Lecture - 3

C Programming

Overview

- Operators
- Decision Making
  - If Statement
  - If else Statement
  - Nested if Statement
  - Switch Statement
  - The ? Operator
  - Break Statement
  - Continue Statement
  - Goto Statement

Operators

- An **operator** is a symbol that tells the compiler to perform specific mathematical or logical functions.
- By definition, an **operator** performs a certain operation on operands. An operator needs one or more operands for the operation to be performed.
- Types of Operators
  - Unary operators – ++ (increment), -- (decrement), ! (NOT), ~ (compliment), & (address of), * (dereference)
  - Arithmetic Operators
  - Relational Operators
  - Logical Operators
  - Assignment Operators
  - Etc.

Relational Operators

Operator	Description	Example
==	Checks if the values of two operands are equal or not. If yes, then the condition becomes true.	(A == B)
!=	Checks if the values of two operands are equal or not. If the values are not equal, then the condition becomes true.	(A != B)
>	Checks if the value of left operand is greater than the value of right operand. If yes, then the condition becomes true.	(A > B)
<	Checks if the value of left operand is less than the value of right operand. If yes, then the condition becomes true.	(A < B)
>=	Checks if the value of left operand is greater than or equal to the value of right operand. If yes, then the condition becomes true.	(A >= B)
<=	Checks if the value of left operand is less than or equal to the value of right operand. If yes, then the condition becomes true.	(A <= B)

Assignment Operators

Operator	Description	Example
=	Simple assignment operator. Assigns values from right side operands to left side operand	C = A + B will assign the value of A + B to C
+=	Add AND assignment operator. It adds the right operand to the left operand and assign the result to the left operand.	C += A is equivalent to C = C + A
-=	Subtract AND assignment operator. It subtracts the right operand from the left operand and assigns the result to the left operand.	C -= A is equivalent to C = C - A
*=	Multiply AND assignment operator. It multiplies the right operand with the left operand and assigns the result to the left operand.	C *= A is equivalent to C = C * A
/=	Divide AND assignment operator. It divides the left operand with the right operand and assigns the result to the left operand.	C /= A is equivalent to C = C / A
%=	Modulus AND assignment operator. It takes modulus using two operands and assigns the result to the left operand.	C %= A is equivalent to C = C % A

Logical Operators

Operator	Description	Example
&&	Called Logical AND operator. If both the operands are non-zero, then the condition becomes true.	(A && B)
||	Called Logical OR Operator. If any of the two operands is non-zero, then the condition becomes true.	(A || B)
!	Called Logical NOT Operator. It is used to reverse the logical state of its operand. If a condition is true, then Logical NOT operator will make it false.	!(A)

Sizeof Operators

Code

```c
#include <stdio.h>

int main(){

   int a = 16;

   printf("Size of variable a: %d \n",sizeof(a));
   printf("Size of int data type: %d \n",sizeof(int));
   printf("Size of char data type: %d \n",sizeof(char));
   printf("Size of float data type: %d \n",sizeof(float));
   printf("Size of double data type: %d \n",sizeof(double));

   return 0;
}
```

Output

```
Size of variable a: 4
Size of int data type: 4
Size of char data type: 1
Size of float data type: 4
Size of double data type: 8
```

Decision Making

- Every programming language including C has decision-making statements to support conditional logic. C has a number of alternatives to add decision-making in the code.

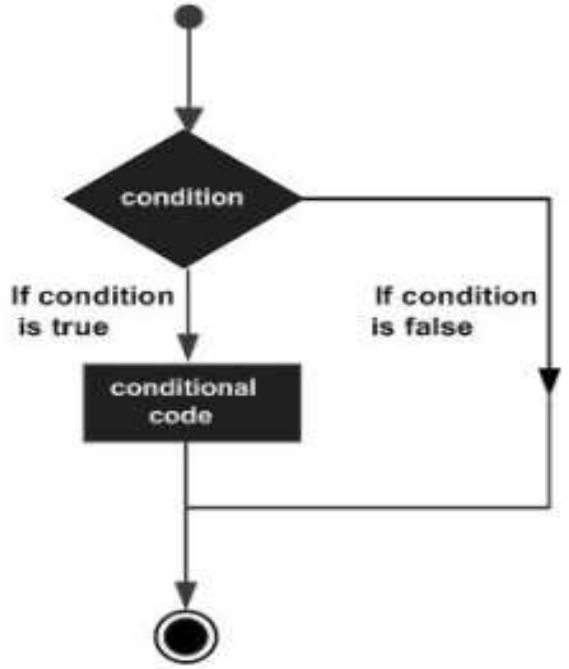
- keywords and operators used in the decision-making statements of a C program − if, else, switch, case, default, goto, the ?: operator, break, and continue

