第一篇

认识篇

1 C 语言的特点与程序结构

计算机语言是学习和了解计算机知识的工具，已经是毋容置疑的公理。C 语言作为高级语言中的低级语言，在软件设计和开发中起着重要的作用，其学习和掌握有其独特的规律。本章介绍一些学习和掌握 C 语言的入门知识。

1.1 C 语言的特点

学习 C 语言，首先应该了解 C 语言的特点。这可以从以下两个方面入手。

1.1.1 C 语言的发展历史

C 语言不是历史最悠久的高级语言,却是生命力最旺盛的高级语言之一。

早期 C 语言的发展是围绕能否与硬件打交道的问题的。人们希望能找到一种取代汇编语言、能够编写操作系统的语言。C 语言是在当时一些已经使用的高级语言 ALGOL 等的基础上发展起来的。其历史可简单描述如下：

1960: ALGOL-60->面向问题的语言,远离硬件。

1963: CPL->在ALGOL-60 基础上,向硬件接近,但规模庞大难以实现。

1967: BCPL- >简化了 CPL。

1970: B 语言->简化 BCPL，基本能够与硬件打交道，产生了简单的 UNIX 操作系统。

1973: C 语言->完善了 B 语言，但移植性较差。

1977: 可移植的 C语言问世。

从此，C 语言开始风靡全球。

1.1.2 C语言的编程特点

从编程设计来讲，C语言具有如下特点：

1.1.2.1 符合结构化、模块化程序设计规范

结构化模块化是 20 世纪 60 年代中期提出的一种程序设计规则，它要求程序按照模块进行结构设计,每个模块具有相对独立性。按照这种要求设计的程序可读性好，移

植性好。由于 C 语言是在 60 年代中后期发展起来的。因此，它一问世就必然满足这种规则。

1.1.2.2 基本数据简单，数据处理内容丰富，范围广泛

一种语言的数据处理能力确定了这种语言的应用范围。虽然 C 语言是同 PASCAL,BASIC 等高级语言一样的高级语言，但是 C 语言具有很强的数据处理能力。需要说明的是，C 语言直接处理的基本数据很简单（只有整数型、实数型、字符型、指针型和 void 类型),但是 C 语言通过一些规则将这些数据扩充成许多丰富的类型。这与 PASCAL,BASIC 等其他语言不一样。PASCAL、BASIC 的基本数据类型多、而扩充处理的灵活性较差。

1.1.2.3 运算丰富,支持简单明了的运算符

与其他语言相比，C 语言的运算符和表达式要简单多了。表 1-1 是 C 语言与 PASCAL 的简单比较。

表 1-1

C 语言的表达式	PASCAL 的表达式
x=y+z	x:=y+z
x++	x:=x+1
x&&y	(x<>0) and (y<>0)

虽然表达式简单，但是 C 语言支持的运算有 34 种之多。

1.1.2.4 功能强大,能进行多种低级操作

C 语言是除了汇编语言以外功能最强大的语言。它能实现许多低级操作如对磁盘、打印机、I/O 端口、鼠标等直接操作。利用 C 语言能编写控制硬件的程序对一些硬件进行操作。例如，用 C 语言编制一些对磁盘操作的程序如创建磁盘目录，删除磁盘文件，修改文件属性，读写磁盘物理扇区等的程序是非常方便的。而像 FORTRAN、FOXPRO 等高级语言则基本难以做到。

1.1.2.5 目标码质量高,可移植性好

C 语言的编译程序是高效编译系统，生成的目标代码质量高。经过实验统计，C 语言编译的目标码只比汇编目标码的效率略低而高于其他所有高级语言的目标码。C 语言生成的目标码可以与其他语言链接生成可执行文件，其移植性能极好。

