CSR BOOTCAMP-25

C PROGRAMMING

17/09/2025 TO 22/09/2025

Henkel

Day 1 Agenda

Introduction to C Programming

HOUR 1

Setup & Introduction :

- *What is C ?*
- *Tool setup*
- *First C Program*
- *Structure Explanation*

HOUR 2

Input / Output Operations :

- *Printf*
- *Scanf functions*
- *Comments in C*

HOUR 3

Variables & Types :

- *Variable Declaration & Format Specifiers*
- *Data Types*
- *Type Conversion & Constants*

Henkel

Day 1 Practical Programs

Code Examples and Exercises for Each Hour

Hour 1 Programs

Hello World Program

Basic Program Structure

Hour 2 Programs

Comments Usage & Compilation

Execution Steps

Variable declaration examples

Hour 3 Programs

Data type demonstrations

Constant definitions

Sum of 2 numbers

Homework Assignment

Print welcome message with name & age using variables

Create personal details program

Henkel

What is C?

 General purpose programming language created in 1972 by Dennis Ritchie at Bell Labs

 The main reason for its popularity is because it is a fundamental language in the field of computer science . Used to make the UNIX operating system

Henkel

Why Learn C?

- 🌍 One of the most popular languages
- 🔑 Helps you learn Java, Python, C++ easily
- 🧠 Understand how computer memory works
- ⚡ Very fast compared to other languages
- 🔧 Very versatile – used in many technologies

Henkel

Tool Setup

Install C :

If you want to run C on your own computer, you need two things:

- A text editor, like Notepad, to write C code
- A compiler, like GCC, to translate the C code into a language that the computer will understand

Henkel

Tool Setup

Install IDE :

- An IDE (Integrated Development Environment) is used to edit AND compile the code.
- Popular IDE's include Code::Blocks, Eclipse, and Visual Studio. They can be used to both edit and debug C code.

Henkel

Tool Setup

Steps to Install :

We can find the latest version of Code::blocks at http://www.codeblocks.org/.

Download the mingw-setup.exe file, which will install the text editor with a compiler.

Henkel

First C Program

Let's create our first C file.

Open Code blocks and go to File > New > Empty File.

Write the following C code and save the file as myfirstprogram.c (File > Save File as):

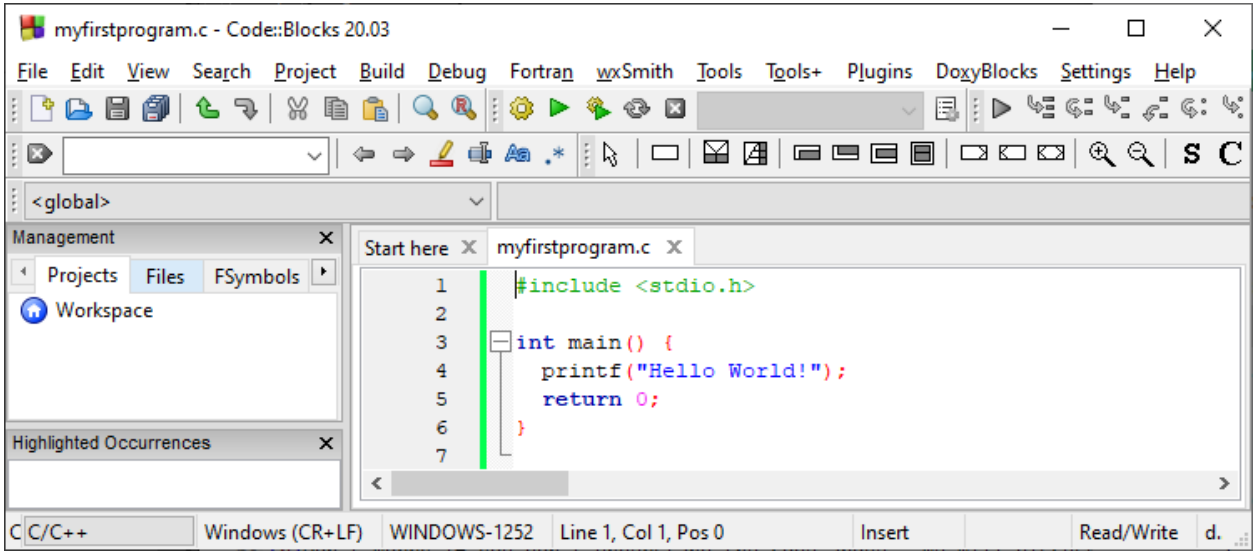
```
#include <stdio.h>

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

Henkel

First C Program

In Code blocks, it should look like this



Henkel

First C Program

Then, go to **Build > Build and Run to run (execute)** the program. The result will look something to this:

