

第一章

c++的类比

从这些不同的实验中产生了程序。这是我们的
我们的经验是:程序不是从一个人的头脑中蹦出来的
或者像我们这样的两个人,而是从日常工作中出来的。

——斯托克利·卡迈克尔和查尔斯·v·汉密尔顿,《黑人权



1.1 c++的历史

1.2 编译过程

1.3 你的第一个 c++程序

1.4 c++程序的结构

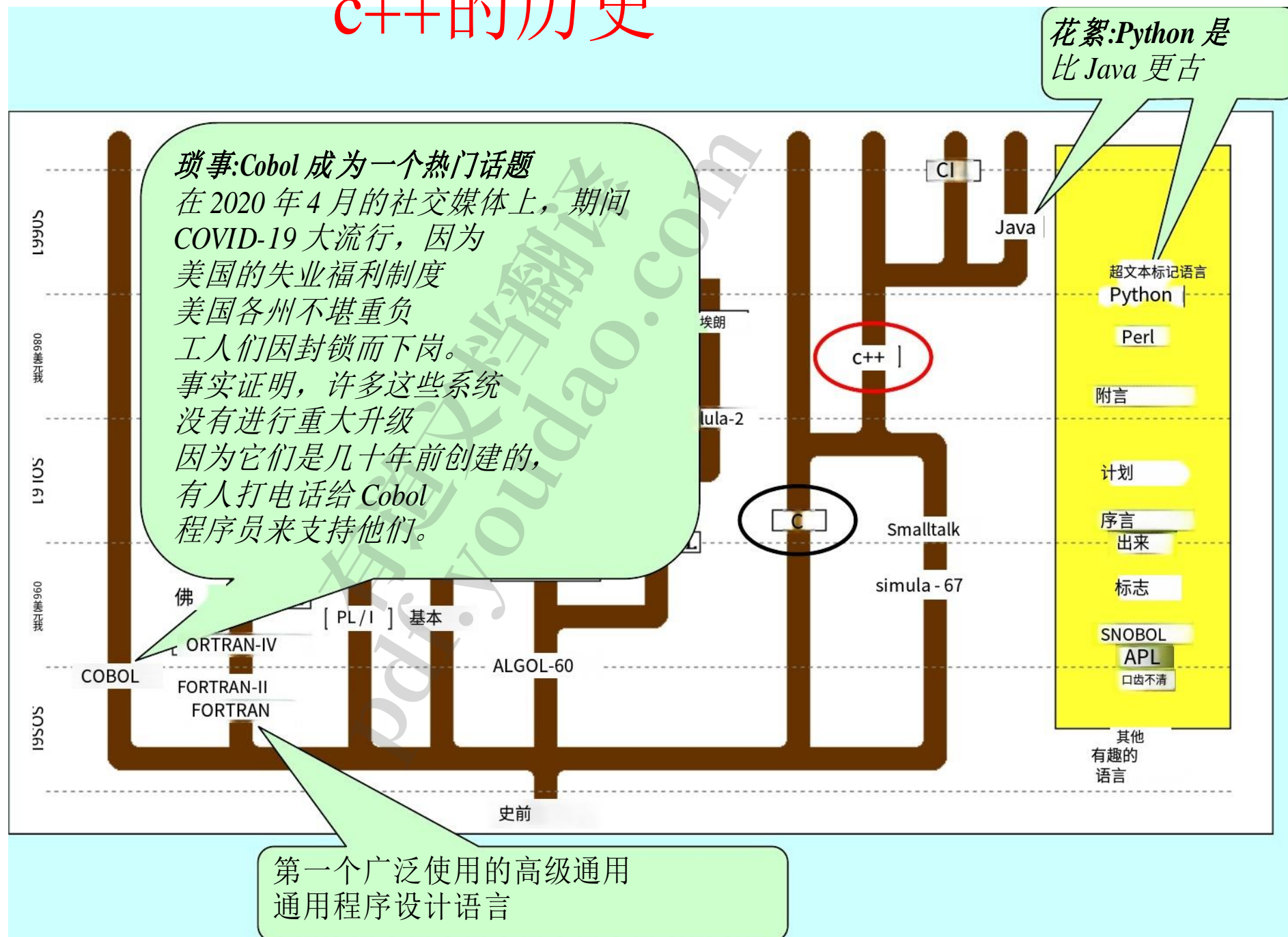
1.5 变量

1.6 数据类型

1.7 表达式

1.8 语句

c++的历史



大 Cobol开发工程师

大连嘉友软件技术有限
公司广州佛山地区·全职

6天前

通过LinkedIn

北 Cobol开发工程师

北京知库咨询有限公
司广州佛山地区·全职

13天前

通过LinkedIn

西 Cobol开发工程师

西安拓达网络科技有限
公司广州佛山地区·全职

7天前

通过LinkedIn

西 Cobol开发工程师

西安拓达网络科技有限
公司广州佛山地区·全职

7天前

通过LinkedIn

招聘20名优秀cobol程序员(
现在就招!)我..

招聘66名COBOL程序员。查看薪
水, 比较评论, 轻松申请, 然后
被录用。新的cobol程序员需要的
职业每天都在增加.....

www.simplyhired.com

4万8千到10万8千美元的Cobol
程序员工作...

www.ziprecruiter.com

7.8万- 13.9万美元的Cobol程序
员工作...

www.ziprecruiter.com

紧急!Cobol程序员工作-
2022年8月(与...

jooble.org

20个最好的cobol程序员
工作(现在招聘!)我..

www.simplyhired.com

c++版本

- c++编程语言是由贝尔实验室的 Bjarne Stroustrup 自 1979 年以来开发的，作为 C 语言的扩展(由 Dennis Ritchie 自 1972 年以来也在贝尔实验室)，因为他想要一种类似于 C 的高效和灵活的语言，C 也提供了高级特性。
- c++最初于 1998 年被国际标准化组织(ISO)标准化为 c++ 98，随后被 c++ 03、c++ 11、c++ 14 和 c++ 17 标准修订。

目前的 c++ 20 标准用新的特性和扩展的标准库取代了这些。

c++ 23 是此后的下一个计划中的标准，保持了目前每三年发布一个新版本的势头。

注意:所有的作业都将使用 c++ 17 的标准版本进行编译。



c++ vs. Python

	c++	Python
执行	编译并链接成可执行文件。	由 an 解释执行引擎。
类型	静态类型，显式声明，绑定到名称，在编译时检查	动态类型，绑定到值，在运行时检查，而且不容易被破坏
内存管理	需要更多的关注簿记和储存细节	几乎不需要注意到内存管理。支持垃圾回收。

