Note 1:
Tokens, Identifiers, Data Types and Variables

1 C Tokens

Tokens are the basic building blocks in C language. A token may be assumed as the smallest individual unit in a C program. The program is constructed using a combination of these tokens.

1. Keywords
2. Variables
3. Constants
4. Strings
5. Special Characters
6. Operators

Some of these might not be included in this note, but will be covered as we progress forward.

2 Keywords

Like every computer language, C has a set of reserved words often known as keywords that cannot be used as an identifier. All keywords are basically a sequence of characters that have a fixed meaning.

The list of keywords is as follows (Keep in mind, learning and remembering these is not important, you will learn what these do and how they work and remember them automatically throughout this course.):

```
auto
break
case
char
const
continue
default
```

```
double
else
enum
extern
float
for
goto
int
long
register
return
short
signed
sizeof
struct
switch
typedef
union
unsigned
void
volatile
do
if
static
while
```

This list is only provided as a reference.

3 Identifiers

Identifiers are names given to program elements such as variables, arrays and functions. Identifiers may consist of sequence of letters, numerals, or underscores.

3.1 Rules for forming Identifier Names

1. Identifiers cannot contain any special characters except the underscore.
2. There cannot be two successive underscores.
3. Keywords cannot be used as identifiers
4. Casing of the letters matter. For eg: first and First are different.
5. Identifiers must begin with an alphabet or an underscore. But use of underscore as the first character is generally avoided.
6. Identifiers must not contain more than 31 characters.

Examples of valid identifiers:

```
roll_number, marks, name, emp_number, basic_pay, HRA, DA, dept_code,
DeptCode, RollNo, EMP_NO.
```

Examples of invalid identifiers:

```
23_student, %marks, @name, #emp_number, basic.pay, -HRA, (DA),
&dept_code, auto.
```

4 Data Types in C.

1. **Char** - 1 byte.
   Used to store characters
   Keyword used - char

2. **Integer** - 2 bytes.
   Used to store integer numbers.
   Keyword used - int

3. **Floating point** - 4 bytes.
   Used to store floating point numbers.
   Keyword used - float

4. **Double** - 8 bytes.
   Used to store big floating point numbers.
   Keyword used - double

5. **Void** - 0 bytes.
   Valueless
   Keyword used - void.
   Use cases:

   (a) To specify a return type of a function (When a function returns no value).
   (b) To specify the parameters of the function. (When the function accepts no arguments).
   (c) To create generic pointers.

5 Variable

A variable is defined as a meaningful name given to a data storage location in computer memory. When using a variable, we actually refer to address of the memory where the data is stored.

To declare a variable, specify the data type of the variable followed by its name. The data type indicates the kind of values that the variable will store. In C, variable declaration always ends with a semicolon.

```
int emp_num;
float salary;
char grade;
double balance_amount;
unsigned short int acc_no;
```

Variables can be declared anywhere in the program but:

1. Variables should be declared before using them.

2. Variables should be declared closest to their first point of use to make the source code easier to maintain.

C allows multiple variables of the same type to be declared in one statement.

```
float temp_in_celcius, temp_in_farenheit;
```

Here both the variables are now declared as floating point values.
In C variables are declared at three basic places as follows:

1. When a variable is declared inside a function it is known as a local variable.

2. When a variable is declared in the definition of function parameters it is known as a formal parameter.

3. When the variable is declared outside all functions, it is known as a global variable.

While declaring variables, we can also initialize them with some value. For example:

```
int emp_num = 7;
float salary = 2156.35;
char grade = 'A';
double balance_amount = 100000000;
```

The initializer applies only to the variable defined immediately before it:

```
int count, flag = 1;
```

In this statement, only the variable flag is initialized to 1 and not the count variable.
To initialize both of them in the same line,

```
int count = 0, flag = 1;
```

When variables are declared but not initialized, they usually contain garbage values.

6 Constants

Constants are identifiers whose values do not change. While values of variables can be changed at any time, values of constants can never be changed. Constants are used to define fixed values like pi so that their values aren't changed in the program even by mistake.
C allows the programmer to specify constants of integer type, floating point type, character type and string type.
Declaring a constant:

```
const float pi = 3.14;
```

The const keyword specifies that the value of pi cannot change.
Another way to designate a constant is to use the pre-processor command define.
define is preceded with a # symbol like other pre processor directives.
Although they can be placed anywhere in the program, they are generally placed at top so they can be modified at a later stage.

```
#define pi 3.14159
```

6.1 Some Rules

1. Constant names are usually written in capital letters to visually distinguish them from other variable names. (Just a convention, not a rule).

2. No black spaces are permitted between the # and the define keyword.

3. Blank space must be used between #define and constant name and between constant name and constant value.

4. #define is a pre-processor compiler directive and not a statement. Therefore, it does not end with a semi-colon.