

TRABAJO PRÁCTICO N° 9

Deben entregarse por grupos en la entrega correspondiente vía Campus los archivos .c correspondientes a los ejercicios indicados.

1. Explicar por qué el siguiente código no compila y corregirlo.

```
#include <stdio.h>

int main(void)
{
    char a = 'h';
    void *p;
    p = &a;
    printf("a: %c\n", *p);
    return 0;
}
```

2. ¿Qué sucede si apuntamos a un puntero con un cast incorrecto? ¿Qué ocurre cuando se lee y escribe a través de ese puntero? Explicar el funcionamiento del siguiente código:

```
#include <stdio.h>

int main(void)
{
    int a = 257;
    void *p;

    printf("a: %d\n", a);
    p = &a;
    *((char *) p) = 5;
    printf("a: %d\n", a);

    return 0;
}
```

3. Explicar por qué el siguiente código no compila. Corregir el código. Detallar en una tabla los valores de las variables a medida que ejecuta el programa.

```
#include <stdio.h>

int main(void)
{
    int a = 5;
    void *p = &a;
    void **pp = &p;

    printf("pp: %p\n", pp);
    printf("*pp: %p\n", *pp);
    printf("**pp: %d", **pp);

    return 0;
}
```

4. La librería `<stdlib.h>` incluye una función `qsort()`¹ que permite aplicar el algoritmo Quicksort para ordenar un arreglo. El usuario puede ordenar cualquier tipo de elementos y debe proveer un puntero a una función que realiza la comparación entre dos elementos, utilizando el concepto de callback.

Utilizar la función `qsort()` para ordenar un arreglo de números enteros.

5. Utilizar la función `qsort()` para ordenar un arreglo de strings en orden alfabético.
6. Crear un arreglo de alumnos, representados por una estructura que almacene nombre, apellido y nota promedio en tres campos distintos. Luego ordenar el arreglo usando la función `qsort()`, de manera que queden ordenados por nota en orden decreciente, si comparten la misma nota por apellido en orden alfabético, y si tienen la misma nota y el mismo apellido ordenados por nombre en orden alfabético.

¹ <http://www.cplusplus.com/reference/cstdlib/qsort/>
Trabajo Práctico N° 9

7. **[ENTREGAR]** Implementar una calculadora simple, a la que se le puedan ir agregando nuevas funciones.

Se generarán dos arreglos (globales): uno contendrá operandos, y el otro punteros a la función que debe realizar la operación correspondiente, que se completarán de manera dinámica al inicializarse el programa.

Se le solicitará al usuario que ingrese la cuenta que desea realizar, de la forma *xxxx operador yyyy* (ejemplo: 5 + 3)

Finalmente deberá mostrar el resultado obtenido.

La calculadora debe implementar como mínimo: suma, resta, multiplicación, división y potenciación (con exponente entero).

```
int add_operation(char o, float (*a) (float, float));
float calc_res(float x, float y, char op);

...

/* Estos arreglos comienzan vacíos (sin inicializar), pero
luego de varias llamadas a add_operation() tendrían la
siguiente forma: */
char operators[MAX_OPERATORS] = {'+', '-', '*'};
float (* actions [MAX_OPERATORS] ) (float, float) = {add,
sub, mult};

// Ejemplo de uso
int main(void)
{
    add_operation('+', suma);
    ...

    float x, y, op;

    get_input(&x, &y, &op);
    printf("% f", calc_res(x, y, op));

    return 0;
}

float suma (float a, float b)
{
    ...
}
```

8. Implementar un módulo con una función, que dada una entrada *en formato de entrada de main por línea de comandos* distinga entre opciones y parámetros y devuelva esta información en tres arreglos a quien la llamó.
- Opción: una opción consiste de dos argumentos: el primero comienza con un guión ("-") y se denomina clave, y el segundo se denomina valor. Un ejemplo con dos opciones (la clave maxclients con valor 4, y la clave path con valor /var/web) es: `webserver -maxclients 4 -path /var/web`
 - Parámetros: un argumento aislado que no comienza con guión. Un ejemplo con una opción y dos parámetros es: `copyfile -isverify 1 archivo1.c archivo2.c`
 - Escribir un *main.c* que utilice la función que escribieron para imprimir en pantalla las opciones y parámetros.

Ejemplo de uso:

```
$ ./programa -graficar si -iteraciones 4 entrada.txt salida.txt

Opcion 1: Clave: graficar Valor: si
Opcion 2: Clave: iteraciones Valor: 4
Parametro 1: entrada.txt
Parametro 2: salida.txt
```