Ayudantía 9

Repaso Pilas y Colas

Benjamín Aceituno

Programación Avanzada S09

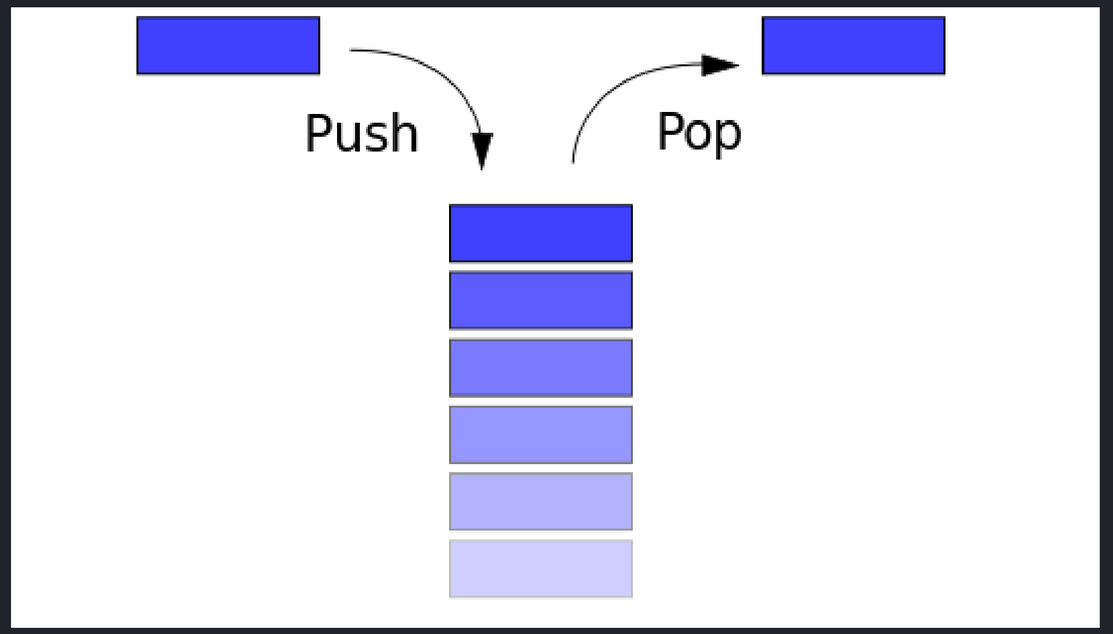
Recordemos lo visto en la ayudantía pasada...

Stack

El stack es un contenedor de objetos que ocupa la metodología LIFO, al igual que el vector, los ultimos elementos en entrar serán los últimos en salir. Se puede entender como por su traducción literal “pila”. Donde los elementos se van apilando dentro del contenedor

Para recordar...

LIFO quiere decir Last In First Out, es decir, último entrar, primero en salir. Intenten verlo como si fuera almacenar una pila de cajas, nosotros ponemos cajas una encima de otra, y si sacamos una caja será la última que hemos puesto.



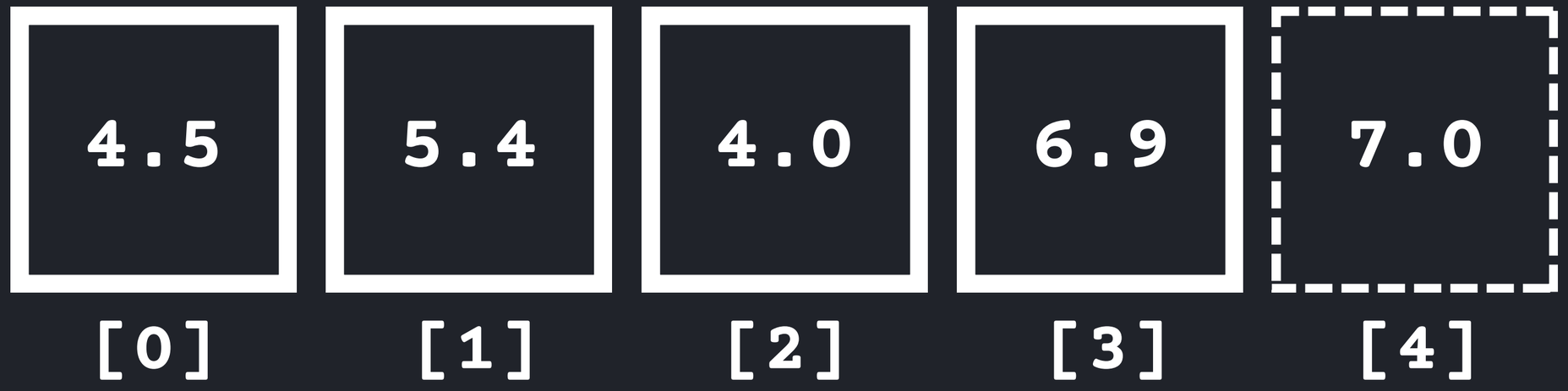
Funciones de stack

```
stack<float> notas;
```


```
notas.push(1.0);
```

