm'); ('f' / 'd' / 'D' / 't' / 'T'), para cada uno de los cuatro ejemplos de la "captura de pantalla" precedente.

Esta función debe poder mostrar las diversas opciones del menú. Deben estar claras para el usuario las opciones válidas que debe ingresar, para hacerlo por teclado y devolver la opción elegida.

Si no se ingresa una opción válida, habrá dos posibilidades:

- donde se invoca a la función de menú, cuando ésta devuelva la opción elegida, se hará la validación y si no es válida se la vuelve a invocar. La validación la hace quien invoca, esto no parece ser una buena alternativa. Estos detalles los trata la función de menú.
- la función de menú hace la validación y recién devolverá la opción elegida cuando esta sea válida (esto parece ser una mejor alternativa).

Para poder cumplir con la alternativa elegida (la segunda), todo lo que hay que hacer es invocar a la función con la cadena de caracteres que se deben mostrar (incluyendo, si hubiera necesidad, las marcas de fin de línea '\n').

- Si la opción a elegir fuera un número entero habría que invocar con un rango de enteros (dos argumentos más para la función), ¿se tendrá un serio problema si son valores



UNLaM

Dto. Ingeniería e Investigaciones Tecnológicas

disjuntos! (esto daría un problema más interesante desde el punto de vista de la algoritmia que se requerirá -se deja por ahora-).

- Si fueran letras, no tendrá que ser un rango, podrá ser tan solo un argumento conteniendo cada una de las letras válidas como respuesta a las opciones.

NOTA: si la función de menú debe desplegar 60 o 70 opciones, lo podrá hacer devolviendo una de las 26 letras mayúsculas, más una de las 26 minúsculas (del alfabeto inglés), más uno de los 10 dígitos, más algún carácter como '+', '-', '*', '/', '\', '|', '@', '!', '?'; etcétera.

Así queda enunciada una buena estrategia (la segunda), para que la función de menú reciba sólo dos parámetros con los que cumplir su cometido.

Invocación (1er. argumento)	Observaciones
<pre>menu("A - Alta\n" "B - Baja\n" "M - Modificacion\n" "C - Consulta\n" "L - Listado\n" "S - Salir\n",</pre>	La cadena de caracteres para el mensaje por desplegar se puede cortar en sub - cadenas y ver en la invocación algo <i>parecido</i> a lo que se muestre en pantalla.
Invocación (2do. argumento)	
<pre>"ABMCLS");</pre>	No necesariamente en ese orden, pueden estar en cualquier orden. Lo que devuelva la función se asigna a una variable

Cuando la función comience a ejecutar:

- debe mostrar su primer parámetro tal como lo recibe.
- debe ingresar una opción.
- controlará si ésta está en su segundo parámetro.
 - en el caso que no estuviera, repetirá desde el comienzo,
 - caso contrario, devolverá la opción ingresada.

Se puede ahora comenzar a esbozar el algoritmo:



UNLaM

Dto. Ingeniería e Investigaciones Tecnológicas

```

menu recibe <-- msj, opc
    mostrar msj
    ingresar -> opta
    si opta no-pertenece-a opc
        ir a (mostrar msj)
    si-no
        devolver opta
    fin-si
fin-menu

```

Con esto queda bastante bien planteada la estrategia. que al mejorarla resulta:

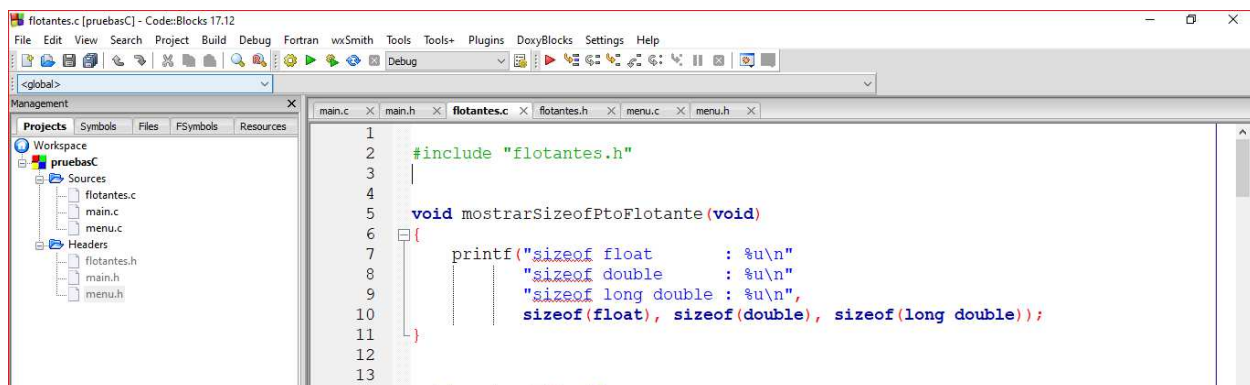
```

menu(msj, opc)
    hacer
        mostrar msj
        ingresar -> opta
    mientras opta no-pertenece-a opc
        devolver opta
    fin-menu

```

Este es el pseudocódigo que más se aproxima a que su codificación, resolviendo la estrategia elegida.

Agregar los fuentes 'menu.c' y 'menu.h' al proyecto [pruebasC].



NOTA:

- **msj**: el mensaje a mostrar es una cadena de caracteres, por lo que será un array de enteros **char**. Una función no puede recibir un array, recibe su dirección de comienzo, en este caso, en un puntero constante porque sólo se lo usa para mostrarlo, no para modificarlo, en el parámetro "**const char *msj**".



UNLaM

Dto. Ingeniería e Investigaciones Tecnológicas

- **opc**: la cadena de caracteres con las opciones válidas, no se la muestra ni modifica, sólo se usa para validar que (coloquialmente *saber si*), la opción que se ingrese esta en ella, se recibirá como `"const char *opc"`.
- antes del ciclo repetitivo deberá estar declarada la variable **opta** que es en la que se ingresa, por teclado, una opción por parte del usuario para luego verificar si es correcta.
- dentro del ciclo repetitivo se muestra, sin cambiarle nada, el **mensaje** recibido.
- la condición del ciclo `"hacer ... mientras ..."` no presentará dificultades. Se determinará si la opción ingresada por teclado está o no dentro de las opciones válidas
- devolver la opción elegida y validada (**opta**) tampoco será problema.

