

CS-1004
Programming
Fundamentals

LAB - 05
Basic Decision Structure

NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES
Fall 2021

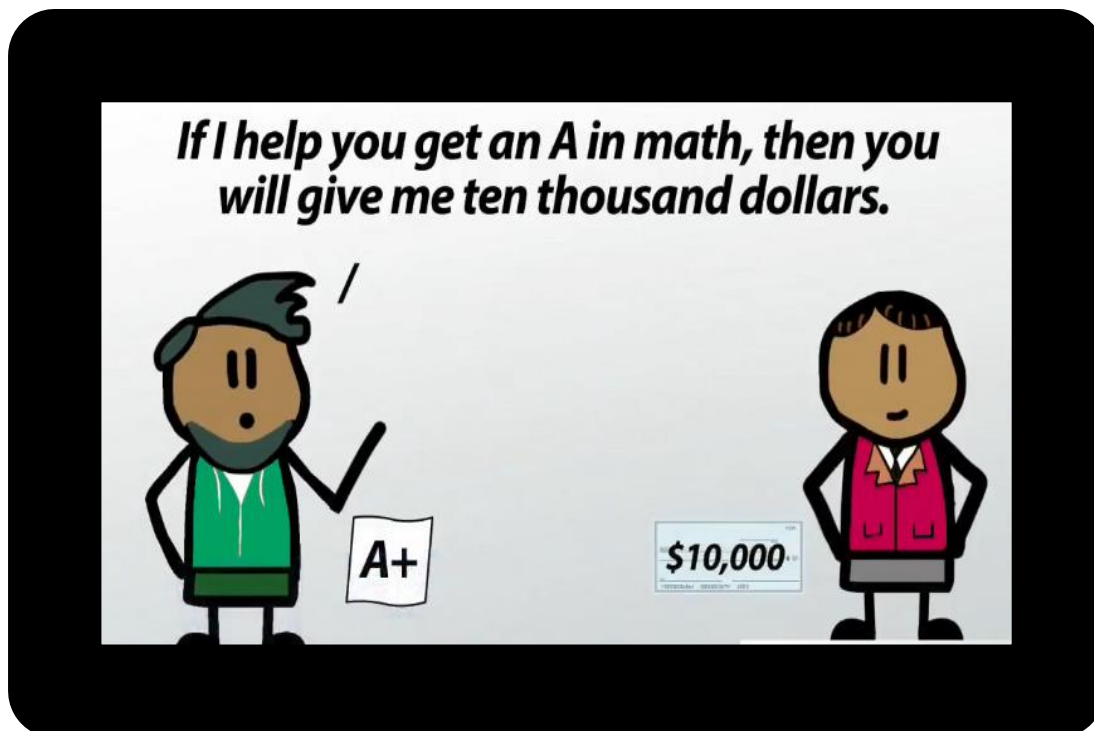
NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES

Learning Objectives

- ❑ Introduction to conditional statements
- ❑ If structure
- ❑ If –else structure
- ❑ If-else-if structure
- ❑ Switch statements