1.1.2.6 丰富多彩的系统工具、库函数

所有的 C 语言系统都有一个共通的特点 提供大量的库函数。现在的 C 系统都拥有几百个库函数。利用这些库函数，程序设计者能够开发丰富多彩的软件。

1.1.2.7 能培养具有良好素质的程序员

不像 PASCAL 那样，编译程序对源程序进行严格的语法检查，在编译时就把一些可能潜在的错误全部排除。C 语言的编译程序对一些语法不作严格的检查。例如，如果我们定义了一个长度为 10 的数组，用这个数组作一个函数的参数，则下面的语句：

```
void myfunc(int array[]);
void myfunc(int array[10]);
void myfunc(int array[100]);
```

在 C 语言里全部能够通过编译，而在 PASCAL 里，只有第二个句子能够通过。

因此像这样的问题，需要编程的入在设计时作好总体规划工作，在实时编程时要心中有数。长期下去，自然会养成良好的习惯，提高素质。

本节要点：

- C 语言不是历史最悠久的高级语言，确是生命力最旺盛的高级语言之一。
- 符合结构化、模块化程序设计规范。
- 基本数据简单,数据处理范围广泛丰富。
- 运算丰富,支持简单明了的运算符。
- 功能强大,能进行多种低级操作。
- 目标码质量高,可移植性好。
- 丰富多彩的系统工具、库函数。
- 能培养具有良好素质的程序员。

小知识——C 语言的家族

C 语言的发展历史虽然短，C 语言的家族却成为计算机高级语言中最庞大的。针对不同的任务和对象,现在有数种 C 系统在被广泛地使用。常见的有：

Turbo C 系列(简称 TC):美 国 Borland 公司的产品，它有一个集成环境 IDE; 使用方便，现在几乎所有的大学采用它作为教学工具。但是它编译出 DOS 或 WINDOWS 实模式的.EXE 文件,因此适用于一般教学、科研。

Borland C/C++ 系列(简称 BC):美 国 Borland 公司的产品，它有一个集成环境 IDE, 使用方便，功能比 TC 更强大。2.0 以上的版本支持面向对象的程序设计思想。 4.0 以下版本只能编译出 DOS 或 WINDOWS 实模式的.EXE 文件,4.0 以上版本需要 32 位操作系统的支持。适用于一般教学，科研。

Microsoft C/C++ 系列(简称 MSC):美 国 Microsoft 公司的产品，6.0 以下的版本没有集成环境； 7.0 以上版本有一个集成环境 IDE。MSC 功能强大,尤其是图形处理方面比 BC 和 TC 提供更多的库函数。7.0 以上版本能开发 WINDOWS 程序。但是由于它早期版本没有集成环境，另外由于 MSC 通常用于专业性较强的开发领域，现在国内使用它的人不多。

WATCOM C/C++ 系列(简称 WC) 加拿大 WATCOM 公司的产品, 9.0 以前版本没有集成环境；10.0 有一个 WINDOWS 集成环境 IDE。WC 的功能强大,许多专业开发需要用到它，

如 NOVELL NETWARE 系统开发，AutoCAD-ADS 开发等。它不仅能够编译出 DOS 或 WINDOWS 实模式的 EXE 文件，也能编译出保护模式下的程序。有些专业开发方面，几乎是非它莫属。但是由于它没有 DOS 下集成环境，国内用户极少。

High C/C++ 系列(简称 HC):MetaWare 的产品。基于 intel80386 及 intel80486 处理器以及 MS-DOS Extender 的 C语言系统。HC 能够编写 DOS 实模式及保护模式的程序以及 WINDOWS 程序。HC 到现在为止，没有集成开发环境。也不能单独使用，它需要 Phar Lap 公司的链接器 386LINK 才能生成可执行文件。HC 的功能较强大、但是由于它的使用条件较苛刻，因此国内用户较少。HC 常用于开发一些专业软件如 AutoCAD 的二次开发等。

Visual C/C++系列(简称 VC)：美国 Microsoft 公司的产品。基于 WINDOWS 环境的 C 语言系统。具有集成开发环境 IDE。VC 是新型的 C 语言系统。由于它基于 WINDOWS 环境，因此特别适用于开发 WINDOWS 应用程序。2.0 以下版本可以开发 DOS 及 16 位 WINDOWS 程序。2.0 以上版本需要 32位操作系统的支持，常用于开发 WIN95 及 WIN－NT 应用程序。