语言
复杂性

复杂且功能齐全。

c++试图给你每一个
阳光下的语言特性
而同时从不
(强行)抽象任何东西
远离那些可能
影响性能。

简单。Python 试图给
你只有一种或几种方法
做事情，那些方法是什么
设计得简单，甚至在
语言的代价
动力或运行效率。

编译器和解释器

编译器和解释器是执行用程序设计语言或脚本语言编写的程序的两种不同方式。

编译器接收整个程序并将其转换为通常存储在文件中的目标代码。目标代码也被引用为二进制代码，在链接后可以直接由机器执行。编译编程语言的例子有 C 和 c++。

- 类比直接执行用编程语言或脚本语言编写的指令，而不事先将它们转换为目标代码或机器码。解释性语言的例子有 Python 和 R。
- 与人类语言的类比……

编译器和解释器

编译器和解释器都将源代码(文本文件)转换为标记。基本的区别是，一个编译器系统，包括一个(内置的或单独的)链接器，生成一个独立的机器码程序，而解释器系统反而执行由高级程序描述的操作。

- 一旦程序被编译，它的源代码对于运行代码就没有用处了。对于解释型程序，每次运行程序都需要源代码。

一般来说，解释型程序比编译型程序运行得慢。

编译器和解释器

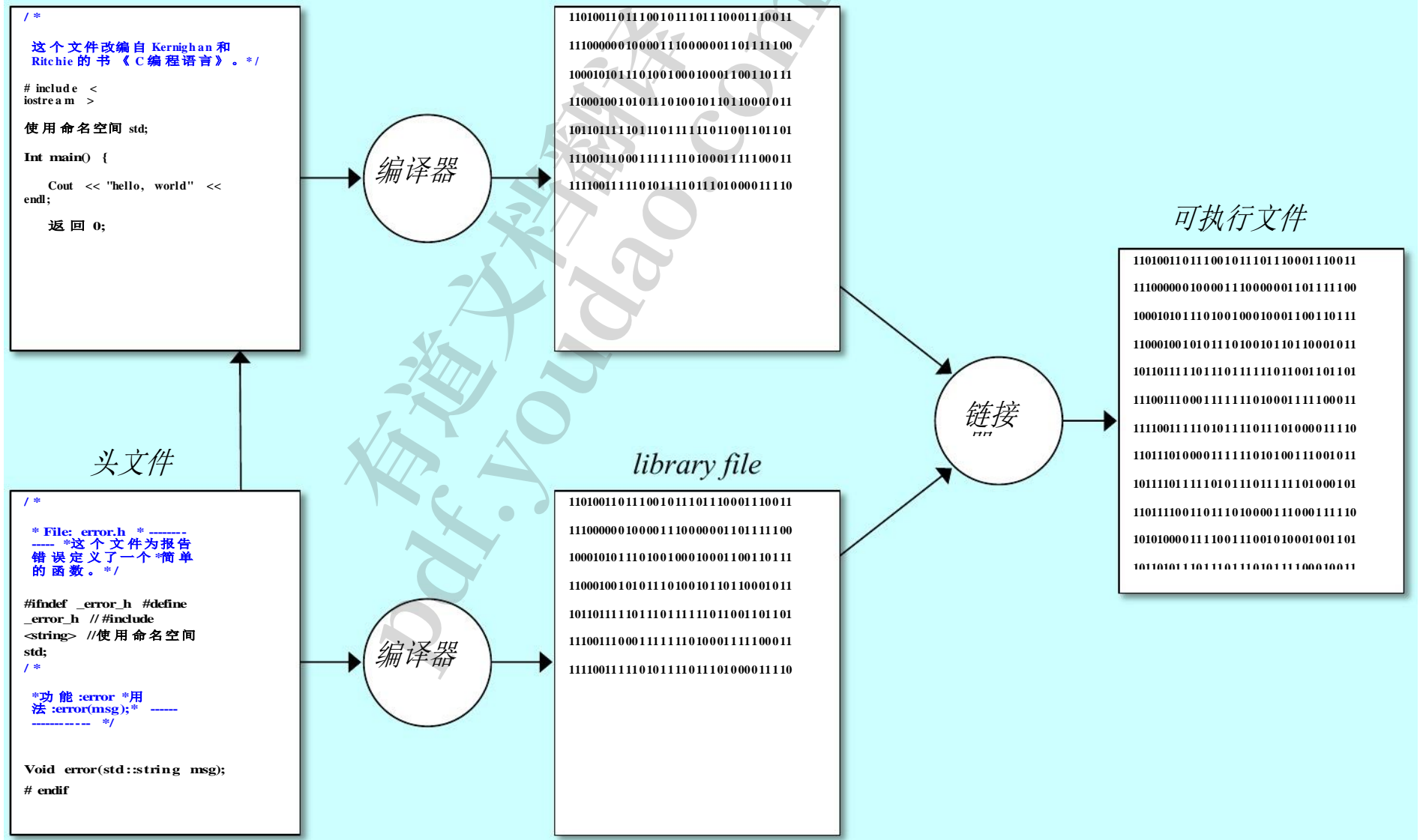
- Java 是一种编译的编程语言，但它的源代码被编译成一种称为字节码的中间二进制形式。
- 字节码随后由 Java 虚拟机(JVM)执行。不同的 jvm 支持不同的系统。写一次，到处运行(WORA)!
- 现代 jvm 使用一种称为即时(Just-In-Time, JIT)编译的技术，在运行时将字节码编译为硬件 CPU 实时理解的本地指令。
- 一些 jvm 的实现可能会选择解释字节码，而不是 JIT 编译成机器码，然后直接运行它。
- 与人类语言的类比……

编译过程

源文件

对象文件

可执行文件



IDE 和编译器

- IDE(集成开发环境):为计算机程序员提供全面的软件开发设施的软件，通常包括源代码编辑器，有时包含：
 - 一个类浏览器，一个对象浏览器，以及一个类层次图，用于面向对象的软件开发
 - GUI(图形用户界面)构造工具
 - 版本控制系统
 - 构建自动化工具(至少必须包含编译器或解释器，或两者兼有，有时也要有链接器)
 - 调试器
- 所有这些组件可能来自第三方。

IDE 和编译器

教程

ide, 例如:

- c++ shell (<http://cpp.sh/>)

- 代码::块

- 微软 Visual Studio

- Apple Xcode

- Qt creator

- 编译器, 例如:

- 微软 Visual c++ 编译器

- GCC (GNU 编译器集合)

- MinGW (Windows 上的极简 GNU)

- GUI 和额外的库支持, 例如,

- Qt (Design Studio)

- MFC(微软基础类)

- StanfordCPPLib(来自教科书)

