


| | | |
|--|--|---------------------------|
|  | Facultad de Ingeniería y Ciencias Agrarias | PROGRAMACIÓN ESTRUCTURADA |
| | | Practica N° 03 |
| | Punteros, Memoria Dinámica y Estructuras | |

Punteros

En cada ejercicio se deberá desarrollar funciones y luego implementarlas en programas donde deberán realizar las pruebas necesarias para verificar su correcto funcionamiento.

Ej. 1: Desarrollar una función que reciba como parámetros dos punteros a int y qué intercambie los valores de las variables apuntadas por dichos punteros.

Ej. 2: Desarrollar una función que reciba como parámetro un arreglo de int y una posición de dicho arreglo. Luego la función deberá retornar la dirección de memoria de la variable que se encuentra en esa posición del arreglo.

Aclaración: La función se resuelve en una sólo línea de código.

Ej. 3: Desarrollar una función que redondee al entero más próximo el valor contenido en una variable externa de tipo double.

Aclaración: La función no deberá retornar valores.

Ej. 4: Dada la estructura:


```
struct s_carta
{
    int numero, valor;
    char palo;
}
typedef struct s_carta t_carta;
```

Desarrollar una función que modifique el contenido de 3 variables externas de este tipo, de manera que las cartas almacenadas en éstas queden ordenadas según su valor en el juego del “Truco”.

Ej. 5: Dada la estructura:

```
typedef struct{
    int a, b, c;
}t_tres;
```

Desarrollar una función que modifique el contenido de una variable externa de este tipo, de manera que los valores contenidos en cada campo queden ordenados. Es decir, el

| | | |
|---|---|------------------------------|
|  PONTIFICIA UNIVERSIDAD CATÓLICA ARGENTINA | Facultad de Ingeniería y Ciencias Agrarias | PROGRAMACIÓN ESTRUCTURADA |
| | | Practica N° 03 |
| | Punteros, Memoria Dinámica y Estructuras | |

campo “a” debe tener el valor más bajo, el campo “b” el segundo más bajo y así sucesivamente.

Ej. 6: Dada la estructura:

```
typedef struct
{
    int n, o , p;
    char p1[26], p2[50];
    double a, b, c;
}
t_varios;
```

Desarrollar una función que intercambie los valores de 2 variables de este tipo, las cuales son externas a la función.

Aclaración: La función podría realizarse en con cinco líneas de código.

ANEXO: Calcular manualmente el tamaño “en byte” de la estructura. Luego pruebe en su programa imprimir el tamaño de la estructura utilizando la función **sizeof()**

Memoria Dinámica

Ej. 7: Desarrollar la función **cargarStrDin()** que permita ingresar una frase desde el teclado hasta presionar ENTER. Y que lo ingresado lo almacene en un arreglo en memoria dinámica y sin desperdiciar espacio. Luego la función deberá remitir al exterior el contenido cargado. Realizar una prueba desde el programa principal imprimiendo en consola el contenido remitido por la función.

Ej. 8: Desarrollar dos versiones de la función del ejercicio anterior **cargarStrDin()** según el siguiente detalles:

Versiones de cargarStrDin():

1. Esta versión **char* cargarStrDin1()**, no se le pasan parámetros y retorna la dirección de memoria del arreglo dinámico creado.
2. Esta versión **void cargarStrDin2(char**)**, no retorna ningún valor, y se le pasa por parámetro la dirección de memoria de un puntero a char.

Desarrollar la función **escribirArch()** que recibe por parámetro el nombre del archivo "frase.txt" y un string. La función deberá escribir en el archivo de nombre pasado por parámetro, el contenido del string también pasado por parámetro.

Además desarrollar la función **imprimirArch()** que recibe por parámetro el nombre del archivo "frase.txt" e imprime su contenido. (*ver prototipos*)

Escribir el programa principal como se detalla a continuación:

```
int main(){
    char* str=NULL;
    str = cargarStrDin1();
    // cargarStrDin2(& str);
    escribirArch("frase.txt", str);
    imprimirArch("frase.txt");
    return 0;
}
```

Al realizar pruebas, ejecutar con **str = cargarStrDin1();** ó con **cargarStrDin2(&str);**, el programa debe funcionar de la misma forma independientemente de con qué función se trabaje.

Prototipos:

```
char* cargarStrDin1()
void cargarStrDin2(char** pCadena)
void escribirArch(const char * nomArch, char* cadena)
void imprimirArch(const char * nomArch)
```

Ej. 9: Desarrollar la función **subcadena()** cuyo prototipo es:

```
char * subcadena (char * p, unsigned int i, unsigned int n);
```

La función **subcadena()** deberá retornar un puntero a un nuevo espacio de memoria dinámica conteniendo el fragmento del string **p** que va desde el carácter posicionado en **i**, tomando desde **i** los **n** caracteres siguientes.

