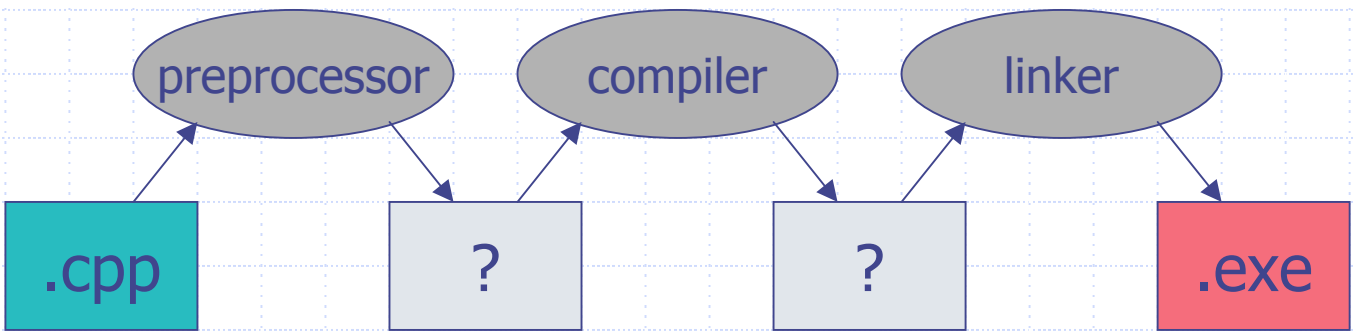
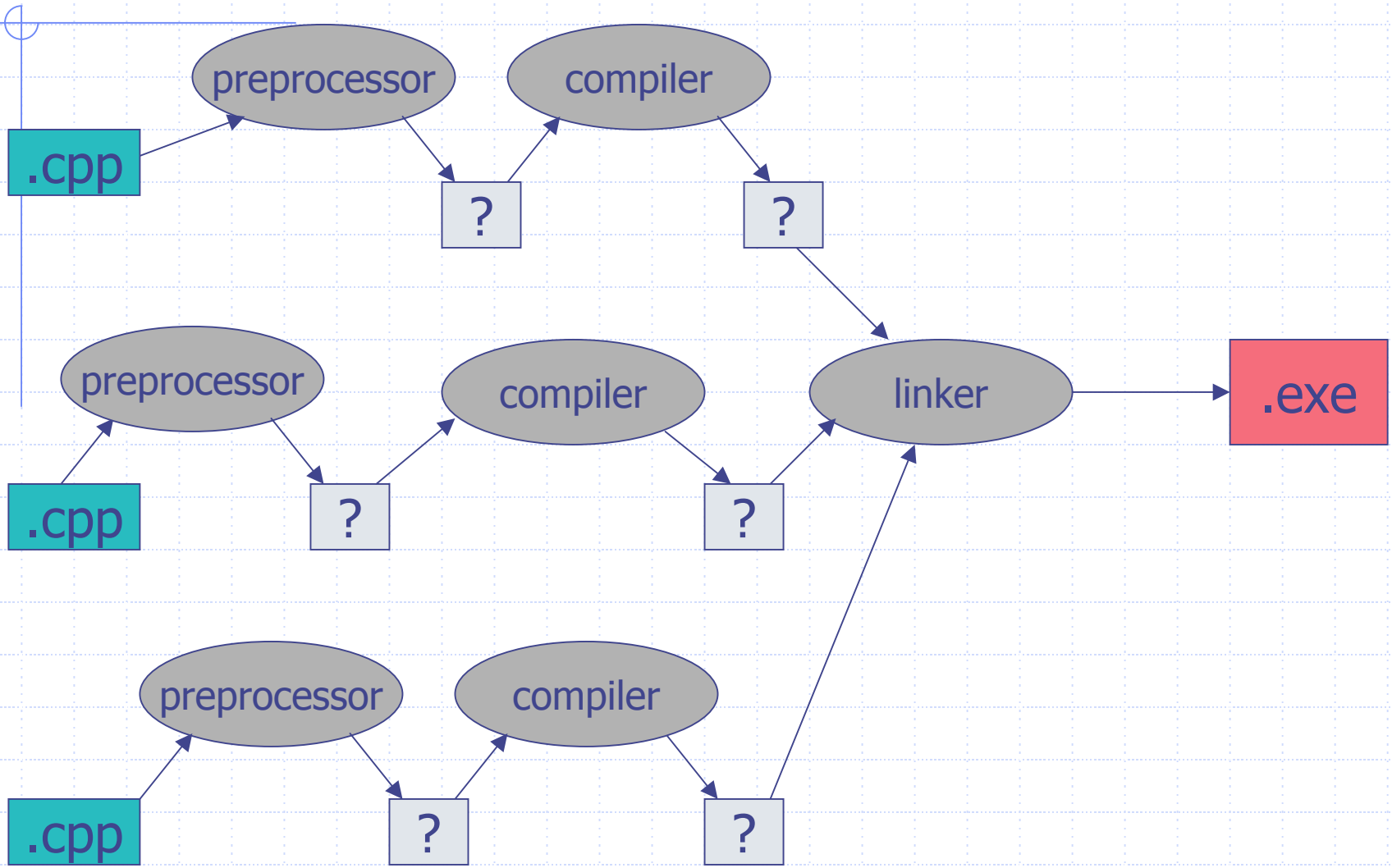
Трансляция

