**阅读该文档的注释说明**：**黑色加粗**：概念、思路中需要注意的地方；**蓝色加粗**：需要有附录参照的内容；**红色加粗**：C语言新标准C99的相关内容；

第一章 计算机编程语言的发展历程

一、机器语言阶段

计算机工作基于二进制，从根本上说，计算机只能识别和接受由0和1组成的指令，人们通过0和1的组合告诉计算机要做什么，然后用纸带穿孔机以人工的方法在特制的黑色纸带上穿孔，在指定的位置有孔代表1，无孔代表0，当运行某个程序时，就将纸带装在光电输入机上，当光电输入机从纸带读入信息时，有孔处产生一个电脉冲，指令变成电信号，让计算机进行各种操作。

这种计算机能直接识别和接受的二进制代码称为机器指令（machine instruction）。机器指令的集合就是计算机的机器语言（machine language）。

二、符号语言阶段

为克服机器语言的缺点，人们创造出了符号语言（symbolic language），用一些英文字母和数字表示一个指令，一般一条符号语言的指令对应转换为一条机器指令。转换的过程称为“代真”“汇编”，故符号语言又称为符号汇编语言（symbolic assembler language）或汇编语言（assembler language）。

不同型号的计算机的机器语言和汇编语言是互不通用的，是完全依赖于具体机器特性的，是面向机器的语言，因此称为计算机低级语言（low level language）。

三、高级语言阶段

为了克服低级语言的缺点，20世纪50年代创造粗了第一个高级语言——FORTRAN语言。这种语言功能很强，且不依赖于具体机器，用它写出的程序对任何型号的计算机都适用（或只需作很少的修改），它与具体机器距离较远，故称为计算机高级语言（high level language）。

计算机同样不能直接识别高级语言程序，也要进行“翻译”，用一种称为编译程序的软件把高级语言写的程序（源程序，source program）转换为机器指令的程序（目标程序，object program），高级语言的一个语句往往对应多条极其指令。

数十年来，全世界出现了2500种以上的高级语言，其中100多种较为流行：

FORTRAN和ALGOL（适合数值计算）、BASIC和QBASIC（适合初学者的小型会话语言）、COBOL（适合商业管理）、Pascal（适合教学的结构程序设计语言）、PL/1（大型通用语言）、LISP和PROLOG（人工智能语言）、C（系统描述语言）、C++（支持面向对象程序设计的大型语言）、Viasual Basic（支持面向对象程序设计的语言）、Java（适于网络的语言）。

1、高级语言的不同发展阶段：

（1）非结构化的语言

初期的语言属于非结构化的语言，编程风格比较随意，只要符合语法规则即可，没有严格的规范要求，程序流程可以随意跳转。早期的BASIC、FORTRAN、ALGOL等。

（2）结构化语言

为了解决上述问题，提出了“结构化程序设计方法”，规定程序必须由具有良好特性的基本结构（顺序结构、分支结构、循环结构）构成，程序的流程不允许随意跳转，程序总是由上而下顺序执行各个基本结构。这种程序结构清晰，易于编写、阅读、维护。例如：AQASIC、FORTRAN 77、C 语言等。

（3）面向对象的语言

以上两种语言都是基于过程的语言，在编写程序时需要具体指定每一个过程的细节。在编写规模较小的程序时，还能得心应手，但是处理规模较大的程序时，就捉襟见肘。从实践出发人们提出了面向对象的程序设计方法，程序面对的不是过程的细节，而是一个个对象，对象是由数据以及对数据进行的操作组成的。

例如：C++、Visual Basic、Java

第二章 C语言的发展概述

一、简单介绍发展时间段：

1967年，英国剑桥大学的Martin Richards推出了没有类型的BCPL（Basic Combined Programming Language）语言。

1969 年，美国的贝尔实验室的Ken Thompson和D.M.Ritchie，使用汇编语言编写了UNIX操作系统。

**1970年**，美国AT&T贝尔实验室的Ken Thompson以BCPL语言为基础，设计出了很简单且接近硬件的B语言（取BCPL的首字母），但B语言过于简单，功能有限。

**1972—1973年**，美国贝尔实验室的D.M.Ritchie在B语言的基础上设计出了C语言，C语言既保持了BCPL和B语言的优点（精炼、接近硬件），又克服了他们的缺点（过于简单、无数据类型），C语言的新特点是具有多种数据类型（字符、数值、数组、结构体、指针）。

1973 年，Ken Thompson和D.M.Ritchie合作把UNIX的90%以上用C语言改写，即UNIX第5版。（最初的C语言只是为描述和实现UNIX操作系统提供一种工作语言而设计的）

1978年以后，随着UNIX的日益广泛使用，C语言先后移植到大、中、小和微型计算机上，C语言很快成为风靡全球高级程序设计语言。

1978年，Brian W.Kernighan和Dennis M.Ritchie以UNIX第7版中的C语言编译程序为基础合著了影响深远的名著The C Programming Language，这本书中介绍的C语言成为后来广泛使用的C语言版本的基础，实际上是第一个C语言标准。

