**1.Research and provide three real-world applications where C programming is extensively used, such as in embedded systems, operating systems, or game development.**

1. **Embedded Systems**: C programming is widely used in microcontroller programming, controlling hardware devices like automotive systems, medical devices, and IoT gadgets due to its efficiency and low-level hardware access.
2. **Operating Systems**: C is the foundation for developing operating systems like Linux and Windows. It provides direct memory management, efficient execution, and hardware interaction, making it ideal for system-level programming.
3. **Game Development**: Many game engines and performance-critical games are built with C due to its fast execution and control over hardware resources, especially for high-performance games like those in the gaming industry.

**2.Install a C compiler on your system and configure the IDE. Write your first program to print "Hello, World!" and run it.**

To install a C compiler and write your first program, follow these steps:

1. **Install Compiler**:
   * **Windows**: Download and install **MinGW** from [MinGW](http://mingw.org/).
   * **Mac**: Install Xcode command line tools by running xcode-select --install in the terminal.
   * **Linux**: Run sudo apt install build-essential (Ubuntu/Debian) or sudo yum groupinstall "Development Tools" (CentOS/Fedora).
2. **Configure IDE** (optional):
   * Install an IDE like **Code::Blocks**, **Visual Studio**, or **Eclipse**. Alternatively, use **VS Code** with the C extension.
3. **Write "Hello, World!" Program**: Open your IDE or a text editor, and write the following code:
4. #include <stdio.h>
5. int main() {
6. printf("Hello, World!\n");
7. return 0;
8. }
9. **Compile and Run**:
   * **Command Line**: Open the terminal, navigate to the file’s directory, and run:
   * gcc hello.c -o hello
   * ./hello
   * **IDE**: Use the "Build" or "Run" option to compile and execute the program.

**3.Write a C program that includes variables, constants, and comments. Declare and use different data types (int, char, float) and display their values.**

Here’s a simple C program that demonstrates variables, constants, comments, and different data types:

#include <stdio.h>

#define PI 3.14 // Constant

int main() {

// Variable declarations

int age = 25;

char grade = 'A';

float salary = 5000.50;

// Display values

printf("Age: %d\n", age);

printf("Grade: %c\n", grade);

printf("Salary: %.2f\n", salary);

printf("PI: %.2f\n", PI); // Display constant value

return 0;

}

**4.Write a C program that accepts two integers from the user and performs arithmetic, relational, and logical operations on them. Display the results.**

Here’s a C program that accepts two integers from the user and performs arithmetic, relational, and logical operations:

#include <stdio.h>

int main() {

int num1, num2;

// Accept two integers from the user

printf("Enter first integer: ");

scanf("%d", &num1);

printf("Enter second integer: ");

scanf("%d", &num2);

// Arithmetic operations

printf("Sum: %d\n", num1 + num2);

printf("Difference: %d\n", num1 - num2);

printf("Product: %d\n", num1 \* num2);

printf("Quotient: %.2f\n", (float)num1 / num2);

// Relational operations

printf("num1 > num2: %d\n", num1 > num2);

printf("num1 < num2: %d\n", num1 < num2);

printf("num1 == num2: %d\n", num1 == num2);

// Logical operations

printf("num1 > 0 && num2 > 0: %d\n", (num1 > 0) && (num2 > 0));

printf("num1 > 0 || num2 > 0: %d\n", (num1 > 0) || (num2 > 0));

return 0;

}

**5.Write a C program to check if a number is even or odd using an if-else statement. Extend the program using a switch statement to display the month name based on the user’s input (1 for January, 2 for February, etc.).**

Here’s a C program to check if a number is even or odd using an if-else statement and display the month name using a switch statement:

#include <stdio.h>

int main() {

int num, month;

// Check if number is even or odd

printf("Enter a number: ");

scanf("%d", &num);

if (num % 2 == 0)

printf("%d is Even\n", num);

else

printf("%d is Odd\n", num);

// Display month name based on user input

printf("Enter a month number (1-12): ");

scanf("%d", &month);

switch(month) {

case 1: printf("January\n"); break;

case 2: printf("February\n"); break;

case 3: printf("March\n"); break;

case 4: printf("April\n"); break;

case 5: printf("May\n"); break;

case 6: printf("June\n"); break;

case 7: printf("July\n"); break;

case 8: printf("August\n"); break;

case 9: printf("September\n"); break;

case 10: printf("October\n"); break;

case 11: printf("November\n"); break;

case 12: printf("December\n"); break;

default: printf("Invalid month number\n"); break;

}

return 0;

}

**6.Write a C program to print numbers from 1 to 10 using all three types of loops (while, for, do-while).**

Here’s a C program that prints numbers from 1 to 10 using while, for, and do-while loops:

#include <stdio.h>

int main() {

// Using while loop

int i = 1;

printf("Using while loop:\n");

while(i <= 10) {

printf("%d ", i);

i++;

}

printf("\n");

// Using for loop

printf("Using for loop:\n");

for(int i = 1; i <= 10; i++) {

printf("%d ", i);

}

printf("\n");

// Using do-while loop

printf("Using do-while loop:\n");

int j = 1;

do {

printf("%d ", j);

j++;

} while(j <= 10);

printf("\n");

return 0;

}

**7.Write a C program that uses the break statement to stop printing numbers when it reaches 5. Modify the program to skip printing the number 3 using the continue statement.**

Here’s a C program that uses the break statement to stop printing numbers when it reaches 5 and the continue statement to skip printing the number 3:

#include <stdio.h>

int main() {

// Using break to stop at 5

printf("Using break statement:\n");

for (int i = 1; i <= 10; i++) {

if (i == 5) {

break; // Stop when i reaches 5

}

printf("%d ", i);

}

printf("\n");

// Using continue to skip number 3

printf("Using continue statement:\n");

for (int i = 1; i <= 10; i++) {

if (i == 3) {

continue; // Skip printing number 3

}

printf("%d ", i);

}

printf("\n");

return 0;

}

**8.Write a C program that calculates the factorial of a number using a function. Include function declaration, definition, and call.**

Here’s a C program that calculates the factorial of a number using a function:

#include <stdio.h>

// Function declaration

int factorial(int n);

int main() {

int num;

// Get user input

printf("Enter a number: ");

scanf("%d", &num);

// Function call

printf("Factorial of %d is %d\n", num, factorial(num));

return 0;

}

// Function definition

int factorial(int n) {

if (n == 0 || n == 1)

return 1; // Base case

else

return n \* factorial(n - 1); // Recursive case

}

**9.Write a C program that stores 5 integers in a one-dimensional array and prints them. Extend this to handle a two-dimensional array (3x3 matrix) and calculate the sum of all elements.**

Here’s a C program that stores 5 integers in a one-dimensional array, prints them, and then handles a 3x3 two-dimensional array and calculates the sum of all elements:

#include <stdio.h>

int main() {

// One-dimensional array

int arr[5] = {1, 2, 3, 4, 5};

printf("One-dimensional array elements:\n");

for(int i = 0; i < 5; i++) {

printf("%d ", arr[i]);

}

printf("\n");

// Two-dimensional array (3x3 matrix)

int matrix[3][3] = {

{1, 2, 3},

{4, 5, 6},

{7, 8, 9}

};

int sum = 0;

printf("Two-dimensional array elements:\n");

for(int i = 0; i < 3; i++) {

for(int j = 0; j < 3; j++) {

printf("%d ", matrix[i][j]);

sum += matrix[i][j]; // Calculate sum

}

printf("\n");

}

printf("Sum of all elements in the matrix: %d\n", sum);

return 0;

}

**10.Write a C program to demonstrate pointer usage. Use a pointer to modify the value of a variable and print the result.**

Here’s a C program that demonstrates pointer usage by modifying the value of a variable through a pointer:

#include <stdio.h>

int main() {

int num = 10;

int \*ptr = &num; // Pointer to num

printf("Before modification, num = %d\n", num);

// Modify the value of num using the pointer

\*ptr = 20;

printf("After modification, num = %d\n", num);

return 0;

}

**11.Write a C program that takes two strings from the user and concatenates them using strcat(). Display the concatenated string and its length using strlen().**

Here’s a C program that takes two strings from the user, concatenates them using strcat(), and displays the concatenated string and its length using strlen():

#include <stdio.h>

#include <string.h>

int main() {

char str1[100], str2[50];

// Take input from the user

printf("Enter first string: ");

fgets(str1, sizeof(str1), stdin);

str1[strcspn(str1, "\n")] = '\0'; // Remove newline character

printf("Enter second string: ");

fgets(str2, sizeof(str2), stdin);

str2[strcspn(str2, "\n")] = '\0'; // Remove newline character

// Concatenate the strings

strcat(str1, str2);

// Display the concatenated string and its length

printf("Concatenated string: %s\n", str1);

printf("Length of concatenated string: %lu\n", strlen(str1));

return 0;

}

**12.Write a C program that defines a structure to store a student's details (name, roll number, and marks). Use an array of structures to store details of 3 students and print them.**

Here’s a C program that defines a structure to store a student's details and uses an array of structures to store the details of 3 students:

#include <stdio.h>

struct Student {

char name[50];

int roll\_no;

float marks;

};

int main() {

struct Student students[3];

// Input details for 3 students

for (int i = 0; i < 3; i++) {

printf("Enter details for student %d:\n", i+1);

printf("Name: ");

fgets(students[i].name, sizeof(students[i].name), stdin);

students[i].name[strcspn(students[i].name, "\n")] = '\0'; // Remove newline

printf("Roll number: ");

scanf("%d", &students[i].roll\_no);

printf("Marks: ");

scanf("%f", &students[i].marks);

getchar(); // Clear newline left by scanf

}

// Print details of 3 students

printf("\nStudent Details:\n");

for (int i = 0; i < 3; i++) {

printf("Student %d - Name: %s, Roll No: %d, Marks: %.2f\n",

i+1, students[i].name, students[i].roll\_no, students[i].marks);

}

return 0;

}

**13.Write a C program to create a file, write a string into it, close the file, then open the file again to read and display its contents.**

Here’s a C program that creates a file, writes a string into it, closes the file, then opens it again to read and display its contents:

#include <stdio.h>

int main() {

FILE \*file;

char str[] = "Hello, this is a test file.";

// Create and write to the file

file = fopen("testfile.txt", "w");

if (file == NULL) {

printf("Error opening file for writing.\n");

return 1;

}

fprintf(file, "%s", str);

fclose(file);

// Open the file again to read its contents

file = fopen("testfile.txt", "r");

if (file == NULL) {

printf("Error opening file for reading.\n");

return 1;

}

char ch;

printf("File contents: ");

while ((ch = fgetc(file)) != EOF) {

putchar(ch);

}

printf("\n");

fclose(file);

return 0;

}