```
Hello World!
```

Structure Explanation

Line 1: #include <stdio.h> is a header file library that lets us work with input and output functions, such as printf() (used in line 4). Header files add functionality to C programs.

Line 2: A blank line. C ignores white space. But we use it to make the code more readable.

Line 3: Another thing that always appear in a C program is main(). This is called a function. Any code inside its curly brackets {} will be executed.

Henkel

Structure Explanation

Line 4: printf() is a function used to output/print text to the screen. In our example, it will output "Hello World!".

Line 5: return 0 ends the main() function.

Line 6: Do not forget to add the closing curly bracket } to actually end the main function.

Henkel

printf Functions

printf() is a function used to output/print text to the screen. In our example, it will output "Hello World!".

You can use as many printf() functions as you want. However, note that it does not insert a new line at the end of the output

Henkel

scanf Functions

printf() is a function used to output/print text to the screen. In our example, it will output "Hello World!".

You can use as many printf() functions as you want. However, note that it does not insert a new line at the end of the output

Henkel

Comments in C

Comments can be used to explain code, and to make it more readable. It can also be used to prevent execution when testing alternative code.

Comments can be singled-lined or multi-lined.

Single-line Comments: Single-line comments start with two forward slashes (//).Any text between // and the end of the line is ignored by the compiler (will not be executed).

Example :

```
// This is a comment

printf("Hello World!");
```

Henkel

Quiz

What is the purpose of #include <stdio.h> in a C program?

a. It defines the main() function

b. It allows the use of input and output functions, such as printf()

c. It declares variables

d. It ends the program

Henkel

Quiz

In C, each code statement must end with a semicolon (;).

a. True

b. False

Henkel

Quiz

Which function is used to print text in C?

a. println()

b. printf()

c. cout()

d. echo()

Henkel

Quiz

Which function is used to get user input in C?

a. printf()

b. scanf()

c. pgets()

d. sgets()

Quiz

What is the meaning of comments in C?

a. To output text

b. To create text variables

c. To explain and document code

d. To print text and numbers with a simple line of code

Henkel

Variables

Variables are containers for storing data values, like numbers and characters.

In C, there are different types of variables (defined with different keywords), for example:

- int - stores integers (whole numbers), without decimals, such as **123 or -123**
- float - stores floating point numbers, with decimals, such as **19.99 or -19.99**
- char - stores single characters, such as **'a' or 'B'**. Characters are surrounded by single quotes

Henkel

Variables Declaration

To create a variable, specify the type and assign it a value:

Syntax

type variableName = value;

Where *type* is one of C types (such as int), and *variableName* is the name of the variable (such as x or myName). The equal sign is used to assign a value to the variable.

So, to create a variable that should store a number

Henkel

Variables Declaration Example

*Create a variable called **myNum** of type int and assign the value **15** to it*

int myNum = 15;

You can also declare a variable without assigning the value, and assign the value later

int myNum; // Declare a variable

myNum = 15; // Assign a value to the variable

Henkel

Format Specifiers

Format specifiers are used together with the **printf()**

A format specifier starts with a **percentage sign %,** followed by a character.

- For example, to output the value of an int variable, use the format specifier %d surrounded by double quotes (""), inside the printf() function:

```
int myNum = 15;
printf("%d", myNum);  // Outputs 15
```

Henkel

Data Types

A variable in C must be a specified data type, and you must use a format specifier inside the printf() function to display it

Data type specifies the size and type of information the variable will store.

Data Type	Size	Description	Example
int	2 or 4 bytes	Stores whole numbers, without decimals	1
float	4 bytes	Stores fractional numbers, containing one or more decimals. Sufficient for storing 6-7 decimal digits	1.99
double	8 bytes	Stores fractional numbers, containing one or more decimals. Sufficient for storing 15 decimal digits	1.99
char	1 byte	Stores a single character/letter/number, or ASCII values	'A'

Data Type - Int

Use int when you need to store a whole number without decimals, like 35 or 1000

Size: *2 or 4 bytes*

Format Specifier: *%d*

```
#include <stdio.h>

int main() {

int myNum = 1000;

printf("%d", myNum);

    return 0;
}

Output :

1000
```

Henkel

Data Type - Char

The char data type is used to store a **single** character.

The size of the character is 1 byte. It is the most basic data type in C.

Range: *(-128 to 127) or (0 to 255)*

Size: *1 byte*

Format Specifier: *%c*

```
int main() {
    char ch = 'A'; // Character variable declaration
    printf("ch = %c", ch);
    return 0;
}
```

Output :

```
ch = A
```

Henkel

Data Type - Float

The float data type is used to store single precision floating-point values.

These values are decimal and exponential numbers.

Range: *(1.2E-38 to 3.4E+38)*

Size: *4 bytes*

Format Specifier: *%f*

```
#include <stdio.h>

int main() {

    float val = 12.45;

    printf("val = %f", val);

    return 0;
}
```

Output :

```
val = 12.450000
```

Henkel

Data Type - Double

The double data type is used to store decimal numbers (numbers with floating point values) with double precision.

It can easily accommodate about 16 to 17 digits after or before a decimal point.

Range: *1.7E-308 to 1.7E+308*

Size: *8 bytes*

Format Specifier: *%lf*

```
#include <stdio.h>

int main() {

    double val = 1.4521;

    printf("val = %lf", val);

    return 0;
}
```

Output :

```
val = 1.452100
```

Henkel

Type Conversion

Process of changing one data type into another.

This can happen automatically by the compiler or manually by the programmer.

Type conversion is only performed between data types where such a conversion is possible.

Ex:

```
int x = 5;

int y = 2;

int sum = 5 / 2;

printf("%d", sum); // Outputs 2
```

Henkel

Constants

Variables whose value cannot be changed.

These variables are called constants and are created simply by prefixing const keyword in variable declaration.

```
const data_type name = value;

const int BIRTHYEAR = 1994;
```

Henkel

Quiz

Which data type is used to store integers (whole numbers) in C?

a. char

b. float

c. int

d. double

Henkel

Quiz

What format specifier is used to print an integer value in C?

a. %f

b. %d

c. %c

d. %s

Henkel

Quiz

What happens when you assign a new value to an existing variable in C?

a. The previous value is overwritten

b. The previous value is deleted, and the variable is cleared

c. The new value is ignored

d. An error occurs

Henkel

Quiz

What is the purpose of the const keyword in C?

a. To declare a variable that can be changed

b. To declare a variable that cannot be changed

c. To declare a variable without assigning a value

d. To declare a variable with a default value

Henkel

Day 2 Agenda

Operators and Conditions

HOUR 1

Operators Overview:

- *Operators*
- *Operator Types*

HOUR 2

Conditional Statements:

- *If – else condition*
- *Nested conditions*

HOUR 3

Conditional Statements(ctd):

- *Ternary Operator usage*
- *Switch Case statements*

Henkel

Day 2 Practical Programs

Code Examples and Exercises for Each Hour

Hour 1 Programs

Operator precedence examples

Simple calculator

Hour 2 Programs

Check even/odd

Age check for school trip

Hour 3 Programs

Student Grading System

Maximum of 2 numbers

Homework Assignment

Check if user can play game (age > 10)

Check Leap Year

Henkel

Operators

Operators are used to perform operations on variables and values. In the example below, we use the + operator to add together two values:

```
int myNum = 100 + 50;
```

It can also be used to add together a variable and a value, or a variable and another variable

```
int sum1 = 100 + 50;       // 150 (100 + 50)
int sum2 = sum1 + 250;     // 400 (150 + 250)
int sum3 = sum2 + sum2;    // 800 (400 + 400)
```

Henkel

Operator Types

C divides the operators into the following groups:

- Arithmetic operators
- Assignment operators
- Comparison operators
- Logical operators
- Bitwise operators

Henkel

Arithmetic Operator

Operator	Name	Description	Example
+	Addition	Adds together two values	x + y
-	Subtraction	Subtracts one value from another	x - y
*	Multiplication	Multiplies two values	x * y
/	Division	Divides one value by another	x / y
%	Modulus	Returns the division remainder	x % y
++	Increment	Increases the value of a variable by 1	++x
--	Decrement	Decreases the value of a variable by 1	--x

Henkel

Assignment Operator

Assignment operators are used to assign values to variables.

Operator	Example	Same As
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
&=	x &= 3	x = x & 3
|=	x |= 3	x = x | 3
^=	x ^= 3	x = x ^ 3
>>=	x >>= 3	x = x >> 3
<<=	x <<= 3	x = x << 3

Henkel

Comparison Operator

Comparison operators are used to compare two values (or variables).

Operator	Name	Example	Description
==	Equal to	x == y	Returns 1 if the values are equal
!=	Not equal	x != y	Returns 1 if the values are not equal
>	Greater than	x > y	Returns 1 if the first value is greater than the second value
<	Less than	x < y	Returns 1 if the first value is less than the second value
>=	Greater than or equal to	x >= y	Returns 1 if the first value is greater than, or equal to, the second value
<=	Less than or equal to	x <= y	Returns 1 if the first value is less than, or equal to, the second value

Henkel

Logical Operator

Logical operators are used to determine the logic between variables or values, by combining multiple conditions:

Operator	Name	Example	Description
&&	AND	x < 5 && x < 10	Returns 1 if both statements are true
||	OR	x < 5 || x < 4	Returns 1 if one of the statements is true
!	NOT	!(x < 5 && x < 10)	Reverse the result, returns 0 if the result is 1

Henkel

Conditional Statements

C has the following conditional statements:

- Use if to specify a block of code to be executed, if a specified condition is true
- Use else to specify a block of code to be executed, if the same condition is false
- Use else if to specify a new condition to test, if the first condition is false
- Use switch to specify many alternative blocks of code to be executed

Henkel

If Statement

Use the if statement to specify a block of code to be executed if a condition is true.

```
int x = 20;

int y = 18;

if (x > y) {

  printf("x is greater than y");

}
```

Output :

x is greater than y

Henkel

The else Statement

Use the else statement to specify a block of code to be executed if the condition is false.

```
int time = 20;

if (time < 18) {

  printf("Good day.");

} else {

  printf("Good evening.");

}
```

Output:
"Good evening."

Henkel

The else if Statement

Use the else if statement to specify a new condition if the first condition is false.

```
int time = 22;

if (time < 10) {

  printf("Good morning.");

} else if (time < 20) {

  printf("Good day.");

} else {
  printf("Good evening.");
}

Output:
"Good evening."
```

Henkel

Ternary Operator

There is also a short-hand if else, which is known as the ternary operator because it consists of three operands.

It can be used to replace multiple lines of code with a single line.

It is often used to replace simple if else statements

```
int time = 20;
if (time < 18) {
  printf("Good day.");
} else {
  printf("Good evening.");
}
Output : Good Evening
```

Henkel

Switch Case Statement

Instead of writing many if else statements, you can use the switch statement.

The switch statement selects one of many code blocks to be executed

This is how it works:

- The switch expression is evaluated once
- The value of the expression is compared with the values of each case
- If there is a match, the associated block of code is executed
- The break statement breaks out of the switch block and stops the execution
- The default statement is optional, and specifies some code to run if there is no case match

Henkel

Switch Case Example

