# Module-01

**Subject: C Programming** **Sem: I**

## 1.1 Character Set, Identifiers, Keywords, Data Types, Constants, Variables

### Character Set

- Letters: A–Z, a–z
- Digits: 0–9
- Special characters: + - * / = ~ ! @ # $ % ^ & * ( ) _ { } [ ] ; : ' " < > , . ? \ |
- White spaces: space, tab, newline

### Identifiers

- Names used for variables, functions, arrays, etc.
- Rules:
  - Start with a letter or underscore _
  - Followed by letters, digits, or underscores
  - Case-sensitive
  - Cannot use keywords

### Keywords

- Reserved words in C (cannot be used as identifiers). E.g.:
- **int, float, return, if, else, for, while, break, continue, void, char, long, double, short, sizeof, struct**

### Data Types

| Type | Description | Example |
|---|---|---|
| **int** | **Integer** | **int a = 10;** |
| **float** | **Floating-point** | **float b = 5.2;** |
| **char** | **Character** | **char c = 'A';** |
| **double** | **Double-precision float** | **double d = 3.14;** |
| **void** | **No value** | **void func();** |

### Constants

- Literal constants: 10, 3.14, 'A', "Hello"
- #define constants:
  - #define PI 3.14
- const keyword:
  - const int x = 100;

### Variables

- Containers for storing data values.
- Must be declared before use.

```
int age;
float price;
char grade.
```

## 1.2 Operators & Expressions

### Arithmetic Operators

| Operator | Description | Example |
|---|---|---|
| + | Addition | a + b |
| - | Subtraction | a - b |
| * | Multiplication | a * b |
| / | Division | a / b |
| % | Modulus | a % b |

### Relational Operators

| Operator | Description | Example |
|---|---|---|
| == | Equal to | a == b |
| != | Not equal to | a != b |
| > | Greater than | a > b |
| < | Less than | a < b |
| >= | Greater or equal | a >= b |
| <= | Less or equal | a <= b |

### Logical Operators

| Operator | Description | Example |
|---|---|---|
| && | Logical AND | (a > 5 && b < 10) |
| \|\| | Logical OR | (a > 5 \|\| b < 10) |
| ! | Logical NOT | !a |

### Assignment Operators

= += -= *= /= %=

### Unary Operators

++a, --a, -a, +a, !a

### Conditional Operator

(condition) ? true_expr : false_expr;

### Bitwise Operators

& | ^ ~ << >>

### Comma Operator

int a = (1, 2, 3); // a = 3

### Other Operators

- sizeof(): Returns size in bytes
- &: Address of variable
- *: Pointer dereference
- ->: Access structure member via pointer

### Expressions

- Combination of variables, constants, operators
- int c = a + b * 2;

### Statements

- Instructions that perform actions

```
a = 5;
printf("Hello");
```

### Library Functions

- Built-in functions in C standard library
- printf(), scanf(), strcpy(), strlen(), pow()

### Preprocessor

- Instructions that begin with #
- #include <stdio.h>  // Includes header file
- #define PI 3.14  // Macro definition

## 🔹 1.3 Data Input and Output

### ✅ Character I/O

```
char ch;
ch = getchar();     // input one character
putchar(ch);        // output one character
```

### ✅ String I/O

```
char str[50];
gets(str);          // input string (unsafe)
puts(str);          // output string
```

### ✅ Formatted I/O

```
int a;
float b;
char c;

scanf("%d %f %c", &a, &b, &c);  // input
printf("%d %.2f %c", a, b, c);  // output
```

### 🏵️ Structure of a C Program

```
#include <stdio.h>      // Preprocessor Directive

int main() {            // Main Function
   int a = 10;          // Variable Declaration
   printf("%d", a);     // Output
   return 0;            // Return Statement
}
```

### 📝 Quick Tips

- ✓ Every statement ends with a semicolon ;
- ✓ main() is the entry point of a C program
- ✓ Use #include to include header files like stdio.h, math.h, etc.
- ✓ Always initialize variables before using them

### ✅ Comments

| Type | Syntax | Description |
|---|---|---|
| **Single-line** | // This is a comment | For short, inline explanations |
| **Multi-line** | /* This is a multi-line comment */ | For longer descriptions or blocks |

### ✅ Escape Sequences

| Escape Code | Description | Example in Code | Output |
|---|---|---|---|
| **\n** | New line | printf("Line 1\\nLine 2"); | Line 1Line 2 |
| **\t** | Horizontal tab | printf("A\\tB"); | A B |
| **** | Backslash | printf("C:\\\\Path"); | C:\Path |
| **\'** | Single quote | printf("\\'Hello\\'"); | 'Hello' |
| **\"** | Double quote | printf("\\\"Hi\\\""); | "Hi" |
| **\r** | Carriage return | printf("Hello\\rWorld"); | World |
| **\b** | Backspace | printf("AB\\bC"); | AC |
| **\a** | Alert (beep sound) | printf("\\a"); | (Makes a beep sound) |
| **\f** | Form feed | Rarely used | (Page break effect) |
| **\v** | Vertical tab | Rarely used | (Vertical space) |
| **\0** | Null character (end of string) | Used in strings | N/A (invisible) |

