Lecture 3.1

Topics

1. Constants in C
2. How-To Code Constants in a C Program

---

1. Constants in C – Revisited

Constants are values that cannot be changed during the execution of the program.

Constants may have types (as being implied) when these values were coded into the program. There are three types of constants of

- **Integer,**
- **Floating-point**, and
- **String**.

Let's consider them next.

1.1 Integer Constants

An integer constant, as coded with a series of digits, has

- Signed integer (`int`) type, or
- Signed long integer (`long`) type if the number is large, or
- Default type to be `int`.

One can override the defaults by using the suffixes `u` (or `U`) and `l` (or `L`). `U` and `L` are generally preferred to avoid any misunderstanding or confusion.

The following examples show the types of the constants.

Literal	Value	Type
123	123	int
-123	-123	int
-123L	-123	long int
123LU	123	unsigned long int

1.2 Floating-Point Constants

The default type for any floating-point constant is `double`. One can use the format specifiers `l` (`L`) and `f` (`F`) to indicate whether it is a `long double` or a `float`.

Some examples are given below.

Literal	Value	Type
123.	123.0	double
-0.123	-0.123	double
123F	-123.0	float
123.0L	123	long double

In addition to these numeric constants, C has two other constants:

- **Character,** and
- **String**.

1.3 Character Constants

Character constants are enclosed between two single quotes.

In C, there are two groups of characters:

(1) Printable characters, and

(2) Non-printable characters.

Printable Characters

Normally, character constants would have shown with only one character quoted. These characters are mostly called **printable characters** that means one can print these characters on paper or see them displayed on screen.

For examples,

```
'a' 'A' 'o' 'O' 'z' 'Z' '1' '9' '0'
```

Note that the digits enclosed in the single quotes should be understood as characters and not numerical values.

Non-Printable Characters

There are other characters that have a backslash (\) added in front of characters (only one character after the backslash). These are called **nonprintable characters**; their values or effects would be either seen or heard when the system tries to display them.

For examples,

'\0'	Null character	'\a'	Bell
'\b'	Backspace	'\t'	Horizontal tab
'\n'	Newline	'\''	Single quote
'\"'	Double quote	'\\'	Backslash

1.4 String Constants

A string constant is a sequence of zero or more characters enclosed in a pair of double quotes.

For examples,

""	Null string
"CIS26"	A five character string
"CIS26\n"	A six character string

2. How-To Code Constants in a C Program

Recall that constants in C can be integer, floating-point, character, or (character) string.

These constants may need to be coded in some computer program.

Let's look at the process or steps to bring these constants into a C program.

There are three different ways to code constants:

- Literal constants,
- Defined constant

- Memory constants

2.1 Literal Constants

A literal constant (or just literal) is an **unnamed constant** used to specify a data point or value.

For examples,

```
'C'       /*A character literal*/
26        /*numeric literal*/
3.14159   /*A floating-point literal*/
"CIS26"   /*A string literal*/
```

2.2 Defined Constants

A preprocessor command (or statement) **`define`** is used to define a constant.

For examples,

```
#define PI 3.14159263536
#define SALE_TAX_RATE .0850
```

These defined constants are represented through predefined names (chosen by the programmer). These **`define`** preprocessor commands are placed at the beginning of the program.

During the compilation, all of these names, if found, are replaced by the defined values.

- These names may be used throughout the program by the programmer, but only their values are what actually being used by the compiler.
- That means these names do NOT exist!

Note that the **`#include`** and **`#define`** are called the **preprocessor directives**. These are the commands that direct the compiler to perform some specific tasks.

2.3 Memory Constants

Memory constants are constants defined through variables that qualified with a constant restriction.

In C, this is done with the **`const`** qualifier (type qualifier).

For examples,

```
const double pi = 3.14159263536;
const int numMonth = 12;
```

Note that the above are statements (with semicolons terminated). The names are variables and they must be initialized.

Each of these variables is initialized with one and only one value; this value must be used as is and cannot be changed during the execution of the program.

Example 1

```
/**
 *Program Name:  cis26L0311.c
 *Written By:    T. Nguyen
 *Discussion:    Constants
 */
#include <stdio.h>
```

```c
#define PI 3.14

const int iMyConstant = 100;

int main() {
  printf("Value of a literal constant : %d\n", 100);
  printf("Value of a defined constant : %f\n", PI);
  printf("Value of a memory constant : %d\n", iMyConstant);

  return 0;
}
```

OUTPUT

```
Value of a literal constant : 100
Value of a defined constant : 3.140000
Value of a memory constant : 100
```