UNL FICH Facultad de Ingeniería y Ciencias Hídricas

Universidad Nacional del Litoral
**Facultad de Ingeniería y Ciencias Hídricas**
Departamento de Informática


Ingeniería Informática

---

# PROGRAMACIÓN ORIENTADA A OBJETOS

UNIDAD 01

## Punteros

Guía de trabajos prácticos

2023

# UNIDAD 01
# **Punteros**

---

**Ejercicio 1**

Declare una variable entera estática y otra variable entera dinámica almacenando su dirección de memoria en un puntero. Asigne a la primera variable el valor 10 y a la segunda variable un valor ingresado por el usuario. Muestre luego las posiciones de memoria de ambas variables. IMPORTANTE: No olvide liberar la memoria de la segunda variable antes de salir.

**Ejercicio 2**

Utilizando notación de punteros, generar un arreglo dinámico y lineal de N elementos numéricos enteros, con valores aleatorios entre 1000 y 1500, y muestre en pantalla la dirección de memoria del mayor elemento. N es un dato ingresado por el usuario.

**Ejercicio 3**

Escriba una función que utilice punteros para buscar y retornar la dirección de un entero dentro de un arreglo. Se pasan como parámetros el arreglo, su tamaño y el entero a buscar. Si el dato no se encuentra, debe retornar la dirección de memoria nula (*nullptr*). ¿Desde un programa cliente, cómo se obtiene el índice del elemento encontrado cuando la función no devuelve *nullptr*?

**Ejercicio 4**

Se tiene un arreglo dinámico de `n` enteros, y se quiere aumentar su tamaño a un nuevo tamaño `c`. Implemente una función *redimensionar()* que reserve dinámicamente memoria para un nuevo arreglo que contenga lugar suficiente para guardar `c` datos (los `n` datos anteriores y los `c-n` nuevos). Copie en la nueva memoria los datos del vector viejo e inicialice con ceros los nuevos elementos. La memoria del primer arreglo debe ser liberada y el struct actualizado para que el programa cliente pueda seguir operando con el nuevo arreglo. La función debería poder invocarse de la siguiente manera:

```
vector_int_dinamico arreglo; (Teoría Pág. 10)
...
redimensionar( arreglo, c );
```

Finalmente, implemente un programa cliente que muestre el arreglo resultante.

**Ejercicio 5**

Escriba un programa que genere un arreglo aleatoriamente, luego permita al usuario ingresar un valor M y una posición P, y finalmente inserte el valor M en la posición P del arreglo. Muestre el arreglo modificado. Nota: Para la inserción, implemente una función *insertar()* utilizando notación de punteros.

**Ejercicio 6**

Analice el código C++ del recuadro de abajo para responder lo siguiente:

```
void func(int *s, int *e, int *sum) {
    (*sum)=0;
    for(int *p=s;p<e;p++)
        (*sum)+=*p;
}

int main() {
    int x[6]={12,34,56,78,39,90};
    int suma;
    func(x, x+6, &suma);
    cout<<suma<<endl;
}
```

a. ¿Qué tipo de parámetros actuales se emplean para llamar a *func()*?.
b. ¿Qué tipos de parámetros formales se definen en *func()*?.
c. ¿Qué tipo de información devuelve la función *func()*?.
d. ¿Cuál es la salida que se obtiene en el programa propuesto?

**Ejercicio 7**

Observe la porción de código C++ del recuadro y determine la salida que se obtiene de los flujos de salida cout propuestos. Considere que las variables se han almacenado en memoria a partir de la dirección 0x000FF09 en forma contigua en el orden en que fueron declaradas. Responda: ¿sería posible revisar si su respuesta fue correcta compilando y ejecutando este código? Justifique su respuesta.

```
.....
int a=90;
int *p=&a;
int b=(*p)++;
int *q=p+2;
cout<<p<<" "<<*p<<" ";
cout<<q<<" "<<*q<<" ";
a=*(++q); b=*(++p);
(*p)++;
cout<<p<<" "<<&p<<" ";
cout<<q<<" "<<&q;
.....
```

# Cuestionario

1. ¿Qué es un puntero? ¿Cómo se declara en C++?

2. Indique tres formas de emplear el * en C++.

3. ¿Cuál es la diferencia entre las dos acciones siguientes que emplean el &? ¿De qué tipos son b y p?

```
char &a=b;
p=&b;
```

4. Explique el significado de las siguientes expresiones de código C++.

a) `int a = 25;` b) `int &ra = a;`
c) `int *pa = &a;` d) `int *pb = new int;`
g) `int *pd = nullptr;` h) `int *pe, pf;`

5. ¿Qué diferencias hay entre estas dos formas de obtener memoria para un arreglo?

a) `int *pc = new int[10];` b) `int v[10];`

6. ¿Qué operaciones pueden realizarse con punteros? Ejemplifique.