1983年，美国国家标准协会（ANSI）成立了一个委员会，根据C语言问世以来各种版本对C语言的发展和扩充，制定了第一个C语言标准草案（83 ANSI C）。

1988年，Brian W.Kernighan和Dennis M.Ritchie修订了他们的著作，按照即将公布的

ASNI C新标准。

1989年，ANSI公布了一个完整的C语言标准—ANSI X3.159-1989（常称ANSI C或C 89）。

1990年，国际标准化组织ISO（International Standard Organization）接受C89作为国际标准ISO/IEC 9899：1990，它和ANSI的C89基本上相同。

1995年，ISO对C90作了修订，即“1995基准增补1（ISO/IEC 9899/AMD1：1995）”。

1999年，ISO又对C语言标准进行修订，增加了一些功能，尤其是C++中的一些功能，命名为ISO/IEC：9899：1999。

2001年和2004年先后进行了两次修正，C99是C89（及1995基准增补1）的扩充。

二、C语言的特点

C语言的祖先是BCPL语言，开发C语言的目的在于尽可能降低用它所写的软件对硬件平台的依赖程度，使之具有可移植性。

C语言允许直接访问物理地址，能进行位（bit）运算，能实现汇编语言的大部分功能，可以直接对硬将进行操作。**既具有高级语言的功能，又具有低级语言的功能，既是成功的系统描述语言，又是通用的程序设计语言。**

C语言的函数库十分丰富，ANSI C建议提供一百多个标准库函数，不同的C编译系统除了提供标准库函数外，还增加了其他一些专门的函数，如Turbo C提供三百多个库函数，不同的编译系统所提供的库函数个数和功能是不完全相同的。

**一些软件公司提供的C编译系统并未完全实现C99建议的功能，大多以C89为基础开发。**

第三章 C语言程序的结构分析与详解

本章主要阐述一个完整的C语言程序由几部分构成并对其中的基础知识进行详细解释与叙述举例

一、C程序的结构详解

1、一个程序由一个或多个源程序文件组成（一个源程序文件可以包含3个部分）

（1）预处理指令

如：#include<studio.h>（还有一些其他预处理指令，如#define等）。C编译系统在对源程序进行“翻译”以前，先由一个“预处理器”（预处理程序、预编译器）对预处理指令进行预处理，对于 #include<studio.h>指令来说，就是将studio.h 头文件的内容读进来，放在#include指令行，取代了#include<studio.h>。由预处理得到的结果与程序其他部分一起，组成一个完整、可以用来编译的最后的源程序，然后由编译程序对该源程序正式进行编译，才得到目标程序。

（2）全局变量

即在函数之外进行的数据声明。例如：将int a,b,sum;放到main函数前面，这就是全局声明，在函数外面声明的变量称为全局变量。

如果是在程序开头（定义函数之前）声明的变量，则在整个源积程序文件范围内有效，在函数中声明的变量是局部变量，只在函数范围内有效。

（3）函数定义

2、函数是C程序的主要组成部分

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，在设计良好的程序中，每个函数都用来实现一个或几个特定的功能。

一个C语言程序是由一个或多个函数组成的，其中必须包含一个main函数（且只能有一个main函数）。

一个小程序只包含一个源程序文件，在一个源程序文件中包含若干个函数（其中有一个main函数）。当程序规模较大时，所包含的函数较多，如果把所有的函数都放在同一个源程序文件中，则此文件显得比较大，不便于编译和调试，为了便于调试和管理，可以使一个程序包含若干个源程序文件，每个源程序文件又包含若干个函数，一个源程序文件就是一个程序模块，即将一个程序分成若干个程序模块。

在进行编译时是以源程序文件为对象进行的，在分别对各源程序文件进行编译并得到相应的目标程序后，再将这些目标程序连接成为一个统一的二进制的可执行文件。

3、一个函数包括两个部分

（1）函数首部

即函数的第1行，包括函数名、函数类型、函数属性、函数参数（形式参数）名、参数类型。例如：

int max ( int x , int y)

函数类型 函数名 函数参数类型 函数参数名 函数参数类型 函数参数名

一个函数名后面必须跟一对圆括号，括号内写函数的参数名及其类型。如果函数没有参数，可以在括号中写void，也可以是空括号，如：

int main（void） int main（）

（2）函数体

即函数首部下面的花括号内的部分。如果在一个函数中包括有多层花括号，则最外层的一对花括号是函数体的范围。函数体一般包括两部分：

**声明部分**：声明部分包括定义在本函数中所用到的变量。

**执行部分**：由若干语句组成，指定在函数中所进行的操作。

在某些情况下也可以没有声明部分，甚至既无声明部分也无执行部分，如：

Void dump（）

{

}

这是一个空函数，什么都不做，但是合法。

4、程序总是从main函数开始执行的

不论main函数在整个程序中的位置如何，main函数可以放在程序最前头，也可以放在程序最后，或在一些函数之前，另一些函数之后。

5、程序中对计算机的操作是由函数中的C语句完成的

如赋值、输入输出数据的操作都是由相应的C语句实现的，C程序书写格式是比较自由的，一行内可以写几个语句，一个语句可以分写在多行上，但是为了清晰起见，习惯上每行只写一个语句。

6、在每个数据声明和语句的最后必须有一个分号

7、C语言本身不提供输入输出语句

输入输出的操作是由库函数scanf和printf等函数来完成的。C对输入输出实行“函数化”，由于输入输出操作涉及具体的计算机设备，把输入输出操作用库函数实现，就可以使C语言本身的规模较小，编译程序简单，很容易在各种机器上实现，程序具有可移植性。

