**一、实验说明**

**1. 环境登录**

无需密码自动登录，系统用户名 shiyanlou 。

**2. 环境介绍**

本实验环境采用 Ubuntu Linux 桌面环境，实验中会用到桌面上的程序：

1. 命令行终端：Linux 命令行终端，打开后会进入 Bash 环境，可以使用 Linux 命令；
2. Firefox 及 Opera：浏览器，可以用在需要前端界面的课程里，只需要打开环境里写的 HTML/JS 页面即可；
3. gvim：非常好用的 Vim 编辑器，最简单的用法可以参考课程[Vim 编辑器](http://www.shiyanlou.com/courses/2)；
4. gedit 及 Brackets：如果您对 gvim 的使用不熟悉，可以用这两个作为代码编辑器，其中 Brackets 非常适用于前端代码开发。

**3. 环境使用**

1. 使用编辑器输入实验所需的代码及文件，使用命令行终端运行所输入的命令进行操作。
2. “实验记录”页面可以在“我的主页”中查看，每次实验的截图、笔记以及有效学习时间（指的是在实验桌面内操作的时间，如果没有操作，系统会记录为发呆时间）都是您在实验楼学习的真实证明。

**4.预备知识**

本课程可以从零基础进行学习，如果你学过《 Linux 》和《大学生计算机》等相关课程，将有助于深入了解本课程的部分内容，但这些课程的学习不是必须的。

**代码下载**

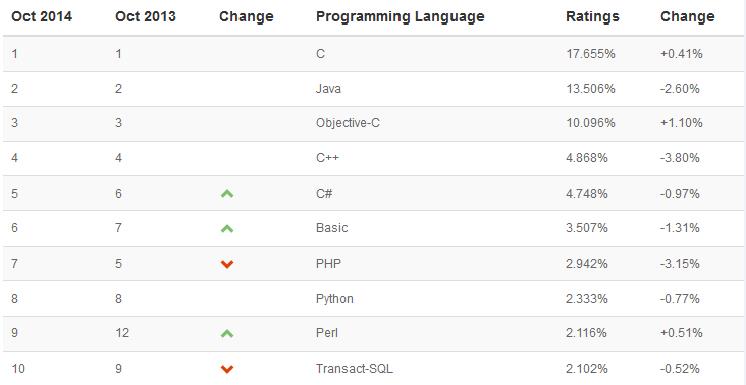
git clone http://git.shiyanlou.com/shiyanlou/c

**二、为什么是 C ——因为你无可替代**

**1. C 语言的前世今生**

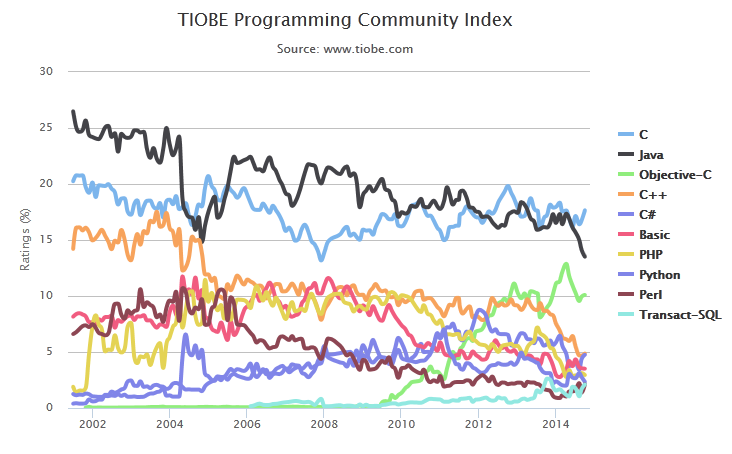
1972-1973 年间，美国[贝尔实验室](http://baike.baidu.com/view/37115.htm?fr=aladdin)为了描述和实现 UNIX 操作系统（一个具有强大影响力的操作系统，我们实验楼的实验环境 Linux 系统就是一种类 UNIX 系统）,于是改良了 B 语言，这就是今天的大名鼎鼎的 C 语言(me:C语言是由B语言发展而来)。 C 语言相当灵活，用于执行几乎所有计算机可以完成的任务，既可以用于编写应用程序，还可用于编写操作系统，伟大的 Windows 和 Linux 操作系统的内核就是使用 C 语言开发的。C 语言包含了基本的编程元素，后来的很多语言（C++、Java 等）都参考了 C 语言，说 C 语言是现代编程语言的开山鼻祖毫不夸张，它改变了编程世界。

**下面这张图是 2014 年 10 月份 TIOBE 根据编程语言在企业中的使用情况统计出来的编程语言排行榜，程序猿们可以了解自己的编程技能的价值，以便跟上编程潮流。这一排行榜亦可以作为初学者们对未来发展方向选择的参考依据。该数据每月都会更新，有兴趣的小伙伴可以去官网**[**http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html**](http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html)**查看最新动态。**



**通过该图大家可以看到 17.8% 的程序是使用 C 语言写的，虽然 C 语言是 70 后的老人，但它依然牢牢地占据在排行榜首位。**

**2. 无可替代的 C 语言**



这张图同样来自 TIOBE，通过这张图大家可以发现从 2002 年至今，C 语言在编程中占有比例一直保持稳定，java 语言却呈现下跌态势，两者还会经常争夺霸主地位。 **但是为什么 java 所占比例一直在下跌，而 C 语言却保持稳定呢？主要是因为能够代替 java 的语言不断涌现和上升，而 C 语言能保持稳定，是因为在很多领域 C 语言是唯一的选择，比如说做操作系统、做嵌入式系统。**

美国一位资深软件专家写了一篇文章，题为“对计算机学生的建议”，可供参考。他说“大学生毕业前要学好 C 语言，C 语言是当前程序员共同的语言，比你在大学学到的现代语言（比如 ML，java，python 或者其它流行的语言）都更接近机器”。他指出“不管你懂多少延续、闭包、异常处理，只要你不能解释为什么 while（\*s++=\*t++）的作用是复制字符串，那你就是在盲目无知的情况下编程，就像一个医生不懂最基本的解剖学就在开处方”。

**三、如何学习c语言**

**1. 勤动手**

C 语言并不是一门纯理论课程，而是一门应用课程。学习过程中要通过大量的例题学习怎么设计一个算法，构造一个程序。初学时一定不要在语法细节上死记硬背，从一开始就应该编写简单程序，然逐步深入。语法细节需要通过长期的编程实践才能熟练掌握。 我们所给的习题，即使它们非常简单，也要自己动手输入。自己亲自输入，以后就不容易忘记。不要害怕用代码进行实验。犯错对编程而言非常有教育性！起初犯错越多，学到的东西就越多。

**2. 勤思考**

学习程序设计，主要是掌握程序设计的思路和方法。学会使用一种计算机语言编程，在需要时改用另一种语言应当也不会太难。不要设想今后一辈子只使用大学学过的某一种语言。在学习时要善于思考，举一反三，完成一个程序需要什么样的算法、怎么去设计，要学会独立思考。

**3. 勤查找**

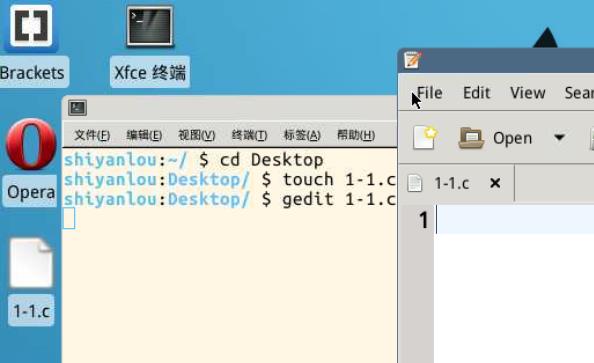
很多同学在学习过程中稍微遇到点困难就想要放弃，遇到一点内容没有懂就不往下学习了，常常半途而废。在学习过程中我们难免遇到各种困难，遇到各种难懂的词汇，遇到各种半天解不出来的题目。当碰到实在不明白的概念，可以放一放往前走。如果这些概念或者题目很重要，不懂就没办法往前走，那就好好利用下百度和 Google 吧。一个勤奋的少年，即使只给他一个 Google 或者百度，他也能学好一门语言。一个不上进的屌丝你给他买好十本 C 语言的书，找个骨灰级的程序猿带他，他也学不会 C 语言编程。

