

# Templates

## Algoritmos y Estructuras de Datos II

# Template

Inglés detectado ▾

# template

Editar

'templət

Español ▾

# modelo

sustantivo **la plantilla**  
template, stencil, insole, jig,  
sock, slipsole

Mostrar menos

## max.h

```
int max(int a, int b) {
    if (a > b) {
        return a;
    } else {
        return b;
    }
}

double max(double a, double b) {
    if (a > b) {
        return a;
    } else {
        return b;
    }
}

#define user_type char
user_type max(user_type a, user_type b) {
    if (a > b) {
        return a;
    } else {
        return b;
    }
}
```

## max.h'


```
char max(char a, char b) {
    if (a > b) {
        return a;
    } else {
        return b;
    }
}
```

## max.h

```
user_type max(user_type a, user_type b) {  
    if (a > b) {  
        return a;  
    } else {  
        return b;  
    }  
}
```

## todos\_loos\_max.h

```
#define user_type int  
#include "max.h"  
#define user_type double  
#include "max.h"  
#define user_type char  
#include "max.h"
```



## todos\_loos\_max.h'

```
int max(int a, int b) {  
    if (a > b) {  
        return a;  
    } else {  
        return b;  
    }  
}  
  
double max(double a, double b) {  
    if (a > b) {  
        return a;  
    } else {  
        return b;  
    }  
}
```

```
char max(char a, char b) {  
    if (a > b) {  
        return a;  
    } else {  
        return b;  
    }  
}
```

```
template<class T>
```

- ▶ Antes de cada declaración o definición
- ▶ Solo afecta esa declaración o definición
- ▶ En ese contexto, T es un tipo de datos, solo que no sabemos cual.

```
template<class T>
```

```
T max(T a, T b);
```

```
template<class T>
```

```
bool esMayor(T x, T y);
```

```
template<class K>
```

```
bool esMayor(K x, K y) {  
    return x > y;  
}
```

## max.h

```
template<class C>  
C max(C a, C b);
```

## max.cpp

```
template<class C>  
C max(C a, C b) {  
    if (a < b) {  
        return a;  
    } else {  
        return b;  
    }  
}
```

max.o

## main.cpp

```
#include "max.h"  
int main() {  
    int x = 5;  
    int y = 6;  
    int j = max(x, y);  
}
```

main.o

main



## max.h

```
template<class C>  
C max(C a, C b);
```

## max.cpp

```
template<class C>  
C max(C a, C b) {  
    if (a < b) {  
        return a;  
    } else {  
        return b;  
    }  
}
```

## max.o

## main.cpp

```
#include "max.h"  
int main() {  
    int x = 5;  
    int y = 6;  
    int j = max(x, y);  
}
```

## main.o

## main



main.cpp:(.text+0x21): undefined reference to 'int max<int>(int, int)'

## max.h

```
template<class C>  
C max(C a, C b);
```

## max.cpp

```
template<class C>  
C max(C a, C b) {  
    if (a < b) {  
        return a;  
    } else {  
        return b;  
    }  
}
```

max.o

## main.cpp

```
#include "max.h"  
int main() {  
    int x = 5;  
    int y = 6;  
    int j = max(x, y);  
}
```

main.o

main



main.cpp:(.text+0x21): undefined reference to 'int max<int>(int, int)'  
Cuando compilo max.cpp, ¿Que tipo uso para C?



## max.hpp

```
template<class C>
C max(C a, C b) {
    if (a < b) {
        return a;
    } else {
        return b;
    }
}
```

## main.cpp

```
#include "max.hpp"
int main() {
    int x = 5;
    int y = 6;
    int j = max(x, y);
}
```

main.o

main

*max.hpp*

```
template<class C>
C max(C a, C b) {
    if (a < b) {
        return a;
    } else {
        return b;
    }
}
```

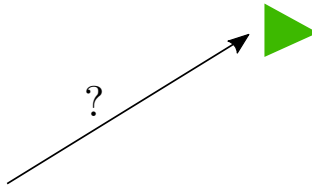
*trimax.hpp*

```
#include "max.hpp"
template<class C>
C trimax(C a, C b, C c) {
    return max(max(a, b), c);
}
```

*main.cpp*

```
#include "max.hpp"
#include "trimax.hpp"
int main() {
    int x = 5;
    int y = 6;
    int z = 10;
    int j = max(x, y);
    int h = trimax(x, y, z);
}
```

main



```
march@elaine:~/algoritmos2/clases/labo/03_templates/ejemplos_codigo/compile_2$ g++ main.cpp -o main
In file included from trimax.hpp:2:0,
                 from main.cpp:3:
max.hpp:3:3: error: redefinition of 'template<class C> C max(C, C)'
  C max(C a, C b) {
    ^
In file included from main.cpp:2:0:
max.hpp:3:3: note: 'template<class C> C max(C, C)' previously declared here
  C max(C a, C b) {
    ^
```



