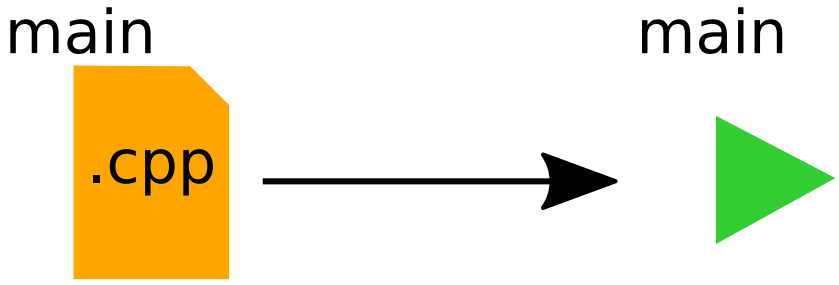
Bajo el capó

Algoritmos y Estructuras de Datos II

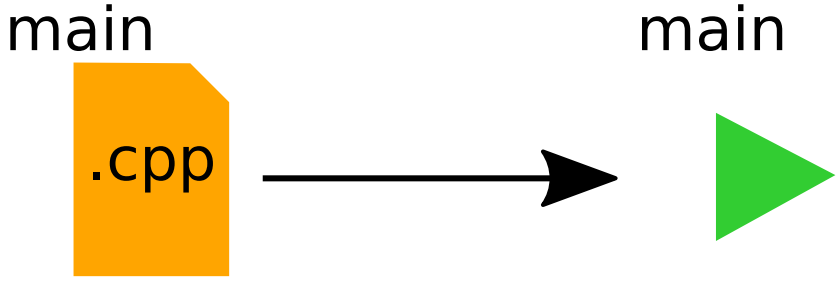
¿Qué pasa cuando compilo?

```
$> g++ main.cpp -o main
```

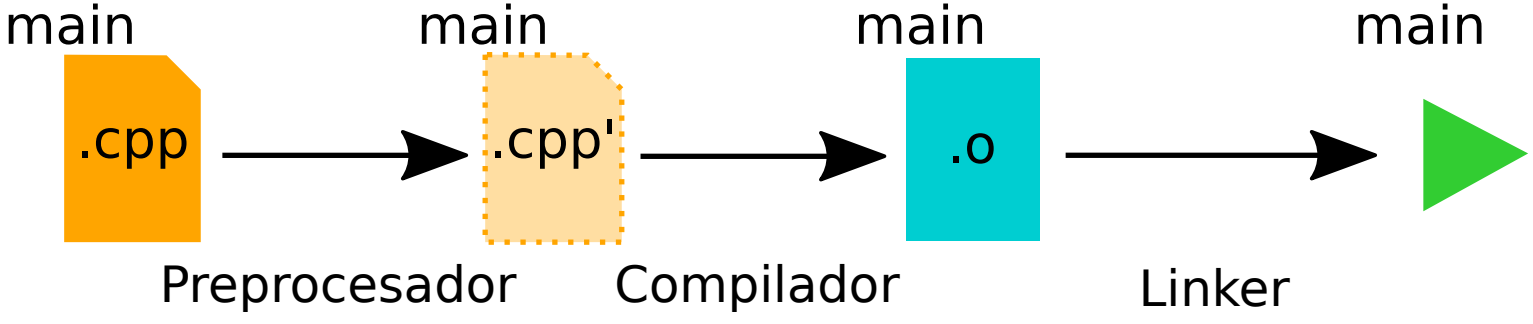


¿Qué pasa cuando compilo?

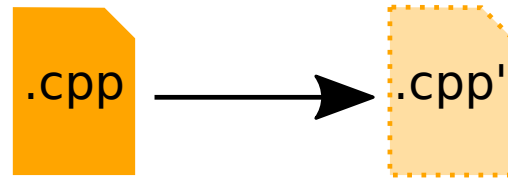
```
$> g++ main.cpp -o main
```




```
$> g++ -c main.cpp -o main.o
$> g++ main.o -o main
```



The C++ Preprocessor[1]

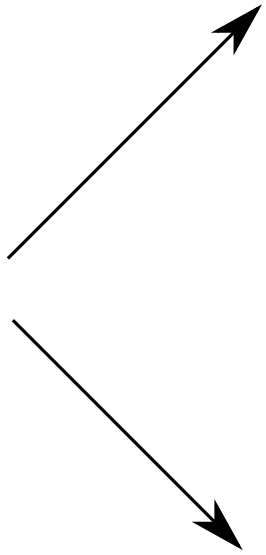


- Directivas específicas para manipular el texto del código
- Directivas empiezan con #
- Ejemplos: #define, #include, #ifdef, #ifndef, #if, #endif