Decision Making

Sr.No.	Statement & Description
1	**if statement** An **if** statement consists of a boolean expression followed by one or more statements.
2	**if...else statement** An **if** statement can be followed by an optional **else** statement, which executes when the Boolean expression is false.
3	**nested if statements** You can use one **if** or **else-if** statement inside another **if** or **else-if** statement(s).
4	**switch statement** A **switch** statement allows a variable to be tested for equality against a list of values.
5	**nested switch statements** You can use one **switch** statement inside another **switch** statement(s).

If Statement

- The if statement is used for deciding between two paths based on a True or False outcome.

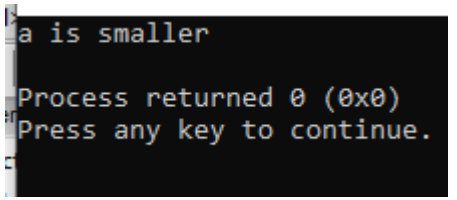


If Statement - Example

Code

```c
#include<stdio.h>

int main()
{
    int a=10,b=20;

    if(a<b)
        printf("a is smaller\n");
}
```

Output

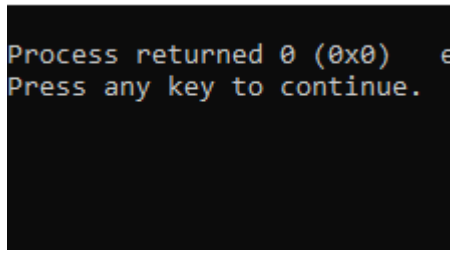


If Statement - Example

Code

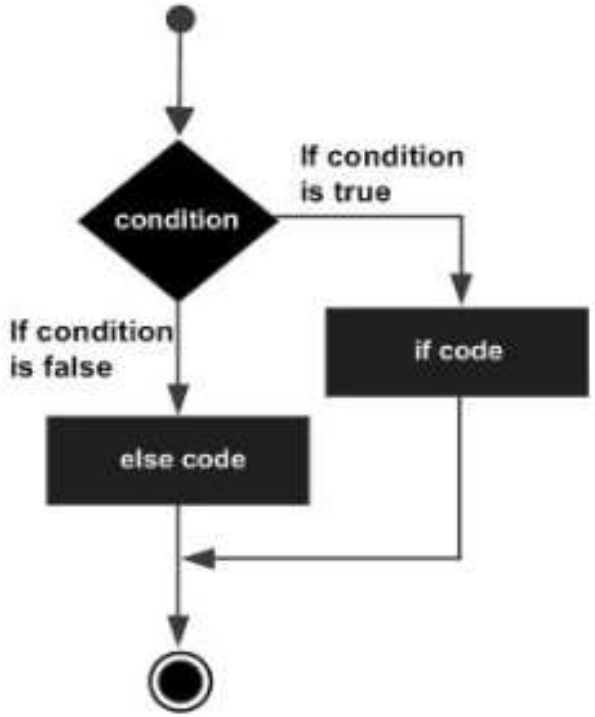
```c
#include<stdio.h>

int main()
{
    int a=30,b=20;

    if(a<b)
        printf("a is smaller\n");
}
```

Output



If else

- he if–else statement offers an alternative path when the condition isn't met.