- OpenCV(开源计算机视觉库)



“Hello World” 程序

c++编程语言是 20 世纪 70 年代初贝尔实验室开发的 C 语言的扩展。C 的主要参考手册由 Brian 克尼根和 Dennis Ritchie 编写。

在他们的书的第一页，作者建议开始学习一门新的编程语言的最好方法是编写一个简单的程序，在显示器上打印消息“hello, world”。这个建议在今天仍然很有道理。

1.1 开始

学习新编程的唯一方法
语言就是用它写程序。第一个
编写的程序对所有语言都是一样的：
打印单词

你好，世界

这是最大的障碍；要跨越它，你必须
能够在某处创建程序文本，
编译它，加载它，运行它，并找到你的
输出了。有了这些机械细节
掌握了这些细节，其他一切就相对容易了。
在 C 语言中，打印“hello, world”的程序是
`#include <stdio.h>`

```
main () {  
    ... printf ("hello, world \n
```

c++中的“Hello World”程序

```
/*文件:HelloWorld.cpp
* -----
*此文件改编自示例
*在克尼根和 Ritchie 的第一页
*预订 C 编程语言。
*/
```

评论

```
# include < iostream >
使用命名空间 std;
```

不用担心字体
颜色，可以自定义
这些在大多数 IDE 编辑器

图书馆的夹杂物

```
Int main() {
    Cout << "hello, world" << endl;
    返回 0;
}
```

主程序

用 c++编写的“Hello World”程序

```
/*  
 *文件:HelloWorld.cpp  
 * -----  
 *此文件改编自示例  
 *在克尼根和里奇的第一页  
 *将 C 程序设计语言编成书。  
 */  
  
# include <  
iostream >  
使用命名空间 std;  
Int main() {  
    Cout << "hello, world" << endl;  
    返回 0;  
}
```

评论

c++程序的结构

注释是一种被编译器忽略的文本，但它向其他程序员(实际上也向你自己)传递了有关源代码的信息。

多行注释。

```
/*注释 1 行注释 2 行注释  
3 行*/
```

- 单行注释

```
//注释 1 行//注释 2 行  
//注释 3 行
```

编写注释是非常重要的。代码告诉你怎么做，注释告诉你为什么！

c++中的“Hello World”程序

```
/*  
 *文件:HelloWorld.cpp  
 * -----  
 *此文件改编自示例  
 *在克尼根和里奇的第一页  
 *将 C 程序设计语言编成书。  
 */  
  
# include <  
iostream >  
使用命名空间 std;  
Int main() {  
    Cout << "hello, world" << endl;  
    返回 0;  
}
```

图书馆的夹杂物

c++程序的结构

库是先前编写的执行有用操作的工具的集合。

`#include <iostream>` 指示编译器从所谓的头文件(.h)中读取相关定义;
对于你自己的库, 写 `#include "myownlib.h"`。(还记得 Python 中的 `import` 吗?)

- 为了确保在大型系统的不同部分定义的名称不会相互干扰, c++将代码分段到称为命名空间(namespace)的结构中, 每个结构都跟踪自己的一组名称。
- 标准 c++库使用一个名为 `std` 的命名空间, 这意味着你不能引用在标准头文件(如 `iostream.h`)中定义的名称, 除非你让编译器知道这些定义属于哪个命名空间。

预处理器指令

- 预处理器指令不是程序语句，而是用于预处理器的指令，预处理器负责检查代码在实际编译之前。

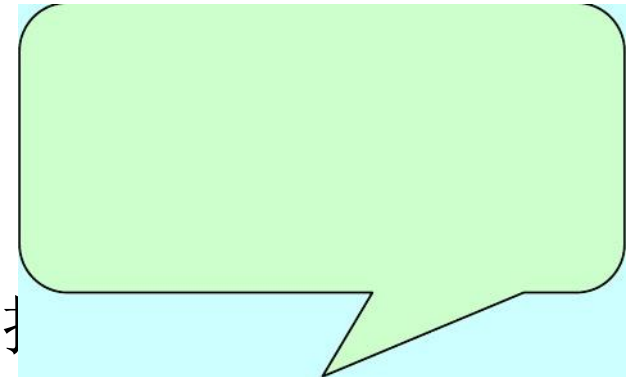
要知道你不需要

哈希/磅字符(#)为了考试记住它。-预处理指令(包括

Undef. if. ifdef. ifndef. else. elif. endif. line,
错误,编译指示)

-参数(取决于指令)-换行符(换行符)

预处理器指令的末尾不需要分号(;), 并且可以通过在换行符之前加上反斜杠(\)将预处理器指令扩展到多行。



使用“使用命名空间”

- 当程序变得非常大并大量使用不同的库时，可能会发生名称冲突。一个标识符(例如，一个函数名)被不同的库使用来引用不同的东西。编译器不会让同一个名称用于两个不同的东西!

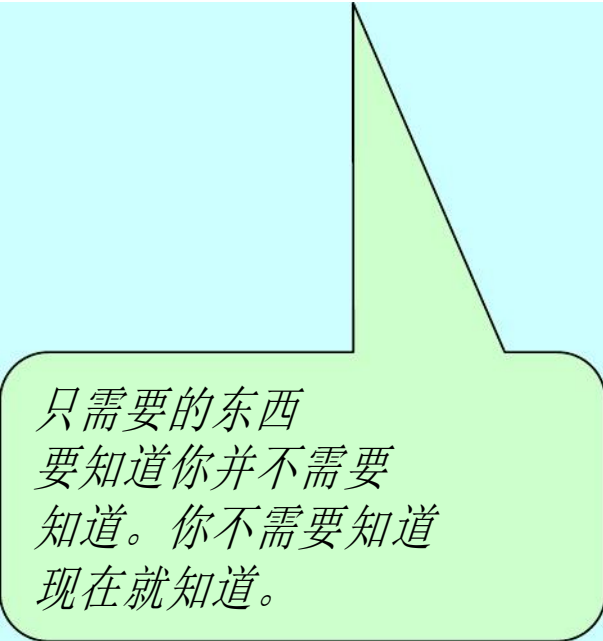
命名空间的想法是，标识符可以分组到单独的集合中，每个这样的集合或命名空间都有一个名称。如果相同的标识符出现在两个库中，每个库都有自己的命名空间，则可以通过用命名空间名称限定标识符来解决冲突。

- 为了避免一直编写命名空间名称的不便，我们可以使用“使用命名空间 x”来让编译器知道一个未限定的标识符可能来自命名空间 x。

使用 “Using namespace”

为了避免总是使用整个命名空间，人们也可以为你需要的名称使用特定的 using 声明，例如，“使用 `x::name1`；使用 `x::name2`；”

