***TP Nº4: Macros y Funciones***

***Ejercicio 1 / 2***

1. Al finalizar la ejecución:

***a = 1 b = 3***

**Rta data por ChatGPT**

| El valor final de la variable a será 1, ya que se incrementa en 1 después de que se usa en la expresión b = PI + a++.  El valor final de la variable b será 3.1415, ya que PI se define como 3.1415 y se suma a a++, que tiene un valor inicial de 0, por lo tanto, b será igual a 3.1415 + 0 = 3.1415.  Cabe mencionar que la variable b es de tipo entero (int), por lo que cualquier valor decimal se truncará. En este caso, 3.1415 se truncará a 3. |
| --- |

1. El programa no compila debido a que se pretende aplicar un operador de incremento a ***PI*** que es una constante. Esto es incorrecto ya que las constantes no son un l-value, es decir, no pueden estar del lado izquierdo de una asignación porque su valor no es modificable.

**Rta dada por ChatGPT**

| En el programa proporcionado, el valor final de la variable a es 0, ya que no se realiza ninguna operación en a aparte de asignarle el valor 0.  El valor final de la variable PI será 4.14, ya que PI++ incrementa el valor de PI en 1 después de usar su valor actual (3.14) en la expresión b = a + PI++.  El valor final de la variable b será 3.14, ya que a tiene un valor inicial de 0 y se suma con el valor actual de PI (antes del incremento), que es 3.14. Nótese que aunque se usa el operador de incremento, PI++, el resultado de la suma no se ve afectado por el incremento, ya que este ocurre después de la suma. Por lo tanto, b será igual a 3.14 + 0 = 3.14.  Cabe mencionar que las variables a y b son de tipo float, por lo que el valor final de b será un valor decimal. |
| --- |

1. La macro ***CUBO*** se debe escribir:

#define CUBO(x) ((x) \* (x) \* (x))

ya que sin los paréntesis, al realizar el preprocesamiento, la macroexpansión quedaría de la siguiente forma:



y sería evaluada con la siguiente precedencia:

a + ( 1 \* a ) + ( 1 \* a) + 1 es decir 4 + 4 + 4 + 1 = 13

que no es lo deseado.

La función ***cubo*** es correcta ya que al ser una función, los parámetros con los cuales es invocada son evaluados en tiempo de ejecución, y recibe a través del stack (o registros) el valor de los mismos:

cubo(a+1) es 5 \* 5 \* 5 = 125

1. En este ejercicio, el valor final de las variables a las que se les asignan las invocaciones a la macro CUBO (***b*** y ***c***) es IMPREDECIBLE. Esto se debe a que el lenguaje ANSI C no especifica el orden en que se evalúan los operadores de igual precedencia dentro de una expresión.

Por ejemplo, en algún compilador el resultado en la variable ***b*** puede llegar a ser 150. La forma en que llega a este valor es la siguiente :

Macroexpansión: b = ++a \* ++a \* ++a;

Inicialmente el valor de la variable ***a*** es 3. Al evaluar el primer término, se detecta que primero hay que hacer un incremento y entonces el valor de la variable ***a*** queda en 4. Luego se requiere evaluar la misma variable para el segundo término entonces nuevamente lo incrementa (antes de utilizar su valor), por lo que el valor de ***a*** es 5. Una vez que incrementó la variable ya la puede usar, por lo que la multiplicación es 5 \* 5 = 25. Luego se multiplica por el tercer término, lo incrementa (queda en 6) y lo multiplica por 25 ( 5 \* 5 \* 6 = 150 ).

En otros compiladores se podría obtener resultados distintos.

