



Ejercicio 2

Docente: Jimmy Nataniel Requena Llorentty

Materia: Programación III

Carrera: Ingeniería En Sistemas

Estudiantes: Joaquin Marcos Maita Flores

Santa Cruz – Bolivia

2025

Gestión Dinámica Completa

```
#include <iostream> // Para std::cout, std::endl

int main() {
    // 1. Asignar memoria para un solo entero
    int *p_entero = nullptr; // Siempre inicializar punteros
    p_entero = new int;      // Solicita memoria en el Heap para un int

    if (p_entero != nullptr) { // Buena práctica: verificar si new tuvo éxito
        (aunque suele lanzar excepción)
        *p_entero = 123;      // Asigna un valor a la memoria recién reservada
        std::cout << "Entero dinamico creado. Valor: " << *p_entero
                  << " en direccion: " << p_entero << std::endl;

        delete p_entero;     // Libera la memoria
        p_entero = nullptr;  // ¡Buena práctica! Evita puntero colgante.
        std::cout << "Memoria del entero dinamico liberada." << std::endl;
    } else {
        std::cout << "ERROR: No se pudo asignar memoria para p_entero." <<
std::endl;
    }

    // 2. Asignar memoria para un arreglo de doubles
    std::cout << "\n--- Arreglo Dinamico ---" << std::endl;
    double *p_arreglo_doubles = nullptr;
    int tamano_arreglo = 5;
    p_arreglo_doubles = new double[tamano_arreglo]; // Solicita memoria para 5
doubles

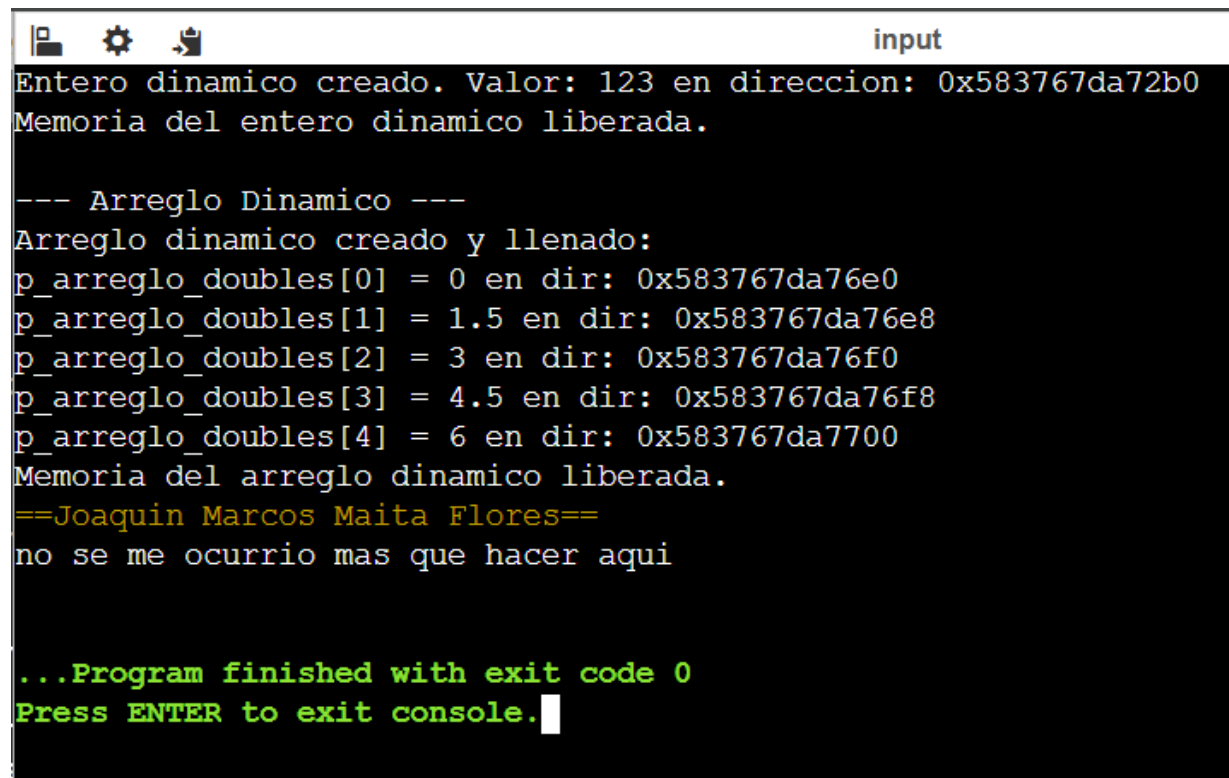
    if (p_arreglo_doubles != nullptr) {
        for (int i = 0; i < tamano_arreglo; ++i) {
            p_arreglo_doubles[i] = i * 1.5; // Asigna valores al arreglo
        }

        std::cout << "Arreglo dinamico creado y llenado:" << std::endl;
        for (int i = 0; i < tamano_arreglo; ++i) {
            std::cout << "p_arreglo_doubles[" << i << "] = " <<
p_arreglo_doubles[i]
                  << " en dir: " << (p_arreglo_doubles + i) << std::endl;
        }

        delete[] p_arreglo_doubles; // ¡IMPORTANTE! Usar delete[] para arreglos
        p_arreglo_doubles = nullptr; // Buena práctica
        std::cout << "Memoria del arreglo dinamico liberada." << std::endl;
    }
}
```

```
    } else {  
        std::cout << "ERROR: No se pudo asignar memoria para p_arreglo_doubles."  
<< std::endl;  
    }  
  
    // Intentar usar un puntero nulo (solo para demostrar, usualmente causa error  
o comportamiento indefinido)  
    // if (p_entero == nullptr) {  
    //     std::cout << "\np_entero es ahora nullptr." << std::endl;  
    //     // *p_entero = 789; // ¡Esto causaría un error de segmentación!  
(Descomentar con precaución)  
    // }  
    std::cout << "\033[33m==Joaquin Marcos Maita Flores==\033[0m" <<  
std::endl;  
    std::cout<< "no se me ocurrio mas que hacer aqui"<<std::endl;  
  
    return 0;  
}
```