你可以声明一些东西并把它们放到你自己的命名空间中，但是在这个课程中我们只需要“使用命名空间 `std`”，或者“使用 `std::cout`；使用 `std::endl`；”。



只需要的东西
要知道你并不需要
知道。你不需要知道
现在就知道。

用 c++编写的“Hello World”程序

```
/*  
*文件:HelloWorld.cpp  
* -----  
*此文件改编自 Kernighan 和 Ritchie 的书  
《C 程序设计语言》第 1 页的示例*。*/
```

```
#使用命名空间 std 包  
含 <iostream>;
```

```
Int main() {  
    Cout << "hello, world" << endl;  
    返回 0;  
}
```

} 主程序

c++程序的结构

每个 c++ 程序都必须包含一个名为 `main` 的函数。它指定计算的起点，并在程序启动时被调用。当 `main` 完成其工作并返回时，程序的执行就结束了。

函数是执行特定操作的有名称的代码段。

`main` 函数可以调用其他函数，例如下一张幻灯片中的 `PowerOfTwo` 程序中的 `raiseToPower`。

命令式编程范式(命令式编程范式):改变程序状态的显式语句序列，指定如何实现结果。

- 结构化: 程序具有干净、无跳转、嵌套的控制结构。
- Procedural: 命令式编程，程序操作数据。

c++程序的结构

```
/*文件:功能:powersoft .cpp
   raiseToPower
   用法:int p =raiseToPower(n,k);* -----
   * -----这个程序生成一个幂的列表* Returnstwoud 到

#include <iostream>
using namespace std;
int result = 1;
for (int i = 0; i < k; i++) {
    result *= n;
}
raiseToPower(n, k);
return result;
}

int main() {
    // ...
    return 0;
}
```

Programcomment
→ 程序注释

库包含函数定义函
数原型

主程序

“先声明后使用”规则

- C++程序由各种实体组成，如变量、函数、类型和命名空间。这些实体中的每一个都必须在使用之前进行声明。(想想程序一开始就包含的库)。
- 注意，函数 `raiseToPower` 是在 `main` 函数之后的函数定义部分定义的。为了遵守“先声明后使用”的规则，函数的声明，即函数原型，必须出现在主函数之前。
- 函数原型为函数指定一个唯一的名称，以及关于其类型和其他特征的信息(稍后详细说明)。
- 我们可以避免使用函数原型吗？
 - 我们总是在使用函数之前定义函数怎么样？



c++ vs. Python

	c++	Python
执行	编译并链接成可执行文件。	由 an 解释执行引擎。
类型	静态类型，显式声明，绑定到名称，在编译时检查	动态类型，绑定到值，在运行时检查，而且不容易被破坏
内存管理	需要更多的关注簿记和储存细节	几乎不需要注意到内存管理。支持垃圾回收。

语言
复杂性

复杂且功能齐全。

c++试图给你每一个
阳光下的语言特性
而同时从不
(强行)抽象任何东西
远离那些可能
影响性能。

简单。Python 试图给
你只有一种或几种方法
做事情，那些方法是什么
设计得简单，甚至在
语言的代价
动力或运行效率。

数据类型

数据类型由域定义，域是属于该类型的值的集合，以及定义该类型行为的一组操作。

- 虽然许多数据类型是用对象类或其他复合结构表示的(稍后会详细介绍)，但 c++ 定义了一组基本类型来表示简单数据。

在 c++ 中可用的所有基本类型中，本文中的程序通常只使用以下四种：

int 这种类型用于表示整数，即整数，如 17 或 -53。

双 这种类型用于表示包含小数的数字，例如 3.14159265。

bool 这种类型表示逻辑值(true 或 false)。这种类型表示
单个 ASCII 字符。

字符

变量

变量是存储某种类型值的命名地址。

在 c++ 中，必须声明一个变量才能使用它。声明建立了变量的名称和类型，在大多数情况下，还指定了初始值。

变量声明最常见的形式是

类型 *name* = *value*;

其中 **type** 是 c++ 基本类型或更复杂类型(如 **class**)的名称，**name** 是表示变量名称的标识符，**value** 是指定初始值的表达式。

- 通过声明，一个内存地址与变量名相关联，这将在后面的内存管理中讨论。

变量

在 c++ 中，变量最容易被想象为能够存储值的盒子。对于下面的语句：

Int total = 42;

(name is) total

(储存于)FFD0

42

(包含一个)int

每个变量都有以下属性：

- 一个名字，它使你能够区分一个变量和另一个变量。
- 类型，指定变量可以包含什么类型的值。
- 一个值，表示变量当前的内容。
- 现在先不考虑地址的问题。

命名变量的地址和类型是固定的。每当你给变量赋一个新值时，它的值就会改变。

问题

- Python 变量和 c++变量之间有什么区别?

有道文档翻译
pdf.youdao.com

变量

- 变量名区分大小写。

命名约定(也是编程风格的一部分)

- 名称必须以字母或下划线(_)开头。
- 名称中其他字符只能为字母、数字或“_”。名称中不允许使用空格和其他特殊字符。
- 使用有意义的单词(好的变量名类似于注释),但不能是保留的关键字之一。

表1-1 c++中的保留词

asm	做	内联int	短	类型id
汽车	双		签署	typename
保龄球	dynamic_cast	长	运算符	联盟
打破	其他的	可变的	静态	无符号
情况下	枚举	名称空间	static_cast	使用
抓	显式的	新	结构体	虚拟
字符	走读生	操作符	开关	无效
类	假	私人	模板	挥发性
常量	浮动	保护公共	这	wchar_t而
const_cast	为	注册	扔	
继续	朋友	reinterpret_cast	真正的	
默认的	转到	返回	尝试定义类	
删除	如果		型	

变量范围和范围

大多数声明以语句的形式出现在函数定义的主体中。以这种方式声明的变量被称为局部变量，只能在该函数内部访问。它们有时也被称为自动变量，因为它们是在函数开始执行时自动创建的，在函数结束执行时自动消失。

变量也可以声明为类的一部分。这些称为实例变量(instance variable)，将在后续章节中介绍。

在任何函数和类定义之外声明的变量都是全局变量。避免全局变量，因为它们可以被程序中的任何函数操作，并且很难使这些函数不相互干扰。



问:全局变量的最佳前缀是什么?

答://

2019年1月21日凌晨4:41

变量作用域和范围

- 在计算机编程中，名称绑定是一个名称与一个实体(例如数据)的关联，例如变量。

名称绑定的作用域是计算机程序中绑定有效的区域，即。这里，这里的名称可以用来指代实体。从被引用实体的角度看，又称实体的可见性。

- 变量的作用域描述的是在程序的文本中该变量可以被使用的位置，而范围(或生命周期)描述的是在程序的执行中，变量何时具有(有意义的)值。

问题

main()函数中定义的变量是全局变量吗?