En el caso de invocaciones a funciones el parámetro se evalúa una única vez y su valor es enviado a través del stack a la función cubo. Los valores finales son:

***d = 4 e = 64 f = 27***

e) Primera invocación: Es correcta

| a = 20 | Mantiene su valor original |
| --- | --- |
| b = 5 | Mantiene su valor original |
| c = 4 | Tiene el resultado de la división |

Segunda invocación: No es correcta. Al usar como parámetros de una macro post/pre incrementos/decrementos no sabemos cuántas veces se utilizará el parámetro dentro de la macro.

| m = 6 | Aparece incrementado en 1, lo que resulta lógico porque en la invocación está postincrementado. |
| --- | --- |
| n = 2 | Aparece decrementado en 2. Este efecto no es previsible, ya que en la invocación se realiza un único postdecremento. |
| p = 1 | Tiene el resultado de la división |

Tercera invocación: Lo marcado en negrita son proposiciones inválidas ya que no se puede realizar una asignación sobre un valor constante y tampoco un postincremento. No son l-values.

{int i;

for (**0 = 0** , i = x –y ; i >= 0; **0++**, i - = y);

}

Corregido este error (reemplazando 0 por z en la invocación) los valores finales son los siguientes:

| x = 15 | Mantiene el valor original |
| --- | --- |
| y = 3 | Mantiene el valor original |
| z = 5 | Tiene el resultado de la división |

De este ejercicio surge como recomendación no usar operaciones de pre/post incremento/decremento como parámetros en la invocación de macros. Tampoco sería correcto usar cualquier otra operación que implique la alteración del contenido de una variable que aparezca más de una vez en la expresión.

***Ejercicio 5 (resuelto en videos)***

#include <stdio.h>

#include "getnum.h"

*/\* IMPORTANTE: Los paréntesis no están de más. a y b pueden ser expresiones \*/*

#define MAXIMO2(a,b,resp) (resp = ((a)>(b)) ? (a) : (b))

int

main(void)

{

int num1,num2,maximo;

num1 = getint("Ingrese primer numero:");

num2 = getint("Ingrese segundo numero:");

MAXIMO2(num1,num2,maximo);

printf("El mayor entre %d y %d es %d\n",num1,num2,maximo);

*/\* Otra forma de hacerlo \*/*

printf("El mayor entre %d y %d es %d\n", num1,num2,

MAXIMO2(num1,num2,maximo));

return 0;

}

Es importante notar que la macro MAXIMO2 está escrita de forma tal que puede formar parte de una expresión (ver segundo ejemplo).

***Ejercicio 6 (resuelto en videos)***

#include <stdio.h>

#include "getnum.h"

#define MAXIMO2(a,b,resp) (resp = ((a)>(b)) ? (a) : (b))

*/\* Esta macro se basa en la anterior, calculando el máximo de los dos*

*\*\* primeros, y luego comparando este máximo con el tercer número*

*\*/*

#define MAXIMO3(a,b,c,resp) (resp = (MAXIMO2(a,b,resp)>(c)) ? resp : (c))

int

main(void)

{

int num1,num2,num3,maximo;

num1= getint("Ingrese el primer numero: ");

num2= getint("Ingrese el segundo numero: ");

num3= getint("Ingrese el tercer numero: ");

MAXIMO3(num1,num2,num3,maximo);

printf("El mayor entre %d, %d y %d es %d\n",num1,num2,num3, maximo);

*/\* Otra forma de hacerlo \*/*

printf("El mayor entre %d, %d y %d es %d\n",num1,num2,num3,

MAXIMO3(num1,num2,num3,maximo));

return 0;

}

***Ejercicio 7***

Presentamos una primera solución con operador condicional.

#include <stdio.h>

#define isdigit(digit) ((digit>='0' && digit<='9') ? 1 : 0)

int

main(void)

{

int letra;

letra=getchar();

if ( isdigit(letra) )

printf("Es un dígito\n");

else

printf("No es un dígito\n”);

return 0;

}

Una segunda solución posible es:

#define isdigit(digit) (digit>='0' && digit<='9')

Presentamos una solución NO válida, que el alumno puede llegar a estar tentado de realizar. Queda como tarea para el alumno explicar por qué no funciona.

