***TP N°2: Expresiones en C - Entrada y Salida***

Esta guía cubre los contenidos vistos en las clases teóricas:

**3. Operadores y Expresiones**

**4. Entrada y salida de datos**

A partir de esta guía se listan ejercicios de la bibliografía básica de la materia que corresponden a los contenidos vistos.

Se recomienda la lectura del libro **El lenguaje de programación C** Segunda Edición de Brian W. Kernighan y Dennis M. Ritchie y la resolución de todos los ejercicios del libro sugeridos en las guías.

Se encuentra disponible en Campus ITBA un documento con el listado de los capítulos seleccionados del libro y su correspondencia con las clases teóricas.

El código fuente de la biblioteca **getnum.c**, junto con su header, se encuentra disponible en Campus ITBA y en el Drive.

Para leer por entrada estándar valores de tipo entero usar la función ***getint***.

Para leer valores numéricos reales, usar la función ***getfloat*** o ***getdouble***.

Al invocar cualquiera de estas funciones, cuando el usuario ingrese un número seguido de caracteres extras, el número será aceptado y el resto del buffer será consumido.

Hay una variante de la biblioteca, el archivo **getnum2.c,** que contiene las mismas funciones que **getnum.c**, pero en caso de recibir caracteres extras después del número (excepto espacios) el mismo es rechazado como error.

***Ejercicio 1***

Decidir si en los siguientes pares de códigos cada una de las variables termina con el mismo valor numérico independientemente de la plataforma. Indicar en las evaluaciones el tipo de aritmética utilizada. **Justificar todas las respuestas**

a) **int** letra, flag; **int** letra, flag;

flag = 0; flag = 0;

if ( letra == (**int**)’x’) if ( letra == ‘x’)

flag=1; flag=1;

terminan con el mismo valor. Ya que x es entero poner (int) x no cambia el valor. arit: entera.

1. **float** x; **float** x;

**int** y; **int** y;

y = 5; y = 5;

x = y / 3.0; x = (**float**) (y / 3);

terminan con distinto valor, ya que dec/dec= dec, int/int= trunca.

1. **float** x; **int** x;

**int** y; **int** y;

y = 5; y = 5;

x = y / 3.0; x = (**float**) y / 3;

terminan con distinto valor, ya que dec/dec= dec, int/int= trunca.

1. **int** x; **int** x;

**float** y; **float** y;

y = 5.5; y = 5.5;

x = y / 3; x = (**int**) y / 3.0;

en ambos casos da el mismo resultado.

1. **double** x; **double** x;

**int** y; **float** y;

x = 25; x = 25.0 ;

y = x / 5; y = x / 5;

en ambos casos da el mismo resultado.

1. **int** x, y; **int** x, y;

x = (**char**) 5; x = 5.0;

y = x++; y = x++;

en ambos casos da el mismo resultado. (por que no da 6?)

1. **int** i, j; **int** i, j;

**int** k; **unsigned** k;

i = j = 20000; i = j = 20000;

k = i + j; k = i + j;

terminan con distinto valor, ya que en uno ocurre desbordamiento de enteros.

1. **char** a, b; **char** a, b;

**char** c; **unsigned** **char** c;

a = b = 60; a = b = 60;

c = a + b; c = a + b;

en ambos casos da el mismo resultado.

1. **char** a, b; **char** a, b;

**char** c; **unsigned** **char** c;

a = b = 100; a = b = 100;

c = a + b; c = a + b;

terminan con distinto valor, ya que en uno ocurre desbordamiento de enteros.

j) **int** x, y; **int** x, y;

x = (**char**) 300; x = 300.0;

y = x++; y = x++;

terminan con distinto valor, ya que en uno ocurre desbordamiento de enteros.

k) **int** x, y, z; **int** x, y, z;

z = (x = 2) + (y = x) x = y = 2;

z = x + y;

en ambos casos da el mismo resultado.

(preguntar si sabe el valor de Z de antemano.)

Revisá el código izquierdo del punto k). ¿Es posible saber el valor de z de antemano? ¿Con qué concepto de la teórica **3. Operadores y Expresiones** lo relacionás?

***Ejercicio 2***

a) Escribir un programa que imprima el mensaje "**Este es un programa en C**" en la salida estándar, colocando cada palabra en una línea. (Usar un único printf)

b) Redireccionar la salida del programa hacia el archivo **output.txt**. Una vez terminada la ejecución del programa, editar dicho archivo y verificar que contenga la salida correcta.