Desarrollar la función **leerArch()** que recibe por parámetro el nombre del archivo "frase.txt" y retorne el puntero de un arreglo dinámico con los datos completo del archivo leído.

```
char* leerArch(const char * nomArch)
```

Escribir el programa principal como se detalla a continuación:

```
int main(){
    char* str=NULL;
    char* subStr=NULL;
    int i=8, n=5;
    str = leerArch("frase.txt");
    subStr = subcadena (str, i, n);

    printf("Para i = %d y n = %d ,Se encontro el substring: ", i,n);
    printf("%s", subStr);

    return 0;
}
```

Ejemplo:

Si el contenido del archivo **frase.txt** es:

Universidad Católica Argentina


Para i=8, n=5

la salida en consola es:

Para i = 8 y n = 5, Se encontro el substring: **dad C**

Algunas consideraciones para realizar una función **subcadena()** con una funcionalidad completa:

- Considerar realizar pruebas con los valores de *i* y/o *n* fuera de rango del tamaño del string y que la función se comporte correctamente.
- Si el *i* está fuera de rango,(o si *n* = 0 para cualquier *i*), la función deberá retornar un string vacío. Un string vacío es un arreglo de char donde el elemento de la posición cero es un '\0'.
- Si *i* dentro de rango y *n* fuera, la función deberá retornar el substring correspondiente hasta el límite del string original.
- Si *n* es negativo, representa desplazamientos a izquierda del valor de posicionamiento *i*.

| | | |
|---|---|--------------------------------------|
|  UCA <hr/> PONTIFICIA UNIVERSIDAD CATÓLICA ARGENTINA | Facultad de Ingeniería y Ciencias Agrarias | PROGRAMACIÓN ESTRUCTURADA |
| | | Practica N° 03 |
| | <i>Punteros, Memoria Dinámica y Estructuras</i> | |

Ej. 10: Dada la estructura:

```
struct s_texto
{
    char * txt;
    unsigned int longitud;
};
typedef struct s_texto t_texto;
```

Desarrollar una función **cargarTexto()** que permita ingresar un texto de longitud indefinida y guardarlo en una variable externa del tipo de dato **t_texto**, completando ambos campos. Se recomienda en esta función llamar a la función del ejercicio 8.2 pero modificarla para que retorne el tamaño del string cargado.

```
t_texto cargarTexto()
```

Luego, desarrollar la función **escribirArchTex()** a la cual se le pasa por parámetro el nombre del archivo "frases_con_longitud.csv" y el puntero a una estructura **t_texto***. La función deberá escribir en el archivo la longitud del texto y el texto, separando ambos elementos por una coma; como se muestra en el siguiente ejemplo:

Ejemplo de archivo de salida (*frases_con_Longitud.csv*)

```
83,No guardes nunca en la cabeza aquello que te quepa en un
bolsillo.(Albert Einstein)
```

```
void escribirArchTex(const char * nomArch, t_texto *)
```

Escribir el programa principal como se detalla a continuación:

```
int main(){
    escribirArchTex("frases_con_longitud.csv", cargarTexto());
    return 0;
}
```

Ej. 11: Dada la estructura:

```
typedef struct{
    int a, b;
}t_dosint;
```

| | | |
|--|---|--------------------------------------|
|  | Facultad de Ingeniería y Ciencias Agrarias | PROGRAMACIÓN ESTRUCTURADA |
| | | Practica N° 03 |
| | <i>Punteros, Memoria Dinámica y Estructuras</i> | |

Programar una función que intercambie los valores de los campos a y b de una variable de este tipo que es externa a la presente función. Dentro de esta función, utilizar la función programada en el **ejercicio 3.1** (sin modificarla).

Ej. 12: Dada la estructura:

```
struct s_texto
{
    char * txt;
    unsigned int longitud;
};
typedef struct s_texto t_texto;
```

Desarrollar las siguientes funciones:

- Una función que me permita ingresar una cantidad indefinida de frases diferentes, las cuales se irán guardando en un arreglo dinámico de variables del tipo t_texto. Para cargar cada frase, utilizar la función del **ejercicio 3.10**, sin modificarla. Utilizar un NULL en el campo txt como marca de finalización. La carga de frases finaliza cuando se ingresa un texto de tamaño cero.
- Una función que reciba como parámetro un arreglo dinámico del tipo t_texto (el cual finaliza con la variable cuyo campo txt contiene NULL) y que lo ordene alfabéticamente.
- Una función que imprima una variable de tipo t_texto por pantalla (*solo el texto*).
- Una función que reciba como parámetro un arreglo dinámico de variables t_texto (el cual finaliza con la variable cuyo campo txt contiene NULL) y que agregue la información al archivo "frases_con_longitud.csv" creado en el ejercicio 3.10

Implementar todas estas funciones en un programa donde se ingresen frases por teclado. Luego ordenar alfabéticamente dicho listado y lo guarde en el archivo "frases_con_longitud.csv".