OTRA:

- el mensaje se recibe en el primer parámetro: **msj**, en tanto que las opciones en el segundo parámetro **opc**, y la variable local para la opción ingresada es **opta**.
- si el mensaje se mostrara mediante `"puts(msj);"`, el cursor quedaría en la línea siguiente al mensaje mostrado.
- hacer uso de `"printf(msj);"` es totalmente válido, siempre que el mensaje recibido no tenga un carácter '%', ya que, si estuviera seguido de un carácter de edición, esperaría un argumento más y produciría una violación de memoria inesperada. La alternativa es: `"printf("%s", msj);"`.
- en la condición del `"do ... while ..."` la función de biblioteca `"strchr(opc, opta)"` permite buscar el **char opta** en la cadena (array de enteros **char**), **opc** y si lo encuentra hasta la marca de fin de cadena `'\0'`, devuelve en qué dirección de memoria lo encuentra, de lo contrario devuelve **NULL** (indicando que no lo encontró), de ahí su uso en la condición del ciclo. La dirección de memoria en la que lo encuentra no es de interés, solo importa saber si no lo encuentra, para repetir el ciclo.

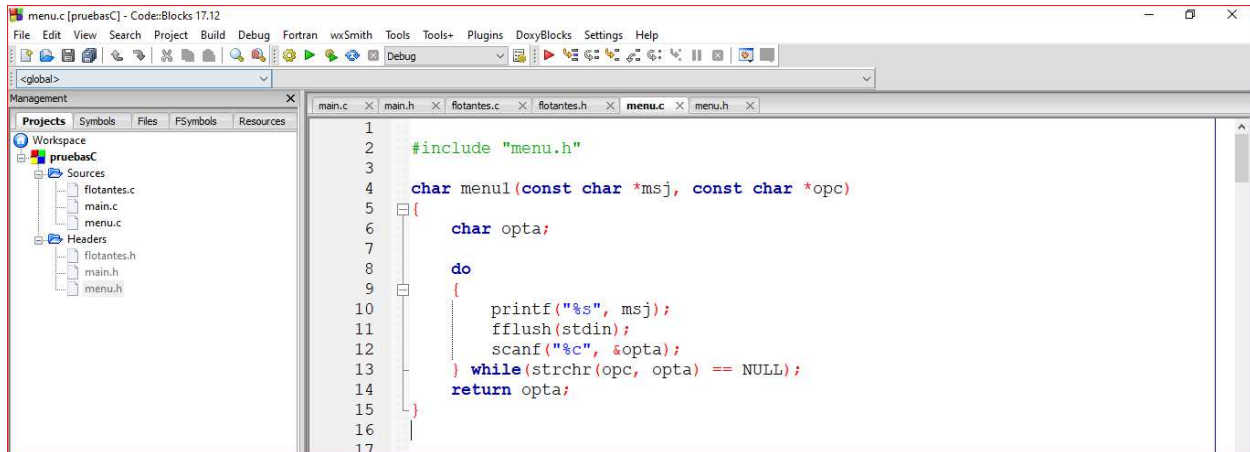


UNLaM

Dto. Ingeniería e Investigaciones Tecnológicas

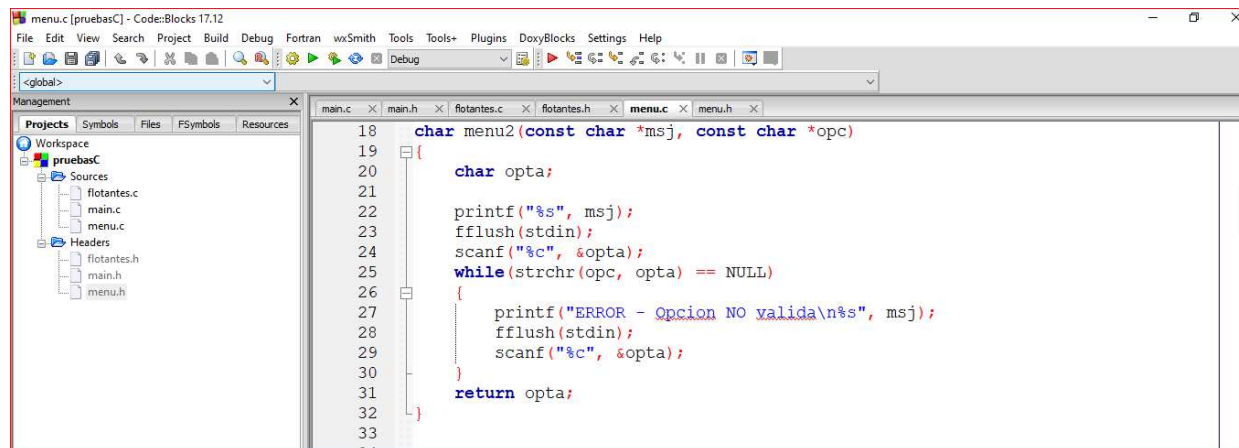
- la condición: `strchr(opc, opta) == NULL` equivale a: `!strchr(opc, opta)`.

La función de menú resultante será:



```
1 #include "menu.h"
2
3
4 char menu1(const char *msj, const char *opc)
5 {
6     char opta;
7
8     do
9     {
10         printf("%s", msj);
11         fflush(stdin);
12         scanf("%c", &opta);
13     } while(strchr(opc, opta) == NULL);
14     return opta;
15 }
```