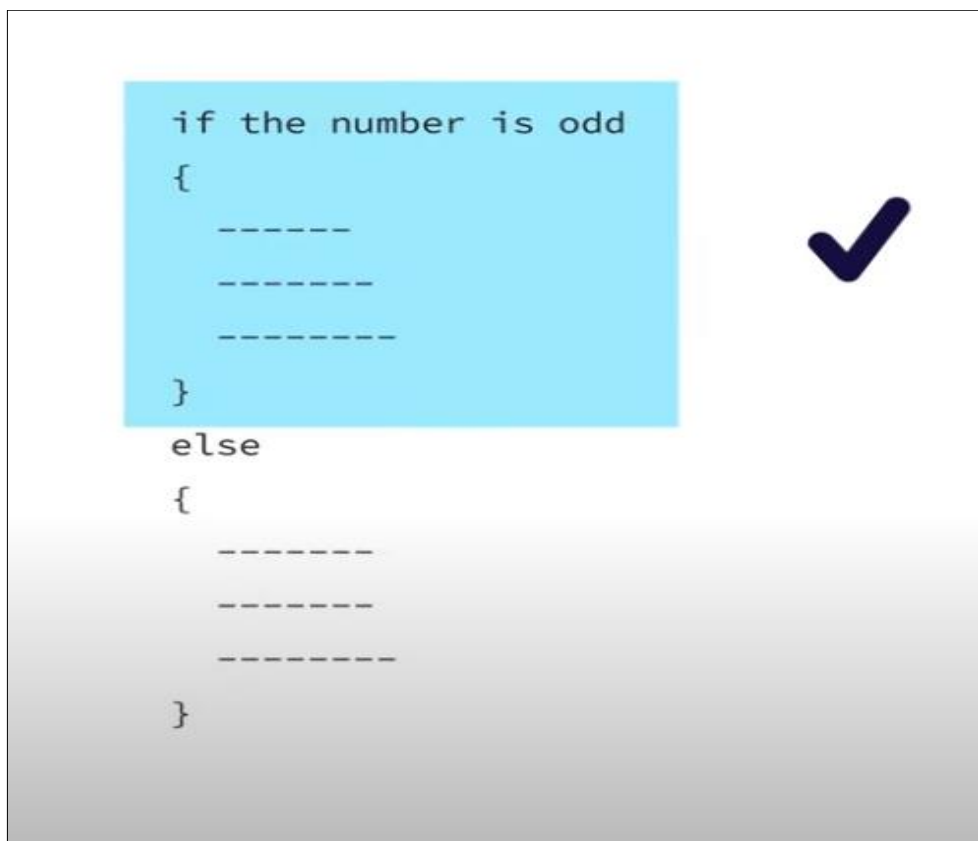
Conditional Statements

In C programming there are decision making statements, we need this kind of statements because while programming we often need to make a lot of decisions. Like shown below in the pictures.

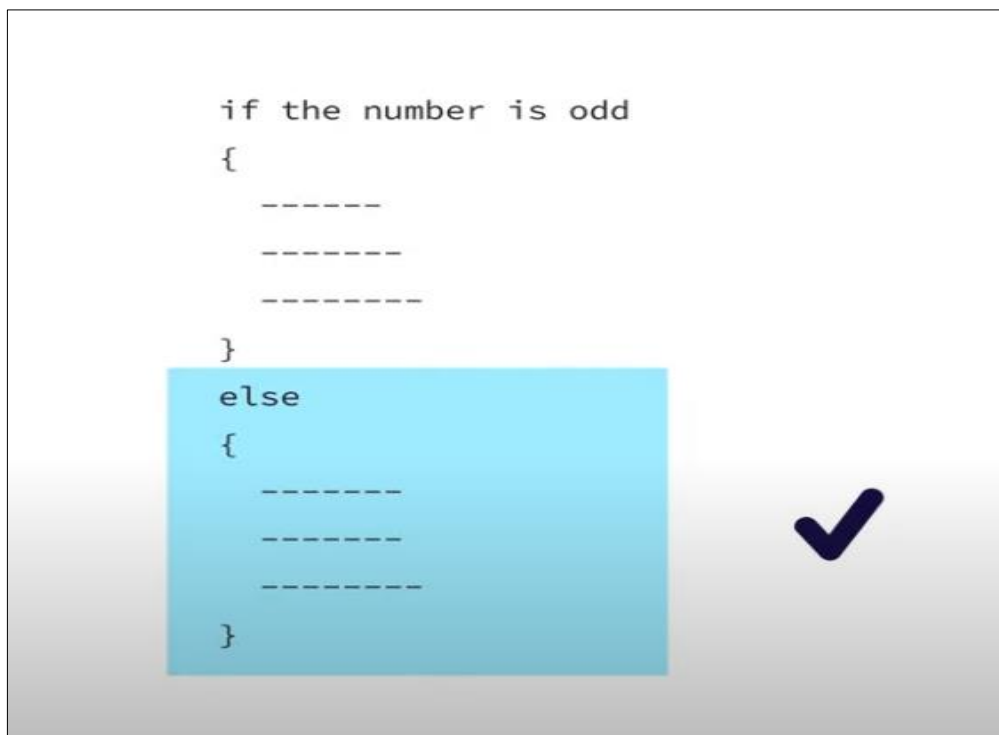




In some cases we need to execute this block of code



Otherwise we want to execute this block



In C, an if/else statement specifies that one block of code should be executed if a condition is true, and another block should be executed if that condition is false.

