Composición de Programas

Modularización

- Los lenguajes de programación nos deben ofrecer maneras de agrupar y organizar partes de nuestro código.
- Esto nos permitirá:
  - Distintos niveles de abstracción
  - Fácil reutilización de código
  - Mejor comprensión de la funcionalidad

Dos niveles

- A nivel de código fuente.
  - Formas de dividir nuestro código fuente donde agrupamos distintas funcionalidades.
  - Normalmente: Funciones, Clases

- A nivel de compilados / ejecutables.
  - Código ya compilado que expone determinada funcionalidad.

Caso C++

- A nivel de código fuente:
  - Clases
  - **Namespaces** (similar a paquetes de Java)

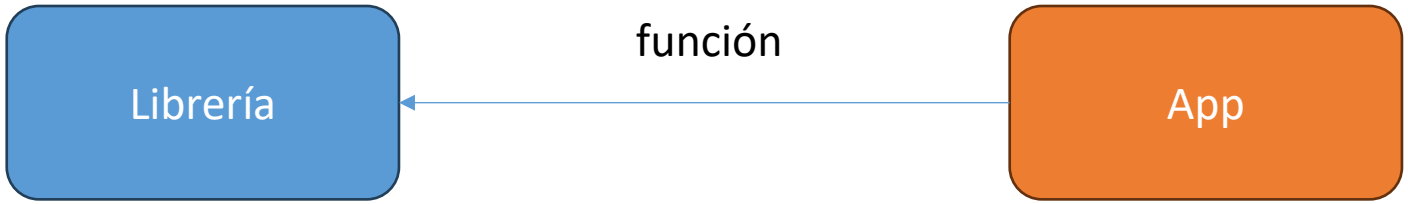
```
#include <iostream>

namespace mathUtils {
    int square(int x) {
        return x * x;
    }
}

int main() {
    std::cout << "Square of 4 is " << mathUtils::square(4) << std::endl;
    return 0;
}
```

Caso C++

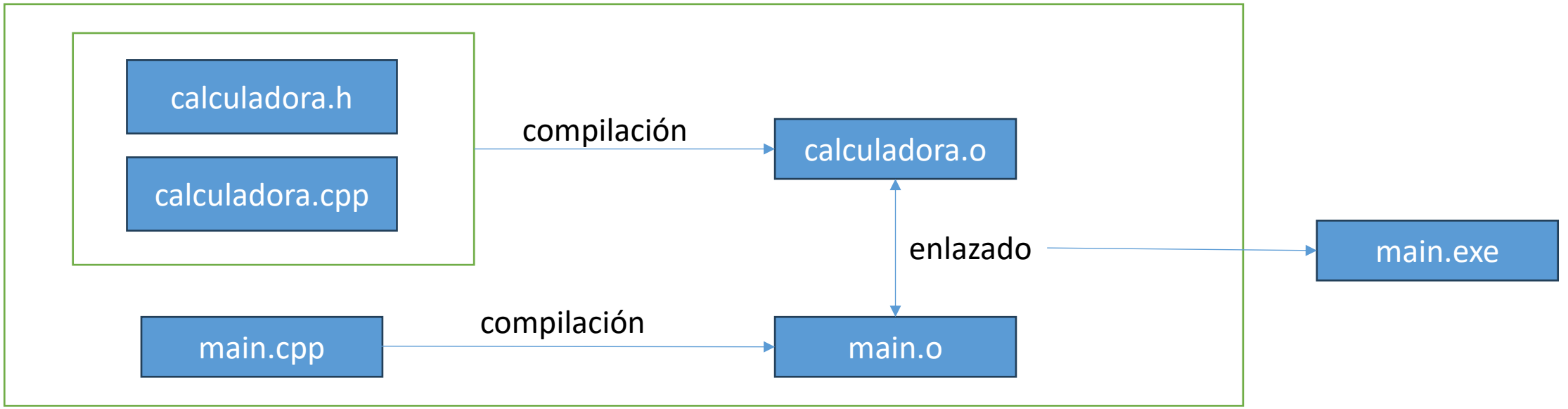
- A nivel de compilación / ejecución:
  - Unidades de Compilación
  - Librerías (estáticas, dinámicas)
    - En caso no queramos distribuir código fuente, pero si la funcionalidad que implementa.

