【1】　　程序

【2】　　结构化程序设计

【3】　　面向对象程序设计

【1】　　程序

我们编写程序是让计算机解决现实问题，那么要解决现实生活中的实际问题首先要将实际的问题表示成计算机还能够认识的数据和信息，也就是说这些信息要用**一定的数据**来表示，并且我们要施加**一定的逻辑**，来对这些数据进行处理，从而**解决实际的问题**，这样我们就得到了一个概念就是利用一定的逻辑来处理数据，程序的本质就是数据处理。

因而 pascal 之父得出了一个公式，他是**结构化程序设计**之父，迪杰斯特拉，他提出了程序：

程序=算法+数据结构

数据结构：是描述数据的。

算法：描述数据的逻辑。

和我们刚才的逻辑是一样的，这就是程序，另外我们经典的教科书还有一句关于程序的描述，**程序是完成一定功能的一系列有序指令的集合。**我们知道一个计算机指令，它是包含两个要素的，**指令=操作码+操作数.**操作码就是操作这些数据的一个逻辑，不同的操作码具有不同的逻辑，或者说是不同的指令，它是完成不同的功能，从这个描述，我们就可以看出来，指令它是有一定的逻辑的，它实际上也是**按照一定的逻辑来处理数据的**，如果我们将这些指令进行整合，**将指令按照一定的顺序进行整合，就形成了程序。**这里面就人施加了我们解决问题的一些逻辑，也就是算法。

(1)程序

更多的是我们要站在机器的角度来解决问题，而不是站在解决问题的本身的空间中解决问题，这时候现实生活中要处理的数据不得不用二进制来表示，我们施加的逻辑也不得不用二进制来表示。因为我们用机器语言的编程我们要进行头脑的切换，将现实生活中的表示方式更改为机器的表示方式，机器给我们提供了东西，也就是说机器给我们提供了东西，我们得出的结论，就是我们要按照机器的思维来解决问题，我们的思维被束缚了。

(2)机器语言和汇编语言

（1）机器语言是机器能读懂得二进制指令集合。

（2）由于机器语言语言编写程序很低，因而在计算机的发展过程中，出现了汇编语言，它将机器指令映射为为可以被人读懂的助记符，如 ADD，SUB 等。但是它的抽象层次还是比较低的。我们很多时候还是需要以机器的思维来考虑问题，比如说我们要考虑寻址方式。

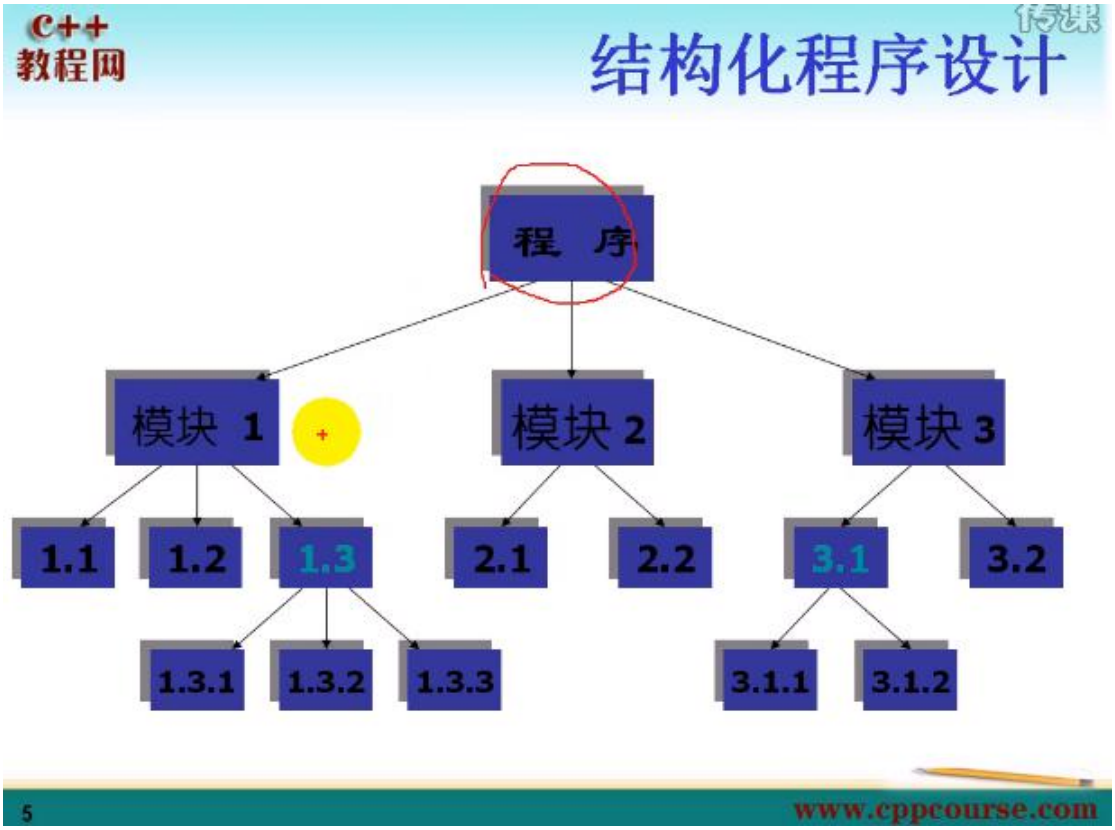
（3）高级语言

高级语言屏蔽了机器的细节，提高了语言的抽象的层次，程序中可以采用具有**一定含义的数据命名**和容易理解的**执行语句**，这使得书写程序的时候可以**联系到程序所描述的具体事物**。也就是我们更加容易站在问题本身来考虑问题。而不是束缚在机器的空间来考虑问题，这就是**程序，汇编语言和机器语言**和**高级语言**的介绍。

