# SEPARATA: ARREGLOS UNIDIMENSIONALES EN C++

## ÍNDICE



---

## INTRODUCCIÓN

Los arreglos unidimensionales, también conocidos como arrays o vectores, son una estructura de datos fundamental en C++ que permite almacenar múltiples elementos del mismo tipo de dato en ubicaciones de memoria contiguas. Son especialmente útiles cuando necesitamos manejar colecciones de datos homogéneos.

### Objetivos de Aprendizaje

Al finalizar esta separata, el estudiante será capaz de:

- Declarar e inicializar arreglos unidimensionales
- Acceder y modificar elementos de un arreglo
- Implementar operaciones básicas con arreglos
- Resolver problemas prácticos utilizando arreglos

---

## CONCEPTOS FUNDAMENTALES

### ¿Qué es un Arreglo Unidimensional?

Un arreglo unidimensional es una colección ordenada de elementos del mismo tipo de dato,

almacenados en posiciones consecutivas de memoria. Cada elemento puede ser accedido mediante un índice numérico.

### Características Principales

- **Homogeneidad**: Todos los elementos deben ser del mismo tipo
- **Tamaño fijo**: El tamaño se define al momento de la declaración
- **Indexación**: Los elementos se acceden mediante índices (0 a n-1)
- **Memoria contigua**: Los elementos se almacenan consecutivamente

### Ventajas

- Acceso rápido a elementos (O(1))
- Uso eficiente de memoria
- Iteración secuencial simple
- Compatible con algoritmos estándar

### Desventajas

- Tamaño fijo una vez declarado
- Inserción y eliminación costosas
- No hay verificación automática de límites

---

## DECLARACIÓN DE ARREGLOS

### Sintaxis Básica

```
cpp

tipo_dato nombre_arreglo[tamaño];
```

### Ejemplos de Declaración

```
cpp
```

```cpp
#include <iostream>
using namespace std;

int main() {
    // Declaración de arreglos de diferentes tipos
    int numeros[5];          // Arreglo de 5 enteros
    float notas[10];         // Arreglo de 10 números flotantes
    char vocales[5];         // Arreglo de 5 caracteres
    double precios[20];      // Arreglo de 20 números double

    return 0;
}
```

### Declaración con Constantes

cpp

```cpp
#include <iostream>
using namespace std;

const int TAMAÑO = 100;

int main() {
    int datos[TAMAÑO];       // Usando constante
    int matriz[10 * 5];      // Usando expresión constante

    return 0;
}
```

---

## INICIALIZACIÓN DE ARREGLOS

### 1. Inicialización en la Declaración

cpp

```cpp
#include <iostream>
using namespace std;

int main() {
    // Inicialización completa
    int numeros[5] = {10, 20, 30, 40, 50};

    // Inicialización parcial (resto se llena con 0)
    int datos[8] = {1, 2, 3};  // {1, 2, 3, 0, 0, 0, 0, 0}

    // Inicialización automática del tamaño
    int valores[] = {5, 10, 15, 20};  // Tamaño = 4

    // Inicialización con todos los elementos en 0
    int ceros[10] = {0};

    return 0;
}
```

## 2. Inicialización Posterior

```cpp
```

```cpp
#include <iostream>
using namespace std;

int main() {
    int arreglo[5];

    // Asignación individual
    arreglo[0] = 100;
    arreglo[1] = 200;
    arreglo[2] = 300;
    arreglo[3] = 400;
    arreglo[4] = 500;

    // Usando bucles
    for(int i = 0; i < 5; i++) {
        arreglo[i] = i * 10;
    }

    return 0;
}
```

---

## ACCESO A ELEMENTOS

### Sintaxis de Acceso

```cpp
nombre_arreglo[índice]
```

### Ejemplo Práctico

```cpp
```

```cpp
#include <iostream>
using namespace std;

int main() {
    int numeros[5] = {10, 20, 30, 40, 50};

    // Lectura de elementos
    cout << "Primer elemento: " << numeros[0] << endl;      // 10
    cout << "Tercer elemento: " << numeros[2] << endl;      // 30
    cout << "Último elemento: " << numeros[4] << endl;      // 50

    // Modificación de elementos
    numeros[1] = 25;
    numeros[3] = 45;

    // Mostrar arreglo modificado
    cout << "Arreglo modificado: ";
    for(int i = 0; i < 5; i++) {
        cout << numeros[i] << " ";
    }
    cout << endl;

    return 0;
}
```

