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

Ans : C programming was developed by Dennis Ricthie at Bell Labs in 1972 as an evolution of the B language, primarily for system programming. It gained popularity through it use in developing the Unix operating system. In the 1980, it was standardized as ANSI C, and over time, updates like C99 and C11 introduced new features. C remains a foundational language, influencing many modern languages and widely used in system-level and performance-critical applications.

C programming is Crucial for system-level programming due to its efficiency, low-level memory access, and portability. It is widely used in developing operating systems, embedded systems, and device drivers. C also forms the foundation for many modern languages like C++, Java, and Python, and is used in performance-critical applications like real-time systems, networking, and gaming. Its role in developing compilers and interpreters further highlights its importance in the software development ecosystem.

Q : 2 Research and provide three real-world applications where C programming is extensively used, such as in embedded systems, operating systems, or game development.

Ans : 1). Operating Systems & Kernels :

Linux kernel, Windows Kernel, MacOS/iOS, Android Low-Level.

- Direct hardware access
- Zero overhead
- Predictable memory usage
- Portability across architecture (x86,ARM, RISC-V)

2). Embedded System & IoT Devices :

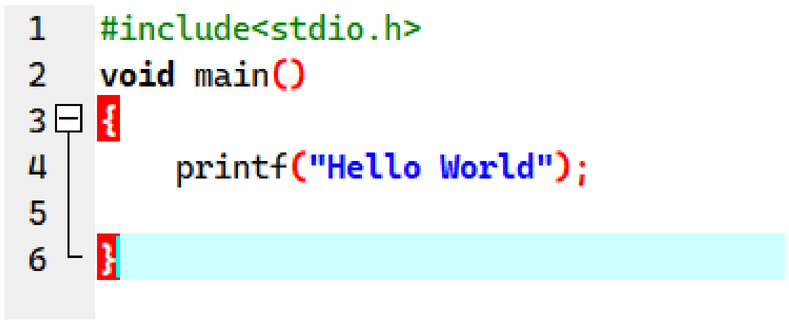
- Car ki ECU (Engine Control Unit)
- Smart TV, AC, Washing Machine, Microwave.
- Medical Devices – ECG machine, Ventilators
- Drone, Robot, Arduino, STM32 microcontrollers.

3). High-Performance Applications & Game Engines :

- Unreal Engine
- Photoshop, Adobe Premiere
- MySQL, PostgreSQL database engines
- Nginx web server

Q : 3 Install a C compiler on your system and configure the IDE. Write your first program to print "Hello, World!" and run it.

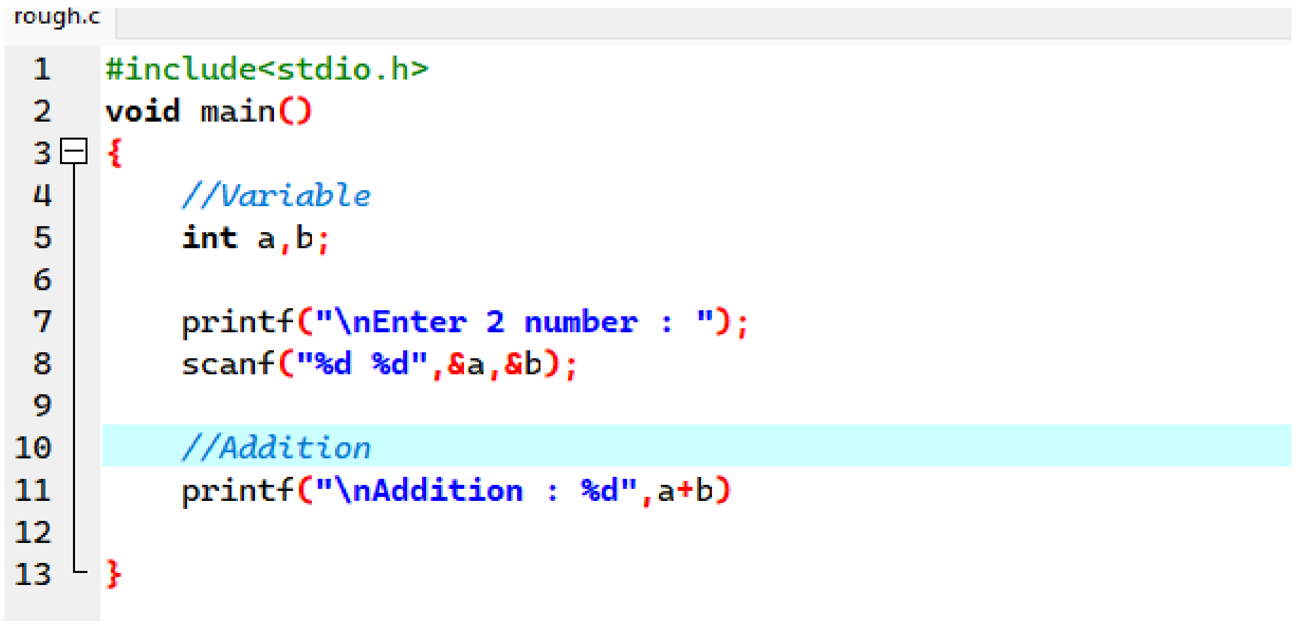
Ans:



```c
#include<stdio.h>
void main()
{
    printf("Hello World");

}
```

Q : 4 Explain the basic structure of a C program, including headers, main function, comments, data types, and variables. Provide examples.

Ans:



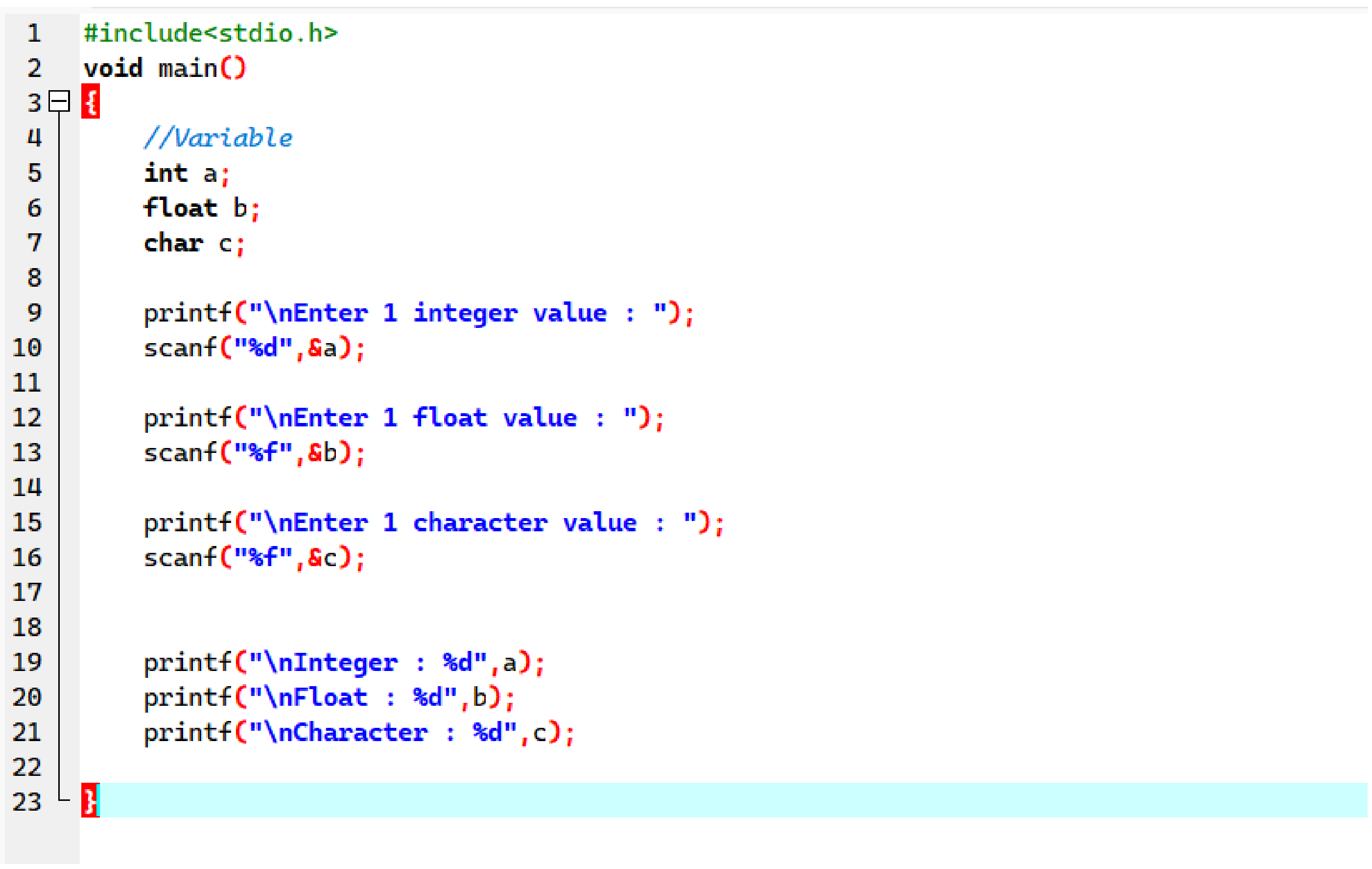
rough.c

```c
#include<stdio.h>
void main()
{
    //Variable
    int a,b;

    printf("\nEnter 2 number : ");
    scanf("%d %d",&a,&b);

    //Addition
    printf("\nAddition : %d",a+b)

}
```

- Header File : `#include<stdio.h>` (*Preprocessor Directive*)
- void main() : main function, from where execution start.
- Variable : a, b
- Data Types : `integer (a, b)`
- Comments : Comments are used to increase readability.

Q : 5 Write a C program that includes variables, constants, and comments. Declare and use different data types (int, char, float) and display their values.

