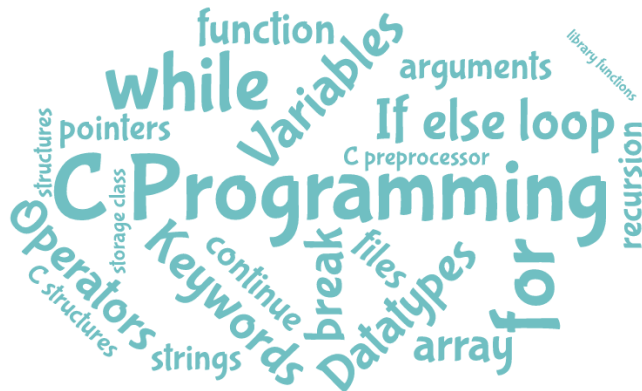


IT 112: Introduction to Programming




Dr. Manish Khare
Dr. Bakul Gohel

C instruction


- I hope before continuing to this lecture you must have written the basic `printf()` and `scanf()` programs. If you didn't then I strongly recommend you to do it.
- A program is nothing but a set of instruction. The program behaves as per the instructions that we give in it.
- Different instructions help us to achieve different task in a program

Type of Instruction

- 
- There are basically three types of instruction in C
 - **Type declaration instruction** – This instruction is used to declare the type of variables used in a C program.
 - **Arithmetic instruction** – This instruction is used to perform arithmetic operations on constants and variables.
 - **Control instruction** – This instruction is used to control the sequence of execution of various statements in a C program

Control Instructions in C

- As the name suggests the ‘Control Instructions’ enable us to specify the order in which the various instructions in a program are to be executed by the computer.
- In other words the control instructions determine the ‘flow of control’ in a program.
- There are four types of control instructions in C.
 - Sequence Control Instruction
 - Selection or Decision Control Instruction
 - Repetition or Loop Control Instruction
 - Case Control Instruction

- 
- The Sequence control instruction ensures that the instructions are executed in the same order in which they appear in the program.
 - Decision and Case control instructions allow the computer to take a decision as to which instruction is to be executed next.
 - The Loop control instruction helps computer to execute a group of statements repeatedly

Decisions! Decisions

- By default the instructions in a program are executed sequentially.
- In serious programming situations, seldom do we want the instructions to be executed sequentially.
- We want a set of instructions to be executed in one situation, and an entirely different set of instructions to be executed in another situation

Decision Control Instructions

➤ **Decision control instruction** can be implemented in C using

- **if** statement
- **if-else** statement
- The conditional operators

➤ C uses the keyword **if** to implement the decision control instruction. The general form of **if** statement:

- *if (this condition is true)*
 - *execute this statement ;*

Decision Control Instructions

➤ Relational Operators

- Condition can be specified using C's 'relational' operators.
- Relational operators allow us to compare two values.

this expression	is true if
$x == y$	x is equal to y
$x != y$	x is not equal to y
$x < y$	x is less than y
$x > y$	x is greater than y
$x <= y$	x is less than or equal to y
$x >= y$	x is greater than or equal to y

Relational Operators

/ Demonstration of if statement */*

```
main( )
```

```
{
```

```
    int num ;
```

```
    printf ( "Enter a number less than 10 " ) ;
```