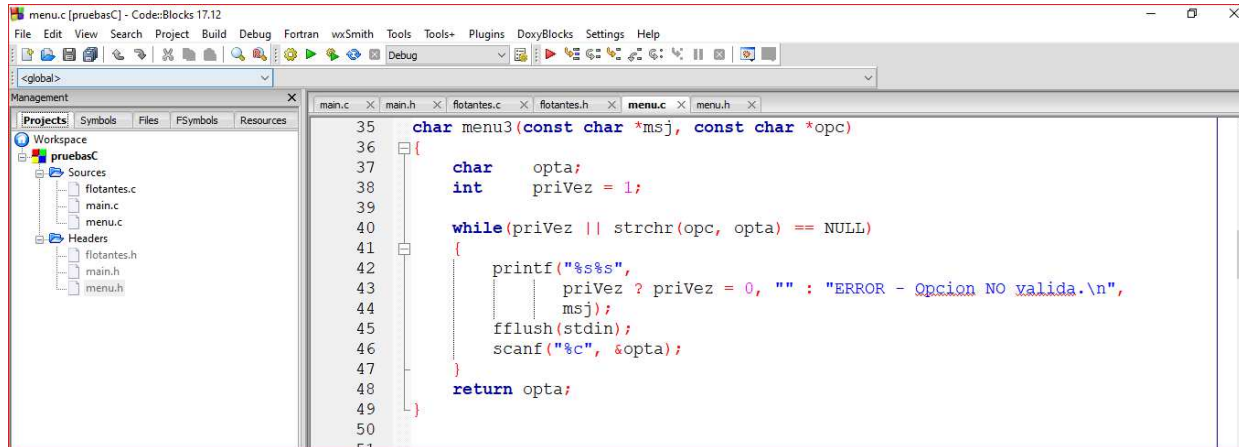
Podría ser necesaria una función de menú que se encargue de mostrar un mensaje de error cuando se ingrese una opción errónea, con lo que se podrían llegar a tener, dos versiones distintas, una con mensaje castigo y otra no.



```
18 char menu2(const char *msj, const char *opc)
19 {
20     char opta;
21
22     printf("%s", msj);
23     fflush(stdin);
24     scanf("%c", &opta);
25     while(strchr(opc, opta) == NULL)
26     {
27         printf("ERROR - Opcion NO valida\n%s", msj);
28         fflush(stdin);
29         scanf("%c", &opta);
30     }
31     return opta;
32 }
```

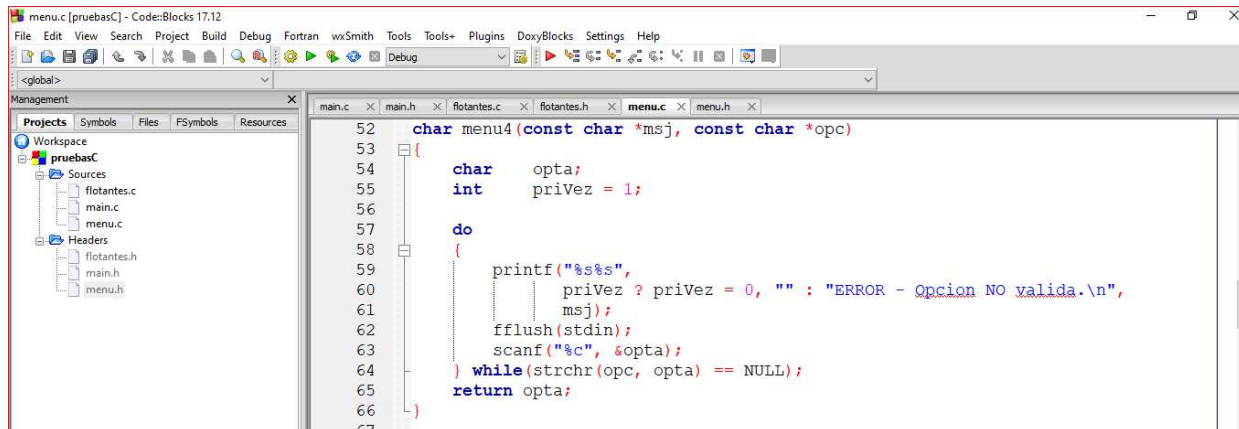
Esta versión con mensaje '*castigo*', se puede mejorar para no duplicar código ...

... quedando (ver más adelante las consideraciones por el uso del operador condicional):



```
35 char menu3(const char *msj, const char *opc)
36 {
37     char    opta;
38     int     priVez = 1;
39
40     while(priVez || strchr(opc, opta) == NULL)
41     {
42         printf("%s%s",
43             priVez ? priVez = 0, "" : "ERROR - Opcion NO valida.\n",
44             msj);
45         fflush(stdin);
46         scanf("%c", &opta);
47     }
48     return opta;
49 }
50
51
```

Esto se puede mejorar, para que no haya una doble condición en el ciclo.



```
52 char menu4(const char *msj, const char *opc)
53 {
54     char    opta;
55     int     priVez = 1;
56
57     do
58     {
59         printf("%s%s",
60             priVez ? priVez = 0, "" : "ERROR - Opcion NO valida.\n",
61             msj);
62         fflush(stdin);
63         scanf("%c", &opta);
64     } while(strchr(opc, opta) == NULL);
65     return opta;
66 }
67
```

Esta ya es una versión bastante depurada, aunque algo críptica, por el uso del operador condicional. Seguramente dejará pensando mucho a un programador 'junior' y a algunos 'semi senior'.

Es bastante depurada porque ...

- ... la condición del ciclo repetitivo deja de tener una condición compuesta para pasar a tener una condición simple.
- ... no hay duplicación de código como en la segunda versión.



UNLaM

Dto. Ingeniería e Investigaciones Tecnológicas

- ... sin duplicar código, dado que a veces (la primera vez), no hay que mostrar un mensaje de error, queda un código bastante compacto al utilizar el operador ternario, que por verdad (la primera vez), pone en no-verdadera a la señal (`privéz = 0`), y no muestra el mensaje de error.

Es algo críptica porque ...

el operador ternario evalúa si su condición es verdadera (`privéz`).

si es verdadera (evalúa dos expresiones separadas por coma).

pone en falso la señal "`privéz = 0`" (o con "`privéz = !privéz`").

evalúa (o toma), la cadena vacía para ser mostrada.

si es falsa.

evalúa (o toma), la cadena del mensaje de error, para ser mostrada.

Y, en definitiva, resulta totalmente equivalente a ...

```
69 char menu5(const char *msj, const char *opc)
70 {
71     char opta;
72     int privéz = 1;
73
74     do
75     {
76         if(!privéz)
77         {
78             privéz = !privéz;
79             puts("ERROR - Opcion NO valida.");
80         }
81         printf("%s", msj);
82         fflush(stdin);
83         scanf("%c", &opta);
84     } while(strchr(opc, opta) == NULL);
85     return opta;
86 }
87
```

... ¡ya que la evaluación se hace siempre!, utilice o no el operador terciario o condicional.

CONCLUSIONES: El caso de `menu4` es el de escribir un código compacto (aunque críptico), e ilustrar el uso del operador condicional. La función `menu1`, es una muy buena alternativa cuando no se quiere mostrar un mensaje 'castigo'. Si se quiere hacer notar al operador el error en la opción ingresada, desde `menu2` hasta `menu5` se hace una



evolución en la que se desarrolla y muestra capacidad de síntesis. La función **menu4**, a pesar de lo críptico, ¿podría ser la que se elija cuando debe haber un mensaje de error, en lugar de **menu5**? De no haber mensaje de error, menu1 será la elegida.

