Programming

Fundamentals

Table of Contents

- IDE and compiler Setup
- C programming Basics
- Escape sequences
- Variables
- Data types in C
- Keywords and identifiers
- Format specifiers
- Variable scope
- Activity

Introduction

Name: Zainab Fatima
Background: Final-year BS Computer Science (CGPA: 3.89), Data Science Intern at 10Pearls and NCBC
Expertise: AI/ML, Software Development, Data Analysis, LLM Applications

SECTION 1:

Setup & First C Program

Introduction to C Programming

Today's Goals:

- Set up programming environment (IDE + Compiler)
- Write and run your first C program
- Understand basic C syntax and structure
- Practice with variables and data types

Why C Programming? Foundation language that powers operating systems, embedded systems, and modern software development.

IDE and Compiler Setup

What You Need:

1. **IDE (Integrated Development Environment):** Your coding workspace
2. **Compiler:** Translates your code to machine language

C Programming Basics

First C Program:

Hello World: Traditional First Program

```c
#include <stdio.h>

int main() {
    printf("Hello, World!");
    return 0;
}
```

C Programming Basics

First C Program:

What Each Line Does:

- **#include <stdio.h>** → Import input/output tools
- **int main() {** → Program starting point begins
- **printf("Hello, World!");** → Display text on screen
- **return 0;** → Tell system: program finished successfully
- **}** → Program starting point ends

C Program Structure

Essential Components:

1. **Preprocessor Directives** → #include <stdio.h>
2. **main() Function** → Entry point where execution begins
3. **Braces { }** → Group code blocks together
4. **Statements** → Instructions ending with semicolon ;

Key Rule: main() is where your program starts running, like the "**Start**" button!

Interactive Challenge 1

Challenge: Change the Hello World program to display:

```
Welcome to Programming Fundamentals CT-175!
Today we learn C programming online.
```

Escape Sequences

Escape Sequence	Meaning
\n	New Line
\t	Horizontal Tab
\b	BackSpace
\r	Carriage Return
\a	Audible bell
\'	Printing single quotation
\"	printing double quotation
\?	Question Mark Sequence
\\	Back Slash
\f	Form Feed
\v	Vertical Tab
\0	Null Value
\nnn	Print octal value
\xhh	Print Hexadecimal value

Special Characters for Formatting

Problem: How do you create new lines, tabs, or quotes in your output?

Solution: Escape sequences (codes starting with \)

Interactive Challenge 2

Challenge: Write a C program that displays the following output exactly as shown using escape sequences (\n and \t):

```
Name:    John Doe
Age:     20
Class:   BSCS
```

Interactive Challenge 3

Challenge: Write a C program to print the following shape using escape sequences.

Expected Output:

```
*       *       *

**      **      **

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

SECTION 2:

Core Concepts

Variables

Variables = Labeled Storage Containers

Real-World Analogy: Think of variables like labeled medicine bottles:

1. Bottle labeled "Aspirin" contains aspirin tablets
2. Variable labeled "age" contains age number

In C Programming:

```
int age;          // declaration and definition of the variable
age = 20;         // assigning the value

int weight = 65;   // definition and initialization
```

Variables

Variable Rules:

- Must declare before using
- One value at a time
- Value can change during program

MINI ACTIVITY: Type in chat: 3 variables you'd need for a student record system

Data Types in C:

Data Types:

“Different Containers for Different Data”

Why Different Types:

Just like you use different containers for liquids vs solids, C uses different types for different data.

Data Types in C:

C Basic Data Types	32-bit CPU		64-bit CPU	
	Size (bytes)	Range	Size (bytes)	Range
char	1	-128 to 127	1	-128 to 127
short	2	-32,768 to 32,767	2	-32,768 to 32,767
int	4	-2,147,483,648 to 2,147,483,647	4	-2,147,483,648 to 2,147,483,647
long	4	-2,147,483,648 to 2,147,483,647	8	-9,223,372,036,854,775,808-9,223,372,036,854,775,807
long long	8	9,223,372,036,854,775,808-9,223,372,036,854,775,807	8	9,223,372,036,854,775,808-9,223,372,036,854,775,807
float	4	3.4E +/- 38	4	3.4E +/- 38
double	8	1.7E +/- 308	8	1.7E +/- 308

Data Types in C:

Domain-Specific Examples:

- **Medical:** int pulse = 72; float temperature = 98.6;
- **Engineering:** double voltage = 12.567; char phase = 'A';
- **Academic:** int credits = 18; float gpa = 3.85;

Keywords and Identifiers

Keywords (Reserved by C - Cannot Use):

int, float, double, char, if, else, while, for,
main, return, const, void, switch, case, break, continue

These have special meanings in C language

Identifiers

Identifiers (Your Variable Names):

Valid Names:

- studentAge, patient_weight, area_circle
- temperature1, voltage_AC, _count

Invalid Names:

- 2student (starts with number)
- patient-weight (contains hyphen)
- int (reserved keyword)
- student age (contains space)

Identifiers

Best Practices:

- Use descriptive names: **patientAge** not **x**
- Use **camelCase** or **snake_case** consistently
- Keep names meaningful but concise

Format Specifiers:

“Communication Bridge: Program ↔️ User”

Output with printf():