**四、第一个 C 语言程序 - Hello ShiYanLou**

在文档右边的虚拟化操作系统中，打开桌面的 Xfce 终端，输入cd Desktop**（cd命令是切换目录命令，cd Desktop表示切换目录至桌面，如果你懂这些内容当然很好，想要学习 Linux 基本命令可以学习实验楼的“ Linux 基础入门”这门课程，即使不懂也没关系，按照我说的做就可以了）**切换至桌面，紧接着输入touch 1-1.c，你便可以看到在桌面建立了一个 1-1.c 的文件。



**接下来我们输入gedit 1-1.c，系统便用 gedit 编辑器将 1-1.c 文件打开了。**



**在 gedit 里面输入以下代码，不懂不要紧，后面的课程将会给大家详细讲述。**

#include<stdio.h>

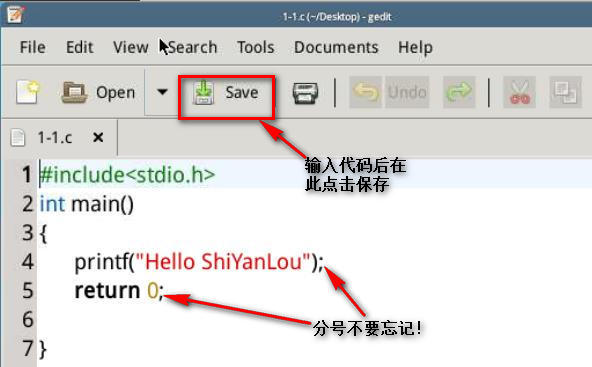
int main()

{

printf("Hello ShiYanLou");

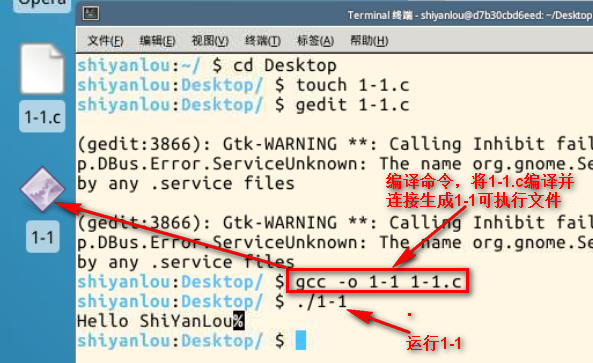
return 0；

}



**点击 save 保存，接下来就可以见证我们编写的程序运行的结果了，有木有很激动！ 在终端输入gcc -o 1-1 1-1.c后，点击回车，这时桌面上生成了一个名称为 1-1 的可执行文件，紧接着我们在终端输入./1-1来执行这个可执行文件。你的屏幕上面是不是显示出了 “ Hello ShiYanLou ”？小伙伴你表现不错，这时你已经编写一个程序并让它运行了。**

**注意：当你写完代码保存之后务必关掉 gedit 编辑器，否则无法在终端输入命令。**



**五、小结**

本节课我们只讲述了 C 语言的前世今生和它经久不衰的生命力，接下来我们将要正式进入 C 语言编程了。是不是想了解一下我们刚才编写的第一个 C 语言程序究竟是怎么执行的呢，接下来的课程将讲解编写 C 语言的开发环境以及对第一个 C 语言的剖析。

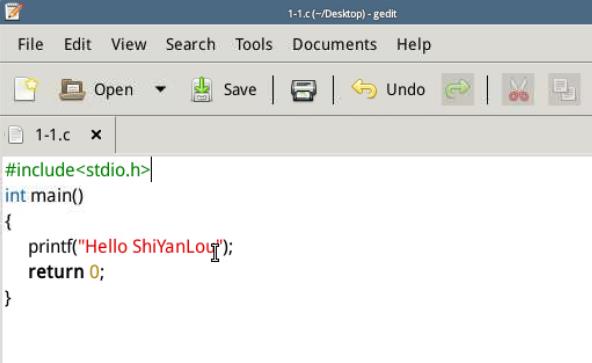
## 一、C 语言开发环境

### 1. 在 Linux 下运行 C 程序

市面上常见的 Linux 都是发行版本，典型的 Linux 发行版包含了 Linux 内核、桌面环境和各种常用的必备工具，国内使用较多的是 Ubuntu（乌班图）、CentOS 、Deepin（深度 Linux）。我们实验楼的环境采用的是 Ubuntu 的操作系统。

#### Gedit

gedit 是一个简单实用的文本编辑器，拥有漂亮的界面，支持语法高亮，比 Vim 更易上手。本教程将 gedit 作为 C 代码编辑器，当然你也可以使用 gvim 等其它编辑器。在上一节课我们已经使用过 gedit 编写代码了，相信大家并不陌生。



#### GCC

Linux 下使用最广泛的 C/C++ 编译器是 gcc，大多数的 Linux 发行版本都默认安装了此编译器，不管是开发人员还是初学者，一般都将 gcc 作为 Linux 下首选的编译工具。本教程毫不犹豫地使用 gcc 来编译 C 程序 （编译概念不懂也不要紧，我们马上就会讲述）。

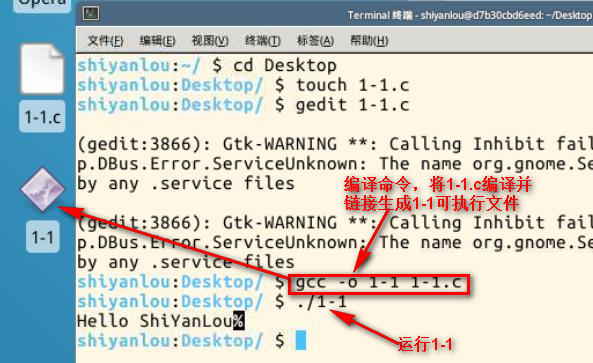
程序编写完成后保存文件并退出，打开终端并 cd 到当前目录，上一节我们写第一个 C 语言程序在终端输入下面的命令时，便调用了 gcc 编译器：

gcc -o 1-1 1-1.c

此操作可以直接将 1-1.c 文件编译链接为可执行文件 1-1 。

gcc –help查看gcc的命令帮助；gcc –o 指定编译后的文件名

可以看到在当前目录下多出一个文件 1-1，这就是可执行文件。不像 Windows，Linux 不以文件后缀来区分可执行文件，Linux 下的可执行文件后缀理论上是可以任意更改的。



### 2. 在 Windows 下运行 C 语言程序

Windows 下的 C 编程常常使用 VC 6.0 或者 Dev C++ 5.7.1，VC 6.0 是一个集成开发环境（ IDE ），简单来讲就是编程、编译、链接和执行集成在了一个软件上面。在我们实验楼暂且不需要你去学习在 Windows 下的 C 语言编程。当然，如果你想要了解如何在 Windows 下编程 C 语言，可以查看该教程[VC 6.0使用教程](http://jingyan.baidu.com/article/f0e83a25c24f8a22e591019f.html)，网上也有非常多的其它相关教程可以供你学习参考。

## 二、剖析第一个 C 语言

**接下来我们将以上节课编写的名称为 1-1 的 C 程序为材料来讲解 C 程序的创建过程以及 C 语言的程序结构。**

### 1. 创建 C 程序

C 程序的创建过程有四个基本步骤：

1. 编辑；
2. 编译；
3. 链接；
4. 执行。

#### 编辑

编辑的过程就是创建和修改 C 程序的源代码——我们使用 gedit 编写的程序便是源代码。其实在 Linux 上，最常用的文本编辑器是 Vim 编辑器，为了让大家更便捷地理解 C 语言我们采用了 gedit 编辑器，有兴趣的小伙伴可以在我们实验楼学习一下 Vim 编辑器。