Funciones de stack

```
stack<float> notas;
```




```
notas.pop();
```

Recorrer un stack o queue

Con stack y queue no tenemos iteradores como tal como en los vectores, solo tenemos las funciones push y pop, que realizan solo la acción de añadir y eliminar.

Estas funciones realizan una “copia” del elemento que se añada, por lo cual no podemos recorrer el stack y el queue de la misma manera que un vector, por ejemplo.

Recorrer un stack

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

int main() {
    stack<int> x;
    x.push(10);
    x.push(20);
    x.push(30);
    x.push(40);
    // Creamos una copia del stack, como un auxiliar
    stack<int> aux = x;
    cout << "Recorriendo la pila... " << endl;

    while (!aux.empty()) {  // Mientras auxiliar tenga elementos...
        cout << aux.top() << endl;  // Imprimimos el elemento de la cima
        aux.pop();  // Y eliminamos el elemento auxiliar...
    }

    return 0;
}
```

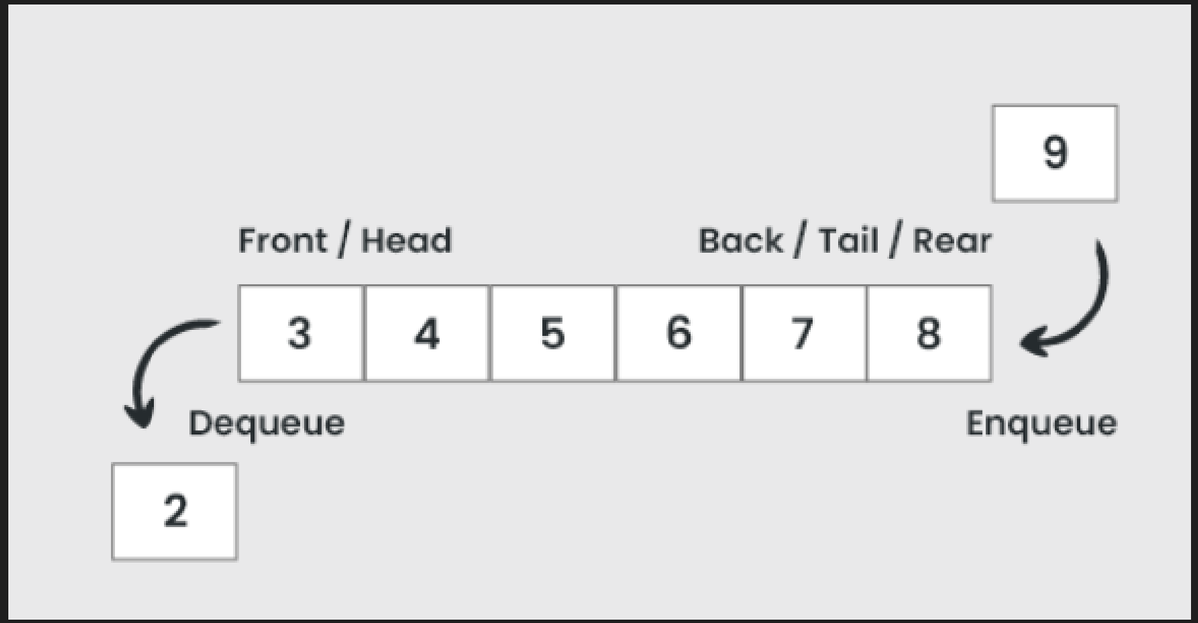
Tenemos que crear una copia del stack ya que si empezamos a eliminar elementos de la original pueden verse afectados

