

面向对象程序设计概述

北京理工大学计算机学院
金旭亮

(使用C#和.NET)

为什么要学习“面向对象程序设计”？

1

面向对象是一种主流的软件开发与设计方法

2

当前人们所使用的各种软件系统，基本上都是应用面向对象的思想与技术设计和开发出来的。

我需要了解哪些背景知识？

在开始学习面向对象程序设计之前，我们需要对软件开发这件事情有一个总体上的了解.....

软件是怎么写出来的？

面临的问题

你需要作出的决策

解决之道

1 如何表达信息以方便计算机处理？

设计和选用合适的数据结构

2 如何设计与选择特定的算法处理信息？

可以选择已有的算法
也可以设计新的算法

算法：分而治之，动态规划……
程序控制结构：分支、循环、递归

3 如何编写程序实现算法？

选择合适的技术手段
设计合理的技术方案

软件系统架构，面向对象的分析与设计，编程语言，开发框架，软件平台……



合抱之木，生于毫末；
九层之台，起于垒土；
千里之行，始于足下。

——《道德经》

写出一个有用的程序，需要掌握哪些基础知识？

你需要知道.....

程序是如何被
计算机执行的？

各种信息是如何用
0-1表达的？

计算机是由哪些元
件组成的？它的工作
原理是什么？

操作系统如何启动
一个程序的执行？

如何编写计算
机可以执行的
程序？

第一步：
编写源代码

第二步：
编译

第三步：
得到可执行的程序

怎样构造求解
问题的算法？

分析问题

设计算法

编写程序



1 程序是如何被计算机执行的?

神奇的“0”和“1”

人类使用从0~9构成的数字来计算，
但计算机只认识两个数字：

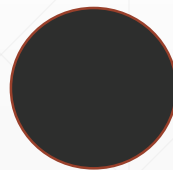
只有两个状态的东西容易找到和制作：比如开和关，高和低，黑和白，有和无.....

在计算机中，我们可以让高电平表示“1”，低电平表示“0”，因为仅仅只需要区分开两个状态，计算机元件的制造难度可以大大下降。

如果使用十进制，则计算机必须使用一种能区分10种状态的电子元器件，这就太麻烦了.....

1

0



用0和1怎样表达信息？

数值信息，直接使用二进制表达

比如：“10”这个十进制数，就可以被表示为“1010”这样的二进制数

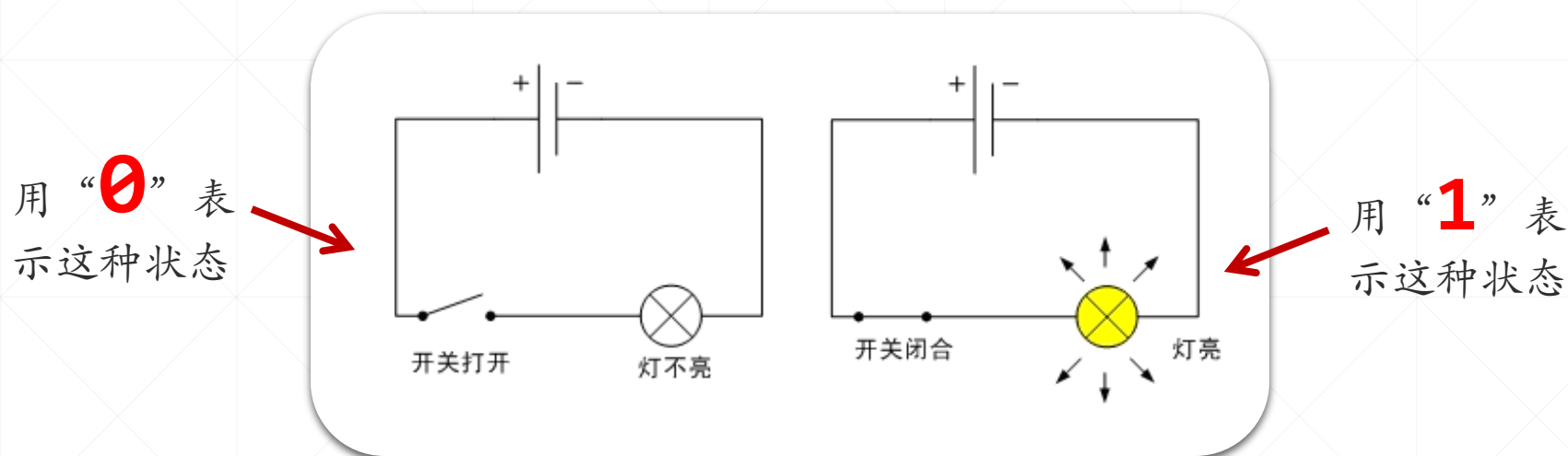
非数值信息，采用“编码”的手段

“**编码**”就是以若干位数码（或符号）的不同组合来表示非数值类型的信息，它人为地为数码(或符号)的每一种组合指定了一种特定的含义。

比如：

- (1) 五笔字型汉字输入法，用一到四个字符对应一个汉字或多个汉字；
- (2) 对于数码图像，可以将图像分解为像素，用数字表达它的颜色和位置等信息，然后把这一大堆数字以特定的格式写到文件中，从而就得到了各种各样的图像文件