2 结构化程序设计

传统的程序设计方法可以归纳为“程序=算法+数据结构”，将程序定义为处理数据的一系列的过程，这种设计方法的着眼点是面向过程的，特点是数据和程序的分离，即数据和数据处理分离。这个我们该怎么理解呢？实际上在结构化程序设计之前，比如说机器语言和汇编语言，这种程序我们可能分不清，那些事数据，那些是数据处理，代码和数据的界限比较模糊，导致我们分不清那些是数据，那些是代码，特别是机器语言（一连串的二进制指令），我们根本看不清，那些是数据，那些是代码。代码和数据是糅合在一起的，这种编程方式，我们戏称为“**面条式的编程方法**”。

接下来我们看一下，结构化程序设计有什么优点呢？它将程序归结为算法和数据结构，它的核心的思想是采用**自顶向下，逐步细化**的设计方法，也就是说它是在问题空间考虑问题，



将一个程序分成模块（或者说子程序的集合），如果这些模块或者子程序比较大，我们还可以进一步进行分解，分成更小的模块，这就是结构化程序设计的思想。

结构化程序设计采用**自顶向下**，将系统分成是**分层的子程序**的集合。结构化程序设计就可以将程序分成**功能不同的模块**，相对于**面条式的编程**方式来说，使得程序更加具有**条理性**。虽然有一些**局部的变量**，但是很多数据仍然属于整个程序，因为结构化程序设计有很多的**全局变量**。这些全局变量在某个地方进行更改，在某个地方进行更改，会对整个程序产生难以预料的影响。这是结构化程序设计的思想。它实际上是可以按照一定的逻辑来考虑问题的。将问题自顶向下，逐步进行细分，我们可以举一个例子，比如说，我们举一个**在家就餐**的例子：

（1）准备菜

（2）烧菜

（3）上菜

也就是说我们处理问题的时候，着眼点是算法和数据结构。以算法为中心，而不是以数据为中心。自顶向下逐步进行细分，

（1）准备菜（到菜市场买菜，洗菜，切菜）

逐步进行细分，这就是结构化程序设计的思想，它就要求一个软件负责人，他了解整个系统的任务分解，但是当问题复杂度超过了人能够理清各个子系统之间的调用关系的时候，这种自顶向下的设计方法可能失效。因而就容易出现软件危机，这就是结构化的程序设计思想。

- 结构化的程序设计方法的基本思想是采用**自顶向下**的、**逐层细化**的设计方法和单输入单输出的控制结构，其理念是将大型的程序分解成小型的程序和便于管理的任务，如果其中的一项任务仍然过大，则将它分解为更小的任务，这个过程将一直持续下去，直到将程序分解成小型的，易于编写的模块，如果其中的一项任务仍然过大，则将它分解为更小的任务，这个过程将一直持续下去，直到将程序划分成小型的，易于编写的模块。

结构化程序设计的缺点

- 结构化程序设计 为了处理复杂问题提供了有利手段，但是 80 年代末，这种设计方法逐渐暴露了一下缺点;

  (1) 程序难于管理

  (2) 数据修改出现问题（比如说程序的全局变量修改了，容易影响其他的部分）

  (3) 程序的重用性差

(4) 用户难以在系统分析阶段准确定义，致使系统在交付使用时产生很多的问题。（如果产生问题，那么有需要重新修改程序，因为程序的可重用性比较 差）。、

(5) 用系统开发每个阶段的成果来控制，不能适应事物变化的要求（如果事物变化的化，有需要重新的开发，所以它**不适合大型软件的开发**）。

- 这些问题的根源就是**数据与数据处理的分离**。（因为程序当中处理的数据仍然是属于整个程序的，不是某个模块的，模块与模块之间需要共享很多的数据，这些数据的共享容易带来很多的问题，所以说就出现新的程序设计思想，面向对象的程序设计思想！）

- 面向对象程序设计思想是通过**交互作用**完成**特定功能**的对象的集合。每个对象用自己的方法来**管理数据**，也就是只有对象**内部的代码**才能够**操作对象**的数据。
  也就是说将程序看成：**程序=对象+对象+对象+消息传递**
  这些对象都有一定的交互作用，这些交互作用就是**消息传递**。通过消息传递来进行**交互作用**。
  这有什么好处呢？这是因为每个对象都有自己的方法来管理自身的数据，也就说每个对象它具有：**对象=算法+数据**
  这些数据只能由对象内部的算法进行管理。这就是一个**封装**的一个思想。这些数据只能由内部的一些算法进行管理，所以外部的一些变换不会影响到这个对象，因为这个对象所对应的类已经编写好之后，就可以重用，外部的变化不会影响到内部。这就是面向对象的一种程序设计方法。
  它的本质是将数据和数据的处理方法封装起来，这些数据不再归整个程序所有，它只归类对象内部所有，数据只能由类内部操作，外部不能进行操作，所以说他能应对变换，这就是面向对象的程序设计思想。
  它主要只有三个特征;
  (1) 封装
  (2) 继承
  (3) 多态