1.2C 语言程序的基本结构

在一个源文件里，C 语言程序的结构基本上是有规律的。一般地，我们可以把一个源文件分成三个大的部分：

```
/*程序头部*/
        #include <xyz.h>          /*当.H 文件在系统路径时 */
        #include "xyz.h"          /*当.H 文件在当前目录时*/
        …
        #define A B               /*程序中定义的各种宏*/
        …
        #define A B               /*程序中定义的各种宏   */
                                  /*其他包含的头文件*/
        void MyFunc();            /*自定义函数的原型*/

        int a,b;                  /*全局变量的定义*/


/*main() 函数*/
        void main()
        {;
          int x,y;                /*main 的局部变量定义区 */
```

```
    …
    x=2;y=0;                    /*main 的运算处理区域*/
    y+=x;
    …
    …
    return;                     /*main 的返回值*/
}
```

/*自定义函数区域*/

```
MyFunc1()
  {·}
MyFunc2()
  {·}
  …
```

1.2.1 程序头部

程序头部是由一些预处理包含命令，宏定义组成。

在一个程序里，我们不可避免地要调用一些系统的库函数或者使用一些系统定义的常数、变量等。这些库函数的调用规范、常数、变量的定义都是描述在一些后缀为 H 的文本文件里。这些文件称为“头文件”。

C 语言程序的第一部分要通过包含头文件来表明该程序调用那些库函数，定义了那些常数、变量。其格式为：

```
#include <C 的头文件名>
```

或：

```
#include "C 语言的头文件名"
```

尖括号·<>表示引用的头文件在系统路径里；

双引号""表示引用的头文件在当前目录里。

例如：#include "stdio.h" 表示 stdio.h 这个文件在当前工作目录下，而 stdio 是由 standard input and output 几个单词而来的；因此，上面的包含句子说明程序中要用到标准输入输出的操作。

常见的一些包含命令有：

```
#include <conio.h>      /* 程序里用到控制台 I/O*/
#include <stdlib.h>     /*程序里用到标准库*/
#include <string.h>     /*程序里用到字符串操作函数*/
#include <bios.h>       /*程序里用到 BIOS 硬件操作的函数*/
```

1.2.2 main 函数

所有 C 语言程序都必须有一个 main 函数。例如，下面是一个最简单的 C 语言程序。

```
#include <stdio.h>
void main()
  {
 printf("Hello,World!\n");
 return;
 }
```

C 语言的 main 函数称为主函数，程序运行时总是从 main 函数开始。main 函数基本上是调用其他函数（也可以递归地调用它本身）或者进行一些基本运算。其基本结构如下：

```
void main()              /*主函数名*/
{
XXXXXXXXX                /*定义 main 函数的变量*/
YYYYYYYYY
Function1();             /*调用函数 Function1*/

……
FunctionN();             /*调用函数 FunctionN*/
return;                  /*main 函数结束返回 */
}
```

1.2.3 其他部分

程序的其他部分基本上是由用户定义的各种函数组成。

除 main 函数外，C 语言源文件的其他部分基本上是由程序设计者自己定义的函数或数据组成。一个源程序里，可能会有成千上万个用户自己定义的函数。程序设计者的主要工作就是设计这些函数。因此，一个 C 语言源文件的结构为：

```
#include <XXXXX.H>
                /* 头文件包含区*/
void main       /*      main 函数*/
{
  ……
}
```

```
/*  以下为用户定义的函数*/
    func1;
    func2;
    ……
    funcn;
```

1.2.4 示例

以下是一个完整的 C 语言程序的例子。

```
#include <stdio.h>        /*预处理语句,表示本程序需要用标准 I/O standard input
                            and output 的有关系统标准函数*/
void main()               /* main 主函数,所有 C 语言程序不可缺少的*/
 {   /   *    {   是 C 语言程序里的分隔符，函数的开头和结尾需要
                            它*/
   printf("Hello,world!\n");          /* printf   是 stdio 定义的一个输出函数*/
   printf("I am learning C language\n");
   puts("Welcome to use this C_CAI tool\n");   /*puts 是 stdio 定义的一个输
                                                出字符串的函数*/
}                                     /*  {} 必须配对*/
```