计算机能直接执行由0和1构成的机器指令



“0101” = 关灯 (0) → 开灯 (1) → 关灯 (0) → 开灯 (1)

以“0”和“1”表达的指令，可以由计算机硬件直接执行，因此，将它们称为“**机器指令**”。

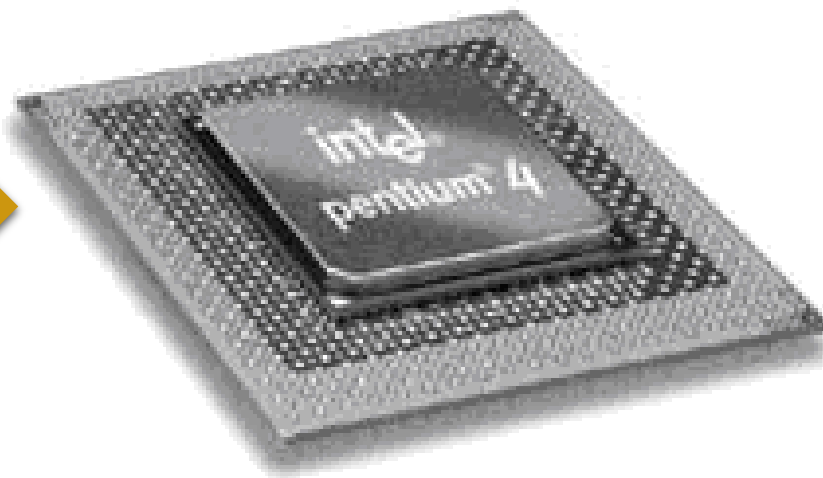
不同体系结构的计算机硬件，能执行的机器指令是不一样的，每种机器能执行的所有机器指令，称为这种机器的“**指令集**”。

计算机的大脑——CPU

CPU可以执行机器指令

机器指令被传送给CPU执行

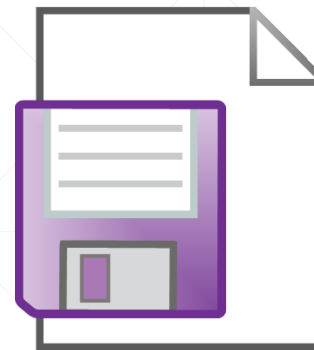
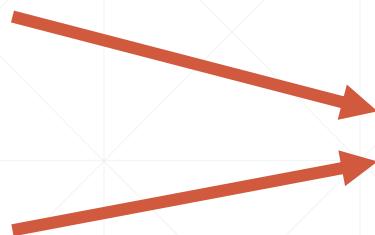
011011.....



指令与数据通常保存在文件中

指令 01101111.....

数据 01101111.....



可执行程序文件
(.exe)

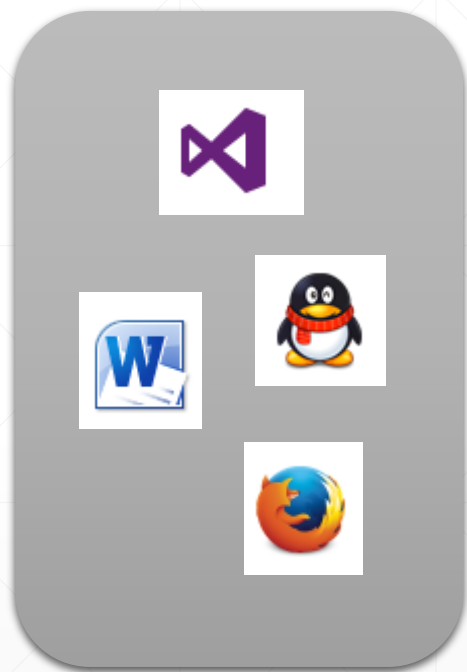
Windows平台下，可以执行的文件通常是以.exe结尾的。另有一种扩展名为.dll的文件（称为“**动态链接库**”），它需要被.exe装入后才能执行。



