Introduction

Structured Programming Language (CSE-1101)

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

Outline

Overview of C programming language

- **What is C?**
- **Why Learn C?**
- **Motivation Behind Learning C?**
- **Brief History of C Programming Language**
- **Structure of a C Program**
- **Compiling a C program**
- **Compiling Process/8**
- **Compiler vs. Interpreter**

Introduction to C Programming

What is C?

- C is a general-purpose programming language created by Dennis Ritchie at the Bell Laboratories in 1972.
- It is a very popular language, despite being old. The main reason for its popularity is because it is a fundamental language in the field of computer science.
- Used for system programming, embedded systems, and application development.
- C is strongly associated with UNIX, as it was developed to write the UNIX operating system.

UNIX is an operating system that allows multiple users to run multiple programs at the same time. It was originally developed in the 1960s (Windows was launched in 1985) by Dennis Ritchie and Ken Thompson. UNIX is a stable, flexible, and powerful operating system that can be used on servers, desktops, and laptops.

Introduction to C Programming

Where C?

- **System Programming :** Involves writing low-level software that interacts with hardware and the operating system.
  - **Examples:** Operating systems (Linux, Windows).
  - Compilers, device drivers, and networking software.
- **Embedded Systems :** Specialized computing systems embedded into devices to perform specific tasks.
  - **Examples**: Microcontroller programming (Arduino, Raspberry Pi).
  - Automotive control systems, medical devices, and IoT devices.
- **Application Development :** Creating software for end-users, such as desktop, web, or mobile applications.
  - **Examples:** Web browsers, media players, and office software (MS Word, Excel).
  - Mobile apps and enterprise software.

Introduction to C Programming

- **Why learn C?**
  - It is one of the most popular programming languages in the world
  - Foundation for many modern languages (C++, Java, Python).
  - If you know C, you will have no problem learning other popular programming languages such as Java, Python, C++, C#, etc, as the syntax is similar
  - C is very fast and efficient, compared to other programming languages, like Java and Python
  - C is very versatile; it can be used in both applications and technologies
  - Portable and widely used in operating systems.

Brief History of C Programming Language

- **First Generation - Machine Language (1940s-1950s) :** Early computers used machine language, consisting of binary code (0s and 1s). These programs were difficult to write and debug.
- **Second Generation - Assembly Language (1950s-1960s)**Assembly language introduced mnemonics (e.g., MOV, ADD) to make programming more readable. However, it was still hardware-dependent.
- **Third Generation - High-Level Languages (1960s-1970s)**
  - In the late 1960s, languages like Fortran, COBOL, and ALGOL made programming more human-readable.
  - **Birth of C (1972):** Dennis Ritchie developed C at Bell Labs to create the UNIX operating system.
  - C was designed to be powerful, efficient, and portable, making it ideal for system programming.

Brief History of C Programming Language

- **Fourth Generation - Non-Procedural Languages (1980s-Present)**

- Focused on simplifying programming with query languages (SQL), report writers, and automation tools.
- C remained popular and influenced many languages like C++, Java, and Python.
- **Object-Oriented Languages (1980s-Present)**
- C++ (an extension of C) introduced object-oriented concepts like classes and objects.
- Today, C is still widely used in system programming, embedded systems, and performance-critical applications.

Brief History of C Programming Language

- **Fifth Generation Languages (5GL) - AI and Logic-Based Languages**

- The Fifth Generation of Programming Languages (5GL) emerged in the 1980s and beyond, focusing on problem-solving, artificial intelligence, and logic-based programming.

- Unlike previous generations, where programmers defined how a task should be performed, 5GL allows developers to specify what needs to be achieved, and the system figures out the "how”.

Key Characteristics of 5GL

- Logic-based and declarative rather than procedural.
- AI-driven, enabling computers to make decisions.
- Natural language processing (NLP) capabilities.
- Used in expert systems, machine learning, and AI applications.