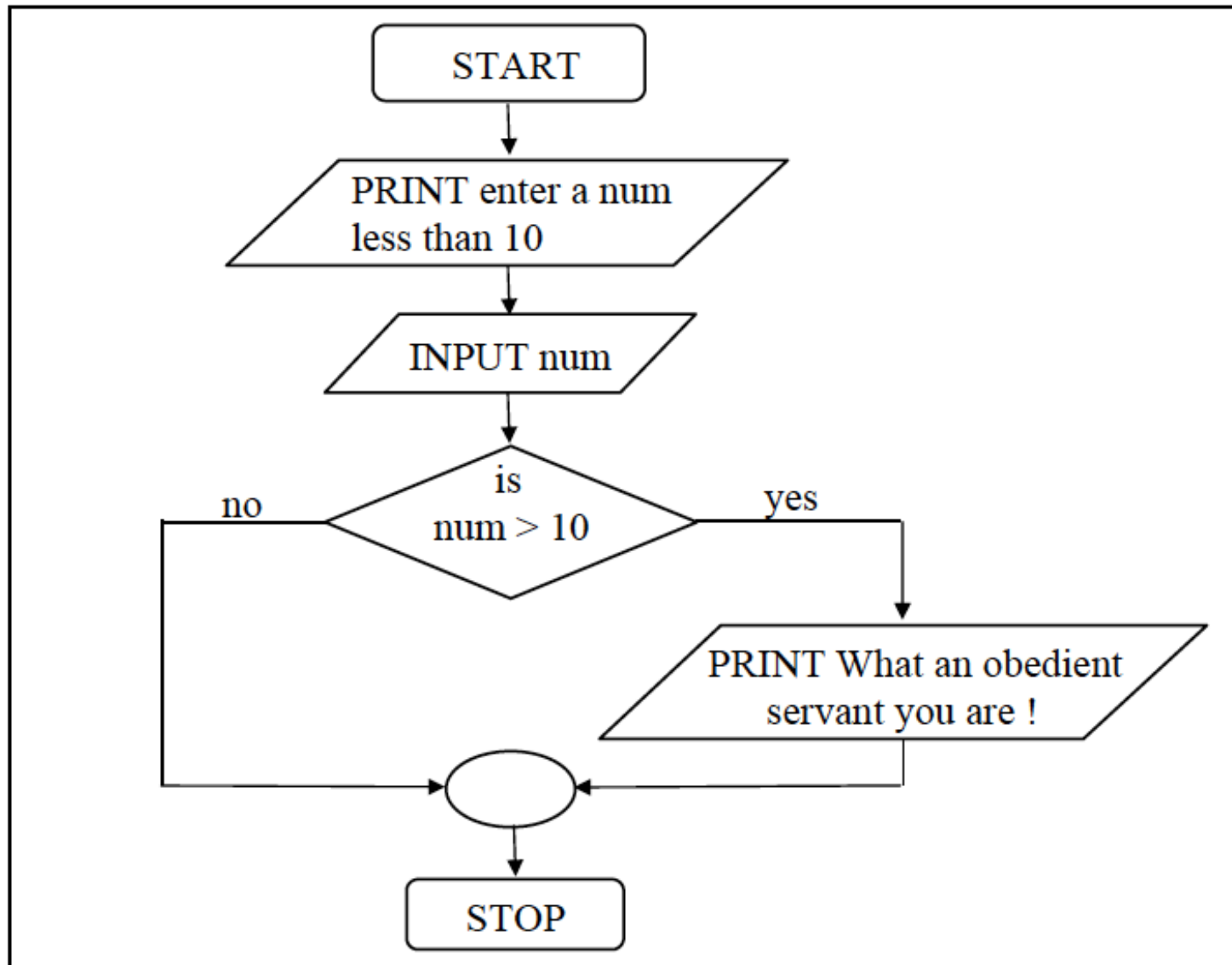
```
    scanf ( "%d", &num ) ;
```

```
    if ( num <= 10 )
```

```
        printf ( "What an obedient servant you are !" ) ;
```

```
}
```

