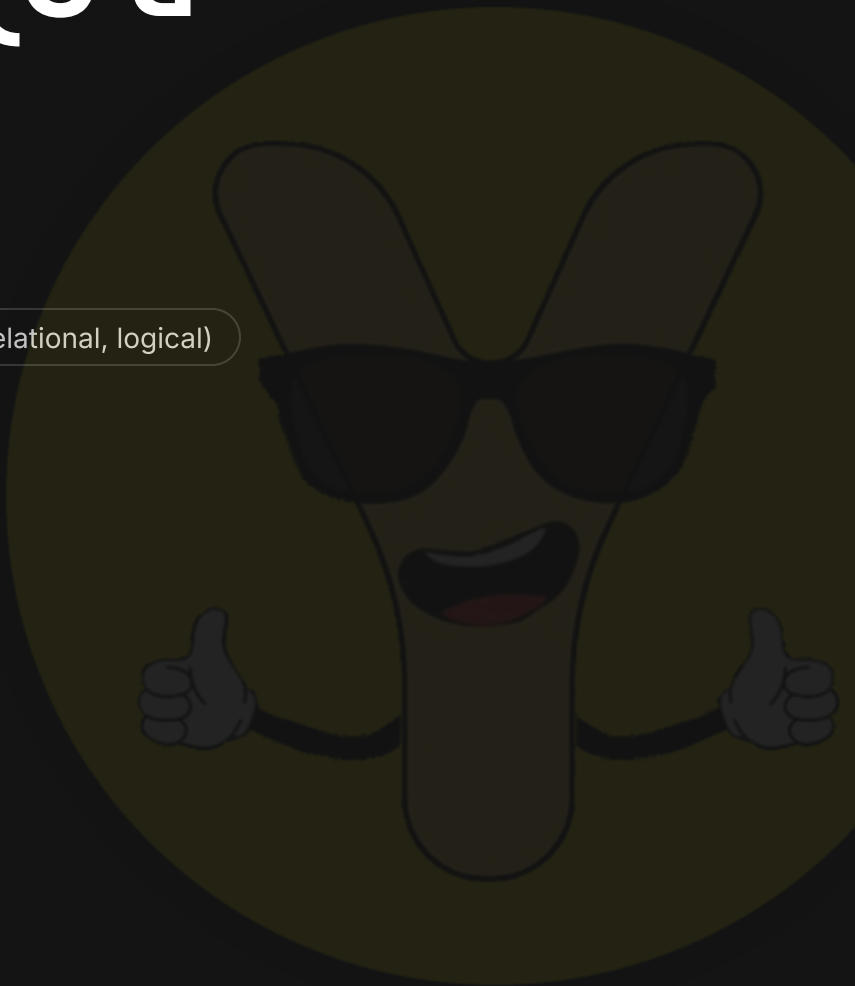
AKTU · 1ST YEAR · SEMESTER 1 · ACS101

Programming Languages (C & Python)

Unit 1 — Programming Basics (C)

- Structure of a C program; compilation and execution
- Data types, variables, constants, operators (arithmetic, relational, logical)
- Input/output — scanf/printf in C
- Type conversion; simple expressions
- Real connect: C powers embedded systems, microcontrollers, and OS kernels.