## Recorrido con Diferentes Bucles

cpp

```cpp
#include <iostream>
using namespace std;

int main() {
    int datos[6] = {2, 4, 6, 8, 10, 12};

    // Recorrido con for tradicional
    cout << "Con for tradicional: ";
    for(int i = 0; i < 6; i++) {
        cout << datos[i] << " ";
    }
    cout << endl;

    // Recorrido con while
    cout << "Con while: ";
    int j = 0;
    while(j < 6) {
        cout << datos[j] << " ";
        j++;
    }
    cout << endl;

    // Recorrido con for-each (C++11)
    cout << "Con for-each: ";
    for(int elemento : datos) {
        cout << elemento << " ";
    }
    cout << endl;

    return 0;
}
```

---

## OPERACIONES BÁSICAS

### 1. Búsqueda Lineal

cpp

```cpp
#include <iostream>
using namespace std;

int buscarElemento(int arreglo[], int tamaño, int elemento) {
    for(int i = 0; i < tamaño; i++) {
        if(arreglo[i] == elemento) {
            return i;  // Retorna el índice donde encontró el elemento
        }
    }
    return -1;  // Elemento no encontrado
}

int main() {
    int numeros[8] = {3, 7, 1, 9, 4, 6, 2, 8};
    int buscar = 6;

    int posicion = buscarElemento(numeros, 8, buscar);

    if(posicion != -1) {
        cout << "Elemento " << buscar << " encontrado en posición: " << posicion << endl;
    } else {
        cout << "Elemento " << buscar << " no encontrado" << endl;
    }

    return 0;
}
```

## 2. Encontrar Máximo y Mínimo

cpp

```cpp
#include <iostream>
using namespace std;

void encontrarMaxMin(int arreglo[], int tamaño, int &maximo, int &minimo) {
    maximo = minimo = arreglo[0];

    for(int i = 1; i < tamaño; i++) {
        if(arreglo[i] > maximo) {
            maximo = arreglo[i];
        }
        if(arreglo[i] < minimo) {
            minimo = arreglo[i];
        }
    }
}

int main() {
    int datos[7] = {15, 3, 9, 21, 7, 12, 5};
    int max, min;

    encontrarMaxMin(datos, 7, max, min);

    cout << "Máximo: " << max << endl;
    cout << "Mínimo: " << min << endl;

    return 0;
}
```

## 3. Calcular Suma y Promedio

cpp

```cpp
#include <iostream>
using namespace std;

int main() {
    float notas[5] = {8.5, 7.2, 9.0, 6.8, 8.9};
    float suma = 0;

    // Calcular suma
    for(int i = 0; i < 5; i++) {
        suma += notas[i];
    }

    // Calcular promedio
    float promedio = suma / 5;

    cout << "Suma de notas: " << suma << endl;
    cout << "Promedio: " << promedio << endl;

    return 0;
}
```

## 4. Ordenamiento Burbuja

cpp

```cpp
#include <iostream>
using namespace std;

void ordenamientoBurbuja(int arreglo[], int tamaño) {
    for(int i = 0; i < tamaño - 1; i++) {
        for(int j = 0; j < tamaño - 1 - i; j++) {
            if(arreglo[j] > arreglo[j + 1]) {
                // Intercambiar elementos
                int temp = arreglo[j];
                arreglo[j] = arreglo[j + 1];
                arreglo[j + 1] = temp;
            }
        }
    }
}

void mostrarArreglo(int arreglo[], int tamaño) {
    for(int i = 0; i < tamaño; i++) {
        cout << arreglo[i] << " ";
    }
    cout << endl;
}

int main() {
    int numeros[6] = {64, 34, 25, 12, 22, 11};

    cout << "Arreglo original: ";
    mostrarArreglo(numeros, 6);

    ordenamientoBurbuja(numeros, 6);

    cout << "Arreglo ordenado: ";
    mostrarArreglo(numeros, 6);

    return 0;
}
```