```c
#include <stdio.h>

int main() {
  int number = 2;

  switch (number) {
    case 1:
      printf("The number is one\n");
      break;
    case 2:
      printf("The number is two\n");
      break;
    case 3:
      printf("The number is three\n");
      break;
    default:
      printf("Invalid number\n");
  }

  return 0;
}
```

Henkel

Quiz

Which of the following operators checks if two values are equal in C?

a. =

b. ==

c. !=

d. <>

Henkel

Quiz

What is the purpose of the else statement in C?

a. To execute a block of code if the if condition is true

b. To execute a block of code if the if condition is false

c. To create a new condition

d. To end the program

Henkel

Quiz

What is the purpose of the ternary operator in C?

a. To write multiple lines of code within an if...else statement

b. To replace a simple if...else statement with a single line of code

c. To handle complex conditions that require nested if statements

d. To declare a new variable

Henkel

Quiz

What is the purpose of the switch statement in C?

a. To select one of many code blocks to be executed

b. To loop through a set of conditions

c. To compare two values

d. To declare multiple variables

Henkel

Day 3 Agenda

Loops and Arrays

HOUR 1

Loop Structures :

- *While Loop*
- *Do/While Loop*
- *For Loop*

HOUR 2

Loop Structures:

- *Nested Loops*
- *Break & continue*

HOUR 3

Array Fundamentals:

- *Array declaration*
- *Accessing elements , Loop through an array*
- *Array Size & Multi-dimensional array operations*

Henkel

Day 3 Practical Programs

Code Examples and Exercises for Each Hour

Hour 1 Programs

Print 1 to 10

Sum of 1 to N

Hour 2 Programs

Print Tables

Hour 3 Programs

Sum of elements in an array

Homework Assignment

Pattern Printing

Henkel

Loop Structures - While

Loops can execute a block of code as long as a specified condition is reached.

Loops are handy because they save time, reduce errors, and they make code more readable.

- **While Loop -** Loops through a block of code as long as a specified condition is true

```
Syntax :

while (condition) {
  // code block to be executed
}
```

Let's see an example of a while loop in a code block

Henkel

Loop Structures – Do/While

The do/while loop is a variant of the while loop. This loop will execute the code block once, before checking if the condition is true, then it will repeat the loop as long as the condition is true.

Syntax :

```
do {
 // code block to be executed
}
while (condition);
```

Let's see an example of a while loop in a code block

Henkel

Loop Structures – Do/While

The do/while loop is a variant of the while loop. This loop will execute the code block once, before checking if the condition is true, then it will repeat the loop as long as the condition is true.

Syntax :

```
do {
 // code block to be executed
}
while (condition);
```

Let's see an example of a do/while loop in a code block

Henkel

Loop Structures – For

When you know exactly how many times you want to loop through a block of code, use the for loop instead of a while loop:

Syntax :

```
for (expression 1; expression 2;
expression 3) {

  // code block to be executed

}
```

Let's see an example of a for loop in a code block

Henkel

Nested Loop

It is also possible to place a loop inside another loop. This is called a **nested loop**.

The "inner loop" will be executed one time for each iteration of the "outer loop

Let's see an example of a nested loop in a code block

Henkel

Break & Continue

The break statement can also be used to jump out of a loop.

The continue statement breaks one iteration (in the loop), if a specified condition occurs, and continues with the next iteration in the loop.

Let's see an example of break & continue in a code block

Henkel

Arrays

Arrays are used to store multiple values in a single variable, instead of declaring separate variables for each value.

To create an array, define the data type (like int) and specify the name of the array followed by square brackets [].

To insert values to it, use a comma-separated list inside curly braces, and make sure all values are of the same data type: **int myNumbers[] = {25, 50, 75, 100};**

Henkel

Accessing Array Elements

To access an array element, refer to its index number.

Array indexes start with 0: [0] is the first element. [1] is the second element, etc

Example :

