**Module 2**

# Name: kanishk Sharma

**1. Write an essay covering the history and evolution of C programming. Explain its importance and why it is still used today.**

* History And Evolution
* C is programming language.
* It is created by Dennis Ritchie at the Bell Laboratories in 1972.
* C is strongly associated with UNIX.
* The main reason for its popularity is because it is a fundamental language in the field of computer science.
* it can be used in both applications and technologies

* Importance of C programming
  + C set of built-in unction.
  + Its access to low-level system resource.
  + C has become popular for embedded system.
  + C is highly portable

* Why It is Still Used
  + It is still used to its efficiency, portability and ability to connect directly with hardware.
  + It is essential in system level programming and performance of critical application.

**2. Describe the steps to install a C compiler (e.g., GCC) and set up an Integrated Development Environment (IDE) like DevC++, VS Code, or CodeBlocks**.

# The Steps To Install C In GCC

1. Download MinGW
   * Go to the [MinGW website.](http://www.mingw.org/)  Download the mingw-get-setup.exe
2. Install MinGW
   * Run the downloaded mingw-get-setup.exe
   * During installation, select the **mingw32-gcc-g++** and **mingw32-gcc** packages for C and C++ compilers.
3. Add MinGW to System PATH
   * After installation, navigate to **Control Panel** > **System and Security** > **System** > **Advanced system settings**.
   * Click on **Environment Variables**
   * In the **System Variables** section, find and select **Path** and click **Edit**.
   * Add the path to the MinGW bin directory, typically located at C:\MinGW\bin

# The Steps To Install VS Code

1. Download VS Code
   * Go to the [VS Code website](https://code.visualstudio.com/Download) and download the installer for

Windows

1. Install VS Code instructions  Run the downloaded installer and follow the installation
2. Install C/C++ Extensions
   * Open VS Code, and go to the Extensions view by clicking window.

the square icon in the Activity Bar on the side of the

* + Search for **C/C++** and install the extension by Microsoft.

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

# Structure Of A C Program

[“//” single comment & “/\* \*/” double comment]

#include<stdio.h>//header

Int main ()//main function

{

(Data Types: - Integer, Float, Character)

Int a; //int is data type & a is variable

Return 0;

}

# Example

#include<stdio.h>

Int main ()

{

Int roll\_no = 22;

Float CGPA = 7.63;

Char Grade = ‘A’;

Printf (“roll\_no:\t%d”, roll\_no);

Printf (“\n CGPA:\t%f”, CGPA);

Printf (“\n Grade:\t%c”, Grade);

Return 0;

}

Output: -- roll no: 22

CGPA:

Grade: A

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

1. Arithmetic operators

|  |  |  |
| --- | --- | --- |
| **Operator** | **Name of the Operator** | **Arithmetic Operator** |
| **+** | Addition | Add two operands |
| **-** | Subtraction | Subtract the second operand from the first operand |
| **\*** | Multiplication | Multiply two operands |
| **/** | Division | Divide the first operand by the second operand |
| **%** | Modulus | Calculate the remainder when the first operand is divided by the second operand |

1. Relational operators

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| **==** | Equal to | 5 == 5(True) |
| **!=** | Not equal to | 5 != 3(True) |
| **>** | Greater than | 5 > 3(True) |
| **<** | Less than | 3 < 5(True) |
| **>=** | Greater than or equal to | 5 >= 3(True) |
| **<=** | Less than or equal to | 3 <= 5(True) |

1. Logical operators

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| **&&** | and | (x < 10 && y > 1)is true |
| **||** | or | (x == 5 || y == 5)is false |
| **!** | not | !(x == y)is true |

1. Assignment operators

|  |  |  |
| --- | --- | --- |
| **Operator** | **Meaning** | **Example** |
| **=** | Simple assignment | X = 10 |
| **+=** | Add and assign | X += 5 |
| **-=** | Subtract and assign | X -= 3 |
| **\*=** | Multiply and assign | X \*= 4 |
| **/=** | Divide and assign | X /= 2 |
| **//=** | Floor divide and assign | X //= 3 |
| **%=** | Modulus and assign | X %=3 |
| **\*\*=** | Exponentiation and assign | X \*\*= 2 |
| **&=** | Bitwise AND and assign | X &= 5 |
| **`** | =` | Bitwise OR and assign |
| **^=** | Bitwise XOR and assign | X ^= 2 |
| **<<=** | Bitwise left shift and assign | X <<= 1 |
| **>>=** | Bitwise right shift and assign | X >>= 1 |