该程序运行的结果是在屏幕上输出以下句了：

```
Hello,world
I am learning C language
Welcome to use this C  CAI tool
```

本节要点

- 程序结构可分为头部区、main 函数区和自定义函数区三部分。
- main 函数内部用 {} 括起来的部分称为函数体,它又可以分为 变量定义区、计算处理区域和 return 返回区域三部分。

1.3 程序设计中的几个名词术语

1.3.1 源程序

源程序就是程序设计者按照一种语言的规范编写的原始程序代码。源程序是程序最基本的数据。所有的源程序都是由 ASCII 码文本，是珍贵的数据资料。源程序又叫

源代码或源文件。通常是我们通过键盘输入的。

1.3.2 编译／解释、编译／解释型语言系统

我们知道，计算机能够执行的是机器语言，机器语言是二进制的。我们编制的源程序是不能直接运行的。计算机运行一个程序，必须首先把它变成二进制代码。目前有两种把源程序变成二进制代码的方法，这就是编译和解释。编译是把整个源程序一次性根据其意义全部变成二进制代码，解释则是根据源程序的每个句子的意义，将一个一个句子地变成相应的二进制代码。通过编译的方法处理源程序的语言系统称为编译型语言系统，通过解释的方法处理源程序的语言系统称为解释型语言系统。

1.3.3 编译程序、目标程序

所有的编译型程序设计系统都提供一个编译程序。编译过程是靠编译程序实现的。当用户把源程序设计好以后，需要利用编译程序对源程序进行编译。在编译的过程中，编译程序对源程序进行扫描，首先检查分析源程序中的错误（语法错误，拼写错误）并把这些错误告诉用户。当用户清除所有这些错误后，编译程序就把源文件编译成一个机器可以理解的二进制程序（称为目标程序或目标文件，后缀为.OBJ）.

源程序、编译程序、编译过程及目标文件的关系可用图 1-1 描述。

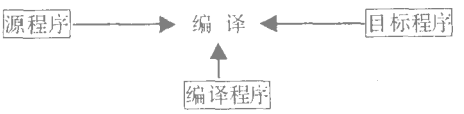

图 1-1 程序编译关系

1.3.4 链接程序与链接

对编译型系统而言，链接是生成可执行文件的最终环节。通俗地讲链接程序将编译生成的目标程序链接成可执行文件。

链接程序、链接过程及可执行文件的关系可由图 1-2 说明。

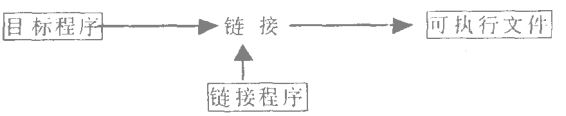

图 1-2 程序链接关系

1.3.5 解释程序与解释

解释程序——解释型程序设计系统提供的一种翻译程序。解释程序把源程序解释成机器能理解的形式。与编译程序不同的是，解释程序并不生成源程序的目标程序而是生成一种另外的可执行的中间代码。因此解释程序每次解释一条指令并立即执行它。

1.3.6 编译型程序系统与解释型程序系统

如果一个程序设计系统提供的编译程序和链接程序，那么这个系统就是编译型程序设计系统，否则就是解释型程序设计系统。

常见的编译型程序设计系统有：PASCAL 系统、C系统、FORTRAN 系统、MASM 宏汇编系统。

常见的解释型程序设计系统有：BASIC 系统、dBASE 系统、FOXBASE 系统、AutolISP 系统。

现在兴起的 FOXPRO 是编译型程序设计系统。

编译型程序设计系统把用户源程序编译成一个可独立运行的.EXE 文件。解释型程序设计系统不能生成能独立运行的可执行文件，它必须要解释程序与源程序一起才能运行。

本节要点

- 源程序是人们编辑输入的程序代码。
- 编译程序、链接程序是编程系统提供的。
- 有两种方法能使我们编制的程序得以运行：编译链接成可执行文件和直接解释它。
- 编译系统效率高。
- 目标程序是源程序被编译的结果,它是机器代码程序(二进制文件)。