装载



外部存储器



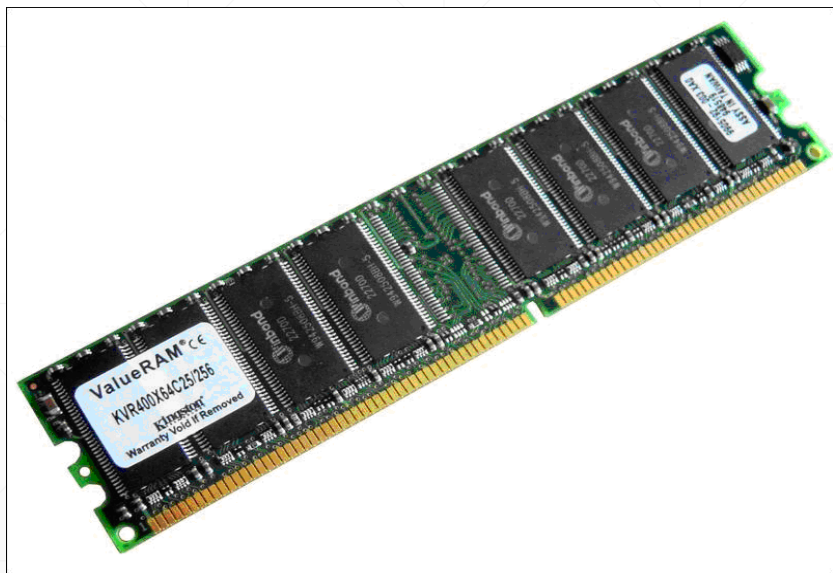
各种程序平时保存在
外部存储器中



PC上，硬盘是当前用得最多的外部存储器，除此之外，光盘和U盘也是非常常见的外部存储器。

Memory (内存) 与内存条

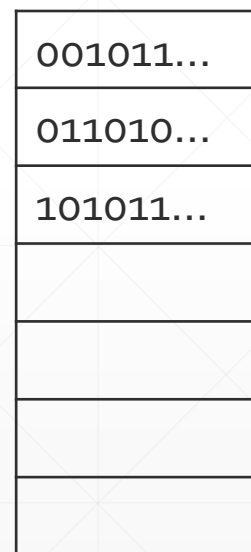
放在外部存储器中的程序并不能直接运行，需要被装入内存后，才能执行。



内存条

内存条中的存储芯片包容许多**存储单元**，用于保存程序指令和数据。

存储单元的集合，称为“**内存**”



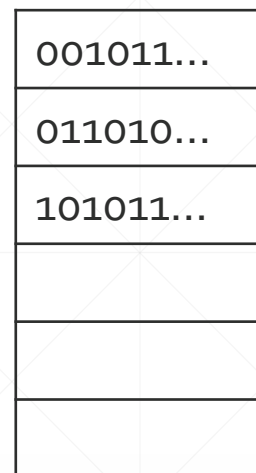
指令与
数据

内存

CPU负责执行程序



读取和写入



指令与数据

CPU能直接存取放在内存存储单元中的指令和数据，并负责执行这些指令。

在资源管理器中双击.exe文件，操作系统读取文件，把它所包容的指令和数据装载到内存中



程序 (.exe文件)
保存在这里.....



当我们购买电脑时，经常被告之：“这台电脑配置了**8G**的内存，**1T (1024G)**的硬盘.....”

内存	
内存容量	8GB
内存类型	DDR3L
插槽数量	2 x SO-DIMM
最大支持容量	16GB
硬盘	
硬盘容量	1T
转速	5400转/分钟
接口类型	SATA 串行

京东所售某电脑之参数信息

这是不是说：

我们编写的程序，最多只能使用**8G**的存储空间？

区分两种不同的内存类型

物理内存

单台计算机上安装的物理存储芯片所提供的内存。

虚拟内存

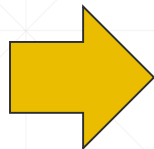
由操作系统所提供的“虚拟内存”，可以比物理内存大很多，“多出来”的内存，操作系统从硬盘上划出一块空间来弥补。