8、程序应当包含注释

二、C程序的预处理

C语言提供编译预处理的功能，它允许在源程序中使用几种特殊的命令（它们不是一般的C语句）。编译系统对程序进行编译之前，先对程序中这些特殊的命令进行“预处理”，如：置换源程序文件中的特定标识符；把指定的头文件嵌入到被编译的源文件里，然后再进行编译处理，以得到目标代码。

1、宏定义指令

2、文件包含指令

3、条件编译指令

三、C语言的词汇集

C语言中使用的词汇统称为C语言的词汇集（即是C语言描述符的集合）。在C语言的词汇集中分为**6类**：标识符、关键字、运算符、分隔符、常量、注释符。

1、标识符

在计算机高级语言中，用来对变量、符号常量名、函数、数组、类型等命名的有效字符序列称为标示符（identifier）。

C语言中规定：标示符只能由字母、数字、下划线3种字符组成，且第一个字符必须为字母或下划线。

编译系统将大写字母和小写字母认为是两个不同的字符。一般而言，变量名用小写字母表示。

2、关键字

C语言规定的具有特定意义的字符串称为关键字，通常也叫做保留字。用户定义的标识符不应与关键字相同。

常见C语言关键字如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 关键字 | 含义 | 关键字 | 含义 |
| auto |  | long |  |
| break |  | register |  |
| case |  | restrict |  |
| char |  | return |  |
| const |  | short |  |
| continue |  | signed |  |
| default |  | sizeof |  |
| do |  | static |  |
| diuble |  | struct |  |
| else |  | switch |  |
| enum |  | typedef |  |
| extern |  | union |  |
| float |  | unsigned |  |
| for |  | void |  |
| goto |  | volatile |  |
| if |  | while |  |
| inline |  | \_bool |  |
| int |  | \_Complex |  |
|  |  | \_Imaginary |  |

3、运算符

C语言中有相当丰富的运算符。运算符与变量、函数一起组成表达式，表示各种运算功能。运算符由一个或多个字符组成。

C语言的运算符按照功能可分为10类；按照运算符需要的操作数的个数，可分为3类：单目运算符（一个操作数），双目运算符（两个操作数），三目运算符（三个操作数）。

**某个类别的运算符都有自己适用的数据类型。**

（1）算术运算符

用于各类数值运算，包括：加（+），减（-），乘（\*），除（/），求余（或称模运算，%），自增（++），自减（--），共7种。

1）基本算术运算符

其中加（+），减（-），乘（\*），除（/），求余（或称模运算，%）这五种运算符均是双目运算符（但是其中的加（+），减（-）可以作为单目运算符出现，即运算符的右边有数据，例如：+5，-3等，代表正数和负数。）

几点注意事项：

1、C语言中没有乘方运算，要计算a的三次方应写作a\*a\*a，或使用标准库函数pow（a，3）；

2、“/”除法运算符，若参与运算的均为整型，结果为整型，舍去小数；若参与运算的均为实型数据，结果为双精度实型；

3、同时又有一些问题，若两个整型数据相除，想要得到实型数据，则必须将其中一个进行数据类型的转换再计算；

4、若进行负数的除法运算也会出现不同结果，例如：-5/3，有的系统结果为-1，而有的系统结果为-2；

5、若出现除数为0的情况，虽然编译的时候不会出错，但是当程序运行时会出现一个被零除的错误；

6、“%”取余运算符，**要求运算对象必须是整型数据**，该运算符的功能是求两数相除的余数，余数的符号与被除数的符号相同。同时当运算中出现负数时，要注意结果的符号，例如：-15%8，商为-1，求余数结果为-7；15%-8，商为-1，求余数结果为7；

2）特殊算术运算符

C语言中两个特殊的运算符，自加运算符（++），自减运算符（--），它们都是单目运算符，运算对象可以位于运算符前面（++i，--j），也可以位于运算符后面（i++，j--）。

**二者的区别：**

1、前缀形式：先对运算对象自增（自减），再引用运算对象的值作为表达式的值；

2、后缀形式：先引用运算对象的值作为表达式的值，再对运算对象自增（自减）；

3、程序举例

#include “studio.h”

main()

{ int i=6,j=6,h=6,m,n,x,y;

m=i++;

n=++j;

x=k--;

y=--h;

printf(“ \n i=%d, m=%d, j=%d, n=%d, ”,I,m,j,n );

printf(“ \n k=%d, x=%d, h=%d, y=%d, ”,I,m,j,n );

}

运行结果：i=7,m=6,j=7,n=7

K=5,x=6,h=5,y=5

**几点注意事项：**

1、**++和—的运算对象只能是整型变量或运算结果为变量的表达式**，不能是常量或运算结果是数值的表达式；例如：5++ (a+2)++ 不合法

2、++和—具有结合性，结合方向为从右到左；

3、如果有多个运算符连续出现时，C系统会尽可能多地从左到右将字符组合成一个运算符；假如想要得到表达式：a+(++b); 假如写作a+++b;那么系统会解读为(a++)+b;那么就会出现错误，因此出现多个运算符同时出现的时候要么使用空格将其隔开，或者使用括号将相应部分括起来；