To write meaningful if/else statements, it is important to know operators which allow us to compare two expressions and produce a Boolean outcome.

In C, however, there are no distinct values for true or false, instead, false is 0, and anything which is non-zero is true. We will refer to true and false because they make more sense conceptually; the distinction should not make a practical difference in most cases.

expr1 == expr2	tests if expr1 is equal to expr2
expr1 != expr2	tests if expr1 is not equal to expr2
expr1 < expr2	tests if expr1 is less than expr2
expr1 <= expr2	tests if expr1 is less than or equal to expr2
expr1 > expr2	tests if expr1 is greater than expr2
expr1 >= expr2	tests if expr1 is greater than or equal to expr2
!expr	computes the logical NOT of expr
expr1 && expr2	computes the logical AND of expr1 and expr2
expr1 expr2	computes the logical OR of expr1 and expr2

In 'C' programming conditional statements are possible with the help of the following two constructs:

1. If statement
2. If-else statement

It is also called as branching as a program decides which statement to execute based on the result of the evaluated condition.

If statement

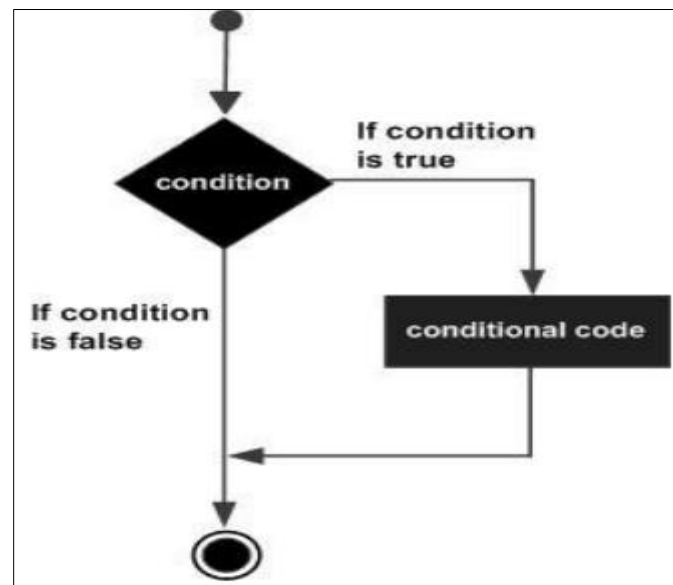
An if statement consists of a conditional expression followed by one or more statements.

If the conditional expression evaluates to true, then the block of code inside the if statement will be executed. If conditional expression evaluates to false, then the first set of code after the end of the if statement (after the closing curly brace) will be executed.

Syntax: The syntax of an if statement in C programming language is:

```
If (condition)
{
    //statements;
}
```

Flowchart:



Example: Checking if the number input by user is 0 or not. If its 0 then print Zero else print Non-zero

```
#include <stdio.h>
int main() {
    int num1, num2;
    printf ("Enter two integers\n");
    scanf ("%d%d", &num1,&num2);
    if (num2!=0)
        printf ("num1/num2 = %d\n", num1/num2);
    return 0;
}
```

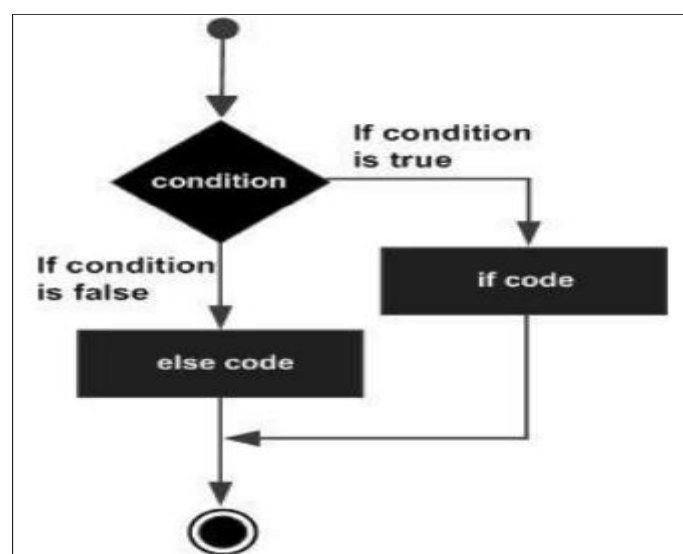
If else statement

An if statement can be followed by an optional else statement, which executes when the boolean expression is false.