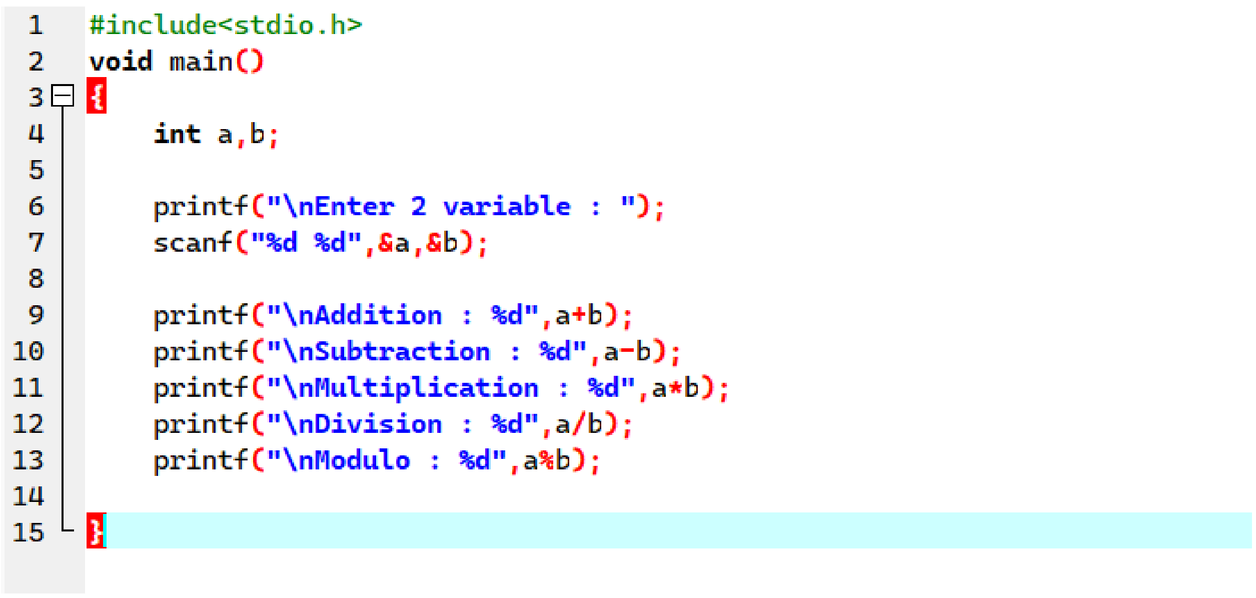
Ans:



```c
#include<stdio.h>
void main()
{
    //Variable
    int a;
    float b;
    char c;

    printf("\nEnter 1 integer value : ");
    scanf("%d",&a);

    printf("\nEnter 1 float value : ");
    scanf("%f",&b);

    printf("\nEnter 1 character value : ");
    scanf("%f",&c);


    printf("\nInteger : %d",a);
    printf("\nFloat : %d",b);
    printf("\nCharacter : %d",c);

}
```

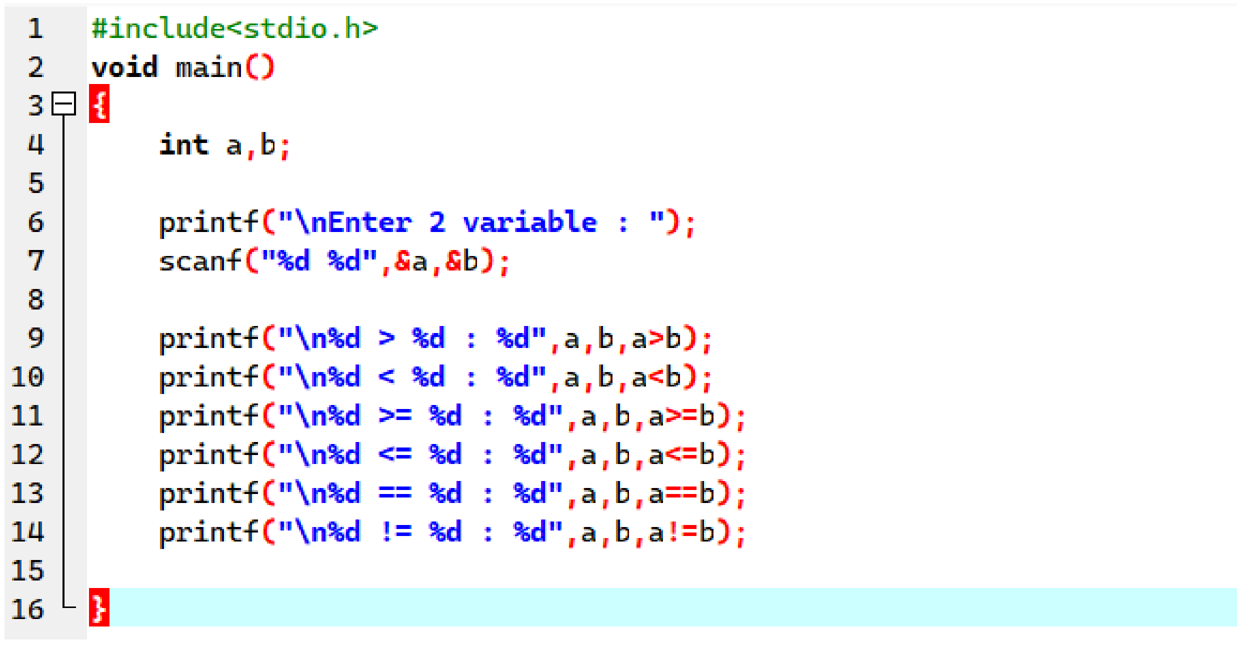
Q : 6 Write notes explaining each type of operator in C: arithmetic, relational, logical, assignment, increment/decrement, bitwise, and conditional operators.

Ans: 1). Arithmetic Operators: Operators that are used to perform mathematical operations like (Addition, Subtraction, Multiplication, Division, Modulo etc).



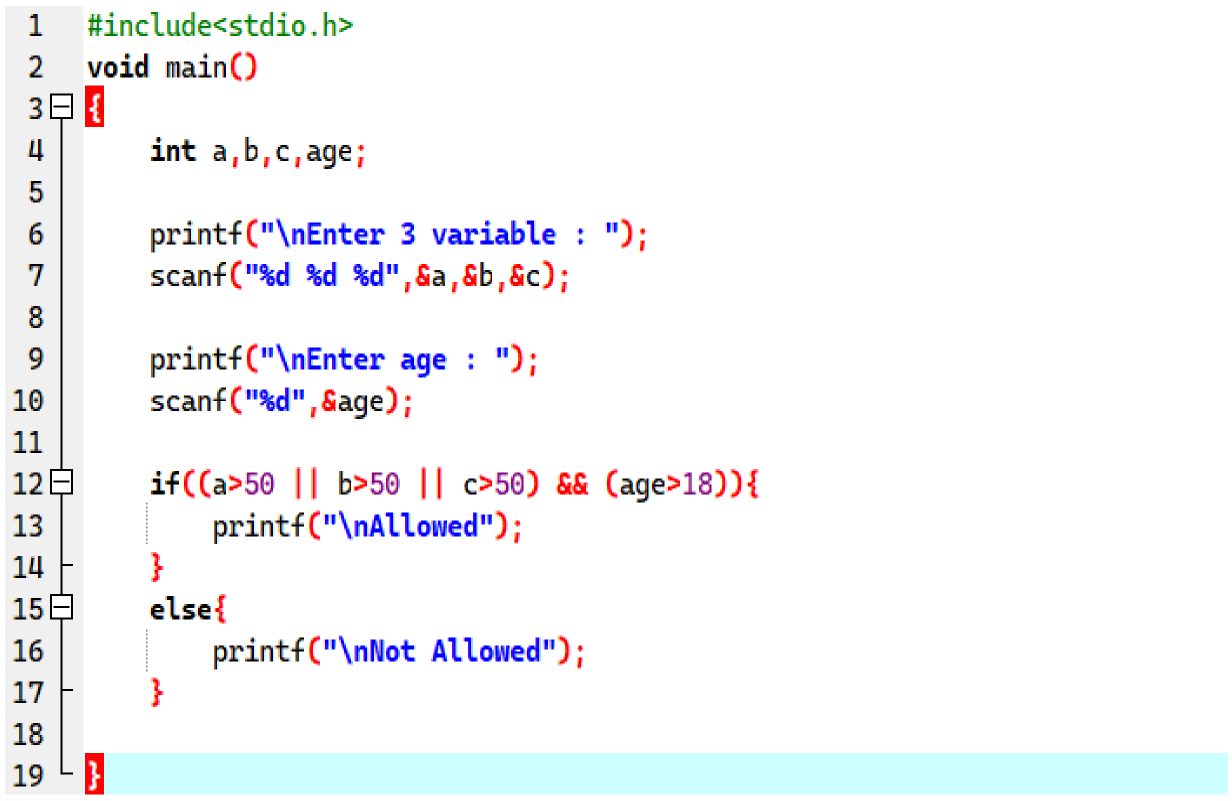
```c
#include<stdio.h>
void main()
{
    int a,b;

    printf("\nEnter 2 variable : ");
    scanf("%d %d",&a,&b);

    printf("\nAddition : %d",a+b);
    printf("\nSubtraction : %d",a-b);
    printf("\nMultiplication : %d",a*b);
    printf("\nDivision : %d",a/b);
    printf("\nModulo : %d",a%b);

}
```

2). Relational Operators: Relational Operator is used to compare two values and give output in form of Boolean value.



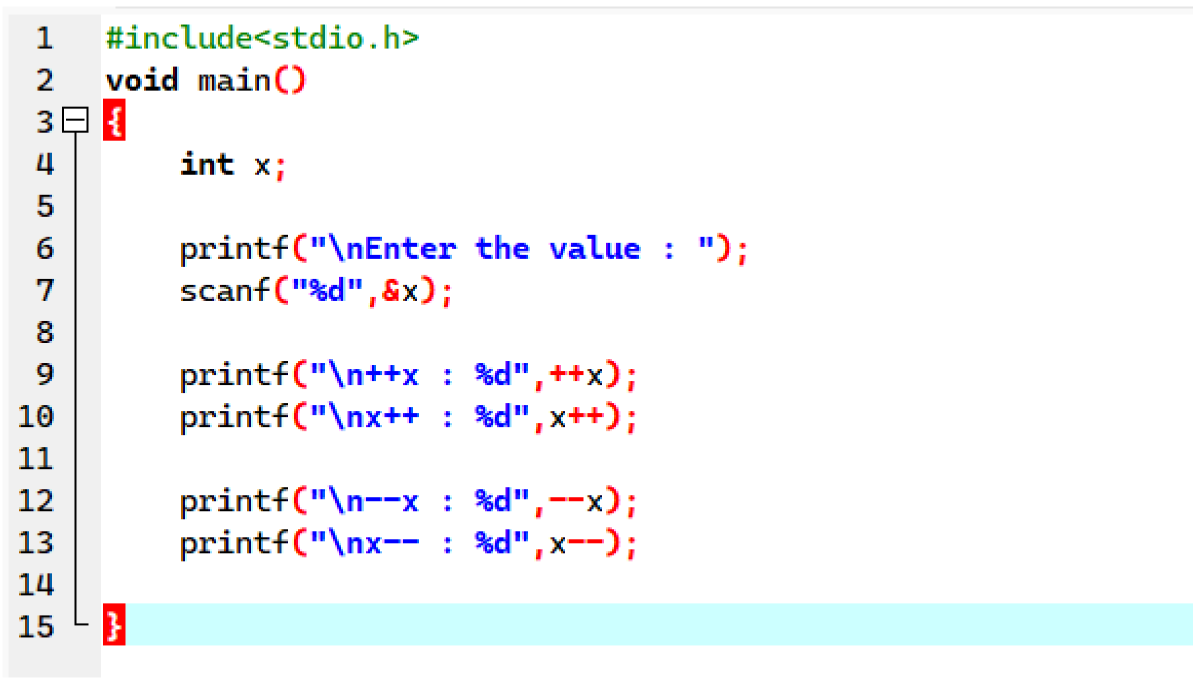
```c
#include<stdio.h>
void main()
{
    int a,b;

    printf("\nEnter 2 variable : ");
    scanf("%d %d",&a,&b);

    printf("\n%d > %d : %d",a,b,a>b);
    printf("\n%d < %d : %d",a,b,a<b);
    printf("\n%d >= %d : %d",a,b,a>=b);
    printf("\n%d <= %d : %d",a,b,a<=b);
    printf("\n%d == %d : %d",a,b,a==b);
    printf("\n%d != %d : %d",a,b,a!=b);

}
```

3). Logical Operators: Logical Operators is used to perform logical operations using (&&, ||,!).



