Unit 4

Control Structure/Statement

- Sequential Statement
- Decision/Selection/Conditional Statement
    - if statement
    - if…else statement
    - if…else if…else statement
    - Nested if…else statement
    - Switch statement
- Loop (for, while and do-while)
- Jump statement (break, continue, goto statement)

Sequential Statement

Sequential statements are statements in computer programming that are executed one after the other in the order in which they appear.

Examples of sequential statements include assignments, function calls, and input/output operations.

Control Statement

- In the C programming language, a control statement is a statement that determines the flow of execution of the program.
- In simpler words, the control statements help users specify the order of execution of the instructions present in a program.

There are two types of control statements

1. Decision making statements

- If statement
- If..else statement
- Else..if statement
- Nested if statements
- Switch statement

2. Loop Statements

- While loop
- Do..while loop
- For loop

Decision/selection/conditional statements

Decision making statements test a condition and allow to execute some statements.

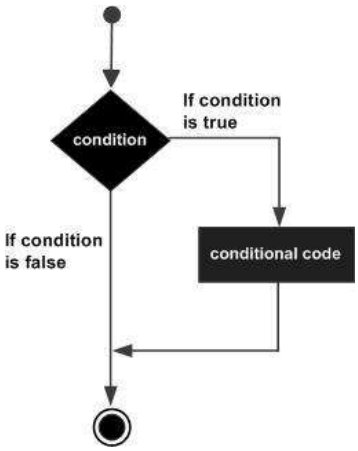
if statement

- In if statement, if the condition is true, then the block of code will be executed.
- If the condition is false, then the block of code will be skipped.

Syntax:

```
if (condition)
{
   // A block of statements
}
```

Flow chart



Example:

```
#include <stdio.h>

int main() {
```

```
  int x = 20;
  int y = 18;
  if (x > y) {
    printf("x is greater than y");
  }
  return 0;
}
```

Example 2: C program to print largest amont 3 numbers

```
#include <stdio.h>
int main()
{
    int num1, num2, num3, largest;

    printf("Enter three numbers: ");
    scanf("%d %d %d", &num1, &num2, &num3);

    // Assuming num1 is the largest
    largest = num1;

    if (num2 > largest)
    {
        largest = num2;
    }

    if (num3 > largest)
    {
        largest = num3;
    }

    printf("The largest number is %d", largest);

    return 0;
}
```

If..else statement

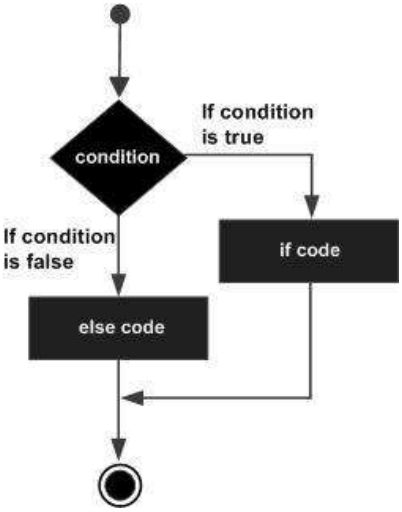
- In if..else statement if the condition is true, then the if block will be executed,
- otherwise, the else block will be executed.

Syntax:

```
if (test expression) {
```

```
    // execute code if the condition is true
}
else {
    // execute code if the condition is false
}
```

Flow chart



Example:

```
// Check whether an integer is odd or even

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

    int number;
    printf("Enter an integer: ");
    scanf("%d", &number);

    // True if the remainder is 0
    if  (number%2 == 0) {
        printf("%d is an even integer.",number);
    }
    else {
        printf("%d is an odd integer.",number);
    }

    return 0;
}
```

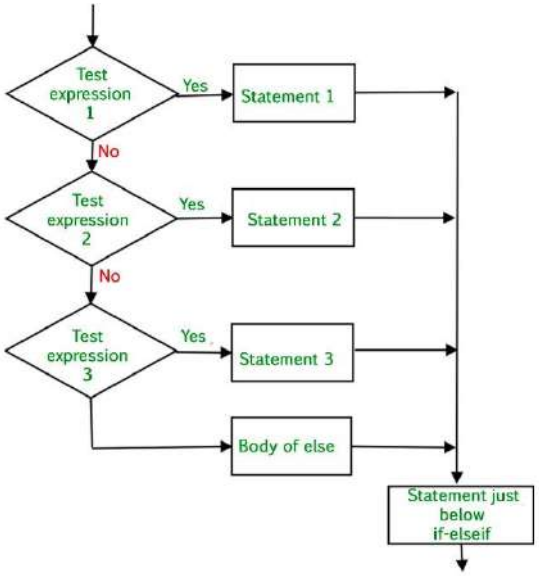
else..if statement

- The else..if statement allows us to check between multiple conditions and execute different statements.
- The conditions are checked from top to bottom.
- As soon as the true condition is found the block is executed.
- If all the conditions become false then final else block gets executed.
  Syntax:

Syntax:

```
if (condition1) {
  // block is executed if condition1 is true
} else if (condition2) {
  // block is executed if the condition1 is false and condition2
is true
} else if(condition3) {
    // block is executed if condition 1 and 2 are false and
condition 3 is true
}else{
  // block is executed if all the conditions are false
}
```

Flow Chart



Example:

```c
#include <stdio.h>
int main()
{
    int percentage;
    printf("Enter your percentage: ");
    scanf("%d", &percentage);
    if (percentage >= 85 && percentage <= 100)
    {
        printf("Congrats ! you scored grade A ...");
    }
    else if (percentage >= 60 && percentage < 85)
    {
        printf("You scored grade B + ...");
    }
    else if (percentage >= 40 && percentage < 60)
    {
        printf("You scored grade B ...");
    }
    else if (percentage >= 30 && percentage < 40)
    {
        printf("You scored grade C ...");
    }
    else
    {
        printf("Sorry you failed ...");
    }

    return 0;
}
```

Nested if statements

Statements (if, if..else, else..if) statements can be written inside the body of if statements.

Simple syntax:

```c
if (condition 1)
{
    if (condition 2)
    {
```

```
        // code is extcuted if conditions 1 and 2 are true
    }
    else
    {
        // code is executed if condition 1 is true and condition
2 is false
    }
}
else
{
    // code is executed if condtion 1 is failed
}
```

Example:

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

    if (marks >= 0 && marks <= 10)
    {
        if (marks >= 2)
        {
            printf("Congratulation: Passed");
        }
        else
        {
            printf("Sorry: Failed");
        }
    }
    else
    {
        printf("Please enter marks between 0-100");
    }

    return 0;
}
```

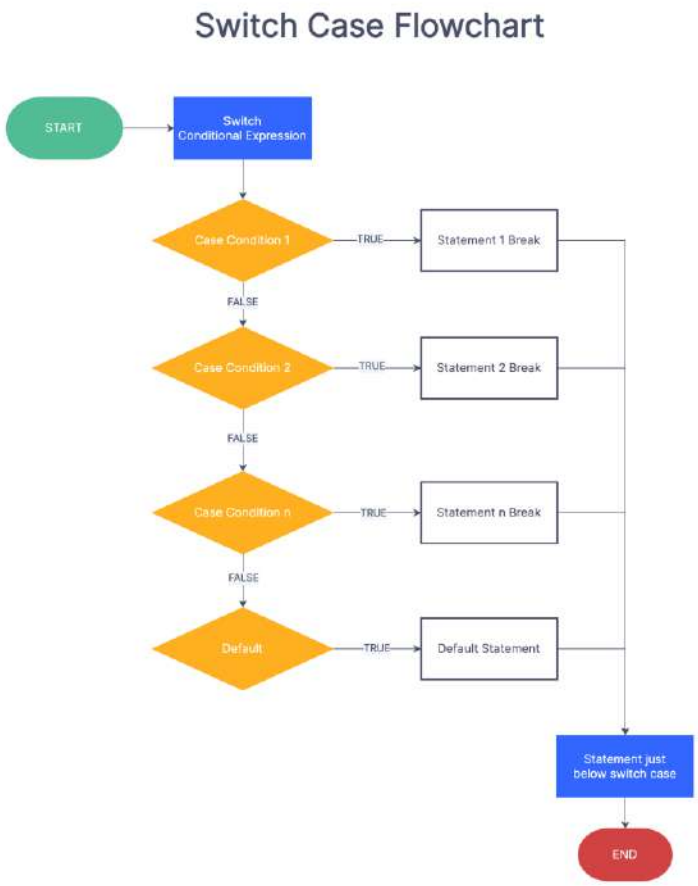
Switch statement:

Switch statements are used when we have multiple possible constant values and based on that values code blocks are executed.

Syntax:

```
switch (expression) {
   case value1:
      // code to be executed if expression matches value1
      break;
   case value2:
      // code to be executed if expression matches value2
      break;
   case value3:
      // code to be executed if expression matches value3
      break;
   // more cases can be added here
   default:
      // code to be executed if expression doesn't match any of the
cases
}
```

Flow chart:



Example: C program to print days of week based on given number between 1 and 7.

