C Programming Basics - Summarization

1. Declaring and Initializing Variables in C

Variables in C are used to store data. Before using a variable, it must be declared with a specific data type.

Basic Data Types in C:

- **int**: Stores integers (e.g., 10, -5, 1000).
- **float**: Stores decimal numbers (e.g., 3.14, -2.5).
- **double**: Stores double-precision floating-point numbers.
- **char**: Stores a single character (e.g., 'A', 'b').

Declaration & Initialization:

```
int age = 20;

float pi = 3.14;

char grade = 'A';
```

2. Performing Arithmetic Operations in C

C supports various arithmetic operations:

- **Addition (+)**
- **Subtraction (-)**
- **Multiplication (*)**
- **Division (/)**
- **Modulus (%)** (returns remainder of division, used for integers)

3. Conditional Statements in C

Conditional statements allow decision-making in programs.

if-else Statement:

```
int num = 10;

if (num > 0) {
    printf("Positive number");
```

```
} else {
    printf("Non-positive number");
}
```

else-if Ladder:

```
int score = 85;
if (score >= 90) {
    printf("Grade: A");
} else if (score >= 80) {
    printf("Grade: B");
} else {
    printf("Grade: C");
}
```

switch-case Statement:

Used when there are multiple conditions based on a single value.

```
char grade = 'B';
switch (grade) {
    case 'A': printf("Excellent"); break;
    case 'B': printf("Good"); break;
    case 'C': printf("Average"); break;
    default: printf("Invalid grade");
}
```

4. Iterating with Loops in C

Loops allow repetitive execution of code.

for Loop:

```
for (int i = 1; i <= 5; i++) {
    printf("%d ", i);
```

```
}
```

while Loop:

```
int i = 1;
while (i <= 5) {
    printf("%d ", i);
    i++;
}
```

do-while Loop:

Executes at least once before checking the condition.

```
int i = 1;
do {
    printf("%d ", i);
    i++;
} while (i <= 5);
```

5. Using Constants in C

Constants are fixed values that do not change during program execution.

Using #define (Preprocessor Directive):

```
#define PI 3.14159
printf("Value of PI: %f", PI);
```

Using const Keyword:

```
const int MAX = 100;
printf("Max value: %d", MAX);
```

6. Error Handling in C

To make programs robust, error handling is essential.

Checking for Input Errors:

```
int num;
```

```
printf("Enter a number: ");
if (scanf("%d", &num) != 1) {
    printf("Invalid input!");
}
```

Handling Division by Zero:

```
int a = 10, b = 0;
if (b != 0) {
    printf("Result: %d", a / b);
} else {
    printf("Error: Division by zero!");
}
```