Syntax: The syntax of an if...else statement in C programming language is:

```
if (condition)
{
    //statements;
}
else
{
    //statements;
}
```

Flowchart:



Example:

Checking the value of variable a, if it contains 10 then print value of a is 10, if it contains 20 then print value of a is 20, if its 30 then print value of a is 20 otherwise print None of the value is matching

```
#include <stdio.h>
int main() {
    int num;
    printf ("Enter any number\n");
    scanf ("%d", &num);
    if (num==0)
        printf("Zero");
    else
        printf("Non-zero");
    return 0;
}
```

If else if statement

An **if** statement can be followed by an optional **else if...else** statement, which is very useful to test various conditions using single **if...else if** statement.

When using **if** , **else if** , **else** statements there are few points to keep in mind:

- An **if** can have zero or one **else**'s and it must come after any **else if**'s.
- An **if** can have zero to many **else if**'s and they must come before the **else**.
- Once an **else if** succeeds, none of the remaining **else if**'s or **else**'s will be tested.

Syntax: The syntax of an **if...else** statement in C programming language is:

If (condition)

```
{  
    //statements;  
}
```

else if (condition)

```
{  
    //statements;  
}
```

else if (condition)

```
{  
    //statements;  
}
```

else

```
{  
    //statements;  
}
```

Example:

```
#include <stdio.h>

int main () {

    int a = 100;

    if( a == 10 ) {

        printf("Value of a is 10\n" );

    } else if( a == 20 ) {

        printf("Value of a is 20\n" );

    } else if( a == 30 ) {

        printf("Value of a is 30\n" );

    } else {

        printf("None of the values is matching\n" );

    }

    printf("Exact value of a is: %d\n", a );

    return 0;

}
```