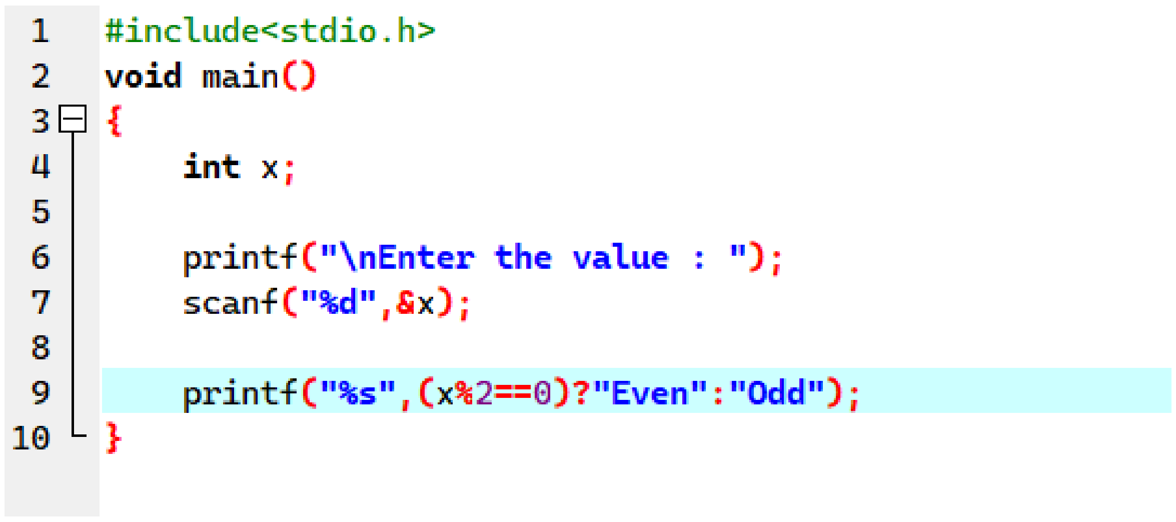
```c
#include<stdio.h>
void main()
{
    int a,b,c,age;

    printf("\nEnter 3 variable : ");
    scanf("%d %d %d",&a,&b,&c);

    printf("\nEnter age : ");
    scanf("%d",&age);

    if((a>50 || b>50 || c>50) && (age>18)){
        printf("\nAllowed");
    }
    else{
        printf("\nNot Allowed");
    }

}
```

4). Increment & Decrement Operators: Increment & Decrement is unary operator and used to increase and decrease the value by 1.



```c
#include<stdio.h>
void main()
{
    int x;

    printf("\nEnter the value : ");
    scanf("%d",&x);

    printf("\n++x : %d",++x);
    printf("\nx++ : %d",x++);

    printf("\n--x : %d",--x);
    printf("\nx-- : %d",x--);

}
```

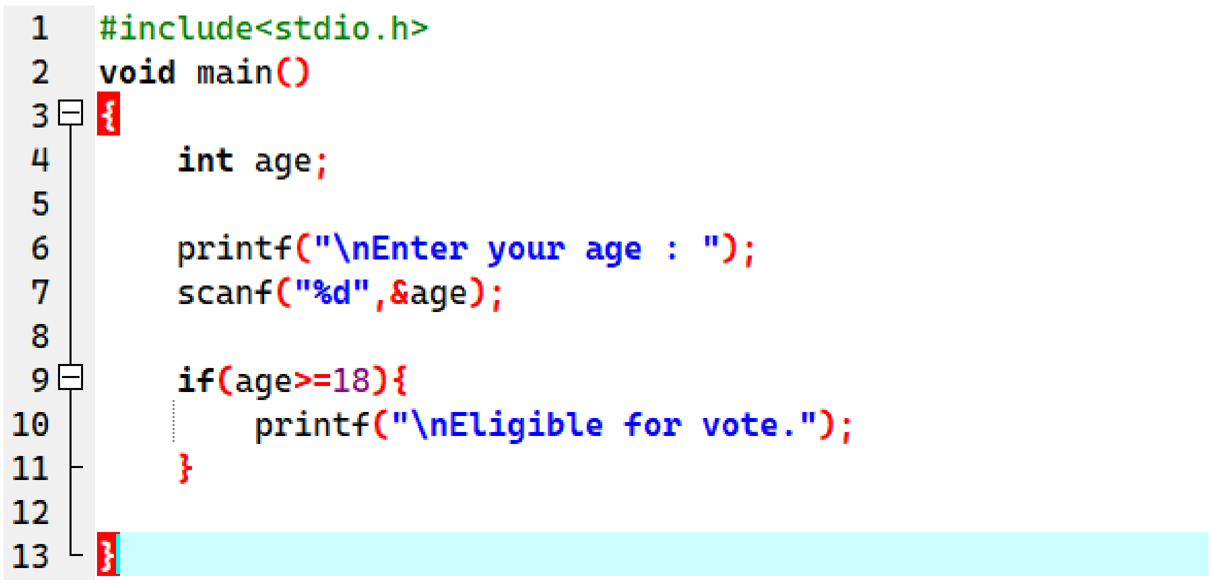
4). Ternary Operator:



```c
#include<stdio.h>
void main()
{
    int x;

    printf("\nEnter the value : ");
    scanf("%d",&x);

    printf("%s",(x%2==0)?"Even":"Odd");
}
```

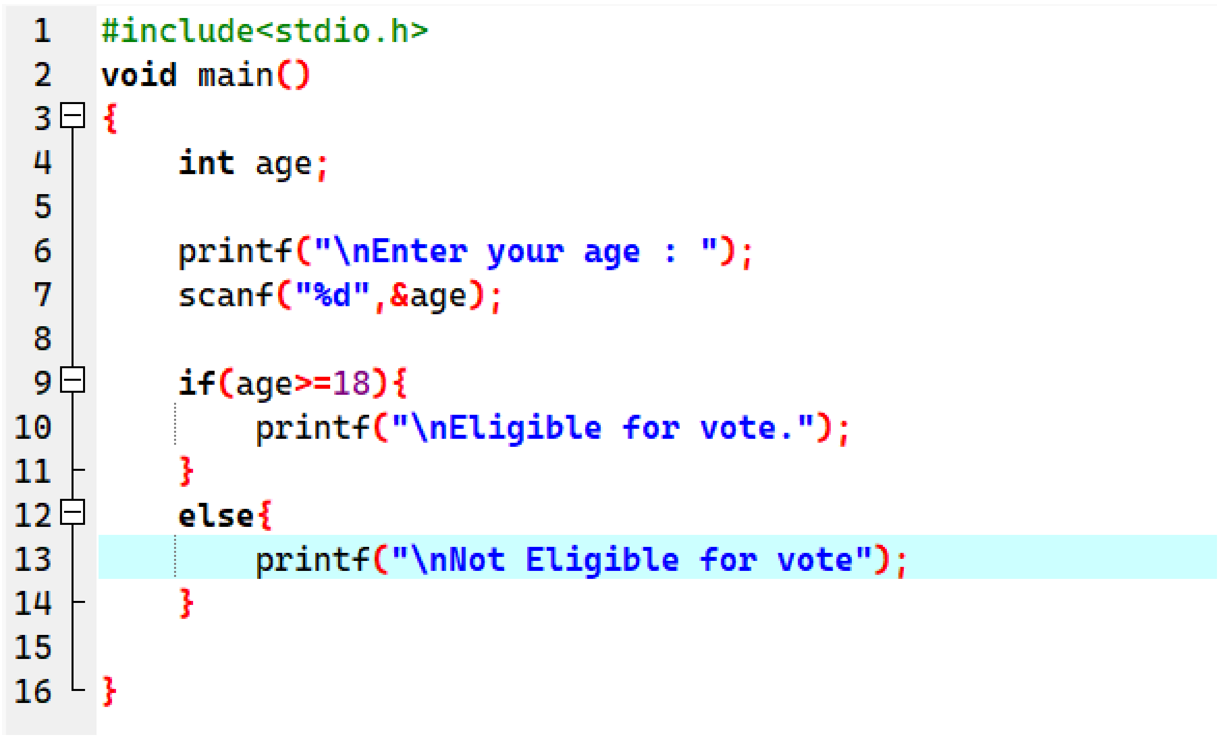
Q : 7 Explain decision-making statement in C (if, else, nested if-else, switch). Provide example of each.

Ans: 1). If - "if" is conditional statement. It used to check condition, if condition is True then body will execute else nothing.



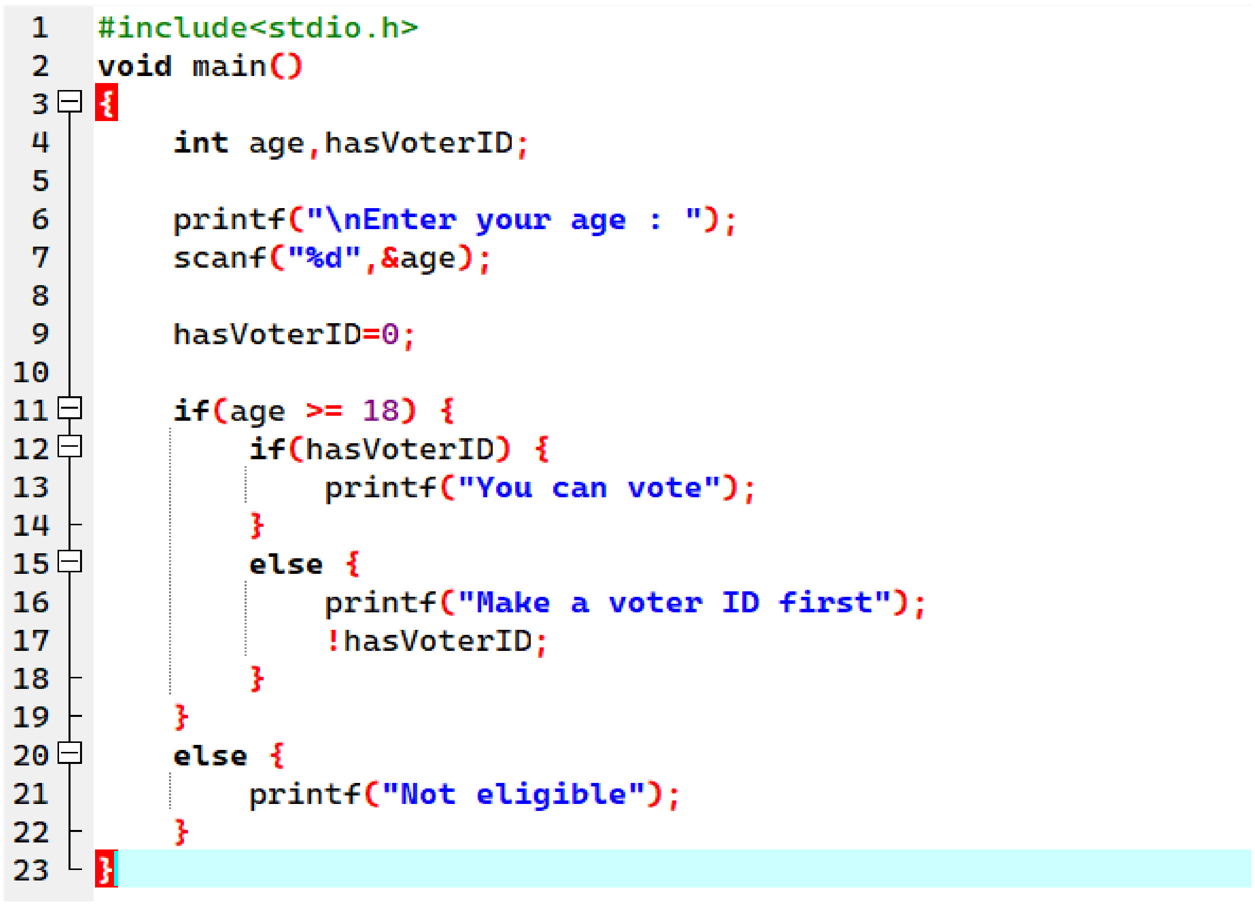
```c
#include<stdio.h>
void main()
{
	int age;

	printf("\nEnter your age : ");
	scanf("%d",&age);

	if(age>=18){
		printf("\nEligible for vote.");
	}

}
```

2). If-else - “if-else” is a condition statement. It used to check condition, if condition is TRUE then it go inside and execute “if” block, if condition is not TRUE and else block will execute.