"If" Statement

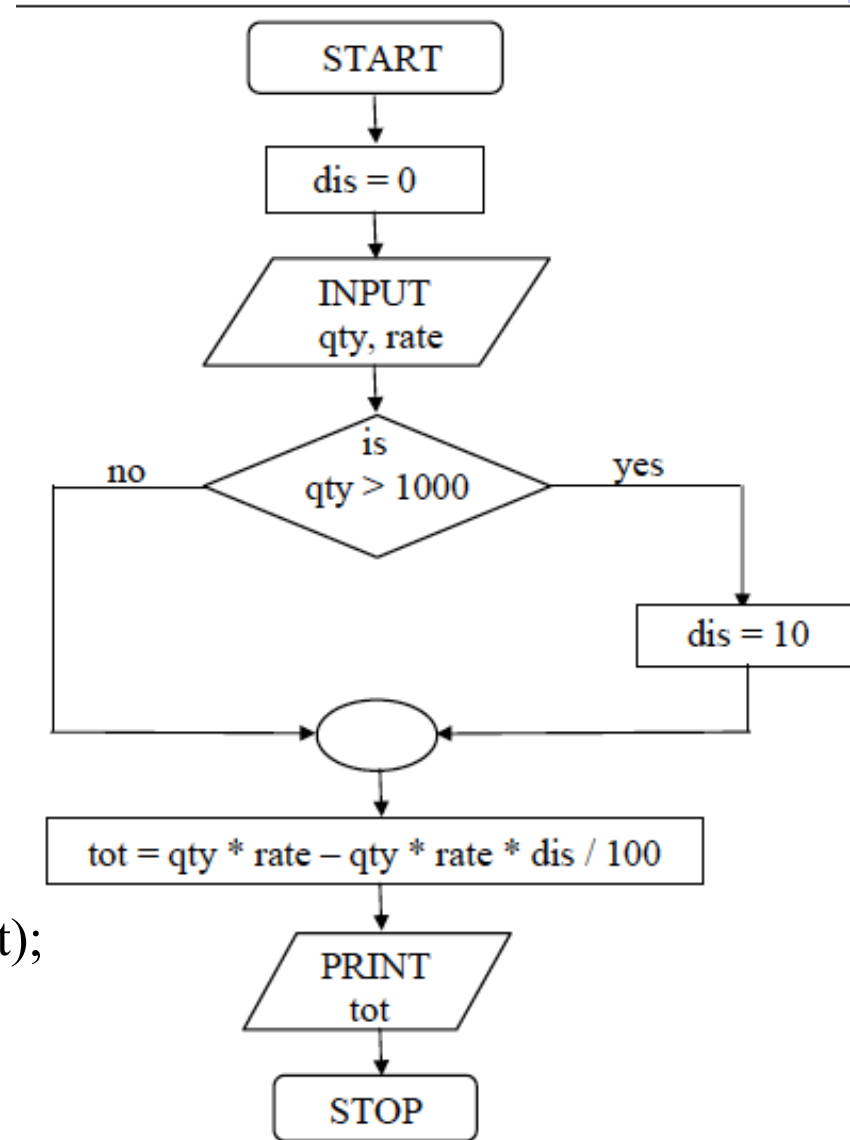


Example (IF)

- While purchasing certain items, a discount of 10% is offered if the quantity purchased is more than 1000. If quantity and price per item are input through the keyboard, write a program to calculate the total expenses.

Flowchart - Example

```
/* Calculation of total expenses */  
main()  
{  
    int qty, dis = 0 ;  
    float rate, tot ;  
  
    printf ("Enter quantity and rate ") ;  
    scanf ("%d %f", &qty, &rate) ;  
  
    if ( qty > 1000 )  
        dis = 10 ;  
    tot=(qty*rate)-(qty*rate*dis/100 ) ;  
    printf ("Total expenses = Rs. %f", tot);  
}
```



The Real Thing with “IF”



```
if (condition)  
    statement;
```

The expression can be any valid expression including a relational expression.

Can even use arithmetic expressions in the **if statement**.

```
if (expression)  
    statement;
```

The Real Thing with "IF"

- `if (3 + 2 % 5)`
 `printf ("This works") ;`
- `if (a = 10)`
 `printf ("Even this works") ;`
- `if (-5)`
 `printf ("Surprisingly even this works") ;`
- `if (a==b==c)`
 Result of `a==b` is compared with `c`

Note that in C a non-zero value is considered to be true, whereas a 0 is considered to be false.

Multiple Statements within “IF”