C#写的.NET程序，无法直接访问物理内存上的特定存储单元，它所访问的是由操作系统（比如Windows）负责提供的“**虚拟内存**”。

程序的开发与运行过程简述



软件工程师用各种编程语言编写计算机程序



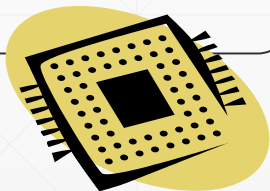
编译器把源程序转换为二进制指令与数据，以文件的方式保存在外部存储器中



操作系统将程序文件从外部存储器中读入到内存中

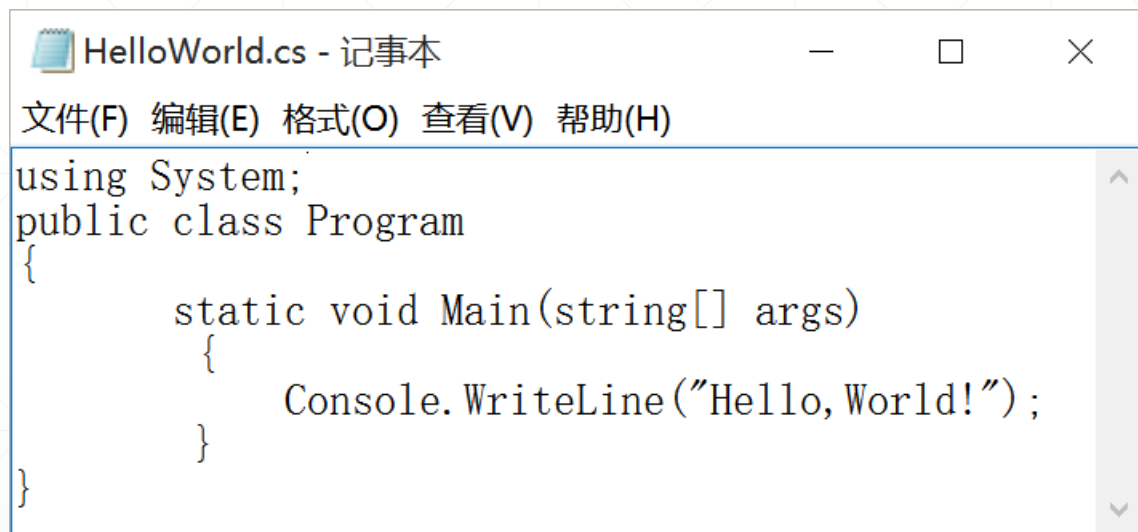


CPU从内存中取出指令执行



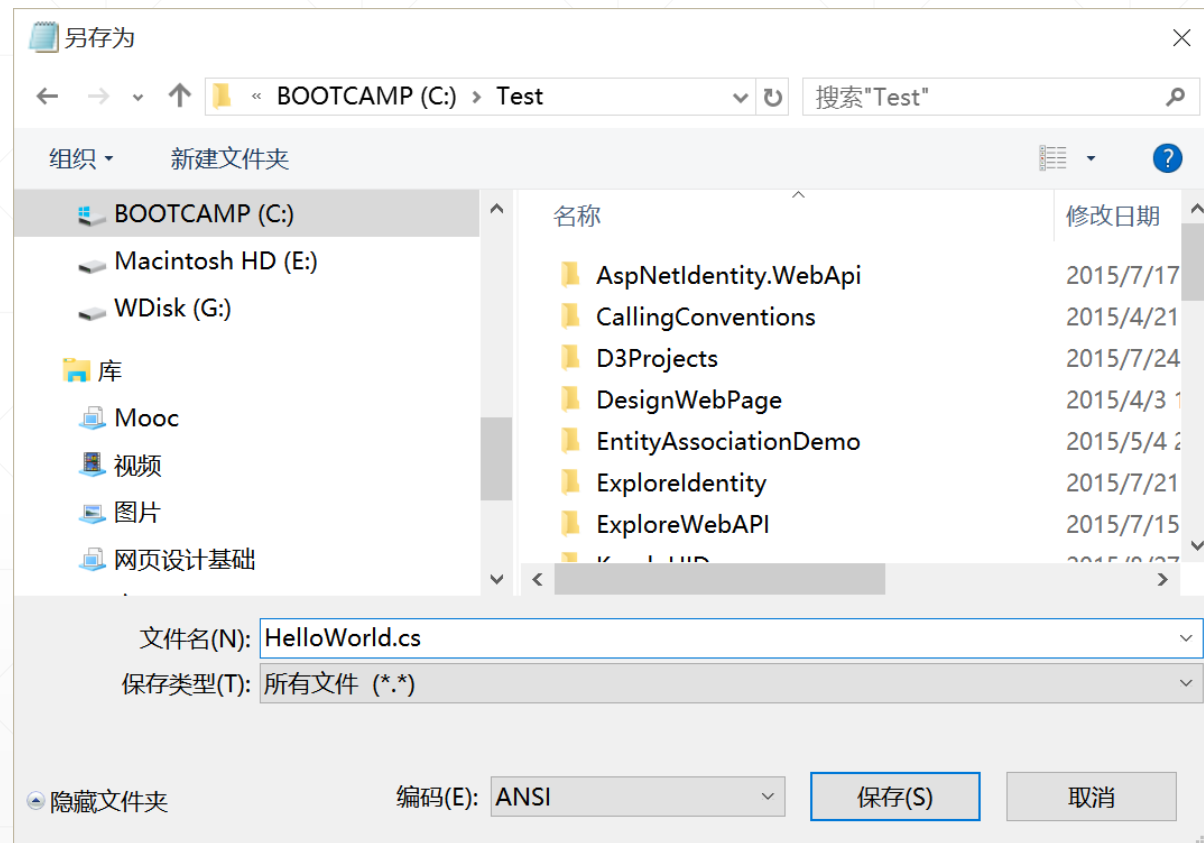
2 如何编写计算机可以执行的程序？

手写的第一个.NET程序——Hello , World !

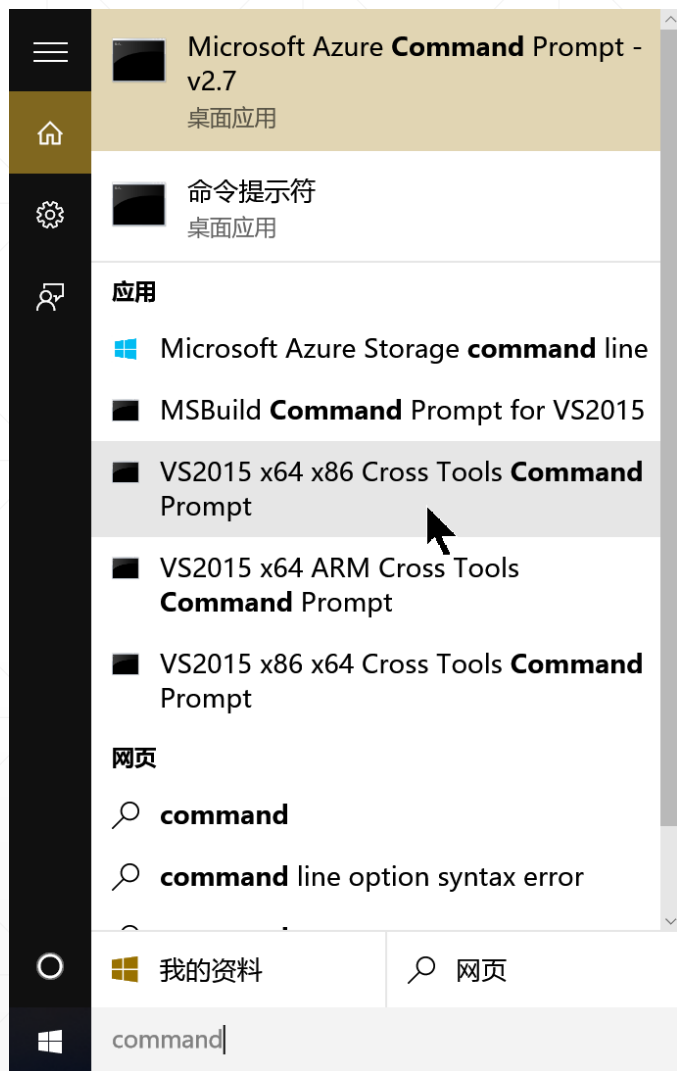


```
using System;
public class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine("Hello, World!");
    }
}
```

用记事本编写以上代码，注意“大小写”



另存为HelloWorld.cs.....