```c
#include <stdio.h>

int main()
{
    int day;
    printf("Enter a number between 1 and 7");
    scanf("%d", &day);

    switch (day)
    {
    case 1:
        printf("Sunday");
        break;
    case 2:
        printf("Monday");
        break;
    case 3:
        printf("Tuesday");
        break;
    case 4:
        printf("Wednesday");
        break;
    case 5:
        printf("Thursday");
        break;
    case 6:
        printf("Friday");
        break;
    case 7:
        printf("Saturday");
        break;

    default:
        printf("Invalid Number");
    }

    return 0;
}
```

Above program can also be written like :

```c
#include <stdio.h>

int main()
```

```c
{
    int day;
    printf("Enter a number between 1 and 7: ");
    scanf("%d", &day);

    switch (day)
    {
    case 1: printf("Sunday"); break;
    case 2: printf("Monday"); break;
    case 3: printf("Tuesday"); break;
    case 4: printf("Wednesday");break;
    case 5: printf("Thursday"); break;
    case 6: printf("Friday"); break;
    case 7: printf("Saturday"); break;

    default: printf("Invalid Number");
    }

    return 0;
}
```

Loops in C

- In programming, a loop is used to repeat a block of code until the specified condition is met.
- A loop statement allows programmers to execute a statement or group of statements multiple times without repetition of code.
- These statements are known as looping/Iterative/Repetitive Statements.

C programming has three types of loops:

1. while loop
2. do...while loop
3. for loop

1. while loop

The while loop evaluates the condition first and executes the statements until the condition is false.

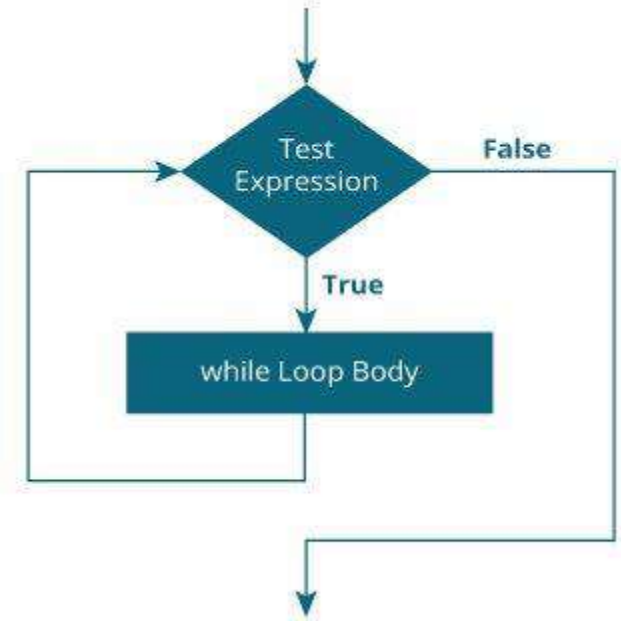
If the conditon is false, the loop terminates (ends).

Syntax:

```c
while (condition) {
  // code block to be executed
```

```
}
```

Flow chart:



Example:

```c
// Print numbers from 1 to 5

#include <stdio.h>
int main()
{
    int i = 1;

    while (i <= 5)
    {
        printf("%d\n", i);
        ++i;
    }

    return 0;
}
```

Example: 2

```c
// Print 10 hello world

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

```c
{
    int i=1;
    while (i <= 5)
    {
        printf("Hello world\n", i);
        i++;
    }

    return 0;
}
```

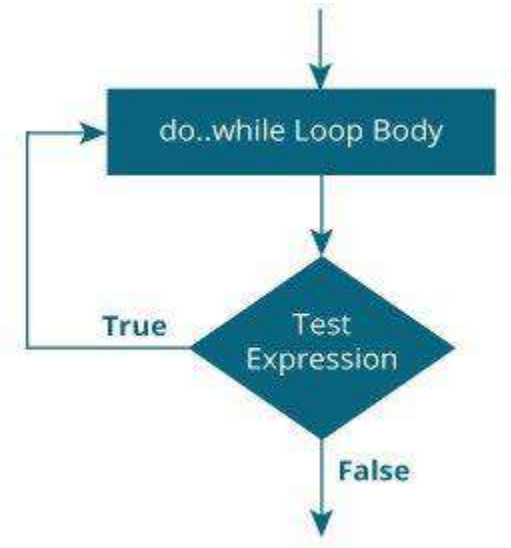
2. do...while loop

- Do.. while loop executes the body first then checks a condition.
- Do..while loop is executed at least once even the condition is false.

