Contents

Unit 5	**Conditional and Control Statements**	**Page No.**
5.0.	Structure of the Unit	1
5.1.	Unit Outcomes	1
5.2.	Conditional Branching Statements: if statement, if then else statement, nested if, switch, loops: for, while, and do-while loop	1
5.3.	jump statement: break, continue, goto, and return	7
5.4.	Self-Assessment Questions and Activities	10
5.5.	Multiple Choice Questions (MCQs)	10
5.6.	Keys to MCQs	12
5.7.	Summary of the Unit	13
5.8.	Keywords	13
5.9.	Recommended Learning Resources	13
5.10.		
5.11.		

Unit 5
Conditional and Control Statements

5.0. Structure of the Unit

5.1. Unit Outcomes
5.2. Conditional Branching Statements:
5.3. jump statement
5.4. break and continue
5.5. goto and return statement
5.6. Self-Assessment Questions and activities
5.7. Multiple Choice Questions (MCQs)
5.8. Keys to MCQs
5.9. Summary of the Unit
5.10 Keywords
5.11 Recommended Reading

5.1. Unit Outcomes

After the successful completion of this unit, the student will be able to:

1. Explain the conditional statements.
2. Describe the examples using multiple and nested conditions.
3. Discuss the for, while, and do-while loops
4. Develop programs using conditional and control statements.

5.2. Conditional Branching Statements

The stored data is managed by functions using sequential, control structure, or looping statements. To achieve an accurate, error-resistant, and maintainable code, there are control structures in C as given below.

1. Sequence structures
2. Branching structures used for selection
3. Looping structures for repetition or iteration

These structures are given in Figure 5.1.

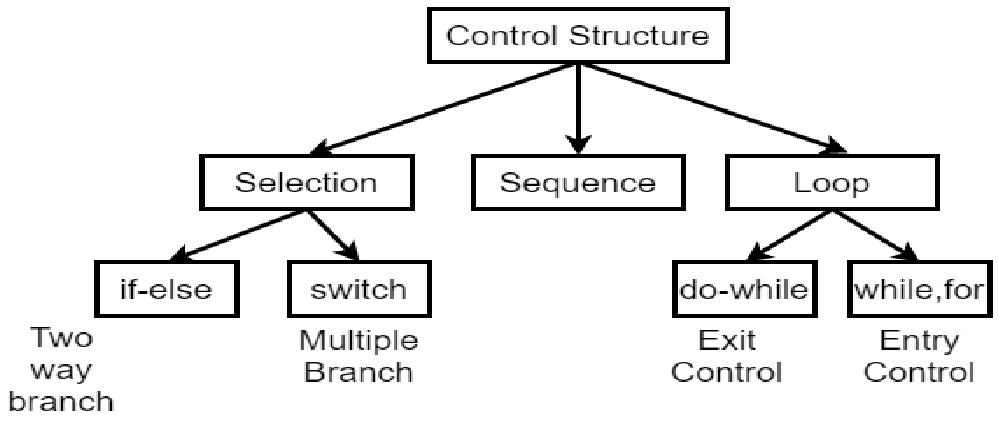

Figure 5.1 Control Structures in C

C programs are coded using the three logical structures shown in the diagram. When more than one control structure is used in a program, then it is known as structured programming. Structured programming is one of the important features used in software development or coding.

