Module 2 – Introduction to Programming

Overview of C Programming:-Theory

Q1. History and importance of C programming. Why is it still used today?

Answer:

**C is a programming language made in 1972 by Dennis Ritchie at Bell Labs.
It was used to build the UNIX operating system.
C is fast, simple, and works close to the hardware.
It helps to build system software, embedded systems, and games.**

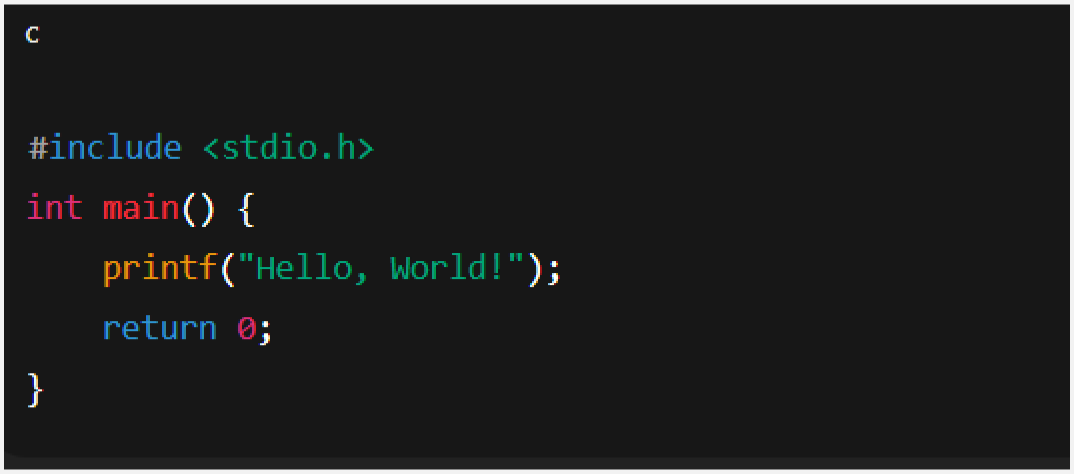
Even today, C is used because:

- **It runs fast**
- **It gives more control**
- **It helps students learn logic**
- **Many new languages like C++, Java, and Python are based on C**

Q2. Steps to install a C compiler and set up an IDE

Answer:

1. **Download a C IDE like DevC++, CodeBlocks, or VS Code.**
2. **Install it on your computer.**
3. **Open the IDE and create a new C file.**
4. **Write a simple program like:**



```c
#include <stdio.h>
int main() {
    printf("Hello, World!");
    return 0;
}
```

5. **Click on Compile or Run to check if it works.**

- **If output shows "Hello, World!", your setup is successful.**

Q3. Explain the basic structure of a C program, including headers, main function, comments, data types, and variables. Provide examples.

Answer:
A C program starts with a header file like #include <stdio.h>.
It contains a main() function.
We use comments using // or /* */.
Variables are declared with data types like int, float, char.

- **Example:**

```c
#include <stdio.h>  // Header file
int main() {
    int a = 10;      // variable declaration
    printf("Value is %d", a);  // output
    return 0;
}
```

Q4. Write notes explaining each type of operator in C: arithmetic, relational, logical, assignment, increment/decrement, bitwise, and conditional operators.

Answer:

- **Arithmetic: +, -, *, /**
  ➤ int c = a + b;
- **Relational: >, <, ==, !=**
  ➤ if (a > b)
- **Logical: &&, ||, !**
  ➤ if (a > 0 && b > 0)
- **Assignment: =, +=, -=**
  ➤ x += 5;
- **Increment/Decrement: ++, --**
  ➤ a++;
- **Bitwise: &, |, ^**
  ➤ int result = a & b;
- **Conditional (Ternary): ?:**
  ➤ int max = (a > b) ? a : b;

Q5. Explain decision-making statements in C (if, else, nested if-else, switch). Provide examples of each.

Answer:
C uses condition statements to make decisions.

1. **if:**

```c
int x = 10;
if (x > 5) {
    printf("x is greater than 5");
}
```

2.if-else:

```c
if (x % 2 == 0) {
    printf("Even");
} else {
    printf("Odd");
}
```

3.Nested if-else:

```c
if (x >= 0) {
    if (x == 0)
        printf("Zero");
    else
        printf("Positive");
} else {
    printf("Negative");
}
```

4 switch:

