Variables in C language

A variable is a symbolic name for a memory location in which data can be stored and subsequently recalled. Variables are used for holding data values so that they can be utilized in various computations in a program. All variables have two important attributes:

- A **type** which is established when the variable is defined (e.g., integer, real, character). Once defined, the type of a C/C++ variable cannot be changed.
- A **value** which can be changed by assigning a new value to the variable. The kind of values a variable can assume depends on its type. For example, an integer variable can only take integer values (e.g., 2, 100, -12).

Listing 1 illustrates the uses of some simple variable.

Listing 1

```
1  #include <stdio.h>

2  int main ( )
3  {
4      int     workDays;
5      float   workHours, payRate, weeklyPay;

6      workDays = 5;
7      workHours = 7.5;
8      payRate = 38.55;
9      weeklyPay = workDays  * workHours * payRate;
10     printf("weeklyPay = %.2f\n",weeklyPay);
11     return 0;
12 }
13
```

Annotation

4 This line defines an `int` (integer) variable called `workDays`, which will represent the number of working days in a week. As a general rule, a variable is defined by specifying its type first, followed by the variable name, followed by a semicolon.

5 This line defines three `float` (real) variables which, respectively, represent the work hours per day, the hourly pay rate, and the weekly pay. As illustrated by this line, multiple variables of the same type can be defined at once by separating them with commas.

6 This line is an **assignment** statement. It assigns the value `5` to the variable `workDays`. Therefore, after this statement is executed, `workDays` denotes the value 5.

7 This line assigns the value `7.5` to the variable `workHours`.

8 This line assigns the value `38.55` to the variable `payRate`.

9 This line calculates the weekly pay as the product of `workDays`, `workHours`, and `payRate` (`*` is the multiplication operator). The resulting value is stored in `weeklyPay`.

10 This line output three items in sequence: the string `weeklyPay = `, the value of the variable `weeklyPay`, and a newline character.

When run, the program will produce the following output:

```
weeklyPay = 1445.625
```

When a variable is defined, its value is **undefined** until it is actually assigned one. For example, `weeklyPay` has an undefined value (i.e., whatever happens to be in the memory location which the variable denotes at the time) until line 9 is executed. The assigning of a value to a variable for the first time is called **initialization**. It is important to ensure that a variable is initialized before it is used in any computation.

It is possible to define a variable and initialize it at the same time. This is considered a good programming practice, because it pre-empts the possibility of using the variable prior to it being initialized. Listing 2 is a revised version of Listing 1 which uses this technique. For all intents and purposes, the two programs are equivalent.

Listing 2

```
 1  #include <stdio.h>

 2  int main (void)
 3  {
 4      int     workDays = 5;
 5      float   workHours = 7.5;
 6      float   payRate = 38.55;
 7      float   weeklyPay = workDays * workHours * payRate;

 8      printf("weeklyPay = %.2f\n",weeklyPay);
 9      return 0;
10  }
11
```