Si se necesita disponer de dos versiones (con mensaje y sin mensaje), en "**mensajes.h**" escriba ambos prototipos (declaraciones), de las funciones, y por ahora (luego se eliminará), la declaración de "**probarMenu**".

A screenshot of the Code::Blocks IDE interface. The title bar reads 'menu.h [pruebasC] - Code::Blocks 17.12'. The menu bar includes File, Edit, View, Search, Project, Build, Debug, Fortran, wxSmith, Tools, Tools+, Plugins, DoxyBlocks, Settings, and Help. The toolbar contains icons for file operations, compilation, and debugging. The 'Management' pane on the left shows a project named 'pruebasC' with sub-items for Sources and Headers. The main editor window displays the content of 'menu.h' with the following code:

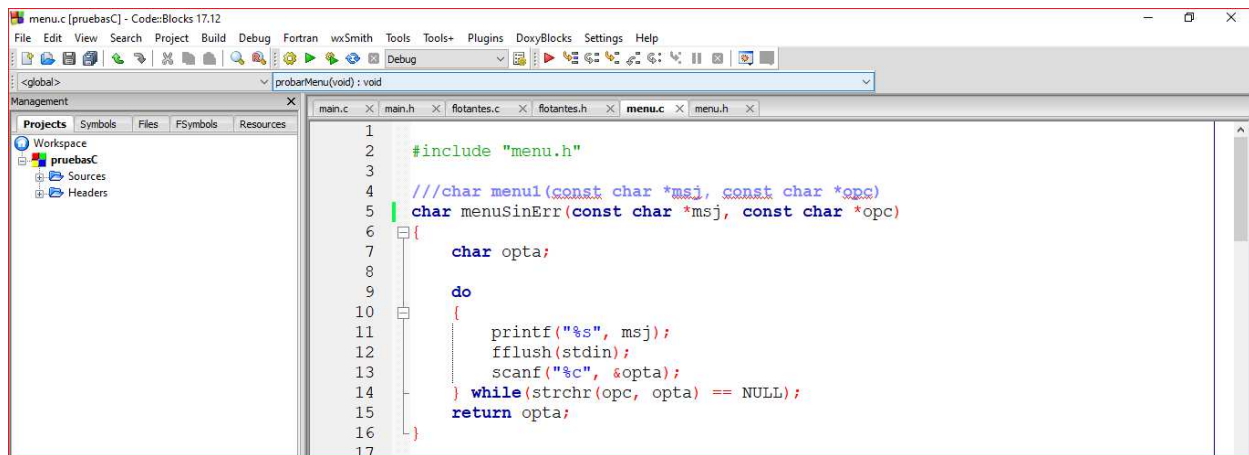
```
1  #ifndef MENU_H_
2  #define MENU_H_
3
4  #include <stdio.h>
5  #include <string.h>
6
7
8  /** funcion de menu que permite el ingreso de una opcion valida
9  ** sin mensaje de error
10 **/
11 char menuSinErr(const char *msj, const char *opc);
12
13 /** funcion de menu que permite el ingreso de una opcion valida
14 ** con mensaje de error
15 **/
16 char menuConErr(const char *msj, const char *opc);
17
18 /** funcion para probar distintas alternativas de funciones de menu
19 **/
20 void probarMenu(void);
21
22
23 #endif // MENU_H_
24
```



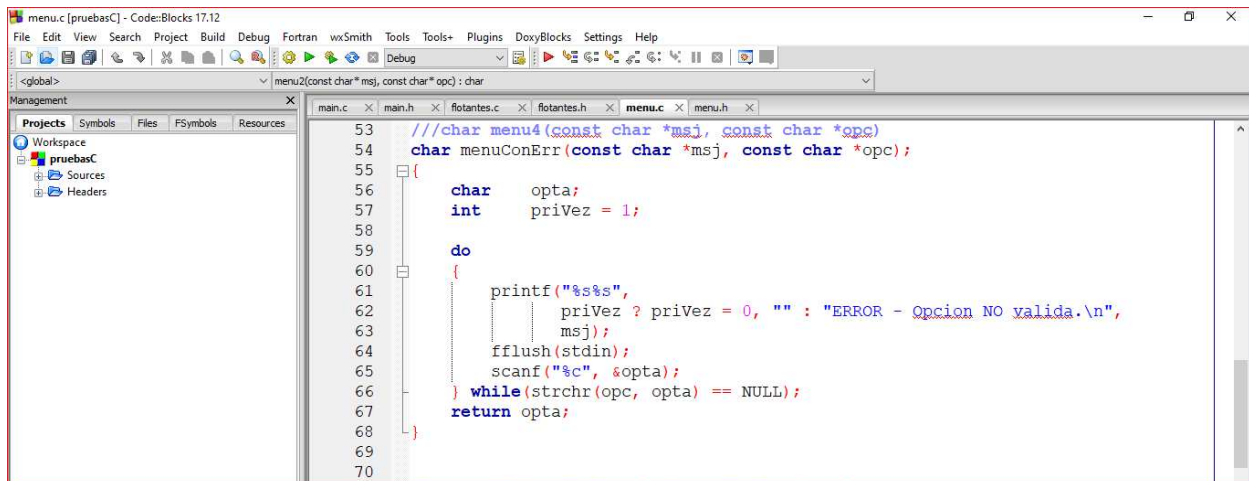
UNLaM

Dto. Ingeniería e Investigaciones Tecnológicas

Copiar los prototipos en "menu.c" reemplazando el encabezado de las funciones "menu1" y "menu4", que es la "mejor lograda" (o si lo prefiere el de "menu5"). No olvidar de quitar el ";" y dejar *comentado*.



```
1  #include "menu.h"
2
3  //char menu1(const char *msj, const char *opc)
4  char menuSinErr(const char *msj, const char *opc)
5  {
6      char opta;
7
8      do
9      {
10         printf("%s", msj);
11         fflush(stdin);
12         scanf("%c", &opta);
13     } while(strchr(opc, opta) == NULL);
14     return opta;
15 }
16
17
```



```
53 //char menu4(const char *msj, const char *opc)
54 char menuConErr(const char *msj, const char *opc);
55 {
56     char opta;
57     int priVez = 1;
58
59     do
60     {
61         printf("%s%s",
62             priVez ? priVez = 0, "" : "ERROR - Opcion NO valida.\n",
63             msj);
64         fflush(stdin);
65         scanf("%c", &opta);
66     } while(strchr(opc, opta) == NULL);
67     return opta;
68 }
69
70
```