2 C 语言的基本数据类型与简单 I/O

数据是程序设计中不可忽视的问题。学习任何一门语言，必须首先了解这种语言系统所支持的数据类型。C 语言的数据可以大致分为基本数据和构造数据。构造数据是按照一定的规则通过基本数据构造出的复杂数据。本章介绍 C 语言的基本数据类型。

2.1 C 语言的基本数据类型

了解一个语言系统的数据类型时，有两个基本点：

(1）系统支持的基本数据类型及其性质；

(2）由基本类型构造复杂类型的原则和方法。

C 语言支持的数据类型可用如下关系如图 2-1 所示。

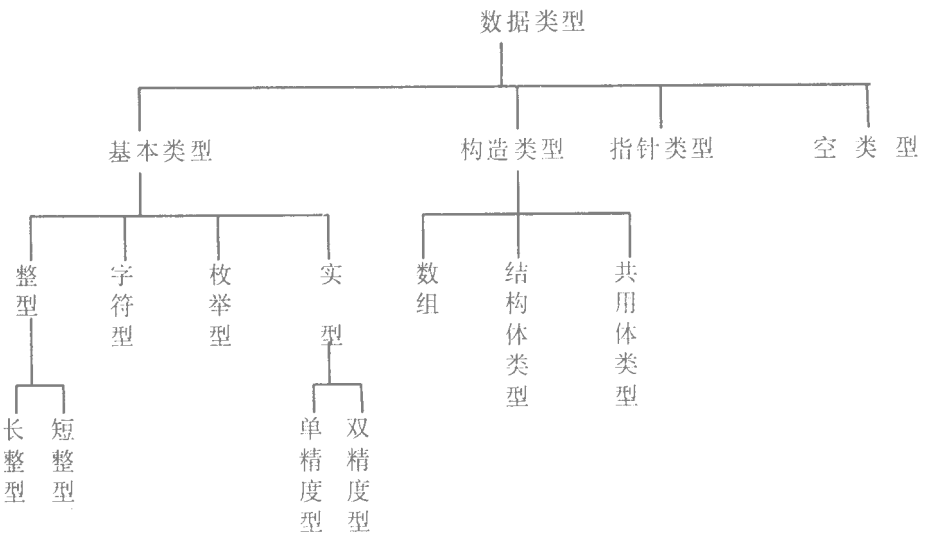

图 2-1 C 语言数据类型关系图

整型、字符类型还可以进一步分为有符号数和无符号(unsigned) 数据 .。例如整型可以有：

```
int 型                    /*一般情况,有符号 */
long int 型               /*有符号长整型 */
short int 型 /*有符号短整型*/
```

unsigned int 型 /*无符号整型*/
unsigned long int 型 /*无符号长整型*/
unsigned short int 型／＊无符号短整型＊／

不同类型数据所适用的范围和在计算机内所占的存储空间也不同。表 2-1 是有关这个方面的综述。

表 2-1 不同类型数据的属性

名 字	属 性
char	范围: -128 ~ 127 存储空间: 1 字节
int	范围: -32768 ~ 32767 存储空间: 2 字节
unsigned int	范围:0 ~ 65535 存储空间: 2 字节
unsigned long	范围:0 ~ 4294967295 存储空间: 4 字节
float	范围:-1.0E38 ~ +1.0E38 存储空间: 4 字节
double	范围:-1.0E308 ~ +1.0E308 存储空间: 8 字节

表中储存空间的字节数又称为数据的长度。它是数据在计算机内存里占据的内存空间的字节数。程序里所有的数据都是要占据内存空间的。

初学者应务必掌握上述每个数据类型的长度范围。在 C 语言里，不同长度的数据类型之间的转换与变换关系是很微妙的。实用程序设计时，数据的长度、范围是永远需要考虑的。

C 语言里允许使用上面的数据类型定义用户数据类型。需要说明的是，这种定义并不是产生一个全新的类型，而是把现有的类型进行综合利用。C 语言里称这样定义出的类型为构造数据类型，这将在后面介绍.

