

CCA-101: Fundamentals of IT & Programming

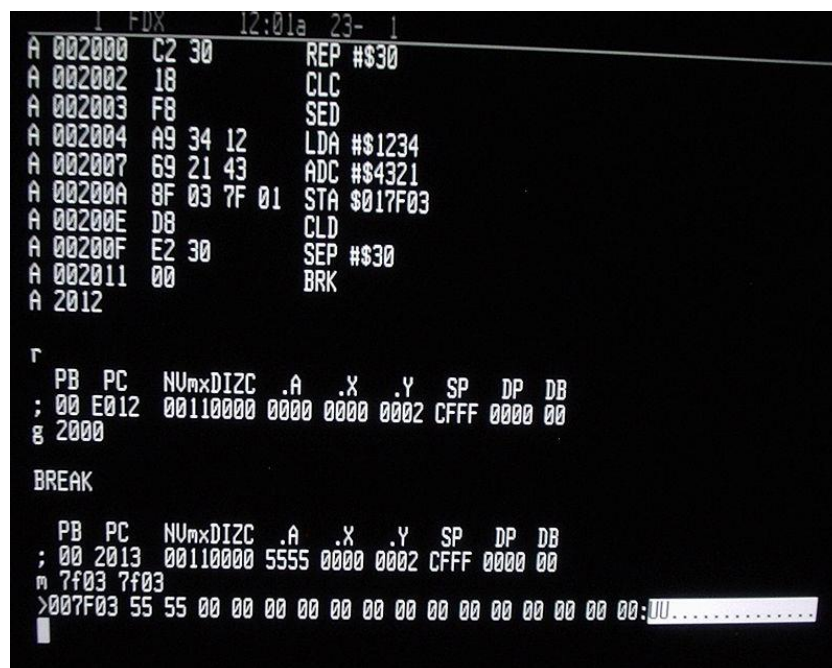
ASSIGNMENT – 2

Q.1. What is the difference between Machine Language and High-Level Language?

Ans. MACHINE LANGUAGE –

Machine language, or machine code, is a low-level language comprised of binary digits (ones and zeros). High-level languages, such as Swift and C++ must be compiled into machine language before the code is run on a computer. Since computers are digital devices, they only recognize binary data. In computer programming, machine code, consisting of machine language instructions, is a low-level programming language used to directly control a computer's central processing unit (CPU). Each instruction causes the CPU to perform a very specific task, such as a load, a store, a jump, or an arithmetic logic unit (ALU) operation on one or more units of data in the CPU's registers or memory.

A much more human friendly rendition of machine language, called assembly language, uses mnemonic codes to refer to machine code instructions, rather than using the instructions' numeric values directly, and uses symbolic names to refer to storage locations and sometimes registers. For example, on the Zilog Z80 processor, the machine code 00000101, which causes the CPU to decrement the B processor register, would be represented in assembly language as DEC B..



```
1 FOX 12:01a 23- 1
A 002000 C2 30 REP #$30
A 002002 18 CLC
A 002003 F8 SED
A 002004 A9 34 12 LDA #$1234
A 002007 69 21 43 ADC #$4321
A 00200A 8F 03 7F 01 STA $017F03
A 00200E D8 CLD
A 00200F E2 30 SEP #$30
A 002011 00 BRK
A 2012

r
PB PC NUMxIIZC .A .X .Y SP DP DB
; 00 E012 00110000 0000 0000 0002 CFFF 0000 00
B 2000

BREAK

PB PC NUMxIIZC .A .X .Y SP DP DB
; 00 2013 00110000 5555 0000 0002 CFFF 0000 00
m 7f03 7f03
>007F03 55 55 00 00 00 00 00 00 00 00 00 00 00 00 00 00:UU.....
```

HIGH LEVEL LANGUAGE –

In computer science, a high-level programming language is a programming language with strong abstraction from the details of the computer. In contrast to low-level programming languages, it may use natural language elements, be easier to use, or may automate (or even hide entirely) significant areas of computing systems (e.g. memory management), making the process of developing a program simpler and more understandable than when using a lower-level language. The amount of abstraction provided defines how "high-level" a programming language is.

In the 1960s, high-level programming languages using a compiler were commonly called autocodes.[2] Examples of autocodes are COBOL and Fortran.[3]A high-level language is a programming language designed to simplify computer programming. It is "high-level" since it is several steps removed from the actual code run on a computer's processor. High-level source code contains easy-to-read syntax that is later converted into a low-level language, which can be recognized and run by a specific CPU. High-level language computer architecture[edit]

Alternatively, it is possible for a high-level language to be directly implemented by a computer – the computer directly executes the HLL code. This is known as a high-level language computer architecture – the computer architecture itself is designed to be targeted by a specific high-level language. The Burroughs large systems were target machines for ALGOL 60, for example.

Most common programming languages are considered high-level languages. Examples include:

C++, C#, Cobol, Fortran, Java, JavaScript, Objective C, Pascal, Perl, PHP, Python, Swift.

Q.2. Discuss about different data types of C programming Language.

Ans. Data types in C Language –

Data types specify how we enter data into our programs and what type of data we enter. C language has some predefined set of data types to handle various kinds of data that we can use in our program. These datatypes have different storage capacities.

C language supports 2 different type of data types:

1. Primary data types:

These are fundamental data types in C namely integer(int), floating point(float), character(char) and void.