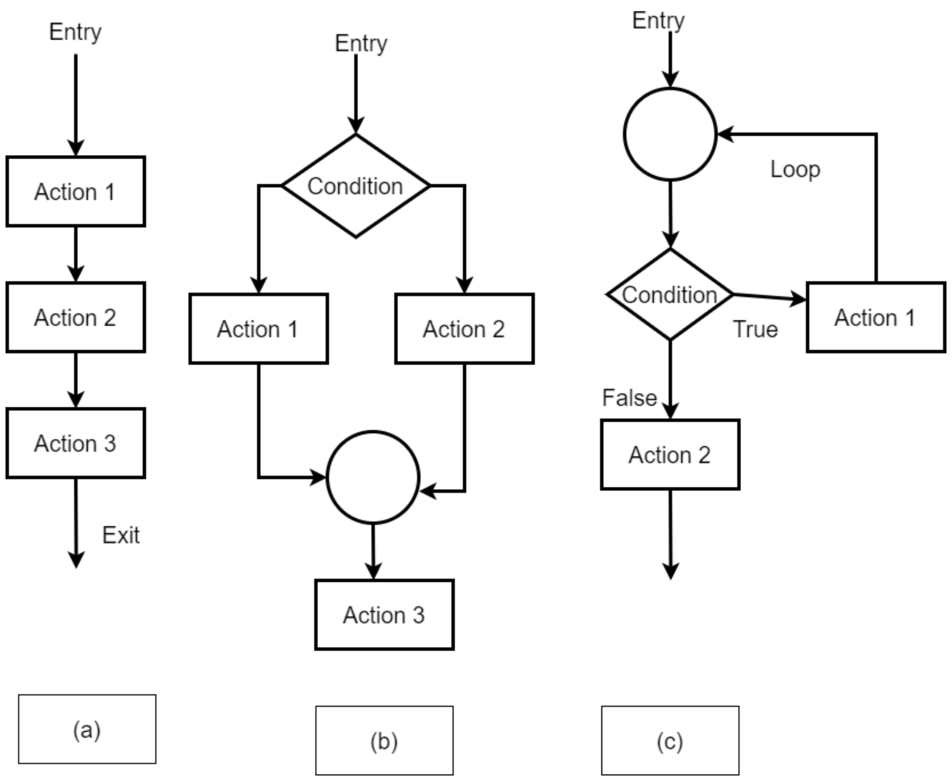

Figure 5.2: Control Structures Used in C

The general format for the control structures if, is-else, switch, while, do-while, and for loops are given below. The control flow structures can be expressed as shown in Figure 5.2.

a) Simple if, if-else statement, and switch statement
b) while and do-while
c) For loop

a) Simple if, if-else, and switch statement control structures with a sample example are shown in Table 5.1.

Table 5.1 if, if-else and switch general format

simple if statement	if-else statement	switch statement
if(expression is true) { action 1; action 2; } action 3; action 4;	if(expression is true) { action 1; action 2; } else { action 3; } Action 4;	switch(expression) { case 1: {action 1; } case 2: {action 2; } case 3: {action 3; } default: {action 4;} } action 5;
Example: `if(a<0) printf("number is negative");`	Example: `if(a%2==0) printf("number is even"); else printf("number is odd");`	`switch(choice) { case 1: printf("one"); break; case 2: printf("two"); break; case 3: printf("three"); break; default: printf("invalid choice"); }`

b) The do while and while statements are the control structures used for looping purposes. The do-while loop is known as an exit-controlled loop as it executes the statements in the loop first and then checks the condition. The while loop is called an entry-controlled loop as the condition is checked first and then the number of statements is executed. The details are shown in Table 5.2.

Table 5.2 while and do-while loops

do { action 1; action 2; } while(expression);	`i = 0; do { printf("print in loop"); i = i+1; } while(i<10);`	Output Description: In this program, the statement printed in the loop will be printed 10 times
while(expression) { Action 1;	`i = 0; while(i<10) {`	Output Description: In this program, the statement printed in the

Action 2;
}
	`printf("print in loop");`
	`i = i + 1;`
	`}`
		loop will be printed 10 times

c) The for loop: The for is an entry-controlled loop. It uses the initial value, the condition, and the auto-increment or decrement values. The syntax of the for loop can be given in Table 5.3.

Table 5.3: for Loop

General Syntax:
for (initial value; test condition; increment)
{ action1;
action2;
action3;
}

`int main()`
`{`
` int i,f,n;`
`printf("enter a number");`
`scanf("%d",&n);`
`f = 1;`
`for(i=1;i<=n;i++)`
`f = f * i;`
`printf("the factorial of a number is %d",f);`
`return 0;`
`}`

As given in the program, the variable `n` is the input and `f` is the output. The for loop calculates the multiplication from `1` to `n`. At the end of the for loop, the factorial of a number is calculated. For example, if the `n = 5,` then the for loop will work as shown in Table 5.4.

Table 5.4: Factorial Input/Output

f = 1 when i = 1
f = 1*2 when i = 2
f = 1*2*3 when i = 3
f = 1*2*3*4 when i = 4
f = 1*2*3*4*5 = 120 when i = 5
The for loop terminates when the value of the i exceeds 5

5.2.1 C programs using if and if-else statements

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

```
      printf("Enter an integer: ");
      scanf("%d", &n);

      // true if the number is greater than 0
      if (n > 0) {
          printf("Number %d is greater than 0\n", n);
      }
      return 0;
}
  Enter an integer: 4
Number 4 is greater than 0
```

Program 1: Illustration of if statement

```
  // Check whether an integer is positive or negative or zero

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

      if  (number== 0) {
          printf("%d is a zero.",number);
      }
      if(number<0)
              printf("%d is a positive integer.",number);
      else
               printf(“%d is a negative number”, number);
      }

      return 0;
}
```

