

# TADESCALONADA.pdf



Anónimo



Análisis de Algoritmos y Estructuras de Datos



2º Grado en Ingeniería Informática



Escuela Superior de Ingeniería  
Universidad de Cádiz



**Que no te escriban poemas de amor  
cuando terminen la carrera**



*(a nosotros por  
suerte nos pasa)*

**WUOLAH**

# WUOLAH

Oh Wuolah wuolita  
Tu que eres tan bonita

```

// postCondicion: elimina un salto de la funcion escalonada y se prolonga el escalon
anterior hasta el siguiente, si el
// punto no existe , no hace nada
// double valorPunto (double xi, double yi );
// postCondicion : devuelve el valor de la funcion en el punto (xi,yi), si el punto no existe
en la funcion devuelve
// un valor muy alto
// double valorMinimo();
// postCondicion : devuelve el minimo de la funcion, si la funcion esta vacia devuelve un
valor muy alto
// double valorMaximo();
// postCondicion : devuelve el maximo de la funcion, si la funcion esta vacia, devuelve un
valor muy bajo
// void translacion ( double w , double z);
// postCondicion : traslada la funcion w unidades horizontales y z unidades verticales
// ~Escalonada();
// destruye la funcion escalonada;
typedef struct {
    double xi , yi ;
    Salto(double a, double b) : xi(a), yi(b) {};
}salto;

class Escalonada{
public:
    explicit Escalonada(int numSaltos, double xi = 0 double yi = 0);
    void añadirSalto( double xi, double yi);
    void eliminarSalto ( double xi, double yi);
    double valorPunto (double xi, double yi );
    double valorMinimo();
    double valorMaximo();
    void translacion ( double w , double z);
    ~Escalonada();

private:
    int saltosMax , saltosActuales;
    Lista<salto> funcion();
    //double minimo , maximo ;
    Lista<salto>::posicion posicion;

};

Escalonada::Escalonada(int numSaltos, double xi = 0 double yi = 0){
    //minimo = max();
    //maximo = min();

```

```

    saltosMax = numSaltos;
    saltosActuales =0;
    funcion.insertar(salto(xi,yi));
}

void Escalonada::añadirSalto( double xi, double yi){
    //modificar lo anterior

}

void Escalonada::eliminarSalto ( double xi, double yi){
    funcion.eliminar(buscar(xi,yi));
}

double Escalonada::valorPunto (double xi, double yi ){

}

double Escalonada::valorMinimo(){
    // recorrer toda la lista y ver el valor del punto
}
double Escalonada::valorMaximo(){
    //

}

void Escalonada::traslacion ( double w , double z){
    // Sumar a todas las x w y a las y z
}
Escalonada::~Escalonada(){
    ~funcion();
}

```