CÓDIGO EJECUTADO



```
input  
Entero dinamico creado. Valor: 123 en direccion: 0x583767da72b0  
Memoria del entero dinamico liberada.  
  
--- Arreglo Dinamico ---  
Arreglo dinamico creado y llenado:  
p_arreglo_doubles[0] = 0 en dir: 0x583767da76e0  
p_arreglo_doubles[1] = 1.5 en dir: 0x583767da76e8  
p_arreglo_doubles[2] = 3 en dir: 0x583767da76f0  
p_arreglo_doubles[3] = 4.5 en dir: 0x583767da76f8  
p_arreglo_doubles[4] = 6 en dir: 0x583767da7700  
Memoria del arreglo dinamico liberada.  
==Joaquin Marcos Maita Flores==  
no se me ocurrio mas que hacer aqui  
  
...Program finished with exit code 0  
Press ENTER to exit console.
```

Este ejemplo enseña:

1. **new y delete** → Reservar y liberar memoria dinámica.
 - `int *p = new int;` → Crea un entero en el *heap*.
 - `delete p;` → Libera la memoria.
2. **Arreglos dinámicos** → Uso de `new[]` y `delete[]`.
 - `double *arr = new double[5];` → Arreglo de 5 doubles.
 - `delete[] arr;` → Libera el arreglo.
3. **Buenas prácticas:**
 - Inicializar punteros con `nullptr`.
 - Verificar si `new` tuvo éxito (`if (p != nullptr)`).
 - Asignar `nullptr` después de `delete` para evitar punteros colgantes.
4. **Aritmética de punteros** → Acceso a arreglos con `p[i]` o `*(p + i)`.
5. **¡Error común!** → No usar `delete` en lugar de `delete[]` para arreglos.