Librería Estática

- Una librería estática es una librería cuya implementación se enlaza con la app que la usa en el momento de la compilación.



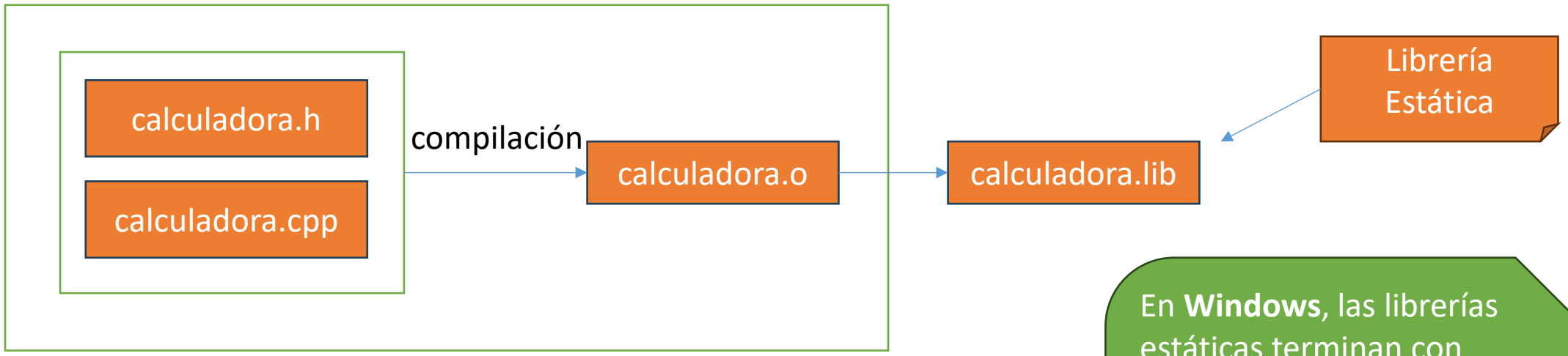
Enlazado en tiempo de compilación




```
g++ -c calculadora.cpp

g++ -c main.cpp

g++ calculadora.o main.o –o main
```

UNIVERSIDAD DE LIMA

Proyecto Librería




```
g++ -c calculadora.cpp
ar rcs calculadora.lib calculadora.o
```

Comando para crear librerías estáticas:

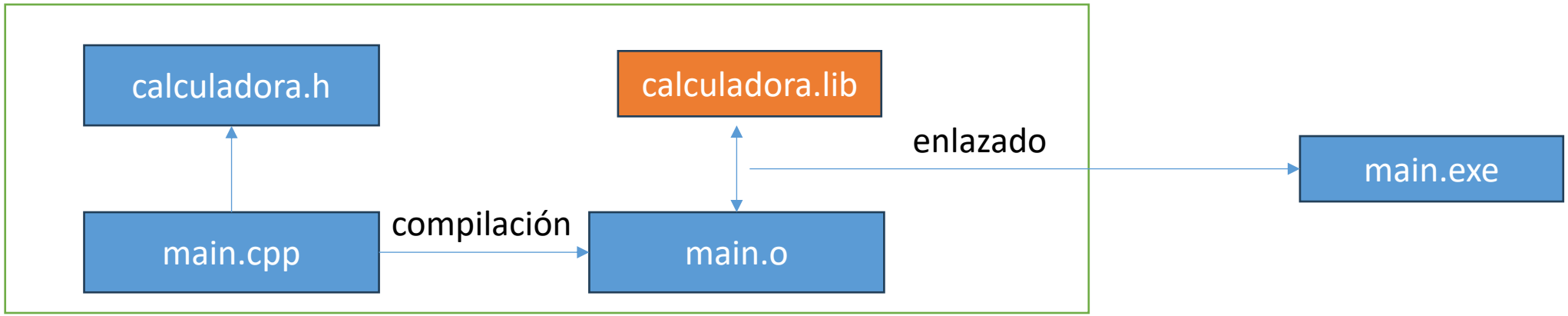
ar rcs <nombre_librería_a_crear> <unidad_compilada>

En **Windows**, las librerías estáticas terminan con extensión lib.