UNLaM

Dto. Ingeniería e Investigaciones Tecnológicas

Y transitoriamente (luego se eliminará o no), la declaración de "**probarMenu**", escrita con fines de prueba.

```
91 void probarMenu(void)
92 {
93     char opcion;
94
95     opcion = menuSinErr("Graba? (S/N): ", "SN");
96     printf("Opcion elegida: %c\n", opcion);
97
98     opcion = menuSinErr("Elija una opcion\n\n"
99                         "A - Alta\n"
100                        "B - Baja\n"
101                        "C - Consulta\n"
102                        "M - Modificacion\n"
103                        "L - Listado\n"
104                        "S - Salir\n"
105                        "--> ",
106                        "ABMCSL");
107     printf("Opcion elegida: %c\n", opcion);
108
109 }
110
111
```

Finalmente, en la función "**main**" de "**main.c**"

```
1
2 #include "main.h"
3
4 int main()
5 {
6     probarMenu();
7     return 0;
8 }
9
10
```

Y al ejecutar el programa, resulta en la salida por pantalla que sigue:

```
C:\Users\USER\Desktop\pruebas\bin\Debug\pruebasC.exe
Graba? (S/N): s
Graba? (S/N): n
Graba? (S/N): P
Graba? (S/N): N
Opcion elegida: N
Elija una opcion

A - Alta
B - Baja
C - Consulta
M - Modificacion
L - Listado
S - Salir
--> a
Elija una opcion

A - Alta
B - Baja
C - Consulta
M - Modificacion
L - Listado
S - Salir
--> S
Opcion elegida: S

Process returned 0 (0x0)   execution time : 54.037 s
Press any key to continue.
```

UNA Pequeña modificación, si las opciones a elegir corresponden a letras que no se repiten en mayúscula / minúscula, se puede pasar a mayúscula (o a minúscula), la opción devuelta por la función de menú, sin necesidad de *obligar* a ingresarla en mayúscula como en este caso. Esto se hace en la invocación a la función.

OTRA Pequeña modificación, el segundo menú, casi seguramente estará dentro de un ciclo repetitivo mientras no quiera dar por terminado el ingreso de opciones.



UNLaM

Dto. Ingeniería e Investigaciones Tecnológicas

```

91 void probarMenu(void)
92 {
93     char opcion;
94
95     opcion = menuSinErr("Graba? (S/N): ", "SNsn");
96     opcion = toupper(opcion);
97     printf("Opcion elegida: %c\n", opcion);
98
99     do
100     {
101         opcion = menuSinErr("Elija una opcion\n"
102                             "A - Alta\n"
103                             "B - Baja\n"
104                             "C - Consulta\n"
105                             "M - Modificacion\n"
106                             "L - Listado\n"
107                             "S - Salir\n"
108                             "--> ",
109                             "ABMCSLlscmah");
110         opcion = toupper(opcion);
111         printf("Opcion elegida: %c\n", opcion);
112     } while(opcion != 'S');
113 }
114
115

```

Por el uso de `"toupper"` (o si prefiere `"tolower"`), no olvidar la biblioteca (a continuación, subrayado en rojo) ...

```

1  #ifndef MENU_H_
2  #define MENU_H_
3
4  #include <stdio.h>
5  #include <string.h>
6  #include <ctype.h>
7
8
9  /** funcion de menu que permite el ingreso de una opcion valida

```

Para ambos menús, la respuesta u opción elegida podrá ser en mayúscula o en minúscula.

En el primer caso se utilizará (por ejemplo), para asignar al miembro sexo de una variable del tipo persona, con lo que siempre se asignará en mayúscula como en este caso (o si se prefiere en minúscula).

En el segundo caso, en el lugar del `"printf"`, irá tal vez el `"switch"` que ejecute las distintas opciones ...

Las letras, en ambos casos, pueden estar en cualquier orden (siempre que estén).

En el primer caso: "SNsn" o "SsNn" o ...

En el segundo: "ABMCSLlscmab" o "ABMCLSabmc1s" o "AaBbMmCcLlSs" o ...

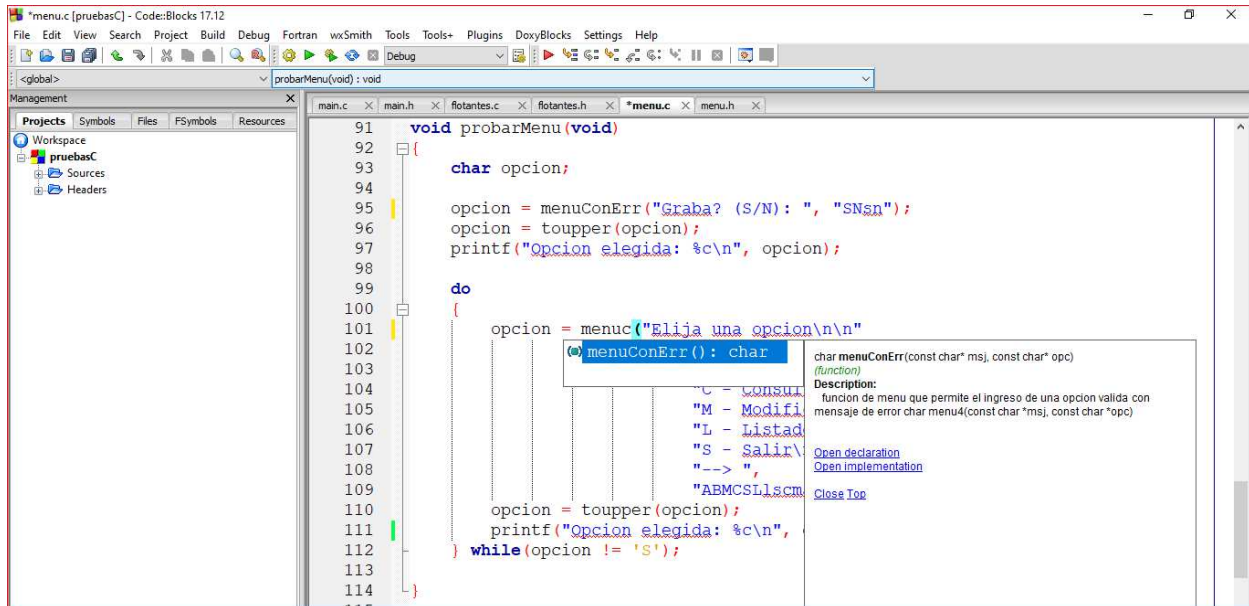
```
C:\Users\USER\Desktop\pruebas\bin\Debug\pruebasC.exe
Graba? (S/N): s
Opcion elegida: S
Elija una opcion

A - Alta
B - Baja
C - Consulta
M - Modificacion
L - Listado
S - Salir
--> l
Opcion elegida: L
Elija una opcion

A - Alta
B - Baja
C - Consulta
M - Modificacion
L - Listado
S - Salir
--> s
Opcion elegida: S

Process returned 0 (0x0)   execution time : 16.226 s
Press any key to continue.
```