#### 编译

编译器可以将源代码转换成机器语言，在编译过程中，会找出错误并报告。这个阶段的输入是在编辑期间产生的文件，常称为源文件。 编译器能找出程序中很多无效的和无法识别的错误，包括结构错误，例如程序的某个部分永远不会执行。编译器输出的结构叫作对象代码，存放它们的文件叫作对象文件。在 Linux 中这些文件的扩展名通常是 .o，在 Windows 下面这些文件的扩展名通常是 .obj 。如果编译成功就会生成一个文件，它与源文件同名。但扩展名为 .o 或者 .obj 。 其实我们在上一节编写第一个 C 语言程序的时候，使用的 gcc -o 1-1 1-1.c 这条命令既包含了编译也包含了链接，所以直接生成了可执行文件 1-1 。 在 Linux 下编译是在源代码文件所在目录输入以下命令（假如源代码文件是上节课的 1-1.c）：

gcc -c 1-1.c

-c代表的是编译

这时源文件所在的目录将会生成 1-1.o 的文件。

#### 链接

链接器将源代码文件中由编译器产生的各种对象模块组合起来，再从 C 语言提供的程序库中添加必要的代码模块，将它们组合成一个可执行文件。链接器也可以检测和报告错误，例如程序中引用了一个根本不存在的库组件。链接一旦成功，就会生成可执行文件，在 Windows 下面可执行文件的扩展名是 .exe，在 Linux 下面，可执行文件没有扩展名，但它的文件类型是可执行的。 在编译生成 .o 文件的基础上我们将会输入以下命令（以编译生成 1-1.o 为例）：

gcc -o 1-1 1-1.o

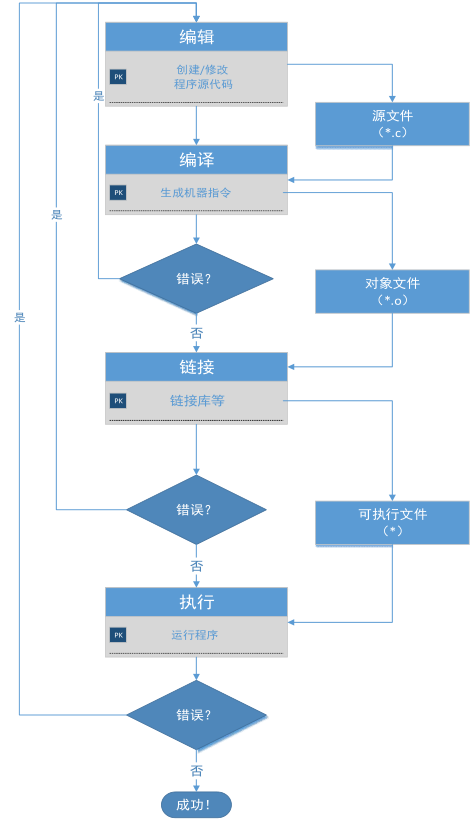
这时 1-1.o 所在的目录将会生成 1-1 可执行文件。 **多数情况下，我们是通过gcc -o 1-1 1-1.c一次性完成编译和链接。**

#### 执行

执行阶段就是成功完成了前述的三个过程后，运行程序。但是这个阶段可能会出现各种错误，包括输出错误，计算机什么也不做哦，甚至是计算机崩溃。无论如何，都需要我们返回编辑阶段，检查并修改源代码。相信大家都还记得上一节课的执行命令，在文件所在目录执行：

./1-1

**创建 C 程序的各个过程：**



### 2. C 语言的简单结构

#### 预处理指令

1-1.c 的第一行代码如下：

#include<stdio.h>

严格地说，它不是可执行程序的一部分，但它很重要，事实上程序没有它是不能执行的。符号 # 表示这是一个预处理指令，告诉编译器在编译源代码之前，要先执行一些操作（me:这些操作是在编译代码之前执行的）。编译器在编译过程开始之前的预处理阶段会处理这些指令。预处理指令的类型相当多，大多放于程序源文件的开头。

在这个例子中，编译器要将 stdio.h 文件的内容包含进来，这个文件被称为头文件，因为通常放在程序的开头处。在本例中，头文件定义了 C 标准库中一些函数的信息，本例要用到标准库中的 printf（）函数，所以必须包含 stdio.h 头文件。stdio 是“ standard input & output ”的缩写，包含了编译器理解 printf（）以及其它输入 / 输出函数所需要的信息。C 语言所有头文件的扩展名都是 .h 。在以后的学习过程中大家会看到很多其它的预处理指令。

#### main（）函数

int main()

{

printf("Hello ShiYanLou");

return 0;

}

main（）函数是“主函数”。每个 C 程序都由一个或多个函数组成，但每个 C 程序都必须有一个 main（）函数——因为每个程序总是从这个函数开始执行。

运行时运行环境会调用main函数，就像java程序运行时jvm会调用main函数一样

程序的几乎全部工作都是由各个函数分别完成的，函数是 C 程序的基本单位，其他oop语言中的基本单位是class，或者准确的说是object，在设计良好的程序中，每个函数都用来实现一个或多个特定的功能。

一个 C 语言程序由一个或者多个函数组成，其中必须包含一个 main（）函数（且只能有一个 main（）函数）。 一个函数包括两个部分：

一是 **函数首部** 即函数的第一行：

int main()

二是 **函数体** 即函数首部下面的花括号内的部分：

{

}

函数的内容我们会在以后的课程中涉及，这里只是简单地提及。

#### 程序框架

#include<stdio.h>

{

XXXX

return 0;

}

我们的课程的所有程序都需要写上这一段，直到学习函数之前，我们的代码都是放在“xxxx”的位置，所以以后你每次写程序的时候，可以先把这个框架写上去，再在这个框架中间写代码。

#### printf（）函数

printf（）是 C 编译系统提供的函数库中的输出函数。printf（）函数中双撇号内的字符串“ Hello ShiYanLou ”按照原样输出，每个语句最后都有一个分号，表示语句结束。

## 三、小结

到了这里，大家可能遇到很多不懂的地方，不要担心自己的智商不够或者实验楼的课程太烂（当然，如果课程有问题或者有不懂的地方，随时提出来），我们会在以后的课程中通过编程练习，让大家懂 C 爱上 C！

## 四、练习

### 1.

我们现在的练习还没有涉及到算法，大家在我们的环境中编写一个 C 语言小程序，实现输出“ I love shiyanlou ”就好。

### 2.

编写以下程序，输出运行结果：

#include<stdio.h>

int main（）

{

printf("23+56=%d\n",23+56);

return 0;

}

**注：%d 说明后面有一个整数要输出在这个位置上。**

## 数据类型一

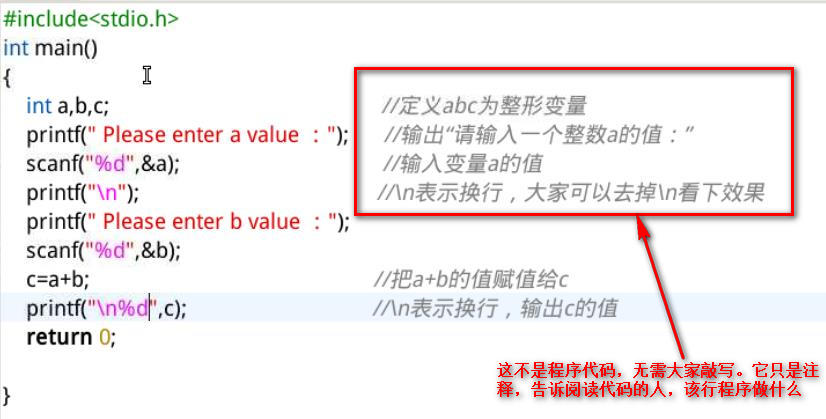
## 一、学习目标

有了前面对 C 语言的初步了解，我们现在开始深入地学习 C 语言程序设计了。 从本章开始我们将要引入大量的 C 语言程序案例，把算法和语法结合起来，通过引导大家由浅入深地编写 C 程序，让大家掌握 C 语言。也希望大家在学习的过程中勤动手，把每一个案例都自己来写一遍。我们接下来将讲述 C 语言的顺序程序设计，由于 C 语言的顺序程序设计并不容易独立成章，也需要大家具有数据类型和表达式等相关的基础知识，所以我们将顺序程序设计这一大的章节分为以下几个小节同时融入顺序程序设计的相关例题。接下来我们将讲述这些内容：

* 数据类型
* 运算符与表达式
* C 语句 + 数据的输入与输出

## 二、顺序程序设计举例

首先我们先举一个例子 3-1，在终端输入 touch 3-1.c 建立一个 C 的源程序文件，自己用 gedit 编写以下代码，编译并运行：



如果该程序成功运行，首先终端会显示 Please enter a value ： 提示大家输入 a 的值，大家写一个整数（注意是整数），然后终端会显示 Please enter b value ： 提示大家输入 b 的值，之后将会运算出 a+b 的结果。如下图输入 4 和 5 将计算出 c 的值为 9：