En **Linux y macOS**, las librerías estáticas tienen el prefijo lib y la extensión a. (para este caso, libcalculadora.a)

UNIVERSIDAD DE LIMA

App que utilizará la librería



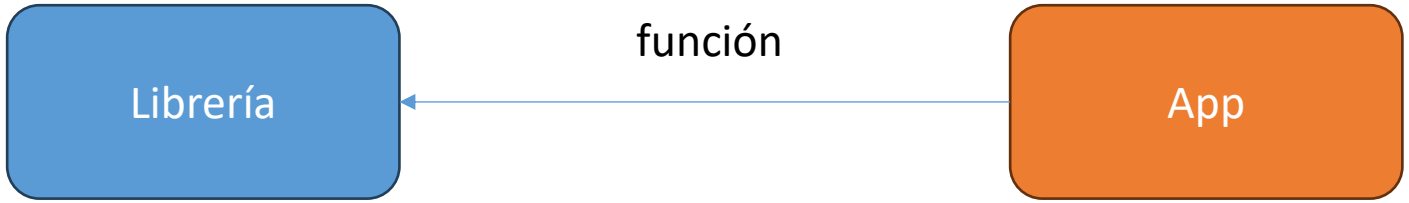

```
g++ -c main.cpp
g++ main.o -L. -lcalculadora -o main
```

Donde:

-L : indica el directorio donde se buscarán las librerías.
-l : las librerías que se quieren enlazar.

Librería Dinámica

- Una librería dinámica es una librería cuya implementación se enlaza con la app que la usa en el momento de la ejecución.
- El beneficio es que el **build** final no incluye la librería.



Enlazado en tiempo de ejecución

```
#ifndef CALCULADORA_H
#define CALCULADORA_H

#ifdef _WIN32
  #ifdef BUILD_DLL
    #define DLL_EXPORT __declspec(dllexport)
  #else
    #define DLL_EXPORT __declspec(dllimport)
  #endif
#else
  #define DLL_EXPORT
#endif

namespace Matematica {
   class DLL_EXPORT Calculadora
   {
   public:
      int sumar(int n1, int n2);
      int restar(int n1, int n2);
   };
}

#endif // CALCULADORA_H
```

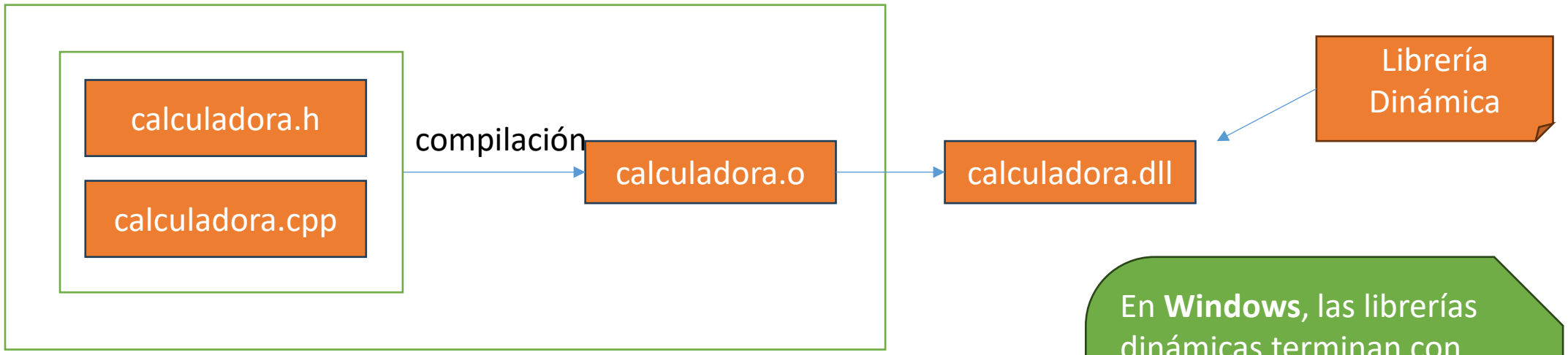
calculadora.h

```
#include "calculadora.h"

using namespace Matematica;

int Calculadora::sumar(int n1, int
n2)
{
   return n1 + n2;
}
int Calculadora::restar(int n1, int
n2)
{
   return n1 - n2;
}
```

