Unit 1

1. **1.Give brief history of C and explain it's features.**

The C programming language was developed by Dennis Ritchie in 1972 at AT&T Bell Laboratories. It was originally designed for developing system software, especially the UNIX operating system. C language evolved from earlier languages such as BCPL and B. Because of its efficiency, flexibility, and powerful features, C quickly became popular and is widely used for developing operating systems, compilers, embedded systems, and application software.

Features of C language:

1. Simple and Efficient: C has a simple syntax and allows programmers to write efficient programs that execute faster.
2. Structured Language: C supports structured programming, which means a program can be divided into functions, making it easier to understand and maintain.
3. Portable: Programs written in C can be run on different machines with little or no modification.
4. Rich Library Functions: C provides a large set of built-in library functions that help in performing various tasks easily.
5. Pointer Support: C supports pointers, which allow direct access to memory and help in efficient memory management.
6. Low-level Programming: C provides features to interact with hardware and memory, making it suitable for system programming.

Because of these features, C is considered a powerful and widely used programming language.

2.Explain Arithmetic operators with example.

Arithmetic operators in C language are used to perform basic mathematical calculations on numeric values. These operators are commonly used in expressions to carry out operations such as addition, subtraction, multiplication, division, and finding the remainder.

The arithmetic operators in C are:

7. Addition (+): Used to add two operands.
8. Subtraction (−): Used to subtract one operand from another.
9. Multiplication (*): Used to multiply two operands.
10. Division (/): Used to divide one operand by another and return the quotient.
11. Modulus (%): Used to find the remainder when one integer is divided by another.

Arithmetic operators can be used with variables, constants, or expressions and are essential for performing calculations in C programs.

Example:

```
#include<stdio.h>
int main()
{
    int a = 10, b = 3;
    printf("Addition = %d\n", a + b);
printf("Subtraction = %d\n", a - b);
printf("Multiplication = %d\n", a * b);
printf("Division = %d\n", a / b);
printf("Modulus = %d\n", a % b);
    return 0;
}
```

In the above example, different arithmetic operators are used to perform calculations on two integer variables.

3.Explain structure of C program.

A C program follows a well-defined structure that helps in writing, understanding, and maintaining the program easily. Each part of the structure has a specific purpose. The general structure of a C program is as follows:

1. Documentation Section:

This section contains comments that describe the purpose of the program, author name, and other details. It improves readability but does not affect program execution.

2. Link Section:

The link section includes header files using the #include directive. These header files contain predefined functions that are used in the program, such as input and output functions.

3. Definition Section:

In this section, symbolic constants are defined using the #define directive. These constants remain fixed throughout the program.

4. Global Declaration Section:

This section declares global variables and function prototypes that can be accessed by all functions in the program.

5. main() Function:

The main() function is the starting point of program execution. It contains executable statements, declarations, and control statements.

6. User-defined Functions:

These functions are written by the programmer to perform specific tasks. They help in breaking a large program into smaller, manageable parts.

Thus, the structured format of a C program makes it easy to develop, debug, and maintain programs efficiently.

4.What is a variable? Explain the rules of declaring a variable.

A variable is a named memory location used to store data during the execution of a program. The value stored in a variable can change while the program is running. Variables help programmers store, retrieve, and manipulate data in a program. Each variable must be declared with a specific data type before it is used so that the compiler knows what type of data it can store.

Rules for declaring a variable in C language:

1. A variable name must begin with a letter (a–z or A–Z) or an underscore (_).
2. It cannot start with a digit.
3. Variable names may contain letters, digits, and underscores only.
4. No special symbols such as %, $, @, or spaces are allowed.
5. Keywords of the C language cannot be used as variable names.
6. Variable names are case-sensitive, meaning Total and total are different.
7. A variable must be declared before it is used in the program.

Thus, variables play an important role in storing and manipulating data in C programs, and following proper naming rules helps avoid errors.

5.Explain Logical and Relational operators with example

In the C programming language, relational and logical operators are mainly used in decisionmaking and conditional statements. They help in comparing values and combining multiple conditions.