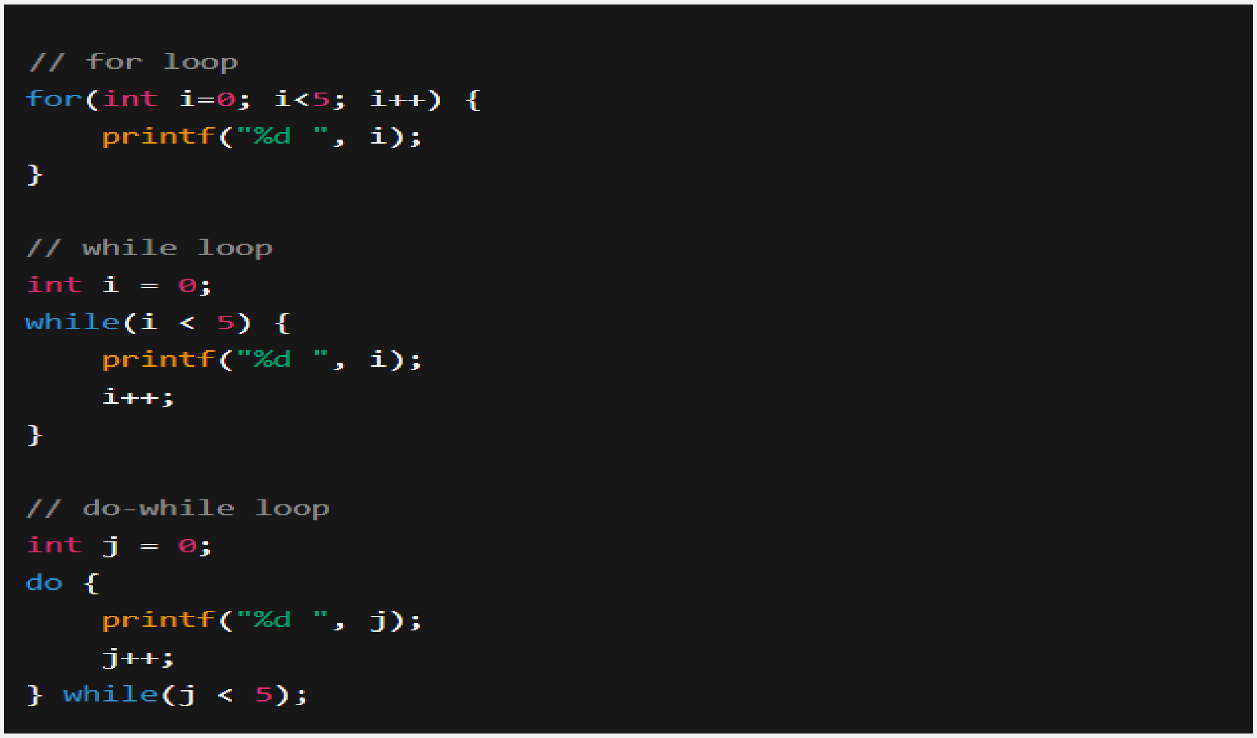
```c
int day = 2;
switch(day) {
    case 1: printf("Monday"); break;
    case 2: printf("Tuesday"); break;
    default: printf("Another day");
}
```

Q6. Compare and contrast while loops, for loops, and do-while loops. Explain the scenarios in which each loop is most appropriate.

Answer:

Loop Type	When to Use	Runs At Least Once	Example
for	Known number of times	No	for(int i=0; i<5; i++)
while	Unknown condition, check before start	No	while(i<5)
do-while	Must run at least once	Yes	do { } while(i<5);

- **Example for all:**

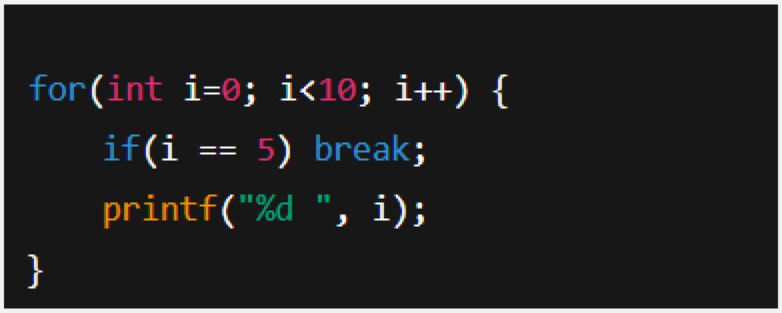


```c
// for loop
for(int i=0; i<5; i++) {
    printf("%d ", i);
}

// while loop
int i = 0;
while(i < 5) {
    printf("%d ", i);
    i++;
}

// do-while loop
int j = 0;
do {
    printf("%d ", j);
    j++;
} while(j < 5);
```

Q7. Explain loop control statements (break, continue, and goto) in C with examples.