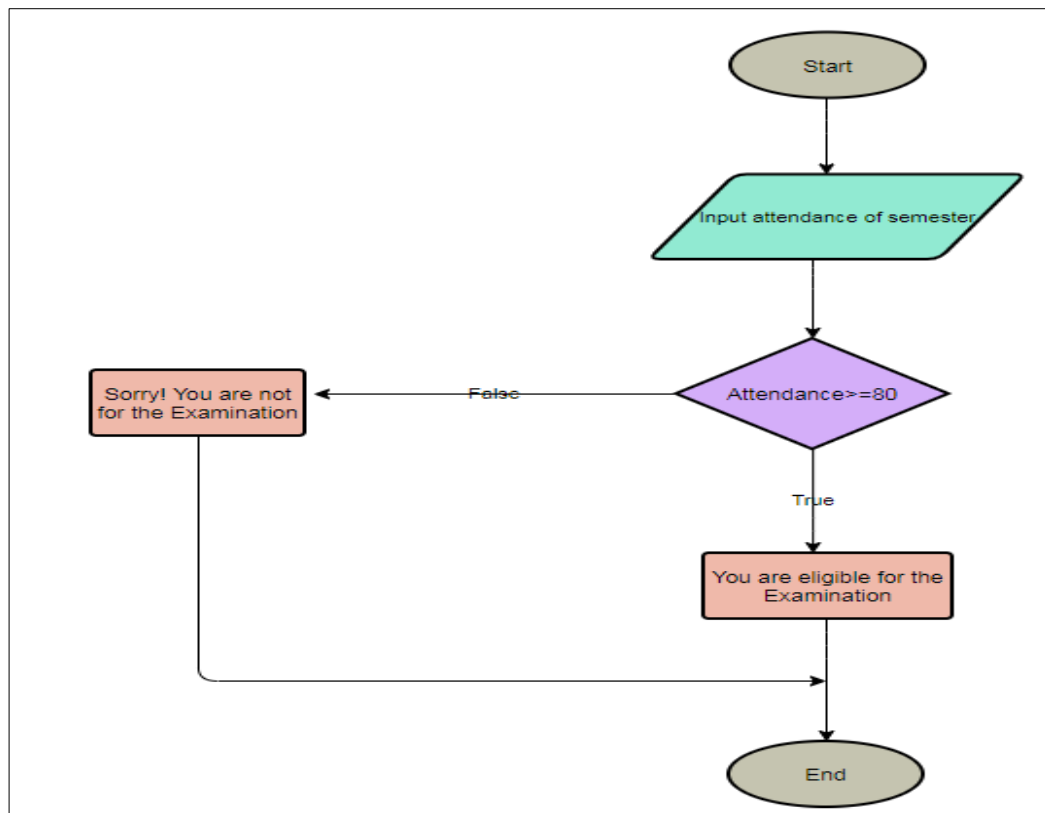
Problem

In FAST University 80% attendance is required for students to appear in the examination otherwise you won't be able to sit in an exam.

Algorithm

```
Start
Input attendance of semester
IF attendance >= 80
    then
        print "You are eligible for the
Examination"
    ELSE
        print "Sorry you are not eligible for
Exam"
END IF
```

Flowchart



C Code

```
#include <stdio.h>
int main() {
    int attendance;
    printf ("Enter Attendance of your semeseter:\n");
    scanf ("%d", &attendance);
    if (attendance>80)
        printf ("You are eligible for the Examination");
    else
        printf ("Sorry! You are not for the Examination");
}
```

Problem

FAST University wants to assign a grade to every PhD student according to the obtained marks. The grading criteria for the PhD students is given below.

Ph.D.	Equivalent %
A+	90 & above
A	86
A-	82
B+	78
B	74
F	70
	66
	62
	58
	54
	50

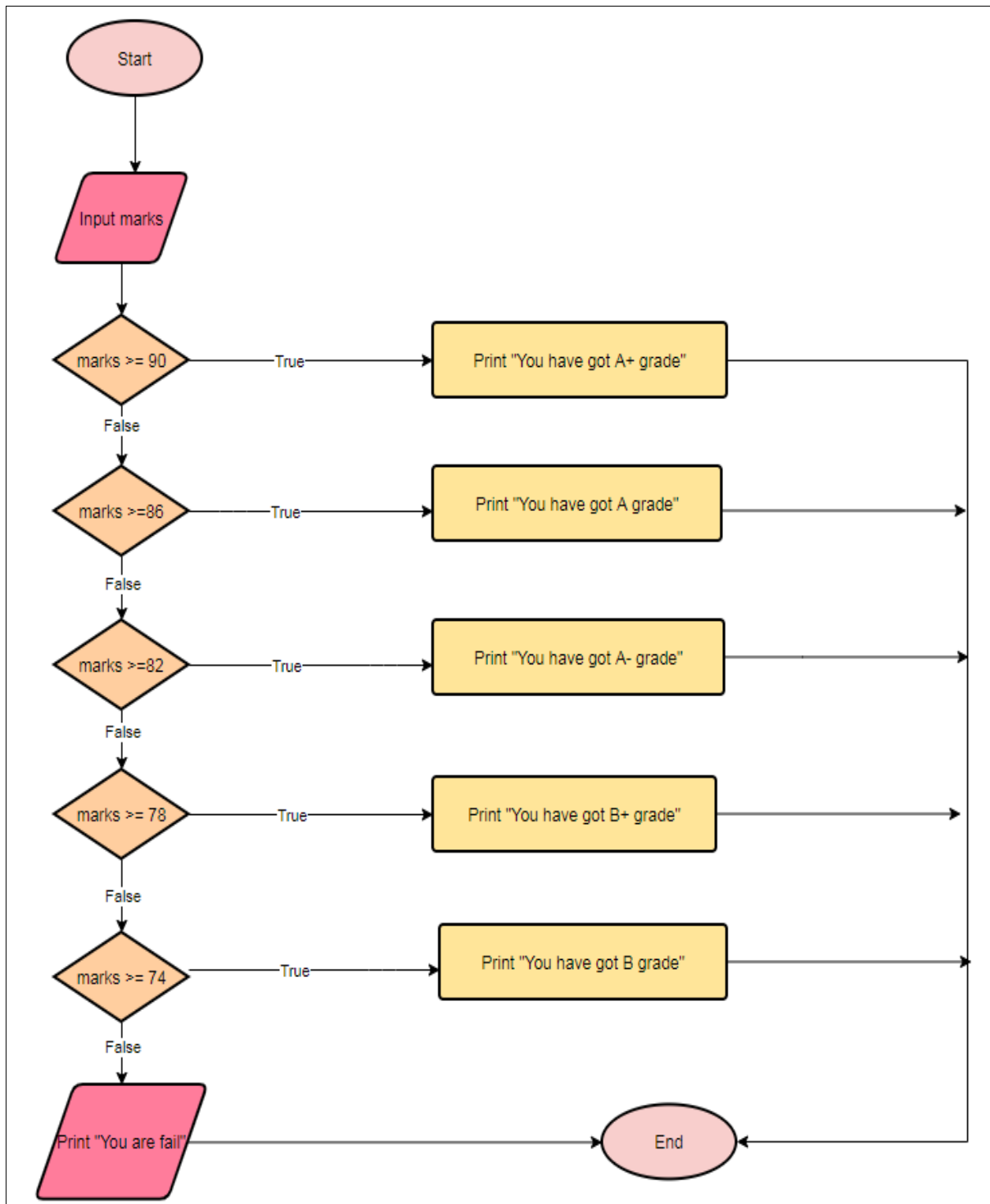
Write a program to assign the grade to each student according to his marks.

```

Start
Input marks
IF marks >= 90
    then
        print "You have got A+ grade"
ELSE
    IF marks >= 86
        then
            print "You have got A grade"
        ELSE
            IF marks >= 82
                then
                    print "You have got A- grade"
                ELSE
                    IF marks >= 78
                        then
                            print "You have got B+ grade"
                        ELSE
                            IF marks >= 74
                                then
                                    print "You have got B grade"
                                ELSE
                                    print "You are fail"
                                END IF
                            END IF
                        END IF
                    END IF
                END IF
            END IF
        END IF
    END IF
END IF
END