选择Visual Studio 2015所提供的命令提示符命令

```
VS2015 x64 x86 Cross Tools Command Prompt

C:\Program Files (x86)\Microsoft Visual Studio 14.0\VC>cd \test

C:\Test>dir *.cs
驱动器 C 中的卷是 BOOTCAMP
卷的序列号是 E621-D935

C:\Test 的目录

2015/08/28 09:22                151 HelloWorld.cs
                1 个文件                151 字节
                0 个目录    9,757,093,888 可用字节

C:\Test>csc.exe /target:exe HelloWorld.cs
Microsoft (R) Visual C# Compiler version 1.0.0.50618
Copyright (C) Microsoft Corporation. All rights reserved.

C:\Test>HelloWorld
Hello, World!

C:\Test>
```

对源码进行编译

运行生成的HelloWorld程序

进行试验

[illegible]

重大发现!

HelloWorld.cs的内容与HelloWorld.exe
的内容是不一样的.....

“编译”是怎么回事？



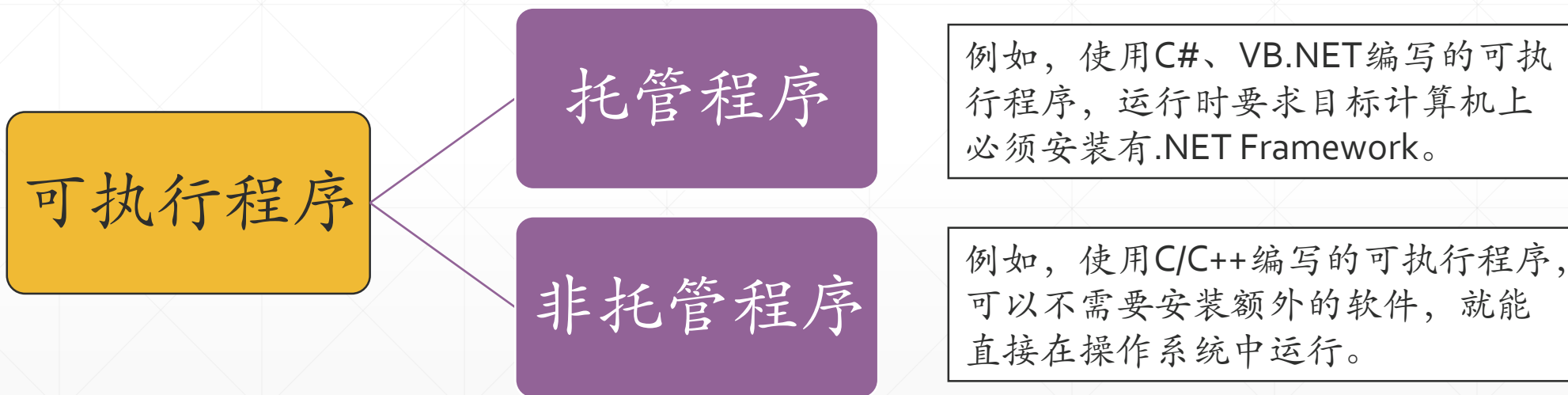
```
using System;
public class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine("Hello,World!");
    }
}
```

0110110111.....