有道文档翻译
pdf.youdao.com

常量

变量是一个占位符，表示可以在程序运行时更新的值，而常量的值不会在程序运行期间改变。

常量的格式取决于它的类型：

- 整数常数由一串数字组成，前面可以有一个减号，如 0、42、-1 或 1000000。
- 浮点常量包括一个小数点，如 3.14159265 或 10.0。浮点常数也可以用科学表示法表示，在数字的数字后面加上字母 E 和指数，因此 5.646E-8 表示 5.646×10^{-8} 。
- 布尔类型的两个常量分别为 true 和 false。
- 字符常量和字符串常量将在后面的讲座中详细讨论。目前，你只需要知道，字符常量是用单引号括起来的，比如 `a`，而字符串常量是由双引号括起来的字符序列组成的，比如 “hello, world”。

命名常量

如果你在程序中多次使用同一个常量，如果能给这个常量一个名字，然后在程序中到处引用它，那就更好了。

要在 c++ 中创建一个命名常量，需要在普通变量声明之前加上关键字 `const`。这告诉编译器已经创建了一个特殊的变量，它的值是不变的。

```
const 双 PI= 3.14159265358979323846;
```

- 创建命名常量时，必须给它赋值。你不能在命名常量初始化后给它赋值。

命名常量

- 在 C 中，预处理器指令 `#define` 用于为常量创建名称，以便预处理器可以在整个代码中用适当的常量替换该名称的每个实例。

#定义 PI 3.14159265358979323846

- 这在 c++ 中仍然有效，但由于以这种方式创建的常量只存在于创建它的文件中，因此有可能在另一个文件中具有完全不同的值的相同定义。这可能会导致灾难性的后果。

使用 `const` 是一个好习惯，因为我们可以控制它的作用域。

表达式

```
/*
```

```
*文件:AddThreeNumbers.cpp
```

```
* -----
```

```
*这个程序将三个浮点数相加，并打印它们的和。*/
```

```
#使用命名空间 std 包  
含 <iostream>;
```

```
Int main() {
```

```
    双 n1, n2, n3;
```

```
    cout << “这个程序加三个数字。” << endl; Cout << "1  
    号:";
```

```
    Cin >> n1;
```

```
    Cout << "2 号:";
```

```
    Cin >> n2;
```

```
    Cout << "第三号:";
```

```
    Cin >> n3;
```

```
    双 和 = n1 + n2 + n3;
```

```
    cout << “sum is” << sum << endl;
```

```
    返回 0;
```

}

有道文档翻译
pdf.youdao.com

表达式

- AddThreeNumbers 程序的核心是线条

Double sum = n1 + n2 + n3;

执行实际的加法。

出现在等号右侧的 $n1 + n2 + n3$ 是一个表达式的例子，它指定了计算中涉及的操作。

c++中的一个是由运算符连接在一起的术语组成的。

每个词都必须是下列之一：

- 一个常量值(例如 3.14159265 或 “hello, world”)
- 常量/变量名(如 PI、n1、n2 或 total)
- 一个返回值的函数调用(例如 readInt())
- 用括号括起来的一个表达式(递归定义)

操作符和操作数

与大多数语言一样，c++程序以算术表达式的形式指定计算，与数学中的表达式非常相似。

c++中最常见的操作符是用于指定算术计算的：

+	除了	*	乘法
-	减法	/	部门
..			
..			
..			
		%	剩余部分

- c++中的操作符通常出现在两个子表达式之间，这两个子表达式被称为操作数。接受两个操作数的操作符称为二元操作符。

-操作符也可以以一元操作符的形式出现，如表达式-x，表示 x 的负数。

除法和类型转换

- 在 c++ 中，当你将二元运算符应用于数值时，如果两个操作数都是 int 类型，结果将是 int 类型，但如果有一个操作数是 double 类型，结果将是 double 类型。

对于除法运算，这条规则有重要的影响。例如，表达式

14/5

似乎它的值应该是 2.8，但是因为两个操作数都是 int 类型，c++ 计算的结果是整数

去掉小数部分。因此结果是 2。如果你想获得数学上正确的结果，你需要

需要将至少一个操作数转换为 double 类型，如

Double (14)/ Double (5) Double (14)/5 14/ Double (5)

转换是通过类型转换来完成的，你通过使用类型名称作为函数调用来指定类型转换。

除法和类型转换

问:下列表达式的结果是什么?

双(14/5)

练习:整数除法的陷阱

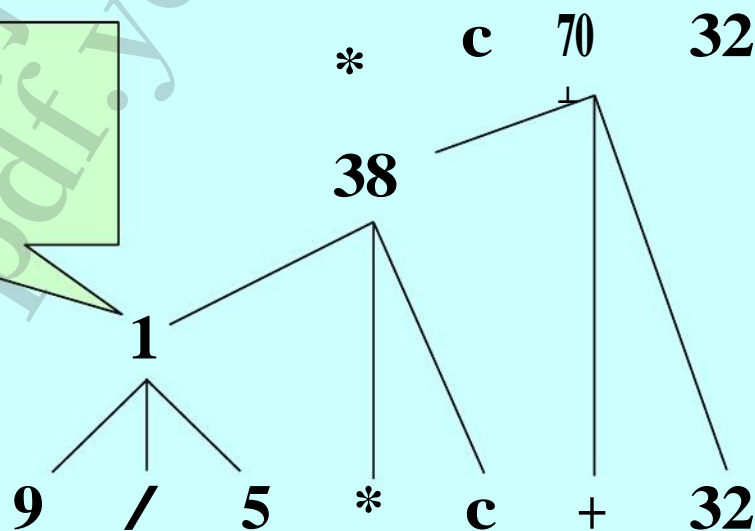
请看下面的 c++ 语句，它们的目的是将 38°C 的温度转换为华氏温度：

```
双 c = 38;  
双 f = 9 / 5 * c + 32;
```



计算过程包括计算下面的表达式：

问题产生于 9 / 5
事实是 9 和 5 都是
类型 int，这意味着
结果也是 int 类型。



整数除法的陷阱

你可以通过将分数转换为 double 来解决这个问题，可以插入小数点，也可以使用类型转换：

```
Double c = 38;  
双 f = 双 (9) / 5 * c + 32;
```

