# Chapter Two
# C Data Types
# 第 2 章　C 数据类型

As in other programming languages, data can be of two types: *constants* and *variables*.

与其他编程语言一样，C语言中的数据分为两种类型：**常量**和**变量**。

## 2.1 Constants（常量）

As the name suggests, a constant does not change its value in a program. Some example of constants are shown in Table 2.1 below.

顾名思义，常量在程序运行过程中其值是保持不变的。

**Table 2.1　some examples of constants**

| Type of constant | Examples | Remarks |
|---|---|---|
| Integer | `100, -3, 0` | Whole numbers that can be positive, negative or zero. |
| Float | `0.34, -12.34, 8.0` | Numbers with decimal parts. |
| Character | `'x', 'X', '*','1'` | Any character enclosed in single quotation marks. |
| String | `"abc", "A100", "1"` | One or more characters enclosed in double quotation marks. |

## 2.2 Variables（变量）

Unlike a constant, a variable can vary its values in a program, and you must also define a variable before you can use it. A variable is defined by giving it a data type and a name.

与常量不同，变量在程序运行过程中其值是可以改变的。在使用变量前，必须先对变量进行定义。所谓定义变量也就是指定变量的类型和名称。

Program Example P2A

```
1  int main()
2  {
3    int v1 ;
4    float v2 ;
5    char v3 ;
6    v1 = 65 ;
7    v2 = -18.23 ;
8    v3 = 'a';
9    return 0 ;
10 }
```

This is an example of a C program.
The line numbers on the left are for reference only and are not part of the program.

这是一个C程序的实例，位于左侧的行号只是为了引用方便，它并不是程序的一部分。

The program starts with the line

```
int main()
```

This line marks the point where a C program starts to execute and must appear once only in a C program.

The program statements are contained within the braces { and } on lines 2 and 10. Each statement ends with a semicolon (;). The spaces before the semicolon and on each side of the equals sign are not essential and are used here only to improve the readability of the program.

Lines 3, 4 and 5 of this program define three variables: `v1`, `v2`, and `v3`. You can give a variable any name you wish, provided you keep within the following four rules:

1. A variable name can only be constructed using letters(a-z, A-Z), numerals (0-9) or underscores(_).
2. A variable name must start with a letter or an underscore.
3. A variable name cannot be a C *keyword*. A keyword is a word that has a special meaning. See appendix A for a list of keywords.
4. A variable name can contain any number of characters, but only the first thirty-one characters are significant to the C compiler.

The valid and invalid variable names are shown in Table 2.2.

**Table 2.2 valid and invalid variable names**

| Name | Remarks |
|---|---|
| `month_1_sales` | This is a valid variable name. |
| `month1sales` | Valid, but not as readable as `month_1_sales`. |
| `1st_month_sales` | Invalid. The name does not start with a letter or an underscore. |
| `month 1 sales%` | Invalid. Spaces and % are not allowed. |
| `Xyz` | Valid, but variable names should be meaningful. |
| `SalesForThisWeek` | Valid and meaningful. |
| `int` | Invalid. This is a keyword. |

Lines 3 to 5 of program P2A define `v1` as an integer variable, `v2` as a floating-point (can hold decimals) variable, and `v3` as a character variable. Note that variable names are case-sensitive, i.e. the variable `V1` is different from the variable `v1`.

Lines 6 to 8 of the program assign values to the variables. The value assigned to each variable is stored in the computer's memory.

The `return` statement on line 9 terminates the execution of the program and returns the integer `0` to the operating system. The `0` value is an indication to the operating system that the program executed successfully with no errors.