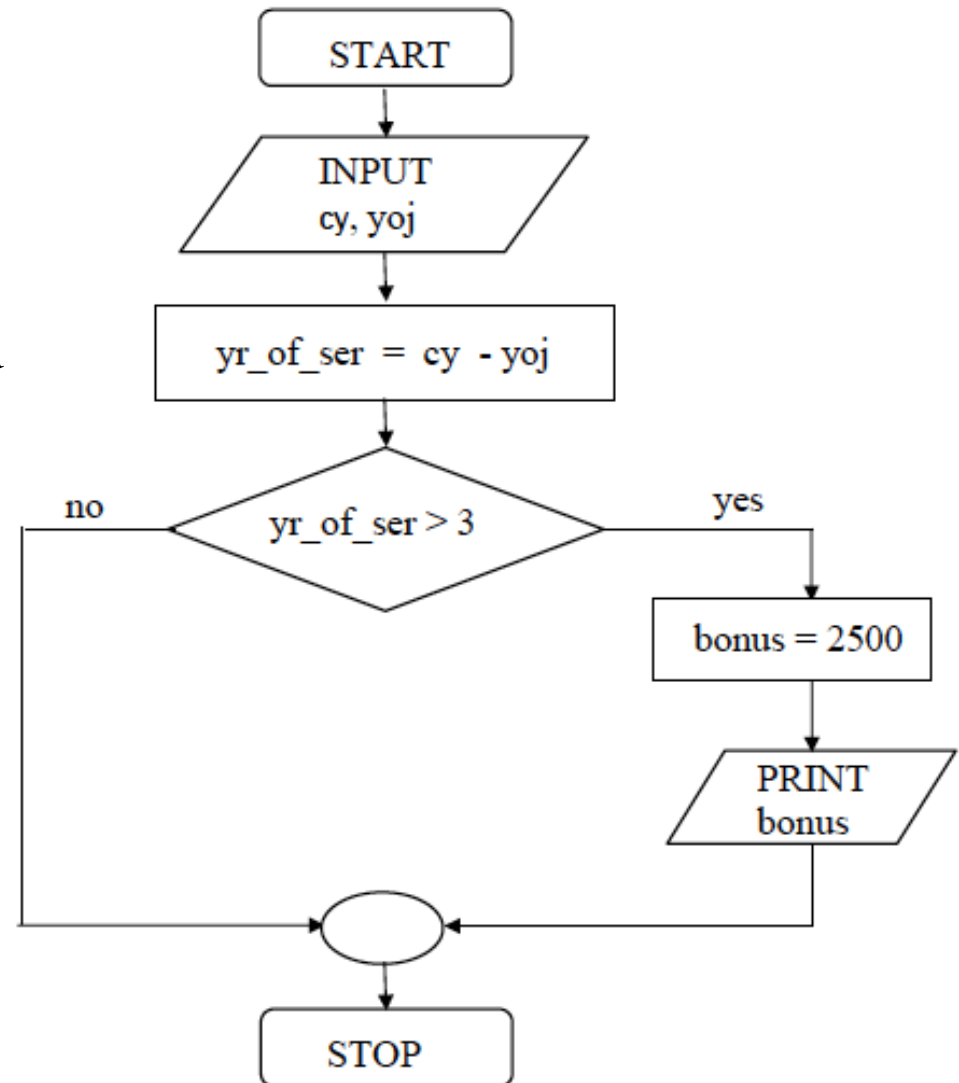
- In a “IF” statement multiple statements are to be executed then
 - They must be placed within a pair of braces.

Example:

The current year and the year in which the employee joined the organization are entered through the keyboard. If the number of years for which the employee has served the organization is greater than 3 then a bonus of Rs. 2500/- is given to the employee. If the years of service are not greater than 3, then the program should do nothing.

Multiple Statements within "IF" - Example

```
/* Calculation of bonus */  
main( )  
{  
    int bonus, cy, yoj, yr_of_ser ;  
    printf ("Enter current year and  
year of joining") ;  
    scanf ("%d %d", &cy, &yoj);  
    yr_of_ser = cy - yoj ;  
    if (yr_of_ser > 3)  
    {  
        bonus = 2500 ;  
        printf ("Bonus = Rs.  
%d", bonus);  
    }  
}
```



The if-else Statement

- The **if statement** by itself will execute a single statement, or a group of statements, when the expression following if evaluates to true.
- It does nothing when the expression evaluates to false.
- Can we execute one group of statements if the expression evaluates to true and another group of statements if the expression evaluates to false?
- Of course! This is what is the purpose of the **else statement**