DECODED BY

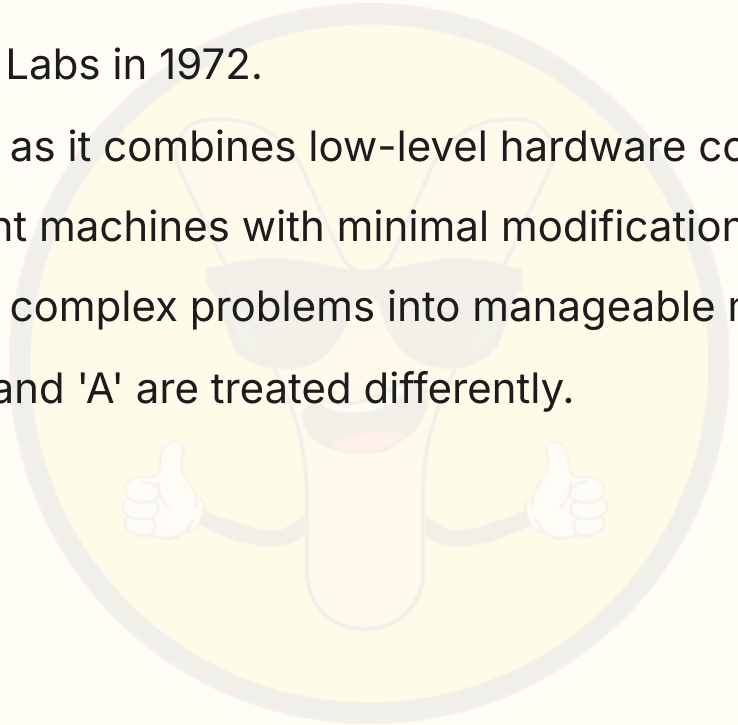
Chirag Jain

Founder, Ycotes · Gold Medalist in Artificial Intelligence, IIT Madras · taught 10,000+ students

