Introduction

Structured Programming Language (CSE-1101)

Course Instructor : Mohammed Mamun Hossain
Assistant Professor, Dept. of CSE, BAUST

Outline

1. Programming Languages
2. Language Translator
3. Basic Structure of C Program
4. Description of simple program
5. Needs of C programming

Uses of Computer

❑Why we use computer?

Simply we say that, it makes our life easier.

❑How or where?

- ➢People in shops, factories, hospitals and schools use computers in lots of different ways to do different types of jobs.
- ➢To solve mathematical equations, communication, analyze data, store information, play games and find information through the Internet.

Application of Computers

- Home
- Education
- Business
- Health Care
- Government
- Media and Communication
- Engineering Design
- Military

and many more…

Application of Computers

How computer understand our command/direction to solve a problem?

Communicating with a Computer

Speaker encodes information

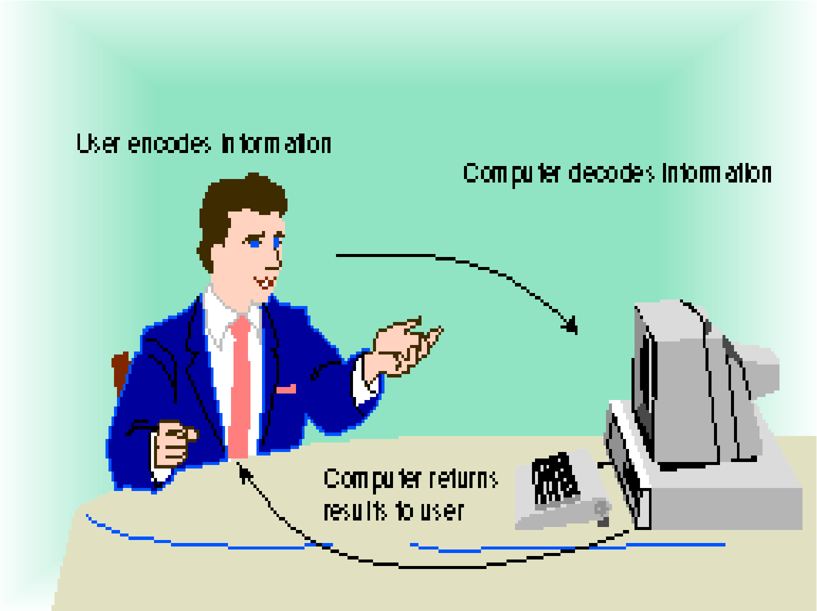
Listener decodes information

Speaker encodes information
Listener decodes information
Listener returns feedback to speaker

Listener returns feedback to speaker

User encodes information

Computer decodes information


User encodes information
Computer decodes information
Computer returns results to user