Syntax:

```c
do {
    // code block to be executed
} while (condition);
```

Flowchart:



Example:

```c
// Print 10 hello world
```

```c
#include <stdio.h>
int main()
{
    int i = 1;

    do{
        printf("Hello world\n", i);
        i++;
    }
    while (i <= 5);

    return 0;
}
```

Example: 2

```c
// Program to print sum of digits of given number

#include <stdio.h>

int main() {
    int num, i = 1, sum = 0;

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

    do {
        sum += i;
        i++;
    } while (i <= num);

    printf("The sum of all numbers from 1 to %d is: %d", num,
sum);

    return 0;
}
```

3. for loop

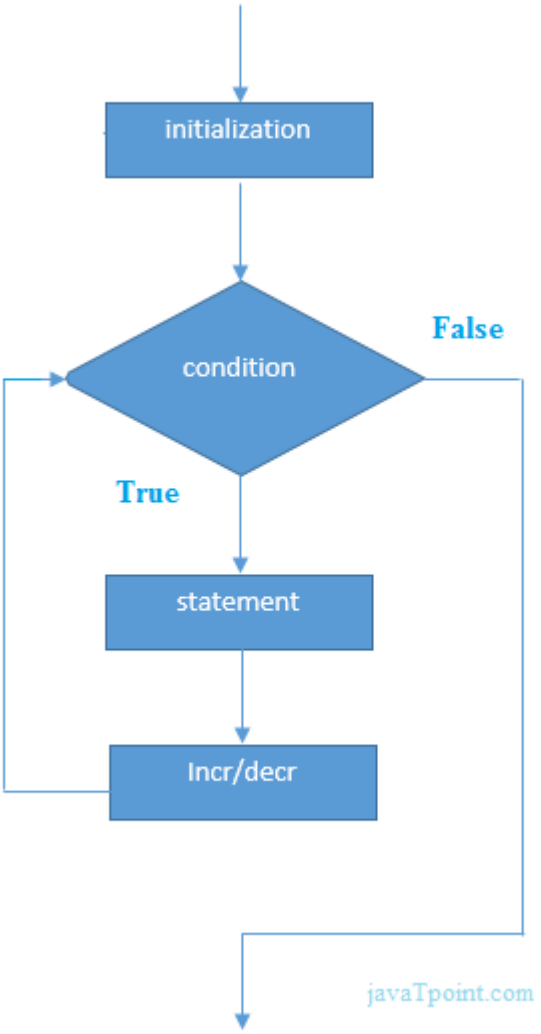
- For loop executes the block of statements for a number of time.
- When the no. of repetition is known in advance then the use of for loop is efficient.

Syntax:

```c
for ( initialization; condition; increment/decrement ) {
```

```
    // statement(s);
}
```

Flowchart:



Example:

```
// Print numbers from 1 to 10
#include <stdio.h>

int main()
{
    int i;
```

```
    for (i = 1; i < 11; ++i)
    {
        printf("%d ", i);
    }
    return 0;
}
```

Example 2:

```
//calculate the factorial of give number
#include <stdio.h>

int main()
{
    int i, n, fact = 1;

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

    for (i = 1; i <= n; i++)
    {
        fact *= i;
    }

    printf("Factorial of %d is %d", n, fact);

    return 0;
}
```

Example 3:

```
//Print multiplication table of given number.

#include <stdio.h>

int main()
{
    int num = 5, i;

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

    printf("Multiplication table of %d:\n", num);

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

```
    {
        printf("%d x %d = %d\n", num, i, num * i);
    }

    return 0;
}
```

Jump statements

In C programming, jump statements are used to transfer program control to a different location within the code. There are three types of jump statements in C:

a) Break
b) Continue
c) and goto.

a. break statement:

The break statement terminates the loop immediately when it is encountered.

Example:

```
#include <stdio.h>

int main()
{
    int i;

    for (i = 0; i < 10; i++)
    {
        if (i == 4)
        {
            break;
        }
        printf("%d\n", i);
    }

    return 0;
}
```

b. Continue statement

- The continue statement skips the current iteration of the loop and continues with the next iteration.
- It doesn't terminate the loop.

Example:

```c
#include <stdio.h>

int main()
{
    int i;

    for (i = 0; i < 10; i++)
    {
        if (i == 4)
        {
            continue;
        }
        printf("%d\n", i);
    }

    return 0;
}
```

c. goto statement

The goto statement allows us to transfer control of the program to the specified label.

```c
/*To print numbers from 1 to 10 using goto statement*/
#include <stdio.h>
int main()
{
    int number;
    number = 1;

repeat:
    printf("%d\n", number);
    number++;

    if (number <= 10)
        goto repeat;

    return 0;
```

```
}
```

Assignment:

Difference between else..if and switch statement.

Difference between while and do..while statement.