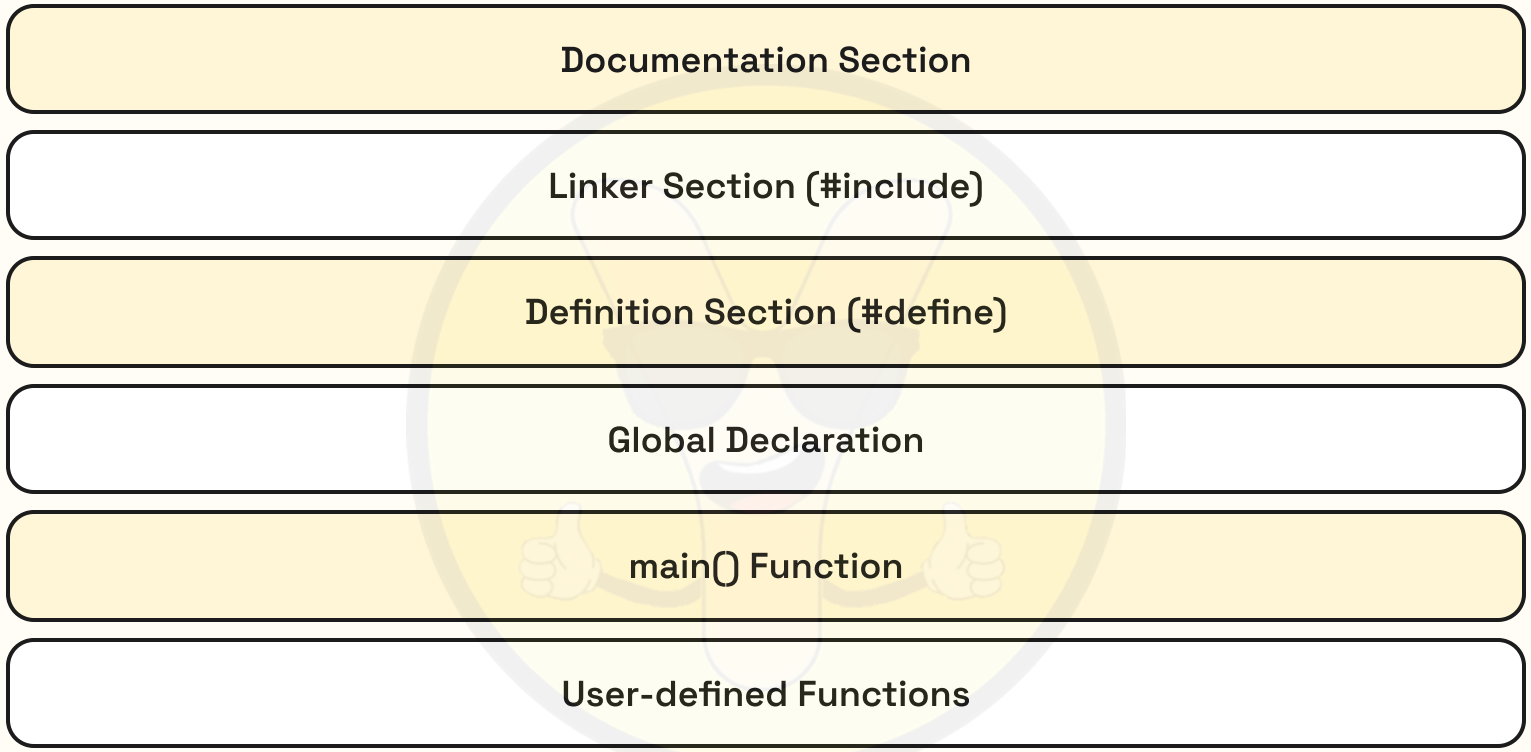
Introduction to C Programming

C is a powerful, procedural programming language widely used for system-level development.

- Developed by Dennis Ritchie at Bell Labs in 1972.
- Known as a 'middle-level' language as it combines low-level hardware control and high-level abstraction.
- Portable: C code can run on different machines with minimal modifications.
- Structured: Uses functions to break complex problems into manageable modules.
- Case-sensitive language where 'a' and 'A' are treated differently.



Structure of a C Program

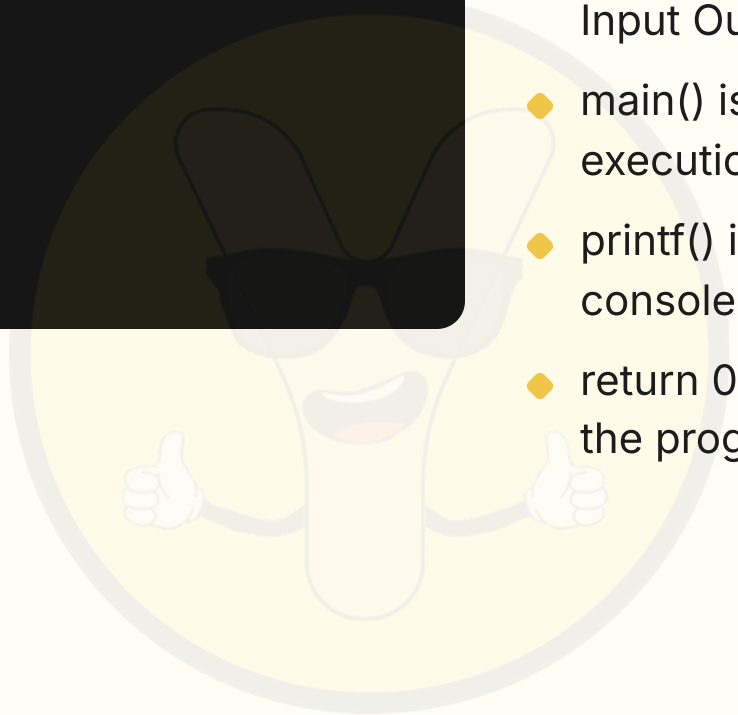


Standard hierarchical structure of a C source file.

Basic C Program Example

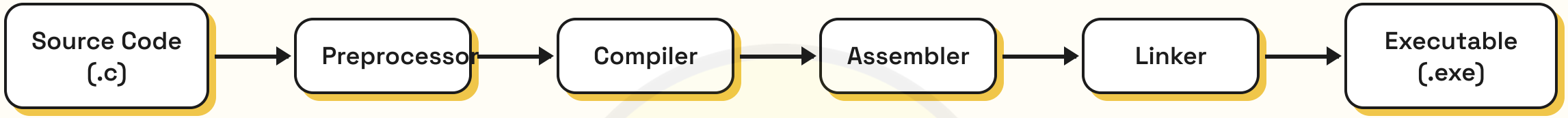
```c
#include <stdio.h>

int main() {
    // Print statement
    printf("Hello, AKTU Students!\n");
    return 0;
}
```



- #include <stdio.h> includes the Standard Input Output library.
- main() is the entry point where program execution begins.
- printf() is used to display output on the console.
- return 0; indicates successful termination of the program.