Стадии трансляции:

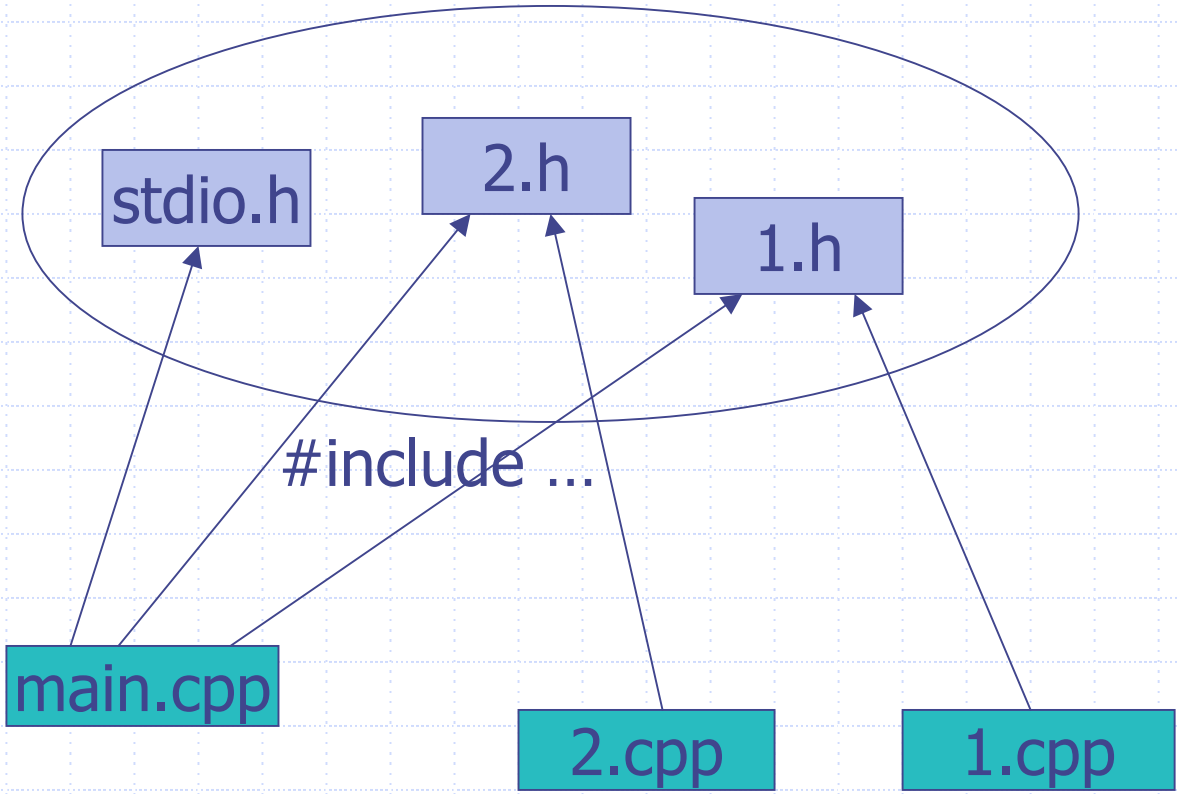
1. Обработка препроцессором
2. Компиляция
3. Компоновка



Трансляция


preprocessor
compiler
.cpp
?
?
preprocessor
compiler
linker
.exe
.cpp
?
?
preprocessor
compiler
.cpp
?
?

Заголовочные файлы



Пример заголовочного файла

```
//list.h
#ifndef _LIST_H_
#define _LIST_H_

typedef double ListData;
struct List{
        ...
};

List * prev (List const * node);
List * insertAfter (List * element, ListData value);

#endif // _LIST_H_
```

Заголовочные файлы

Содержат:

1. Описания типов
2. Описания функций (прототипы)
3. Описания глобальных переменных (???)