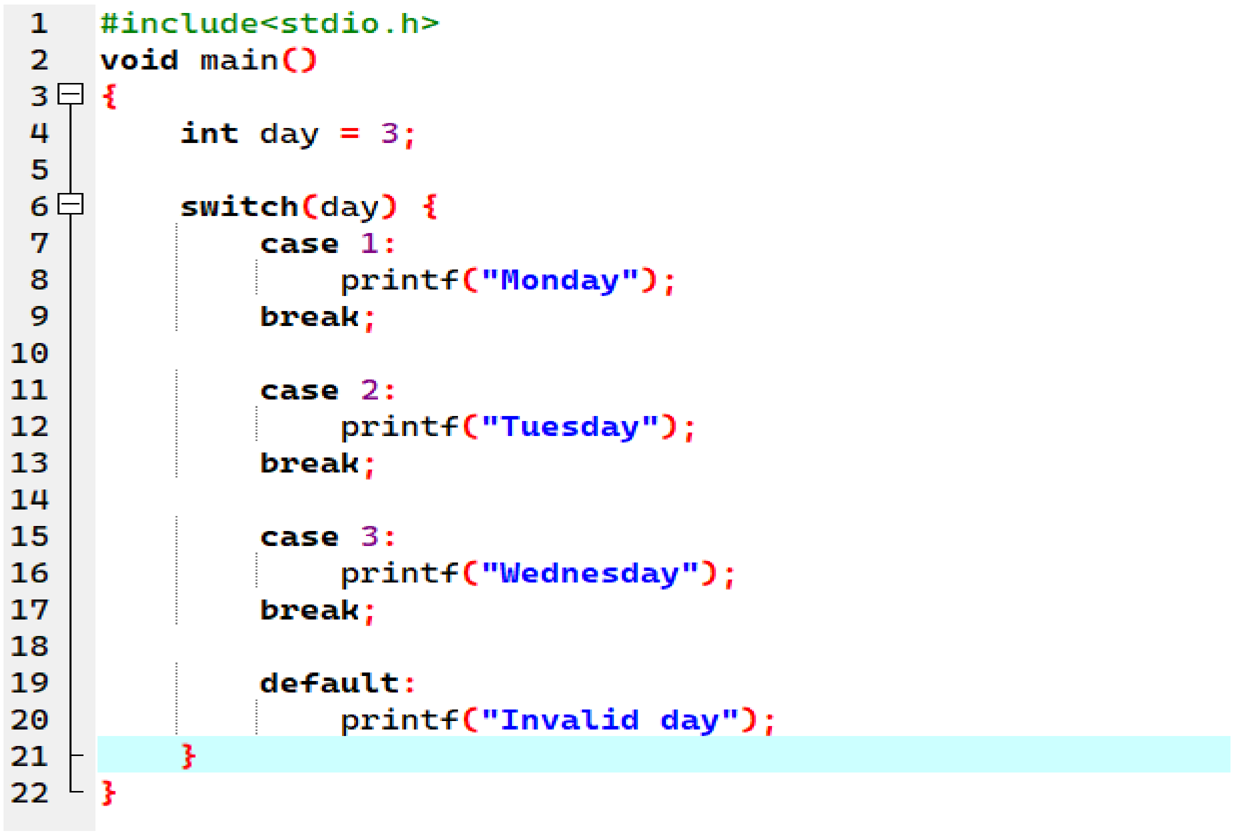
```c
#include<stdio.h>
void main()
{
	int age;

	printf("\nEnter your age : ");
	scanf("%d",&age);

	if(age>=18){
		printf("\nEligible for vote.");
	}
	else{
		printf("\nNot Eligible for vote");
	}

}
```

3). Nested if-else – “nested if-else” is a conditional statement. It used to check condition, if condition is TRUE then it go inside and execute “if” block, inside “if” block it again check “if” condition is its TRUE then it execute if condition. if condition is not TRUE and else block will execute.

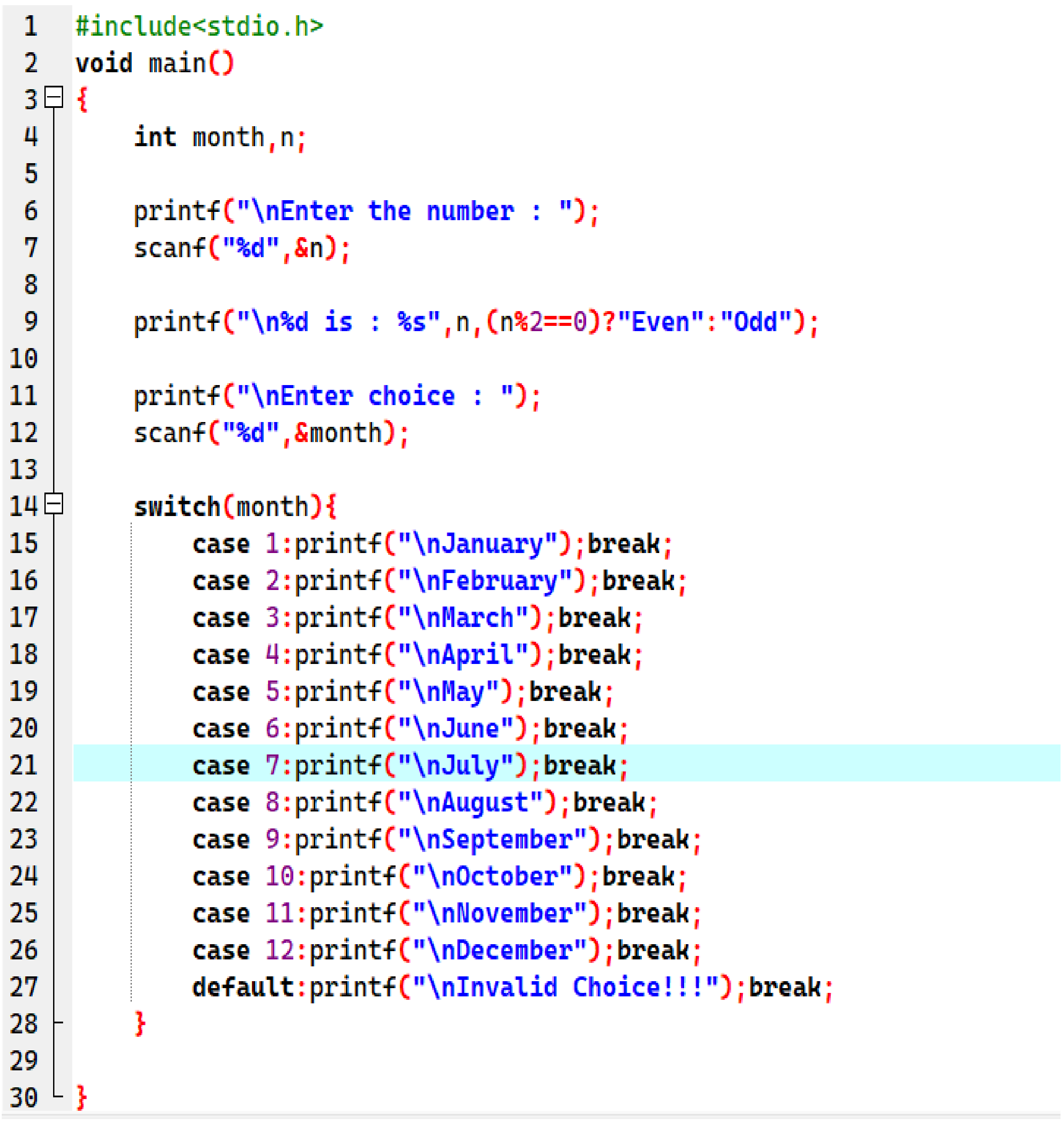


```c
#include<stdio.h>
void main()
{
    int age,hasVoterID;

    printf("\nEnter your age : ");
    scanf("%d",&age);

    hasVoterID=0;

    if(age >= 18) {
        if(hasVoterID) {
            printf("You can vote");
        }
        else {
            printf("Make a voter ID first");
            !hasVoterID;
        }
    }
    else {
        printf("Not eligible");
    }
}
```

4). Switch – "switch" is also conditional statement. It checks the condition based on choice entered by the user.



```c
#include<stdio.h>
void main()
{
    int day = 3;

    switch(day) {
        case 1:
            printf("Monday");
        break;

        case 2:
            printf("Tuesday");
        break;

        case 3:
            printf("Wednesday");
        break;

        default:
            printf("Invalid day");
    }
}
```

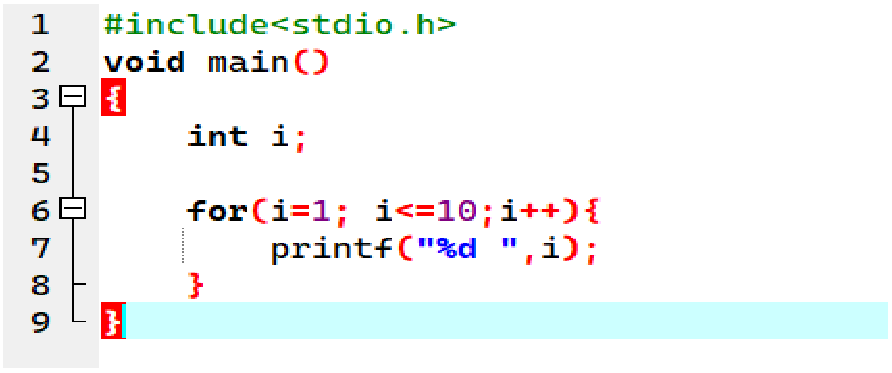


```c
#include<stdio.h>
void main()
{
    int month,n;

    printf("\nEnter the number : ");
    scanf("%d",&n);

    printf("\n%d is : %s",n,(n%2==0)?"Even":"Odd");

    printf("\nEnter choice : ");
    scanf("%d",&month);

    switch(month){
        case 1:printf("\nJanuary");break;
        case 2:printf("\nFebruary");break;
        case 3:printf("\nMarch");break;
        case 4:printf("\nApril");break;
        case 5:printf("\nMay");break;
        case 6:printf("\nJune");break;
        case 7:printf("\nJuly");break;
        case 8:printf("\nAugust");break;
        case 9:printf("\nSeptember");break;
        case 10:printf("\nOctober");break;
        case 11:printf("\nNovember");break;
        case 12:printf("\nDecember");break;
        default:printf("\nInvalid Choice!!!");break;
    }

}
```

Q : 8 Compare and Contrast while Loops, and do-while loops. Explain the scenarios in which each loop is most appropriate.