```
int myNumbers[] = {25, 50, 75, 100};

printf("%d", myNumbers[0]);


// Outputs 25
```

Henkel

Change an Array Element

To change the value of a specific element, refer to the index number:

Example :

```
int myNumbers[] = {25, 50, 75, 100};

myNumbers[0] = 33;


printf("%d", myNumbers[0]);


// Now outputs 33 instead of 25
```

Henkel

Loop through an Array

You can loop through the array elements with the for loop.

The following example outputs all elements in the myNumbers array:

```
int myNumbers[] = {25, 50, 75, 100};
int i;

for (i = 0; i < 4; i++) {
  printf("%d\n", myNumbers[i]);
}
```

Henkel

Set Array Size

Another common way to create arrays, is to specify the size of the array, and add elements later:

```
#include <stdio.h>

int main() {
  // Declare an array of four integers:
  int myNumbers[4];

  // Add elements to it
  myNumbers[0] = 25;
  myNumbers[1] = 50;
  myNumbers[2] = 75;
  myNumbers[3] = 100;

  printf("%d\n", myNumbers[0]);

  return 0;
}
```

Output:

```
25
```

Henkel

Multidimensional Arrays

A multidimensional array is basically an array of arrays.

Arrays can have any number of dimensions.

Two-Dimensional Arrays : A 2D array is also known as a matrix (a table of rows and columns).

To create a 2D array of integers, take a look at the following example:

int matrix[2][3] = { {1, 4, 2}, {3, 6, 8} };

	COLUMN 0	COLUMN 1	COLUMN 2
ROW 0	1	4	2
ROW 1	3	6	8

Henkel

Quiz

What is the purpose of the while loop in C?

a. To execute a block of code a fixed number of times

b. To execute a block of code as long as a specified condition is true

c. To execute code only if a condition is false

d. To declare multiple variables in a loop

Henkel

Quiz

What is the main difference between a do/while loop and a while loop?

a. The do/while loop checks the condition first

b. The do/while loop executes the code block at least once before checking the condition

c. The do/while loop only runs if the condition is initially false

d. The do/while loop is only used for infinite loops

Henkel

Quiz

What is the main use of a for loop in C?

a. To execute a block of code indefinitely

b. To loop through a block of code a specific number of times

c. To check conditions without executing any code

d. To run a loop based on user input only

Henkel

Quiz

What does the following declaration represent? **int matrix[2][3] = { {1, 4, 2}, {3, 6, 8} };**

a. A 1-dimensional array of 5 elements

b. A 2-dimensional array with 2 rows and 3 columns

c. A 3-dimensional array with 2 rows and 3 columns

d. A 1-dimensional array with 6 elements

Henkel

Day 4 Agenda

Functions & Recursion

HOUR 1

Function Concepts :

- *Functions*
- *Functions Parameters & Arguments*

HOUR 2

Variables in C:

- *Variable Scope in C (Local & Global)*
- *Naming Variables*

HOUR 3

Recursion:

- *Recursion in C*

Henkel

Day 4 Practical Programs

Code Examples and Exercises for Each Hour

Hour 1 Programs
Sum of 2 numbers with function

Hour 2 Programs

Basic Calculator Using Functions

Hour 3 Programs

Add menu with switch-case, Add new operation to calculator

Homework Assignment

Make a calculator that loops until the user wants to quit

Henkel

Functions

A function is a block of code which only runs when it is called.

You can pass data, known as parameters, into a function.

Functions are used to perform certain actions, and they are important for reusing code: Define the code once, and use it many times.

Henkel

Predefined Functions

main() is a function, which is used to execute code, and printf() is a function; used to output/print text to the screen

Example :

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

Henkel

Create a Function

To create your own function, specify the name of the function, followed by parentheses () and curly brackets {}:

Syntax :