（2）赋值运算符

用于赋值运算，包括：简单赋值（=），复合算术赋值（+=，-=，\*=，/=，%=）和复合位运算赋值（&=，|=，^=，>>=，<<=），共3类11种。

1）简单赋值运算符

简单赋值作用是将一个表达式的值赋给一个变量存储起来。

**赋值运算的几项规则：**

1、赋值运算符左边的量（通常称为左值）必须是变量，不能是常量或用运算符结合起来的表达式。正确的：int a,b; a=b;b=8; 不合法： 6=a；a+b=14;

2、赋值运算符的优先级比较低，仅优先于“ ，”，结合性为从右至左；

3、**赋值表达式的值等于赋值运算符右边表达式的值，而结果的类型由赋值运算符左边的变量的类型决定；（结合函数的返回值等的理解）**

4、赋值号与数学中的等号含义不同。例如，数学中的a=b等价于b=a，而C语言中却不可等价；

2）复合赋值运算符

复合赋值运算符的作用是把“运算”和“赋值”两个动作结合起来，成为一个复合运算符。使用复合赋值运算符的目的，一是为了简化程序，二是为了提高编译效率。其十分有利于编译处理，能提高编译效率并产生质量较高的目标代码。

**赋值语句中的赋值转换规则：**

赋值转换指的是，在在赋值语句中，如果赋值运算符两边的类型不一样，但都是数值型或字符型数据，则C语言允许表达式右边的值得类型自动转换为左边变量的类型。

1、将整型数据赋给实型变量，这时数值不变，但以实型数据形式存放到变量中；

2、将实型数据赋值给整型数据，这时舍弃小数部分，取整数部分赋给整型变量；

3、将int型整型数据赋给无符号整型变量，int型数据按存储单元中实际二进制位的内容直接赋值给unsigned型变量。若int型数据为正数，即存储单元中最高位为0，赋值后，则int型数据与unsigned型数据的数值相同；若int型数据为负数，即存储单元中最高位为1，赋值后，unsigned型数据在使用时，最高位将代表一定的数值；

4、字符型数据赋给整型变量。字符型数据占1个字节，而整型变量占2个字节，因此在赋值转换中，存在位的扩展：

a、对于unsigned型变量，字符型数据（8位）传送给unsigned变量的低8位，而unsigned变量的高8位只需补0；

b、对于int型变量，若字符型数据最高位为0，则int型变量的最高8位补0；若字符型数据最高位为1，则int型变量的高8位补1，这样能够保持数值不变；

5、整型赋予字符型，只把低8位赋予字符型变量即可；

（3）逗号运算符

用于把若干表达式组合成一个表达式（，）。逗号运算符的优先级最低，结合性是自左至右，逗号表达式是指用逗号运算符将两个表达式结合起来的表达式。

1）逗号表达式的格式

表达式1，表达式2，表达式3，……，表达式n

这个表达式的功能是，先计算表达式1，再计算表达式2，……，最后计算表达式n。从左到右顺序计算各个表达式的值，**最后一个表达式的值即为此逗号表达式的值。**

2）逗号运算符的优先级

在所有运算符中，逗号运算符级别是最低的。

3）逗号表达式的值

**逗号表达式的值是组成逗号表达式的若干个表达式中最后一个表达式的值，类型也是最后一个表达式的类型。**

针对逗号表达式的这种特性，利用逗号表达式可实现在一天C语句中对多个变量赋予不同值得功能。然而，很多时候程序员并不关心逗号表达式的值，而是关心逗号表达式中各个表达式的功能。例如：a=3,b=4,c=5 这个表达式的功能是将3,4和5分别赋给a,b,c。

（4）关系运算符

用于比较运算，包括：大于（>），小于（<），等于（==），大于等于（>=），小于等于（<=），不等于（！=），共6种。

所谓的“关系运算”，实际上就是“比较运算”，即将两个数据进行比较，判断两个数据是否符合给定的关系。

1）关系运算符的作用

关系运算符的作用就是对其左、右两个操作数进行比较，测试它们之间是否具有所要求的关系。**操作数可以是数值型，也可以是字符型，但是不能对两个字符串比较大小。**关系运算符的优先级，前4种关系运算符的优先级相同，后2种的也相同，前4种高于后2种。

2）关系运算表达式的书写注意

1、关系运算符的顺序不可颠倒。例如：a=<b为错误的表达式；

2、关系运算符不允许出现空格。例如：x+y> =z为错误的表达式；

3、等号与赋值号要区分开来。例如：3==b和b==3为正确的，而a=3和3=a则有一个非法的；如果两个操作数是数值型，则按其大小比较，若为字符型数据则按照存储的ASCII值进行比较；

4、关系表达式的运算结果为“真”时，值等于“1”；结果为“假”时，值等于“0”；同时关系表达式可以参加算术运算，因为关系表达式的值为整型的“0”和“1”；

5、**关系表达式是构成选择和循环条件最重要的表示方法**；

3）关系运算与数学运算的几点区别

1、数学上，判断x是否在区间[a,b]时，习惯上写成a<=b<=c，但在C语言中，无法用单独的关系运算实现；

2、字符是按照ASCII码存储的，因此’0’>0的值为逻辑1，而’\0’>0的逻辑值为0；

3、当判断两个浮点数是否相等时，由于存储上的误差，可能会有错误的结果。例如：在数学上，恒等式1.0/3.0\*3.0=1.0，在C语言中写成1.0/3.0\*3.0==1.0的话，其表达式结果为逻辑值0；