## Format Specifiers

| Specifier | Type | Description | Example Value |
|---|---|---|---|
| %d | int | Signed decimal integer | 10, -25 |
| %u | unsigned int | Unsigned decimal integer | 25 |
| %f | float/double | Decimal floating-point | 3.14, -0.5 |
| %.nf | float/double | Floating-point with n decimals | %.2f → 3.14 |
| %c | char | Single character | 'A' |
| %s | string | Null-terminated character array | "Hello" |
| %ld | long int | Long signed integer | 1234567890 |
| %lu | unsigned long | Unsigned long integer | 4000000000 |
| %lf | double | Double precision float | 3.1415926 |
| %p | pointer address | Memory address | 0x7ffe... |
| %% | literal % | Prints a percent sign | % |

## Practice Questions:

- WAP in C to Calculate area of rectangle. **(Hint: area = length * width;)**
- WAP in C to Calculate area of Circle. **(Hint: area = PI * radius * radius;)**
- WAP in C to Calculate volume of a cylinder. **(Hint: volume = PI * radius * radius * height;)**
- WAP in C to Calculate square of a number. **(Hint: square = number * number;)**
- WAP in C to Calculate average of three numbers. **(Hint: average = (num1 + num2 + num3) / 3;)**
- WAP in C to accept number from user and find remainder after dividing it by 2 and 3.
  **(Hint: remainder2 = number % 2;**
  **remainder3 = number % 3;)**
- WAP in C to accept two digit number from user and display it in reverse order.
  **(Hint: int tens = number / 10;**
  **int ones = number % 10;**
  **reversed = ones * 10 + tens;)**
- WAP in C to accept float number and display integer part using type casting operator.
  **(Hint: scanf("%f", &num);**
  **intPart = (int)num;)**
- WAP in C to accept number and display equivalent ASCII using type casting.
  **(Hint: int num;**
  **char asciiChar;**
  **scanf("%d", &num);**
  **asciiChar = (char)num;)**
- Find output:

```
#include <stdio.h>

int main() {
    int age = 25;            // Variable
    const float pi = 3.14;   // Constant
    char grade = 'A';        // Character variable

    printf("Age: %d\n", age);
    printf("Pi: %.2f\n", pi);
    printf("Grade: %c\n", grade);

    return 0;
}
```

- Find output:

```
#include <stdio.h>

int main() {
   int a = 5, b = 3;
   int sum = a + b;
   int rel = (a > b);          // Relational
   int logical = (a > 0 && b > 0); // Logical
   int conditional = (a > b) ? a : b; // Conditional

   printf("Sum: %d\n", sum);
   printf("Is a > b? %d\n", rel);
   printf("Logical AND result: %d\n", logical);
   printf("Conditional (max): %d\n", conditional);

   return 0;
}
```

- Find output:

```
#include <stdio.h>

int main() {
   int a = 5, b = 8;
   printf("a == b: %d\n", a == b);
   printf("a != b: %d\n", a != b);
   printf("a < b: %d\n", a < b);
   printf("a > b: %d\n", a > b);
   printf("a <= b: %d\n", a <= b);
   printf("a >= b: %d\n", a >= b);
   return 0;
}
```

- Find output:

```
#include <stdio.h>

int main() {
   int a = 5, b = 0;
   printf("(a > 0 && b > 0) = %d\n", (a > 0 && b > 0));
   printf("(a > 0 || b > 0) = %d\n", (a > 0 || b > 0));
   printf("!(a > b) = %d\n", !(a > b));
   return 0;
}
```

- Find output:

```
#include <stdio.h>

int main() {
   int a = 5;
   printf("a = %d\n", a);
   printf("++a = %d\n", ++a);  // pre-increment
   printf("a++ = %d\n", a++);  // post-increment
   printf("a after post-increment = %d\n", a);
   return 0;
}
```

- Find output:

```
#include <stdio.h>

int main() {
   unsigned int a = 5, b = 9;
   printf("a & b = %d\n", a & b);
   printf("a | b = %d\n", a | b);
   printf("a ^ b = %d\n", a ^ b);
   printf("~a = %d\n", ~a);
   printf("a << 1 = %d\n", a << 1);
   printf("b >> 1 = %d\n", b >> 1);
   return 0;
}
```

- Find output:

```
#include <stdio.h>

int main() {
   int a = (printf("Hello, "), 10 + 20);
   printf("\na = %d\n", a);
   return 0;
}
```