# Práctica 1

1. Comparación por tipo de datos

|  |  |  |  |
| --- | --- | --- | --- |
| **Pascal** | **C** | **Memoria** | **Cadena de formato** |
| char | char | 1 byte | %c |
| integer | int | 2 bytes | %d |
| longint | long int | 4 bytes | %d |
| word | unsigned int | 2 bytes | %u |
| real | float | 4 bytes | %f |
| double | double | 8 bytes | %f |

1. a) conio.h

b) stdio.h

1. a) char A;

A = 'a';

printf(" %c ", A); a

printf(" %d ", A); 97

A = 'a' + 10;

printf(" %c ", A); k

printf(" %d ", A); 107

b) int B;

B = 5;

printf(" %d ", B); 5

printf(" %c ", B); ♣

B = 5 + 'A';

printf(" %d ", B); 70

printf(" %c ", B); F

c) #define num 15

printf(" %d ", num); 15

printf(" %c ", num); Carácter de control -> no muestra nada

d) #define num 15

num = 5 + 'A'; <- Error: no compila porque no se puede modificar constantes

printf(" %d ", num);

printf(" %c ", num);

1. a) z < 5 && ( a < = 5 || c > 0 )  
   b) z = CHR( 0 ) || !( total == 0 )
2. a = 1; b = 2; c = 3; d = 4
   1. a + b \* c / d = 1 + 2\*3 /4 = 1 + 6 div 4 = 1+1 = 2
   2. (a + b) \* c % d = (1+2) \* 3 % 4 = 3\*3 % 4 = 9 % 4 = 1
   3. d < 65 && 3 == d = 1 && 0 = 0
   4. a <= b || (3 > a) = 1 || 1 = 1
   5. (a >= d) && (2 - 14 == c + 1) = 0 \* 0 = 0
   6. a>b || a<c && a = c || a >= b Error porque al lado izquierdo de la asignación no hay una variable.
3. a) int a;

/\* leer a = 3 \*/

scanf("%d", &a); 3

scanf("%f", &a); 0.0000000

scanf("%c", &a); El carácter 3 o el entero 51

b) char x;

/\* leer x = 3 \*/

scanf("%d", &x); Carácter ETX (fin de texto)

scanf("%f", &x); Carácter nulo

scanf("%c", &x); 3

c) char w;

/\* leer w = 2.5 \*/

scanf("%d", &w); ☻

scanf("%f", &w); Carácter nulo

scanf("%c", &w); 2

10. a) c = (a<b)? a: b

b) printf((a < b && b < c)? “ordenados”: “desordenados”);

19. a) No, porque los arreglos no se pasan por valor sino que siempre se pasan por referencia, sin importar si son de entrada y/o salida.

b) Sí, el valor obtenido representa la dirección de memoria RAM en la que se encuentra el arreglo.

22. a) No es correcta la sentencia scanf(“%s”, &cad) porque cad es un vector de caracteres y un arreglo ya se interpreta internamente como un puntero porque es más fácil pasarlo por referencia que pasar cada uno de sus valores. Por lo tanto, no haría falta un &, sino que con un scanf(“%s”,cad) está más que bien.

b) Es más conveniente gets porque está hecho para leer cadenas de texto. De hecho, no hace falta poner “%s” con gets por lo que es más intuitivo. De todas formas, con scanf también se podría leer cadenas de caracteres pero habría que especificar qué tipo de dato queremos leer.

c) strcmp es una función que, en base a 2 cadenas de caracteres, devuelve un número producto de su comparación según el orden alfabético. En el ejemplo strcmp(cad1,cad2):

* Devuelve un número negativo si cad1 es menor a cad2
* Devuelve un 0 si cad1 es igual a cad2
* Devuelve un número positivo si cad1 es mayor a cad2

# Práctica 2

1. int \*p, a = 4; b = 5;

p = &b; <- A p se le asigna donde esta b. \*p pasa a valer 5

\*p \*= 2; <- Duplicar a \*p (pasa a valer 10)

printf("b=%d \*p=%d\n", b, \*p); b=10 \*p=10

printf("&b=%p p=%p &p=%p\n", &b, p, &p); &b=AA01 p=AA01 &p=B3F0

b = \*p \*3;

printf("b=%d \*p=%d\n", b, \*p); b=30 \*p=30

printf("&b=%p p=%p\n", &b, p); &b=AA01 p=AA01

a = b;

p = &a;

(\*p)++;

printf("b=%d a=%d \*p=%d\n", b, a, \*p); b=30 a=31 \*p=31

printf("&b=%p &a=%p p=%p &p=%p\n", &b, &a, p, &p); &b=AA01 &a=AA02 p=AA02 &p=B3F0