4、在数学中，3>6>7<8无效，但C语言中是允许的，其逻辑值为0；

5、关系表达式的结果可以当成整型数；

（5）逻辑运算符

用于逻辑运算，包括：与（&&），或（||），非（！），共3种。

（6）条件运算符

这是一个三目运算符，用于条件求值（？ ：）。

（7）位操作运算符

按二进制位对参与运算的量进行运算，包括位与（&），位或（|），位非（~），位异或（^），左移（<<），右移（>>），共6种。

（8）指针运算符

用于取内容（\*）和取地址（&）两种运算。

（9）求字节数运算符

用于计算数据类型所占的字节数（sizeof）。

（10）特殊运算符

包括：括号（），下标[ ]，成员（->，. ）等几种。

（11）运算符的优先级

（12）运算中的数据转换

4、分隔符

在C语言中采用的分隔符有逗号和空格两种。逗号主要用在类型说明和函数参数表中，分隔各个变量；空格多用于语句单词之间，作间隔符。在关键字和标识符之间必须要有一个以上的空格符作为间隔，否则将会出现语法错误。

5、常量

常量：在程序中，其值不能改变的量称为常量。

常量的分类：

1. 对于字符常量很简单，只要看到是由单撇号括起来的单个字符或转义字符就是字符常量；
2. 对数值常量要按照数值的范围进行划分，主要结合后面整型数据的四种类型进行区分。
3. 整型常量。如：1000，0，12345，—345等
4. 实型常量。有两种表示形式
5. 十进制小数形式，由数字和小数点组成。如：123.345，0.123，—54.2332等
6. 指数形式，以字母e或E代表以10为底的指数，e或E之前必须有数字，且e或E后面必须为整数，如：12.34e3（代表12.34\*1000）不能写成：e4，12e2.5
7. 字符常量。有两种形式的字符常量
8. 普通字符，用**单撇号**括起来的一个字符。单撇号只是界限符，字符常量只能是一个字符，不包括单撇号，**字符常量一般以ASCII代码的形式存储在计算机**存储单元中例如：‘a’‘z’‘？’‘#’等，不能写成“ab”“12”。注意：**ASCII字符与代码对照表**。

C语言并没有指定使用哪一种字符集，由各个编译系统自行解决采用哪一种字符集，但是C语言规定：**基本字符集中的每一个字符必须用一个字节表示（即8位二进制），空字符也占一个字节**

1. 转义字符，除了以上形式的字符常量外，C还允许使用一种特殊形式的字符常量，就是以字符”\”开头的字符序列，**参照转义字符表及其作用**。

转义字符：意思是将”\”后面的字符转换成另外的意义。

1. 字符串常量。用**双撇号**把若干个字符括起来，字符串常量是双撇号之前的全部字符（但是不包括双撇号本身）

**字面常量**（直接常量），从字面形式上即可识别的常量，字面常量是没有名字的不变量。

1. 符号常量。用#define指令，指定用一个字符名称代表一个常量。（**习惯用大写表示**）

**# define PI 3.14 //注意行末没有分号**

经过上述指定后，本文件从此行开始所有的PI都代表3.14。在对程序进行编译时，预处理器先对PI进行处理，把所有PI全部置换为3.14，这种用一个符号名代表一个常量的，称为符号常量。在预编译后，符号常量已全部变成**字面常量**（3.14）。

优点：

1. 含义清楚，看到PI大致就可以联想到圆周率，在一个规范的程序中不提倡使用很多的常数，如：sum=15\*30\*23.5\*43，在检查程序时搞不清各个常数究竟代表什么，应尽量使用“见名知意”的变量名和符号常量。
2. 方便更改，一改全改。**注：**注意区分符号常量和变量，符号常量不占用内存，只是一个临时符号，在预编 译后这个符号就不存在了，故不能对符号常量赋予新值，为与变量名区别，符号常量全为大写。

### 变量

变量代表一个有名字的、具有特定属性的一个存储单元，它用来存放数据，也就是存放变量的值，在程序运行期间，变量的值是可以改变的。

变量必须先定义，后使用。在定义的时指定该变量的名字和类型。一个变量有名字方便引用，但要注意区分变量名和变量值，**变量名实际上是以一个名字代表的一个存储地址**，在对程序编译连接时由编译系统给每一个变量分配对应的内存地址。

从变量中取值时，实际上是通过变量名找到相应的内存地址，从该存储单元中取数据。Int a=3；3是变量a的值。

### 常变量

C99中允许使用常变量，const int a=3；表示a被定义为一个整型变量，指定其值为3，而且在变量存在期间其值不能改变。

对比：**常变量与常量 常变量与符号常量**

（1）常变量与常量

常变量具有变量的基本属性：由类型、占存储单元，只是不允许改变其值。

常变量是有名字的不变量；（有名字就便于在程序中被引用）

常量是没有名字的不变量；（字面常量是没有名字的不变量；（常量概念中的字符串常量））

（2）常变量与符号常量

#define PI 3.1415926 //定义符号常量

const float pi=3.1415926; //定义常变量

从使用角度来看，常变量具有符号常量的优点，而且使用方便。有了常变量以后，可以尽量少用符号常量。

6、注释符

C语言的注释符是以“/\*”开头并以“\*/”结尾的串。在“/\*”和“\*/”之间的即为注释。程序编译时，不对注释做任何处理。注释可出现在程序中的任何位置。注释用来向用户提示和解释程序的意义。在调试程序中对暂不使用的语句也可用注释符标识，使编译跳过不做处理，待调试结束后再去掉注释符。