```

Algorithm



C Code

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

    if(marks >= 90){
printf("You have got A+ grade");
    }
    else if ( marks >=86){
printf("You got A grade");
    }
    else if ( marks >=82){
printf("You got A- grade");
    }
    else if ( marks >= 78){
printf("You got B+ grade");
    }
    else if ( marks >= 74){
printf("You got B grade");
    }
    else
    { printf("You are fail");
    }
return 0;
}
```

Switch Statement

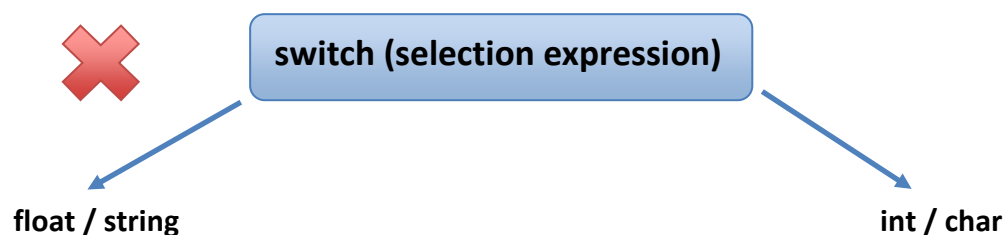
Another way that programs can make decisions is to use switch/case. The syntax of switch/case is shown in the figure below.

Syntax:

```
switch (selection expression) {  
    case 1:  
        //statement  
        break;  
    case 2:  
        //statement  
        break;  
    default:  
        //statement  
}
```

Here, when the execution arrow reaches the switch statement, the selection expression—in parenthesis after the keyword switch—is evaluated to a value.

This value is then used to determine which case to enter. The execution arrow then jumps to the corresponding case—the one whose label (the constant immediately after the keyword case) matches the selection expression's value. If no label matches, then the execution arrow jumps to the default case if there is one, and to the closing curly brace of the switch if not.



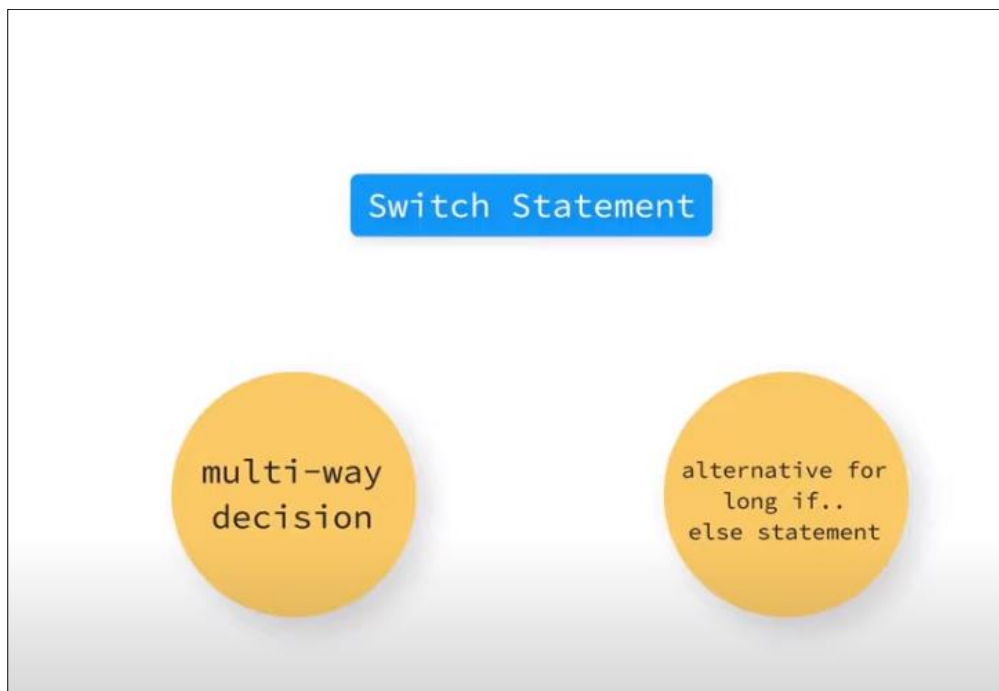
The following rules apply to a **switch** statement:

- The **expression** used in a **switch** statement must have an integral or enumerated type, or be of a class type in which the class has a single conversion function to an integral or enumerated type.
- You can have any number of case statements within a switch. Each case is followed by the value to be compared to and a colon.

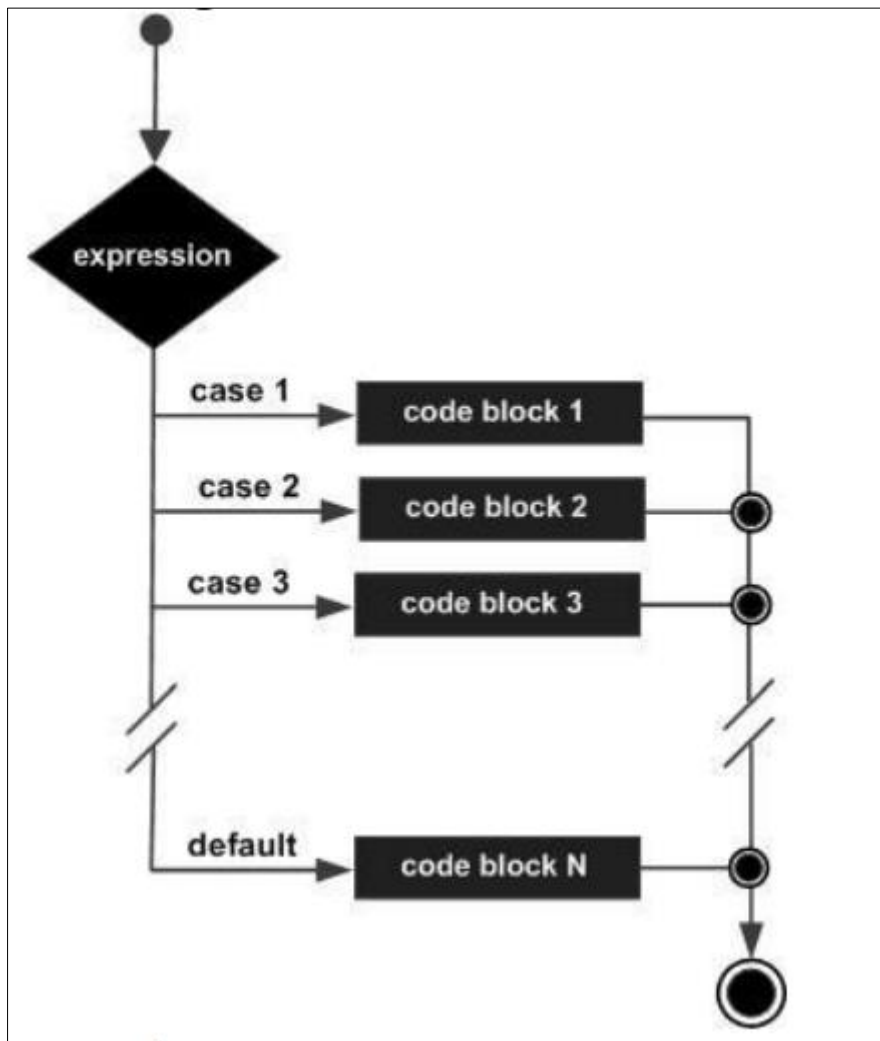
Switch statement is better than if else statement

A switch statement is **usually more efficient than a set of nested ifs**