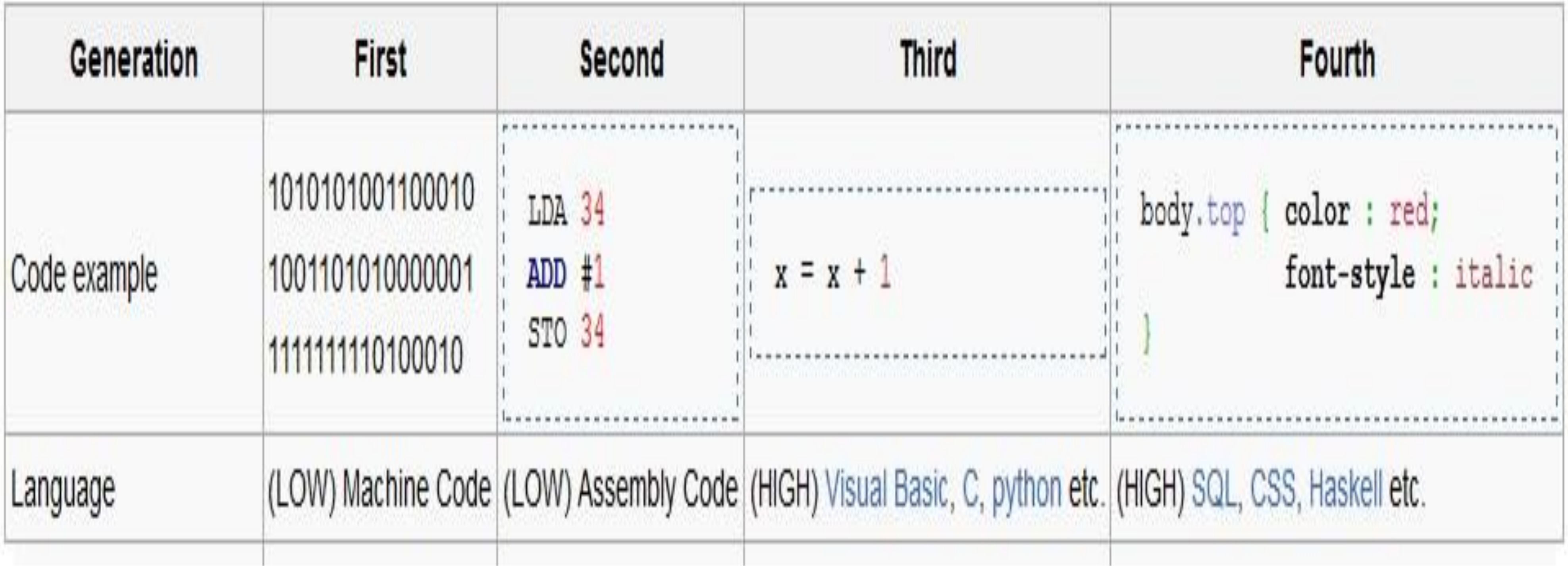
Computer returns results to user

Programs and Programming Languages

- Programming languages bridge the gap between human thought processes and computer binary circuitry.
- **Programming language**: A series of specifically defined commands designed by human programmers to give directions to digital computers.
- Commands are written as sets of instructions, called **programs**.
- All programming language instructions must be expressed in binary code before the computer can perform them.

Programs and Programming Languages

- ❖In the beginning… To use a computer, you needed to know how to program it.
- ❖Today… People no longer need to know how to program in order to use



Generation	First	Second	Third	Fourth

Code example	1010101001100010
	1001101010000001
	1111111110100010
		LDA 34
		ADD #1
		STO 34
			x = x + 1	body.top { color : red;
				font-style : italic
				}

Language	(LOW) Machine Code	(LOW) Assembly Code	(HIGH) Visual Basic, C, python etc.	(HIGH) SQL, CSS, Haskell etc.

✓Fifth Generation - Natural Languages

Definition | Programming Languages

- **First Generation - Machine Language (code):** Machine language programs were made up of instructions written in binary code.

- **Second Generation - Assembly Language**: Assembly language programs are made up of instructions written in mnemonics.

- **Third Generation - People-Oriented Programs**: Instructions in these languages are called statements. High-level languages: Use statements that resemble English phrases combined with mathematical terms needed to express the problem or task being programmed.

- **Fourth Generation - Non-Procedural Languages**: Programming-like systems aimed at simplifying the programmers task of imparting instructions to a computer. Many are associated with specific application packages. Query Languages, Report Writers, Application Generators.

  Object-Oriented Languages: A language that expresses a computer problem as a series of objects a system contains, the behaviors of those objects, and how the objects interact with each other.

Definition | Programming Languages

- Fifth Generation - Natural Languages: Languages that use ordinary conversation in one’s own language.

- Research and experimentation toward this goal is being done.
  - ✓ Intelligent compilers are now being developed to translate natural language (spoken) programs into structured machine-coded instructions that can be executed by computers.
  - ✓ Effortless, error-free natural language programs are still some distance into the future.

Translator | Programming Languages

A computer language translator is a program that translates a set of code written in one programming language into a functional equivalent of the code in another programming language or binary code or intermediate form which computer can understand.

- ✓ Assemblers.
- ✓ Interpreters.
- ✓ Compilers.

