

Tarea - 6 Sobrecarga de operadores

Alumno: Ángel Alonso Galarza Chávez
Profesor: Dr. Cuauhtemoc Mancillas López
Curso: Programación Avanzada

Introducción

La sobrecarga de operadores permite expandir el uso de los operadores (+, -, /, *) para ser utilizados en tareas mas avanzadas definidas por el programador en relación a una clase. Permite una facilidad de manejo de código al encapsular todo en una función a la cual se puede utilizar tan solo usando operadores, lo que ayuda a implicar el entendimiento del código volviéndolo mas intuitivo. Esto permite que los objetos de estas clases se comporten de manera similar a los tipos de datos primitivos, lo que puede hacer que el código sea más intuitivo.

En este ejemplo, se realizo una clase llamada Arreglo, en la cual haremos una sobrecarga de operadores para realizar operaciones de aritmética tales como sumar dos arreglos ($arr1 + arr2$), resta de dos arreglos ($arr1 - arr2$), incremento del arreglo ($++arr1$), decremento de un arreglo ($--arr1$), multiplicación de un numero a cada elemento del arreglo ($arr1 * n$), divisiones de un numero al arreglo ($arr1 / n$) y la división de arreglos ($arr1 / arr2$).// Un operador puede estar sobrecargado mas de una vez, para permitir realizar la acción según sea los parámetros utilizados, esto permite obtener el mismo resultado de distintas maneras ayudando a la lectura y entendimiento del código.

La sobrecarga de operadores no solo esta para los operadores de aritmética, también esta disponibles para operadores de lógica como && (AND), || (OR) y los operadores unarios como ! (NOT)

Código C++

A continuación se mostrara el código realizado para la sobrecarga de operadores con el ejemplo de los arreglos. En la clase arreglo se definieron el tamaño del arreglo como un arreglo que podemos asignarle una cantidad de elementos según sea necesitado al momento de llamar al constructor. Los operadores sobrecargados son los operadores de la aritmética tales como la suma, resta, multiplicación y división, ademas de incorporar los incrementos y decrementos.

```
1 #include <iostream>
2
3 class Arreglo {
4 private:
5     int *arr;
6     int size;
7
8 public:
9     Arreglo(int s) : size(s) {
10         arr = new int[size];
11         for (int i = 0; i < size; ++i) {
12             arr[i] = 0;
13         }
14     }
15
16     ~Arreglo() { delete[] arr; }
17     // Metodo para print el contenido del arreglo
18     void print() const {
19         for (int i = 0; i < size; ++i) {
20             std::cout << arr[i] << " ";
21         }
22         std::cout << std::endl;
23     }
24
25     // Sobrecarga del operador ++
26     Arreglo &operator++() {
27         for (int i = 0; i < size; ++i) {
28             arr[i] += 1;
```

```

29     }
30     return *this;
31 }
32
33 // Sobrecarga del operador --
34 Arreglo &operator--() {
35     for (int i = 0; i < size; ++i) {
36         arr[i] -= 1;
37     }
38     return *this;
39 }
40
41 // Sobrecarga del operador *=
42 Arreglo &operator*=(int n) {
43     for (int i = 0; i < size; i++) {
44         arr[i] *= n;
45     }
46     return *this;
47 }
48
49 // Sobrecarga del operador +=
50 Arreglo &operator+=(Arreglo &arr_2) {
51     if (this->size != arr_2.size) {
52         std::cout << "los arreglos no son de la misma longitud" << '\n';
53         return *this;
54     }
55     for (int i = 0; i < size; i++) {
56         arr[i] += arr_2.arr[i];
57     }
58     return *this;
59 }
60
61 // Sobrecarga del operador +=
62 Arreglo &operator+=(int n) {
63     for (int i = 0; i < size; i++) {
64         arr[i] += n;
65     }
66     return *this;
67 }
68
69 // Sobrecarga del operador -=
70 Arreglo &operator-=(Arreglo &arr_2) {
71     if (this->size != arr_2.size) {
72         std::cout << "los arreglos no son de la misma longitud" << '\n';
73         return *this;
74     }
75     for (int i = 0; i < size; i++) {
76         arr[i] -= arr_2.arr[i];
77     }
78     return *this;
79 }
80
81 // Sobrecarga del operador -=
82 Arreglo &operator-=(int n) {
83     for (int i = 0; i < size; i++) {
84         arr[i] -= n;
85     }
86     return *this;
87 }
88
89 // Sobrecarga del operador /=
90 Arreglo &operator/=(int n) {
91     for (int i = 0; i < size; i++) {
92         arr[i] /= n;
93     }
94     return *this;
95 }
96
97 // Sobrecarga del operador /=
98 Arreglo &operator/=(Arreglo &arr_2) {
99     if (this->size != arr_2.size) {
100         std::cout << "los arreglos no son de la misma longitud" << '\n';
101         return *this;

```

```

102     }
103     for (int i = 0; i < size; i++) {
104         arr[i] /= arr_2.arr[i];
105     }
106     return *this;
107 }
108 };