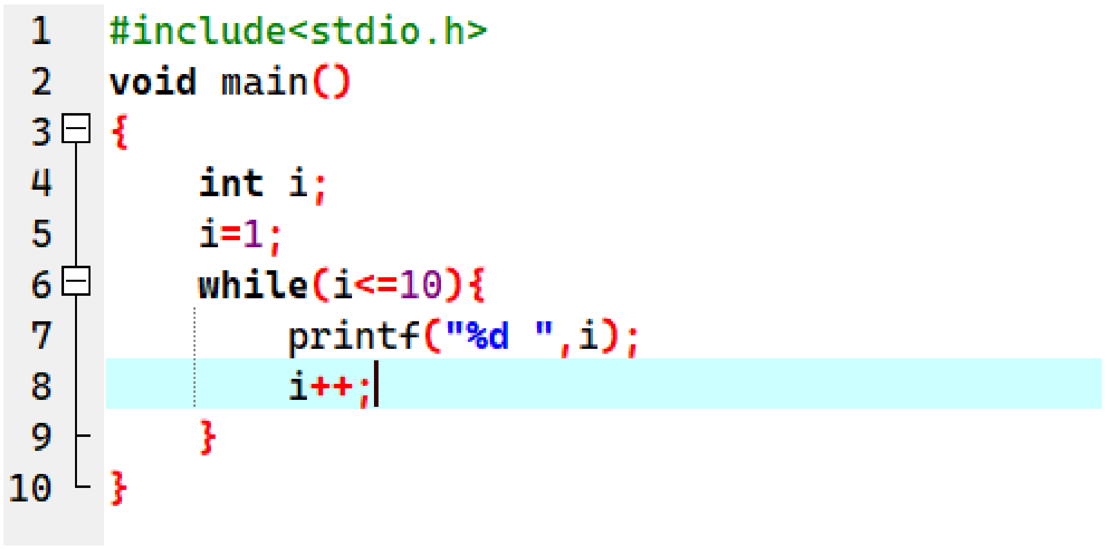
Write C program to print number from 1 to 100 using all three types of loops.

Ans: For Loop: Used when number of iterations is already known.



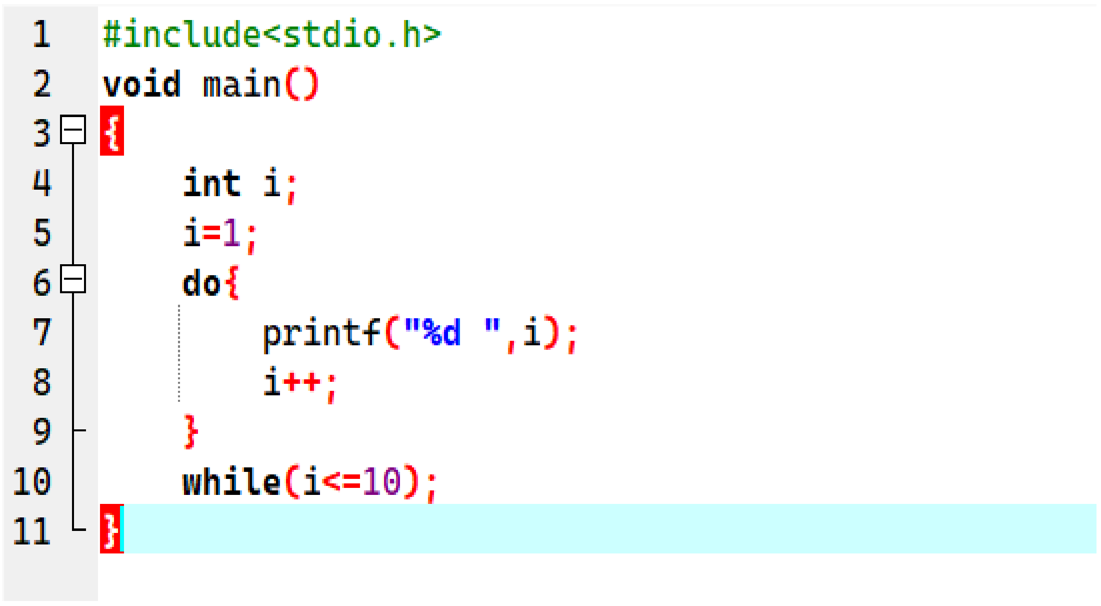
```c
#include<stdio.h>
void main()
{
    int i;

    for(i=1; i<=10;i++){
        printf("%d ",i);
    }
}
```

While Loop: Used when we do not know the number of iterations.



```c
#include<stdio.h>
void main()
{
	int i;
	i=1;
	while(i<=10){
		printf("%d ",i);
		i++;
	}
}
```

Do-while Loop: Used when we do not know the number of iterations, and "do-while" will execute the loop at least one time.



```c
#include<stdio.h>
void main()
{
	int i;
	i=1;
	do{
		printf("%d ",i);
		i++;
	}
	while(i<=10);
}
```

Q : 9 Explain the use of "break", "continue", and "goto" statements in C. Provide examples of each.

Write a C program that uses the break statement to stop printing numbers when it reaches 5.

Modify the program to skip printing the number 3 using the continue statement.

Ans: 1) Break:

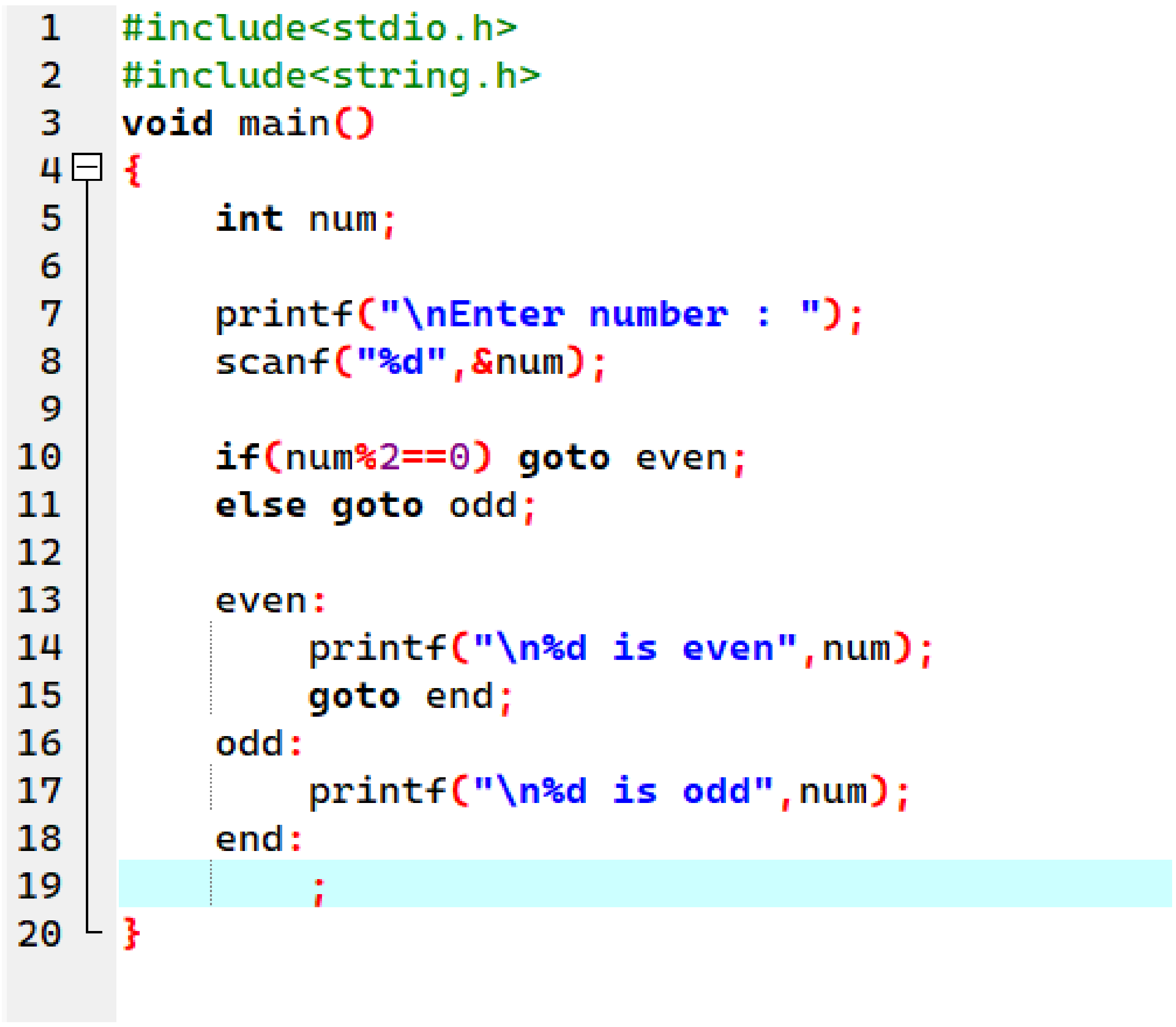
The break statement is used to terminate a loop or a switch block immediately. Control is transferred to the statement just after the loop/switch.

2). Continue:

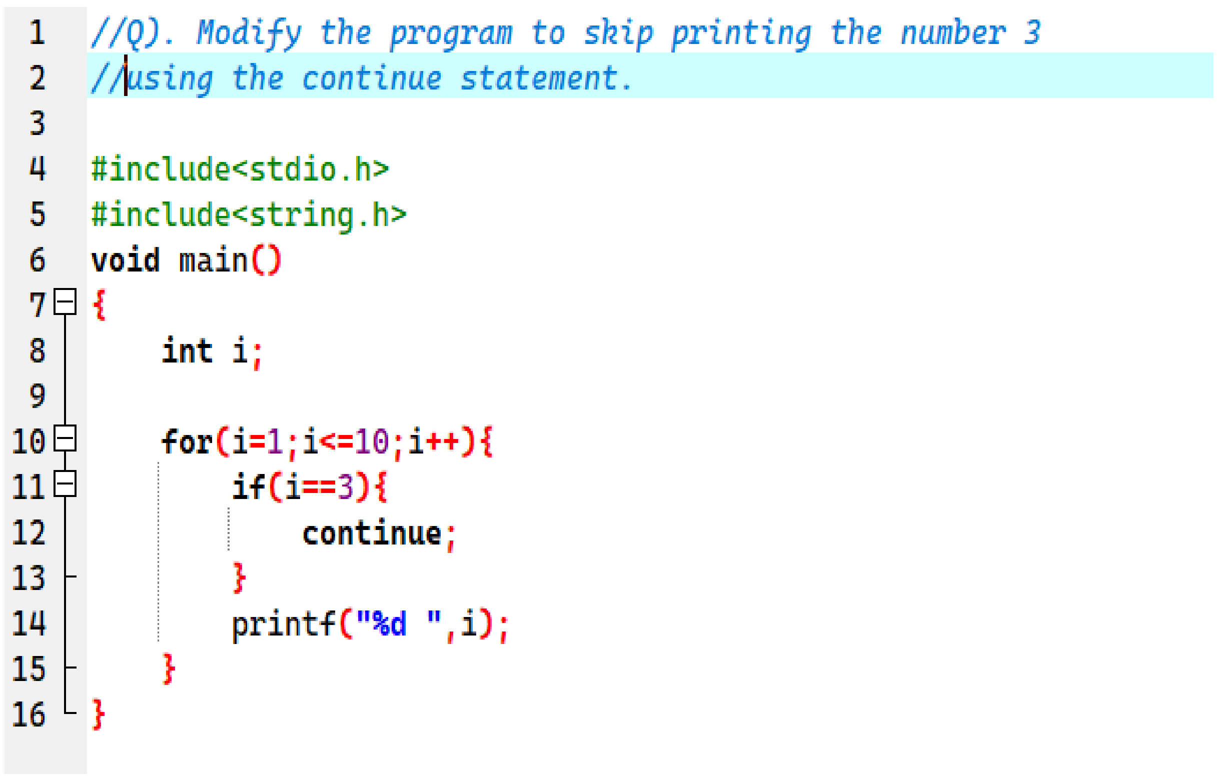
The continue statement skips the current iteration of a loop and transfers control to the next iteration. It does not terminate the loop.