Definition | Assembler

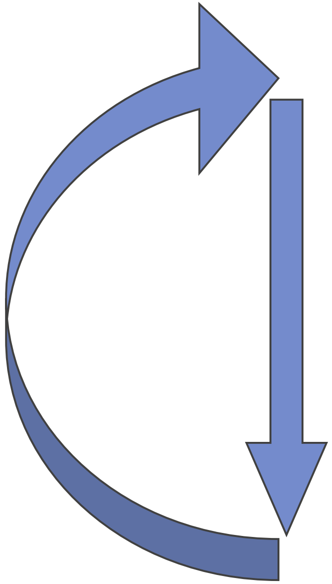
Assembled languages:

- **Assembler:** a program used to translate Assembly language programs.
- Produces one line of binary code per original program statement.
  - ✓ The entire program is assembled before the program is sent to the computer for execution.

Definition | Interpreter

Interpreted Languages:

- ❖ **Interpreter:** A program used to translate high-level programs.
- ❖ Translates one line of the program into binary code at a time:
  - ✓ An instruction is **fetched** from the original source code.
  - ✓ The Interpreter checks the single instruction for errors.
  - ✓ The instruction is translated into binary code.
  - ✓ The binary coded instruction is **executed**.
  - ✓ The fetch and execute process repeats for the entire program.



Definition | Compiler

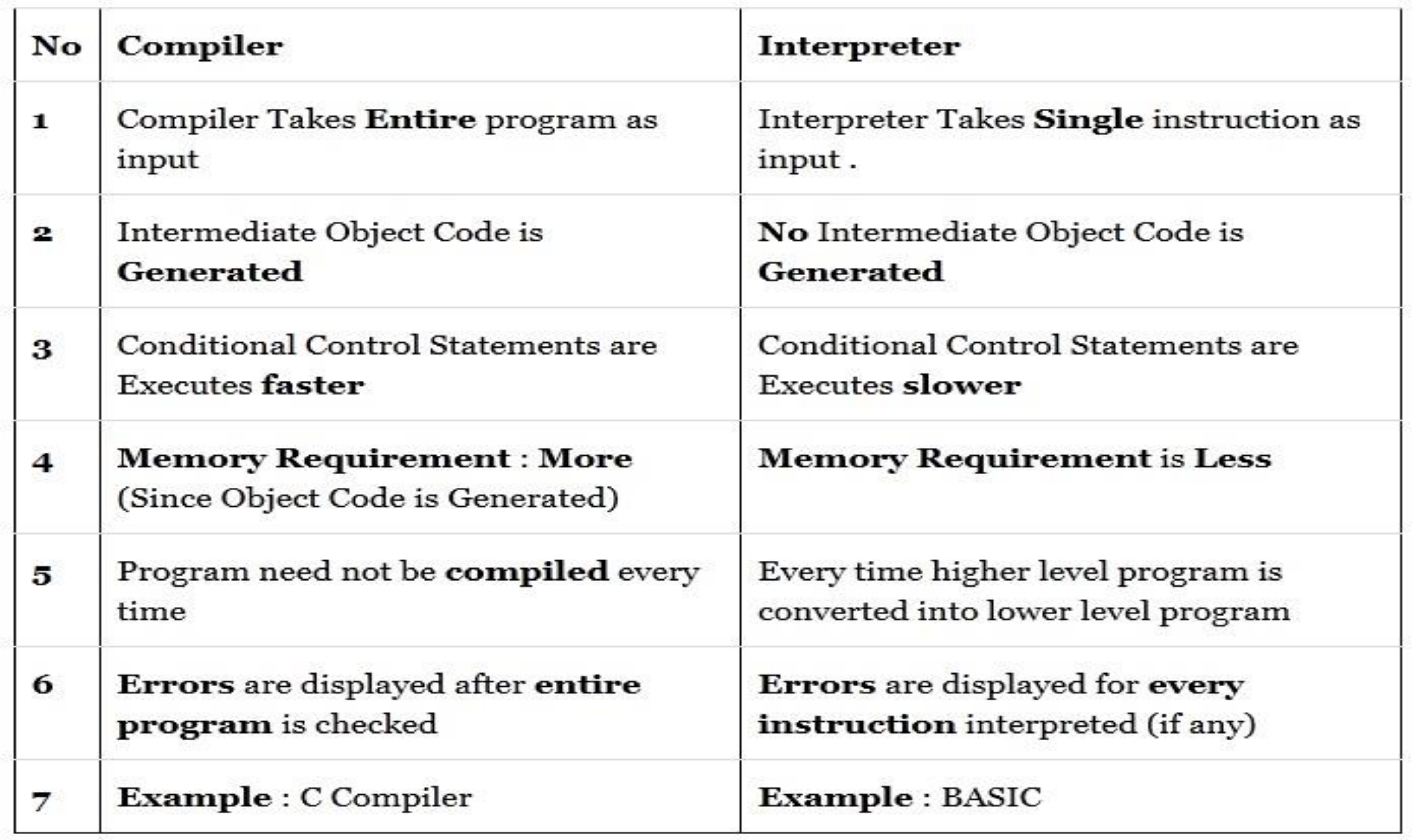
Compiled languages:

- **Compiler:** a program used to translate high-level programs.
- Translates the entire program into binary code before anything is sent to the CPU for execution. The translation process for a compiled program:
    - ✓ First, the Compiler checks the entire program for syntax errors in the original **source code**.
    - ✓ Next, it translates all of the instructions into binary code.
        - Two versions of the same program exist: the original **source code** version, and the binary code version (**object code**).
    - ✓ Last, the CPU attempts execution only after the programmer requests that the program be executed.

Interpreter Vs Compiler

- A complier converts the high level instruction into lower level language (e.g., assembly language or machine code) while an interpreter converts the high level instruction into an intermediate form.
- The compiler executes the entire program at a time, but the interpreter executes each and every line individually.
- List of errors is created by the compiler after the compilation process while an interpreter stops translating after the first error.
- Autonomous executable file is generated by the compiler while interpreter is compulsory for an interpreter program.
- Interpreter is smaller and simpler than compiler
- Interpreter is slower than compiler.

Interpreter Vs Compiler



No	**Compiler**	**Interpreter**
1	Compiler Takes **Entire** program as input	Interpreter Takes **Single** instruction as input .
2	Intermediate Object Code is **Generated**	**No** Intermediate Object Code is **Generated**
3	Conditional Control Statements are Executes **faster**	Conditional Control Statements are Executes **slower**
4	**Memory Requirement** : **More** (Since Object Code is Generated)	**Memory Requirement** is **Less**
5	Program need not be **compiled** every time	Every time higher level program is converted into lower level program
6	**Errors** are displayed after **entire program** is checked	**Errors** are displayed for **every instruction** interpreted (if any)
7	**Example** : C Compiler	**Example** : BASIC

Building A Program

Whatever type of problem needs to be solved, a careful thought out plan of attack, called an algorithm, is needed before a computer solution can be determined.

1) Developing the algorithm.
2) Writing the program.
3) Documenting the program.
4) Testing and debugging the program.