Глобальные переменные в заголовочных файлах

```
//test.cpp
#include “test.h”

int value = 10;
```

```
//test.h
#ifndef __TEST_H_
#define __TEST_H_

//int value; - ошибка

extern int value;

#endif
```

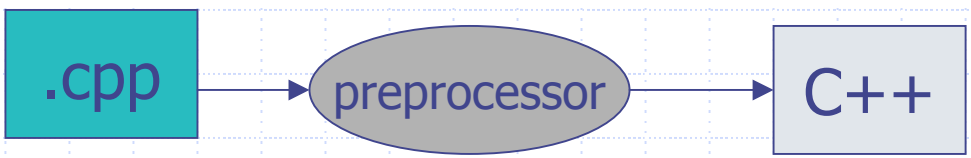
```
//main.cpp
#include “test.h”

int main (void)
{
        print (“%d\n”, value);
}
```

Препроцессор

Обрабатывает директивы препроцессора (#include, #define, #ifndef…)

Результатом работы является программа на языке C++, но без директив препроцессора



Директива #include

#include <имя_файла>
#include “имя_файла”

Директива заменяется текстом указанного файла независимо от его содержимого. Операция рекурсивна.

В первом случае файл ищется в стандартных путях для заголовочных файлов.

Во втором случае файл ищется сначала в текущей директории, затем в стандартных путях.