[1] http://en.cppreference.com/w/cpp/preprocessor

```
#define DEBUG

int foo(int x) {
  if (x % 2 == 0) {
    return x / 2;
    #ifdef DEBUG
    cout << "x era par" << endl;
    #endif
  } else {
    return x + 1;
  }
}
```



```
int foo(int x) {
  if (x % 2 == 0) {
    return x / 2;
  } else {
    return x + 1;
  }
}
```

```
int foo(int x) {
  if (x % 2 == 0) {
    return x / 2;
    cout << "x era par" << endl;
  } else {
    return x + 1;
  }
}
```

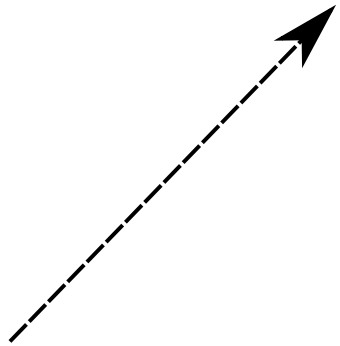
a.cpp

```
int foo(int x) {
    return x + 5;
}
```

b.cpp

```
#include "a.cpp"

int bar(int y) {
    return foo(y) + 4;
}
```



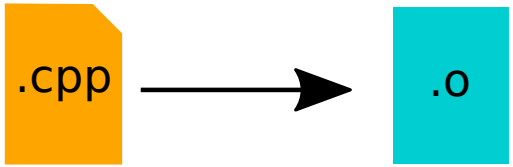
b.cpp'

```
int foo(int x) {
    return x + 5;
}

int bar(int y) {
    return foo(y) + 4;
}
```

```
#include <vector>
```

Compilación (esquemático)




```
$> g++ -c bar.cpp -o bar.o
```

```
int external(int x);

int foo() {
    int a = 4;
    int b = a + 2;
    b++;
    b = external(b);
    return b;
}

int bar() {
    int x = 5;
    x + foo();
    return x;
}
```

```
int external(int x);

int foo() {
   01100101011111
   00110001011101
   10111000110010
   01   external(b);

}

int bar() {
   11100101100110
   110 foo();
   10001110100001
}
```

```
int external(int x);

int foo() {
   01100101011111
   00110001011101
   10111000110010
   01   external(b);

}

int bar() {
   11100101100110
   110 foo();
   10001110100001
}
```

Compilación (esquemático)

Compilación

```
$> g++ -c bar.cpp -o bar.o
```

```
int external(int x);

int foo() {
    int a = 4;
    int b = a + 2;
    b++;
    b = external(b);
    return b;
}

int bar() {
    int x = 5;
    x + foo();
    return x;
}
```

→

```
int external(int x);

int foo() {
   01100101011111
   00110001011101
   10111000110010
   01   external(b);

}

int bar() {
   11100101100110
   110 foo();
   10001110100001
}
```

→

```
int external(int x);

int foo() {
   01100101011111
   00110001011101
   10111000110010
   01   external(b);

}                    → ?

int bar() {
   11100101100110
   110 foo();
   10001110100001
}
```

```
$> g++ -c ext.cpp -o ext.o
```

```
int external(int x) {
  return x + 10;
}
```

→

```
int external(int x) {
   00110001011101
}
```

```
$> g++ -c main.cpp -o main.o
```

```
int bar();

int main() {
  bar();
}
```

→

```
int bar();

int main() {
  bar();
}
```

→

```
int bar();

int main() {
  bar(); ——————→ ?
}
```

Linkeo

```
int foo() {
   01100101011111
   00110001011101
   10111000110010
   01   external(b);

}

int bar() {
   11100101100110
   110 foo();
   10001110100001
}

int external(int x) {
   00110001011101
}

int main() {
  bar();
}
```

```
$> g++ bar.o
       ext.o
       main.o
    -o exec
```