#include <stdio.h>

int main (void)

{

printf(" este \n es \n un \n programa \n en \n c. \n ");

return 0;

}

1. ok. (./ej >> output2.txt)

# ***Ejercicio 3***

¿Es válida la siguiente sentencia ?

**printf("Hola que**

**tal \n");**

Está mal ya que el compilador no va a entender nada.

# ***Ejercicio 4***

Probar las siguientes instrucciones e investigar cuales son equivalentes entre sí. Sacar conclusiones.

**printf("\a\n");**

**printf("%c\n", 7); \a \n, %c\n son equivalentes.**

**printf("%d\n", 7); imprime el valor entero de 7**

**printf("%f\n", 7); te imprime el 0.00000**

**printf("%g\n", 7); %g %f son iguales los outputs. <te imprime el numero en otro sistema numerico.**

# ***Ejercicio 5***

Dada la siguiente declaración de variable:

**char c = 't';**

armar un programa que imprima en la salida estándar dicha variable, con los siguientes formatos: **%c, %d, %f.** Indicar qué valores imprime y de dónde obtiene dichos valores.

%c: te tira la letra de la tabla ascii.

%d te tira el valor de tabla ascii.

%f te lo define como un double y te vuelve 0. ¿?

#include <stdio.h>

int main (void)

{

char c = 't';

printf( "%f \n" , c);

return 0;

}

***Ejercicio 6***

Indicar qué salida se obtiene en cada ***printf*** del siguiente programa:

| | **#include <stdio.h> #define palabra "ultima prueba"  int main(void) {  int num1 = 53, num2 = 4;  float num3 = 6.874;   printf("num1= %10d\n", num1);  printf("num2= %010d\n", num2);  printf("num1= %-10d\n", num1);  printf("num1+num2= %5i\n", num1 + num2);  printf("num1+num2= %5f\n", num1 + num2);  printf("num3= %-3.2f\n", num3);  printf("num1= %-4d\nnum2= %-4d\nnum3= %3.1f\n", num1,num2, num3);  printf("num3(como entero)= %d\n", num3);  printf("num1 / num2 = %d\n", num1 / num2);  printf("num2 / num1 = %d\n", num2 / num1);  printf("esta es la %s\n", palabra);   return 0; }** | | --- |   **num1= 53**  **num2= 0000000004**  **num1= 53**  **num1+num2= 57**  **num1+num2= 0.000000**  **num3= 6.87**  **num1= 53**  **num2= 4**  **num3= 6.9**  **num3(como entero)= -825720160**  **num1 / num2 = 13**  **num2 / num1 = 0**  **esta es la ultima prueba** |
| --- | --- |

# ***Ejercicio 7***

Dada la siguiente definición de variables

**int edad = 25;**

**float longitud = 185.654;**

**char letra = ´Z´;**

imprimir en la salida estándar:

a) la variable ***edad***, encolumnada a izquierda, en un ancho de campo de 5 espacios.

b) la variable ***edad***, encolumnada a derecha, en un ancho de campo de 10 espacios.

c) la variable ***longitud***, encolumnada a la izquierda en un ancho de campo de 10 espacios, con 2 dígitos decimales.

d) la variable ***letra***, como número entero, en un ancho de campo de 8 espacios.

# ***Ejercicio 8***

Escribir un programa que lea una variable tipo ***float*** representando una velocidad en m/s, e imprima en la salida estándar el equivalente en km/h. **Utilizar una única variable.**

Para utilizar la función ***getfloat*** de la librería **getnum**, en el archivo fuente que estés utilizando para el ejercicio, necesitás incluir el archivo de encabezado correspondiente **#include “getnum.h”**

Recordá que al ser una librería de inclusión (y no una perteneciente a la biblioteca estándar), el *include* se realiza con comillas **(“**lib.h**”**) y no con menor y mayor (**<**lib.h**>**)

Para obtener un ejecutable que utilice funciones de la librería **getnum** necesitás enviar **getnum.c** como un archivo fuente más a la invocación habitual de *gcc*. Por ejemplo:

$> gcc tp2ej8.c getnum.c -o tp2ej8 -Wall -pedantic -std=c99

**#include <stdio.h>**

**#include "../biblio/getnum.h"**

**int main (void)**

**{**

**flota a;**

**a = getfloat("ingrese valor de la velocidad en m/s:") \* 3.6;**

**printf("%f\n" , a);**

**return 0;**

**}**

# ***Ejercicio 9***

Leer un número entero positivo y escribir en la salida estándar el valor de dicho número dividido 2, usando decalaje.