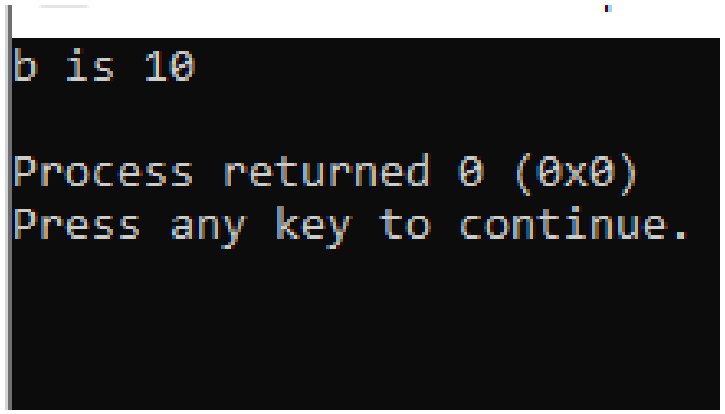


If else example

Code

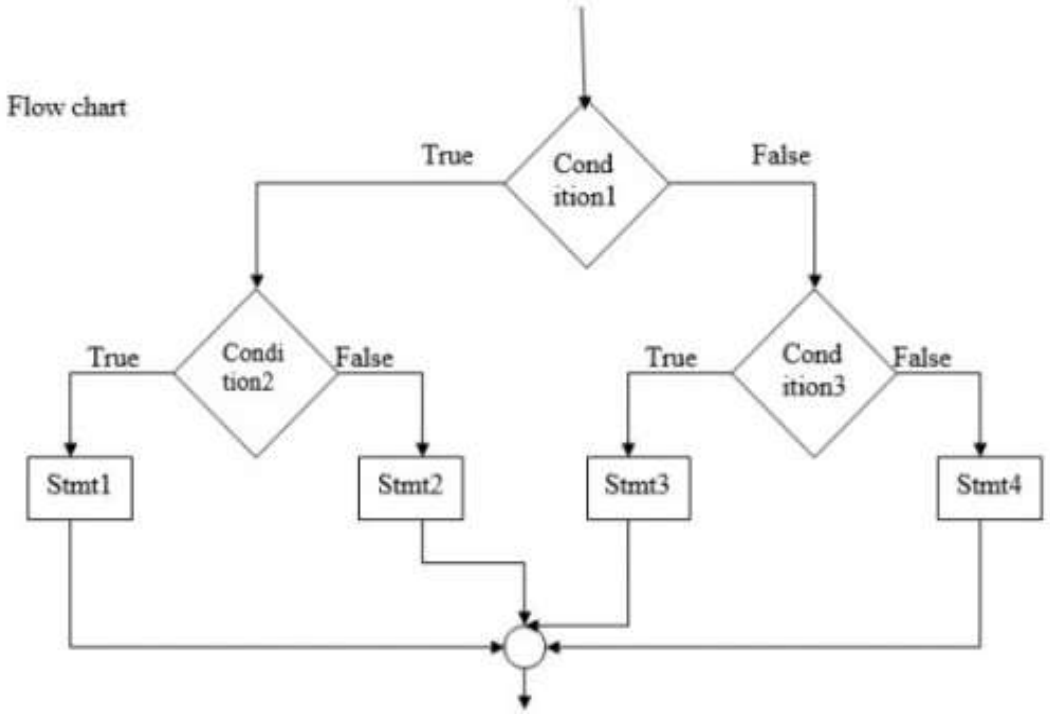
```c
#include<stdio.h>

int main()
{
    int a,b=10;
    if(a==10)
        printf("a is 10\n");
    else if (b==10)
        printf("b is 10\n");
    else if(a==10 && b==10)
            printf("a and b both are 10\n");
    else printf("neither a nor b is 10\n");
    return 0;
}
```

Output



Nested if

- Nested if statements are required to build intricate decision trees, evaluating multiple nested conditions for nuanced program flow.



Nested if Example
(Problem Statement)

- Write a C program that takes a student's score as input and prints the corresponding grade. The grading criteria are as follows:
  - Score >= 90: Grade A
  - 80 <= Score < 90: Grade B
  - 70 <= Score < 80: Grade C
  - 60 <= Score < 70: Grade D
  - Score < 60: Grade F

Nested if Example (1/2)

Code

```c
int main() {
    int score;
    printf("Enter the student's score: ");
    scanf("%d", &score);


    if (score >= 90) {
        printf("Grade: A\n");
    } else {
        if (score >= 80) {
            printf("Grade: B\n");
        } else {
            if (score >= 70) {
                printf("Grade: C\n");
            } else {
                if (score >= 60) {
                    printf("Grade: D\n");
                } else {
                    printf("Grade: F\n");
                }
            }
        }
    }

    return 0;
}
```

Output

```
Enter the student's score: 82
Grade: B

Process returned 0 (0x0)   exe
Press any key to continue.
```

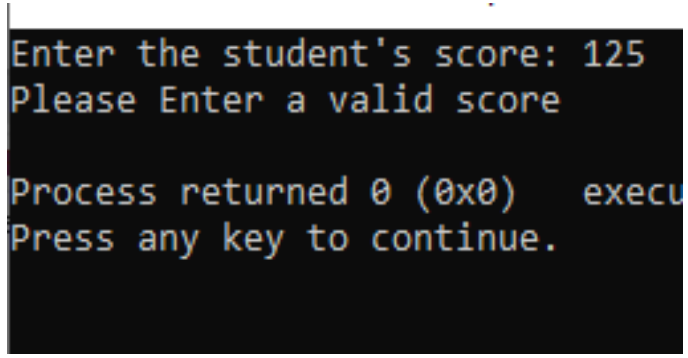
Nested if Example (2/2)