C Programming

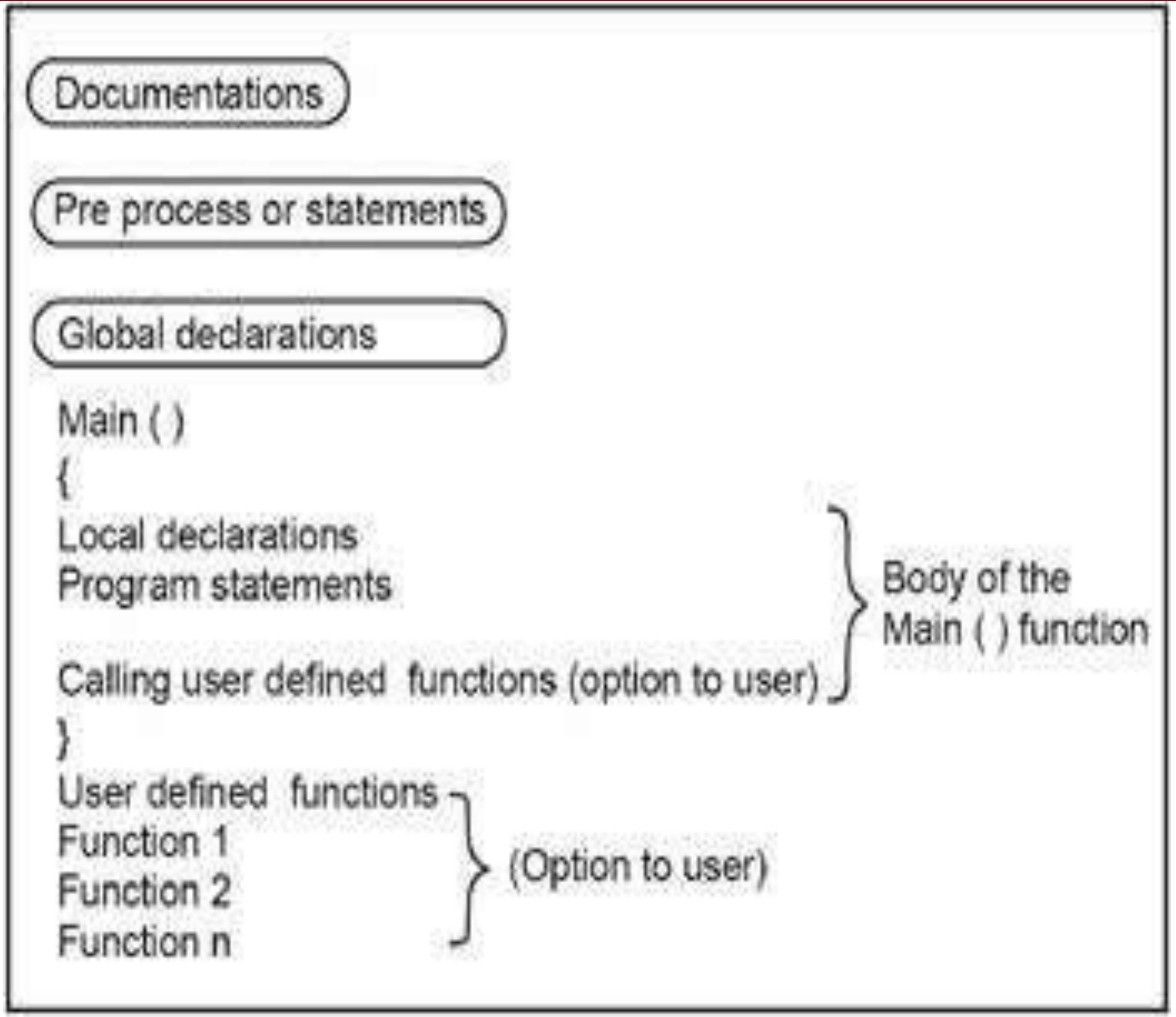
Why C?

- Operating System (OS)
- Embedded System (ES)
- Microcontroller based programming (Robotics)
- System Programming
- Programming Language Development
- Game Engine
- Programming Contest☺

Importance of C:

- C language is efficient and fast.
- C is highly portable.
- C language is well suited for structured programming.
- C is a machine independent language.
- C has the ability to extend itself.

Basic Structure of C Program



Documentations

Pre process or statements

Global declarations

Main ( )
{
Local declarations
Program statements

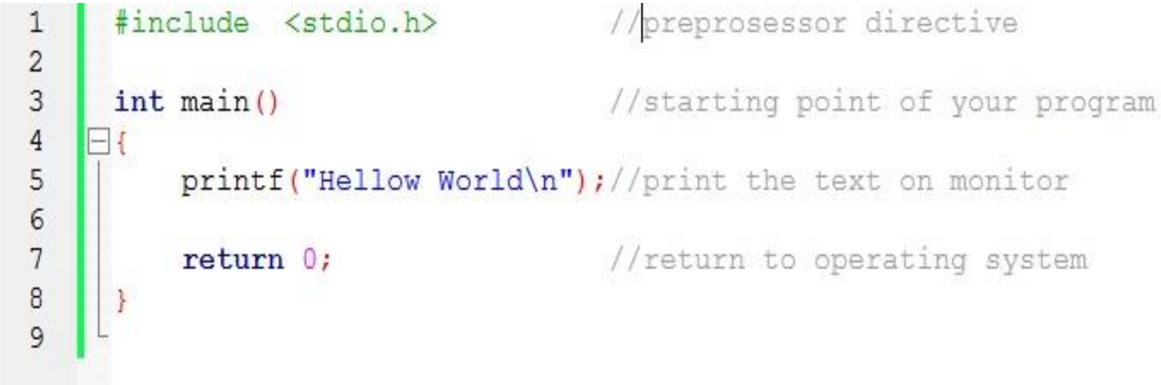
Calling user defined functions (option to user)
}