7. Si p es un puntero a enteros y q un puntero a datos de tipo double, cuál es la diferencia en bytes entre:

a) p y p+4 b) p y p-1 c) q y q+5 d) q y q++

8. Proponga 2 maneras de acceder a la dirección del sexto elemento de un arreglo x. Proponga 3 maneras de acceder al elemento de la tercera fila, quinta columna de una matriz m.

9. Si una cantidad n es sumada a una variable puntero, ¿cuál es el resultado de dicho cálculo? ¿Y si se restan del puntero?.

10. Si se declara un arreglo a de 10 elementos enteros: int x[10]={110, 120, 130, 140, 150, 160, 170, 180, 190, 200}; y el elemento inicial x[0] se ubica en la dirección de memoria 0x00011E4. Determine lo que que representan las siguientes expresiones:

a) `x;` c) `(x+4);` e) `*x+3;`
b) `*x;` d) `*(x+4);` f) `(*x)+3;`

11. De acuerdo con la siguiente declaración de las variables dato y p:

```
int dato=25;
int *const p=&dato;
```

Determine si son válidas las siguientes proposiciones. Justifique en caso de responder negativamente

a) `dato=100;` b) `p=&(dato+1);`
c) `p++;` d) `p=nullptr;`

12. Explique la diferencia entre las dos declaraciones siguientes:

```
a) float  func();       b) float (*func)();
```

13. Explique qué reciben y qué retornan cada una de estas funciones:

```
a) int fa(int *v, int n);
b) int fb(int *pi, int *pf);
c) int *fc(int a, int b, int c);
```

    Piense en un ejemplo concreto donde podría ser útil cada caso.

14. ¿Cuál es la ventaja de emplear variables dinámicas ? ¿Cómo las define en C++?

15. ¿Cuál es la diferencia entre crear dinámicamente una matriz con un puntero a un arreglo y con un arreglo de punteros?

# Ejercicios Adicionales

**Ejercicio 1**

Explique el significado de las siguientes expresiones de código C++.

a) `double *v1[30];` b) `double (*v2)[20];`

c) `float (*pf)(int a, int b);`

¿En qué casos utilizaría cada una?

**Ejercicio 2**

Usando la sintaxis de C++ escriba el código para:

a. Declarar un puntero a datos de tipo flotante y otro puntero a datos de doble precisión.
b. Declarar un puntero a una lista de 10 arreglos contiguos de 15 elementos enteros cada uno. Reserve, además, la memoria necesaria.
c. Declarar un arreglo de punteros para representar una matriz de 10x30 elementos flotantes.
d. Declarar un puntero a una función que acepte 2 parámetros de doble precisión y no devuelva resultado alguno.
e. Reservar memoria para un entero y un arreglo de 10 datos tipo float.
f. Liberar la memoria reservada en el apartado anterior.

**Ejercicio 3**

Escriba una función que reciba un arreglo de enteros (y su tamaño), y busque elementos repetidos dentro del mismo. La función debe retornar un struct con dos punteros apuntando a dos elementos del arreglo que tengan el mismo valor. Si hay más de dos elementos repetidos, la función puede retornar cualquier par de ellos. Si no hay elementos repetidos, la función debe retornar NULL en ambos punteros.

**Ejercicio 4**

Implemente un programa cliente para la función del ejercicio anterior que elimine de un arreglo todos sus elementos repetidos. No es necesario conservar el orden en los elementos, por lo que puede eliminar un elemento simplemente intercambiandolo con el último y reduciendo el tamaño del arreglo.

**Ejercicio 5**

Generar aleatoriamente una matriz de números reales de doble precisión de 10 filas por 6 columnas mediante un puntero a un arreglo. Usando notación de punteros determine e informe: a) el promedio de la fila que el usuario ingrese como dato; b) la suma de cada columna.
Analice: ¿sería posible, manteniendo la estructura de puntero a arreglo, que el usuario ingresara el tamaño de la matriz (filas x columnas)?

**Ejercicio 6**

Modifique el ejercicio anterior de manera que sea posible reservar memoria para una matriz de tamaño arbitrario (tanto para filas como columnas) mediante un arreglo de punteros. Responda: ¿se almacenan los datos en memoria de la misma forma que en una matriz convencional?

**Ejercicio 7**

Escriba una función *busca_mayor()* que permita buscar el mayor elemento de un arreglo de structs según diferentes criterios. La función debe recibir como argumentos el arreglo de structs de tipo alumno, el tamaño del mismo, y un puntero a otra función que utilizará para comparar dos elementos de tipo alumno y saber cual es el mayor de ellos.

```
struct alumno {
    char nombre[50];
    int dni, edad, calificación;
};
```

Implemente funciones que comparen dos alumnos según su dni, según su edad, según su calificación, y según su nombre alfabéticamente, y genere un programa cliente que permita ingresar los datos de N alumnos y buscar el mayor según el criterio que el usuario seleccione mediante un menú.