#include <stdio.h>

#include "../biblio/getnum.h"

int main(void)

{

char a, b;

a = getint("ingresar el valor de a:");

b = (a>>1);

printf("%d\n" , b);

return 0;

}

# ***Ejercicio 10***

Escribir un programa que lea una variable entera que representa un intervalo de tiempo expresado en segundos. Realizar la conversión para imprimir en la salida estándar el equivalente en horas, minutos y segundos. **No utilizar variables auxiliares.**

#include <stdio.h>

#include "../biblio/getnum.h"

int main (void)

{

int a;

a = getint("tiempo transcurridos enn segundos:");

printf("hora %d\n" , a/3600);

printf("minuto %d\n" , (a%3600)/60);

printf("segundos %d\n" , (a%3600)%60);

return 0;

}

# ***Ejercicio 11***

Leer dos caracteres desde la entrada estándar e imprimir en la salida estándar el de mayor valor ASCII, usando el operador condicional. Redireccionar la entrada de manera tal que ***getchar*** lea desde el archivo ***entrada.txt.*** Dicho archivo será escrito con un editor de texto sin formato y deberá tener el siguiente contenido:

a) **ABCD**

b) **A**

**B**

**C**

**D**

c) **A B** (usando tabulador entre A y B)

Indicar la salida generada en cada punto.

# ***Ejercicio 12***

Leer un caracter desde la entrada estándar e imprimir en la salida estándar si el mismo es una letra o no. Considerar que el conjunto de caracteres es el del lenguaje inglés, o sea ignorar la ñ y vocales acentuadas. Usar el operador condicional.

#include <stdio.h>

int main(void)

{

char a;

printf("ingrese caracter: ");

a = getchar();

if ((a >= 'a' && a <= 'z') || (a >= 'A' && a <= 'Z')) //si el carater esta en ese rango de valores, entonces es una letra.

{

printf("es una letra.");

}

else

{

printf("no es una letra."); // en caso de no estar en ese rango de valores no es una letra.

return 0;

}

}

# ***Ejercicio 13***

Escribir un programa que lea dos enteros, e imprima si el segundo es múltiplo del primero. Usar el operador condicional.

#include <stdio.h>

#include "../biblio/getnum.h"

int main (void)

{

int a, b;

a = getint("ingrese primer numero: ");

b = getint("ingrese segundo numero: ");

if (a%b == 0)

{

printf("el segundo es multiplo del primero. ");

}

else

{

printf("el segundo numero no es multiplo del primero. ");

}

return 0;

}

# ***Ejercicio 14***

Escribir un programa que lea dos enteros y a continuación imprima el promedio, la suma, el menor y el mayor de ellos (o indicar que son iguales). Usar el operador condicional.

*Ejemplo:*

Si tuviésemos **int a, b;**

y se leyeran los valores **5** y **8**, se debería imprimir:

**El promedio es 6.5**

**La suma es 13**

**El menor es 5**

**El mayor es 8**

**no son iguales**

**#include <stdio.h>**

**#include "../biblio/getnum.h"**

**int main(void)**

**{**

**int a, b;**

**a = getint("ingrese 1° numero: ");**

**b = getint("ingrese 2° numero: ");**

**printf("la suma es: %d\n" , a + b);**

**printf("el promedio es: %f\n" , (a + b)/(float)2);**

**if (a > b)**

**{**

**printf("el mayor es: %d\n" , a);**

**printf("el menor es: %d\n" , b);**

**}else**

**{**

**printf("el mayor es: %d\n" , b);**

**printf("el menor es: %d\n" , a);**

**}**

**if (a == b)**

**{**

**printf("los numeros son iguales.");**

**}else**

**{**

**printf("los numeros no son iguales.");**

**}**

**return 0;**

**}**

# ***Ejercicio 15***

Leer un carácter e imprimirlo en mayúscula. Considerar que el conjunto de caracteres es el del lenguaje inglés, o sea ignorar la ñ y vocales acentuadas. (Usar operador condicional).

