**Introduction to C**

Table of Contents

[Tokens 2](#_Toc65497738)

[Data Types 3](#_Toc65497739)

[Rules of Identifiers 4](#_Toc65497740)

[Arrays 5](#_Toc65497741)

[Scope of Variables 6](#_Toc65497742)

[Files 7](#_Toc65497743)

[Format Specifiers 8](#_Toc65497744)

[Comments 11](#_Toc65497745)

## Tokens

A token is the smallest individual unit of coding. In the case of the C language, these are called C tokens. (Ref: Page 25 – ANSCII Book)

C Tokens:

1. Keywords – words that are fixed for C and cannot be used by the user (e.g. float, int, while etc.)
2. Identifiers – any words that are used in the program (except keywords)
3. Constants – numbers used in the program (can be decimals)
4. Strings – characters used in the statement (e.g. “cat”, “dog”)
5. Special Symbols – symbols used for specific purposes in coding (e.g. (), {}, ; etc.)
6. Operators – symbols of addition, subtraction etc.

## Data Types

int – integers – 16 bit/32 bit size

char – characters – 8 bit

long int – long integers – 32 bit size always

long long int – 64 bit size always

float – decimal numbers – 32 bit size

double – 64 bit

long double – 80 bit

Computers store the values in binary 0s and 1s. For 9 bits, the first bit is used for the sign and the other 7 bits are used to represent binary numbers from 0 to 27 – 1.

The character “A”, for example, could be stored in a char variable. In actual memory, it would be stored using something called ASCII code. The character “A” has a value of 65 in this code, “B” has a value of 66 and so on. These values are used to later identify the characters. There is a total of 256 characters in ASCII code, and only English characters can be stored.

## Rules of Identifiers

1. It can only contain alphabets, digits and underscores
2. It cannot start with a digit
3. It cannot have spaces
4. Cannot use keywords as identifiers

e.g. int a OK int \_AB OK

int x7 OK int 3ab NO

int \_43 OK int a b NO

int a = 3.5; (constant)

"abcdef" - string (anything in double quotations)

int a = 3 + 4; - declaration

- operator (so in the program, a = 7)

## Arrays

Arrays: used to add more than one element of a data type

char a[10] = "ABC..."

C

If more than 10 characters are given, the characters after the tenth will be lost (not stored).

a[0] = 'A'  
a[1] = 'B'

C

….and so on.

## Scope of Variables

int main () *// main function*{  
 *// (inside second brackets)* int a; *// function: block of words  
 // scope: each scope has its own identifier* a = 3;  
}  
a = 3; *// not accepted since it is outside the scope*

C

If a variable is declared outside a scope, that can be declared in any scope.

int a *// global variable*int main ()  
{  
 int b *// local variable (can only be declared in this scope)*}

C

The same variable cannot be declared in different functions, as doing so would confuse the computer as to which value to accept.

E.g.

int a = 5 // a = 5

int b = 5 // b = 3

int c = a-b // c = 2

double d = 5.7 // d = 5.7

int e = 3.2 // e = 3

double f = d-e // f = 2.7

double g = d – (double)e // g = 2.5

C

In some code, if the main and local variable is the same and we change the declaration of the variable, the computer will prioritize the local variable.

## Files

Files - C Files (file.c)

- H Files (file.h) - included using #include e.g. #include <stdio.h>

* ‘stdio’ means standard input and output. By including this, we can use basic functions like printf, scanf etc. in the program.
* Writing the line #include <math.h> will allow the use of sin, cos, asin, acos etc.
* While making projects, we can create our own header files by putting the required code in a specified folder.

## Format Specifiers

[Refer to Ch 4 Table 4.3]

Understand format specifier from the book. It is used in the code to help the computer understand which type of variable is being used.

Format Specifiers:

int - %d

long - %ld

long long - %lld

float - %f

char - %c

string - %s

If we want to include a variable with a string,

int a = 5;  
printf ("Hello World %d", a);

C

Output: Hello World 5

The function scanf is the opposite of printf. It is used to gather input from the user.

e.g.

char ch [100];  
printf ("Enter a sentence: ");  
scanf ("%s", &ch);  
printf ("%s", ch);

C

Output: Enter a sentence: (Say the sentence “I am CSE” is entered)

Prints: I

This occurs since %s works in such a way, that it will not accept any characters after a space. To fix this, gets (ch) can be used instead of scanf ("%s", &ch).

[https://www.cppreference.com](https://www.cppreference.com/) - to see all header files in C

These functions will help us understand what is included in the code:

isnum() - numbers

isalpha() - alphabets

isdigit() - digits

islower() - lower case

ispunct() - punctuation

isspace() - space

isupper() - upper case

Note: In ANSCII, A = 65 and a = 97

E.g.

For 4.8, isnum() and isdigit() will show 1 but other functions will show 0. For ".", only ispunct() will show 1.

char ch[100], int a;  
printf ("Enter a sentence \n");  
scanf ("%d", &a); *// insert getchar () here*gets (ch);  
puts (ch); *// another way to write printf*

C

In this case, the user will be asked to write an integer. Say 3 is written. 3 will then be printed. A character wasn’t even taken, even though the program was asked to print the characters. To solve this, getchar() is used, which helps us take the characters in the code.

int a = 1234;  
printf ("%2d", a);

C

Output: 12

int a = 1234;  
printf ("%6d", a);

C

Output: 001234

## Comments

*/\* multi-line  
 comment \*/  
// single line comment*

C