```
Recorriendo la pila...
40
30
20
10
```

Queue

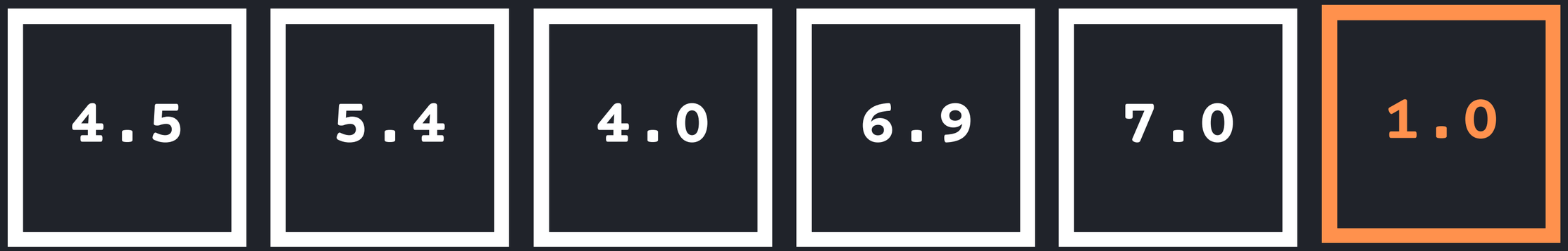
El queue es un contenedor de objetos en C++ que sigue la metodología FIFO (First In, First Out). A diferencia de un stack, donde los últimos elementos en entrar son los primeros en salir, en un queue, los primeros elementos en entrar son los primeros en salir. Se puede entender por su traducción literal como "cola", similar a una fila de personas donde el primero en entrar es el primero en ser atendido.

En este ejemplo se muestra como los elementos se van “encolando” y el elemento de al frente es el primero en ser “desencolado”



Funciones de queue

```
queue<float> notas;
```




```
notas.push(1.0);
```

Funciones de queue

```
queue<float> notas;
```


```
notas.pop();
```

Recorrer una queue

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

int main() {
    queue<int> x;
    x.push(10);
    x.push(20);
    x.push(30);
    x.push(40);
    // Creamos una copia del queue, como un auxiliar
    queue<int> aux = x;
    cout << "Recorriendo la cola... " << endl;

    while (!aux.empty()) {  // Mientras auxiliar tenga elementos...
        cout << aux.front() << endl;  // Imprimimos el elemento de la cima
        aux.pop();  // Y eliminamos el elemento auxiliar...
    }

    return 0;
}
```

Tenemos que crear una copia de la cola ya que si empezamos a eliminar elementos de la original pueden verse afectados

```
Recorriendo la cola...
10
20
30
40
```

Ejercicio

Realice un código en C++ que sea capaz de invertir un stack de números enteros para que esté en un orden correcto, es decir, si se ingresa 1,2,3,4,5 sabemos que al ser un stack quedará como 5,4,3,2,1. Haga un método que sea capaz de leer los datos ingresados en su orden correcto, solo usando stacks.

```cpp
#include <iostream>
#include <stack>

using namespace std;

void leerStack(stack<int> pila){
    stack<int> aux = pila;
    while (!aux.empty()) {  // Mientras auxiliar tenga elementos...
        cout << aux.top() << endl;  // Imprimimos el elemento de la cima
        aux.pop();  // Y eliminamos el elemento auxiliar...
    }
}

void leerStackEnOrden(stack<int> pila) {
    // Stack auxiliar para revertir el orden de los datos
    stack<int> pilaAux;

    // Transferimos los datos del stack principal al stack auxiliar para revertir el orden
    while (!pila.empty()) {
        pilaAux.push(pila.top()); // Ingresamos elementos de una pila a otra pila, asi invertimos los elementos.
        pila.pop();
    }

    // Leemos los datos en el orden original de ingreso (1,2,3,4,5)
    cout << "Lectura de los datos en orden de ingreso: " << endl;
    while (!pilaAux.empty()) {
        cout << pilaAux.top() << endl;
        pilaAux.pop();
    }
}

int main() {
    stack<int> pila;
```