C程序是以int main( )行开始的，它标志着程序执行的入口点，在程序中只能出现一次。程序通过花括号{ }将第2~10行之间的语句括起来，每条语句均以分号结束，分号前面和等号两边的空格不是必需的，它们仅仅是为了提高程序的可读性。

程序第3、4、5行语句定义了三个变量v1、v2、v3。在遵守以下规则的前提下，可以根据自己的喜好对其随意命名。

1. 变量名只能由英文字母（a~z，A~Z）、数字（0~9）和下画线（_）构成。
2. 变量名只能以字母或下画线开头。
3. 不能用C语言的关键字对变量命名。**关键字**是系统定义的具有特殊含义的单词，参见附录A中的关键字列表。
4. 变量名可以包含任意数量的字符，但对于C编译器来说，只有前31个字符才是有意义的。

程序P2A的第3~5行定义了整型变量v1、浮点型（可以带小数）变量v2、字符型变量v3。注意，变量名是大小写敏感的，例如变量V1与变量v1是不同的变量。

程序的第6~8行为变量赋值，每个变量的值都存储在内存中。

第9行的return语句用于结束程序的执行，并返回一个整数0给操作系统。返回给操作系统的0值表示程序是正常结束的，没有出现错误。

## 2.3 Simple output to the screen （简单的屏幕输出）

Now that you have assigned values to the variables, how do you display their values on the screen? You can do this with **printf** (pronounced print-f) as shown in the next program.

既然已经对变量进行了赋值操作，那么如何将其显示到屏幕上呢？答案是使用printf。

Program Example P2B

```
1  #include <stdio.h>
2  int main()
3  {
4    int v1 ;
5    float v2 ;
6    char v3 ;
7    v1 = 65 ;
8    v2 = -18.23 ;
9    v3 = 'a' ;
10   printf( "v1 has the value %d\n", v1 ) ;
11   printf( "v2 has the value %f\n", v2 ) ;
12   printf( "v3 has the value %c\n", v3 ) ;
13   printf( "End of program\n" ) ;
14   return 0 ;
15 }
```

When you compile and run this program you will get the following on your screen:

```
v1 has the value 65
v2 has the value -18.230000
v3 has the value a
End of program
```

Line 1 is an example of a *preprocessor directive*. This line will be in nearly every program that you write. (Preprocessor directives are covered in Chapter Fourteen.)

程序第1行是一条**预处理指令**，这一行几乎出现在每个程序中（预处理指令将在第14章介绍）。

**printf()** is a standard library function for displaying data on the screen. The `printf()` functions in lines 10 to 12 have a string of characters enclosed in double quotation marks (called the *format string*) followed by a comma and a variable name. The variable name is not in quotation marks. Let's look at the first `printf()` in line 10 and see how it works. All the characters between the double quotation marks up to the `%` symbol are displayed. The screen will then show:

函数printf( )是一个用于在屏幕上显示数据的标准库函数，程序第10~12行的printf( )的括号中都有一个用双引号括起来的字符串，称为**格式字符串**，双引号后面是一个逗号，接着是变量名，变量名不能放到格式字符串所在的双引号中。

双引号中位于%之前的字符都被显示到屏幕上。

```
v1 has the value
```

The letter `d` is called a conversion character and is preceded by a `%` sign. The `%d` (`%i` can also be used) displays the

字符d称为转换字符，以其前面的%作为其标识，%d（或者用%i）用于以十进制整型

variable v1 as a decimal integer on the screen. The variable v1 has the value 65, so the entire printf() on line 10 displays:

格式将v1显示到屏幕上。

```
v1 has the value 65
```

The newline character (\n) at the end of the format string is an instruction to skip to the next line of the screen.

In the printf() functions on lines 11 and 12, the %f displays v2 as a floating-point number and the %c displays v3 as a character.

It is not always necessary to have a variable in a printf(). The printf() on line 13 has no variable and simply displays the line:

格式化字符串最后的\n为换行符，其作用是将光标跳转到屏幕下一行的起始位置。在第11行和第12行的printf( )中，使用%f以浮点型格式输出v2的值，使用%c以字符型格式输出v3 的值。

printf( )中的变量不是必须有的，如第13行的printf( )中就没有变量，只是简单地输出字符串。

```
End of program
```

## 2.4 Comments（注释）

Comments are added to a C program to make it more readable for the programmer, but they are completely ignored by the computer. Comments start with the characters /* and end with the characters */. We can add some comments to the last program.