### 首先我们依托这个程序讲解几个知识点，之后再做分析

#### 1. printf（）函数

格式输出函数

一般形式：printf（格式控制，输出表列）。例如：printf（"%d,%d",a,b）; 括号内包含两个部分：

（1）**“格式控制”**是用双撇号括起来的一个字符串，称“转换控制字符串”，简称“格式字符串”，它包括两个信息：

* **格式声明**：格式声明由 % 和格式字符组成，如 %d （%d 代表输出整数，%f 代表输出实数），它的作用是将输出的数据转换为指定的格式然后输出。格式声明总是由 % 字符开始。
* **普通字符**：普通字符即在需要输出时原样输出的字符。例如上例中的 printf（"Please enter a value："）中的 Please enter a value 即为原样输出。

（2）**“输出表列”**是程序需要输出的数据。看下面例子：

printf("I love %d and %d",x,s);

第一个 %d 对应的是 x 的值，第二个 %d 对应的是 s 的值。 I love 和 and （注意这里包括空格）都是普通字符会原样输出。 假如 x 的值是 3，s 的值是 4，这条语句将会输出“ I love 3 and 4 ”。

#### 2. scanf（）函数

格式输入函数

一般形式：scanf（格式控制，地址表列）。**“格式控制”**的含义同 printf 函数。“地址表列”是由若干地址组成的表列，可以是变量的地址。

看下面的例子：

scanf("a=%d,b=%d",&a,&b);

在格式字符串中除了有格式声明的 %f 以外，其它普通字符原样输出（如“ a= ”，“ b= ”和“，”），假如给 a 和 b 分别赋值 5 和 6 ，将显示“ a=5，b=6 ”。 **注意：scanf（）函数中的表列是地址表列。scanf("a=%d,b=%d",&a,&b); 中 a 和 b 前面的 & 不能省掉，这一点要和 printf 作区分。**

printf（）函数和 scanf（）函数我们会在以后的“数据的输入与输出”版块继续讲述。

#### 3. 注释

位于“ /\*........ \*/ ”中的和“ // ”后面的内容为注释，用来对程序进行说明；注释在编译时会被自动忽略。

#### 这是一个简单的计算程序，通过定义变量让用户可以自由设定 a 和 b 的值，之后通过 c=a+b;这条语句实现把 a 和 b 的和计算出来并赋值给 c 。究竟什么是变量，什么是常量呢？接下来我们来一一讲述。

## 三、常量与变量

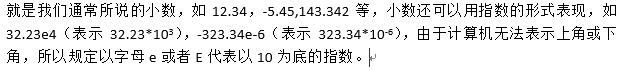
### 1. 常量

顾名思义，值不能被改变的量称为常量。如 5 、7 、19 或者 0.54 、4.33 这些值，常见的常量分为以下类型：

#### 1.1 整形常量：

如 0 、100 、-30 等整数都是整形常量。

#### 1.2 实型常量：



**注意：e 或者 E 之前必须有数字，且 e 或者 E 后面必须为整数，不能是 12e4.1 或者 e3 这种形式。**

#### 1.3 字符常量，字符常量有两种：

（1）普通字符：用单撇号括起来的一个字符，如 'a' 、'E' 、'%' 、'3' 。不能写成 'ab' 、'12' 。字符常量只能是一个字符，不包括单撇号。

（2）转义字符：除了以上形式的字符常量外，C 语言还允许用一种特殊形式的字符常量，就是以字符 \ 开头的字符序列，比如我们本节课的 3-1.c 中，\n 代表的就是换行符，显示跳转到下一行。这是一种在屏幕上无法显示的“控制字符”。

**常用的控制字符：**



现在大家不必记住这张表格，只是大概了解便可，以后每天写代码的时候自然会记住这些内容。

## 2. 变量

在例子 3-1 中的 a，b，c 都是变量。变量代表一个有名字的、具有特殊属性的存储单元。它可以用来保存数据。变量的值是可以改变的。变量在程序中定义的一般形式就是： <类型名称> <变量名称>。例如： int a； int b; int a,b; int price; int amount;等。

**int 代表定义的变量是整数类型。** 我们在 3-1.c 的例子中便是直接定义了 a，b，c 三个变量为整数类型：

int a，b，c；

### 标识符

变量需要一个名字，变量的名字便是一种“标识符”，用来区别它和其它不同的变量。用来对变量、函数、数组等命名的字符序列统称为标识符，上面提到的 price 、amount 是标识符，函数名 printf 也是一种标识符。C 语言规定标识符只能由字母、数字和下划线构成，且第一个字符必须为字母或下划线。￥qa，1ew，#22 这些都是非法的标识符。

### 2.1 变量的赋值与初始化

int price=0;

变量必须先定义后使用。这一行定义了一个变量，变量的名字是 price，类型是 int，初始的值是 0 。

**注意：和数学不同，a=b 在数学中表示关系，即 a 和 b 的值一样；而在程序设计中，a=b 表示要求计算机做一个动作：将 b 的值赋值给 a 。关系是静态的，而动作是动态的。在数学中，a=b 和 b=a 是等价的，而在程序设计中，两者意思相反。**

## 四、数据类型

我们之前的案例中讲到了整数类型 int 定义整型变量，在程序中我们还会用到浮点类型（float）来表示具有小数点的实数，讲解数据类型之前我们先来看一个用到浮点型数据的例子 3-2： **一台拖拉机耕地一亩耗油 0.85kg，它的油箱的容积是 100 升（0.1m3），柴油的密度是 850kg/m3，该拖拉机装满油后最多耕地的亩数是多少？**

我们在函数中首先要定义几个变量，定义油箱的容积的变量为浮点型类型 tank\_volume，油的密度为浮点类型 oil\_density，油箱装在油的总质量为浮点型 oil\_kg，该拖拉机装满油最多耕地的亩数是浮点型 area 。下面我们建立一个程序 3-2.c 用 gedit 打开并输入以下代码：

#include<stdio.h>

int main()

{

float tank\_volume; // 定义油箱的容积为浮点类型变量 tank\_volume

float oil\_density; // 定义油的密度为浮点类型变量 oil\_density

float oil\_kg;

float area;

tank\_volume=0.1; // 给变量 tank\_volume 赋值

oil\_density=850; // 给变量 oil\_density 赋值

oil\_kg=tank\_volume\*oil\_density; // 求 tank\_volume 和 oil\_density 的积并赋值给 oil\_kg

area=oil\_kg/0.85; // 求 oil\_kg/0.85 的商并赋值给 area

printf("Most farming is %f mu",area);

return 0;

}

代码说明： 下面我们来解释这个程序。下面的语句定义了四个变量：

float tank\_volume;

float oil\_density;

float oil\_kg;

float area;

下面这两条语句是给 tank\_volume 和 oil\_density 这两个变量赋值：

tank\_volume=0.1; // 给变量 tank\_volume 赋值

oil\_density=850; // 给变量 oil\_density 赋值

其实 tank\_volume=0.1; 与 oil\_density=850; 这两条语句可以和前面的 float tank\_volume; 与 float oil\_density; 这两条语句放在一起，定义变量的同时初始化赋值，即：

float tank\_volume=0.1;

float oil\_density=850;

接下来我们使用乘法运算符 \* 将油箱的容积和油的密度相乘，计算出油箱可以存放油的总质量：

oil\_kg=tank\_volume\*oil\_density; // 求 tank\_volume 和 oil\_density 的积并赋值给 oil\_kg

之后我们利用除法运算符 / 求出最多可耕地的亩数，并赋值给 area：

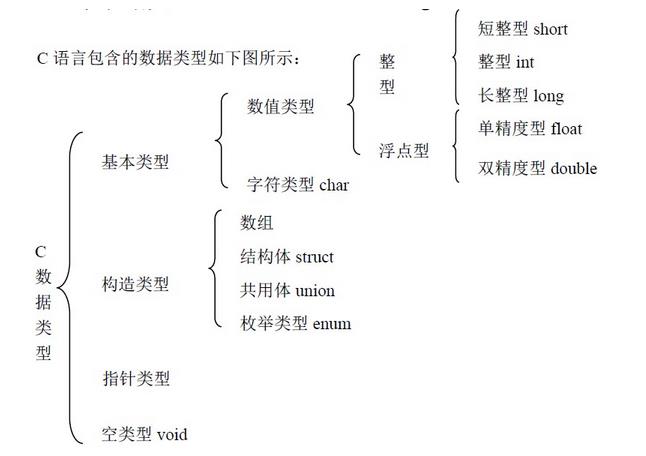
area=oil\_kg/0.85;