Brief History of C Programming Language

Examples of 5GL Languages

- Prolog (used in AI and expert systems).
- LISP (used in AI and symbolic computation).
- Mercury (logic programming with efficiency).
- Wolfram Language (used in scientific computation and AI).

How 5GL Differs from C

- C is a 3rd-generation language (3GL) focused on performance and control.
- 5GL languages use AI techniques to process queries and find solutions automatically.

Future of 5GL

With advancements in machine learning, AI, and quantum computing, 5GL languages continue to evolve, making software more autonomous and intelligent.

History| Research and experimentation

- 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.

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

Communicating with a Computer

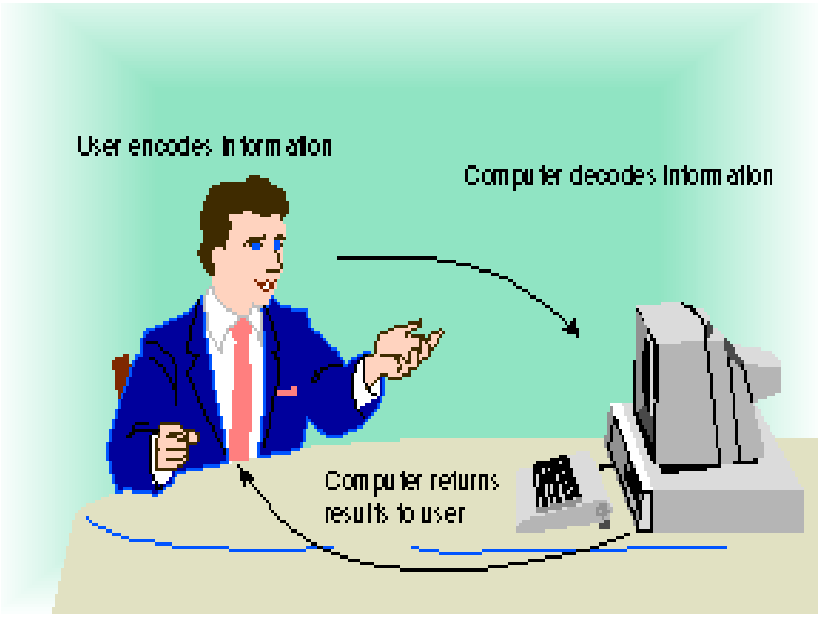
Speaker encodes information

Listener decodes information

User encodes information

Computer decodes information

Listener returns feedback to speaker



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.
- **Programs:** 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.

Structure of a C Program

```c
#include <stdio.h>    // Preprocessor directive
int main() {          // Main function
    printf("Hello, World!\n");  // Output statement
    return 0;         // Return statement
}
```

```
Hello World!
Process returned 0 (0x0) execution time : 0.011 s
Press any key to continue.
```

Explanation:

- `#include <stdio.h>` → Standard input-output header file.
- `main()` → Entry point of the program.
- `printf()` → Function to display output.
- `return 0;` → Ends the program successfully.

C Program Compilation and Execution

Steps to run a C program:

- Write code in a text editor.
- Save file as .c (e.g., program.c)
- Compile using gcc (GNU Compiler Collection)

  gcc program.c -o program
- Run the executable

Common Errors & Debugging:

- Syntax errors (missing semicolon ;).
- Compilation errors (undeclared variables).
- Runtime errors (division by zero, infinite loops).

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

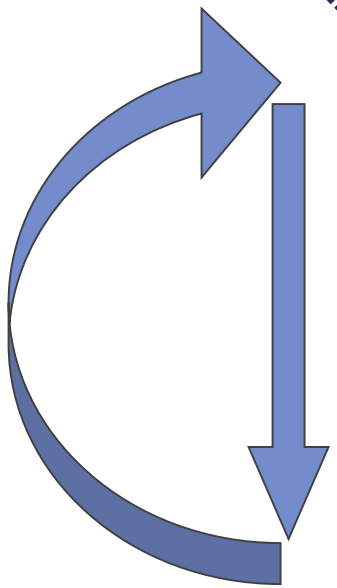
Assembled languages:

- **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

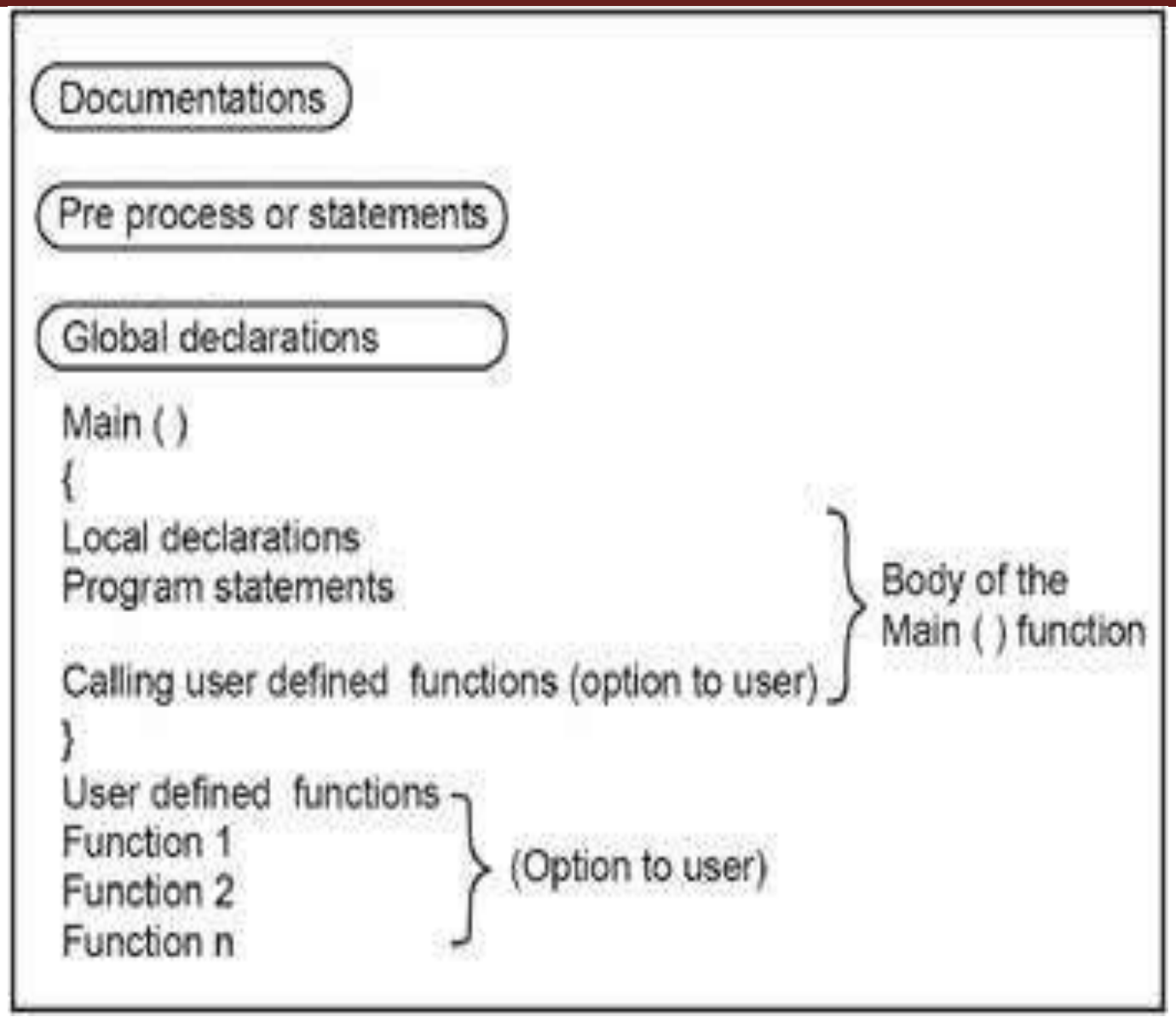
Why C?