“**编译 (compile)**”，有点类似于“翻译”，粗略地说，它负责把人编写的源代码“翻译”为计算机可以识别并执行的二进制代码。

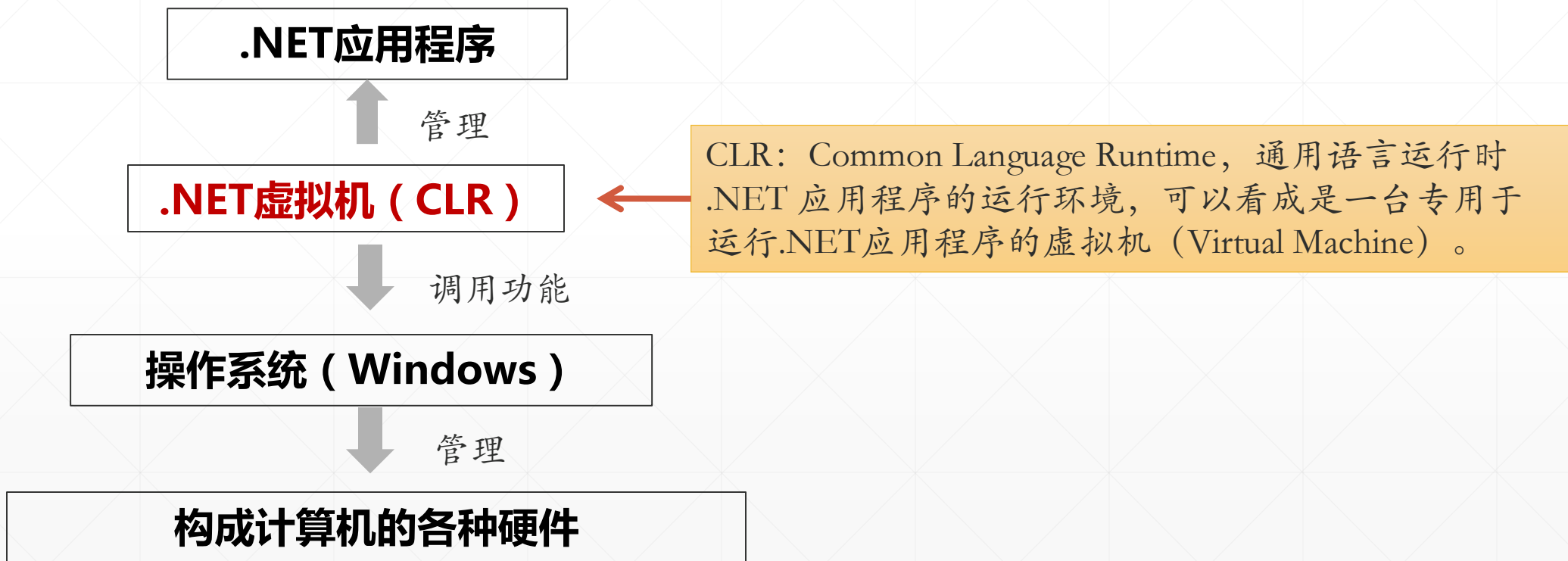
可执行程序也是分类的

可执行程序文件也分多种类型，在Windows平台下，可分为：



“托管的” 应用程序是什么意思？

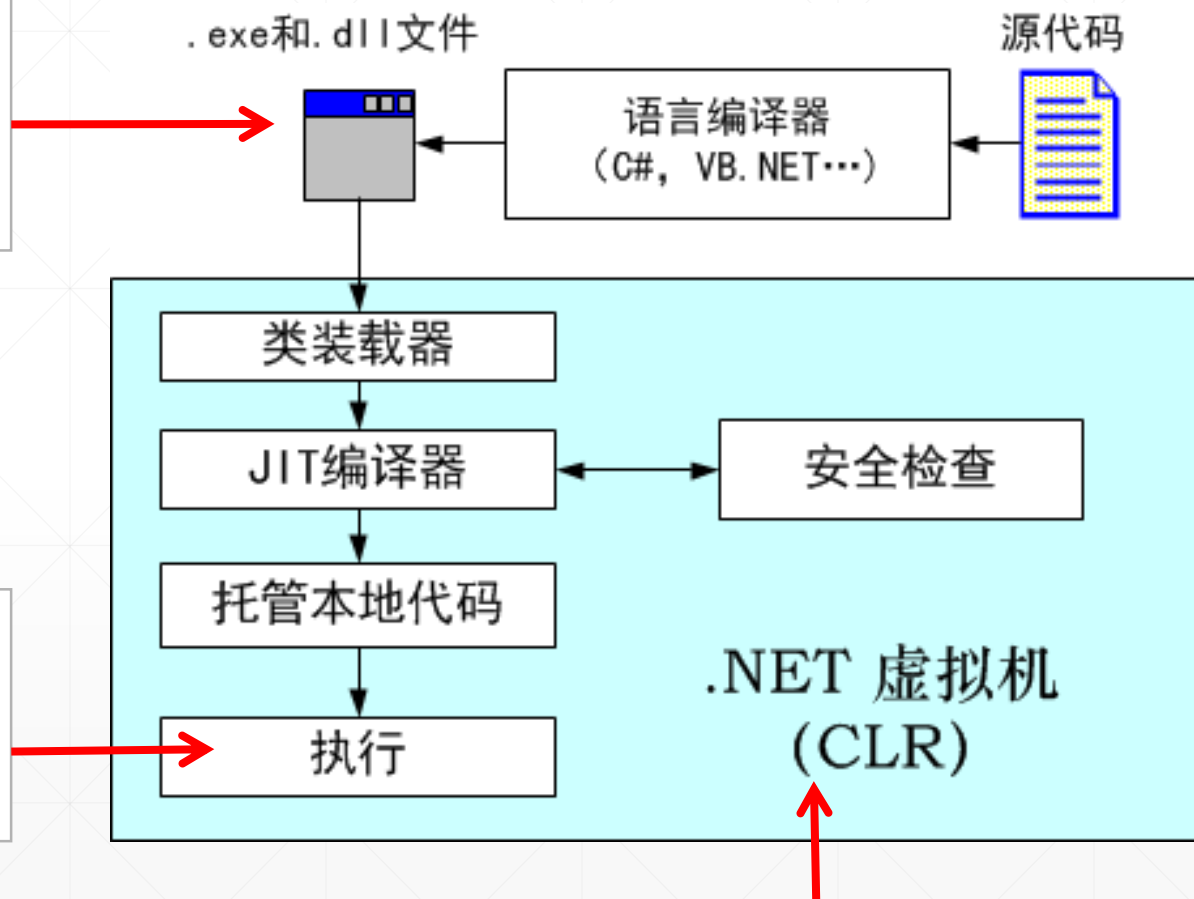
.NET应用程序是“**托管 (Managed)**”的，意思是它必须在一个独立的运行环境（即CLR）中运行，并受到这个运行环境的管理与控制。