#### 为什么在用计算机运算时，需要指定数据的类型呢？

在数学中，数值是不区分类型的，数值的运算是绝对准确的，例如：1/3 的值是 0.33333......（循环小数）。数学是研究抽象的学科，数和数的运算都是抽象的。而在计算机中，数据是存储在计算机中的一个个单元里面，它是具体存在的（me：总之，数学中的数是抽象的，而计算机中的数是具体的，是需要具体进行运算的。）。而且，存储单元是由有限的字节构成的，每一个存储单元存储的数据是有限的，区分多种数据类型也是为了减少空间的消耗，不可能存放无限大的数，也不能存放无限循环小数，例如计算和输出 1/3：

printf("%f",1.0/3.0);

得到的结果是 0.333333，只能得到六位小数，而不是无穷位的小数。

**C 语言允许使用的数据类型见下图：** 

大家可以大致浏览该图，不需要现在背下来这些数据类型。接下来我们将要讲述基本类型里面的整型和浮点型。

### 1. 整数类型

#### 计算机内存

为了方便大家理解数据在计算机中的存储方式，我们首先给大家讲述一下计算机内存。计算机在执行程序的时候，组成程序的指令和程序所操作的数据都必须存储在某个地方，这个地方就是计算机的内存，也称为 RAM 。 可以将计算机的 RAM 想象成一排井然有序的盒子。每个盒子都有两个状态：满为 1，空为 0 ，因此每个盒子代表一个二进制数：0 或 1 。计算机有时用真和假表示它们：1 为真，0 为假。每个盒子称为一个位（bit）。每 8 个位组成一个字节，在计算机中，一个英文字母（不分大小写）占一个字节的空间，一个中文汉字占两个字节的空间。  
计算机中常用的单位是千字节（KB）、兆字节（MB）、千兆字节（GB）。 这些单位的意义如下： 1KB 是 1024 字节。其中 1024=2^10（2 的 10 次方）， 1MB=1024KB，1GB=1024MB 。如果大家对二进制、字节这些概念不是很熟悉，可以点击查看[字节](http://www.baidu.com/s?wd=%E5%AD%97%E8%8A%82&ie=utf-8&f=8&rsv_bp=1&tn=92452588_hao_pg&rsv_pq=ba04834400003dcb&rsv_t=b84fNAt7xN3JAP9MLMpbNJ5%2FLzP5ak08CohxLg1fSYX606AfE%2FDahoZxUKjZd%2FMAO1N0pCeT&bs=%E4%BD%8D)、[二进制](http://baike.baidu.com/view/18536.htm?fr=aladdin)学习了解相关概念。

#### 整数类型的分类

##### （1）基本类型（int 类型）

编译系统分配给 int 类型数据 2 个字节或者 4 个字节（由具体的编译系统自行决定，编译器也需要根据操作系统的情况而决定具体占据的字节数）。我们使用的 gcc 编译器为每个整数类型分配四个字节（32 个二进位）。在存储单元中的存储方式是：用整数的补码形式存放。所以当 4 个字节的整数类型取值范围是 -2^31~（2^31-1）。无符号的基本整型表示为 unsigned int，和 int 类型占有的字节数相同，取值范围是 0~（2^32-1）。

##### （2）短类型（short 类型）

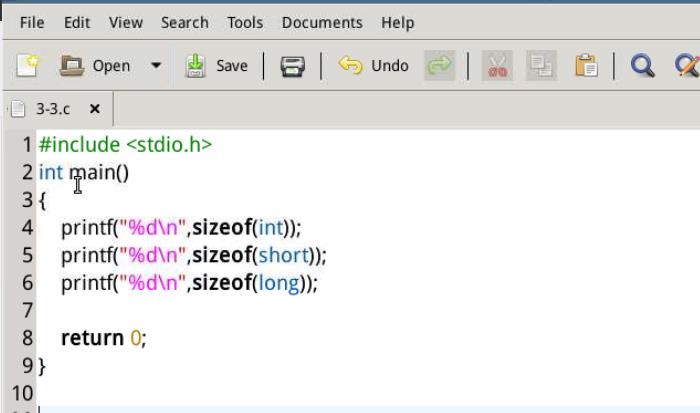
短整型的类型名为 short，gcc 编译系统分配给 short 类型 2 个字节，存储方式和 int 类型一样，也是补码的形式存储，取值范围是 -2^15~（2^15-1），无符号短整型 unsigned short 的取值范围是 0~（2^16-1）。

##### （3）长整形（long 类型）

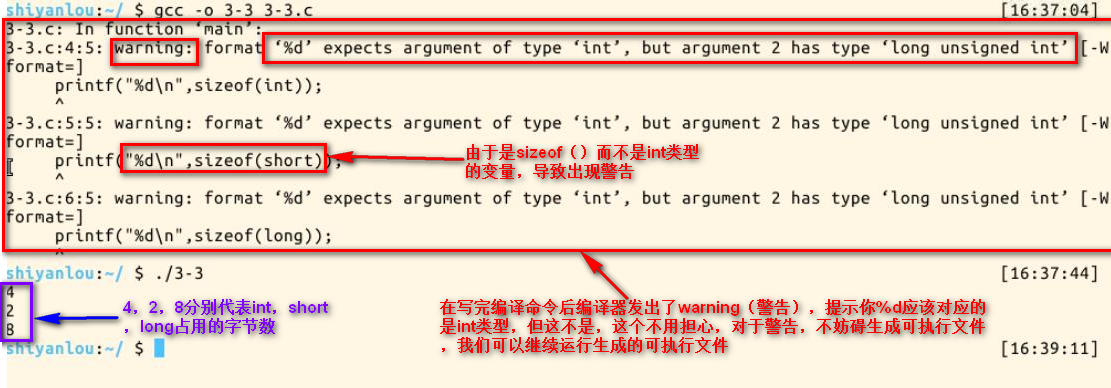
gcc 编译系统分配给 long 类型 8 个字节，存储方式和 int 类型一样，也是补码的形式存储，取值范围是 -2^63~（2^63-1），无符号长整型 unsigned long 的取值范围是 0~（2^64-1）。

#### 在这里大家可以通过 sizeof（）运算符查看各类型的常量占据多少字节：

在终端输入 touch 3-3.c 命令建立一个名称为 3-3.c 的源文件，用 gedit 打开输入以下代码：



保存后编译运行显示以下结果：



## 四、小结

我们本节课讲述了常量与变量，数据类型中的整数类型。下一节课我们将接着讲述数据类型中的浮点型和字符型，以后的课程我们会逐步增多编程的数量，小伙伴们要做好心理准备，知识的海洋等着你去遨游。

## 五、练习

追风少年小明骑电瓶车的速度是 40km/h，他以这样的速度从家到公司花费了 1 小时 30 分钟，红绿灯时间忽略不计，小明家与公司的距离是多远？

数据类型二

## 一、学习目标

上节课我们讲述了c语言的常量与变量以及数据类型中的整数类型，同时我们还涉及了一些数据存储的知识。很多小伙伴反应课程中的编程数量太少，学习起来很不带劲。小编痛定思痛，决定增加课程的实验数量。这节课我们将接着上一节课的内容，继续讲述数据类型中的浮点型和字符型。

## 二、浮点型数据

浮点型数据是用来表示具有小数点的实数的。**想知道为什么在c中把实数称为浮点数么？**在C语言中，实数是以指数的形式存放在存储单元的。一个实数表示为指数可以不止一种形式，如4.3242可以表示为4.3242\\*10^0,0.43242\\*10^1,0.043242\\*10^2,432.42\*10^-2等，他们代表同一个值。可以看到小数点的位置是可以在43242几个数字之间浮动的，只要在小数点位置浮动的同时改变指数的值，就可以保证它的值不会改变。由于小数点的位置可以浮动，所以实数的指数形式成为浮点数。