2. Derived data types:

Derived data types are nothing but primary datatypes but a little twisted or grouped together like **array**, **structure**, **union** and **pointer**. These are discussed in details later. Data type determines the type of data a variable will hold. If a variable x is declared as int. it means x can hold only integer values. Every variable which is used in the program must be declared as what data-type it is.

Integer Type –

Integers are used to store whole numbers.

Size and range of Integer type on 16-bit machine:

Type	Size(bytes)	Range
int or signed int	2	-32,768 to 32767
unsigned int	2	0 to 65535
short int or signed short int	1	-128 to 127
unsigned short int	1	0 to 255
long int or signed long int	4	-2,147,483,648 to 2,147,483,647
unsigned long int	4	0 to 4,294,967,295

Floating Point Type –

Floating types are used to store real numbers.

Size and range of Integer type on 16-bit Machine -

Type	Size(bytes)	Range
Float	4	3.4E-38 to 3.4E+38
double	8	1.7E-308 to 1.7E+308
long double	10	3.4E-4932 to 1.1E+4932

Character Type –

Character types are used to store characters value.

Size and range of Integer type on 16-bit Machine -

Type	Size(bytes)	Range
char or signed char	1	-128 to 127
unsigned char	1	0 to 255

VOID TYPE –

Void type means no value. This is usually used to specify the type of functions which returns nothing. We will get acquainted to this datatype as we start learning more advanced topics in C language, like functions, pointers etc.

Q.3. Find the output of the following expressions –

a) $X = 20/5 * 2 + 30 - 5$

b) $Y = 30 - (40/10 + 6) + 10$

c) $Z = 40 * 2 / 10 - 2 + 10$

Ans. a) $x = \frac{20}{5} \times 2 + 30 - 5$

Solution – $x = 33$

b) $y = 30 - \left(\frac{40}{10} + 6\right) + 10$

Solution – $y = 30$

c) $z = 40 \times \frac{2}{10} - 2 + 10$

Solution – $z = 16$

Q.4. Describe the syntax of the following statements:

a) If – else statement

b) for loop

c) while loop

d) do-while loop

Ans. a) If – else Statement –

An if statement can be followed by an optional **else if...else** statement, which is very useful to test various conditions using single if...else if statement.

When using if...else if...else statements, there are few points to keep in mind –

- An if can have zero or one else's and it must come after any else if's.
- An if can have zero to many else if's and they must come before the else.

- Once an else if succeeds, none of the remaining else if's or else's will be tested.

Syntax –

The syntax of an **if...else if...else** statement in C programming language is–

```
If (boolean_expression 1) {  
    /* Executes when the boolean expression 1 is true */  
} else if( boolean_expression 2) {  
    /* Executes when the boolean expression 2 is true */  
} else if( boolean_expression 3) {  
    /* Executes when the boolean expression 3 is true */  
} else {  
    /* executes when the none of the above condition is true */  
}
```

Example –

```
#include <stdio.h>  
  
int main () {  
  
    /* local variable definition */  
    int a = 100;  
  
    /* check the boolean condition */  
    if( a == 10 ) {  
        /* if condition is true then print the following */  
        printf("Value of a is 10\n" );  
    } else if( a == 20 ) {  
        /* if else if condition is true */  
        printf("Value of a is 20\n" );  
    } else if( a == 30 ) {  
        /* if else if condition is true */  
        printf("Value of a is 30\n" );  
    }
```

```

    } else {
        /* if none of the conditions is true */
        printf("None of the values is matching\n" );
    }
    printf("Exact value of a is: %d\n", a );
    return 0;
}

```

b) for Loop –

syntax

```

for (initializationStatement; testExpression; updateStatement)
{
    // statements inside the body of loop
}

```

Example 1: for loop

```

// Print numbers from 1 to 10
#include <stdio.h>

int main() {
    int i;

    for (i = 1; i < 11; ++i)
    {
        printf("%d ", i);
    }
    return 0;
}

```

C) While Loop –

Syntax –

```

while (testExpression)
{
    // statements inside the body of the loop
}

```



```
}
```

Example 1: while loop

```
// Print numbers from 1 to 5  
  
#include <stdio.h>  
  
int main()  
{  
    int i = 1;  
    while (i <= 5)  
    {  
        printf("%d\n", i);  
        ++i;  
    }  
    return 0;  
}
```

d) do-while loop -

Syntax –

```
do {  
    statement(s);  
} while (condition);
```

Example

```
import std.stdio;  
  
int main () {  
    /* local variable definition */  
    int a = 10;  
    /* do loop execution */
```

```

do{
    writeln("value of a: %d", a);

    a = a + 1;

}while( a < 20 );

return 0;

}

```

Q.5. Find the output of the following program segments.

Ans –

a)	b)	c)
<pre> #include <stdio.h> int main() { int i; for (i=1; i<2; i++) { printf("IMS Ghaziabad\n"); } } </pre>	<pre> #include <stdio.h> int main() { int i = 1; while (i <= 2) { printf("IMS Ghaziabad\n"); i = i + 1; } } </pre>	<pre> #include <stdio.h> void main() { int a = 10, b=100; if(a > b) printf("Largest number is %d\n", a); else printf("Largest number is %d\n", b); } </pre>
Output – IMS Ghaziabad IMS Ghaziabad	Output – IMS Ghaziabad IMS Ghaziabad	Output – Largest number is 100