1. Increment/Decrement operators

|  |  |  |  |
| --- | --- | --- | --- |
| **Operator** | **Operation** | **Effect** | **Example** |
| **++x** | Pre-increment | Increases value before using it | Int x = 5;  Int y = ++x; |
| **X++** | Post-increment | Increases value after using it | Int x = 5;  Int y = x++; |
| **--x** | Pre-increment | Decreases value before using it | Int x = 5;  Int y = --x; |
| **x--** | Post-increment | Decreases value after using it | Int x = 5;  Int y = x--; |

1. Bitwise operators

|  |  |  |  |
| --- | --- | --- | --- |
| **Operator** | **Name** | **Description** | **Example** |
| **&** | AND | Performs bitwise AND between two bits | 5 & 3 |
| **`** | ` | OR | Performs bitwise OR between two bits |
| **^** | XOR | Performs bitwise XOR between two bits | 5 ^ 3 |
| **~** | NOT | Performs bitwise NOT(inverts  the bits) | ~5 |
| **<<** | Left Shift | Shifts bits to the left by the specified number | 5 << 1 |
| **>>** | Right Shift | Shifts bits to the right by the specified number | 5 >> 1 |

1. Conditional operators

|  |  |  |
| --- | --- | --- |
| **Condition** | **Result(True)** | **Result(False)** |
| **True** | value\_if\_true | N/A |
| **False** | N/A | Value\_if\_false |

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

* Explain decision-making statements in C
  + The decision-making statement also known as Conditional statement.
  + The decision-making statements in programming languages decide the direction of the flow of program execution.

* IF Statements

**#**include<stdio.h> Int main ()

{

If ( 22 > 19 )

{

Printf (“ 20 is greater than 19 ”);

}

Return 0;

}

* IF…ELSE Statements

**#**include<stdio.h>

Int main ()

{ int time = 22; if (time < 19)

{

Printf ("Good day.");

} else

{

Printf ("Good evening.");

}

Return 0;

}

* Nested IF…ELSE Statement

#include<stdio.h>

Int main ()

{

If (22>19)

{

Printf ("20 is Biggest Number.");

}

Else if (20==20)

{

Printf (20 & 20 is The Equal Number.");

}

Else (20<10

{

Printf ("10 is The Lowest Number.");

}

Return 0;

}

* Switch Case Statement

#include<stdio.h>

Int main ()

{

Switch (age > 19)

{

Case 1:

Printf ("\you Are Eligible for Voting"); Break; Case 0:

Printf ("\you Are Not Eligible For Voting");

Break;

}

Return 0;

}

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

# For Loop

* It is an Entry controlled loop
* It is a set of instruction to get executed for a specific number of iterations.

#  Syntax

For (initialization; test condition; Update Expression) {

//Code to be executed;

}

#  Example

int i;

for (i = 0; i < 10; i++)

{

printf("%d\n", i);

}

# While Loop

* It is an Entry controlled loop.
* While loop used, when we don’t know the number of interaction.
* It is repeatedly series of instruction till a condition is true.
* Syntax

While (test condition)

{

//code to be executed

}

* Example int i;

while ( i <= 3)

{

printf("Hello”);

i++;

}

# Do-While Loop

* It is an Exit controlled loop.
* It’s give the output at least once before the checking the condition.
* Syntax

Do

{

//code Tobe executed

}

While (condition)

* Example int i; do { printf("Hello”);

i++; }

while ( i <= 3)

1. **Explain the use of break, continue, and goto statements in C. Provide examples of each.**

* Break Statement
  + - Break statement is used to exit from a loop or a switch case statement.
    - Generally, when logical condition becomes false we can exit from the loop but if we want to exit the loop before the logical condition becomes false then use break statement.
    - Example

* Continue Statement

 Continue statement is used to move the control to the next repetition of the loop.

* Goto Statement
  + - * Goto statement is used for unconditional jumping.
      * The “goto” statement repeat a block by returning to a previously labelled block.