Compilation and Execution Process

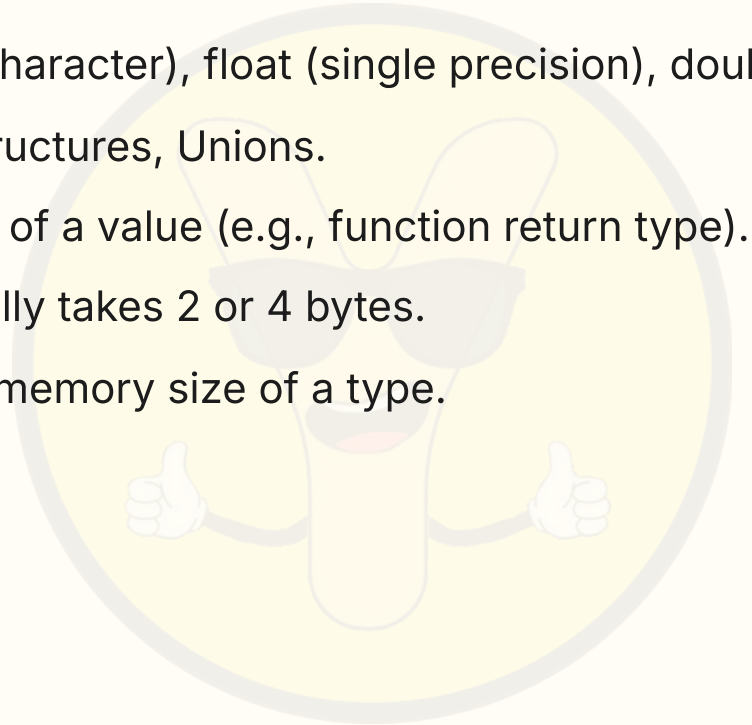


The transformation pipeline from source code to machine-executable binary.

Data Types in C

Data types define the type of data a variable can hold and the memory allocated to it.

- Primary Types: int (integer), char (character), float (single precision), double (double precision).
- Derived Types: Arrays, Pointers, Structures, Unions.
- Void Type: Represents the absence of a value (e.g., function return type).
- Size varies by architecture: int usually takes 2 or 4 bytes.
- Use sizeof() operator to determine memory size of a type.



Variables and Constants

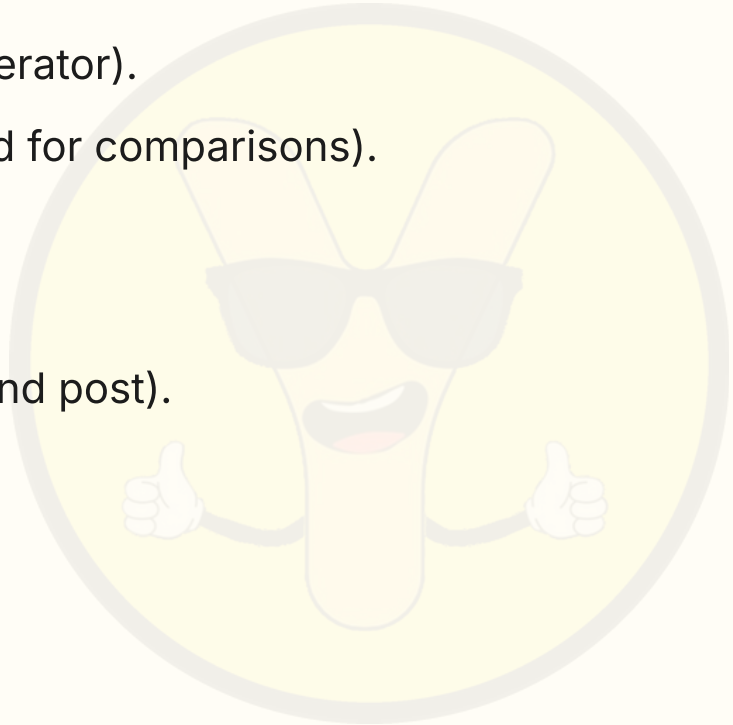
Variables store data, while constants represent fixed values.

