SLIIT

Discover Your Future

IT1010 – Introduction to Programming

Lecture 1 – Part 2

Introduction to C programming

Objectives

- At the end of the Lecture students should be able to
  - Describe the evolution of programming languages
  - Write a simple C program

Computer Program

Computer Program

Set of instructions given to the computer.

Programmers write programs in various programming languages.

Programming languages can be mainly divided into :

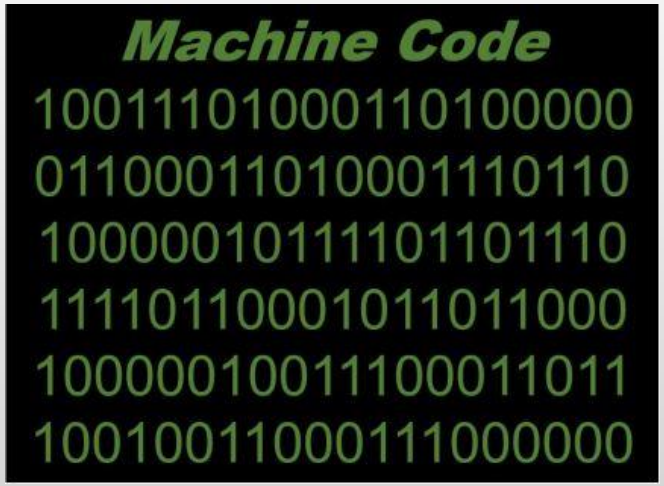
1. Low-level programming languages
   - Machine Language
   - Assembly Language
2. High-level programming languages

Programming Languages

Low-level programming Languages

Machine Languages

- Consist of 1 s and 0 s
- Machine dependent
- Computer can directly understand its own machine language
- Tedious and error prone for programmers



Programming Languages

Assembly Languages

- English-like abbreviations called mnemonics formed the basis
- Clearer to humans but incomprehensible to computers
- Need to translate to machine language using translator programs called assemblers

Example:

```
load  salary
add   bonus
store total
```

Programming Languages

High Level Programming Languages

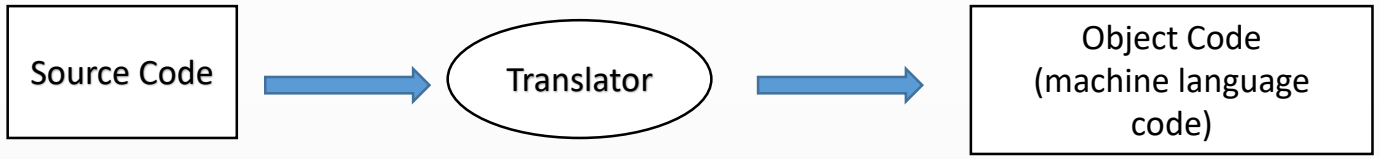
- Instructions look almost like English and mathematical notations
- Substantial tasks can be accomplished from a single statement
- Easy for humans to understand
- Translator programs convert high-level programming languages into machine language

Example:

```
total = salary + bonus;
```

- C, C++, Python, Visual Basic and Java are some of the high level programming languages.

Program Code Translation



Translator

- Assemblers (convert assembly language programs to machine language)
- Compilers (convert high-level language programs to machine language)
- Interpreters (execute high-level language programs line by line)

History of C

- C language was evolved from two previous languages, BCPL and B by Dennis Ritchie at Bell Laboratories in 1972.
- C initially became widely known as the developing language of the UNIX operating system.
- C99 and C11 are revised standards for C programming language that refines and expands the capabilities of C.

Simple C Program

Source Code

```
/* First program in C
This program displays a message */
#include <stdio.h>

// function main begins program
execution
int main(void)
{
        printf("welcome to C!");

        return 0;
} // end of function main
```

Output

```
welcome to C!
```

Simple C Program Description

```
/* First program in C
   This program displays a message */
```

```
// function main begins program execution
```

These are comments. Comments are used to document programs and it improves the program readability. C compiler ignores comments.

Line comments begin with // and continue for the rest of the line. /* … */ multi-line comments can also be used. Everything from /* to */ is a comment.

```
#include <stdio.h>
```

This is a directive to C preprocessor. Lines begin with # are processed by the preprocessor before compiling the program. Here, preprocessor includes the content of stdio.h in the program. stdio.h is a header file which contains information about standard input/output library function calls such as *printf*.

Simple C Program Description

```
int main(void)
```

Every C program has this *main* function and it begins executing at main. "int" to left of main indicates that the function returns an integer. "void" in parentheses indicates that *main* does not receive any information

```
{
```

Left brace , { , begins the body of every function. Functions ends with a corresponding right brace. The portion of the program between the braces is called a block.

```
printf( "welcome to C! " );
```

Entire line is called a statement. It instructs the computer to perform an action. A statement must end with a semicolon. This statement prints a string of characters marked by the quotation marks on the screen.

