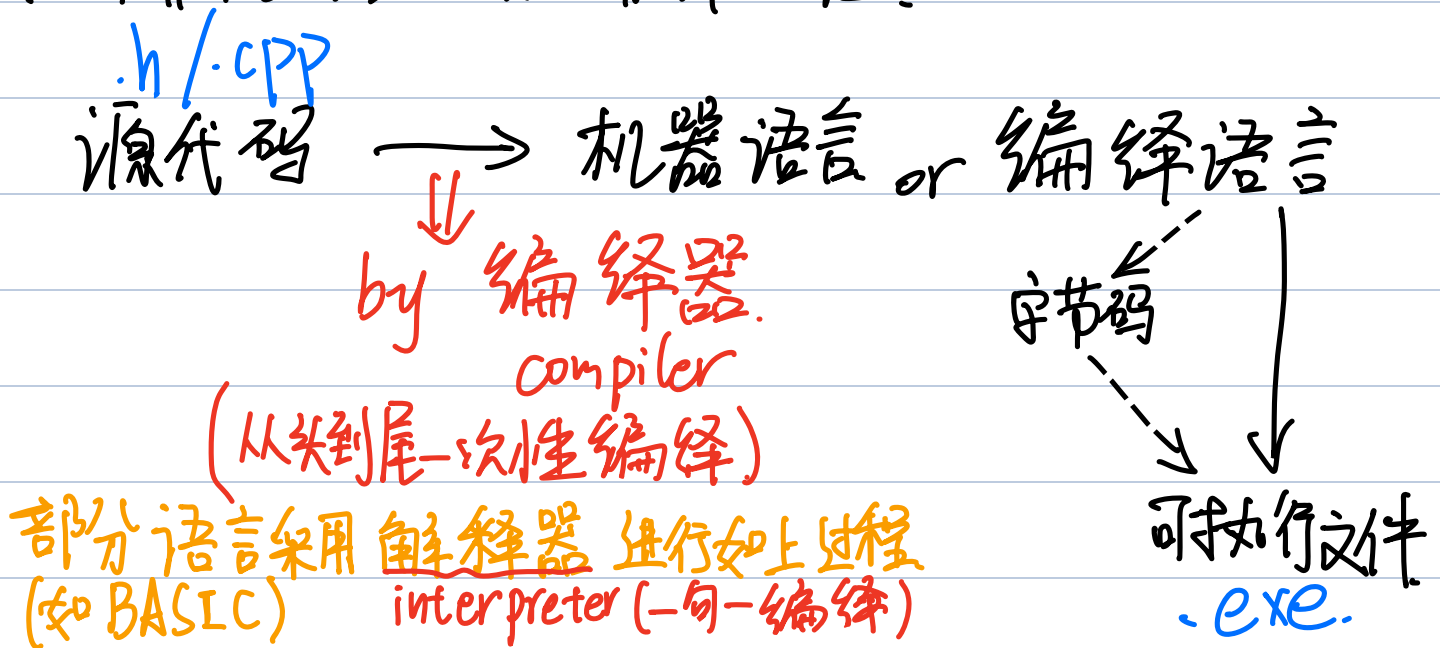


第一节 类 & 对象 引入

- object**
1. 对象 — 实体, 有其值
属性 — 实体的限制、性质 or 特点

* 对象的属性可能也含有对象 → 类的嵌套

2. 计算机语言的编译过程



3. 分离式编译

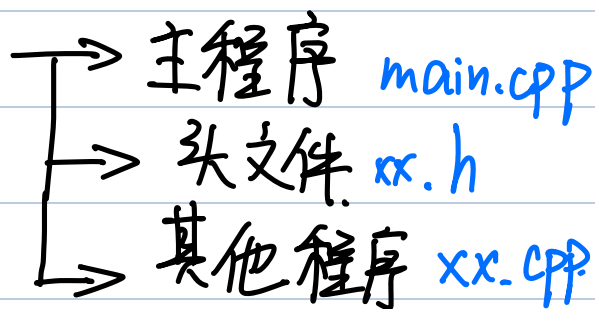
即一个程序中, 各函数将分别进行编译.

最后 一同发挥效用. ↘

执行过程还是自上而下的

4. 头文件

一个项目应包含三个部分
project



头文件中应包含该项目中使用的外部函数并在编译过程中先行编译。

5. 语法 —— 流输入/输出

在 *iostream* 头文件中

**endl*
为 *endl* 换行符

流输入 *cin* 使用流提取符 *>>* 获得输入
都为预定义对象 \rightarrow 定义在 *istream* 类内 \downarrow 标志流的对象
流输出 *cout* 使用流插入符 *<<* 进行输出 \uparrow

6. 语法 —— 字符串类

在 *string* 头文件中

可以直接定义变量为字符串类型 *string*
并进行字符(串)间的基本操作

7. 语法 —— 文件读写流

在 `fstream` 头文件中. 且以 `iostream` 为依赖

对文件进行读操作 `ifstream` 类

通用类 `fstream` 类

对文件进行写操作 `ofstream` 类

成员函数:

① `open (const char* 路径, ios::openmode 打开模式)`



a. `ios::app` 追加写入

b. `ios::ate` 打开后定位到文件末尾

c. `ios::in` 只读模式

d. `ios::out` 只写模式

e. `ios::trunc` 打开文件前先截断内容.

* 打开模式可以组合如:

```
fstream fin; fin.open("debug.txt", ios::in | ios::out)
```

以读写模式打开

8. 语法 — vector 容器 → 类模板

vector 是一个具有动态长度的容器, 其中所容纳的数据类型由用户自行定义 可理解为变长数组

vector <类型> 容器名

使用时要

调用 vector 头文件

对其对象进行访问时, 可利用类似数组下标的方式
即 容器名 [下标]

成员函数:

1. size() 大小

2. push_back() 后插入

3. pop_back() 后删除

4. clear() 清除

5. empty() 判空

6. swap() 交换两元素

7. back() 返回最后一个元素

8. front() 返回第一个元素

9. getline 函数

在 string 头文件中, getline 定义为:

istream& getline (istream& 流对象, string& 存储数组, char 分隔符)

实现将流对象按行 '/n' 或按分隔符读入并存入存储数组中。

↓
可以为 istream 类或 fstream 类

在 istream 头文件中, getline 为类的成员函数

↓
以输入流对象调用

getline (string &存储数组, char 分隔符)

↓
默认为'\n'

功能同上。