# EJEMPLOS PRÁCTICOS

## Ejemplo 1: Sistema de Calificaciones

cpp

```cpp
#include <iostream>
#include <iomanip>
using namespace std;

int main() {
    const int NUM_ESTUDIANTES = 5;
    float calificaciones[NUM_ESTUDIANTES];
    float suma = 0;
    int aprobados = 0, reprobados = 0;

    // Entrada de datos
    cout << "=== SISTEMA DE CALIFICACIONES ===" << endl;
    for(int i = 0; i < NUM_ESTUDIANTES; i++) {
        cout << "Ingrese calificación del estudiante " << (i+1) << ": ";
        cin >> calificaciones[i];
        suma += calificaciones[i];

        if(calificaciones[i] >= 7.0) {
            aprobados++;
        } else {
            reprobados++;
        }
    }

    // Cálculos y resultados
    float promedio = suma / NUM_ESTUDIANTES;

    cout << "\n=== RESULTADOS ===" << endl;
    cout << fixed << setprecision(2);
    cout << "Promedio del grupo: " << promedio << endl;
    cout << "Estudiantes aprobados: " << aprobados << endl;
    cout << "Estudiantes reprobados: " << reprobados << endl;

    // Mostrar todas las calificaciones
    cout << "\nCalificaciones registradas: ";
    for(int i = 0; i < NUM_ESTUDIANTES; i++) {
        cout << calificaciones[i] << " ";
    }
    cout << endl;

    return 0;
}
```

## Ejemplo 2: Análisis de Ventas Mensuales

cpp

```cpp
#include <iostream>
#include <string>
using namespace std;

int main() {
    const int MESES = 12;
    string nombresMeses[MESES] = {"Enero", "Febrero", "Marzo", "Abril",
                                  "Mayo", "Junio", "Julio", "Agosto",
                                  "Septiembre", "Octubre", "Noviembre", "Diciembre"};
    float ventas[MESES];
    float totalVentas = 0;
    int mejorMes = 0, peorMes = 0;

    // Entrada de datos
    cout << "=== ANÁLISIS DE VENTAS ANUALES ===" << endl;
    for(int i = 0; i < MESES; i++) {
        cout << "Ventas de " << nombresMeses[i] << ": $";
        cin >> ventas[i];
        totalVentas += ventas[i];

        // Encontrar mejor y peor mes
        if(ventas[i] > ventas[mejorMes]) {
            mejorMes = i;
        }
        if(ventas[i] < ventas[peorMes]) {
            peorMes = i;
        }
    }

    // Mostrar resultados
    cout << "\n=== REPORTE ANUAL ===" << endl;
    cout << "Ventas totales: $" << totalVentas << endl;
    cout << "Promedio mensual: $" << totalVentas/MESES << endl;
    cout << "Mejor mes: " << nombresMeses[mejorMes] << " ($" << ventas[mejorMes] << ")" << endl;
    cout << "Peor mes: " << nombresMeses[peorMes] << " ($" << ventas[peorMes] << ")" << endl;

    return 0;
}
```

### Ejemplo 3: Contador de Frecuencias

```cpp
#include <iostream>
using namespace std;

int main() {
    const int TAMAÑO = 15;
    int numeros[TAMAÑO] = {1, 2, 3, 2, 1, 4, 3, 2, 5, 1, 3, 4, 2, 1, 3};
    int frecuencias[6] = {0}; // Para números del 1 al 5

    // Contar frecuencias
    cout << "Arreglo: ";
    for(int i = 0; i < TAMAÑO; i++) {
        cout << numeros[i] << " ";
        frecuencias[numeros[i]]++;
    }
    cout << endl;

    // Mostrar frecuencias
    cout << "\n=== TABLA DE FRECUENCIAS ===" << endl;
    for(int i = 1; i <= 5; i++) {
        cout << "Número " << i << ": " << frecuencias[i] << " veces" << endl;
    }

    return 0;
}
```

---

## EJERCICIOS PROPUESTOS

### Ejercicio 1: Manipulación Básica

**Dificultad: Básico**

Escriba un programa que:

1. Declare un arreglo de 10 enteros
2. Permita al usuario ingresar los valores
3. Muestre el arreglo original
4. Muestre el arreglo en orden inverso