3). goto:

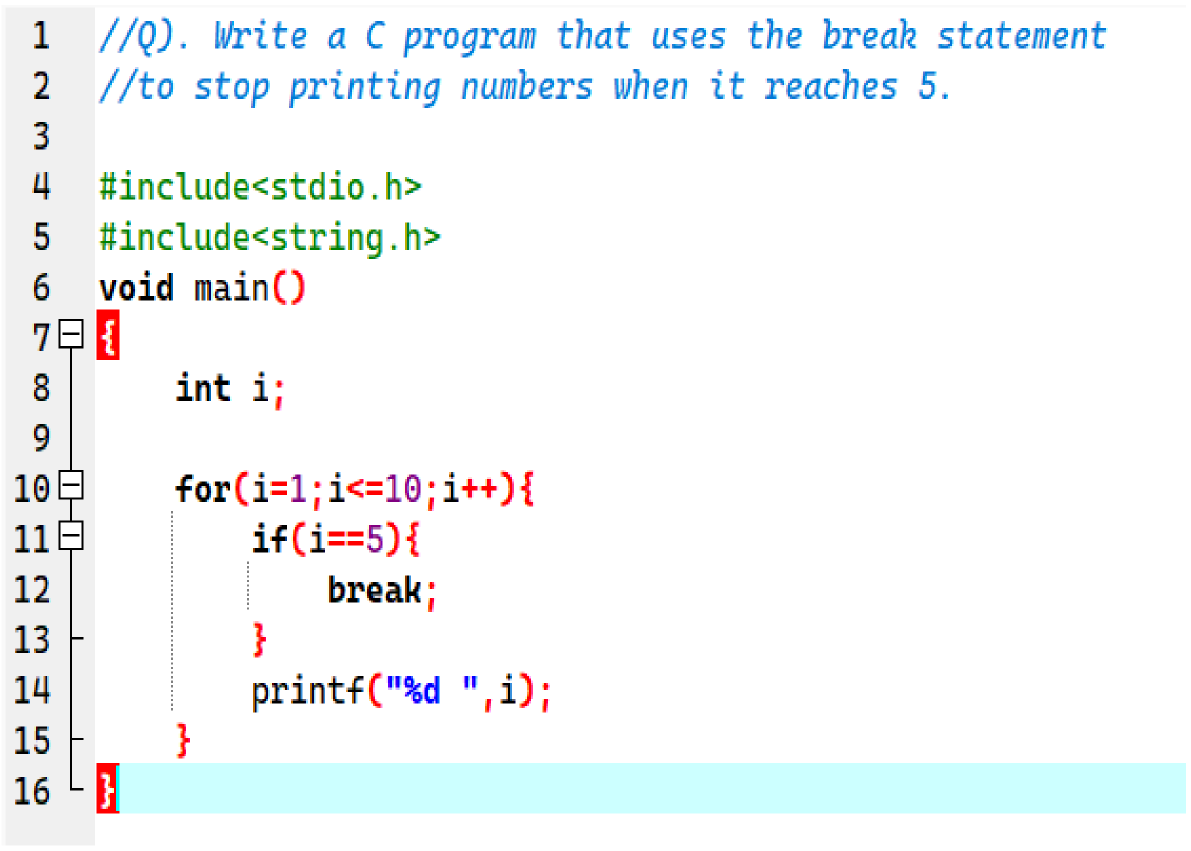
The goto statement is used to jump directly to a labelled statement. It can be used for breaking out of deeply nested loops or for simple error handling. However, it should be used carefully because it can make code hard to understand.



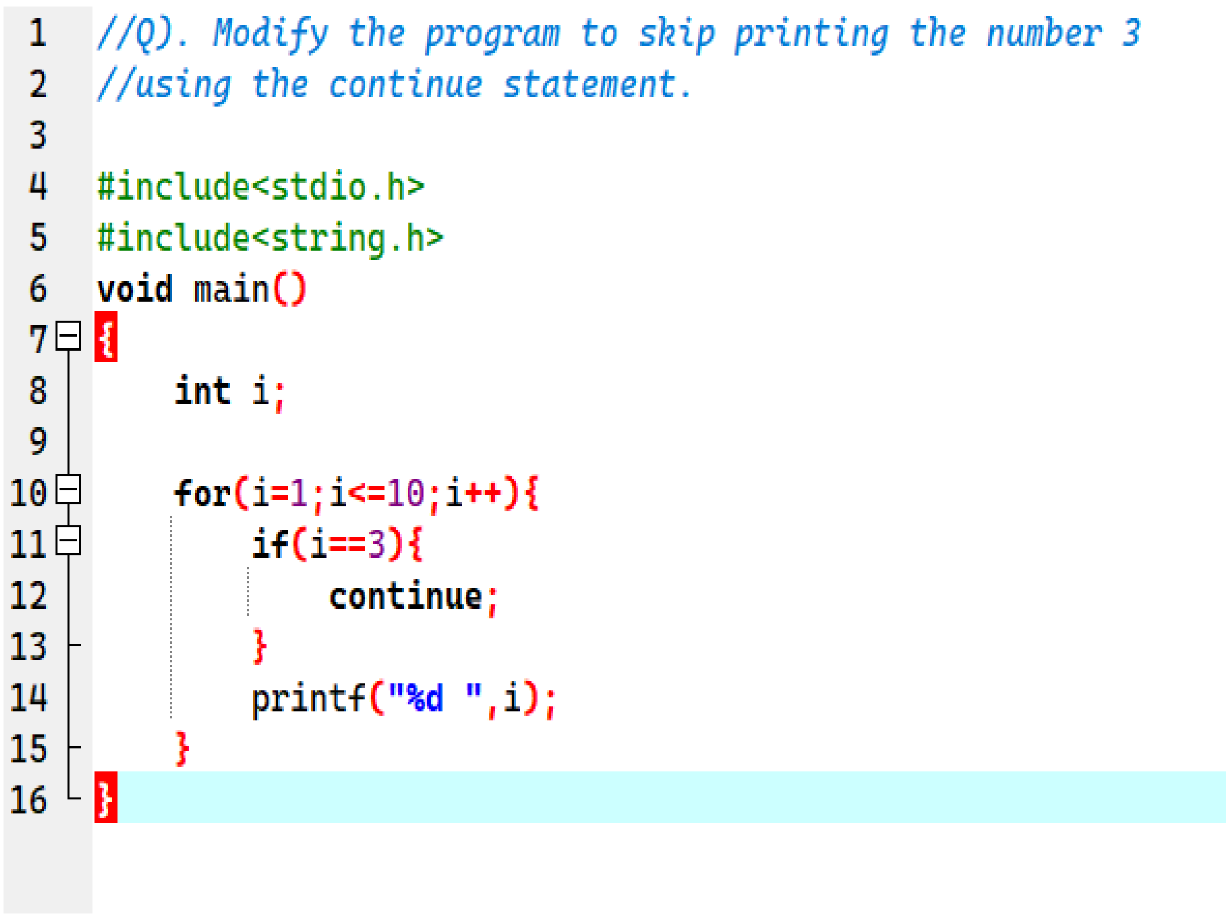
```c
#include<stdio.h>
#include<string.h>
void main()
{
	int num;

	printf("\nEnter number : ");
	scanf("%d",&num);

	if(num%2==0) goto even;
	else goto odd;

	even:
		printf("\n%d is even",num);
		goto end;
	odd:
		printf("\n%d is odd",num);
	end:
		;
}
```



```c
//Q). Modify the program to skip printing the number 3
//using the continue statement.

#include<stdio.h>
#include<string.h>
void main()
{
	int i;

	for(i=1;i<=10;i++){
		if(i==3){
			continue;
		}
		printf("%d ",i);
	}
}
```



```c
//Q). Write a C program that uses the break statement
//to stop printing numbers when it reaches 5.

#include<stdio.h>
#include<string.h>
void main()
{
	int i;

	for(i=1;i<=10;i++){
		if(i==5){
			break;
		}
		printf("%d ",i);
	}
}
```



```c
//Q). Modify the program to skip printing the number 3
//using the continue statement.

#include<stdio.h>
#include<string.h>
void main()
{
    int i;

    for(i=1;i<=10;i++){
        if(i==3){
            continue;
        }
        printf("%d ",i);
    }
}
```

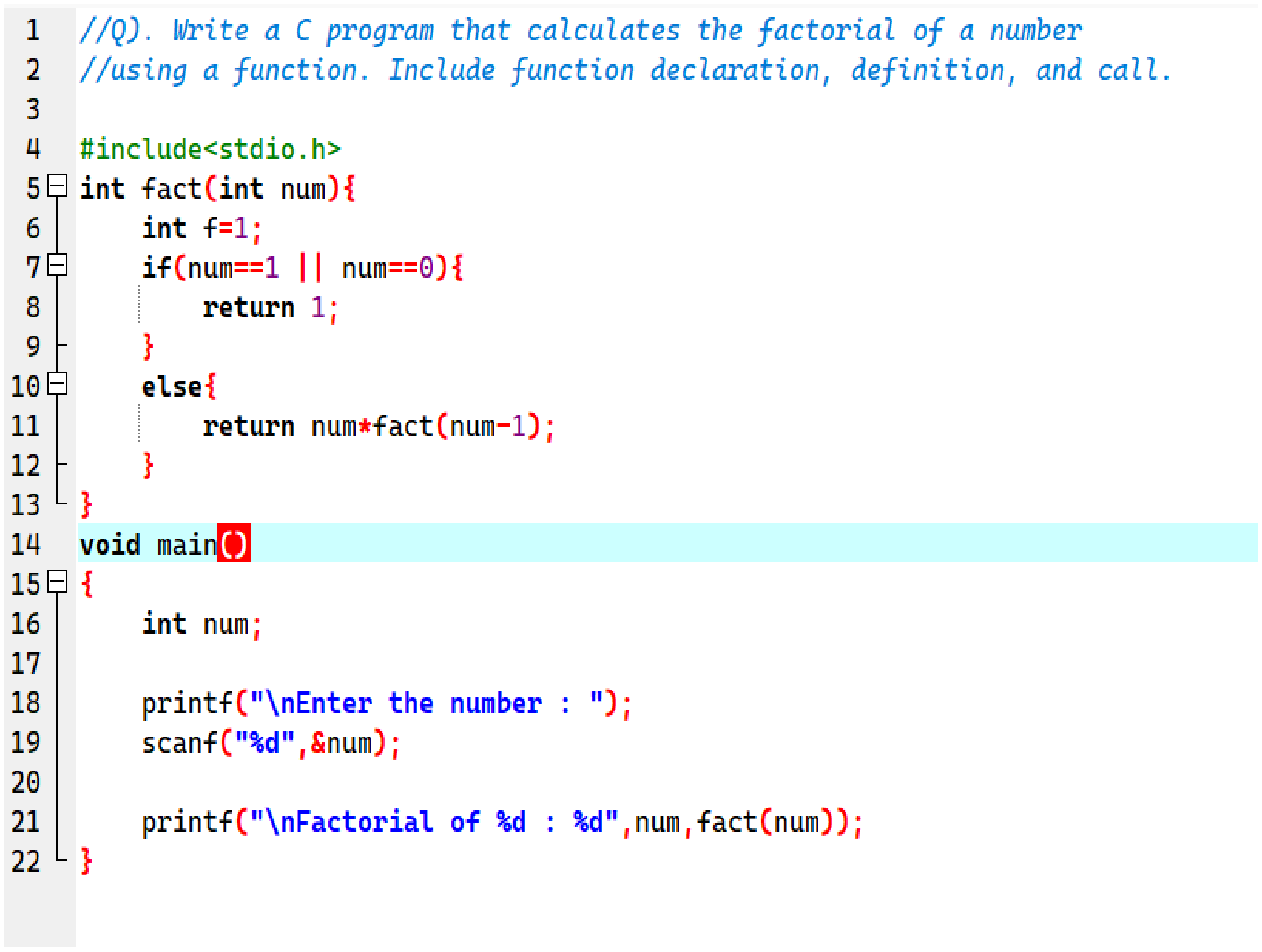
Q : 10 What are functions in C? Explain function declaration, definition, and how to call a function. Provide examples.