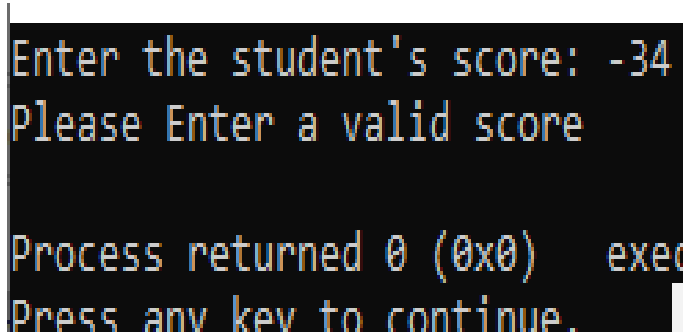
(Comprehensive Code of the Previous Example)

Code

```c
#include <stdio.h>

int main() {
    int score;
    printf("Enter the student's score: ");
    scanf("%d", &score);

    if(score>100 || score<0)
    {
       printf("Please Enter a valid score\n");

    }

   else if (score >= 90) {
        printf("Grade: A\n");
    }
    else {
        if (score >= 80)
        {
            printf("Grade: B\n");
        }
        else
        {
            if (score >= 70)
            {
                printf("Grade: C\n");
            }
            else {
                if (score >= 60)
                {
                    printf("Grade: D\n");
                }
                else
                {
                    printf("Grade: F\n");
                }
            }
        }
    }

    return 0;
}
```

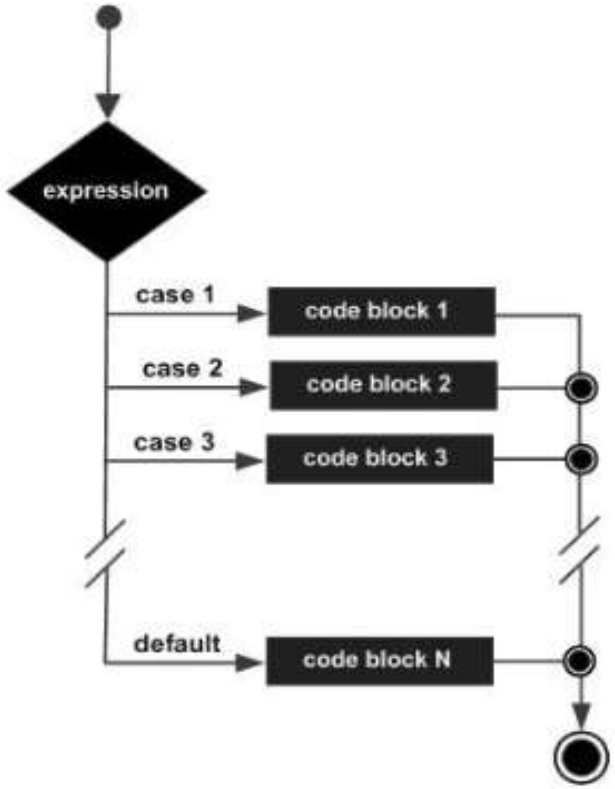
Output





Switch Statement

- A switch statement simplifies multi-way choices by evaluating a single variable against multiple values, executing specific code based on the match. It allows a variable to be tested for equality against a list of values.



Switch- Example

Code

```c
#include <stdio.h>
int main() {
    int day;
    printf("Enter a number (1 to 3) to display the corresponding day: ");
    scanf("%d", &day);
    switch (day) {
        case 1:
            printf("Monday. Welcome to your Bengali class.\n");
            break;
        case 2:
            printf("Tuesday. Welcome to your English class.\\n");
            break;
        case 3:
            printf("Thursday. Welcome to your Arabic class.\\n");
            break;

        default:
            printf("Please enter a number between 1 and 3.\n");
            break;
    }

    return 0;
}
```

Output

```
Enter a number (1 to 3) to display the corresponding day: 2
Tuesday. Welcome to your English class.\n
Process returned 0 (0x0)   execution time : 1.191 s
Press any key to continue.
```

The ?: Operator in C Programming

- can be used to replace if-else statements. It condenses an if-else statement into a single expression, offering compact and readable code.