四、C语言的C表达式与C语句

1、C语句分类

C语句的作用是向计算机系统发出操作指令，要求执行相应的操作。一个C语句经过编译后产生若干条机器指令，声明部分不是语句，它不产生机器指令，只是对有关数据的声明。

（1）控制语句

控制语句用于完成一定的控制功能，C语言有9种控制语句：

1）if（ ）~~else~~ 条件语句

2）for（ ）~~ 循环语句

3）while（ ）~~ 循环语句

4）do~~ while（） 循环语句

5）continue 结束本次循环语句

6）break 中止执行switch或循环语句

7）switch 多分支选择语句

8）return 从函数返回语句

9）goto 转向语句，在结构化程序中基本不用goto语句

（2）函数调用语句

函数调用语句由一个函数调用加一个分号构成

例：printf（“This is a c statement”）；其中printf（“This is a c statement”）是一个函数调用，加一个分号成为一个语句。

（3）表达式语句

表达式语句由一个表达式加一个分号构成，最典型的是，有赋值表达式构成一个赋值语句。例如：a=3是一个赋值表达式，而a=3；是一个赋值语句。可以看到**一个表达式的最后加一个分号就成了一个语句**。一个语句必须在最后有一个分号，分号是语句中不可缺少的组成部分，而不是两个语句间的分隔符。i=i+1（是表达式，不是语句）i=i+1；（是语句）；同样x+y；也是一个语句，作用是完成x+y的操作，它是合法的，但是并不把x+y的和赋值给另一变量，所以它并无实际意义。

表达式构成语句是C语言的一个重要特色，其实“函数调用语句”也属于表达式语句，因为函数调用（如sin（x））也属于表达式一种。只是为了便于理解和使用，才把“函数调用语句”和“表达式语句”分开来讲述。

（4）空语句

下面的语句是一个空语句： ； 此语句只有一个分号，它什么也不做。但它又有其他作用：可以用来作为流程的转向点（流程从程序其他地方转到此语句处）；也可以用来作为循环语句中的循环体（循环体是空语句，表示循环体什么也不做）。

（5）复合语句

可以用{ }把一些语句和声明括起来成为复合语句（又称语句块）。可以在复合语句中包含声明部分，C99允许将声明部分放在复合语句中的任何位置，但是习惯上把它放在语句块的开头位置，复合语句常用在if语句或循环中，此时程序要执行一组语句。

注：复合语句中最后一个语句中最后的分号不能忽略不写。

五、C语言的库函数

1、标准输入输出函数

第四章 C语言的数据类型

一、数据类型总述

所谓类型，就是对数据分配存储单元的安排，包括**存储单元的长度**（占多少字节）以及**数据的存储形式**，不同的类型分配不同的长度和存储形式。

1、为什么必须指定数据类型？

数学是抽象的；计算机时具体的。在数学中，数值是不分类型的，数值的运算是绝对准确的，例如：78与97之和为175；1/3的值为0.33333…….（无限循环小数）。数学是一门研究抽象的学科，数和数的运算是抽象的。

在计算机中，数据是存放在存储单元中的，它是具体存在的。而且，存储单元是由有限的字节构成的，每一个存储单元中存放的数据的范围是有限的，不能存放无穷大也不能存放循环小数。

用计算机进行的计算不是抽象的理论值的计算，而是用工程的方法实现计算，在许多情况下，只能得到近似的结果。

2、数据类型的存储形式

（1）整型数据

以整数的补码形式存储。

（2）字符型数据

按照其代码（整数）形式存储，即以ASCII码编号的形式存储。

（3）实数类型（浮点型）

以指数形式存储在存储单元中。

（n）关于C99的新规定的解释

**在新的C标准C99中，由于字符是按其代码（整数）形式存储的，因此C99中把字符型数据作为整数类型的一种。**

之所以如此必然有一定的道理，在此只是谈一下自己的看法：

关于**整型数据**的确是将其数值经过二进制计算后按照01组合存储在内存中，在其需要输出的时候，则进行反向运算即可得到该整型数据的值。

对于**字符型数据**，则是一写具体的图形（ASCII字符集）对应一批编码，想要存储某一个图形的时候则要存储相应的代码，由于其代码为0—255的整数范围，想要存储形式为整数的代码则仍需要按照整型数据的存储方式进行存储，在存储之后若要输出则需进行反向操作，最终得到想要的图形（ASCII代码）。