Figura: Acordate del ifndef, luke!

max.hpp

```
#ifndef MAX_HPP
#define MAX_HPP
template<class C>
C max(C a, C b) {
    if (a < b) {
        return a;
    } else {
        return b;
    }
}
#endif
```

trimax.hpp

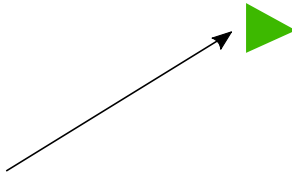
```
#ifndef TRIMAX_HPP
#define TRIMAX_HPP

#include "max.hpp"
template<class C>
C trimax(C a, C b, C c) {
    return max(max(a, b), c);
}
#endif
```

main.cpp

```
#include "max.hpp"
#include "trimax.hpp"
int main() {
    int x = 5;
    int y = 6;
    int z = 10;
    int j = max(x, y);
    int h = trimax(x, y, z);
}
```

main



### Par.cpp

```
Par::Par(int izq, char der) : _izq(izq), _der(der) {  
}  
  
int Par::primero() const {  
    return _izq;  
}  
  
char Par::segundo() const {  
    return _der;  
}
```

### Par.h

```
class Par {  
public:  
    Par(int izq, char der);  
    int primero() const;  
    char segundo() const;  
  
private:  
    int _izq;  
    char _der;  
};
```

### main.cpp

```
#include "Par.cpp"  
  
int main() {  
    Par p(5, 'c');  
    Par p2(10, 'd');  
}
```

Par. h

```
class Par {
public:
    Par(int izq, char der);
    int primero() const;
    char segundo() const;

private:
    int _izq;
    char _der;
};
```

Par.cpp

```
Par::Par(int izq, char der)
    : _izq(izq), _der(der) { }

int Par::primero() const {
    return _izq;
}

char Par::segundo() const {
    return _der;
}
```

par.hpp

```
template<class Tizq, class Tder>
class Par {
public:
    Par(Tizq izq, Tder der);
    Tizq primero() const;
    Tder segundo() const;

private:
    Tizq _izq;
    Tder _der;
};

template<class Tizq, class Tder>
Par<Tizq, Tder>::Par(Tizq izq, Tder der)
    : _izq(izq), _der(der) { }

template<class Tizq, class Tder>
Tizq Par<Tizq, Tder>::primero() const {
    return _izq;
}

template<class Tizq, class Tder>
Tder Par<Tizq, Tder>::segundo() const {
    return _der;
}
```

par.hpp

```
template<class Tizq, class Tder>
class Par {
public:
    Par(Tizq izq, Tder der);
    Tizq primero() const;
    Tder segundo() const;

private:
    Tizq _izq;
    Tder _der;
};

template<class Tizq, class Tder>
Par<Tizq, Tder>::Par(Tizq izq, Tder der) : _izq(izq), _der(der) {}

template<class Tizq, class Tder>
Tizq Par<Tizq, Tder>::primero() const {
    return _izq;
}

template<class Tizq, class Tder>
Tder Par<Tizq, Tder>::segundo() const {
    return _der;
}
```

```
#include "par.hpp"
#include <string>

using namespace std;

int main() {
    Par<int, int> p(5, 10);
    Par<char, int> q('d', 15);
    Par<string, string> r("Hola", "mundo");
    Par<Par<string, int>, char> x(Par<string, int>("Cinco", 5), '5');
}
```



## ¿Puedo usar cualquier tipo para un template?

```
template<class T>
T max(T a, T b) {
    if (a < b) {
        return a;
    } else {
        return b;
    }
}

int main() {
    int m_int = max(3, 5);
    Par<int, int> m_par = max(Par<int, int>(10, 2),
                             Par<int, int>(4, 10));
}
```

## ¿Puedo usar cualquier tipo para un template?

```
template<class T>
T max(T a, T b) {
    if (a < b) {
        return a;
    } else {
        return b;
    }
}
```

```
int main() {
    int m_int = max(3, 5);
    Par<int, int> m_par = max(Par<int, int>(10, 2),
                             Par<int, int>(4, 10));
}
```

```
130 march@elaine:~/algoritmos2/clases/labo/03_templates/ejemplos_codigo) g++ tipos.cpp -o tipos
tipos.cpp: In instantiation of 'T max(T, T) [with T = Par<int, int>]':
tipos.cpp:15:51:   required from here
tipos.cpp:5:11: error: no match for 'operator<' (operand types are 'Par<int, int>' and 'Par<int, int>')
    if (a < b) {
        ^
```