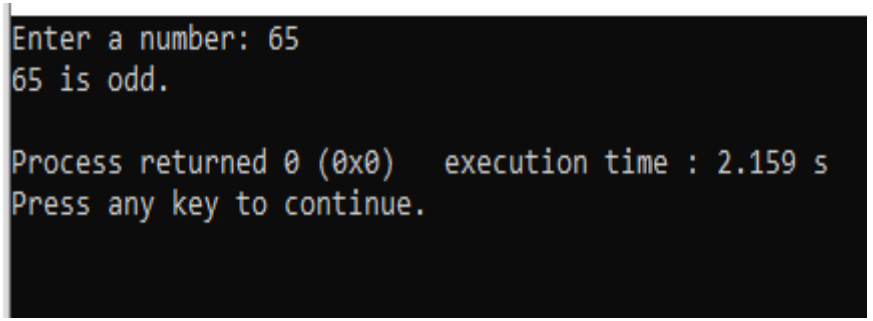
- Format: Exp1 ? Exp2 : Exp3;
  - Where Exp1, Exp2, and Exp3 are expressions. Notice the use and placement of the colon (:). The value of a "?" expression is determined like this −
    - Exp1 is evaluated. If it is true, then Exp2 is evaluated and becomes the value of the entire ? expression.
    - If Exp1 is false, then Exp3 is evaluated and its value becomes the value of the : expression.

The ?: Operator in C Programming- Example

Code

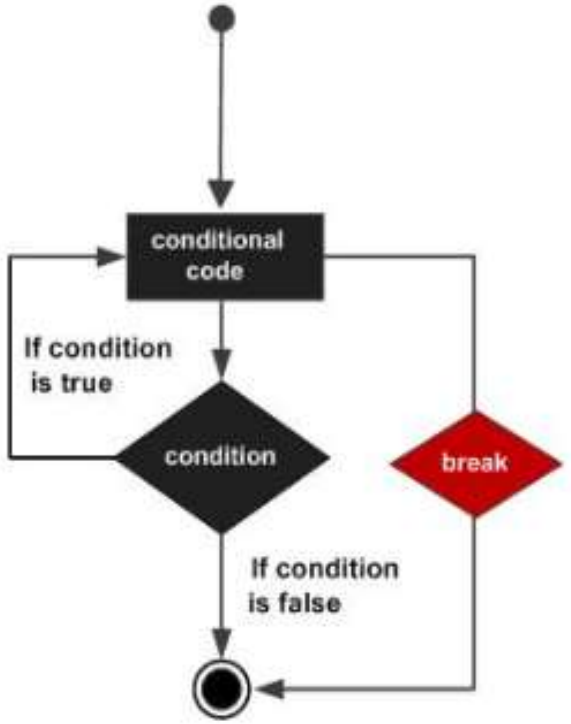
```c
#include <stdio.h>

int main() {
    int num;

    printf("Enter a number: ");
    scanf("%d", &num);

    (num % 2 == 0) ? printf("%d is even.\n", num) : printf("%d is odd.\n", num);

    return 0;
}
```

Output



Break Statement

- In C, the break statement is used in switch–case constructs as well as in loops.
  - When used inside a loop, it causes the repetition to be abandoned.