Relational Operators

Relational operators are used to compare two values. The result of a relational operation is either true (1) or false (0).

The relational operators are:

- Less than (<) – checks if one value is less than another
- Greater than (>) – checks if one value is greater than another
- Less than or equal to (<=) – checks if one value is less than or equal to another
- Greater than or equal to (>=) – checks if one value is greater than or equal to another
- Equal to (==) – checks whether two values are equal
- Not equal to (!=) – checks whether two values are not equal

These operators are commonly used in conditions such as if, while, and for statements. Logical Operators

Logical operators are used to combine two or more conditions and return a true or false result.

The logical operators are:

- Logical AND (&&): Returns true if all conditions are true
- Logical OR (||): Returns true if any one condition is true
- Logical NOT (!): Reverses the result of a condition

Logical operators are mainly used when multiple conditions need to be checked at the same time.

6.Explain basic datatypes and enumerated datatypes.

In C language, data types are used to specify the type of data that a variable can store. They help the compiler understand how much memory to allocate and what type of operations can be performed on the data.

Basic Data Types in C

Basic data types are the fundamental data types provided by the C language. They are used to store simple values.

1. int:

This data type is used to store whole numbers without decimal points. It generally occupies 2 or 4 bytes of memory depending on the system.

2. float:

The float data type is used to store decimal or fractional numbers. It provides single-precision floating-point values.

3. double:

The double data type is used to store large decimal numbers with double precision, providing more accuracy than float.

4. char:

The char data type is used to store a single character, such as a letter, digit, or symbol. It occupies 1 byte of memory.

Basic data types are widely used for storing numerical and character data in C programs.

Enumerated Data Type (enum)

The enumerated data type, also known as enum, is a user-defined data type that consists of a set of named integer constants. Each name in an enumeration represents an integer value, starting from 0 by default.

The enumerated data type improves program readability and reduces errors by using meaningful names instead of numeric values. It is mainly used when a variable can have a fixed set of possible values.

Unit II

1. Explain formatted printf() and scanf() functions with example.

In C language, printf() and scanf() are formatted input and output functions. They are defined in the stdio.h header file and use format specifiers to specify the type of data being printed or read.

printf() Function

The printf() function is used to display output on the screen. It prints data in a formatted manner using format specifiers. These specifiers tell the compiler how the data should be displayed.

Common format specifiers used with printf() are:

- %d for integer
- %f for floating-point number
- %c for character
- %s for string

The printf() function can display text, variables, and expressions.

scanf() Function

The scanf() function is used to accept input from the user through the keyboard. It also uses format specifiers to specify the type of data to be entered. While using scanf(), the address of the variable must be provided using the ampersand (&) operator.

Common format specifiers used with scanf() are:

- %d for integer input
- %f for float input
- %c for character input
- %s for string input Example:

```
#include<stdio.h>
int main()
{ int age;
float salary;

printf("Enter age: "); scanf("%d",
&age); printf("Enter salary: ");
scanf("%f", &salary);
printf("Age = %d\n", age);
printf("Salary = %.2f", salary);
return 0; }
```

2.the following putch() and puts() functions with example.

In C language, putch() and puts() are output functions used to display data on the screen. These functions are mainly used for character and string output.

putch() Function

The putch() function is used to display a single character on the screen. As soon as the character is printed, the cursor remains on the same line. This function is commonly used in character-based output.

Features of putch():

- Displays only one character
- Does not move the cursor to the next line • Mainly used for simple character output

puts() Function

The puts() function is used to display a string on the screen. After printing the string, it automatically moves the cursor to the next line. It is commonly used for printing messages or text.

Features of puts():

- Displays a string of characters
- Automatically adds a new line at the end
- Easy to use for text output Example:

```
#include<stdio.h>
#include<conio.h>
int main()
{
   putch('A');
   puts(" Welcome to C Programming");
return 0;
}
```

3.Differentiate between while and do…while loop

Both while and do–while loops are used for repetitive execution of statements in C programming. However, they differ in the way the condition is checked and how the loop executes.