Директива #define

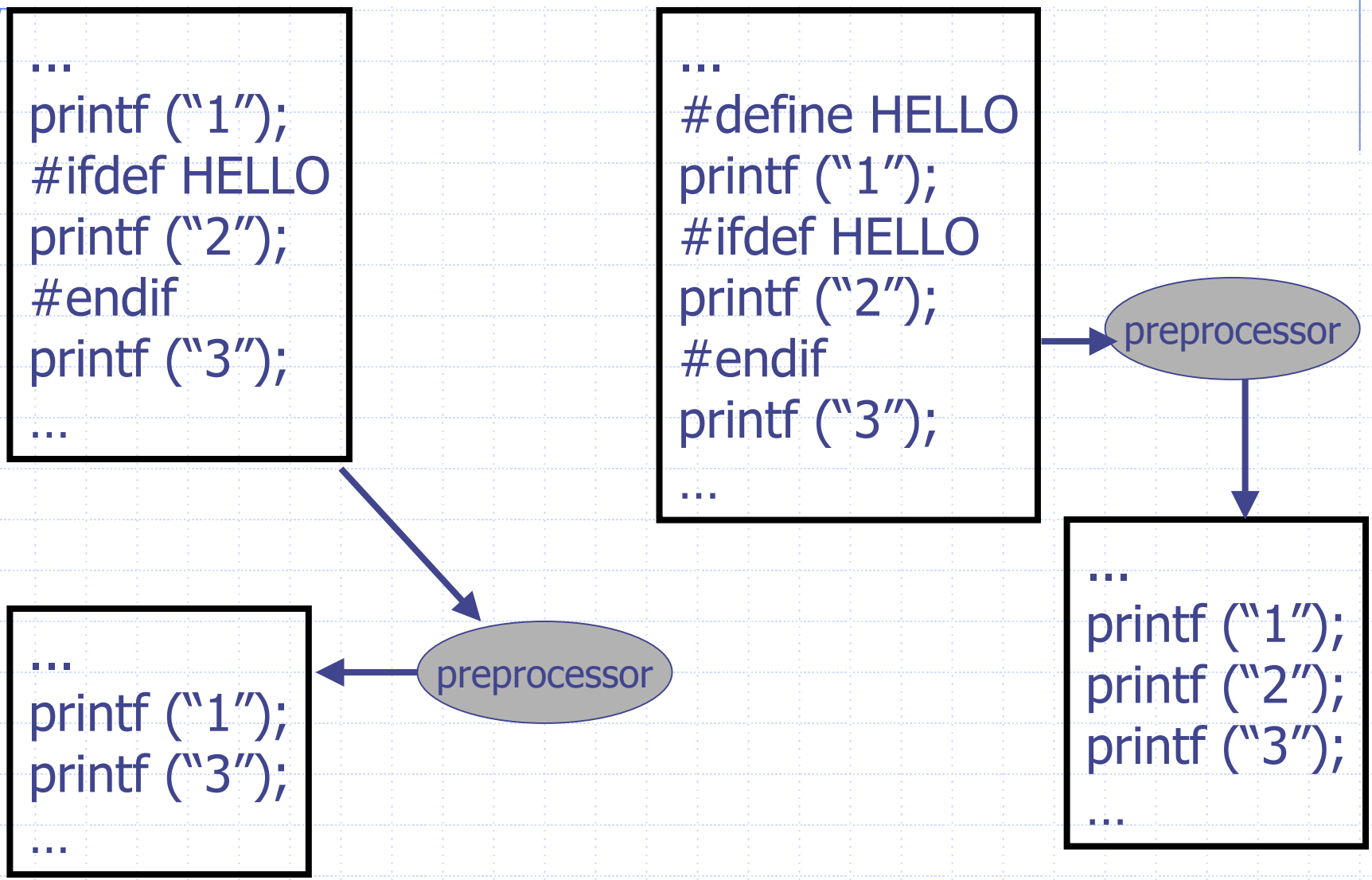
#define имя значение

Определяет «переменную» препроцессора

```
…
#define VAL 1
#define VAL 2
printf (“%d”, VAL);
#undef VAL
printf (“%d”, VAL);
#undef VAL
//printf (“%d”, VAL);
…
```


```
/*после обработки
препроцессором*/
…
printf (“%d”, 1);
printf (“%d”, 2);
//printf (“%d”, VAL);
…
```

Условные директивы препроцессора



Условные директивы препроцессора

```
#ifndef имя          //блок остается, если имя
                     //не определено

#ifdef имя           //блок остается, если имя
                     //определено

#endif               //конец блока
```

Условные директивы в заголовочных файлах

```
//1.h
#ifndef __1_H_
#define __1_H_

int doSomething1 (void);

#endif //__1_H_
```

```
//2.h
#ifndef __2_H_
#define __2_H_

#include "1.h"

int doSomething2 (void);

#endif //__2_H_
```

```
//main.cpp
#include "1.h"
#include "2.h"
…
```

Компиляция

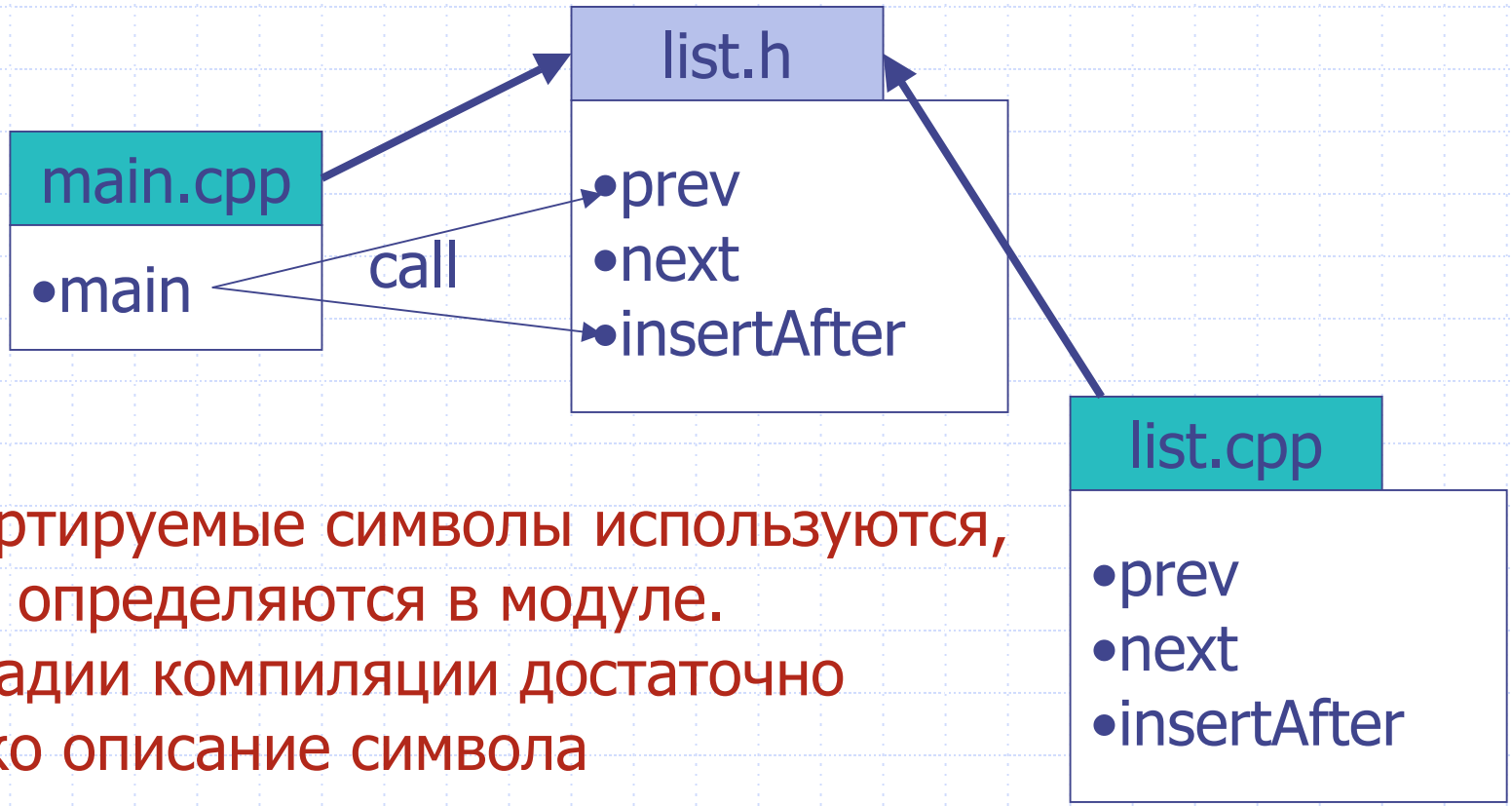
Переводит текст на языке C++ в объектный модуль