Prueba del menú con mensaje de error (o coloquialmente "*mensaje castigo*"):

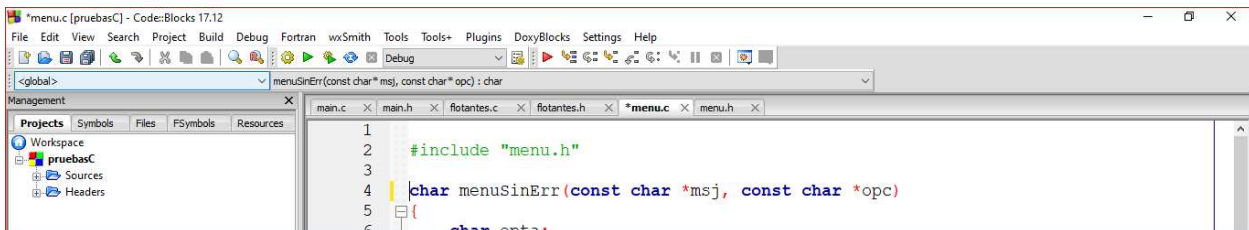


```

91 void probarMenu(void)
92 {
93     char opcion;
94
95     opcion = menuConErr("Graba? (S/N): ", "SNsn");
96     opcion = toupper(opcion);
97     printf("Opcion elegida: %c\n", opcion);
98
99     do
100     {
101         opcion = menuConErr("Elija una opcion\n\n"
102                             "C - Consultar\n"
103                             "M - Modificar\n"
104                             "L - Listado\n"
105                             "S - Salir\n"
106                             "--> ", "ABMCSLscm");
107
108         opcion = toupper(opcion);
109         printf("Opcion elegida: %c\n",
110               );
111     } while(opcion != 'S');
112 }
113
114
115
  
```

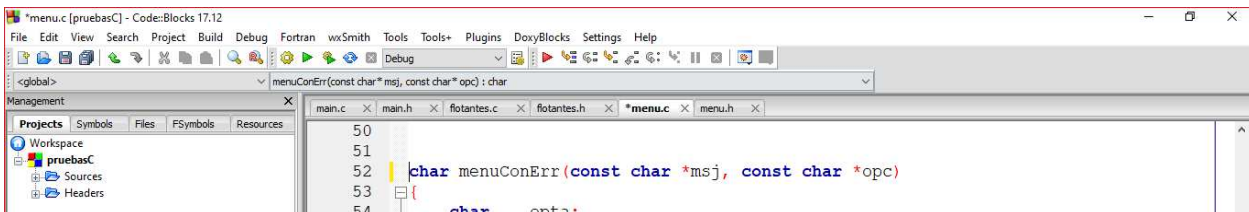
char menuConErr(const char* msj, const char* opc)
 (function)
 Description:
 funcion de menu que permite el ingreso de una opcion valida con mensaje de error char menu4(const char* msj, const char* opc)
[Open declaration](#)
[Open implementation](#)
[Close Top](#)

Ya debe haber notado que, al comenzar a escribir la invocación a la función, se despliega el comentario de documentación escrito en la declaración (prototipo), de la función en "**menu.h**" (comienza con `/*`); al que le agrega lo que dejamos al comienzo de la definición (desarrollo), de la función en "**menu.c**" (comienza con `///`). Sería conveniente eliminar estos últimos.



```

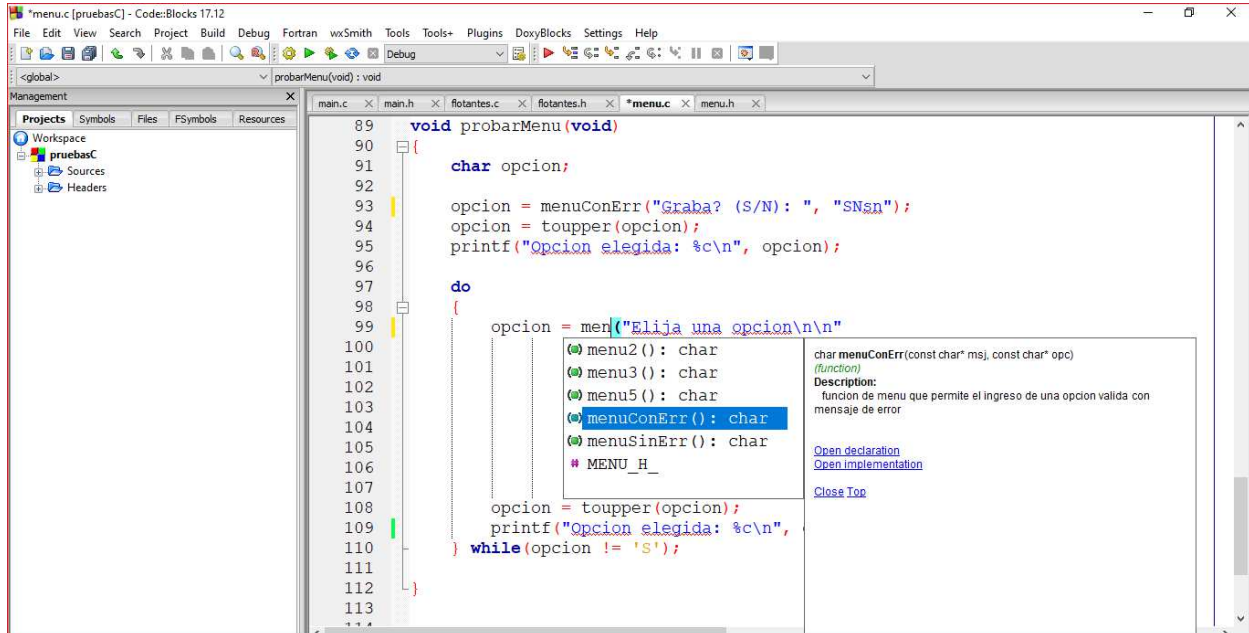
1
2 #include "menu.h"
3
4 char menuSinErr(const char *msj, const char *opc)
5 {
6     char opta;
  