Write a C program that calculates the factorial of a number using a function. Include function declaration, definition, and call.

Ans: A function in C is a block of code that performs a specific task.

It helps in:

- Modularity (dividing code into smaller parts)
- Reusability (use the same code multiple times)
- Readability (makes code easier to understand)
- Debugging (errors can be fixed easily)

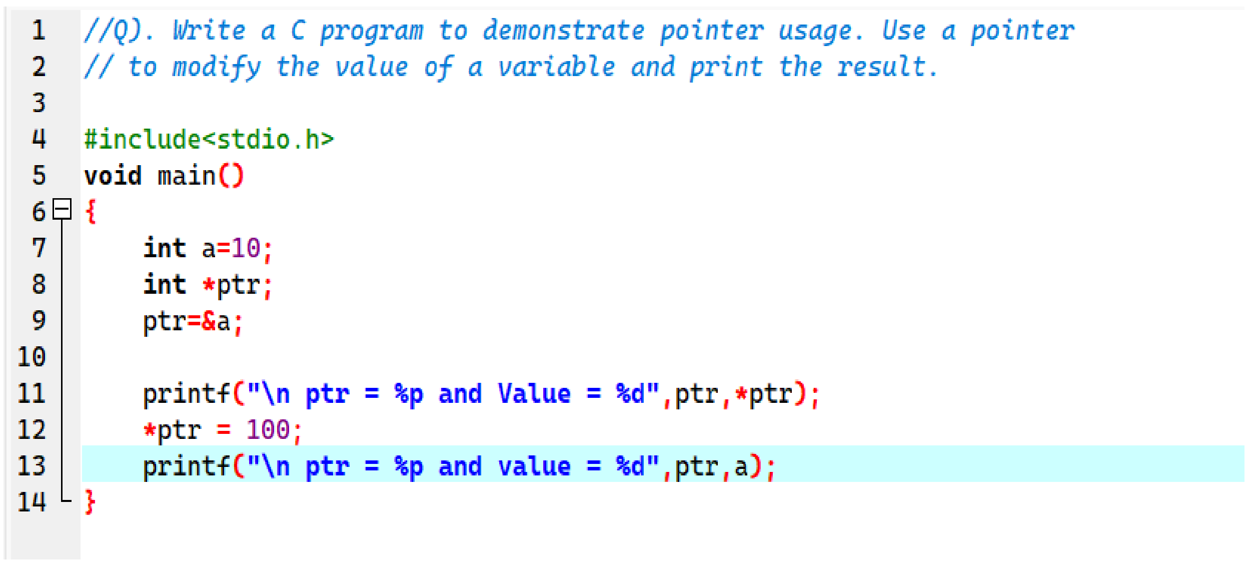


```c
//Q). Write a C program that calculates the factorial of a number
//using a function. Include function declaration, definition, and call.

#include<stdio.h>
int fact(int num){
    int f=1;
    if(num==1 || num==0){
        return 1;
    }
    else{
        return num*fact(num-1);
    }
}
void main()
{
    int num;

    printf("\nEnter the number : ");
    scanf("%d",&num);

    printf("\nFactorial of %d : %d",num,fact(num));
}
```

Q : 11 Explain what pointers are in C and how they are declared and initialized. Why are pointers important in C?

Write a C program to demonstrate pointer usage. Use a pointer to modify the value of a variable and print the result.

Ans: A pointer in C is a variable that stores the memory address of another variable. Instead of storing a value directly, a pointer stores where that value is located in memory.



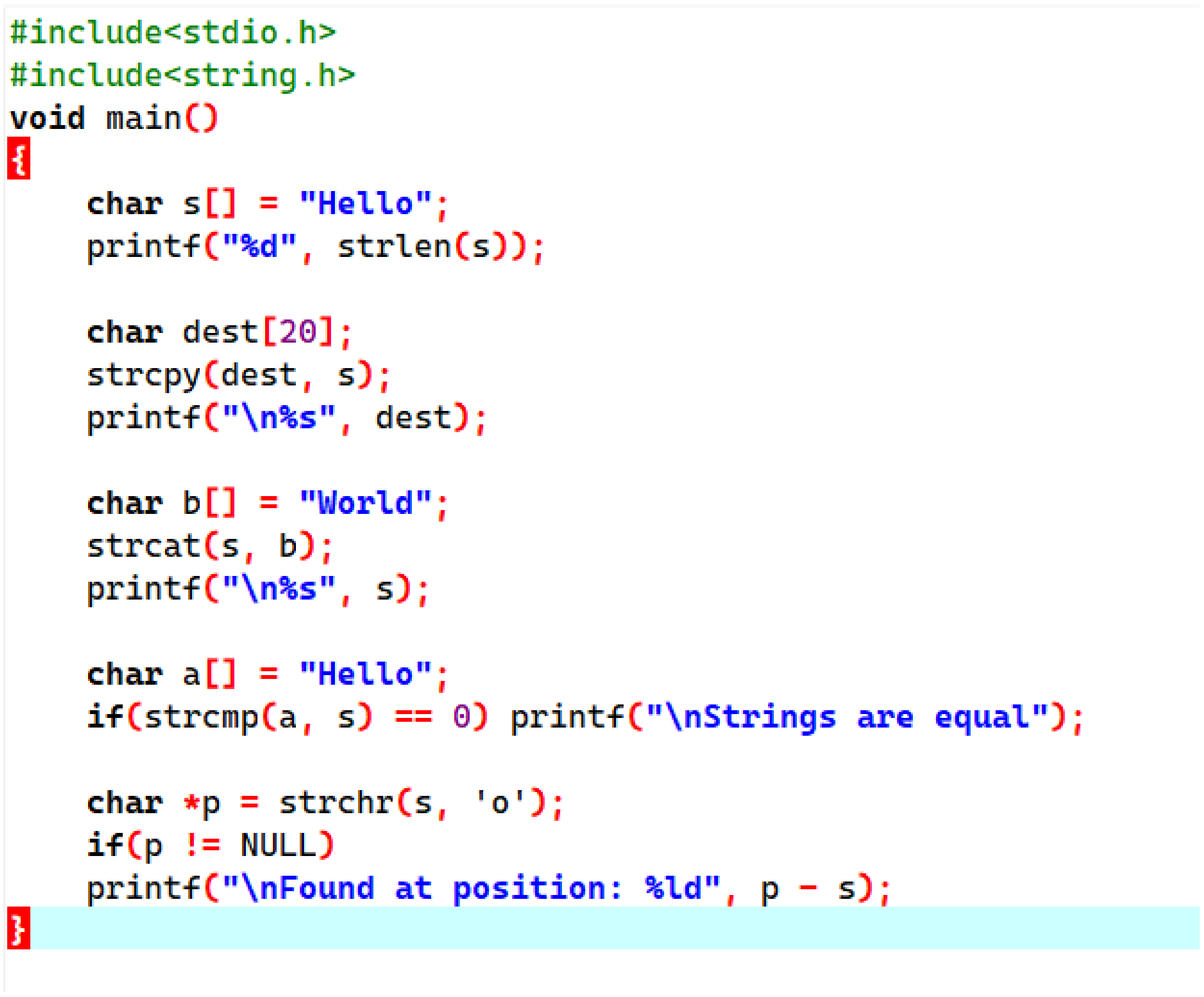
```c
//Q). Write a C program to demonstrate pointer usage. Use a pointer
// to modify the value of a variable and print the result.

#include<stdio.h>
void main()
{
    int a=10;
    int *ptr;
    ptr=&a;

    printf("\n ptr = %p and Value = %d",ptr,*ptr);
    *ptr = 100;
    printf("\n ptr = %p and value = %d",ptr,a);
}
```

Q : 12 Explain string handling functions like `strlen()`, `strcpy()`, `strcat()`, `strcmp()`, and `strchr()`. Provide examples of when these functions are useful.

Write a C program that takes two strings from the user and concatenates them using `strcat()`. Display the concatenated string and its length using `strlen()`.

Ans :

- `strlen()` : Returns the number of characters in a string excluding the null character `'\0'`.
- `strcpy()` : Copies source string into destination string (including `'\0'`).
- `strcat()` : Appends source string to the end of destination string.
- `strcmp()` : Compares two strings character by character.
- `strchr()` : Returns a pointer to the first occurrence of a character in a string.