Body of the
Main ( ) function

User defined functions
Function 1
Function 2
Function n

(Option to user)

Simple C Program

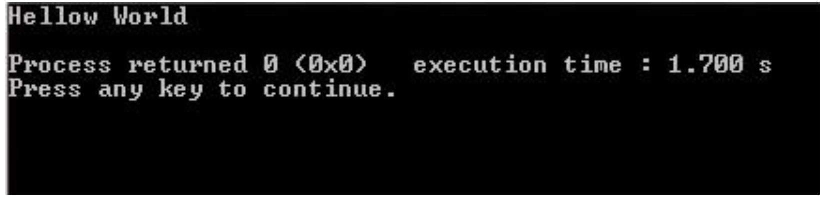


```c
#include  <stdio.h>            //preprosessor directive

int main()                     //starting point of your program
{
    printf("Hellow World\n");//print the text on monitor

    return 0;                  //return to operating system
}
```

Output



```
Hellow World

Process returned 0 (0x0)   execution time : 1.700 s
Press any key to continue.
```

- ❖A **comment** is descriptive text used to help a reader of the program understand its content.
- ❖ A C program line begins with # provides an instruction to the C preprocessor. It is executed **before** the actual compilation is done.
- ❖Every program must have a **function** called **main**. This is where program execution begins.
- ❖ The statement return 0; indicates that main() returns a value of zero to the operating system.

Home Work

- ❑ Find the relation between programming languages and translators (assembler, compiler, interpreter).

- ❑ Write a program which shows the given output using printf():

SL	Output
1.	Your **name**
2.	**Dept. of CSE,** **Bangladesh Army University of Science and Technology,** **Saidpur.**
3.	Your **address** with name, father’s name, mother’s name, village/road, district, division etc.
4.	1, 2, 3, 4, 5, 6, 7, 8, 9, 10
5.	i) ii) iii) iv)

```
i)            ii)              iii)                 iv)

*             * * * * *            *                * * * * *
* *             * * * *           * *                * * * *
* * *             * * *          * * *                * * *
* * * *             * *         * * * *                * *
* * * * *             *        * * * * *                *
```

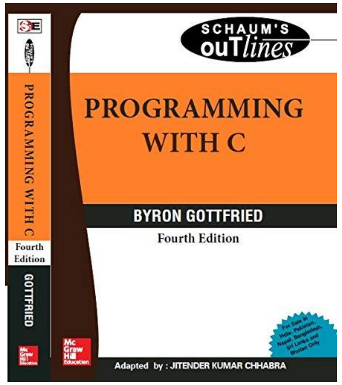
Thank You.

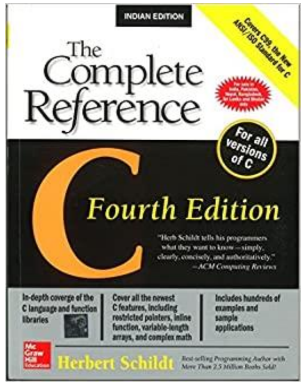
Questions and Answer

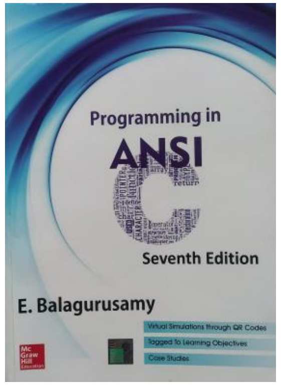
References

Books:

1. Programming With C. *By Byron Gottfried*
2. The Complete Reference C. *By Herbert Shield*
3. Programming in ANSI C *By E. Balagurusamy*
4. Teach yourself C. *By Herbert Shield*







Web:

1. www.wikbooks.org

and other slide, books and web search.