#define isdigit(digit) { if(digit>='0' && digit <='9') \

return 1; \

else \

return 0; \

}

***Ejercicio 8***

1. No se puede definir una función dentro de otra.
2. Falta el retorno de la función: **return respuesta;**
3. El punto y coma luego del encabezado de la función es incorrecto. Además falta el retorno de la función para la rama del else: **return n + 1**;
4. No se puede redefinir “a”. “a” es el nombre del parámetro de la función, por lo tanto no puede ser a la vez el nombre de una variable.
5. ***Letra****:* debe ir entre comillas dobles y ***\n***entre comillas simples.

Rtas dadas por ChatGPT

1. Hay un error en la sintaxis de las funciones en este código. El problema es que la función f2 está definida dentro de la función f1, lo cual no está permitido en C.

Una posible solución es definir la función f2 fuera de la función f1. De esta manera, ambas funciones estarán definidas en el mismo ámbito y se pueden llamar una desde la otra si es necesario.

1. La función suma en el código proporcionado no está devolviendo ningún valor a pesar de que su tipo de retorno es int. Este es un error, ya que la función debe devolver el valor de la suma de x e y.

Para corregir el error, simplemente agregue una instrucción de retorno para que la función devuelva el valor correcto

1. Hay un error de sintaxis en la función suma del código proporcionado. El problema es que el punto y coma (;) después de la lista de argumentos de la función indica el final de la declaración de la función, lo que significa que el bloque de código que sigue no forma parte de la función. Como resultado, el cuerpo de la función no se está ejecutando como se espera.

Para corregir el error, simplemente elimine el punto y coma después de la lista de argumentos y agregue una instrucción de retorno para que la función devuelva el valor correcto

1. Hay un error en la función f2 del código proporcionado. El problema es que se está redefiniendo la variable a en la declaración de la función, lo que genera un conflicto de nombres y produce un error de compilación.

Para corregir el error, simplemente elimine la declaración de float a en la definición de la función, ya que a ya está siendo pasado como un argumento.

1. Hay un error en la función f3 del código proporcionado. El problema es que se está utilizando comillas simples (') en lugar de comillas dobles (") en la cadena de formato de la función printf, lo que genera un error de compilación. Además, se está utilizando comillas dobles (") en lugar de comillas simples (') en el argumento de la función putchar, lo que también produce un error de compilación.

Para corregir el error, simplemente cambie las comillas simples a comillas dobles en la cadena de formato de printf, y cambie las comillas dobles a comillas simples en el argumento de putchar.

***Ejercicio 9***

No funciona correctamente. Por ejemplo: si el horario de entrada es a las 9:30hs, entonces:

**ENT\_HORA=9 y ENT\_MINUTOS=30**

Si un empleado llega a las 8:45, la función devuelve falso:

**(8<= ENT\_HORA && 45 <= ENT\_MINUTOS)**

La solución es la siguiente:

**((hora < ENT\_HORA) || (hora == ENT\_HORA && minutos <= ENT\_MINUTOS))**

Otra solución podría ser construir un número que tenga el "peso" de las horas y los minutos.

**(hora \* 100 + minutos <= ENT\_HORA \* 100 + ENT\_MINUTOS)**

***Ejercicio 10***

int

dcm ( int a, int b)

{

int auxi ;



**auxi = a;** 

while (auxi>0)

{

a = b;

b = auxi;

auxi = a % b ;

}

**return b;**

}

***Ejercicio 11***

Se presentan varias soluciones, algunas más claras que otras.

int

maximo (int a, int b, int c)

{

int m;

m = (a > b)? ((a > c)? a : c) : ((b > c)? b : c);

return m;

}

int

maximo (int a, int b, int c)

{

return (a > b)? ((a > c)? a : c) : ((b > c)? b : c);

}

int

maximo (int a, int b, int c)

{

int m;

m = ( a > b )? a : b;

return ( m > c )? m : c;

}

***Ejercicio 12***

Al ejecutar el programa del punto a) se obtiene el valor esperado (-2), al ejecutar el programa de punto b) se obtiene un valor erróneo. Esto se debe a que en ambos casos falta la inclusión, en el primer archivo, del prototipo de la función **neg** por lo que el compilador no conoce el tipo de datos que recibe y retorna la función. Al no conocer los tipos de datos involucrados el compilador siempre asume *por default* que todos los tipos desconocidos son **int**. Cuando se compila el archivo que contiene la función **main**, asume *por default* que la función **neg** recibe un entero y retorna un entero entonces en el caso a) por casualidad asume los tipos correctos y en el caso b) NO.