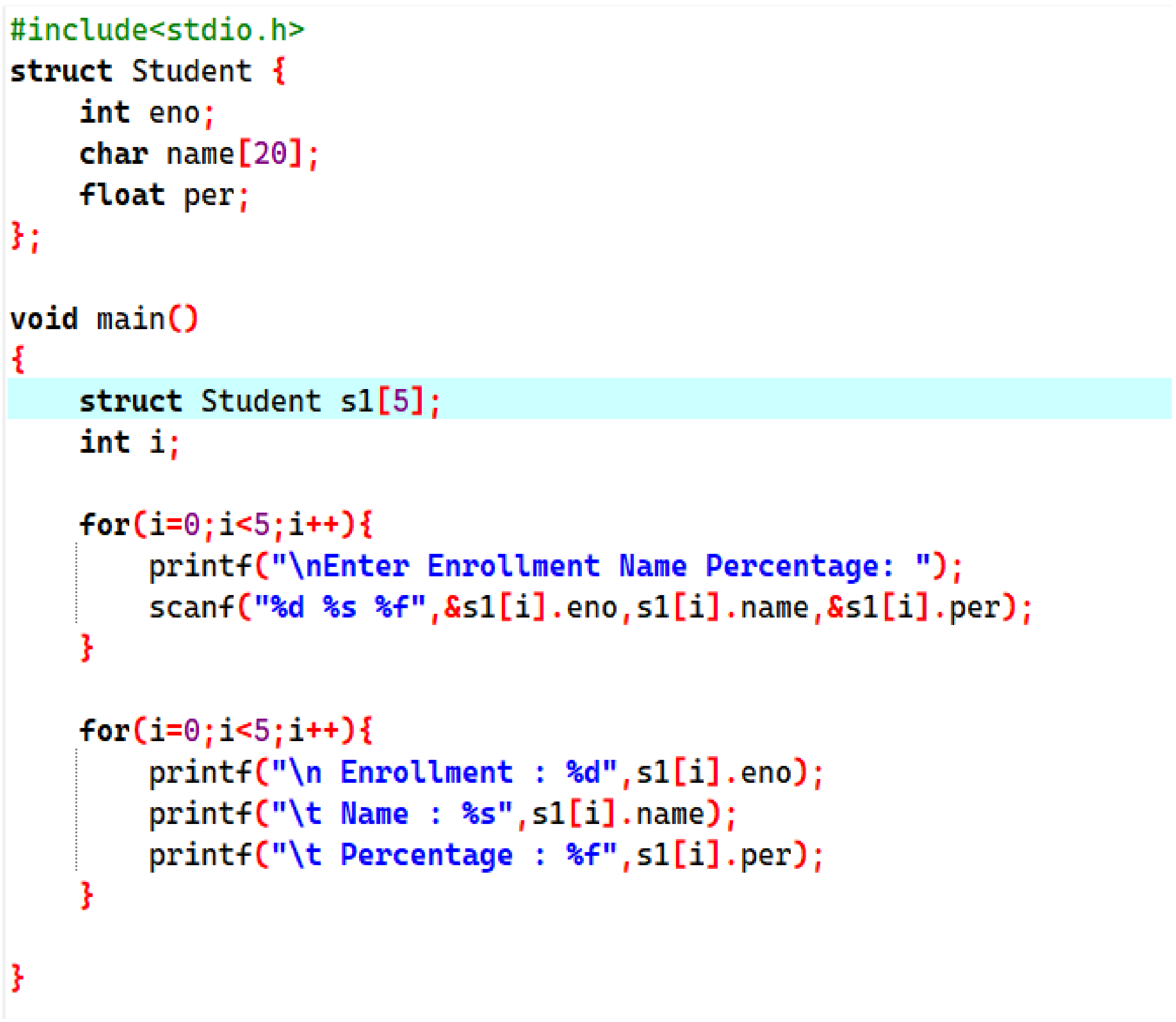


```
#include<stdio.h>
#include<string.h>
void main()
{
    char s[] = "Hello";
    printf("%d", strlen(s));

    char dest[20];
    strcpy(dest, s);
    printf("\n%s", dest);

    char b[] = "World";
    strcat(s, b);
    printf("\n%s", s);

    char a[] = "Hello";
    if(strcmp(a, s) == 0) printf("\nStrings are equal");

    char *p = strchr(s, 'o');
    if(p != NULL)
    printf("\nFound at position: %ld", p - s);
}
```

Q : 13 Explain the concept of structures in C. Describe how to declare, initialize, and access structure members.

Write a C program that defines a structure to store a student's details (name, roll number, and marks). Use an array of structures to store details of 3 students and print them.

Ans : A structure in C is a user-defined data type that allows you to group different data types under a single name. Structures are used when related data of different types needs to be stored together.



```c
#include<stdio.h>
struct Student {
    int eno;
    char name[20];
    float per;
};

void main()
{
    struct Student s1[5];
    int i;

    for(i=0;i<5;i++){
        printf("\nEnter Enrollment Name Percentage: ");
        scanf("%d %s %f",&s1[i].eno,s1[i].name,&s1[i].per);
    }

    for(i=0;i<5;i++){
        printf("\n Enrollment : %d",s1[i].eno);
        printf("\t Name : %s",s1[i].name);
        printf("\t Percentage : %f",s1[i].per);
    }

}
```

Q : 14 Explain the importance of file handling in C. Discuss how to perform file operations like opening, closing, reading, and writing files.

Write a C program to create a file, write a string into it, close the file, then open the file again to read and display its contents.

Ans : File handling in C allows a program to store data permanently on secondary and retrieve it later, instead of losing data when the program ends.

C provide file handling through the standard library `<stdio.h>`.

1). Data Permanence

2). Large Data Storage

3). Data Sharing

4). Backup Recovery

- Opening a File (fopen()) :

Syntax :
```
FILE *fp;
fp = fopen(“filename”,”mode”);
```

- Closing a File (fclose()) :

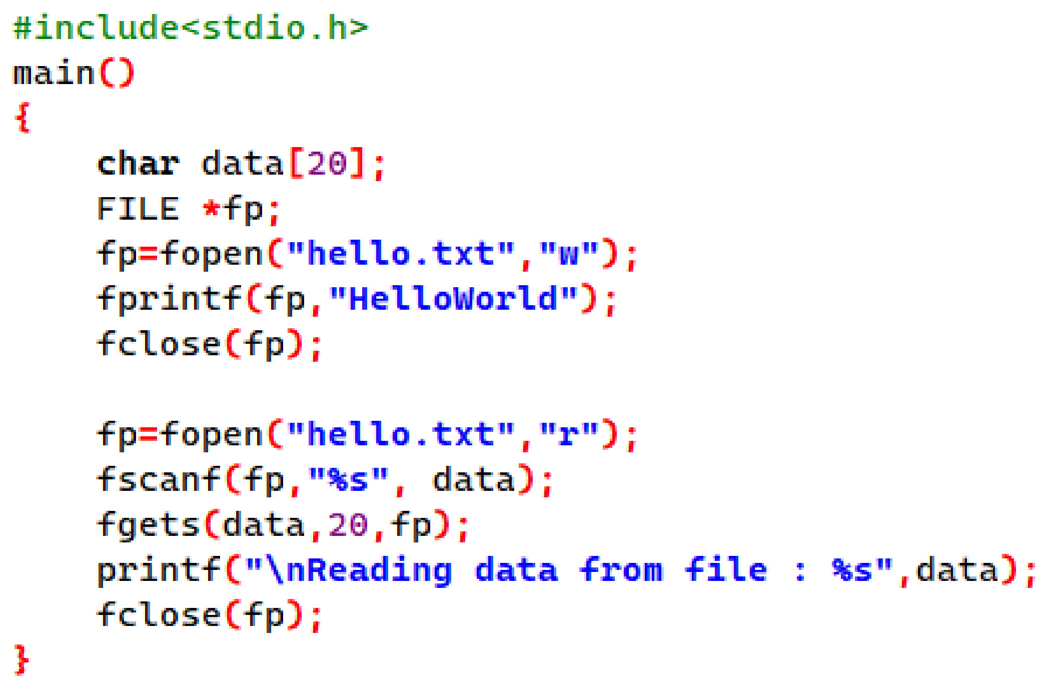
Syntax :
```
FILE *fp
fp = fopen(“filename”,”mode”);
fpclose(fp);
```

- Writing to a File (fclose()) :

Syntax :
```
FILE *fp
fp = fopen(“data.txt”,”w”);
fprintf(fp,”Welcome to file Handling in C”);
fpclose(fp);
```

- Reading a File (fclose()) :

Syntax :
```
FILE *fp
Char str[50];

fp = fopen(“data.txt”,”r”);
fscanf(fp,”%s”,str);
printf(“%s”,str);
fpclose(fp);
```



```
#include<stdio.h>
main()
{
    char data[20];
    FILE *fp;
    fp=fopen("hello.txt","w");
    fprintf(fp,"HelloWorld");
    fclose(fp);

    fp=fopen("hello.txt","r");
    fscanf(fp,"%s", data);
    fgets(data,20,fp);
    printf("\nReading data from file : %s",data);
    fclose(fp);
}
```

Q : 15 Explain the concept of arrays in C. Differentiate between one-dimensional and multi-dimensional arrays with examples.

Write a C program that stores 5 integers in a one-dimensional array and prints them. Extend this to handle a two-dimensional array (3x3 matrix) and calculate the sum of all elements.

Ans : An array in C is a collection of elements of the same data type stored in contiguous memory locations

and accessed using a common name with an index.

- 1-D Array :

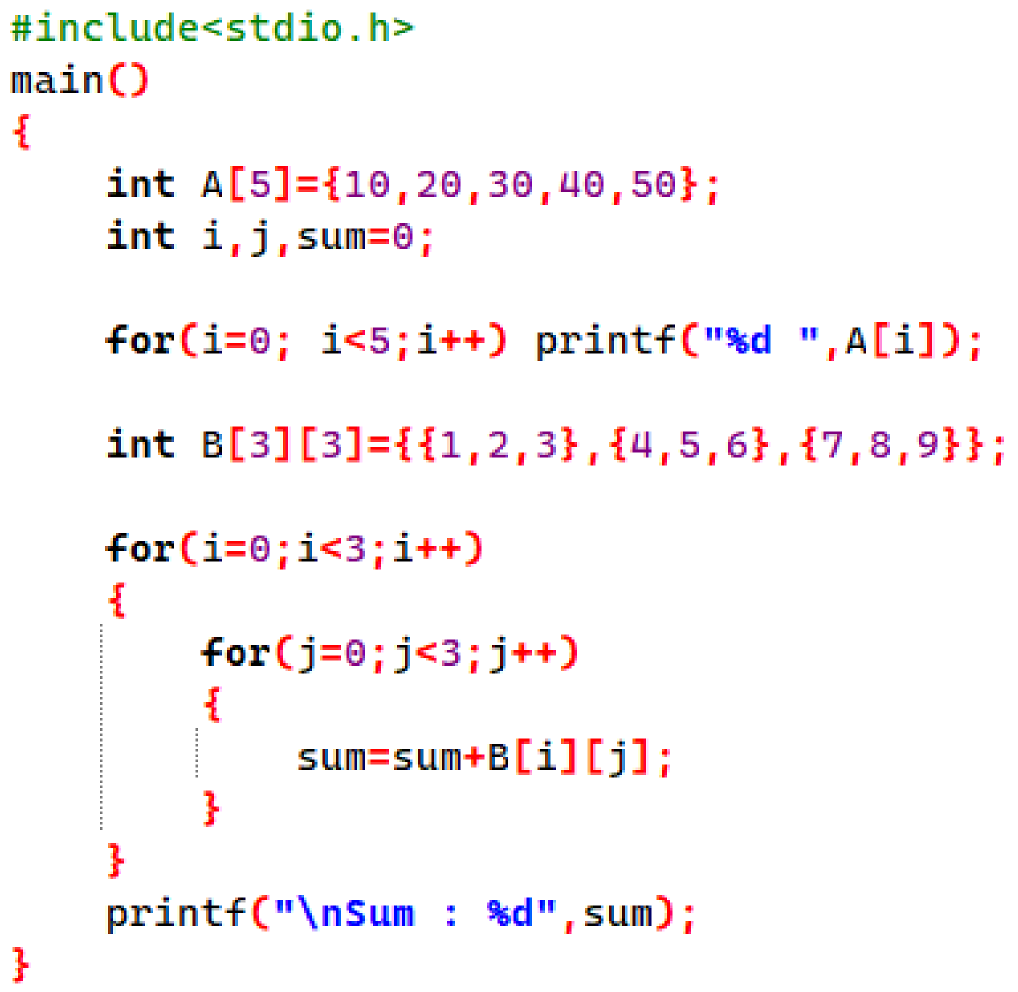
  A 1D array is like a list or linear sequence of elements.

  Ex:`int marks[5]={70,80,90,85,75};`

- 2-D Array :

  A 2D array contains more than one dimension. Most common is the 2D array, which works like a table (rows × columns).

  Ex:`int marks[2][3]={{1,2,3},{4,5,6}};`



```c
#include<stdio.h>
main()
{
    int A[5]={10,20,30,40,50};
    int i,j,sum=0;

    for(i=0; i<5;i++) printf("%d ",A[i]);

    int B[3][3]={{1,2,3},{4,5,6},{7,8,9}};

    for(i=0;i<3;i++)
    {
        for(j=0;j<3;j++)
        {
            sum=sum+B[i][j];
        }
    }
    printf("\nSum : %d",sum);
}
```