正因为对**字符型数据**的存储是按照整型数据的存储方式进行的，某个图形与其相对应的整数代码是统一的，那么自然就可以进行整型数据与字符型数据的算术运算了。

当然后面的**布尔型数据**归纳到整型数据中也有这方面的考虑

3、C语言的数据类型表

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  | | | 基本整型(int) | |
|  |  |  | | | 短整型(short int) | |
|  |  |  | | | 长整型(long int) | |
|  |  | 整型类型 | | | 双长整型(long long int)(C99新加) | |
|  | 基本类型 | （整型数据） | | | 字符型(char) | |
|  |  |  | | | 布尔型(bool) (C99新加) | |
|  |  |  | | |  | |
|  |  |  | | | 单精度浮点型(float) | |
|  |  | 浮点类型 | | | 双精度浮点型(double) | |
| 数据类型 |  | （浮点型数据） | | | 复数浮点型(float-complex，double-compe，long long-comple) | |
|  |  |  | |  | | |
|  | 枚举类型(enum) | |  |  | | |
|  |  |  | |  | | |
|  | 空类型(void) | |  |  | | |
|  |  |  | |  | | |
|  |  | 指针类型(\*) | | | |  |
|  |  | 数组类型([ ]) | | | |  |
|  | 派生类型 | 结构体类型(struct) | | | |  |
|  |  | 共用体类型(union) | | | |  |
|  |  | 函数类型 | | | |  |

注释：基本类型和枚举类型变量的值都是数值，统称为**算术类型**（arithmetic type）

算术类型和指针类型，因为其变量的值是以数字来表示的，统称为**纯量类型**（scalar type）

枚举类型是程序中用户定义的**整数类型**。

数组类型和结构体类型统称为**组合类型**（aggregate type）

共用体类型不属于组合类型，因为在同一时间内只有一个成员具有值。

函数类型用来定义函数，描述一个函数的接口，包括函数的返回值得数据类型和参数类型。

二、整型类型

按照前面图表的分类整型类型包括：基本整型、短整型、长整型、双长整型、字符型、布尔型，总共6个小类。

探讨数据类型要从**两个方面**着手，包括存储单元的长度（占多少字节）以及数据的存储形式，下面对各个数据类型根据编译系统的不同来阐述这两个方面。

1、基本整型（int）

在**Turbo C 2.0**中为每一个整型数据分配2个字节**（16个二进制位）**，那么存储单元中能存放的最大值为0111 1111 1111 1111，第1位为0代表正数，后面15位全为1，此值为（2e15）—1，即十进制的32767，最小值为1000 0000 0000 0000，此数为—2e15，即—32768，超过此范围将溢出。

在**Visual C++**中为每一个整型数据分配4个字节**（32位）**，其能容纳的数值范围为

—2e31……（2e31）—1，即—2147483648….2147483647。

**关于存储方式：用整数的补码（complement）形式存放。**

一个正数的补码是此数的二进制形式，如5的二进制形式是101，若用两个字节存放一个正数，则在存储单元中的形式为：5的补码：0000 0000 0000 0101

一个负数的补码，先要将此数的绝对值写成二进制形式，然后对其后面所有各二进制位逐位取反，再加1。—5的补码：

5的原码： 0000 0000 0000 0101

逐位取反： 1111 1111 1111 1010

再加1： 1111 1111 1111 1011

2、短整型（short int）

类型名为short int或short。

Visual C++6.0，编译系统分配给基本整型（int）4个字节，短整型（short）2个字节。存储方式与int型相同。范围是—32768…. 32767

3、长整型（long int）

类型名为long int或long。

Visual C++6.0，编译系统分配给long数据4个字节，范围是—2e31……（2e31）—1，即—2147483648….2147483647

在一个整数的末尾加大写字母L或小写字母l，表示它是长整型（long int）。

4、双长整型（long long int）

**类型名为long long int或long long。**

**这是C99中新增的类型，一般分配8个字节，但是许多C编译系统尚未实现。**

小结1：关于以上四种类型的

C标准没有具体规定各种类型数据所占用存储单元的长度，这是由编译系统自行定义决定的。C标准只是要求long型数据长度不短于int型，short型不长于int型。

即：sizeof（short）<=sizeof（int） <=sizeof（long）<=sizeof（long long）

在Turbo C2.0 中int和***short型数据都是2个字节***（16位），而**long型数据是4个字节**（32位）。

在Visual C++ 6.0 中，***short数据的长度为2字节***，int数据的长度为4字节，**long数据的长度为4字节**。

在通常做法中，把long定为32位，把short定为16位，而int可以是16位，也可以是32位。

小结2：整型变量的符号属性

在实际中，有的数据的范围只有正值（学号、年龄。。。），为了充分利用变量的值的范围，可以将变量定义为“无符号”类型，加上unsigned修饰符，表示指定该变量是“无符号整数”类型。如果加上signed修饰符，则是“有符号类型”。

由此可以将以上4种整型数据扩展为以下8种整型数据：

|  |  |  |
| --- | --- | --- |
| 类型 | 字节数 | 取值范围 |
| [signed] int（基本类型） | 2 | —2e15…..（2e15）—1 |
| 4 | —2e31…..（2e31）—1 |
| unsigned int （无符号基本整型） | 2 | 0…..（2e16）—1 |
| 4 | 0…..（2e32）—1 |
| [signed] short（短整型） | 2 | —2e15…..（2e15）—1 |
| unsigned short（无符号短整型） | 2 | 0…..（2e16）—1 |
| [signed] long（长整型） | 4 | —2e31…..（2e31）—1 |
| unsigned long（无符号长整型） | 4 | 0…..（2e32）—1 |
| [signed] long long（双长型） | 8 | —2e63…..（2e63）—1 |
| unsigned long long（无符号长整型） | 8 | 0…..（2e64）—1 |