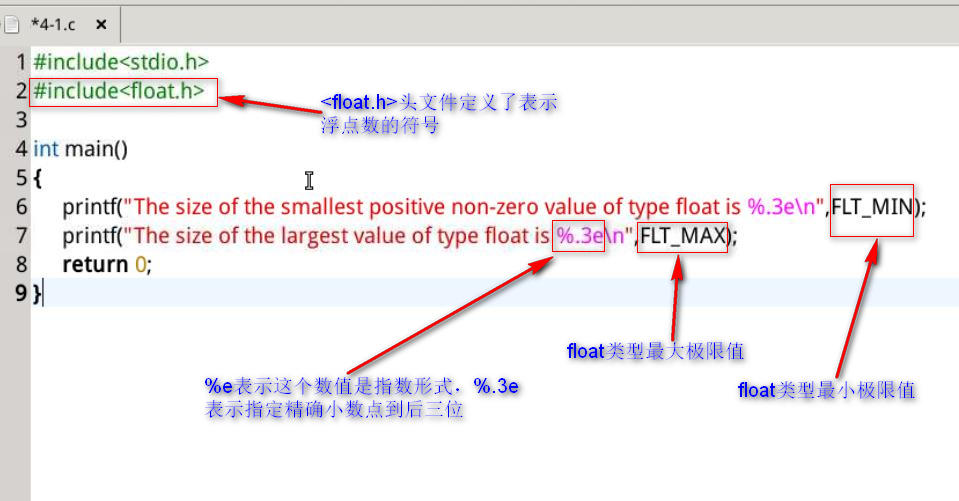
**知识小点** **规范化的指数形式：在指数形式的多种表示方式中把小数部分中小数点前的数字为0，小数点后第1位数字不为0的表示形式称为规范化的指数形式，如0.43242\*10^1就是4.3242的规范化的指数形式。一个实数只有一个规范化的指数形式。**

浮点数类型包括float（单精度浮点型）、double（双精度浮点型）、long double（长双精度浮点型）。

### 1.float型（单精度浮点型）

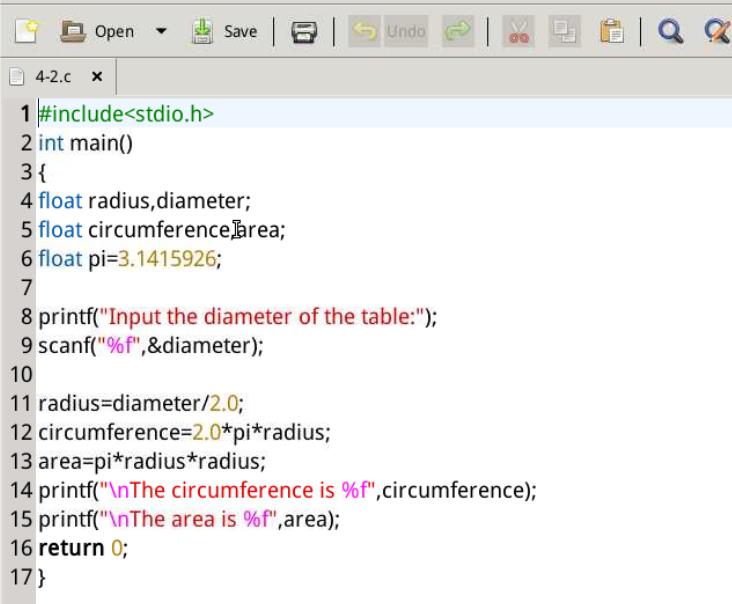
gcc编译系统为每一个float型变量分配4个字节，只有int类型是2字节或者4个字节的，这是由编译器参照操作系统决定的，数值以规范化的二进制数指数形式存放在存储单元中。在存储时，系统将实型数据分成小数部分和指数部分两个部分、分别存储，也就是说在计算机内存中浮点数的整数部分和小数部分是分别存储的。如3.14159在内存中的存放形式如下图： 

图中是用十进制来示意的，实际在计算机中是用二进制数来表示小数部分以及用2的幂次来表示指数部分的。在4个字节（32位）究竟用多少位表示小数部分，多少位表示指数部分，是由c语言编译系统自定的。

如果想要知道float的取值范围，我们可以编写4-1.c程序： 

%.3e格式指定用科学计数法来表示。

如果程序你成功运行的话，你将会看到float数据类型的取值范围。现在我们试着编写一个计算题吧！ 我们要做的是利用输入的直径计算一个圆桌的周长及面积。计算圆的周长或者面积时，数学公式要使用pi（周长=2\*pi\*r，面积=pi\*r^2,其中r是半径）。如果不记得这些公式也不用担心。这不是数学课本，所以只要理解程序是如何运行的即可。

**编写4-2.c程序：** 

### 2.double型（双精度浮点型）

为了能扩大数字的范围，用8个字节（64位）存储一个double型数据，可以得到15位有效数字，double型的数值范围大家可以按照4-1案例查看，其中double的极限值符号的下限为：DBL\_MIN,上限为DBL\_MAX（两个数字在float.h中定义）。double型的存储方式和float的存储方式相同。 浮点型数据所占的内存空间以及取值范围： 

#### 在平时我们使用的浮点型数据的时候要注意这几个点哦！

**（1）超过有效位的数字被舍去，可能产生舍入误差** 编写4-3.c程序，输入一下代码

#include <stdio.h>

int main()

{

float a , b ;

a = 123456.789e5 ; /\* 相当于 123456.789 \* 10^5 \*/

b = a+20 ; /\* 20加上无意义 \*/

printf(" %f ", b) ;

return 0;

}

其运行结果为：

12345678848.000000

为什么计算出来不是正确的结果呢？是因为float数据的有效位是7位，*a = 123456.789e5*这条语句中赋值给a的值超过了float的有效位 ;输出结果中*12345678848.000000*中只有前7位才是有效数字（7位有效位后面的位数是根据情况编译器近似的）

现在我们再次用gedit打开4-3.c,把程序修改如下：

#include <stdio.h>

int main()

{

double a , b ; //把float改为double类型

a = 123456.789e5 ;

b = a+20 ;

printf(" %f ", b) ;

retrun 0;

}

程序修改完成后保存并再次编译gcc -o 4-3 4-3.c。运行程序得到了以下结果：

12345678920.000000

这个时候我们发现运行的结果是一个正确的计算结果。因为double型的有效数字是15~16位，而123456.789e5（其实是12345678900）其有效数字是11位，把它赋值给a不会出现溢出。 在我们以后的项目编程中一定要小心数值溢出的问题，不要以为这种情况很难出现。举个简单的例子，我国很多城市的地铁造价每公里超过了5亿，我国2014年上半年国内生产总值269044亿元，这个时候如果我们定义变量为float甚至是double类型，都是很危险的。

## 三、字符型

**c语言中，字符型的基本类型符是char。字符串是字符的数组**

上一节中我们讲到了字符常量，字符型常量是用单引号括起来的一个字符。如'A','a','?'d等等。 在所有的编译系统中都规定以1个字节（8位）来存放一个字符。字符型数据在存储时，并不是把该字符本身存放到内存单元中，而是把该字符相应的ASCII码值存放到该存储单元中。（什么是ASCII码，为什么采用ASCII来存储，大家可以查看百度百科[ASCII](http://baike.baidu.com/view/15482.htm?fr=aladdin)。） 如x的十进制ASCII码是120，y的十进制ASCII码是121。对字符变量a、b赋予'x'和'y'值： a='x'; b='y'; 实际上是在a、b两个单元内存放120和121的二进制代码： 

另外还有需要注意的是，字符常量是区分大小写的，例如，字符'c'的ASCII码值是99，'C'的ASCII码值是67，两者并不是同一个字符。大家可以通过[ASCII码表](http://maxiang.info/)查看字符对应的ASCII码值。

我们编写程序4-4，代码如下

#include<stdio.h>

int main()

{

char a,b; //定义a和b为字符型变量

a='c'; //把字符常量'c'赋值给变量a

b=121;

printf("%c,%c\n",a,b); //%c表示以字符的形式输出

printf("%d,%d\n",a,b); //%d表示以有符号十进制形式输出整数型

return 0;

}

编译并运行之后的结果是：

c,y

99,121

本程序中定义a，b为字符型，在赋值语句中给a赋以字符值，但是给b却赋以整型值。从结果看，a，b值的输出形式取决于printf函数格式串中的格式符，当格式符为"c"时，对应输出的变量值为字符，当格式符为"d"时，对应输出的变量值为整数。**也就是说，一个在字符的数据既可以以字符的形式输出，也可以以整数的形式输出。以字符形式输出时，先将存储单元中的ASCII码转换成相应的字符再输出；以整数的形式输出时，直接输出其ASCII码。**