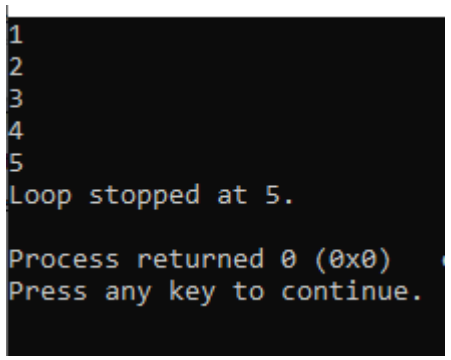


Break- Example

Code

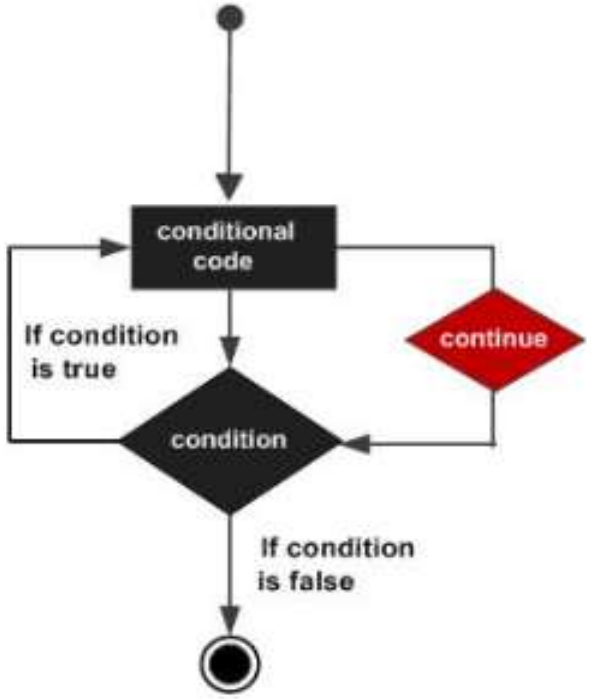
```c
#include <stdio.h>

int main() {
    int i;
    for (i = 1; i <= 10; i++) {
        printf("%d\n", i);

        // Break the loop if i equals 5
        if (i == 5) {
            printf("Loop stopped at %d.\n", i);
            break;
        }
    }

    return 0;
}
```

Output



Continue Statement

- In C, the continue statement causes the conditional test and increment portions of the loop to execute.

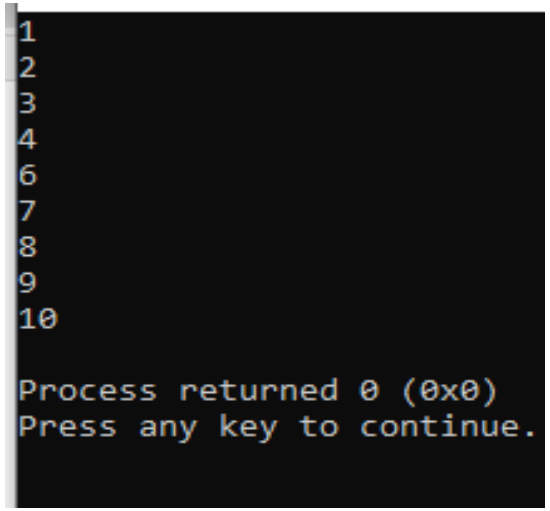


Continue- Example

Code

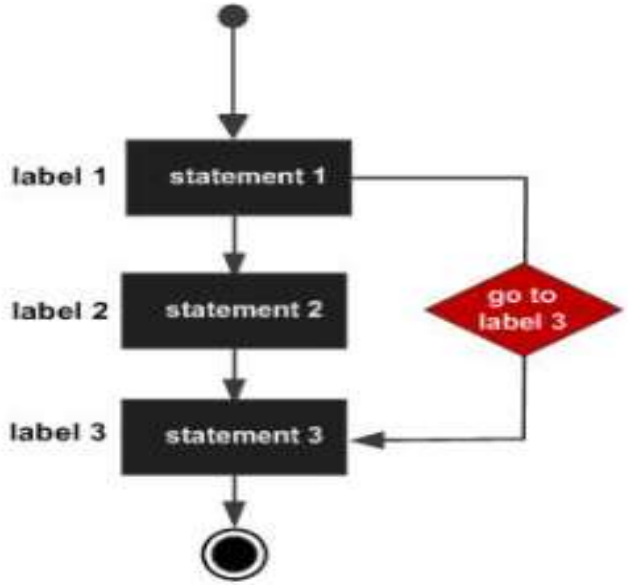
```c
#include <stdio.h>

int main() {
    int i;

    for (i = 1; i <= 10; i++) {

        if (i == 5 ) {
            continue;
        }

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

    return 0;
}
```

Output



Goto Statement

- The goto statement in C enables you to jump to any labeled section of the code. A label is defined by a name followed by a colon (:), and the goto statement transfers control to this label when executed.



Goto- Example

Code

```c
#include <stdio.h>

int main() {
    int num;

    printf("Enter a positive number: ");
    scanf("%d", &num);

    if (num < 0) {
        goto error; // Jump to the 'error' label if the number is negative
    }

    printf("You entered a positive number: %d\n", num);
    return 0;

error: // Label to jump to in case of an error
    printf("Error: You entered a negative number!\n");
    return 1;
}
```

Output

```
Enter a positive number: 5
You entered a positive number: 5

Process returned 0 (0x0)   execution time : 3.876 s
Press any key to continue.
```

```
Enter a positive number: -23
Error: You entered a negative number!

Process returned 1 (0x1)   execution time
Press any key to continue.
```

Reference

- Geeksforgeeks
- GrayCode
- Programiz.com
- TutorialsPoint

Thank You