现在的计算是这样的：

100.4

68.4

1.8

9.0

(9) / 两倍 5 * c + 32

余数运算符

唯一没有直接数学对应项的算术运算符是%，它只适用于整数操作数，并在第一个操作数除以第二个操作数时计算余数：

14% 5	<i>返回</i>	4
14% 7	<i>返回</i>	0
7% 14	<i>返回</i>	7

只有当两个操作数都是正数时，%操作符的结果才直观。教科书中的例子并不依赖于知道%如何处理负数。

余数运算符在很多编程应用中都很有用，非常值得学习。

优先级

如果一个表达式包含多个运算符，c++会使用优先级规则来确定求值的顺序。算术运算符的相对优先级如下：

表1 - 4

c++中可用的运算符，按优先级组组织的运算符()

	结合性左
-> .	
一元操作符: - ++ -- ! * ~ (类型) 运算符	正确的
* / %	左
+ -	左
<< >>	左
< > >=	左
== !=	左
	左
^	左
	左
& &	左
	左
吗?	正确的
= op =	正确的

优先级

优先级只在两个操作符竞争同一个操作数时生效。如果两个操作符具有相同的优先级，则按照结合性指定的顺序应用它们，这表示该操作符是向左分组还是向右分组。

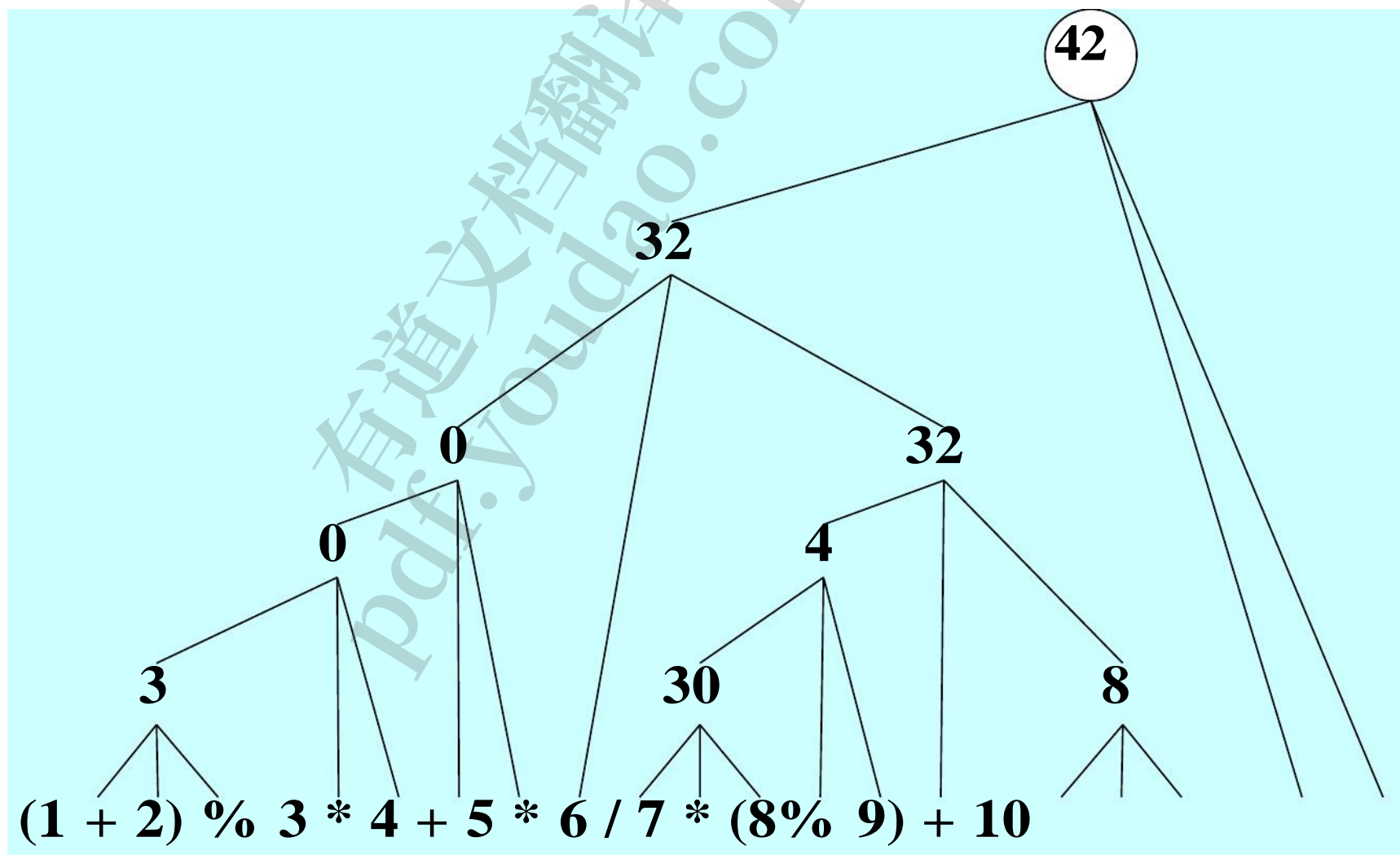
- `c++`中的大多数运算符都是左结合的，这意味着最左边的运算符首先被求值。

少数操作符——最明显的赋值操作符——是右结合的，这意味着它们从右到左进行分组。

括号可以用来改变运算符的顺序。使用括号来明确区分优先级和结合性(就像写注释一样)总是一个好习惯。

练习:优先级求值

屏幕下方的表达式值是多少?



赋值语句

你可以在程序中使用赋值语句来改变变量的值，它的一般形式是：

$$\text{变量} = \text{表达式};$$

赋值语句的作用是计算等号右边表达式的值，并将该值赋给等号左边的变量。因此，赋值语句

$$\mathbf{Total = Total + value;}$$

将变量 total 和的当前值相加

Value，然后将总和存储回变量 **total** 中。将新值赋给变量时，的旧值

这个变量就会丢失。

问题



作为一个数学方程，这当然是错误的。但作为 c++ 中的赋值语句，它合法吗，为什么？

L-value: 有地址，可以出现在赋值语句的两边。R-value: 没有地址，不能出现在赋值的左侧。

速记作业

- 语句，如

```
Total = Total + value;
```

如此常见，以至于 c++ 允许使用以下简写形式：

```
Total += value;
```

速记作业的一般形式是

```
变量 op= expression;
```

其中 op 是 c++ 的任意二元运算符。这条语句的效果与

```
Variable = 变量 op(表达式);
```

例如，下面的语句将工资乘以 2。

```
工资 *= 2;
```

递增和递减运算符

c++程序中另一个经常出现的重要简写形式是递增运算符，最常见的写法是紧跟在变量后面，像这样：

x ++; **++ x;**

这句话的作用是给 x 的值加 1，也就是说这句话等价于

x += 1;