.NET程序的开发与运行“全景图”

.exe和.dll在.NET中统称为“**程序集 (Assembly)**”，其中保存的是“**IL (中间语言)**”指令。

真正被CPU执行的还是机器指令，它是由JIT编译器在程序运行时动态地由IL指令翻译而成，称为“**本地代码**”。



每个.NET应用程序运行时，操作系统会创建一个CLR实例，最终由CLR负责装入.NET应用程序并执行之。

3 怎样构造求解问题的算法？

什么是“算法（ algorithm ）”？

计算机中的**算法**，主要指为了解决某个问题而设计的一种解决方案，包容一系列计算机能够执行的有着特定顺序的命令，计算机执行完这些命令，将得到某种结果，意味着某个问题已经有了一个结论。

算法的针对性很强，专用于解决特定的问题。

算法的设计，通常与数学有着很密切的联系，并且是**独立于**特定的编程语言和软件平台的。这就是说：

可以使用多种编程语言，以多种方式，在不同的平台上实现同一个算法。

让我们从一道算术题开始.....

请同学们手工计算出1999年5月10日到2006年3月8日期间一共有多少天？

答案是：一共有2494天，你算对了吗？

请仔细回想一下.....

你自己是如何解决上述日期计算问题的？
用到了哪些背景知识？
经历了哪些具体的计算步骤？

小学时我们就学过.....

一年有365天，但闰年有366天；

一年有12个月，分为大月和小月，大月31天，小月30天；

2月最特殊，普通年（平年）有28天，闰年有29天。

将整个“日期计算”的任务进行分解.....

(1) 1999到2006期间有多少个整年？

6个整年，2个闰年： $6 \times 365 + 2 = 2192$ 天

(2) 1999年5月10日到年底有多少天？

4个大月，3个小月： $4 \times 31 + 3 \times 30 = 214$ 天

5月10日到本月底还有 $31 - 10 = 21$ 天

所以：一共有 $214 + 21 = 235$ 天

(3) 2006年元旦到2006年3月8日有多少天？

$31 + 28 + 8 = 67$ 天

结论：一共有 $2192 + 235 + 67 = 2494$ 天。

我们为什么要花时间编程来干“日期计算”这件事？

日期计算这个问题，有必要让计算机来处理吗？

- 我们经常需要计算两个日期之间间隔的天数；
- 有些间隔很长的日期，要计算起来工作量比较大，比如要计算出公元前221年3月9日到公元后9876年4月3日之间有多少天，人工计算就很麻烦了，数字比较大，计算量不小；
- 有关日期的计算有多种，计算两个日期之间的天数仅是其中一种，如果我们能把常用的日期计算工作都让计算机来完成，无疑是件“一劳永逸”的事情。

决定了要用计算机来解决日期计算的问题，现在的新问题是：

如何编程实现？

从“结构化”到“面向对象”

如何编程解决“日期计算”的问题？

程序设计可以看成是一种“抽象”的艺术.....

使用“抽象”的思维方式，构造软件系统的顶层模型



尼克劳斯·沃斯 (Niklaus Wirth) 教授，Pascal 系列语言之父，提出“**结构化程序设计**”这一革命性概念。

程序 = 数据结构 + 算法

数据结构——对数据进行抽象

先确定一种数据结构，然后基于此数据结构设计算法

```
struct MyDate {  
    public int Year ;    //年  
    public int Month ;  //月  
    public int Day ;    //日  
}
```

MyDate这个数据结构，封装了“年”、“月”、“日”三个基本信息

在程序设计中，依据要解决的特定问题，分析它所涉及的相关数据和其中所蕴含的各种信息，按照特定编程语言所支持的语法特性，将它们转换为特定的数据结构，往往是整个开发中至关重要的一步。

基于数据结构确定算法

将人的计算方法转为计算机算法：

- (1) 计算出两个日期之间的整年天数
- (2) 计算出两个日期之间的整月天数（去掉中间的整年）

每个算法步骤用一个函数来实现：

→ CalculateDaysBetweenTwoYear()

→ CalculateDaysBetweenTwoMonth()