本节要点

- C 语言支持的基本数据类型为：
  整形(int),字符性(char)，浮点型(float)，双精度型(double), void 类型和指针类型。
- int 型和 char 型可以有 unsigned 的变种。
- 不同类型的数据在 RAM 里占据的存储字节数不同，因此表示的数据范围也不同。
- C 语言可以按照一定的规则用基本类型构造其他复杂的数据类型。

2.2C 语言的常量与变量

2.2.1 常量

在程序运行过程中，其值不能改变的量为常量。需要说明的是，计算机语言里常量的概念不同于数学、物理学中的常数。理解计算机语言里的常量，应该从两个方面：①常量是某个程序里的常量，当该程序没有运行时，常量只是一个抽象的概念；只有当程序运行于计算机内存时，常量才有实际意义。②在程序运行期间，常量通常储存在该程序所控制的内存区段的某个固定的内存段，直到程序运行结束。

2.2.1.1 常量的类型

常量也区分为不同的类型，C 语言的常量有 4 种。

(1）整数常量。由一个或多个数字系列组成，整数常量可分为：

十进制整数，由 0～9 的若干个数字组成，如 123、-789 等；

八进制整数，最高位为 0,其余位由 0～7 的若干个数字组成，如 0123、0731 等；

十六进制整数，最高位由 0X 或 0x,其余位由 0～9 及 A～F 的若干个数字组成，如 0x12、0x1b、0x0d 等；

使用 L 或 l 于数字尾部的长整型常量，如 0xa5L、0x77L。

(2）浮点型常量。浮点型常量由整数、小数和小数点组成，如 3.14159、0.01745、1.25E-5 等。

需要说明的是，C 语言支持小数表示和科学计数法表示。如上面的 0.12345 是小数表示，12345E-5 是科学计数法表示。

(3）字符型常量。C 语言的字符型常量由一对单引号和一个 ASCII 字符表示。如'A'、'a'、'B'、'b'等。

值得指出的是，C 语言的大小写字符是不同的，因此'A'与'a'不同。

C 语言里还规定了一些特殊的字符常量，它们列于表 2-2。

表中\叫|转义符，意即它后面的字符有特殊的意义。值得指出的是，有两个经常使用的符号'\n'和|'\r'，许多读者通常混淆了它们的用法。 '\n'是回车换行符，屏幕输出时，如果遇到这个符号，光标将到下一行的第一列；如果遇到'\r'（回车符),则光标到下一行的当前列。

(4)字符串常量。用一对双引号括起来的字符系列,如 "ABC"、"This is a string"、"\\" 等。

表 2-2 C 语言中的特殊字符常量

字符	ASCII 码	含义
\a	0x07	声音报警
\b	0x08	打印退格
\f	0x0c	打印走纸
\n	0x0a	回车换行
\r	0x0d	回车
\t	0x09	水平制表
\v	0x0b	垂直制表
\\	0x5c	反斜线
\'	0x2c	单引号
\"	0x22	双引号
\?	0x3f	问号
\ddd		八进制整数 ddd
\xhhh		十六进制整数 hhh

2.2.1.2 定义常量的方式

在 C 语言里，常量可以通过两个方式定义。

(1) 利用宏定义#define 的方法，如：

```
#define    PI   3.14159
#define    EP   2.178
```

以上定义 PI 为 3.14159，EP 为 2.178.