Sample Input/Output

```
Enter an integer:4
4 is a positive integer
```

Program 2: Illustration of if-else statement

```
  #include <stdio.h>
  int main() {
      int marks;
      printf("Enter marks: ");
      scanf("%d", &marks);
      if (marks >= 50)
          printf("Student is qualified for the main exam");
      else
          printf("Student is not qualified for the main exam");
      return 0;
}
```

Sample Input/Output

```
Enter marks: 50
The student is qualified for the main exam
```

Program 3: Illustration of if statement

```
  #include <stdio.h>
  int main()
  {
     int num, temp, sum = 0, r;
     printf("Enter an integer\n");
     scanf("%d", &num);
     temp = num;
     while (temp != 0)
     {
        r = temp % 10;
        sum       = sum + r;
        temp         = temp / 10;
     }
     printf("Sum of digits of %d = %d\n", num, sum);
     return 0;
}
```
Sample Input/Output
```
  Enter an integer
  123
The sum of digits of 123 = 6
```
Program 4: Illustration of while loop

```
  #include <stdio.h>
  int main()
  {
  int ch;
  printf("Enter your choice");
  scanf("%d",&ch);
  switch(ch)
  {
  case 1:  printf("one");break;
  case 2:  printf("Two");break;
  case 3:  printf("Three");break;
  default: printf("Default Choice");
  }
  return 0;
}
```
Sample Input/Output
```
  Enter your choice 4
   Default Choice
```
Program 5: Illustration of switch

```
  include <stdio.h>
      int main() {
          char letter  = 'a';
          switch (letter) {
              case 'a': printf("Value is a");
                        break;
              case 'e': printf("Value is e");
                        break;
              case 'i': printf("Value is i");
                        break;
              case 'o': printf("Value is o");
                        break;
              case 'u': printf("Value is u");
                        break;
              default:
                  printf("Consonants");
                  break;
          }
          return 0;
    }
Sample Input/Output
Value is a
```

```
  #include <stdio.h>
  int main()
  {
     int i=0;
     do
     {
      printf("%d\n",i);
      i = i + 1;
     }while(i<=5);
     printf("Out of loop");
}
Sample Input/Output
  0
  1
  2
  3
  4
  5
Out of loop
```

5.3. Jump Statement

In C language, when the flow of the program is changed from one statement to another, the jump is used. There are four types of jump statements.
break: This statement is used to break the loop, or a block of statements without executing the remaining statements. It is used inside for, while, or do-while loops and switch statements. The general syntax and examples of the break are as follows.

```
break;
```

```
  // C program to illustrate the break in c loop
  #include <stdio.h>
  int main()
  {
        int x,n,i;
        printf("Enter the maximum limit");
        scanf("%d",&n);
        for (i = 1; i <= n; i++)
        {
            printf("Enter a number:");
            scanf("%d",&x);
            printf("%d\n",x);
            if(x<0)
            {
              printf("\n Program is terminated");
              break;
            }
         }
        return 0;
}
```

Sample Input/Output

```
  Enter the maximum limit 4
  Enter a number:1
  1
  Enter a number:2
  2
  Enter a number:-1
  -1
 Program is terminated
```

Program to Illustrate break statement

In the above program, the numbers are entered and printed till a negative number.

continue: The continue statement is used to skip the current loop cycle and continue with the next one. The cycle is also known as iteration. The syntax of continue is as follows.

```
continue;
```

A sample program using continue is given below.

```
  #include <stdio.h>
  int main()
  {
    int x;
    printf(“enter an integer”);
    scanf(“%d”, &x);
    for (int i = 1; i <= 10; i++) {
    // if i is equal to x , continue to next iteration without
        printing x.
    if (i == x) {
      continue;
    }
    else{
      // otherwise print the value of i.
      printf("%d ", i);
     }
   }
  return 0;
}
```

```
Sample Input/Output
Enter an integer 5
1 2 3 4 6 7 8 9 10
```

Program using continue statement

In the above program, when the value of x = 5, that iteration is skipped, and the loop is continued to the next iteration.

goto statement: The goto statement is used to shift the program control from one point to another specified by a label. The general syntax is given below.

```
  #include <stdio.h>
  int main()
  {
     int MAX = 100;
     int i, num;
     float average, sum = 0.0;
     for (i = 0; i < MAX; i++)
     {
        printf("Enter a number: ");
        scanf("%d", &num);

        // go to jump if the user enters a negative number
        if (num < 0.0)
        {
           goto newplace;
        }
        sum = sum + num;
     }
  newplace:
     average = sum/i;
     printf("Sum = %f\n", sum);
     printf("Average = %f", average);
     return 0;
}
Sample Input/Output
  Enter a number: 1
  Enter a number: 2
  Enter a number: 3
  Enter a number: -1
  Sum = 6.000000
Average = 2.000000
```