- Variable: A named memory location whose value can change during execution.
- Variable Declaration: int age = 20; (type followed by name and optional initialization).
- Constant: A value that cannot be modified after definition.
- Defining Constants: Use 'const' keyword or '#define' preprocessor directive.
- Naming Rules: Must start with a letter or underscore; no special characters allowed.

Operators in C

Operators are symbols that perform specific mathematical or logical manipulations.

- Arithmetic: +, -, *, /, % (modulus operator).
- Relational: ==, !=, >, <, >=, <= (used for comparisons).
- Logical: && (AND), || (OR), ! (NOT).
- Assignment: =, +=, -=, *=, /=
- Increment/Decrement: ++, -- (pre and post).

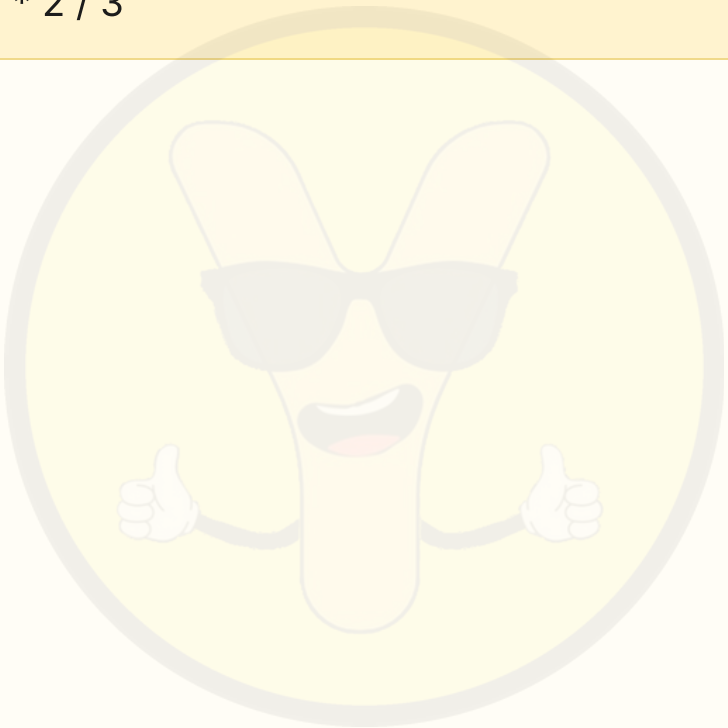


Arithmetic Expression Example

Evaluate the expression: result = (10 + 5) * 2 / 3

1. Parentheses first: (10 + 5) = 15
2. Multiplication: 15 * 2 = 30
3. Division: 30 / 3 = 10

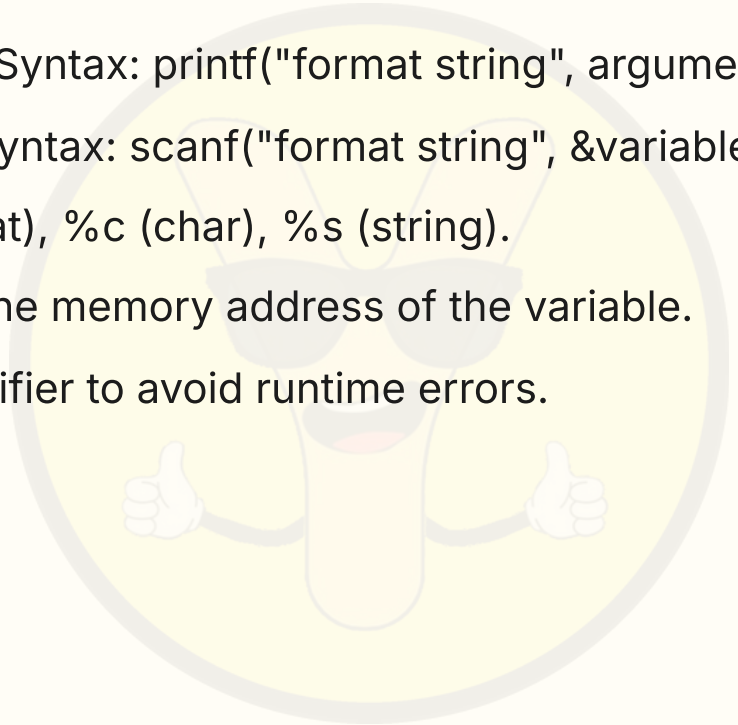
The final value of result is 10.