calculadora.cpp

UNIVERSIDAD DE LIMA

Proyecto Librería



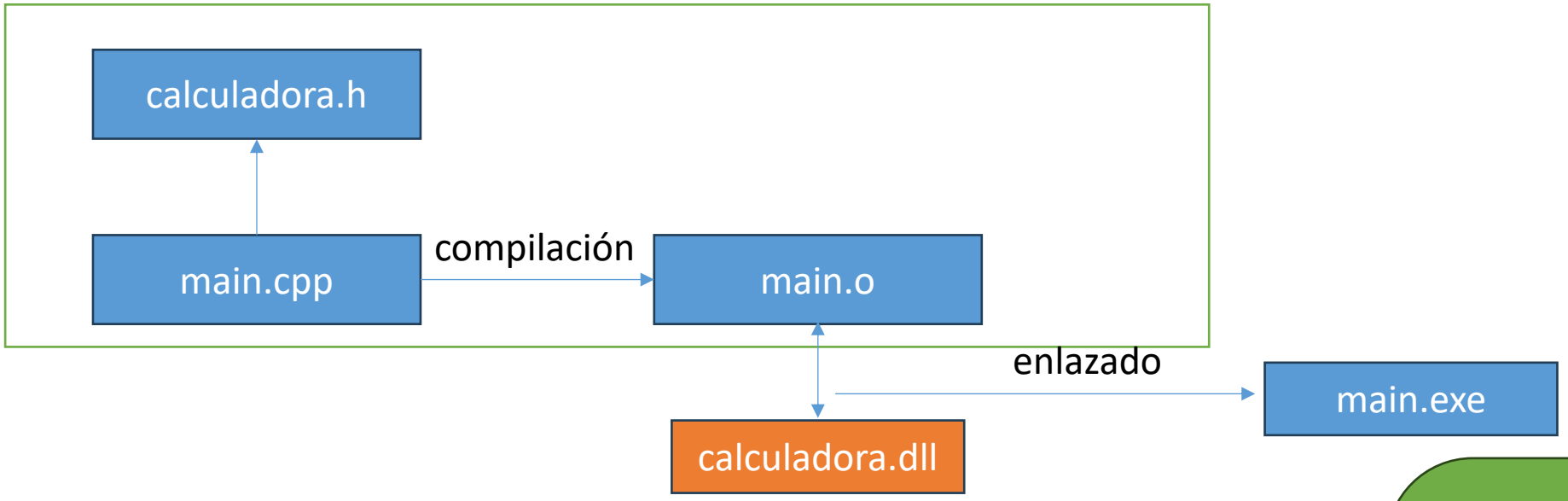

```
g++ -DBUILD_DLL -c calculadora.cpp
g++ -fPIC -shared calculadora.o -o calculadora.dll
```

En **Windows**, las librerías dinámicas terminan con extensión dll.

En **Linux y macOS**, las librerías dinámicas tienen el prefijo lib y la extensión so. (para este caso, libcalculadora.so)

UNIVERSIDAD DE LIMA

App que utilizará la librería




```
g++ -c main.cpp

g++ main.o -L. -lcalculadora -o main
```

Donde:

-L : indica el directorio donde se buscarán las librerías.
-l : las librerías que se quieren enlazar.

Para la ejecución:
En **Windows**, el .dll debe de estar en la misma carpeta del .exe o en la carpeta System32.

En **Linux y macOS**, el .so debe de estar en la misma carpeta o configurar una variable de entorno LD_LIBRARY_PATH.

Caso Java

- A nivel de código:
  - Clases
  - Paquetes (package)

```
// File: Square.java
package com.example.utils;

public class Square {
    public static int square(int number) {
        return number * number;
    }
}
```

Está en el director: com/example/utils

```
// File: MainApp.java
package com.example.app;

import com.example.utils.Square;

public class MainApp {
    public static void main(String[] args) {
        int number = 5;
        int result = Square.square(number);
        System.out.println("The square of " + number + " is " +
result);
    }
}
```

Está en el director: com/example/app

UNIVERSIDAD DE LIMA

Caso Java

- A nivel de ejecutables:
  - Librerías JAVA (archivos .jar)

```
$ javac com/example/utils/Square.java com/example/app/MainApp.java

$ jar cvfe MyApp.jar com.example.app.MainApp -C com .

$ java -jar MyApp.jar
```