TP N°1: Punteros

1) Realizar una función que reciba un número positivo entero por parámetro por referencia, y cambie su signo a negativo.

2) Realizar una función que reciba dos números enteros por parámetro por referencia, y que dentro de la función el usuario cargue sus valores.

3) Hacer la función “validar edad” que reciba por parámetro un puntero a la variable edad (un puntero a entero). La función debe permitir que el usuario del sistema ingrese una edad y luego verificar que la edad sea mayor a 0 y menor o igual a 18. Si la edad ingresada no respeta esos requisitos, se debe informar al usuario esta situación (que el valor ingresado es incorrecto y cuál es el requisito correcto) y se le debe volver a pedir el dato al usuario. Hasta tanto no lo ingrese correctamente, no debe salirse del **bucle**. Como resultado de la función y del uso del puntero, en la variable original en el Main debe quedar ingresado el dato de manera tal que cumpla con los requisitos.

4) Hacer la función “dividir” que reciba por parámetro dos números enteros: dividendo y divisor. La función la función debe “retornar” el resultado de la división y el resto de la misma, ambos como enteros (Ojo, solo puede haber un return, sólo puede retornarse un valor, pensar bien cómo lograr ese resultado).

5) Hacer la función “convertirTiempo”, que reciba un tiempo en segundos**.** La función debe “retornar” dicho tiempo convertido a minutos y convertido a horas, ambos resultados como enteros (Ojo, solo puede haber un return, sólo puede retornarse un valor, pensar bien cómo lograr ese resultado).

6) Analizar el siguiente código y responder que se mostrara por pantalla al ejecutarlo (responder sin escribirlo en la IDE. Luego, para verificar si lo pensado es correcto, escribirlo en la IDE y ejecutarlo para ver si coincide con la respuesta pensada).

```
void suma_dos (int *x, int *y, int *z)
{
*x=*x+2;
*y=*y+2;
*z=*z+2;
}
void main(void){
int x,y,z;
```

```
x=3;
y=10;
z=15;
suma_dos (&x, &y, &z);
printf("%d %d %d", x, y, z);
}
```

7) Analizar el siguiente código y responder que se mostrara por pantalla al ejecutarlo (responder sin escribirlo en la IDE. Luego, para verificar si lo pensado es correcto, escribirlo en la IDE y ejecutarlo para ver si coincide con la respuesta pensada).

```
void datos16 (int *x, float *y, char *c)
{
printf("%d %f %c", x, y, c);
*x=8;
*y=4.2;
*c='g';
}
void main(void){
int x=9;
float y=44.6;
char c='a';
datos16 (&x, &y, &c);
printf("%d %f %c", x, y, c);
}
```

8) Analizar el siguiente código y responder que se mostrara por pantalla al ejecutarlo (responder sin escribirlo en la IDE. Luego, para verificar si lo pensado es correcto, escribirlo en la IDE y ejecutarlo para ver si coincide con la respuesta pensada).

```
void datos17 (int x, float y, char c)
{
Printf ("%d %f %c", x, y, c);
x=8;
y=4.2;
c='g';
}
void main(void){
int x=9;
float y=44.6;
char c='a';
datos17 (x, y, c);
```

```
printf(“%d %f %c”, x, y, c);
}
```

9) Analizar el siguiente código y responder que se mostrara por pantalla al ejecutarlo (responder sin escribirlo en la IDE. Luego, para verificar si lo pensado es correcto, escribirlo en la IDE y ejecutarlo para ver si coincide con la respuesta pensada).

```
int datos18 (int x, float y, char c)
{
printf(“%d %f %c”, x, y, c);
x=8;
y=4.2;
c=’g’;
return x;
}
void main(void){
int x=9;
float y=44.6;
char c=’a’;
x=datos18 (x, y, c);
printf(“%d %f %c”, x, y, c);
}
```

10) Analizar el siguiente código y responder que se mostrara por pantalla al ejecutarlo (responder sin escribirlo en la IDE. Luego, para verificar si lo pensado es correcto, escribirlo en la IDE y ejecutarlo para ver si coincide con la respuesta pensada).

```
char datos19 (int *x, float *y, char *c) {
printf(“%d %f %c”,x, y, c);
*x=8;
*y=4.2;
*c=’g’;
return ‘h’ ;
}
void main(void) {
int x=9;
float y=44.6;
char c=’a’;
c= datos19 (&x, &y, &c);
printf(“%d %f %c”, x, y, c);
}
```