- 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



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

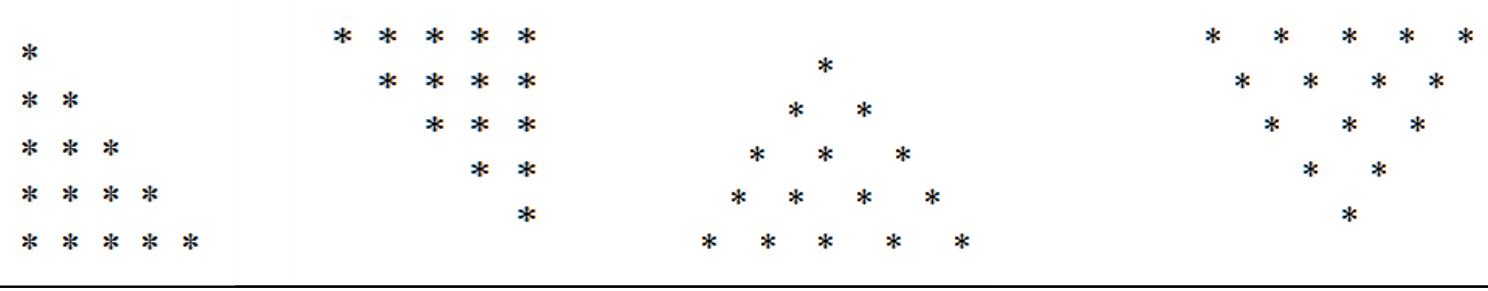
```
Hellow World

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) 

Home Work

- Can you discuss the trade-offs between using an interpreted language versus a compiled language for a project? What factors would you consider when making this decision?

Interpreted Language:

Pros: Faster development, platform portability, easier deployment of source code.

Cons: Slower execution, higher memory usage, limited optimization.

Compiled Language:

Pros: Faster execution, efficient memory use, stronger performance.

Cons: Longer development, platform-specific binaries, more complex deployment.

Consider:

Performance needs, development speed, platform portability, memory constraints, security, team skills, maintenance.

Home Work

Can you discuss the trade-offs between using an interpreted language versus a compiled language for a project? What factors would you consider when making this decision?

In conclusion, the decision between using an interpreted language or a compiled language depends on the specific needs of your project. If performance and optimization are crucial, a compiled language might be more suitable. On the other hand, if rapid development, flexibility, and ease of deployment are important, an interpreted language could be a better fit. In some cases, hybrid approaches are used, where certain performance-critical parts are written in a compiled language and integrated with an interpreted language for higher development speed.

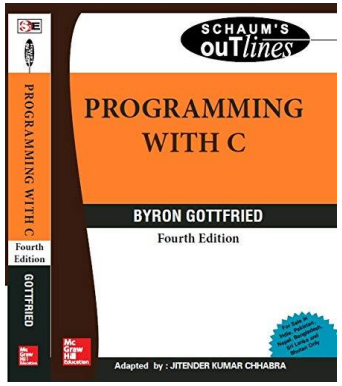
Thank You.

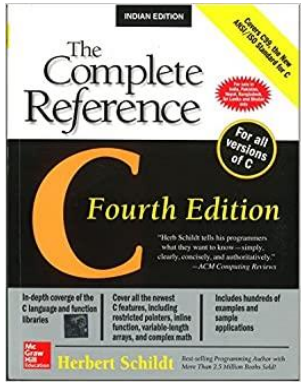
Questions and Answer

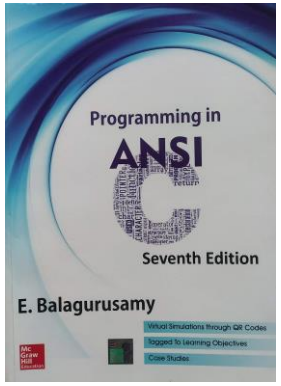
References

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.