(2) 利用常量标识符 const 定义，如：

```
const float PI=3.14159;
const float EP=2.178;
const char CH='A';
const char *Str="MESSAGE.ERR";
```

上面两种定义的区别是：用 #define 定义的常量，可以用 #undef 取消，如

```
#undef PI          /*取消 PI 的定义*/
```

而用 const 定义的常量则无法取消，它在整个程序里有效。

2.2.2 变量

在程序运行过程中不断改变值的量，变量也分为不同的类型。

变量是数据传递的载体.程序里所有数据的流动都是以变量为载体的。变量是语言学习的一个重要部分。在 C 语言里规定，所有变量都必须先定义后使用（称为对变量的

引用)。

C 语言里定义变量的方法为：

类型变量名 l(标识符),变量名 2, …,变量名 N;

例如：

```
int x,y;          /*定义两个整型变量 x,y*/
float a,b;        /* 定义两个浮点型变量 a,b*/
double u,v;       /*定义两个双精度变量 u,v*/
char ch1,ch2;／＊定义两个字符型变量 ch1,ch2*/
char *str;        /*定义一个字符型指针变量*/
unsigned z;       /* 定义一个无符号整型变量 */
unsigned long z1;／＊定义一个无符号长整型变量＊／
```

C 语言里可以在定义变量的同时给它赋一个初值(赋初值)。例如：

```
int x=3,y=50;
float a=0.1,b=3.45;
char *str="ccai.hlp";
```

值得说明的是，C 语言的每一个变量都一定的存储空间对应。每定义一个变量时，系统将会按其类型分配相应的存储空间，以存放该变量的值。学习 C 语言的人务必要记住这一点。

不同类型的变量是不能互相传递数据的,除非：

(1)类型间有兼容性；

(2）进行强制类型转换。

C 语言的类型间的兼容性为如下规则：

浮点型 ——兼容→ 整型 ——兼容→ 字符型

即可以把字符型变量的值传递给整型，也可以把整形变量的数据传递给浮点型，但反过来就有问题。

当一个整形变量的值在-128 ~ 127 之间时，从数据范围上讲，整型变量和字符型变量之间是一致的，一般情况下可以把整型变量的数据传递给字符型变量，但由于两者的存储字节数不一样，有时也出错。因此建议大家不要混用。

可以通过强制类型转换的方法进行不同类型变量间数据的传递，转换规则如下：

目的变量=(目的变量的类型)源变量；

例如,如下操作合法：

```
int x;float z;
x=(int)z;         /* 将浮点变量转换成整型*/
```

值得说明的是，强制类型转换通常导致丢失或增加一些字节的数据。它的特点是增加了程序的可读性，如上例中 x=(int)z 的结果是 x 取 z 的一部分字节。

C 语言里还可以通过赋值的方法进行不同类型间数据的转换（这种转换有时也叫隐式转换）。这种转换的特点是不管是什么类型，都要转换成与赋值号(=)左边的类型一致，

如果赋值号右边的操作数具有较高级别的类型，则进行取舍。例如；

```
int a;float b;
a=b=10/3.0;
```

结果是 a 为 3，b 为 3.333333.

2.2.3 标识符

标识符是计算机语言里常用的一个术语。按照通俗的说法，标识符就是名字，标识某个事物的符号。我们在定义变量、常量以及函数时，需要给不同的变量以不同的符号名字以便计算机能够识别他们。例如：语句 int x,y;里有两个标识符 x 和 y 标识两个整形变量。

计算机语言对标识符有一定的规则：①标识符必须以英文字母或下划线开头；②标识符内不得含有运算符；③标识符不能是语言系统的关键字(保留字)。

C 语言的关键字有：

auto	break	case	char	const
continue	default	do	double	else
enum	extern	float	for	goto
if	int	long	register	return
short	signed	sizeof	static	struct
switch	typedef	union	unsigned	void
volatile	while			

本节要点

- 在程序运行过程中，其值不能改变的量为常量。
- C 语言的常量有 4 种：整数，浮点数，字符，字符串。

  定义常量的两种方法：

  用宏定义　　#define M 10

  用 const 定义　　const int 10
- 在程序运行过程中不断改变值的量为变量。

  定义变量的方法：

  类型　　变量名字列表
- 不同类型的变量一般是存在相容性问题。
- 强制转换不同类型变量的方法：

  目标变量=(目标类型)源变量