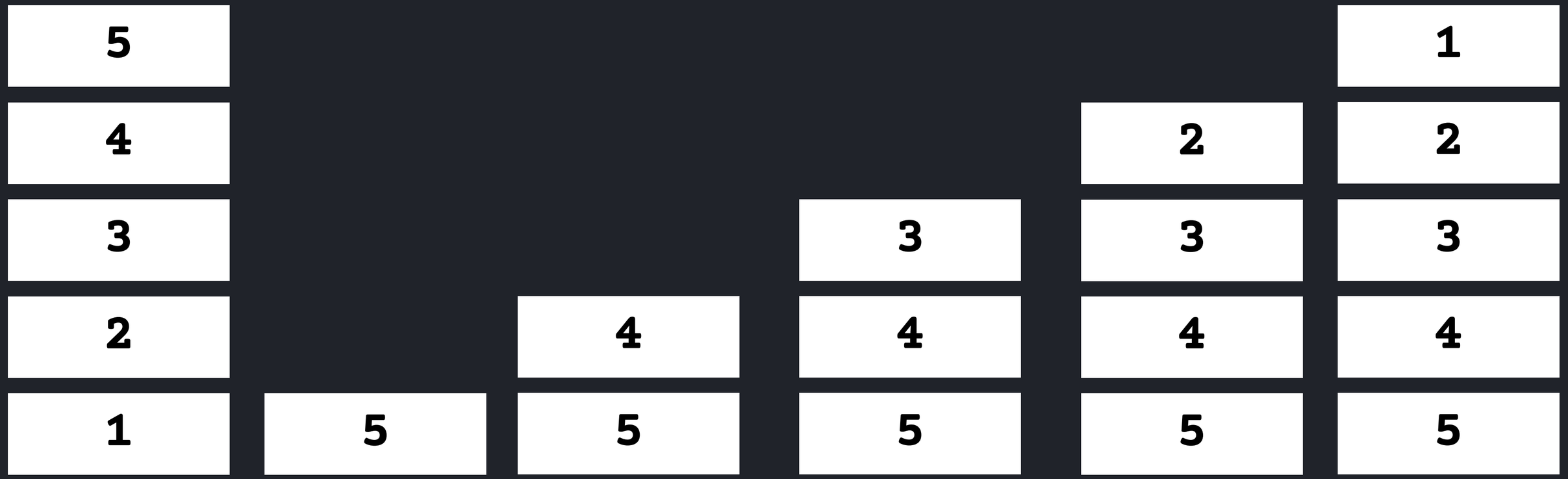
```cpp
int main() {
    stack<int> pila;

    // Ingresamos los datos directamente en el main
    pila.push(1);
    pila.push(2);
    pila.push(3);
    pila.push(4);
    pila.push(5);

    cout << "Datos ingresados en el stack." << endl;
    cout << "Lectura de los datos del stack." << endl;
    leerStack(pila);
    // Llamamos a la función para leer el stack en el orden de ingreso
    leerStackEnOrden(pila);

    return 0;
}
```

Los elementos se van a invertir al ser añadidos de la cima del stack a otro stack.



Ejercicio

Debido al calor que ha hecho ultimamente en la capital, se le pide que realice el siguiente ejercicio en C++. Debe implementar una clase clima, que tiene distintos pronosticos, un pronostico es un mapa que tendrá un stack de string días y una queue de temperaturas int. Debe rellenar en el main los distintos días de lunes a domingo con sus temperaturas igualmente, implemente un método llenarPronostico, que llena el pronostico con días y sus temperaturas, un método que encuentre el día más frío y más caluroso, y finalmente un método que indique los días calurosos, es decir, los días superiores a 30 grados, donde se debe indicar que tiene que usar bloqueador solar.