Program using goto statement

return: A return statement in C is used to end the normal execution of a program and pass the control to the function. There may or may not be value to be returned. The return data type can be integer, float, or any other type. A return statement returns a single value. A main() program in the above-mentioned programs returns a zero value.

int factorial(int n){
----
return num; ← Called Function
}

```
int main() {
----------------
factorial(n);   <----   Calling Function
------------
}
```

A program with a return statement is shown as follows.

```
  #include <stdio.h>
  int factorial(int n)
  { int i, f=1;
    for(i=1;i<=n;i++)
     f = f*i;
    return f;
  }

  int main(void)
  {
      int n, fact = 0;
      printf("Enter a number");
      scanf("%d",&n);
      fact = factorial(n);
      printf("\n Factorial of %d is %d",n,fact);
      return 0;
}
Sample Input/Output
Enter a number 5
Factorial of 5 is 120
```

Program using return statement

5.4. Self-Assessment Questions

Q1. What is the purpose of printf() and scanf() statements? Explain with an example.
Q2. Write a C program to compute the area and perimeter of a square.
Q3. Write a C program to print even numbers from 1 to n.
Q4. Explain a switch statement with an example.
Q5. Compare while and do-while loops with a suitable example.
Q6. Write a C program to find a factorial of a number using a for loop.
Q7. Explain the syntax of if and if-else with an example.
Q8. How the break and continue statements work in a C program.
Q9. Demonstrate the use of the goto statement.
Q10. Illustrate arithmetic and logical operators.
Q11. Illustrate the use of switch statement for developing a calculator application.

5.5. Multiple-Choice Questions

Q1. What is the output of the following program?

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

```
{
      printf("Value of x is  %d", x);
      return 0;
}
```

A. Value of x is %d
B. Garbage value
C. Compile error
D. 0

Q2. What is the output of the following program?

```
#include <stdio.h>
int main()
{
    int x = 10, y = 20;
    printf("%d", x, y);
    return 0;
}
```

A. 10
B. Error
C. 20
D. 10 20

Q3. What is the output of the following program?

#include <stdio.h>

```
int main()
{
    int x = 20 + 40/2 + 60 * 2;
    printf("%d", x);
    return 0;
}
```

A. 150
B. 160
C. 120
D. Error

Q4. What is the output of the following program?

```
#include <stdio.h>
int main()
{
  int x, y = 10, z = 10;
  x = y == z;
  printf("%d", x);
  return 0;
}
```

A. 10
B. 0
C. 10 10
D. 1

Q5. The entities that remain unchanged are known as_____

A. Functions
B. Tokens
C. Constants
D. Variables

Q6. The number of keywords in C re____

A. 22
B. 32
C. 30
D. 42

Q7. Identify the correct keyword from the following

A. default
B. go to
C. longer
D. compiler

Q8. What is the output of the following code?

```
#include <stdio.h>
int main()
{
  int i, x;
  for(i = 0; i < 5; i++)
 {
   x = i;
  }
  printf("%d", x);
  return 0;
}
```

A. 0
B. 4
C. 5
D. Error

Q9. _____ of the following is not a reserved word

A. int
B. float
C. main
D. while

Q10. The identifier in C cannot begin with____

A. Number
B. Special symbol other than underscore
C. Both a and b
D. An alphabet

5.6. Keys to Multiple-Choice Questions

Q1. Compiler Error (C)

Q2. 10 (A)

Q3. 160 (B)

Q4. 1 (D)

Q5. Constants (C)

Q6. 32 (B)

Q7. default (A)

Q8. 4 (B)

Q9. main (C)

Q10. Special symbol except underscore (B)

5.7. Summary of the Unit

In every C program, the flow control cab changed using if, if-else, switch, and different looping statements. These statements are used to change the execution using conditions. The flow of control can also changed using unconditional statements such as return and goto statements. Each control structure is with a conditional statement or an expression. Based on the result of an expression that is either true or false the rest of the program is executed.

5.8. Keywords:

- if
- if-else
- switch
- for
- while
- do-while
- break
- return
- goto

5.9. Recommended Learning Resources

[1] Herbert Schildt. (2017). *C Programming: The Complete Reference*. 4th ed. USA: McGraw-Hill.