объектный модуль содержит в себе символы, через которые происходит связь между модулями
Символы бывают:

1. Экспортируемые (определенные)
2. Импортируемые (неопределенные, но используемые)

Экспортируемые и импортируемые символы

Экспортируемые символы определяются в модуле и могут использоваться в других модулях



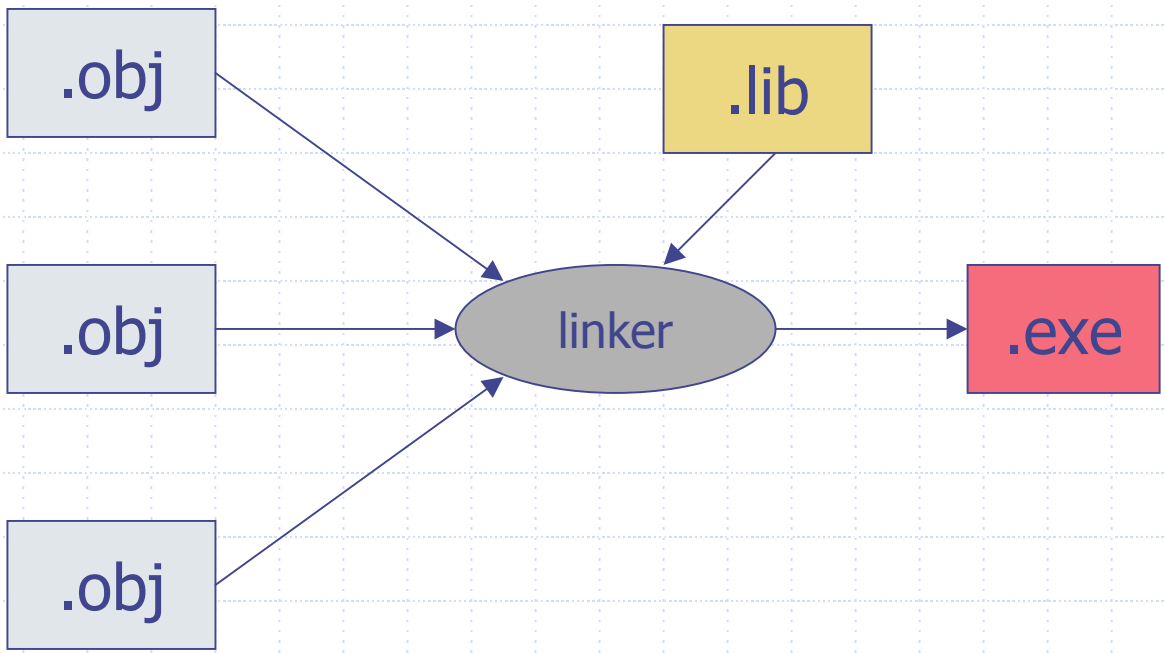
Импортируемые символы используются, но не определяются в модуле.
На стадии компиляции достаточно
Только описание символа

Символы в объектном модуле

```
00000118    T element
00000000    T insertAfter
000001c0    T main
000000e8    T next
00000100    T prev
0000014c    T printList
            U printf
0000006c    T removeElement
000000c0    T removeList
```

Компоновка

Компоновщик собирает все объектные модули и библиотеки в исполняемый файл



Алгоритм компоновки

1. Ищем модуль с символом main
2. Извлекаем из все импортируемые (внешние) символы
3. Для всех внешних символов ищем модули, где эти символы экспортируются
4. Повторяем с пункта 2

Статические библиотеки

Компоновщик разворачивает библиотеку в набор объектных модулей, их которых она собрана