在C程序中添加注释可以增加程序的可读性。但是在程序执行的过程中，计算机是完全忽略它们的。C语言中的注释以/*开始、以*/结束。

Program Example P2C

```
1  /* Program Example P2C
2     Introduction to variables in C. */
3  #include <stdio.h>
4  int main()
5  {
6    int v1 ;   /*  v1 is an integer variable.       */
7    float v2 ; /*  v2 is a floating-point variable. */
8    char v3 ;  /*  v3 is a character variable.      */
9    /* Now assign some values to the variables. */
10   v1 = 65 ;
11   v2 = -18.23 ;
12   v3 = 'a' ;
13   /* Finally display the variable values on the screen. */
14   printf( "v1 has the value %d\n", v1 ) ;
15   printf( "v2 has the value %f\n", v2 ) ;
16   printf( "v3 has the value %c\n", v3 ) ;
17   printf( "End of program\n" ) ;
18   return 0 ;
19 }
```

Comments can be placed anywhere in a C program and can span more than one line. Just make sure a comment starts with /* and ends with */.

注释可以放置在程序中的任何位置，并且可以跨越多行，只要确保注释的内容是以/*开始、以*/结束即可。

Comments cannot be *nested*, i.e. you cannot have a comment within another comment.

注释不可以嵌套，即不能在一个注释中添加另一个注释。

Typically, comments are placed at the start of the program to describe the purpose of the program, the author, date written and any other relevant information, such as the version number. For example:

通常在程序的开始位置添加注释，用于说明程序的用途、作者、编写的日期及其他一些相关的信息，如版本号等。

```
/*  Program name  : P2A.
    Introduction to variables in C.
    Written by    : Paul Kelly and Su Xiaohong.
    Date          : June 29, 2012.
    Version number: 1.0                    */
```

Comments are also used to describe in plain language the function of a particular section of a program. Get into the habit of using good explanatory comments; the more complicated the program becomes, the more valuable they will be to you and any other programmer reading your program.

注释还可用于通过简洁的文字来说明某段程序要实现的功能。要养成在程序中添加良好注释的习惯，程序越复杂，注释对阅读程序的程序员而言价值就越大。

## 2.5 Data types（数据类型）

In previous programs, it was shown how to declare a variable and associate it with a particular data type (`char`, `int` or `float`). The C language has a variety of other data types besides the three basic types of `char`, `int` and `float`. Different data types require different amounts of memory and therefore vary in the range of values they can store.

在前面的程序中，我们已经看到如何声明变量，如何将变量与一种特定的数据类型（如char、int、float）关联起来。除了char、int、float这三种基本的数据类型之外，C语言还提供了许多其他的数据类型。不同的数据类型占据不同大小的内存空间，因此也拥有不同的取值范围。

Details of the various data types in the C language are given in appendix D.

### 2.5.1 Short integer data types

A short integer has a smaller range of values than an integer and consequently uses less memory.

短整型能表示的数值范围小于基本整型，相应地短整型占用的内存也更少。

The following defines a variable `v1` as a `short integer`:

```
short int v1 ;
```

The keyword `int` is optional, so `v1` can also be defined as:

在这里，关键字int是可选的，所以v1也可以定义为如下形式。

```
short v1 ;
```

A `short int` variable, like an `int` variable, is displayed using `%d` as the format specifier in `printf()`. For example:

和基本整型变量一样，使用printf( )函数输出短整型变量的值，同样也使用%d格式转换符。

```
printf( "%d", v1 ) ;
```

### 2.5.2 Long integer data types

A long integer has a larger range of values than an integer. The following defines a variable `v2` as a `long` integer:

长整型比基本整型的取值范围大。

```
long int v2 ;
```

The keyword `int` is optional here, so `v2` can also be defined as:

```
long v2 ;
```

To display a `long int` variable, the format specifier `%ld` is used in `printf()`. For example:

使用printf( )函数输出长整型变量的值，应使用%ld格式转换符。

```
printf( "%ld", v2 ) ;
```

### 2.5.3 Unsigned integer data types

The keyword `unsigned` extends the positive range of an integer variable but does not allow negative values to be stored.
The following defines unsigned integer variables `v3` and `v4`:

关键字unsigned扩展了整型变量可以表达的正整数的范围，但同时也使其不能再存储负数。

```
unsigned int v3 ;
unsigned long int v4 ;
```