我们还可以对字符型数据进行算术运算，此时相当于对他们的ASCII码进行运算，编写程序4-5：

#include<stdio.h>

int main()

{

char a,x;

int b;

a='c';

b=1;

x=a+b;

printf("%c\n",x);

printf("%d\n",x);

return 0;

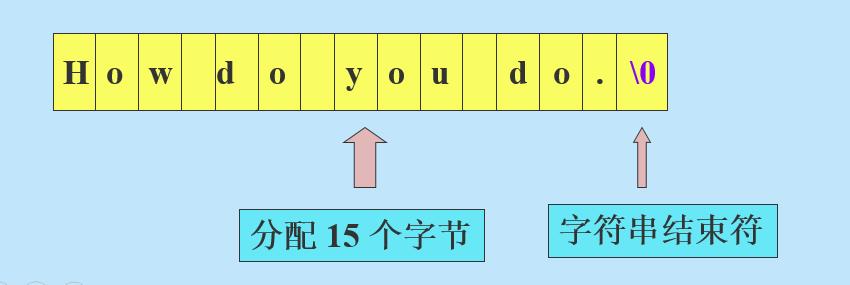
}

程序运行的结果是：

d

100

## 四、字符串常量

字符串常量是用一对双引号括起来的零个或多个字符组成的序列，所以java中存在CharSequence类，java中的字符串也是由字符数组来实现的，如"hello","China","b"都是字符串常量。 字符串常量的存储与字符常量的存储是不同的。字符串中的每个字符占用一个字节，在存储字符串常量时还要自动在其末尾加上'\0'作为字符串结束的标志，自动在字符数组末尾加上空标志代表字符串的末尾，所以字符串的长度是字符数组的长度+1。 我们先来一起看下"How do you do."是如何存储的吧！ 

因此大家不要将字符常量和字符串常量混淆哦，'b'和"b"是完全不同的。前者是字符常量，在内存中占用的字节数为1；而后者是字符串常量，在内存中占用的字节数为2，包含字符b和\0。

**注意：在c语言中没有专门的字符串变量，如果你想要将一个字符串存放在变量中，必须使用字符数组，数组中每一个元素存放一个字符，数组的内容我们会在以后的课程中和大家详细讲述**

## 五、小结

各位实验楼的小伙伴们，学完这一节课，我们的数据类型就讲解结束了，大家有没有感觉收获了很多东西。常量、变量、整形（int short long）、浮点型（float double）、字符型（char），还有字符串常量，，，，，如果你说你什么都不记得了，我建议你重温我们的故事，从第三章到这里在大概的浏览一遍吧！下面有进入了我们的答题时间了。有木有很期待！

运算符和数据转换

## 一、学习目标

小伙伴们，前两节课程我们学习了c语言的数据类型，了解了整形、浮点型、字符型和字符串的相关知识。这节课我们将要讲述c语言的运算符和表达式，各位会做很多计算题，这些将要大量的涉及到前两节课程中数据类型。学习内容包括

**1.基本的算数运算符**

**2.自增、自减运算符**

**3.不同数据类型之间的混合运算**

**4.强制类型转换**

**5.数据的输入和输出**

## 二、基本的算数运算符

**常用的算术运算符表**

* x + y：将x与y相加
* x - y：将x与y相减
* x \* y：将x与y相乘
* x / y：x除以y
* x % y：求x除以y的余数

#### 注意

1.x/y中，两个实数（亲！注意说的是实数）相除的结果是双精度实数，两个整数相除的结果为整数。如5/3的结果为1，舍去小数部分。

2.%运算符要求参加运算的对象为整数，结果也是整数。如7%3，结果为1，除了%以外的运算符的操作数都可以是任何算数类型。

下面我们举例说明上述注意事项：

**编写程序5-1.c**

# include<stdio.h>

int main()

{

int a=22;

int b=3;

printf("%d\n",a/b);

return 0;

}

**运行后之后，大家可以看到运行的结果是7而非7.33333**

**我们用gedit从新打开5-1.c作出以下修改**

# include<stdio.h>

int main()

{

float a=22.5;

int b=3;

printf("%d\n",a%b);

return 0;

}

现在我们从新编译该程序，显示出以下结果。



编译出现error表示编译未成功（出现warning表示警告，但是编译已经成功生成了可执行文件），并提示了错误内容。

**这也验证了我们在注意中提到的%运算符要求参加运算的对象为整数**

## 三、自增、自减运算符

作用是使变量的值加 1 或减 1 ，例如：++i ，--i（在使用 i 之前，先使 i 加（减）1 ）；i++，i--（在使用 i 之后，使 i 的值加（减）1 ）。

猛地一看，++i 和 i++ 的作用相当于 i=i+1 。但是 ++i 和 i++ 的不同之处在于 ++i 是先执行 i=i+1 后，再使用 i 的值；而 i++ 是先使用 i 的值之后，再执行 i=i+1 。我们看下面的例子：

**编写源程序5-2.c**

# include<stdio.h>

int main()

{

int a=5;

printf("%d\n",a++);

printf("%d\n",++a);

return 0;

}

编译之后的结果是

5

7

大家千万不要以为结果应该是

5

6

因为a++以后a的值已经成为了6，而非是5，经过++a后的值是7. 这个程序虽然简单，大家还是需要认真思考一番。

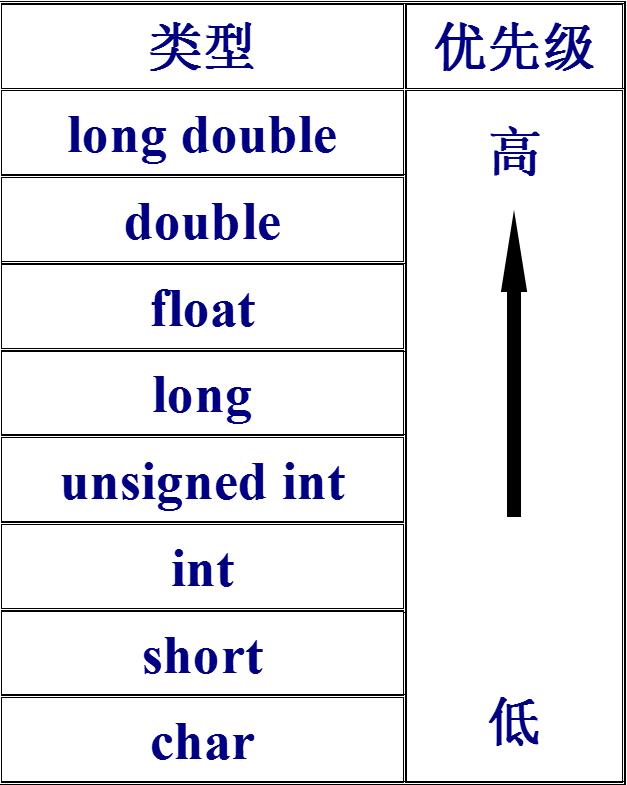
#### 注意

自增运算符（++）和自减运算符（--）只能用于变量，而不能用于常量或表达式。如5++或者（a+b）++都是不合法的。因为5是常量，常量的值是不能改变的。（a+b）++也是不可能实现，加入a+b的值是5，自增后变为6放在什么地方呢？无变量可供存放。

#### 现在大家可能还不能感受到自增和自减的用处，等我们学到循环语句和指针的时候，你将会在大量的程序中看到自增和自减。

## 四、不同数据之间的混合运算

在程序中经常会遇到不同类型的数据进行运算，比如7\*3.5。如果一个运算符的两侧数据类型不同，则先进行类型的转换，使两者具有同一种类型，然后进行运算。因此整形、浮点型、字符型数据之间可以进行混合运算。 数据类型计算的时候究竟是如何转换类型的呢？大家先看下表：



（1）如果int类型的数据和float或double型数据进行运算时，先把int型和float型数据转换为double型数据，然后进行运算，结果为double型。其他的大家可以按照上图来做。

（2）字符（char）型数据和整形数据进行运算，就是把字符的ASCII代码与整形运算。如4+'B',由于字符'B'的ASCII代码是66，相当于66+4=70。字符型数据可以直接和整形数据进行运算。如果字符型数据和浮点型数据运算，则将字符的ASCII码先转化为double型，然后在进行运算。