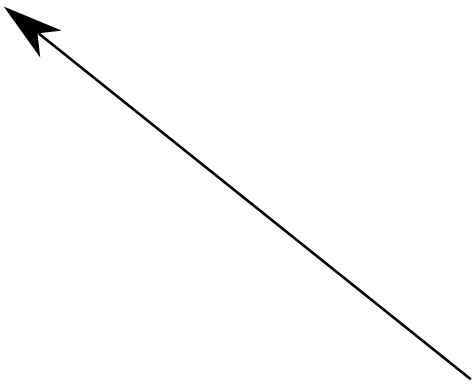
a.cpp

```
int foo(int x) {
    return x + 5;
}
```

b.cpp

```
#include "a.cpp"

int bar(int y) {
    return foo(y) + 4;
}
```

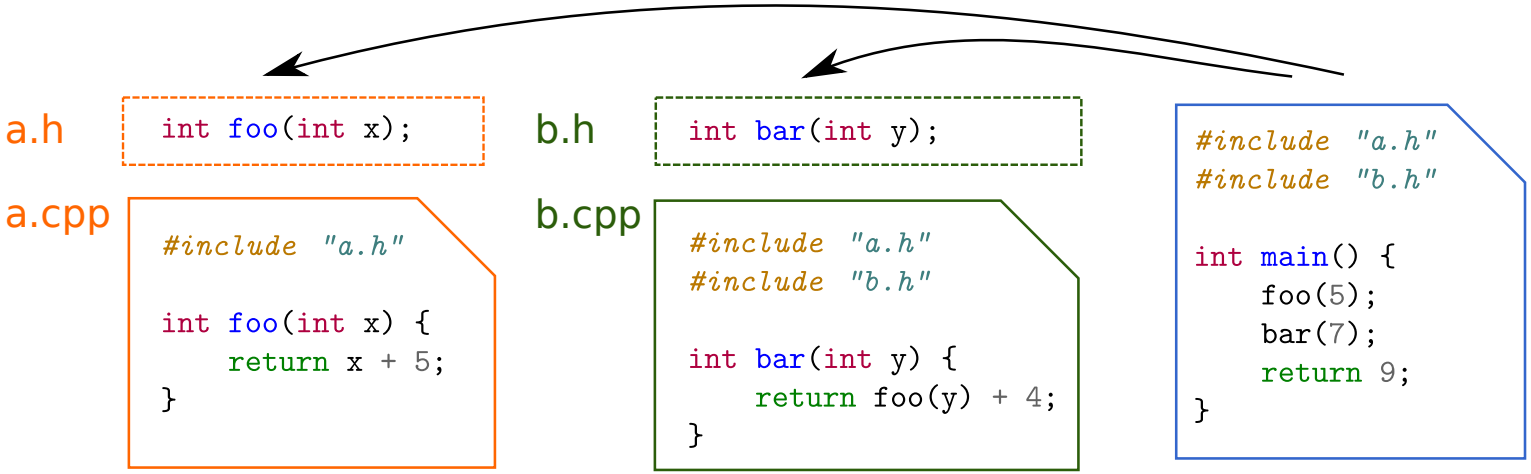


```
#include "a.cpp"
#include "b.cpp"

int main() {
    foo(5);
    bar(7);
    return 9;
}
```

```
$> g++ -c a.cpp -o a.o
$> g++ -c b.cpp -o b.o
$> g++ -c a_b_main.cpp -o a_b_main.o
$> g++ a.o b.o a_b_main.o -o a_b_main
```

```
In file included from b.cpp:1:0,
                 from a_b_main.cpp:2:
a.cpp: In function ‘int foo(int)’:
a.cpp:1:5: error: redefinition of ‘int foo(int)’
 int foo(int x) {
     ^
In file included from a_b_main.cpp:1:0:
a.cpp:1:5: note: ‘int foo(int)’ previously defined here
 int foo(int x) {
     ^
```




```
$> g++ -c a.cpp -o a.o

$> g++ -c b.cpp -o b.o

$> g++ -c a_b_main.cpp -o a_b_main.o

$> g++ a.o b.o a_b_main.o -o a_b_main
```

class.h

```
#ifndef CLASS_H
#define CLASS_H

#include <string>

using namespace std;

class MiClase {

public:
    MiClase();

    int obs_1();
    string obs_2();

private:
    int val_1;
    string val_2;

};

#endif
```

class.cpp

```
#include "class.h"
#include <string>

using namespace std;

MiClase::MiClase() {
    val_1 = 10;
    val_2 = "Hola";
}

int MiClase::obs_1() {
    return val_1 + 5;
}

string MiClase::obs_2() {
    if (val_1 + 5) {
        return val_2;
    } else {
        return val_2 + " mundo";
    }
}
```