5. Calcule y muestre la suma de elementos pares

## Ejercicio 2: Estadísticas de Temperaturas

**Dificultad: Intermedio**

Desarrolle un programa que almacene las temperaturas de una semana y:

1. Calcule la temperatura promedio
2. Encuentre la temperatura máxima y mínima
3. Cuente cuántos días la temperatura estuvo por encima del promedio
4. Muestre un reporte completo

## Ejercicio 3: Búsqueda y Reemplazo

**Dificultad: Intermedio**

Implemente un programa que:

1. Tenga un arreglo predefinido de 15 números
2. Permita buscar un elemento específico
3. Si lo encuentra, lo reemplace por un nuevo valor
4. Muestre el arreglo antes y después de la operación
5. Indique en qué posiciones se hicieron cambios

## Ejercicio 4: Ordenamiento por Selección

**Dificultad: Intermedio-Avanzado**

Implemente el algoritmo de ordenamiento por selección:

1. Cree una función para ordenar el arreglo
2. Muestre el paso a paso del ordenamiento
3. Compare el número de intercambios vs. ordenamiento burbuja
4. Permita ordenamiento ascendente o descendente

## Ejercicio 5: Análisis de Datos

**Dificultad: Avanzado**

Desarrolle un sistema completo que:

1. Almacene las calificaciones de 20 estudiantes

2. Calcule estadísticas completas (media, mediana, moda)
3. Genere histograma de frecuencias por rangos (0-2, 3-4, 5-6, 7-8, 9-10)
4. Identifique estudiantes con calificaciones atípicas (outliers)
5. Genere un reporte detallado

---

## TEST DE AUTOEVALUACIÓN

### Pregunta 1 (Conceptos Básicos)

¿Cuál es la principal diferencia entre un arreglo y una variable simple?

a) Los arreglos usan más memoria
b) Los arreglos pueden almacenar múltiples valores del mismo tipo
c) Los arreglos son más rápidos
d) No hay diferencia

### Pregunta 2 (Indexación)

Si tengo un arreglo `int datos[8]`, ¿cuál es el índice del último elemento?

a) 8
b) 7
c) 9
d) Depende del contenido

### Pregunta 3 (Inicialización)

¿Qué sucede con el siguiente código?

```cpp
int arreglo[5] = {1, 2};
```

a) Error de compilación
b) Los elementos no inicializados quedan con valores aleatorios
c) Los elementos no inicializados se llenan con 0
d) Solo se crean 2 elementos

### Pregunta 4 (Código)

¿Cuál será la salida del siguiente código?

```cpp
int nums[4] = {10, 20, 30, 40};
nums[1] = nums[0] + nums[3];
cout << nums[1];
```

a) 20

b) 30

c) 40

d) 50

### Pregunta 5 (Análisis)

Para buscar un elemento en un arreglo no ordenado, ¿cuál es la complejidad temporal en el peor caso?

a) O(1)

b) O(log n)

c) O(n)

d) $O(n^2)$

### Pregunta 6 (Programación)

¿Cuál es la forma correcta de pasar un arreglo a una función en C++?

a) `void funcion(int arreglo[10])` b) `void funcion(int arreglo[])` c) `void funcion(int *arreglo)` d) Todas las anteriores son válidas

### Pregunta 7 (Memoria)

¿Cómo se almacenan los elementos de un arreglo en memoria?

a) En posiciones aleatorias

b) En posiciones consecutivas

c) En orden inverso

d) Depende del compilador

### Pregunta 8 (Límites)

¿Qué sucede si intento acceder a `arreglo[10]` en un arreglo declarado como `int arreglo[5]`?

a) Error de compilación

b) Comportamiento indefinido

c) Se retorna 0

d) Se crea automáticamente el elemento

### Pregunta 9 (Algoritmos)

En el ordenamiento burbuja, ¿cuántas comparaciones se realizan en el peor caso para un arreglo de n elementos?

a) n
b) n-1
c) n(n-1)/2
d) $n^2$

### Pregunta 10 (Aplicación)

¿Cuál sería la mejor estructura para almacenar las temperaturas diarias del año?

a) 365 variables individuales
b) Un arreglo de 365 elementos
c) 12 arreglos de diferentes tamaños
d) Una variable que se actualice cada día