**NUNCA SE DEBE DEJAR LIBRADA LA INTERPRETACION DE TIPOS AL DEFAULT DEL COMPILADOR (Normas Indian Hill).**

***Ejercicio 13***

En el archivo tp4\_13a.c no se declararon las funciones **getfloat**, **f1** y **f2**, por lo que el compilador, al igual que en el ejercicio anterior, asume que los tipos de datos involucrados son de tipo entero, lo que provocará un error de ejecución.

Para solucionarlo se debe incluir en el archivo tp4\_13a.c el archivo de encabezado "getnum.h" por un lado y por otro definir un archivo "tp4\_13.h" con el prototipo de las funciones f1 y f2.

***Ejercicio 14***

No hay errores de preprocesamiento ni compilación. Sí habrá un error al linkeditar ambos archivos pues la función funAuxiliar está definida en más de una oportunidad, la solución para este error es eliminar una de las dos definiciones. Al igual que en el archivo anterior se produce un error de ejecución tanto al invocar la función getfloat como al invocar la función f2.

***Ejercicio 15***

Al preprocesar el archivo tp4\_15a.c se produce un error pues no existe el archivo tp4\_15.h.

Al compilar el archivo tp4\_15main.c se producirá un error pues la función fAuxiliar está declarada más de una vez y con distinto prototipo, lo cual provoca un error de "conflicto de tipos".

Al linkeditar los 3 archivos se producirá un error por la múltiple definición de la función fAuxiliar.

Los dos últimos errores están relacionados y derivan del hecho que la función fAuxiliar es una función auxiliar tanto para promedio3 como para mayor3, pero que si bien tienen el mismo nombre son dos funciones distintas. La solución es no incluir el prototipo en el archivo de encabezados y declararla como static, tanto en tp4\_15a.c como en tp4\_15b.c. Los códigos finales deben ser:

Archivo tp4\_15main.c

#include "getnum.h"

#include <stdio.h>

#include "tp4\_15a.h"

#include "tp4\_15b.h"

int

main(void)

{

int x,y,z;

x = getint("Ingrese un numero: ");

y = getint("\nIngrese otro numero: ");

z = getint("\nIngrese ultimo numero: ");

printf("\nEl mayor es %d\n", mayor3(x,y,z));

printf("El promedio es %.2g\n", promedio3(x,y,z));

return 0;

}

Archivo tp4\_15a.h

*/\* Biblioteca para obtener el mayor de 3 numeros \*/*

int mayor3 (int n, int m, int p);

Archivo tp4\_15b.h

*/\* Biblioteca para obtener el promedio de 3 enteros \*/*

float promedio3 (int n, int m, int p);

Archivo tp4\_15a.c

*/\* Biblioteca para obtener el mayor de 3 numeros \*/*

static int fAuxiliar (int m, int n);

int

mayor3 (int n, int m, int p)

{

return fAuxiliar( fAuxiliar(n,m), p);

}

static int

fAuxiliar (int m, int n)

{

int resp;

if ( m > n )

resp = m;

else

resp = n;

return resp;

}

Archivo tp4\_15b.c

*/\* Biblioteca para obtener el promedio de 3 enteros \*/*

static int fAuxiliar (int n, int m, int p);

float

promedio3 (int n, int m, int p)

{

return fAuxiliar(n,m,p) / 3.0;

}

static int

fAuxiliar (int n, int m, int p)

{

return n + m + p;

}

***Ejercicio 17***

#define ELAPSED(h1, m1, h2, m2) \

((h1) \* 60 + (m1) > (h2) \* 60 + (m2) ? \

((h1) - (h2)) \* 60 + (m1) - (m2) : \

((h2) - (h1)) \* 60 + (m2) - (m1))

En la siguiente guía se retomará la resolución de este ejercicio para poder mejorarla con el uso de una función de la biblioteca estándar.