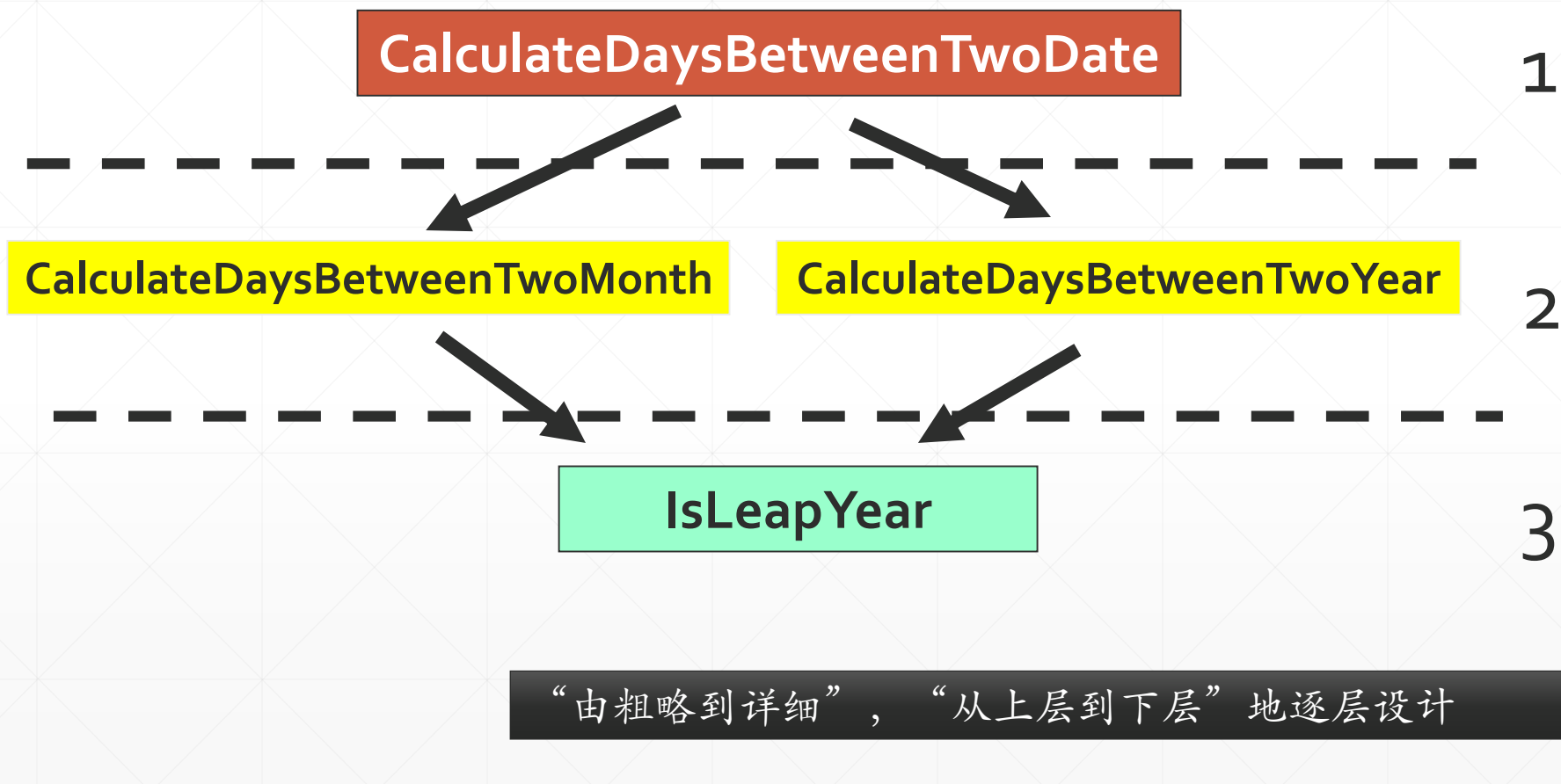
算法就是一系列的命令，计算机通过执行这些命令，完成特定的数据处理工作。

进一步细化与调整设计方案

需要判断是否是闰年，所以应该添加一个**isLeapYear()**函数。

再添加一个顶层主控函数**CalculateDaysBetweenTwoDate()**，将前面设计得到的函数“装配”起来，从而实现整个算法。

最终的技术设计方案



问： 如何确定开发顺序？

代码实例： CalculateDaysForSP

确定开发顺序的基本方法

具体开发时，“从下层到上层”地逐层开发，就象盖楼一样……

开发被别人调用，自己不调用别人的函数

IsLeapYear()

开发中间层函数，它需要调用底层已经实现好的函数

CalculateDaysBetweenTwoMonth()
CalculateDaysBetweenTwoYear()

开发顶层函数，它需要调用中间层已经实现好的函数，通常情况下，避免跨层调用。

CalculateDaysBetweenTwoDate()

将“结构化”代码转换为“面向对象”的代码

转为面向对象实现

给函数搬一个新家---类

类

字段

方法

示例项目：CalculateDaysForOO

重大变化：

(1) DateCalculator这个类的**职责很明确**，它负责“计算日期”，除此之外，它什么也不干

(2) 外界只能“看到”并调用它所定义的唯一一个“**公有 (public)**”方法 CalculateDaysOfTwoDate()，根本就不知道它内部到底是怎么计算出来的。

面向对象带来的好处：

(1) 从使用者角度，DateCalculator类因为“**简单**”，所以“**易用**”。

(2) 具体计算日期的算法被封装到了 DateCalculator类的内部，**在必要时可以修改算法**，**外部调用者不会受到影响**，其调用代码不需要改变。

我们还可以想办法“偷懒”.....

.NET基类库中内置了日期处理的相关功能，可以直接使用它来完成计算两个日期之间相隔天数的问题。

```
DateTime d1 = new DateTime(1999, 5, 10);  
DateTime d2 = new DateTime(2006, 3, 8);  
// 计算结果  
double days = (d2 - d1).TotalDays;
```

从这个示例中，我们可以看到，如果有一个功能强大的组件库，基于这些组件开发应用程序，可以大大地提升软件开发效率，因为我们可以重用别人的工作，不再需要一切从头开始。

对比前后三个程序，你有什么感悟？