To display these variables, `%u` and `%lu` are used as the format specifiers in `printf()`. For example:

为了利用printf( )函数输出无符号整型变量的值，应使用%u和%lu格式转换符。

```
printf( "%u %lu", v3, v4 ) ;
```

### 2.5.4 Double floating-point data type

The `double` data type allows you to increase the range and precision (or accuracy) of a floating-point number.
The following defines a variable `v5` as a `double` data type:

double类型提高了浮点型数据的取值范围和精度。

```
double v5 ;
```

To display the value in a `double` variable, either `%lf` or `%f` can be used as the format specifier in `printf()`. For example:

用printf( )函数输出double型变量的值时，既可以使用%lf格式转换符，也可以使用%f格式转换符。

```
printf( "%lf", v5 ) ;
```

## 2.6 Data type sizes（数据类型的大小）

The next program uses the **sizeof()** operator to display the number of bytes of memory required by some of the common data types in C.

下面的程序使用sizeof( )运算符来显示C语言中一些常用数据类型所占内存空间的字节数。

Program Example P2D

```
1  /* Program Example P2D
2     Program to display the amount of memory required by
3     some of the common data types in C. */
4  #include <stdio.h>
5  int main()
6  {
7   int char_size, int_size, short_size, long_size,
8       float_size, double_size ;
9
10  char_size = sizeof( char ) ;
11  int_size = sizeof( int ) ;
12  short_size = sizeof( short int ) ;
13  long_size = sizeof ( long ) ;
14  float_size = sizeof( float ) ;
15  double_size = sizeof( double ) ;
16
17  printf( " Data type       Number of bytes\n" ) ;
18  printf( " ---------       ---------------\n" ) ;
19  printf( "  char               %d\n", char_size) ;
20  printf( "  int                %d\n", int_size ) ;
21  printf( "  short int          %d\n", short_size ) ;
22  printf( "  long int           %d\n", long_size ) ;
23  printf( "  float              %d\n", float_size) ;
24  printf( "  double             %d\n", double_size ) ;
25  return 0 ;
26 }
```

Lines 7 and 8 of this program define six integer variables.

Lines 10 to 15 use the `sizeof()` operator to assign the size in bytes of each data type to one of the six variables.

Lines 17 to 24 display the value in each of the six variables.

The output from this program is:

```
Data type       Number of bytes
---------       ---------------
  char               1
  int                4
  short int          2
  long int           4
  float              4
  double             8
```

程序的第7行和第8行定义了6个整型变量，第10~15行使用sizeof( )运算符来计算每种数据类型所占内存空间的字节数，然后赋值给相应的6个变量。第17~24行显示这6个变量的值。

## Programming pitfalls

1. Do not type a semicolon after either

```
#include <stdio.h>
```

or

```
int main()
```

1. 不要在#include <stdio.h>和int main( )后面添加分号。

2. Comments start with `/*` and end with `*/`.
Forgetting to end a comment with `*/` may cause part of your program to be ignored by the compiler. For example:

```
printf( "Line1\n" ) ;
/* This comment does not end properly
printf( "Line2\n" ) ;
printf( "Line3\n" ) ;
/* The compiler thinks the comment ends here -> */
printf( "Line4\n" ) ;
```

The statements above will only display `Line1` and `Line4`. All the statements between `/*` at the start of the second line and `*/` at the end of the fifth line are regarded as a comment.

2. 注释以/*开始，以*/结束。忘记以*/ 结束将导致部分程序被编译器所忽略。

上面的语句只是显示Line1和Line4。位于第2行开头的/*和第5行行末的*/之间的语句都作为注释来处理。

3. Comments cannot be nested, i.e. you cannot have a comment within another comment.

```
/* This is an error because you cannot /*nest*/ comments */
```

3. 注释不能嵌套，即不能在一个注释中包含另一个注释。

4. Use the correct format specifier when displaying a variable using `printf()`. For example,

```
float var ;
var = 123.56 ;
printf( "%d", var ) ;
```

will incorrectly display the value in the variable `var`, because `var` is a `float` and `%d` is used for displaying an integer.

