



Práctica 0

1. Escriba un programa que declare algunas variables locales, e imprima las direcciones de memoria de las mismas. Pruebe declarar un arreglo de caracteres, y verifique las direcciones de sus elementos son contiguas.
2. Implemente la función `char *itoa(int)`, que dado un entero devuelve su representación como string.
3. Definir una estructura `carta` para representar una carta de la baraja española (represente el palo con una enumeración). Cree un arreglo de 48 cartas, llámelo mazo y llénelo con las cartas correspondientes.
4. Implemente una función `struct carta azar(struct carta[], int)` que reciba un mazo, la longitud del mismo, y devuelva una carta al azar del mazo pasado.
5. Explicar la diferencia entre:
 - a) `int (*f)[3];`
 - b) `int *f[3];`
 - c) `int *(f[3]);`
 - d) `int *f();`
 - e) `int (*f)();`
6. Implemente una función `void setzerozero(int [])` que ponga en cero el primer elemento del arreglo recibido. Verifique desde la función llamante que efectivamente modifica este valor. ¿Por qué pasa esto? ¿No llama a la función por valor?
7. Explique el tipo de la función `malloc`, ¿qué valor retorna la función en caso de que no pueda reservar el espacio solicitado?
8. Implemente una estructura `contacto` que tenga como campos: una cadena para el nombre de una persona, una cadena para el número de teléfono, y un entero sin signo para llevar la edad de la persona.
9. Implemente una función `struct contacto crearcontacto(void)` que pida por teclado los datos pertinentes, rellene una estructura `contacto` y la devuelva.
10. Implemente una estructura para representar puntos en el plano, y una función `medio` que dados dos de estos puntos, calcule el punto medio.
11. Implemente una función `setin(int *)` que toma un puntero a un entero, y reemplaza el entero apuntado por un 1 si el entero apuntado era diferente a 0, y 0 en caso contrario.
12. Implemente una función `void swap(int *, int *)` que dados dos punteros a variables, intercambie el contenido de las variables apuntadas.
13. Implemente una versión de la función `swap` que no utilice una variable temporal para hacer el intercambio de valores (leer propiedades del operador xor). ¿Qué ventaja puede traer esto?
14. Implemente una función `char *getline(void)` que ingrese una línea por teclado (hasta `\n`), y devuelva un puntero a la cadena ingresada.
15. Implemente una función `void actualizaredad(struct contacto *)` que dado un puntero a una estructura `contacto`, pida una nueva edad por teclado y actualice la estructura.

16. Implemente una función `int prom(struct contacto *, int)` que reciba un puntero a estructuras `contacto`, y la cantidad de estructuras contiguas, y devuelva el promedio de la edad de estas estructuras.

¿Qué diferencia existe con tomar un arreglo de estructuras en lugar de un puntero?

17. Defina una estructura `agenda` que almacene un arreglo de estructuras `contacto`, y un entero para llevar la cantidad de estructuras completadas del arreglo.

Escriba un programa que permita: dar de alta un contacto, modificar la edad de un contacto, y ver los datos de los contactos cargados.

18. Escriba un programa que reserve un espacio de memoria de 100 bytes, y luego libérela dos veces. ¿Se produce algún error?

19. Implemente una función `int apply(int (*)(int), int)` que toma un puntero a función, y un entero, y aplica la función al entero y retorna el valor dado.

20. Implemente una función `void applyin(int (*)(int), int *)` que toma un puntero a función, un puntero a un entero, y reemplaza el entero apuntado por el valor de ejecutar la función apuntada sobre el valor apuntado.

21. Implemente una función `void recorre(VisitorFunc, int [], int)`¹ que toma un puntero a una función, un arreglo de enteros, y su longitud, y aplica la función a cada elemento del arreglo. Utilice esta función para re-escribir la función `printints` de la práctica anterior.

22. Otra signatura válida para `main` es `int main(int argc, char *argv[])`. El argumento `argc` cuenta la cantidad de argumentos pasados, y `argv` es un arreglo de cadenas que representan cada uno de los argumentos pasados al programa.

Escribir un programa que imprime en pantalla la cantidad de argumentos que recibió, y el contenido de los mismos.

¹`VisitorFunc` está definido por `typedef void (*VisitorFunc)(int)`