Input and Output in C

Standard I/O functions are provided by the stdio.h header file.

- printf(): Used for formatted output. Syntax: printf("format string", arguments);
- scanf(): Used for formatted input. Syntax: scanf("format string", &variable);
- Format Specifiers: %d (int), %f (float), %c (char), %s (string).
- The '&' operator in scanf provides the memory address of the variable.
- Always use the correct format specifier to avoid runtime errors.



Type Conversion

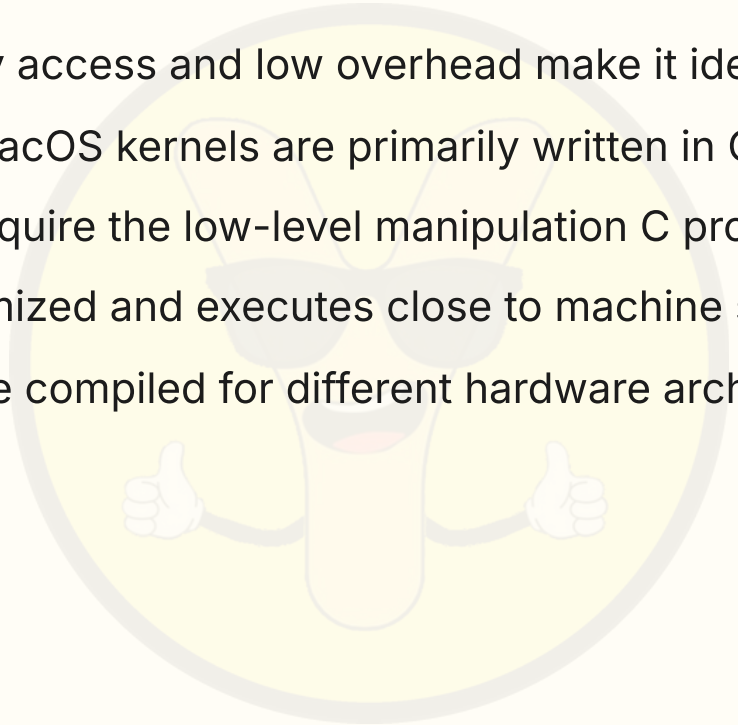
C allows converting one data type to another to perform operations.

- Implicit Conversion: Automatic conversion by the compiler (e.g., int to float).
- Explicit Conversion (Type Casting): Manual conversion using the cast operator.
- Syntax: (type_name) expression;
- Example: float x = (float) 10 / 3; // Result is 3.333
- Data loss may occur when converting from larger to smaller types.

Real World Connect: Why C?

C remains the backbone of modern computing infrastructure.

- Embedded Systems: Direct memory access and low overhead make it ideal for microcontrollers.
- OS Kernels: Linux, Windows, and macOS kernels are primarily written in C.
- Drivers: Hardware device drivers require the low-level manipulation C provides.
- Performance: C code is highly optimized and executes close to machine speed.
- Portability: The same C code can be compiled for different hardware architectures.



Exam corner

- **Q1** Explain the structure of a C program with a neat diagram.
- **Q2** Differentiate between implicit and explicit type conversion with examples.
- **Q3** What are the different types of operators in C? Explain with examples.
- **Q4** Explain the role of the preprocessor, compiler, and linker in C.
- **Q5** Write a C program to take two numbers as input and print their sum.
- **Q6** Why is C preferred for embedded systems and OS kernel development?
- **Q7** Explain the significance of scanf() and printf() format specifiers.

Unit 1 done. You are exam-ready.

Revise the Exam corner once more, then open Unit 2 on theycotes.com. Every AKTU subject, unit by unit, free. Notes by Chirag Jain, Ycotes.

