**Objective C**

**Introduction:**

Objective C is a superset of C Language. It is the combination of C and the OOPS features. The C and Objective C are having same basic syntax of declaring and initializing the variables and the control statements etc. The simple Objective C program is as shown below:

#import<Foundation/Foundation.h>

int main()

{

//simple objective C program

NSLog(@"Hello World");

return 0;

}

Here the method NSLog(@" ") present in the library Foundation, hence we import it.

**Comments:**

Comments are used to give extra information about the block of code and are ignored by the compiler.

We have two ways to comment the text:

1) Inline Comment: A single line can commented.

//This is the inline comment

2) Block comment: More than one lines can be commented.

/\*This is Block comment.

We can write comment to more than one line\*/

**Constant :**

The keyword const is used to represent the constant, it tells the compiler that the variable is never allowed to change.

For eg: double const pi = 3.14159;

pi = 4.200000; //compile error

**Objective C Variables:**

* Variables are the container which holds the value and the variables gets the values statically and the programmer shoul mention the type of the value it should hold.
* The syntax for variable Declarartion : "<variable type><variable name>;".eg.: int aInt;
* We have different types of variables like integer type variables, floating type variables etc.
* The variables can be intialized with a value which is compatible.eg.: int bInt=3;
* The variable of one type can be casted to the other type, syntax is <target type><value>.
* The value of the variable is printed using the statement NSLog(@"string format");.eg.:to print the integer value we use: NSLog(@"%d");

**Objective C Primitive Datatypes:**

**1) BOOL:** It is primitive data type which checks whether true or false. It takes the values "YES" or "NO". If we use the string format "%d" it prints '0' for "NO" and '1' for "YES".

Eg: BOOL isbool = YES;

NSLog(@"%d", isbool);

NSLog(@"%@", isbool ? @"YES" : @ "NO");

**2) char:** It is primitive data type which holds both signed and unsigned the character value. The string format "%c" is used to print the single character value given to the variable, "%hhd" to print its ASCII value or "%hhu" to print the unsigned value given to the variable.

Eg: char aChar = 'a';

unsigned char uChar = 256;

NSLog(@"%c ASCII is @%hhd", aChar, aChar);

NSLog(@"%hhu", uChar);

**3) short:** It is primitive data type which holds both signed and unsigned the short value. The string format "%hd" is used to print the integer value given to the variable, "%hu" to print unsigned value given to the variable of type short.

Eg: short aShort = -56789;

unsigned = 256;

NSLog(@"%c ASCII is @%hhd", aChar, aChar);

NSLog(@"%hhu", uChar);

**4) int:** It is primitive data type which holds both signed and unsigned the integer value. The string format "%d" is used to print the integer value given to the variable, "%u" to print unsigned integer value.

Eg: int aInt = -67562;

unsigned int unInt = 76876;

NSLog(@"%d", aInt);

NSLog(@"%u", unInt);

**5) long:** It is primitive data type which holds both signed and unsigned long values. The string format "%ld" is used to print the long value given to the variable, "%u" to print unsigned long value.

Eg: long aLong = -675627864767;

unsigned long unLong = 76876876846;

NSLog(@"%ld", aLong);

NSLog(@"%lu", unLong);

**6) long long:** It is primitive data type which holds both signed and unsigned the long long value. The string format "%lld" is used to print the value given to the variable of type long long, "%llu" to print unsigned long long value.

Eg: long long aLongLong = -67562786476786565327;

unsigned long long unLongLong = 76876876846765475642;

NSLog(@"%lld", aLongLong);

NSLog(@"%llu", unLongLong);

**7) float:** It is primitive data type which holds both float value. The string format "%f" is used to print the value given to the variable of type float, "%8.2f" determines the padding and number of digits to be printed after decimal point.

Eg: float aFloat = -67.01f;

NSLog(@"%f", aFloat);

NSLog(@"%8.2f", aFloat);

**8) double:** By default the literal decimal values are double, we can explicitly mark it as float data type by adding 'f' at the end of the value assigning to the variable. It is primitive data type which holds double value. The string format "%8.2f" is used to print the value given to the variable of type double, "%e" is used to format the value as scientific notation.

Eg: double aDouble = -67.01;

NSLog(@"%8.2f", aDouble);

NSLog(@"%e", aDouble);

**9) long double:** By default the literal decimal values are double, we can explicitly mark it as long double by adding 'L' at the end of the value assigning to the variable. It is primitive data type which holds long double value. The string format "%Lf" is used to print the value given to the variable of type long double, "%Le" is used to format the value as scientific notation.

Eg: long double aLongDouble = -67.01e8L;

NSLog(@"%Lf", aLongDouble);

NSLog(@"%Le", aLongDouble);

**Arithmetic Operations:**

As similar to C programming we can perform the basic arithmetic operations such as, addition, subtraction, multiplication, division and modulus.

For eg.:NSLog(@"6+3=%d", 6+3); //9

NSLog(@"6-3=%d", 6-3); //3

NSLog(@"6\*3=%d", 6\*3); //18

NSLog(@"6/3=%d", 6/3); //2

NSLog(@"6%3=%d", 6%3); //0