我们编写一个程序分析一下他们转换的编译过程，编写程序5-3.c

# include<stdio.h>

int main()

{

int i=3;

float f=4.3;

double d=7.5;

double sum;

sum=10+'a'+i\*f-d/3;

return 0;

}

在sum=10+'a'+i\*f-d/3;这条语句中，右边的表达式从左到右扫描，运算次序为：

* 10+'a'的运算，'a'的值是整数97，运算结果为107.
* i\*f的运算。先将i与f都转换为double型，两者运算12.9，double型。
* 整数107与i\*f的值相加，结果为119.9，double型。
* d/3的运算，现将3转换double类型，d/3的结果为2.5，double型
* 将119.9与2.5相减，117.4，double型

编写5-4.c，给定一个大写字母要求用小写字母输出。 **解题思路：**我们前面已经介绍过，字符数据以ASCII码存储在内存中，形式与整数的存储形式相同，所以字符型数据和其他算数型数据之间可以相互赋值和运算。 通过ASCII码表我们可以找到同一个字母的大写形式和小写形式之间有什么内在的联系。那就是同一个字母，用小写表示的字符的ASCII码比用大写表示的ASCII码大32。 编写5-4程序如下

#include<stdio.h>

int main()

{

char c1,c2;

printf("Please enter a capital letter:");

scanf("%c",&c1);

c2=c1+32;

printf("\n%c",c2);

return 0;

}

我们的程序运行如下：



## 五、强制类型转换

可以利用强制类型转换运算符将一个表达式转换为所需的类型，例如： （double）a (将a转换成为double型) （int)(x+y) （将x+y的值转换成为int类型) 其一般形式就是

**（类型名）（表达式）**

上述例子中如果写成（int）x+y 则只将x转换成为整形在与y相加。

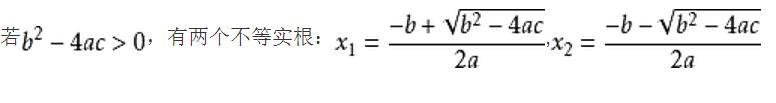
**需要说明的是，在强制类型转换时，得到一个所需类型的中间数据，而原来变量的类型未发生变化。例如： a=（int）x 如果已经定义了x为float型变量，a为整形变量，进行强制类型运算（int）x后得到一个int类型的临时值，它的值等于x的整数部分，把它赋值给a，注意x的值和类型都未变化，仍为float型，该临时值在赋值后就不再存在了。**

## 六、数据的输入和输出

之前为了方便大家学习，我们粗略的讲述过scanf和printf函数。这一小节内容，我们系统的讲述数据的输入和输出。

### 1.举例说明

求ax^2+bx+c=0方程的根。a,b,c由键盘输入，设b^2-4ac>0 **解题思路**我们知道



有了这个式子，我们只要知道a,b,c的值，就能求出方程的两个根。接下来我们编写程序5-5 ，程序如下：

#include<stdio.h>

#include<math.h> //程序中要调用求平方根函数sqrt

int main()

{

double a,b,c,disc,x1,x2,p,q;

scanf("%lf%lf%lf",&a,&b,&c);

disc=b\*b-4\*a\*c;

p=-b/(2.0\*a);

q=sqrt(disc)/(2.0\*a);

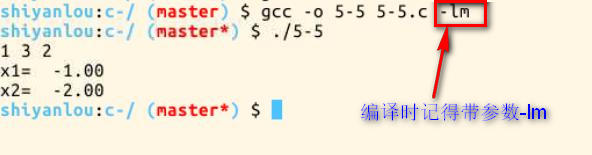
x1=p+q,x2=p-q;

printf("x1=%7.2f\nx2=%7.2f\n",x1,x2);

return 0;

}

**编译时需要注意** **在Linux系统下，Ｃ源文件若调用了math库里的函数，则编译时要加上-lm（是字母l，不是数字１），表示链接到math库。**



#### 程序分析

（1）在scanf函数中，格式声明为"%lf",表示输入的是双精度型实数。从上面的运行结果来看输入“1 3 2”时，两个数字之间是用空格分开，如果用其他符号（如逗号）会出错。我们输入的1 3 2虽然是整数，但是由于指定了用%lf的格式输入，因此系统会先把这3个整数装换成为实数1.0,3.0,2.0然后赋值给变量a，b，c。

（2）在printf函数中，不是简单地用%f格式声明，而是在格式符f的前面加了“7.2”。表示的意思是在输出x1，x2时，指定数据占7列，其中小数占2列。这样做有两个好处。 1.可以根据实际需求来输出小数的位数，因为并不是任何时候都需要6位小数的。 2.如果输出多个数据各占一行，用同一个格式输出，即使输出的数据整数部分不同，但输出时上下行必然按小数点对齐，使输出数据整齐美观。

（3）因为我们假设了a,b,c的值满足b^2-4ac>0，但是实际我们输入的abc三个值并不一定满足要求，所以我们在程序中先判断a,b,c的值是否满足b^2-4ac>0。只有满足，我们采用使用上述方法计算，判断我们将会在下一节课程中讲述。

### 2.printf和scanf中的格式字符

前面我们已经介绍了，在输入输出时，对不同的数据要指定不同的格式声明，而格式声明最重要的就是格式字符，我们在此做下总结：



**这个表不用死记，开始时会用比较简单的形式输入数据即可。在这里我们也不会详细的阐述，如果想要学习相关内容，可以买相关书籍**

**小知识点：** （1）在输入函数时，用%c格式声明输入字符时，空格字符和转义字符都是作为有效字符输入，例如：

scanf("%c%c%c",&c1,&c2,&c3);

在执行这个程序时，需要我们连续输入3个字符，中间不要有空格，如下：

abc

下面插入空格的形式是错误的：

a b c

这种形式第一个字符'a'送给c1，第二个字符是空格字符' '送给了c2，第三个字符'b'送给了c3

（2）在使用%d输出时，我们可以指定输出的宽度。 具体用法

* %d：按照整型数据的实际长度输出。
* %md：以m指定的字段宽度输出，右对齐。
* %ld：输出长整型数据。
* %mld：输出指定宽度的长整型数据。

我们编写5-6.c程序可以体验一下

#include<stdio.h>

int main()

{

int a=12,b=-3456;

long int c=123456;

printf("%5d\n",a);

printf("%d\n",b);

printf("%ld\n",c);

printf("%6d\n",c);

return 0;

}

会得到输出结果

12

-3456

123456

123456

### 3.字符数据的输入输出

除了可以用printf函数和scanf函数输出和输入的字符外，C库函数还提供一些专门用于输入和输出的字符的函数。

#### （1）用putchar函数输出一个字符

**一般形式：** putchar(c);

**功能：** 输出变量c所代表的一个字符。

**说明：** c为字符型变量或整型变量。

我们建立5-7.c程序，输入以下内容：

#include <stdio.h>

main()

{

char a,b,c;

a='O';b='K';c='\n';

putchar(a);putchar(b);putchar(c);

}

输出结果为

OK

通过此例我们可以看到：用putchar函数既可以输出能在显示器屏幕上显示的字符，也可以输出屏幕控制字符，如\n的作用就是输出一个换行符，使输出的当前位置移到下一行的开头。

#### （2）用getchar函数输入一个字符

**一般形式：** getchar（）

**功能：** 要求用户从终端（键盘）输入单个字符。

**返回值：** 从输入设备上得到的字符。

我们现在改写之前的一个练习，实现在键盘上输入一个大写字母，显示一个对应的小写字母。建立5-8.c，输入一下程序:

#include <stdio.h>

int main()

{

char c;

printf("Input an uppercase letter:\n");

c=getchar();

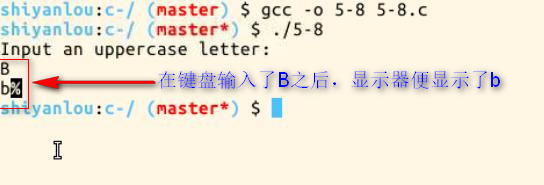
putchar(c+32);

return 0;

}

**注意：运行程序时，系统等待用户输入，注意回车也是一个合法字符。**

输出结果：



## 七、小结

到这里我们本节课程就已经结束了，在这节课我们讲述了运算符，数据类型转换以及数据的输入和输出。也是到这里我们就可以说是大家已经了解了c语言的基础知识，从下一节课，我们将要讲述c语言的选择结构。