Differences between while and do–while loop

1. Type of Loop:

The while loop is an *entry-controlled loop*, whereas the do–while loop is an *exitcontrolled loop*.

2. Condition Checking:

In a while loop, the condition is checked before executing the loop body.

In a do–while loop, the condition is checked after executing the loop body.

3. Minimum Execution:

The while loop may execute zero or more times if the condition is false initially.

The do–while loop executes at least once, even if the condition is false.

4. Syntax Difference:

The while loop does not end with a semicolon.

The do–while loop ends with a semicolon after the condition.

5. Use Case:

The while loop is used when the number of iterations is not known and execution depends on the condition.

The do–while loop is used when the loop body must execute at least once.

6. Example Situation:

Input validation is better handled using a do–while loop, while condition-based repetition is suitable for a while loop.

4.Explain getche() and putche() functions with example

In C language, getche() and putche() are character input and output functions. These functions are mainly used for handling single characters and are defined in the conio.h header file.

getche() Function

The getche() function is used to read a single character from the keyboard and immediately display (echo) the character on the screen. It does not require the Enter key to be pressed.

Features of getche():

- Reads only one character
- Automatically displays the entered character
- Does not wait for the Enter key
- Commonly used in character-based input

putche() Function

The putche() function is used to display a single character on the screen. It outputs the character and keeps the cursor on the same line.

Features of putche():

- Displays one character
- Does not move the cursor to the next line
- Used for character output Example:

#include<stdio.h> #include<conio.h>

```
int main()
{    char ch;
   printf("Enter a character: ");    ch =
getche();

   printf("\nYou entered: ");    putche(ch);

   return 0;
}
```

5.Discuss syntax of switch statement with purpose and suitable example.

The switch statement in C is a multi-way decision-making statement. It is used to execute one block of statements from multiple choices based on the value of an expression. The switch statement provides a clearer and more efficient alternative to multiple if–else statements when there are many conditions to check.

Purpose of switch statement

The main purpose of the switch statement is to:

- Simplify complex decision-making logic
- Improve readability of the program
- Execute different code blocks based on a single variable or expression
- Provide faster execution compared to multiple if–else statements in certain cases

Syntax of switch statement

```
switch(expression)
{    case constant1:
statements;
break;

   case constant2:
statements;       break;

   case constant3:
statements;       break;

   default:
      statements;
}
```

Explanation of syntax

- The expression inside the switch is evaluated once.
- The value of the expression is compared with the values of each case label.
- When a matching case is found, the corresponding statements are executed.
- The break statement is used to terminate the switch block and prevent execution of the next case.
- The default case is executed if none of the case values match the expression. Example:

```
#include<stdio.h>
int main()
```

```
{
   int choice;
   printf("Enter a number between 1 and 3: ");    scanf("%d",
&choice);

   switch(choice)
   {       case 1:           printf("You selected
option 1");           break;       case 2:
printf("You selected option 2");
break;       case 3:           printf("You
selected option 3");           break;
default:
         printf("Invalid choice");
   }
   return 0;
}
```

6.Explain if-else statement with suitable example.

The if–else statement in C is a decision-making statement used to execute different blocks of code based on whether a given condition is true or false. It helps the program take decisions and perform actions accordingly.

Purpose of if–else statement

The if–else statement is used to:

- Compare conditions
- Control the flow of program execution
- Execute one block of code when a condition is true and another when it is false

Working of if–else statement

- First, the condition inside the if statement is evaluated.
- If the condition is true, the statements inside the if block are executed.
- If the condition is false, the statements inside the else block are executed. Syntax

of if–else statement

```
if(condition)
{
   statements;
} else
{
   statements;
}
```

Suitable Example:

```
#include<stdio.h>
int main()
{
   int marks;
   printf("Enter marks: ");     scanf("%d",
&marks);

   if(marks >= 40)
printf("Result: Pass");     else
      printf("Result: Fail");

   return 0; }
```