

Programación Orientada a Objetos

Un ejemplo: La clase matriz

José Fidel Argudo Argudo Francisco Palomo Lozano
Inmaculada Medina Bulo Gerardo Aburrizaga García



Versión 1.0



```
1  // Clase para matrices de números en coma flotante y doble precisión .

3  #ifndef MATRIZ_H_
4  #define MATRIZ_H_
5  #include <valarray>
6  #include <initializer_list>

8  using std::valarray;
9  using std::slice;
10 using std::slice_array;
11 using std::initializer_list;


13 class matriz {
14 public:
15     // Constructores
16     explicit matriz(size_t m = 1, size_t n = 1, double y = 0.0);
17     matriz(size_t m, size_t n, double f(size_t i, size_t j));
18     matriz(const initializer_list<valarray<double>>& l); // C++11
```

```
20 // Copia y movimiento (C++11)
21 matriz(const matriz&) = default;
22 matriz(matriz&&) = default;
23 matriz& operator =(const matriz&) = default;
24 matriz& operator =(matriz&&) = default;
```

```
26 // Dimensión
27 size_t filas() const;
28 size_t columnas() const;
```

```
30 // Operadores de indización
31 double operator ()(size_t i, size_t j) const;
32 double& operator ()(size_t i, size_t j);
33 valarray<double> operator [] (size_t i) const;
34 slice_array<double> operator [] (size_t i);
35 valarray<double> operator ()(size_t j) const;
36 slice_array<double> operator ()(size_t j);
```

Sobrecarga de operadores de lectura y lectura/escritura de filas [] y columnas ()
Valarray devuelve una referencia al vector en cuestión



```
38  // Asignación de valor constante
39  matriz& operator =(double y);

41  // Operadores de asignación compuesta
42  matriz& operator +=(const matriz& a);
43  matriz& operator -=(const matriz& a);
44  matriz& operator *=(const matriz& a);
45  matriz& operator *=(double y);

47  // Matriz opuesta
48  friend matriz operator -(const matriz& a);

50  private:
51      size_t m, n;
52      valarray<double> x;
53  };
```

filas y columnas

Se representa la matriz como una fila a continuación de otra.

```
55 // Prototipos de operadores externos no amigos que trabajan con matrices.
56 const matriz& operator +(const matriz& a);
57 matriz operator +(const matriz& a, const matriz& b);
58 matriz operator -(const matriz& a, const matriz& b);
59 matriz operator *(const matriz& a, const matriz& b);
60 matriz operator *(double y, const matriz& a);
61 matriz operator *(const matriz& a, double y);

63 // Definiciones en línea.
64 #include "matriz-inline.h"

66 #endif // MATRIZ_H_
```

matriz-inline.h

```
1 // Notas:
2 //
3 // Este fichero contiene definiciones en línea de funciones y operadores
4 // relacionados con la clase matriz y se incluye en la cabecera "matriz.h".
5 // Por lo tanto, no debe incluirse ni compilarse por separado.

7 // Constructor: matriz constante

9 inline matriz::matriz(size_t m, size_t n, double y):
10     m(m), n(n), x(y, m * n)
11 {}

13 // Dimensión.

15 inline size_t matriz::filas() const
16 { return m; }

18 inline size_t matriz::columnas() const
19 { return n; }
```

matriz-inline.h

21 *// Operadores de indización*

23 *// Selección de elemento*

```
25 inline double matriz::operator()(size_t i, size_t j) const
26 { return x[i * n + j]; }
```

```
28 inline double& matriz::operator()(size_t i, size_t j)
29 { return x[i * n + j]; }
```

31 *// Selección de fila .*

```
33 inline valarray<double> matriz::operator [] (size_t i) const
34 { return x[slice(i * n, n, 1)]; }
```

```
36 inline slice_array<double> matriz::operator [] (size_t i)
37 { return x[slice(i * n, n, 1)]; }
```

```
39 // Selección de columna.
```

```
41 inline valarray<double> matriz::operator()(size_t j) const
```

```
42 { return x[slice(j, m, n)]; }
```



```
44 inline slice_array<double> matriz::operator()(size_t j)
```

```
45 { return x[slice(j, m, n)]; }
```

```
47 // Asignación de valor constante
```

```
49 inline matriz& matriz::operator=(double y)
```

```
50 {
```

```
51     x = y;
```

```
52     return *this;
```

```
53 }
```


matriz-inline.h

55 *// Operadores de auto-suma, auto-resta y auto-multiplicación .*

57 inline matriz& matriz::operator +=(const matriz& a)