```
void myFunction() {
  // code to be executed
}
```

myFunction() is the name of the function

void means that the function does not have a return value

Henkel

Call a Function

Declared functions are not executed immediately. They are "saved for later use" and will be executed when they are called.

To call a function, write the function's name followed by two parentheses () and a semicolon ;

Henkel

Parameters & Arguments

Information can be passed to functions as a parameter. Parameters act as variables inside the function.

Parameters are specified after the function name, inside the parentheses.

When a parameter is passed to the function, it is called an argument.

You can add as many parameters as you want, just separate them with a comma:

Syntax :

```
returnType functionName(parameter1, parameter2, parameter3) {
  // code to be executed
}
```

Henkel

Variables in C – Local Scope

A variable created inside a function belongs to the local scope of that function and can only be used inside that function. A local variable cannot be used outside the function it belongs to.

Example :

```
void myFunction() {
  // Local variable that belongs to myFunction
  int x = 5;

  // Print the variable x
  printf("%d", x);
}

int main() {
  myFunction();
  return 0;
}
```

Henkel

Variables in C – Global Scope

A variable created outside of a function, is called a global variable and belongs to the global scope.

Global variables are available from within any scope, global and local. A variable created outside of a function is global and can therefore be used by anyone:

Example :

```
// Global variable x
int x = 5;

void myFunction() {
  // We can use x here
  printf("%d", x);
}

int main() {
  myFunction();

  // We can also use x here
  printf("%d", x);
  return 0;
}
```

Henkel

Naming Variables

If you operate with the same variable name inside and outside of a function, C will treat them as two separate variables; One available in the global scope (outside the function) and one available in the local scope (inside the function):

Example :

```
// Global variable x
int x = 5;

void myFunction() {
  // Local variable with the same name as the global variable (x)
  int x = 22;
  printf("%d\n", x); // Refers to the local variable x
}

int main() {
  myFunction();

  printf("%d\n", x); // Refers to the global variable x
  return 0;
}
```

Henkel

Recursion

Recursion is the technique of making a function call itself. This technique provides a way to break complicated problems down into simple problems which are easier to solve.

Let's see an example of recursion in code block

Henkel

Quiz

What does the keyword void indicate when used with a function in C?

a. The function returns an integer

b. The function can only be called once

c. The function does not return a value

d. The function can accept any type of parameter

Henkel

Quiz

What will the following code output?

```
void myFunction(char name[]) {
 printf("Hello %s\n", name);
}
int main() {
 myFunction("Alice");
 return 0;
}
```

a. Hello World

b. Hello Alice

c. Alice

d. Hello

Henkel

Quiz

Where can a variable with local scope be accessed?

a. Inside the function it was created

b. Anywhere in the program

c. Only inside the main function

d. Only in global scope

Henkel

Quiz

What is recursion in C programming?

a. A function calling another function

b. A function calling itself

c. Repeating a loop indefinitely

d. Using a function inside another function

Henkel

Day 5 Agenda

Project Development

HOUR 1

HOUR 2

HOUR 3

Number Guessing :

- *Introduce the concept of random number generation using rand()*

Finish Project + Review

Recap all Topics

Henkel

Day 5 Practical Programs

Code Examples and Exercises for Each Hour

Hour 1 Programs

logic of the number guessing game (user guesses a number between 1 to 100)

Hour 2 Programs

Add features like: Count the number of attempts , Give hints (e.g., "Too high", "Too low")

Hour 3 Programs

Programs Quiz

Henkel

Rand ()

Computers can pretend to choose a random number using a special function. In C, we use **rand()** to ask the computer to give us a random number

Imagine a box with numbers from 1 to 100. You close your eyes and pick one — that's what the computer is doing with rand()!

Henkel

Game Rules

- The computer picks a number between 1 and 100.
- You (the player) try to guess it.
- The computer tells you if your guess is too high or too low.
- Keep guessing until you find the right number.

Henkel

THANK YOU.