En C++

// Interfaz del TAD. Pre-declaraciones:

```
template <typename Elemento> struct Pila;

template <typename Elemento> void vacia(Pila<Elemento>& p);
template <typename Elemento> void apilar(Pila<Elemento>& p, const Elemento& dato);
template <typename Elemento> void desapilar(Pila<Elemento>& p);
template <typename Elemento> void cima(const Pila<Elemento>& p, Elemento& dato, bool& error);
template <typename Elemento> bool esVacia(const Pila<Elemento>& p);
template <typename Elemento> int altura(const Pila<Elemento>& p);
template <typename Elemento> void duplicar(const  Pila<Elemento>& pOrigen,
                                           Pila<Elemento>& pDestino);
template <typename Elemento> bool operator== (const Pila<Elemento>& p1, const Pila<Elemento>& p2);
template <typename Elemento> void liberar(Pila<Elemento>& p);
template <typename Elemento> void iniciarIterador(Pila<Elemento>& p);
template <typename Elemento> bool existeSiguiente(const Pila<Elemento>& p);
template <typename Elemento> bool siguiente(Pila<Elemento>& p, Elemento& dato);

…
```

En C++

// Declaración

```
template <typename Elemento> struct Pila{

    friend void vacia<Elemento>(Pila<Elemento>& p);
    friend void apilar<Elemento>(Pila<Elemento>& p, const Elemento& dato);
    friend void desapilar<Elemento>(Pila<Elemento>& p);
    friend void cima<Elemento>(const Pila<Elemento>& p, Elemento& dato, bool& error);
    friend bool esVacia<Elemento>(const Pila<Elemento>& p);
    friend int altura<Elemento>(const Pila<Elemento>& p);
    friend void duplicar<Elemento>(const  Pila<Elemento>& pOrigen, Pila<Elemento>& pDestino);
    friend bool operator==<Elemento> (const Pila<Elemento>& p1, const Pila<Elemento>& p2);
    friend void liberar<Elemento>(Pila<Elemento>& p);
    friend void iniciarIterador<Elemento>(Pila<Elemento>& p);
    friend bool existeSiguiente<Elemento>(const Pila<Elemento>& p);
    friend bool siguiente<Elemento>(Pila<Elemento>& p, Elemento& dato);
…
```

En C++

```
// Representación de los valores del TAD

private:
        struct unDato {
                Elemento dato;
                unDato* sig;
        };

        unDato* cim ;
        int alt;
        unDato* iter;
};

// Implementación de las operaciones
```

… transparencia siguiente

En C++

```
// Implementación de las operaciones

template<typename Elemento> void vacia(Pila<Elemento>& p) {
        p.alt = 0;
        p.cim = nullptr;
}

template <typename Elemento> void apilar(Pila<Elemento>& p, const Elemento& e){
        typename Pila<Elemento>::unDato* aux;
        aux = new typename Pila<Elemento>::unDato;
        aux->dato = e;
        aux->sig = p.cim;
        p.cim = aux;
        p.alt++;
}

// etc etc implementación de las demás operaciones
```