---

## SOLUCIONES

### Soluciones del Test

**1. b)** Los arreglos pueden almacenar múltiples valores del mismo tipo **2. b)** 7 (los índices van de 0 a n-1) **3. c)** Los elementos no inicializados se llenan con 0 **4. d)** 50 (10 + 40 = 50) **5. c)** O(n) - búsqueda lineal **6. d)** Todas las anteriores son válidas **7. b)** En posiciones consecutivas **8. b)** Comportamiento indefinido **9. c)** n(n-1)/2 **10. b)** Un arreglo de 365 elementos

### Solución Ejercicio 1

```cpp
```

```cpp
#include <iostream>
using namespace std;

int main() {
    int arreglo[10];
    int sumaPares = 0;

    // Entrada de datos
    cout << "Ingrese 10 números enteros:" << endl;
    for(int i = 0; i < 10; i++) {
        cout << "Número " << (i+1) << ": ";
        cin >> arreglo[i];
    }

    // Mostrar arreglo original
    cout << "\nArreglo original: ";
    for(int i = 0; i < 10; i++) {
        cout << arreglo[i] << " ";
    }

    // Mostrar en orden inverso
    cout << "\nArreglo inverso: ";
    for(int i = 9; i >= 0; i--) {
        cout << arreglo[i] << " ";
    }

    // Calcular suma de pares
    for(int i = 0; i < 10; i++) {
        if(arreglo[i] % 2 == 0) {
            sumaPares += arreglo[i];
        }
    }

    cout << "\nSuma de elementos pares: " << sumaPares << endl;

    return 0;
}
```

## Solución Ejercicio 2

cpp

```cpp
#include <iostream>
#include <string>
using namespace std;

int main() {
    string dias[7] = {"Lunes", "Martes", "Miércoles", "Jueves",
                      "Viernes", "Sábado", "Domingo"};
    float temperaturas[7];
    float suma = 0, promedio;
    float maxTemp, minTemp;
    int diasSobrePromedio = 0;

    // Entrada de datos
    cout << "=== TEMPERATURAS DE LA SEMANA ===" << endl;
    for(int i = 0; i < 7; i++) {
        cout << "Temperatura del " << dias[i] << ": ";
        cin >> temperaturas[i];
        suma += temperaturas[i];
    }

    // Calcular promedio
    promedio = suma / 7;

    // Encontrar máximo y mínimo
    maxTemp = minTemp = temperaturas[0];
    for(int i = 1; i < 7; i++) {
        if(temperaturas[i] > maxTemp) maxTemp = temperaturas[i];
        if(temperaturas[i] < minTemp) minTemp = temperaturas[i];
    }

    // Contar días sobre promedio
    for(int i = 0; i < 7; i++) {
        if(temperaturas[i] > promedio) diasSobrePromedio++;
    }

    // Mostrar reporte
    cout << "\n=== REPORTE SEMANAL ===" << endl;
    cout << "Temperatura promedio: " << promedio << "°C" << endl;
    cout << "Temperatura máxima: " << maxTemp << "°C" << endl;
    cout << "Temperatura mínima: " << minTemp << "°C" << endl;
    cout << "Días sobre el promedio: " << diasSobrePromedio << endl;
```

```
    return 0;
}
```

---

## REFERENCIAS Y RECURSOS ADICIONALES

### Libros Recomendados

- "C++ Primer" por Stanley Lippman
- "Effective C++" por Scott Meyers
- "Programming: Principles and Practice Using C++" por Bjarne Stroustrup

### Recursos Online

- cppreference.com - Documentación completa de C++
- GeeksforGeeks - Tutoriales y ejemplos
- Codecademy C++ Course - Curso interactivo

### Herramientas de Desarrollo

- Code::Blocks - IDE gratuito
- Dev-C++ - Compilador ligero
- Visual Studio Code - Editor moderno
- Online Compilers: Repl.it, OnlineGDB

---

**Nota**: Esta separata ha sido diseñada como material de apoyo para el aprendizaje de arreglos unidimensionales en C++. Se recomienda practicar con los ejercicios propuestos y experimentar con variaciones de los ejemplos presentados.