58 {

59 x += x;

60 return *this;

61 }

Suma elemento a elemento de la matriz

63 inline matriz& matriz::operator -=(const matriz& a)

64 {

65 x -= a.x;

66 return *this;

67 }

69 inline matriz& matriz::operator *=(const matriz& a)

70 { return *this = *this * a; }

matriz-inline.h

```
72 inline matriz& matriz::operator *=(double y)
73 {
74     x *= y;
75     return *this;
76 }

77 // Operadores de signo.

78 inline const matriz& operator +(const matriz& a)
79 { return a; }

80 inline matriz operator -(const matriz& a)
81 {
82     matriz c(a);
83     c.x = -c.x;
84     return c;
85 }
```

90 *// Operadores de suma y resta .*

92 `inline` matriz operator +(const matriz& a, const matriz& b)

93 { `return` matriz(a) + b; }

95 `inline` matriz operator -(const matriz& a, const matriz& b)

96 { `return` matriz(a) -= b; }

98 *// Operadores de producto por escalar*

100 `inline` matriz operator *(double y, const matriz& a)

101 { `return` matriz(a) *= y; }

103 `inline` matriz operator *(const matriz& a, double y)

104 { `return` y * a; }

```
1  #include "matriz.h"

3  // Constructor: matriz definida por una función
4  matriz::matriz(size_t m, size_t n, double f(size_t i, size_t j)):
5      m(m), n(n), x(m * n)
6  {
7      for (size_t i = 0; i < m; ++i)
8          for (size_t j = 0; j < n; ++j)
9              (*this)(i, j) = f(i, j);
10 }

12 // Constructor desde lista de filas de la matriz (C++11)
13 matriz::matriz(const initializer_list<valarray<double>>& l):
14     m(l.size()), n(l.begin()->size()), x(m * n)
15 {
16     auto it = l.begin();
17     for (size_t i = 0; i < m && it != l.end(); ++i, ++it)
18         (*this)[i] = *it;
19 }
```

en cada posición asignamos el valor que nos devuelve la función f

```
21 // Operador de multiplicación
22 matriz operator *(const matriz& a, const matriz& b)
23 {
24     matriz c(a.filas(), b.columnas()); // Matriz nula
25     for (size_t i = 0; i < a.filas(); ++i)
26         for (size_t j = 0; j < b.columnas(); ++j)
27             c(i, j) = (a[i] * b[j]).sum();
28     return c;
29 }
```

Programa de prueba

```
1  #include <iostream>
2  #include "matriz.h"
3  using namespace std;

4
5  // Inserción de una matriz en un flujo de salida .
6  ostream& operator <<(ostream& fs, const matriz& a)
7  {
8      for (size_t i = 0; i < a.filas(); ++i) {
9          for (size_t j = 0; j < a.columnas(); ++j)
10             fs << a(i,j) << ' ';
11         fs << endl;
12     }
13     return fs;
14 }

15
16 // Función delta de Kronecker.
17 inline double delta(size_t i, size_t j)
18 {
19     return i == j;
20 }
```

Programa de prueba

```
22 // Prueba.

24 int main()
25 {
26     matriz A(3, 3);           // Matriz nula de 3 x 3
27     matriz B(3, 3, 2.0);     // Matriz de 3 x 3 con todos sus elementos a 2
28     matriz C(3, 3, delta);   // Matriz identidad de 3 x 3
29     matriz D = { {1, 2, 3},
30                 {4, 5, 6},
31                 {7, 8, 9} };
32     A = C;
33     cout << "A_□=\n" << A << endl;
34     B += -A;
35     cout << "B_□=\n" << B << endl;
36     C *= C += C;
37     cout << "C_□=\n" << C << endl;
38     cout << "2A_□+□B_□*□C_□=\n"
39         << 2 * A + B * C << endl;
40     cout << "D=\n" << D << endl;
```

Programa de prueba

```
41  matriz tD(3, 3);
42  for (size_t i = 0; i < D.filas(); ++i)
43      tD[i] = D(i);
44  cout << "traspuesta de D=\n" << tD << endl;
45  for (size_t i = 0; i < D.filas(); ++i)
46      for (size_t j = 0; j < D.columnas(); ++j)
47          --D(i,j);
48  cout << "D=\n" << D << endl;
49  D = -1.;
50  cout << "D=\n" << D << endl;
51 }
```