```
int age = 25;
float weight = 65.5;
char grade = 'A';

printf("Age: %d years\n", age);          // %d for integers
printf("Weight: %.1f kg\n", weight);     // %f for floats
printf("Grade: %c\n", grade);            // %c for characters
```

Format Specifiers:

Input with scanf():

```
int marks;
printf("Enter your marks: ");
scanf("%d", &marks); // & is address operator
```

Format Specifier Reference:

- **%d** → int (whole numbers)
- **%f** → float (use **%.2f** for **2** decimal places)
- **%lf** → double (for scanf with double)
- **%c** → char (single character)

QUICK PRACTICE:

Type in chat - What format specifier for student's GPA?

OPTIONS:

A. %d
B. %f
C. %c
D. %lf

Variable Scope

Where Can Variables Live?

Local Variables (Function Scope):

```
#include <stdio.h>

int main() {
    int localAge = 20;       // Only exists inside main()
    printf("Age: %d", localAge);
    return 0;
}
// localAge dies here - cannot use outside main()
```

Variable Scope

Global Variables (Program Scope):

```
#include <stdio.h>

int globalCounter = 0;       // Exists everywhere

int main() {
   globalCounter = 5;       // Can use global variable
   printf("Counter: %d", globalCounter);
   return 0;
}
```

Golden Rule: Variables live only within the **{ }** where they're declared.

QUICK PRACTICE:

Type in chat - If I declare int score inside main(), can I use it outside main()?

OPTIONS:

A. Yes
B. No
C. Sometims.

SECTION 3:

Practice & Application

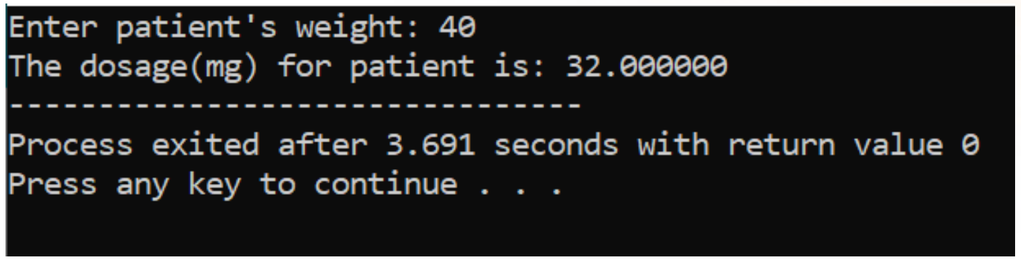
Hands-On Activity 1:

Medical Dosage Calculator

Real-World Problem:

Calculate medicine dosage based on patient weight.
Formula: Dosage (mg) = Patient Weight (kg) × 0.8.

Expected Output:



Enter the patient's weight: 40
The dosage(mg) for the patient is 32.00

Hands-On Activity 2:

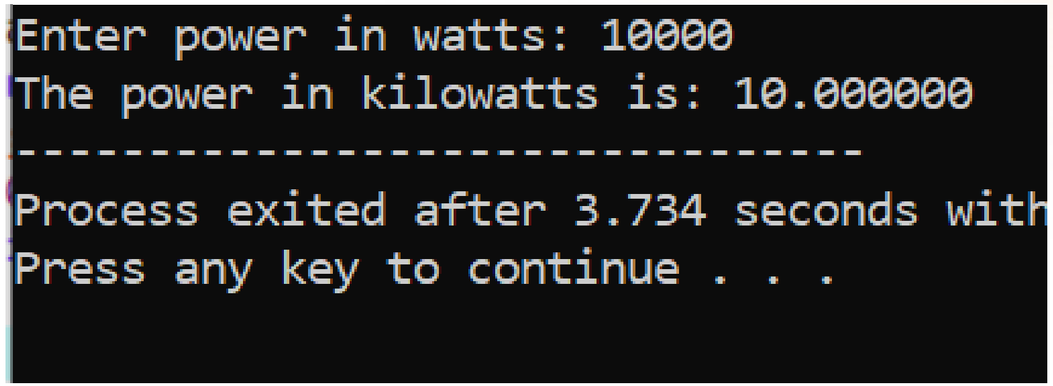
Engineering Unit Converter

Real-World Problem:

Convert electrical power from watts to kilowatts.

Formula: Kilowatts = Watts ÷ 1000

Expected Output:



Enter power in watts: 10000

The power in kilowatts is: 10.00

Thank You