#include <stdio.h>

int main(void)

{

char a, b;

a = getchar();

if(a >= 'a' && a < 'z')

{

b = a + ('A' - 'a');

printf("%c\n" , b);

}

else

{

b = a - ('A' - 'a');

printf("%c\n" , b);

}

return 0;

}

# ***Ejercicio 16***

Leer dos caracteres seguidos de la entrada estándar e imprimir en la salida estándar si son iguales, si el primero es mayor o si el primero es menor, utilizando operadores condicionales:

a) desde teclado, hacia pantalla

b) redireccionando la entrada desde archivo, hacia pantalla

c) desde teclado, redireccionando la salida a un archivo

d) redireccionando la entrada y la salida con archivos

*Ejemplo:*

Si se ingresaran los caracteres **'a'** y **'g'**

se debería imprimir **El caracter 'a' es menor al caracter 'g'**

**#include <stdio.h>**

**int main(void)**

**{**

**char a, b;**

**printf("ingrese caracteres: ");**

**a = getchar();**

**b = getchar();**

**if(a == b)**

**{**

**printf("son iguales.");**

**}**

**else if(a > b)**

**{**

**printf("el primero es mayor. ");**

**}**

**else**

**{**

**printf("el segundo es mayor. ");**

**}**

**}**

**return 0;**

**}**

# ***Ejercicio 17***

Indicar, para aquellas expresiones que compilen correctamente, el valor que devuelve.

**int  
 main(void)  
 {  
 "3" + "4";  
 '3' + '4';  
 3 + 4;  
 '3' + 4;  
 3 + '4';  
 "3" + '4';  
 "3" + 4;  
 return 0;  
 }**

| **"3" + "4"; No compila  '3' + '4'; Da ascii de ‘3’+’4’ (103 o ‘g’)  3 + 4; Da 7  '3' + 4; Da ‘7’  3 + '4'; Da ‘7’  "3" + '4'; Da algo raro  "3" + 4; No compila  return 0;** |
| --- |

# ***Ejercicio 18***

Indicar si las siguientes afirmaciones son verdaderas o falsas, justificar

1. La función printf siempre empieza a imprimir al comienzo de una línea nueva. V
2. El operador de módulo (%) puede ser usado sólo con operandos enteros. V
3. Las declaraciones de variable pueden aparecer en cualquier parte dentro de la función **main**. V
4. Un programa C que imprime tres líneas de texto debe contener tres sentencias con **printf F**

# ***Ejercicio 19***

Indicar cuál de los fragmentos de código cuenta con un mejor estilo de programación.

**int c1 = getchar();**

**int c2 = getchar();**

**c1 > c2 ? printf("%c es mayor\n", c1) : printf("%c es mayor\n", c2); Esta mal, usa dos printf**

**int c1 = getchar(); Para mi (keoni) este es el mejor**

**int c2 = getchar();**

**printf("%c es mayor\n", c1 > c2 ? c1 : c2);**

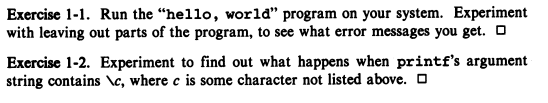
**int c1, c2;**

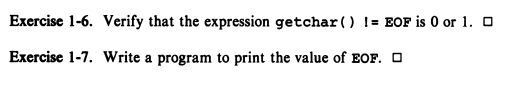
**printf("%c es mayor", (c1 = getchar()) > (c2 = getchar()) ? c1 : c2); Está mal, no se sabe cuál va a evaluar primero el compilador**

# ***Ejercicio 20***

Realizar los siguientes ejercicios del libro de texto básico:

**1.1**,  **1.2**, **1.6** y  **1.7**





# ***Ejercicio 21***

Analizar el programa tp2\_21.c

1. Sin ejecutarlo decir qué va a imprimir.
2. Generar el ejecutable y verificar que se corresponda con el punto anterior Si
3. Volver a generar el ejecutable pero agregando el flag Te avisa que hay overflow **-fsanitize=signed-integer-overflow**.

¿El resultado es el mismo que en el punto anterior? De no ser así, analizar por qué

# 

# ***Ejercicio 22***

El siguiente programa funciona correctamente en algunas computadoras, pero no en otras. Indicar cuál puede ser el problema (si bien aún no se vió en detalle, con la instrucción while se puede ejecutar un ciclo mientras la condición sea verdadera, en este caso queremos leer todos los caracteres de la entrada estándar hasta que no haya más caracteres por leer)

| #include <stdio.h>  **int**  main(**void**)  {  **char** c;  **while** ((c = getchar()) != EOF)   putchar(c);    **return** 0; } |
| --- |