1. **What are functions in C? Explain function declaration, definition, and how to call a function. Provide examples.**

* What Are The Function In C.
  + Function is a block of code that performs a specific task.
  + You can pass the data known as parameters into a function.
  + It’s define the code once and use it many times.
* A Function Typically Has Three Key Components

i. Function Declaration

* + - * + A function declaration tell the complier about a function’s name, return type and parameters (argument). ii. Function Definition
        + The function definition provide the actual implementation of the function.
        + This includes the return type, the function name, the parameters, and the body of the function. iii. Function Calling
        + A Function call instruct to the compiler to execute the function.
* There are 4 types of function

I. With Return Type With Argument

Example:-

#include<stdio.h>

Int add (int a , int b);

Int main ()

{

Int ans = add (20, 10); Printf (“Addition: %d”, ans);

Return 0;

}

Int add (int a , int b)

{

Int ans = a + b;

Return ans;

|  |  |
| --- | --- |
| II.  III. | }    With Return Type Without Argument  Example:-  #include <stdio.h>  int sub(); int main()  {  int ans = sub();  printf("Subtraction: %d", ans); return 0;  }  int sub()  {  int a, b;  printf("Enter The Value Of A:\t"); scanf("%d", &a);  printf("\nEnter The Value Of B:\t"); scanf("%d", &b); int ans = a - b; return ans;  }    Without Return Type With Argument  Example:-  #include <stdio.h> void mul(int a, int b); int main()  {  int a, b;  printf("Enter The Value Of A:\t"); scanf("%d", &a);  printf("\nEnter The Value Of B:\t"); scanf("%d", &b); mul(a,b);  13 |

return 0; }

void mul(int a, int b)

{

int ans = a \* b;

printf("Multiplication:\t%d",ans);

}

IV. Without Return Type Without Argument

Example:- #include <stdio.h> void div(); int main() { div(); return 0; } void div() { int a, b;

printf("Enter The Value Of A:\t"); scanf("%d", &a);

printf("\nEnter The Value Of B:\t"); scanf("%d", &b); int ans = a / b;

printf("Division:\t%d",ans);

}

1. **Explain the concept of arrays in C. Differentiate between one-dimensional and multi-dimensional arrays with examples.**

|  |  |
| --- | --- |
| **One Dimension Array** | **Multi Dimension Array** |
| Store a single list of the element of a similar data type | Store a ‘lists’ of the element of a similar data type |
| Represent multiple data items as a list | Represent multiple data items as a table consisting of rows and columns |
| Syntax:-  elements\_type array\_name  [array\_size] = {value1, value2, …}; | Syntax:-  elements\_type array\_name  [array\_size1] [array\_size2] = {{v1, v2  …}, {v1, v2…}}; |
| Example:- int n[3] = {1,2,3}; | Example:-  int n[3][3] = {{1,2,3},{1,2,3}}; |

1. **Explain what pointers are in C and how they are declared and initialized. Why are pointers important in C?**
2. **Explain string handling functions like strlen(), strcpy(), strcat(), strcmp(), and strchr(). Provide examples of when these functions are useful.**

# Strlen()

* This function calculate the length of string.
* Syntax

Strlen (variable\_name);

* Example

Char s[100]=”Hello”;

Strlen(s);

# Strcpy ()

* It is a standard library function.
* It is used to copy one string into another.

#  Syntax  Example

Strcpy (variable\_name1, variable\_name2);

Char s1[100] = ”Hello”;

Char s2[100] = “Hi”;

Strcpy (s1,s2);

# Strcat ()

* The Strcat () function is used to string concatenation.
* It make a sentence by including string together.
* Syntax

Strcat (variable\_name1, variable\_name2);

* Example

Char s1[100] = ”Hello”;

Char s2[100] = “How Are You”;

Strcat (s1,s2);

# Strcmp ()

* This function is built-in library function.
* It is take two string as argument & compare those string.
* Syntax

Strcmp (variable\_name1, variable\_name2);

#  Example

Char s1[100] = ”Hello”;

Char s2[100] = “Hi”;

Strcmp (s1,s2);

# Strchar ()

* This function searches for the first occurrence of a character in a string. It is useful for finding specific characters.

 Example

Char str[100] = “Kanishk sharma”

Char str[100] = “Bachelor of technology ”

1. **Explain the concept of structures in C. Describe how to declare, initialize, and access structure members.**
2. **Explain the importance of file handling in C. Discuss how to perform file operations like opening, closing, reading, and writing files.**