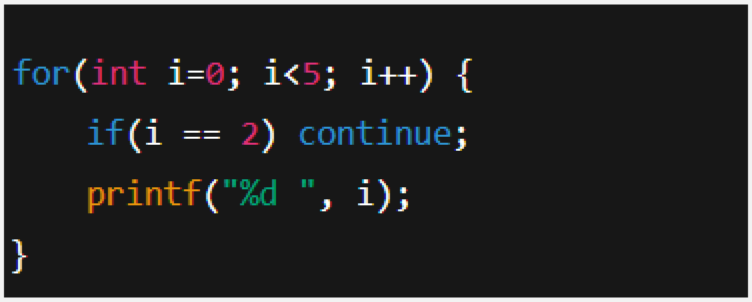
Answer:

1. **break: stops the loop**



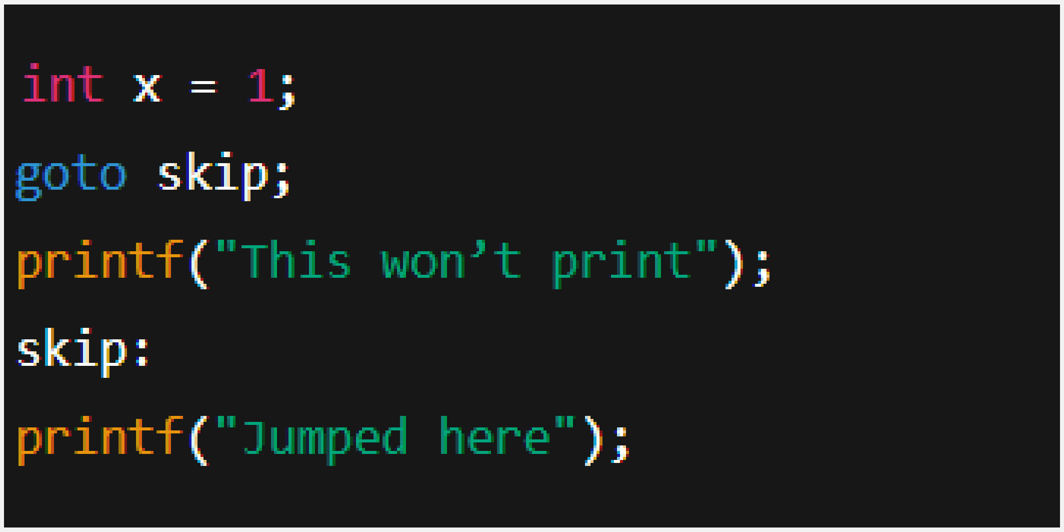
```c
for(int i=0; i<10; i++) {
    if(i == 5) break;
    printf("%d ", i);
}
```

2. **continue: skips current iteration**



```c
for(int i=0; i<5; i++) {
    if(i == 2) continue;
    printf("%d ", i);
}
```

3. **goto: jumps to a label**



```c
int x = 1;
goto skip;
printf("This won’t print");
skip:
printf("Jumped here");
```

Q8. What is a function in C? Explain the concept of function declaration, definition, and calling.

Answer:
A function is a block of code that performs a task.

- **Declaration – tells the compiler about the function.**
- **Definition – actual code inside the function.**
- **Calling – running the function.**

◆ **Example:**

```
// Declaration
int add(int, int);

// Definition
int add(int a, int b) {
    return a + b;
}

// Calling
int result = add(5, 10);
printf("Sum is %d", result);
```

Q9. What are arrays in C? Explain with examples of single and multi-dimensional arrays.

Answer:
An array stores multiple values of the same data type in one variable.

1D Array Example:

```
int numbers[5] = {1, 2, 3, 4, 5};
printf("%d", numbers[0]);
```

2D Array Example:

```
int matrix[2][2] = {{1, 2}, {3, 4}};
printf("%d", matrix[0][1]);  // Output: 2
```

Q10. Explain pointers in C and their importance. Provide a simple example.

Answer:
A pointer stores the address of another variable.
They are used in dynamic memory, arrays, functions, etc.

- **Example:**

```
int a = 10;
int *p = &a;
printf("Value = %d", *p);   // Output: 10
```

Q11. Describe string handling in C using common string functions.

Answer:
Strings are arrays of characters ending with \0.

◆ **Common string functions:**

- **strlen() – returns length**
- **strcpy() – copies one string into another**
- **strcat() – joins two strings**
- **strcmp() – compares two strings**

◆ **Example:**

```
char str1[20] = "Hello";
char str2[20] = "World";

strcat(str1, str2);
printf("%s", str1);  // Output: HelloWorld
```

Q12. What is a structure in C? How does it differ from an array?

Answer:
A structure is used to group different data types.

📌 **Difference from Array:**

Array	Structure
Same data type	Different data types
int marks[5];	struct Student { int id; char name[20]; };

◆ **Example:**

```c
struct Student {
    int id;
    char name[20];
};

struct Student s1 = {1, "Jay"};
printf("%d %s", s1.id, s1.name);
```

Q13. Explain file handling in C. Describe how to open, read, write, and close a file.

Answer:
In C, we use FILE type and functions to work with files.

🔹 **Common Functions:**

- **fopen() – open file**
- **fprintf() / fputs() – write**
- **fscanf() / fgets() – read**
- **fclose() – close file**

🔹 **Example:**

```c
FILE *fp;
fp = fopen("data.txt", "w");
fprintf(fp, "Hello File!");
fclose(fp);
```