注：若未注明signed或unsigned，默认为“有符号类型”

有符号整型数据存储单元中最高位代表符号（0为正，1为负），若指定unsigned（无符号）型，则存储单元中全部二进制位都用作存放数值本身，而没有符号，不存放负数。

由于左边最高为位不再用来表示符号，用来表示数值，因此无符号整型变量中可以存放的正数的范围比一般整型变量中正数的范围扩大一倍。

只有整型（包括字符型）数据可以加signed或unsigned修饰符，实型数据不能加。

**特例：**

在将一个变量定义为无符号整型后，不应向它赋予一个负值，否则会得到错误的结果，如：

unsigned short price=—1；

printf（“%d\n”，price）；

得到的结果为65535，显然与愿意不符。

原因：系统对—1先转换成补码形式，就是全部二进制位都是1，然后把它存入变量price中，由于price是无符号短整型变量，其左面第一位不代表符号，按“%d”格式输出，就是65535。

6、布尔型（bool）

先录入一个小区别C与Java，在Java中bool类型不可以和整型类型进行转换。true==0；false==1；这种类似应用不可以。

三、浮点类型

C语言中将实数称为浮点数？在C语言中，实数是以指数形式存放在存储单元中的。一个实数表示为指数可以有不止一种形式（3.1415926；0.31415926\*10e1；0.031415926\*10e2。。。。。等），小数点的位置可以在31415926几个数字之间和之前或之后浮动，只要在浮动的同时改变指数的值即可。

由于小数点位置可以浮动，所以实数的指数形式称为**浮点数**。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的表现形式称为**规范化的指数形式**。

在程序以指数形式输出一个实数时，**必然以规范化的指数形式输出**，如：0.31415926e001。

1、单精度浮点型（float）

编译系统为每一个float型变量分配4个字节，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，分别存放。小数部分的小数点之前的数位0；（符合规范化指数形式）

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。

**（解释：在利用规范化的指数形式存储一个实数时，将小数点以后的数值化作二进制形式，存储在编译系统分配的相应位数中，然后将指数形式中10的幂次化作二进制形式，存储在编译系统给定的二进制位中。）**

**继续解释：**

在IEEE754标准中进行了单精度浮点数(float)和双精度数浮点数(double)的定义。float有32bit，double有64bit。它们的构成包括符号位、指数位和尾数位。   
这些位的构成如下：   
种类-------符号位-------------指数位----------------尾数位----   
float---第31位(占1bit)---第30-23位(占8bit)----第22-0位(占23bit)   
double--第63位(占1bit)---第62-52位(占11bit)---第51-0位(占52bit)   
取值范围主要看指数部分：   
float的指数部分有8bit(2^8)，由于是有符号型，所以得到对应的指数范围-128~128。   
double的指数部分有11bit(2^11)，由于是有符号型，所以得到对应的指数范围-1024~1024。   
由于float的指数部分对应的指数范围为-128~128，所以取值范围为： -2^128到2^128，约等于-3.4E38 — +3.4E38   
**精度(有效数字)主要看尾数位： float的尾数位是23bit，对应7~8位十进制数，所以有效数字有的编译器是7位，也有的是8位**

在4个字节（32位）中，究竟有多少位来表示小数部分，多少位来表示指数部分，C标准并无具体规定，由各个C语言编译系统自定：有的以24位表示小数（包括符号）；以8位表示指数部分（包括指数的符号）

可以在表示的实数后面加上字母F或字母f，表示是float类型

float a=3.1415926f；

float类型数据能得到**6位有效数字**，数值范围为-3.4E38 — +3.4E38

2、双精度浮点型（double）

为了扩大能表示的数值范围，用8个字节存储一个double型数据，可以得到15位有效数字，数值范围为—1.7e—308~~1.7e308；在C语言中进行浮点数的算术运算时，将float型数据都自动转换为double型，然后进行计算。

3、长双精度浮点型（long double）

不同的编译系统对long double型处理方法不同，Turbo C对long double型分配16个字节；而Visual C++ 6.0则对long double型和double型一样处理，分配8个字节。

4、复数浮点型（float\_complex,double\_comple,long long\_comple）

关于浮点型（实数）数据的数据范围总结

|  |  |  |  |
| --- | --- | --- | --- |
| 类型 | 字节数 | 有效数字 | 数值范围 |
| float | 4 | 6 | —3.4e38~~—1.2e—38；0；1.2e—38~~3.4e38 |
| double | 8 | 15 | —1.7e308~~—2.3e—308；0； 2.3e—308~~1.7e308 |
| long double | 8 | 15 | —1.7e308~~—2.3e—308；0； 2.3e—308~~1.7e308 |
| 16 | 19 | —1.1e4932~~—3.4e—4932；0；3.4e—4932~~1.1e4932 |

**注释：**由于用有限的存储单元存储一个实数，不可能完全精确的存储，例如float型变量能存储的最小正数位1.2e—38，不能存放绝对值小于此值的数，例如：1.2e—40，因此才会出现上面数值范围中的三段表示。

四、枚举类型

五、空类型

六、派生类型

1、指针类型

2、数组类型

3、结构体类型

4、共用体类型

5、函数类型

第五章 C语言的控制结构

第六章 C语言的函数定义

第七章 C语言的存储管理

第八章 C语言的文件操作

第N章 C语言的经验总结