```

Listing 1: Clase Arreglo

En el siguiente código se muestra como se utilizan los operadores sobrecargados para la clase Arreglo.

```

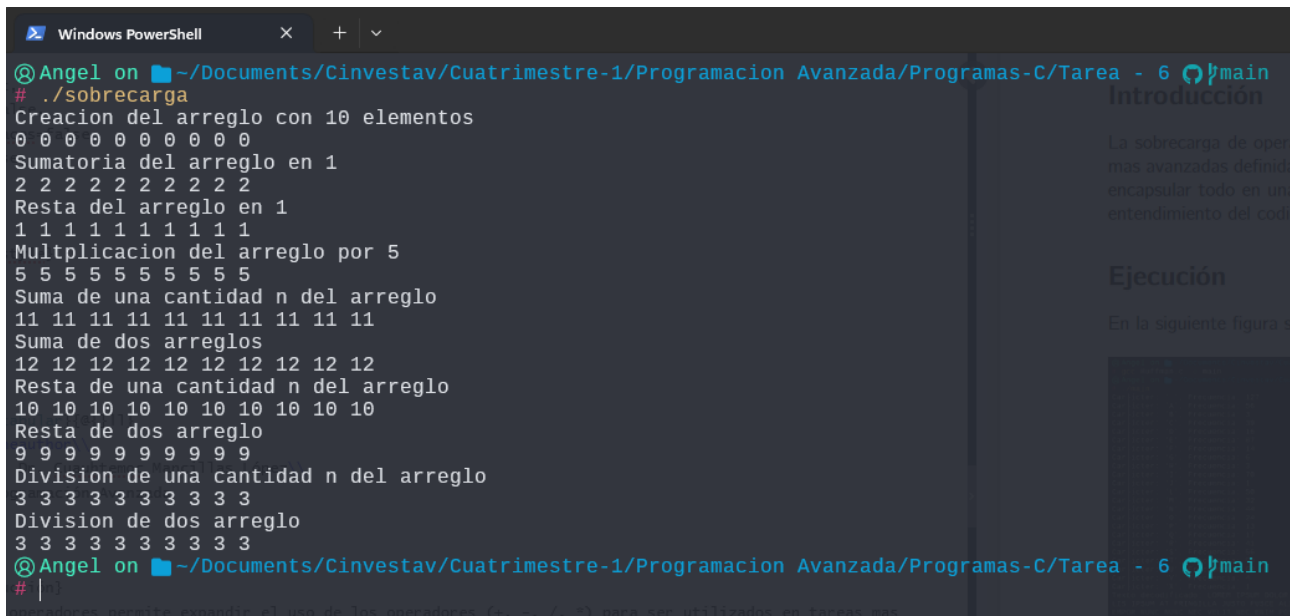
1 int main() {
2     std::cout << "Creacion del arreglo con 10 elementos" << '\n';
3     Arreglo arr(10);
4     arr.print();
5
6     // Suma en 1
7     std::cout << "Sumatoria del arreglo en 1" << '\n';
8     ++arr;
9     ++arr;
10    arr.print();
11
12    // Resta en 1
13    std::cout << "Resta del arreglo en 1" << '\n';
14    --arr;
15    arr.print();
16
17    // Multiplicacion de cada elemento
18    std::cout << "Multiplicacion del arreglo por 5" << '\n';
19    arr *= 5;
20    arr.print();
21
22    // Suma de una cantidad n a cada elemento
23    std::cout << "Suma de una cantidad n del arreglo" << '\n';
24    arr += 6;
25    arr.print();
26
27    // Sumatoria de dos arreglos
28    std::cout << "Suma de dos arreglos" << '\n';
29    Arreglo arr_2(10);
30    ++arr_2;
31    arr += arr_2;
32    arr.print();
33
34    // Resta de una cantidad n a cada elemento
35    std::cout << "Resta de una cantidad n del arreglo" << '\n';
36    arr -= 2;
37    arr.print();
38
39    // Resta de dos arreglos
40    std::cout << "Resta de dos arreglos" << '\n';
41    arr -= arr_2;
42    arr.print();
43
44    // Resta de una cantidad n a cada elemento
45    std::cout << "Division de una cantidad n del arreglo" << '\n';
46    arr /= 3;
47    arr.print();
48
49    // Resta de dos arreglos
50    std::cout << "Division de dos arreglo" << '\n';
51    arr /= arr_2;
52    arr.print();
53
54    return 0;
55 }

```

Listing 2: Funcion main

Ejecución

En la siguiente figura se mostrara la ejecución del código para codificar y decodificar un texto.



```
Windows PowerShell
@Angel on ~\Documents\Cinvestav\Cuatrimestre-1\Programacion Avanzada\Programas-C\Tarea - 6 main
# ./sobrecarga
Creacion del arreglo con 10 elementos
0 0 0 0 0 0 0 0 0 0
Sumatoria del arreglo en 1
2 2 2 2 2 2 2 2 2 2
Resta del arreglo en 1
1 1 1 1 1 1 1 1 1 1
Multiplicacion del arreglo por 5
5 5 5 5 5 5 5 5 5 5
Suma de una cantidad n del arreglo
11 11 11 11 11 11 11 11 11 11
Suma de dos arreglos
12 12 12 12 12 12 12 12 12 12
Resta de una cantidad n del arreglo
10 10 10 10 10 10 10 10 10 10
Resta de dos arreglo
9 9 9 9 9 9 9 9 9 9
Division de una cantidad n del arreglo
3 3 3 3 3 3 3 3 3 3
Division de dos arreglo
3 3 3 3 3 3 3 3 3 3
@Angel on ~\Documents\Cinvestav\Cuatrimestre-1\Programacion Avanzada\Programas-C\Tarea - 6 main
#
```

Figure 1: Ejecución del programa