或者更长的形式

x = x + 1;

- ——运算符(也叫减量运算符)类似，只是减 1 而不是加 1。
- ++和——操作符比这里展示的更复杂，但将细节推迟到后面讨论是有意义的。

布尔表达式

用于布尔数据类型的操作符分为两类:关系操作符和逻辑操作符。

有 6 个关系操作符可以比较其他类型的值并产生布尔结果:

== = <小于>大于

= Not = <=小于或等于>=大于
等于

例如, 表达式 $n \leq 10$ 的值为 true, 如果 n 为小于或等于 10, 否则值为 false。还有 3 个逻辑运算符:

&& 逻辑与	P && q 同时表示 P 和 q
 逻辑或	P q 表示 P 或 q(或两者兼有)
! 逻辑不	p 是 p 的反义词

布尔运算符的注意事项

请记住，c++使用=来表示赋值。要测试两个值是否相等，必须使用==操作符。

c++中不允许像数学中那样在一次比较中使用多个关系操作符。要表达数学表达式中所体现的思想

$$0 \leq x \leq 9$$

6

你需要使两个比较都显式，如

$$0 \leq x \ \&\& \ x \leq 9$$

||运算符表示其中之一或两者兼有，在英语中 or 的解释并不总是很清楚(例如 exclusive or)。

当你把 or 和!把!连缀成&&和||，因为它们的意思和非正式英语不一样。

短路的评估

c++使用一种称为短路模式(short-circuit mode)的策略来计算&&和||操作符，在这种模式中，它仅在需要时才计算正确的操作数。

- 例如，如果 n 为 0，则&& in 的右操作数

N != 0 && x % N == 0

根本没有被计算，因为 n != 0 为 false。因为表达式

False && anything

总是 false，表达式的其余部分不再重要。

- 短路求值的优点之一是可以使用&&和||来防止执行错误。如果在前面的例子中 n 为 0，计算 x % n 会导致“除零”错误。

语句

c++中的语句有三种基本类型:

- 简单语句
- 复合语句
- 控制语句

简单语句是通过在 c++表达式的末尾添加分号(;)形成的。

复合语句(也称为块)是括在大括号中的语句序列。

控制语句分为两类:

- 指定某种测试的条件语句
- 指定重复的迭代语句

使用反斜杠(\)可以打断一长行代码。大多数情况下编译器可以识别断行, 但有时(例如, 在预处理器指令中)需要\。

if 语句

最简单的控制语句是 if 语句，有两种形式。只有在特定条件为真时需要执行操作时，才使用第一种形式：

```
If(条件){  
    条件为真时执行的语句  
}
```

条件是
表达式，
求值为 a
布尔值

当需要在两条可选路径中进行选择时，可以使用第二种形式，一种用于条件为真时，另一种用于条件为假时：

```
If(条件){  
    条件为真时执行的语句} else {  
    条件为假时执行的语句}
```

if 语句的常见形式

本书中的例子只使用了以下几种形式的 if 语句:

单行 if 语句

If(条件)语句

带大括号的多行 if 语句

**If(条件){语句…更多语句…
}**

带大括号的 If/else 语句

**If(条件){
 语句_{true} }
else {
 语句_{false} }**

层叠 if 语句

**如果条件₁{语句₁
} else if(条件₂) {
 语句₂ ……更多 else/if 条
件…} else {
 语句_{else} }**

?:接线员

除了 if 语句之外，c++还提供了一种更简洁的方式来表达条件执行，这在某些情况下非常有用。这个功能被称为?:操作符(发音为问号-冒号)，是表达式结构的一部分。?:运算符的形式如下：

条件?表达式₁:表达₂

- 当 c++求?:操作符时，首先确定 condition 的值。如果 condition 为真，c++计算 expression 的值₁并将其作为值;如果 condition 为 false, c++会计算 expression 的值₂代替。

可以使用?:操作符将较大的 x 和 y 赋值给变量 max，如下所示：

Max = (x > y) ?X: y;

switch 语句

The When C++ if none evaluate the c++ of looks statement the executes the values for statement a in provide the case provides witch case clause in statement, a convenience convenience case clause that clause it match matches begin syntax syntax until the expression. 通过选择 reach these evaluation expression, if c++ between expression break 求 expression, statement, 一个 set was which 的可能相等哪个 must statements 来产生路径: v should, c++ in appearance 类整数将默认选择 end clause. the value. of second each clause. clause.

Switch(表达式){

例 v_1 :

要执行的语句, 如果 $expression = v_1$ 打破;

例 v_2 :

要执行的语句, 如果 $expression = v_2$

打破;

.....如果需要更多的 case 子句...

默认值:

如果没有值 match break, 将执行的语句;

}

有道文档翻译
pdf.youdao.com

switch 语句的例子

```
int  
int  
main ()  
月;  
{如果一个案例中不包含 break,  
cout << "Enter numeric month (Jan=1):all statements ";后面的 case 是  
Cin >>月份;也执行了,直到一断就是  
Switch(月){到达或结束的开关块。  
    案例 2:  
        Cout << “ 28 天(闰年 29 天)” << endl;  
        打破;  
    案例 4:案例 6:案例 9:案例 11:  
        Cout << “ 30 天 ” << endl;  
        打破;  
    案例 1:案例 3:案例 5:案例 7:案例 8:案例 10:案例 12:  
        Cout << “ 31 天 ” << endl;  
    返回 0;  
}
```

while 语句

while 语句是 c++中最简单的迭代控制语句，其形式如下：

```
While(条件){  
    要重复的语句  
}
```

- 当 c++语言遇到 while 语句时，它首先计算括号中的条件。

如果 condition 的值为 true, c++就执行循环体中的语句。

在每个循环结束时，c++重新计算 condition 的值，看看它的值是否发生了变化。如果 condition 求值为 false, c++退出循环并继续执行 while 语句体末尾右花括号后面的语句。

for 语句

c++中的 for 语句是一个特别强大的工具
独立于循环体执行的操作来指定循环的控制结构。语法看起来
像这样:

```
For (init; 测试; Step) { 要重复的  
    语句  
}
```

c++通过执行以下步骤来计算 for 语句:

- 1.评估 init, 它通常声明一个控制变量。
- 2.如果值为 false, 评估测试并退出循环。
- 3.执行循环体中的语句。
- 4.评估步骤, 通常更新控制变量。
- 5.回到步骤 2, 开始下一个循环循环。

比较 for 和 while

for 语句

```
For (init; 测试; 步){  
    要重复的语句
```

在功能上等价于下面使用 while 的代码:

```
    初  
    始  
    化;  
While (test){语句要重复的步  
    骤;  
}
```

for 语句的优点是, 所有你需要知道的用来理解循环将运行多少次的信息都明确地包含在标题行中。

练习:读取 for 语句

请描述以下 for 语句的效果:

1.For (int I = 1;I <= 10;我++)

这条语句执行循环体 10 次, 控制变量*i* 将依次接受 1 到 10 之间的每个值。

2.For (int I = 0;i < N;我++)

这条语句执行循环体 *N* 次, *i* 从 0 数到 *N* - 1。这个版本是标准的 *repeat - n* 次惯用法。

3.For (int n = 99;N >= 1;N -= 2)

这条语句从 99 到 1 乘 2 倒数 50 次。

4.For (int x = 1;X <= 1024;X *= 2)

这条语句执行循环体, 变量 *x* 取 2 从 1 到 1024 的连续幂, 共 11 次。

c++编程风格指南

风格，也称为可读性，是我们所说的规范 c++代码的规范。

“风格”这个术语有点名不符实，因为这些约定涵盖的远远不止源代码文件的格式化。

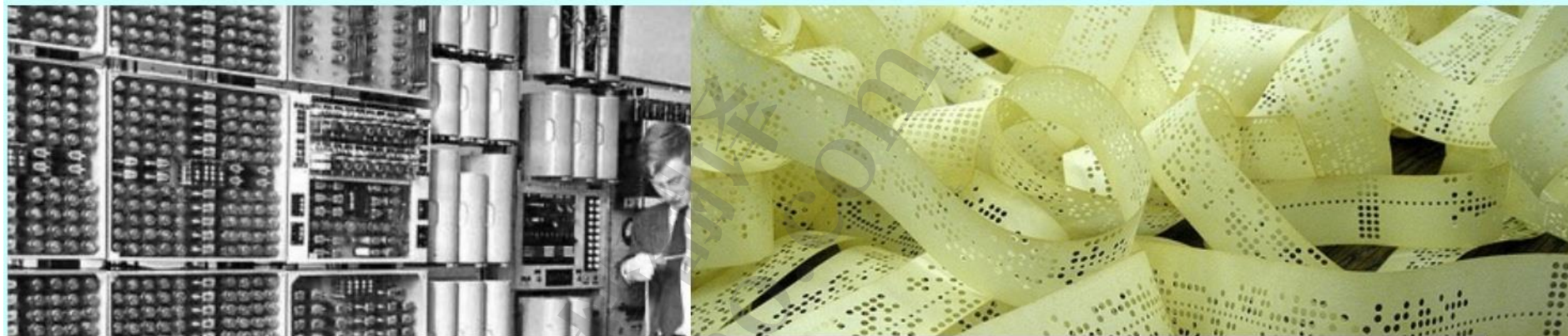
使用常识，保持一致。

- <https://google.github.io/styleguide/cppguide.html>

- 风格很重要!



调试(Debug):寻找 bug



Joe Groff
@jckarter

在printf之后，sleep是第二好的调试器

凌晨2:16 - 2019年2月15日

@爱可可-爱生活

u.com

单元测试

作为程序员，最重要的职责之一是尽可能彻底地测试代码。虽然测试永远不能保证没有错误，但采用一种深思熟虑的测试方法可以帮助你至少找到代码中的一些错误。

每当你编写一个模块时，创建一个测试来检查该模块的正确性，与其他代码隔离，这是一个很好的实践。这样的测试称为单元测试。

您可以使用<cassert>库中的 assert 宏来实现单元测试策略。

如果 assert 宏中的条件表达式为 true，则执行正常继续。

如果表达式为 false，assert 宏打印一条标识错误来源的消息，并退出程序。

宏是一个有名字的代码片段。无论何时使用这个名称，它都会被宏的内容所取代。

assert 宏的例子

```
int main ()
{
    Int x = 7;

    /*介于和 x 之间的一些大代码
      不小心变成了 9*/
    X = 9;

    //程序员假定代码中的 x 为 7
    断言 (x == 7);

    /*其余代码*/

    返回 0;
}
```

只需要的东西
要知道你并不需要
知道。我不需要你知道
现在完全明白了。

断言失败:x==7, 文件测试。Cpp, 第 13 行

这个应用程序要求运行时以一种不寻常的方式终止它。请联系应用程序的支持团队了解更多信息。

快速演示(vsc) hello, world !!

VSC -扩展- Makefile



The screenshot shows the Visual Studio Code extension marketplace interface for the 'Makefile的工具' (Makefile Tools) extension. The extension is published by Microsoft (微软) and has a version of v0.6.0. It has a rating of 4.5 stars based on 16 reviews. The description states that it provides Makefile support in VS Code, specifically for C/C++ IntelliSense. The interface includes buttons for '禁用' (Disable), '卸载' (Uninstall), and a settings gear icon. A note at the bottom indicates that the extension is globally enabled.

Makefile的工具 v0.6.0 [预览](#)

微软 | 785321年 | ★★★★★ (16)

在VS Code中提供makefile支持:C/ c++ IntelliSen...

禁用 ▼ 卸载 ▼ ⚙️

此扩展是全局启用的。

开放
你的
文件夹,
首页

Makefile



Makefile - 启动目标

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14
{
  "makefile.launchConfigurations": [
    {
      "鹿": "/用户/ kinleylam / OneDrive / CSC3002 / CPP /任务",
      "binaryPath": "/Users/kinleylam/OneDrive/CSC3002/CPP/Assignment2",
      "binaryArgs": []
    },
    {
      "鹿": "/用户/ kinleylam / OneDrive / CSC3002 / CPP / helloworld",
      "binaryPath": "/Users/kinleylam/OneDrive/CSC3002/CPP/helloworld/",
      "binaryArgs": []
    }
  ]
}
```

构建
只

建立
运行

Makefile: 项目

配置:[未设置]构建目标:[全
部]启动目标:[hellow

Alternative-Without 使

= I o o 。 O =

helloworld G

helloworld.cpp C

helloworld.h =

helloworld.o

GMakefi

打开到侧面打开
With..

在 Finder 中显示 CR 打开在集成终端中，选择为比较打开时间线

降低
复制

☐ x
☐ C

kinleymaii21871254 - macbook - pro helloworld % make clean
rm -f .o *.a

kinleym@I121871254s-MacBook-Pro helloworld % make alu
g++-std=c++17 -c helloworld.cpp -o helloworla.o

g++-std=c++17 -c foo.cpp -o foo.o

g++-std=c++17 -o helloworld helloworld.o foo.o。

kinlevlam@IT21871254s-MacBook-Pro helloworld。 /helloworldi

欢迎来到 cSC3002

我爱它!!!!

有道文档翻译
pdf.youdao.com

结束