Switch statement acts as a substitute for a long **if-else-if** ladder that is used to test a list of cases.



Flowchart



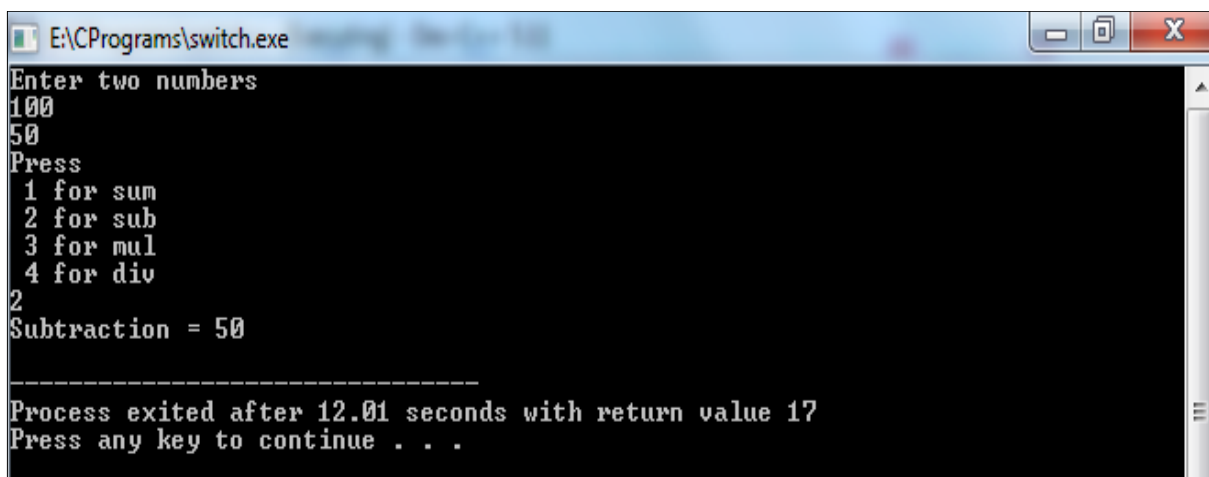
Example: Calculator using numbers to choose operators, if 1 is pressed then addition is performed, if 2 is pressed then subtraction if 3 is pressed then multiplication, if 4 is pressed then division otherwise print invalid choice.

```
#include <stdio.h>
int main() {
    int num1, num2, choice;
    printf ("Enter two numbers\n");
    scanf ("%d%d", &num1, &num2);

    printf("Press \n 1 for sum \n 2 for sub \n 3 for mul \n 4 for div\n");
    scanf ("%d", &choice);

    switch (choice) {
        case 1:
            printf("Sum = %d\n", num1 + num2);
            break;
        case 2:
            printf("Subtraction = %d\n", num1 - num2);
            break;
        case 3:
            printf("Multiplication = %d\n", num1 * num2);
            break;
        case 4:
            printf("Division = %d\n", num1 / num2);
            break;
        default:
            printf("Enter valid choice\n");
    }
}
```

Output:



```
E:\CPrograms\switch.exe
Enter two numbers
100
50
Press
1 for sum
2 for sub
3 for mul
4 for div
2
Subtraction = 50

-----
Process exited after 12.01 seconds with return value 17
Press any key to continue . . .
```