Recuerde que debe invertir el stack para que mantenga el mismo orden que la queue.

```cpp
#include <iostream>
#include <stack>
#include <queue>
#include <string>
#include <map>

using namespace std;

class Clima {
public:
    map<string, int> pronostico;

    // Método para llenar el mapa pronostico con los días y sus temperaturas
    void llenarPronostico(queue<int> temperaturas, stack<string> dias) {
        dias = invertirStack(dias);  // Invertimos el stack para que quede en orden de lunes a domingo

        while (!temperaturas.empty() && !dias.empty()) {
            pronostico[dias.top()] = temperaturas.front();
            temperaturas.pop();
            dias.pop();
        }
    }

    // Método para encontrar el día con la temperatura más alta
    string diaMasCaluroso() {
        int maxTemp = -1000;
        string diaCaluroso;

        map<string, int>::iterator it = pronostico.begin();
        while (it != pronostico.end()) {
            if (it->second > maxTemp) {
                maxTemp = it->second;
                diaCaluroso = it->first;
            }
            ++it;
        }

        return diaCaluroso;
    }
```

```cpp
// Método para encontrar el día con la temperatura más baja
string diaMasFrio() {
    int minTemp = 1000;
    string diaFrio;

    map<string, int>::iterator it = pronostico.begin();
    while (it != pronostico.end()) {
        if (it->second < minTemp) {
            minTemp = it->second;
            diaFrio = it->first;
        }
        ++it;
    }

    return diaFrio;
}

// Método para imprimir los días con temperatura mayor a 30 grados
void diasCalurosos() {
    cout << "Días con temperaturas mayores a 30 grados (usar bloqueador solar):" << endl;

    map<string, int>::iterator it = pronostico.begin();
    while (it != pronostico.end()) {
        if (it->second > 30) {
            cout << it->first << ": " << it->second << " grados - Recuerde usar bloqueador solar!" << endl;
        }
        ++it;
    }
}
// Función auxiliar para invertir un stack
stack<string> invertirStack(stack<string> original) {
    stack<string> invertido;
    while (!original.empty()) {
        invertido.push(original.top());
        original.pop();
    }
    return invertido;
}
```

```
        return invertido;
    }
private:

};

int main() {
    // Stack de días de la semana en orden de Lunes a Domingo
    stack<string> dias;
    dias.push("Lunes");
    dias.push("Martes");
    dias.push("Miércoles");
    dias.push("Jueves");
    dias.push("Viernes");
    dias.push("Sábado");
    dias.push("Domingo");

    // Cola de temperaturas para cada día (ordenada de Lunes a Domingo)
    queue<int> temperaturas;
    temperaturas.push(28); // lunes
    temperaturas.push(32); // martes
    temperaturas.push(29); // miercoles
    temperaturas.push(35); // jueves
    temperaturas.push(31); // viernes
    temperaturas.push(27); // sabado
    temperaturas.push(33); // domingo

    // se crea una instancia de la clase Clima y llenar el pronóstico
    Clima clima;
    clima.llenarPronostico(temperaturas, dias);

    // encuentra y muestra el día más caluroso y más frío
    string diaCaluroso = clima.diaMasCaluroso();
    string diaFrio = clima.diaMasFrio();

    cout << "Día más caluroso: " << diaCaluroso << " con " << clima.pronostico[diaCaluroso] << " grados." << endl;
    cout << "Día más frío: " << diaFrio << " con " << clima.pronostico[diaFrio] << " grados." << endl;

    // Muestra los días con temperaturas mayores a 30 grados
    clima.diasCalurosos();

    return 0;
}
```

Contacto

+569 86031881

benjamin.aceituno@mail.udp.cl

benja.mp4

https://github.com/benjamp4/prog.av-2024-2