Simple C Program Description

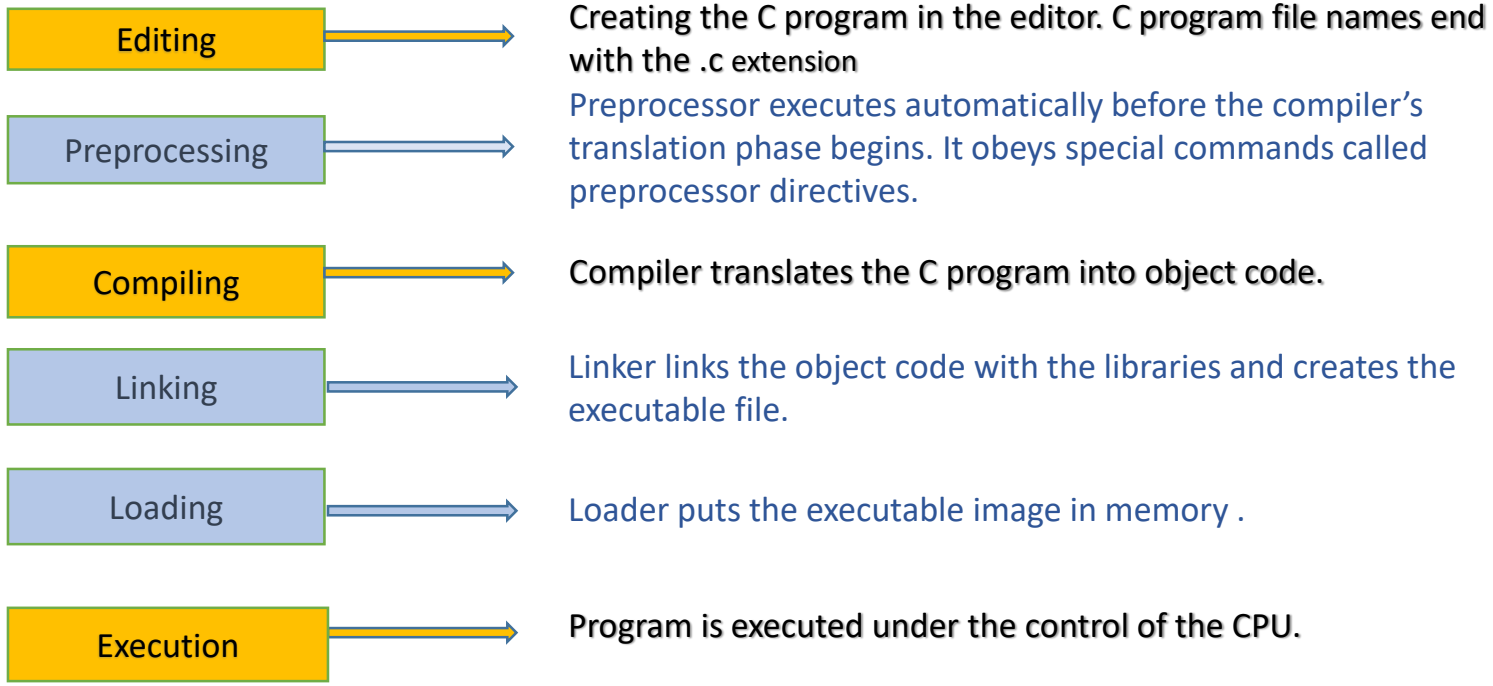
```
return 0;
```

Included at the end of every main function. It is used to exit the main function and the value 0 indicates a successful termination.

Blank Lines and White Space

Blank lines, space characters and tab characters are known as white space. White-space characters are normally ignored by the compiler.

C Program Development Environment



More Examples

```
// A text break into two printf statements

#include <stdio.h>

/* function main begins program execution */
int main(void)
{
         printf( "welcome ");
         printf( "to C!\n");

         return 0;
} // end of main function
```

```
// A text printed in multiple lines using single
printf

#include <stdio.h>

/* function main begins program execution */
int main(void)
{
         printf("welcome \nto C! \n");

         return 0;
} // end of main function
```

- The backslash (\) is called escape character.
- The escape sequence \n means newline.

printf() function

printf(“welcome to C!\n”);

can be written as,

```
puts( “welcome to C!” );
```

puts function adds newline automatically.

printf(“welcome “);

can be written as,

```
printf ( “%s”, “welcome ” );
```