```



```

50
51
52 char menuConErr(const char *msj, const char *opc)
53 {
54     char opta;
  
```


Tras eliminar los comentarios de documentación de las definiciones de las funciones:

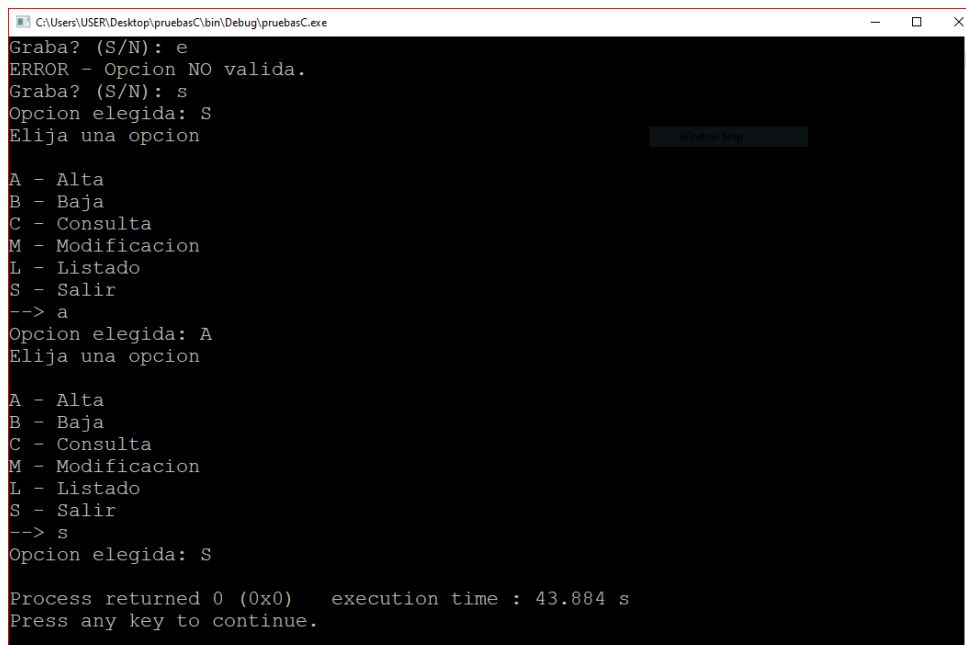


```

89 void probarMenu(void)
90 {
91     char opcion;
92
93     opcion = menuConErr("Graba? (S/N): ", "SNsn");
94     opcion = toupper(opcion);
95     printf("Opcion elegida: %c\n", opcion);
96
97     do
98     {
99         opcion = menuConErr("Elija una opcion\n\n", "AaBbCcMmLlSs");
100
101         // menu2(): char
102         // menu3(): char
103         // menu5(): char
104         // menuConErr(): char
105         // menuSinErr(): char
106         // MENU_H_
107
108         opcion = toupper(opcion);
109         printf("Opcion elegida: %c\n", opcion);
110     } while(opcion != 'S');
111 }
112
113
  
```

char menuConErr(const char* msg, const char* opc)
 (function)
 Description:
 funcion de menu que permite el ingreso de una opcion valida con mensaje de error
[Open declaration](#)
[Open implementation](#)
[Close Top](#)

Al ejecutarlo se tendrá, por ejemplo, la siguiente salida por pantalla



```

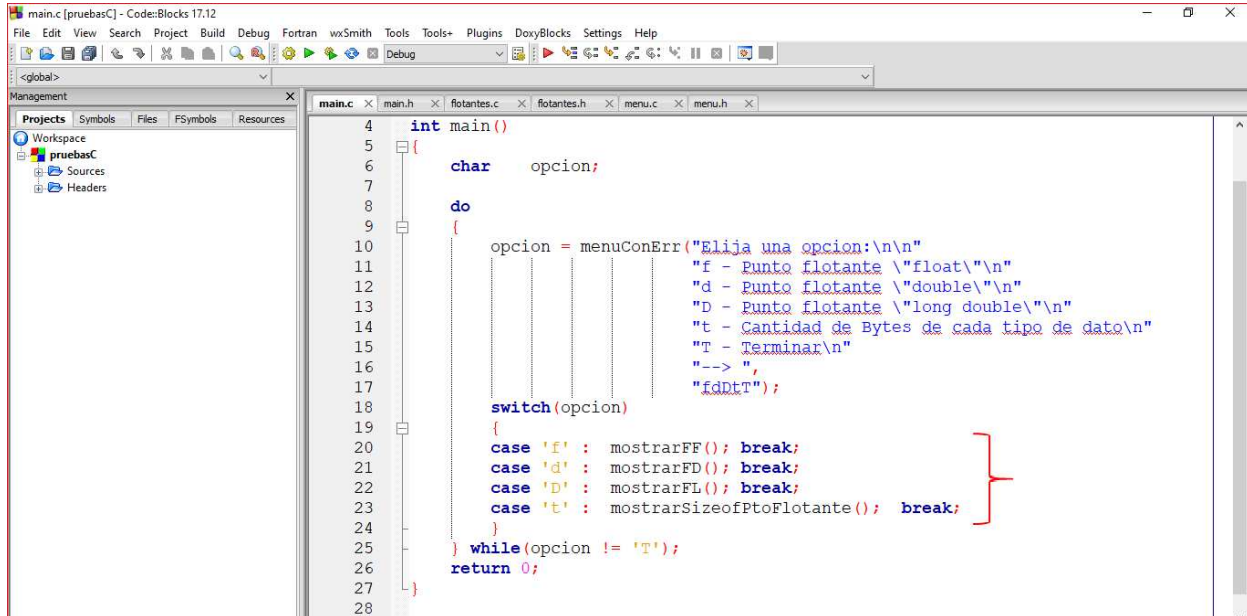
C:\Users\USER\Desktop\pruebasC\bin\Debug\pruebasC.exe
Graba? (S/N): e
ERROR - Opcion NO valida.
Graba? (S/N): s
Opcion elegida: S
Elija una opcion

A - Alta
B - Baja
C - Consulta
M - Modificacion
L - Listado
S - Salir
--> a
Opcion elegida: A
Elija una opcion

A - Alta
B - Baja
C - Consulta
M - Modificacion
L - Listado
S - Salir
--> s
Opcion elegida: S

Process returned 0 (0x0)   execution time : 43.884 s
Press any key to continue.
  