4. 使用printf( )函数显示变量值时要使用与变量类型相对应的、正确的格式转换符。下面语句就不能正确显示变量var的值，因为var是float类型，而%d是显示整型数据的格式转换符。

5. The second character in the `printf()` format specifiers `%ld` and `%lf` is the letter *ell*, not the number one.

5. printf( )函数的格式转换符%ld和%lf中的第2个字符是字母l，不是数字1。

## Quick syntax reference

At the end of each chapter the most important features of the C syntax covered in the text are briefly summarised. While not covering the strict definition of the syntax, which can be complex for a beginner, it should prove to be a useful “memory jog” while writing programs.

在每章的最后，将本章中出现的最重要的C语言语法做一个简要的总结，没包含令初学者头疼的语法的严格定义。在读者编写程序时，这对于唤醒你的记忆是非常有帮助的。

| | Syntax | Examples |
|---|---|---|
| **Start of program** | `#include <stdio.h>`<br>`int main()`<br>`{` | |
| **Defining variables** | `char variable(s) ;`<br>`int variable(s) ;`<br>`float variable(s) ;`<br>`short int variable(s) ;`<br>`double variable(s) ;`<br>`unsigned variable(s) ;` | `char any_letter, y_or_n ;`<br>`int distance ;`<br>`float average, pay, tax ;`<br>`short int temperature ;`<br>`double total, number ;`<br>`unsigned int employee_num ;` |
| **Assignment** | `=` | `tax = 59.75 ;` |
| **Comments** | `/*  */` | `/* Explanatory text. */` |
| **Display on the screen** | `printf(text) ;`<br>`printf("Tax Program\n") ;` | `printf(format,variables) ;`<br>`printf("Tax is %f",tax) ;` |
| **End of program** | `return 0 ;`<br>`}` | |

## Exercises

1. Which of the following are valid variable names? If valid, do you think the name is a good mnemonic (i.e. reminds you of its purpose)?
   (a) `stock_code`
   (b) `money$`
   (c) `Jan_Sales`
   (d) `X-RAY`
   (e) `int`
   (f) `xyz`
   (g) `1a`
   (h) `invoice_total`
   (i) `john's_exam_mark`
   (j) `default`
2. Identify the data type of each of the following constants:
   (a) `'x'`
   (b) `-39`
   (c) `39.99`
   (d) `-39.0`
3. Which of the following are valid variable definitions?
   (a) `integer account_code ;`
   (b) `float balance ;`
   (c) `decimal total ;`
   (d) `int age ;`
   (e) `double int ;`
   (f) `char c ;`

4. Write a variable definition for each of the following:
   (a) integer variables `number_of_transactions` and `age_in_years`
   (b) floating-point variables `total_pay`, `tax_payment`, `distance` and `average`
   (c) a character variable `account_type`
   (d) a double variable `gross_pay`
5. Write the most appropriate variable definition for each of the following:
   (a) the number of students in a class
   (b) an average price
   (c) the number of days since the 1st of January 1900
   (d) a percentage interest rate
   (e) the most common letter on this page
   (f) the population of China (estimated to be 1,339,724,852 in November 2010).
6. Assuming the following:

```
int i ;
char c ;
```

   which of the following are valid C statements?

```
c = 'A' ;
i = "1" ;
i = 1 ;
c = "A" ;
c = '1';
```

7. Write a C program to assign values to the variables in question 4 and display the value of each variable on a separate line.
8. Write a C program that displays the following:

```
***************
* Hello World *
***************
```

9. Write a C program to display your name and address on separate lines.
10. ASCII codes are used to represent letters, digits and other symbols inside the computer's memory. Using the ASCII table in appendix C, look up the ASCII code for each of the following characters:

    `'A' 'B' 'Y' 'Z' 'a' 'b' 'y' 'z' '0' '1' ',' ' '` (a space)

11. In program P2C, change the `%d` in line 14 to `%c` and the `%c` in line 16 to `%d`.
    Compile and run the modified program. Can you explain the output?
    (Hint: See ASCII table in appendix C.)