```

Modificando "main" de "main.c".



```
4 int main()
5 {
6     char opcion;
7
8     do
9     {
10         opcion = menuConErr("Elija una opcion:\n\n"
11                             "f - Punto flotante \"float\"\n"
12                             "d - Punto flotante \"double\"\n"
13                             "D - Punto flotante \"long double\"\n"
14                             "t - Cantidad de Bytes de cada tipo de dato\n"
15                             "T - Terminar\n"
16                             "--> ",
17                             "fdDtT");
18
19         switch(opcion)
20         {
21             case 'f' : mostrarFF(); break;
22             case 'd' : mostrarFD(); break;
23             case 'D' : mostrarFL(); break;
24             case 't' : mostrarSizeofPtoFlotante(); break;
25         }
26     } while(opcion != 'T');
27     return 0;
28 }
```

NOTAS:

- el estilo mostrado (en la codificación), es un estilo habitual.
- en las opciones del "**switch ... case ...**", se admite que, si se ejecuta sólo una expresión (en este caso, invocar una función), ésta se escriba en el mismo renglón del "**case ... :**" junto con el "**break;**". (ver } en la figura anterior).
- la llave abierta del "**switch**", en su propio renglón en la misma columna.
- los distintos "**case ... :**", si lo hubiera el "**default ... :**" y la llave cerrada "}" encolumnados del mismo modo.
- si el mensaje utilizado como argumento en la invocación a la función de menú ocupara varios renglones en el código, debería considerarse el uso de un macro reemplazo (ver a continuación), y además, manteniendo un estilo, también el macro reemplazo para las opciones válidas.



- si el macro reemplazo requiere más de un renglón, llevará al final la marca de continuación de macro: "\". En el último renglón, no lleva la marca de continuación.

```

1  #ifndef MAIN_H_
2  #define MAIN_H_
3
4  #include <stdio.h>
5  #include <stdlib.h>
6
7  #include "flotantes.h"
8  #include "funciones.h"
9  #include "menu.h"
10
11 #define MENU_MSJ1 "Elija una opcion:\n\n" \
12                  "f - Punto flotante \"float\"\n" \
13                  "d - Punto flotante \"double\"\n" \
14                  "D - Punto flotante \"long double\"\n" \
15                  "t - Cantidad de Bytes de cada tipo de dato\n" \
16                  "T - Terminar\n" \
17                  "--> "
18
19 #define MENU_OPC1 "fdDtT"
20
21 |
22 #endif // MAIN_H_
23

```

Utilizados en la función "main" ...

```

1  #include "main.h"
2
3  int main()
4  {
5      char opcion;
6
7      do
8      {
9          opcion = menuConErr(MEN
10          switch(opcion)
11          {
12              case 'f': mostra
13              case 'd': mostra
14              case 'D': mostra
15              case 't': mostra
16          }
17          while(opcion != 'T')
18      } while(opcion != 'T')
19      return 0;
20  }
21
22

```

Macro definition for MENU_MSJ1:

```

MENU_MSJ1
"Elija una opcion:\n\n" "f - Punto flotante \"float\"\n" "d - Punto
flotante \"double\"\n" "D - Punto flotante \"long double\"\n" "t -
Cantidad de Bytes de cada tipo de dato\n" "T - Terminar\n" "--> "

```

Utilizando los macro reemplazos resultará:



UNLaM

Dto. Ingeniería e Investigaciones Tecnológicas

```
1 #include "main.h"
2
3
4 int main()
5 {
6     char opcion;
7
8     do
9     {
10        opcion = menuConErr(MENU_MSJ1, MENU_OPC1);
11        switch(opcion)
12        {
13            case 'f' : mostrarFF(); break;
14            case 'd' : mostrarFD(); break;
15            case 'D' : mostrarFL(); break;
16            case 't' : mostrarSizeofPtoFlotante(); break;
17        }
18    } while(opcion != 'T');
19    return 0;
20 }
21
22
```

Ejecutando nuevamente:

```
Select C:\Users\USER\Desktop\pruebasC\bin\Debug\pruebasC.exe
Elija una opcion:
f - Punto flotante "float"
d - Punto flotante "double"
D - Punto flotante "long double"
t - Cantidad de Bytes de cada tipo de dato
T - Terminar
--> S
ERROR - Opcion NO valida.
Elija una opcion:
f - Punto flotante "float"
d - Punto flotante "double"
D - Punto flotante "long double"
t - Cantidad de Bytes de cada tipo de dato
T - Terminar
--> f
Mostrando un "float".
Mostrando 5.7234567451234562222 con %f
5.723457
1-23456789012345678901234567890
Mostrando 5723.4567451234562222 con %f
5723.456860
1234-56789012345678901234567890
Mostrando 5.7234567451234562222 con %.10f
5.7234568596
```



UNLaM

Dto. Ingeniería e Investigaciones Tecnológicas

```
Select C:\Users\USER\Desktop\pruebasC\bin\Debug\pruebasC.exe
1-23456789012345678901234567890

Mostrando 123222123433222111333444.1112223330 con %f
123222124713355690000000.000000
123456789012345678901234-567890

Parece ser que los primeros siete digitos tienen precision?
Podremos afirmar que el octavo esta redondeado?
Saque sus conclusiones!

Elija una opcion:

f - Punto flotante "float"
d - Punto flotante "double"
D - Punto flotante "long double"
t - Cantidad de Bytes de cada tipo de dato
T - Terminar
--> D
Hay que hacer "algo" antes que esto ande
poner __USE_MINGW_ANSI_STDIO en [Project] / [Build optionsà] / [#defines]
Elija una opcion:

f - Punto flotante "float"
d - Punto flotante "double"
D - Punto flotante "long double"
t - Cantidad de Bytes de cada tipo de dato
T - Terminar
```

```
C:\Users\USER\Desktop\pruebasC\bin\Debug\pruebasC.exe
--> t
sizeof float      : 4
sizeof double     : 8
sizeof long double : 12
Elija una opcion:

f - Punto flotante "float"
d - Punto flotante "double"
D - Punto flotante "long double"
t - Cantidad de Bytes de cada tipo de dato
T - Terminar
--> T

Process returned 0 (0x0)   execution time : 766.899 s
Press any key to continue.
```

Continúa en: USO DE CADENAS DE CARACTERES.