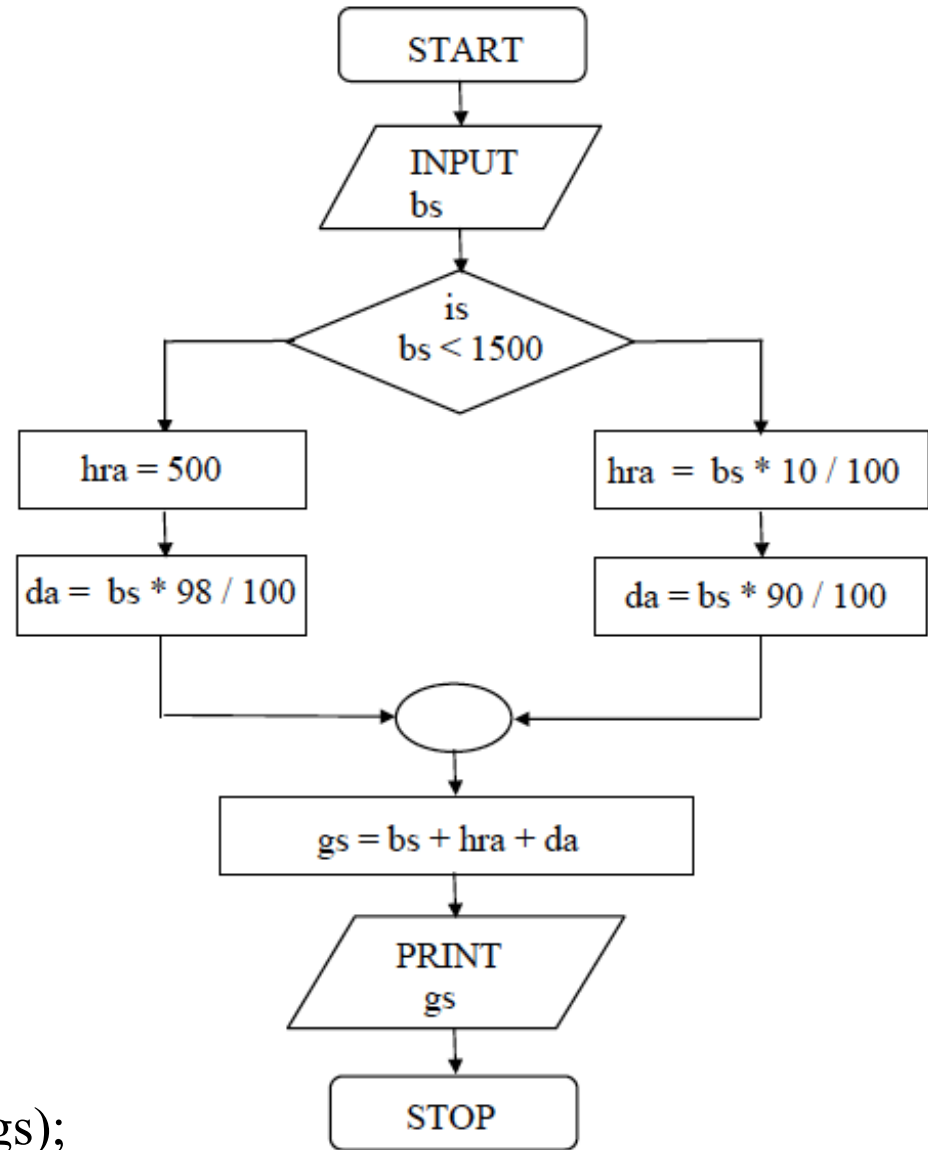
Example (IF-ELSE)

In a company an employee is paid as under:

If his basic salary is less than Rs. 1500, then HRA = 10% of basic salary and DA = 90% of basic salary. If his salary is either equal to or above Rs. 1500, then HRA = Rs. 500 and DA = 98% of basic salary. If the employee's salary is input through the keyboard write a program to find his gross salary.

"IF-ELSE" - Example

```
/* Calculation of gross salary */
main()
{
    float bs, gs, da, hra ;
    printf ( "Enter basic salary " ) ;
    scanf ( "%f", &bs ) ;
    if ( bs < 1500 )
    {
        hra = bs * 10 / 100 ;
        da = bs * 90 / 100 ;
    }
    else
    {
        hra = 500 ;
        da = bs * 98 / 100 ;
    }
    gs = bs + hra + da ;
    printf ("gross salary=Rs. %f", gs);
}
```



Points to Remember

- The group of statements after the **if** upto and not including the **else** is called an 'if block'. Similarly, the statements after the **else** form the 'else block'.
- Notice that the **else** is written exactly below the **if**. The statements in the if block and those in the else block have been indented to the right.
- Had there been only one statement to be executed in the if block and only one statement in the else block we could have dropped the pair of braces.
- As with the **if** statement, the default scope of **else** is also the statement immediately after the **else**. To override this default scope a pair of braces as shown in the above example must be used.

Nested IF-ELSESES

- Write an entire **if-else** construct within either the body of the **if** statement or the body of an **else** statement.
 - This is called 'nesting' of ifs

Nested IF-ELSESES (Example)

```
/* A quick demo of nested if-else */
main( )
{
    int i ;
    printf ( "Enter either 1 or 2 " ) ;
    scanf ( "%d", &i ) ;
    if ( i == 1 )
        printf ( "You would go to heaven !" ) ;
    else
    {
        if ( i == 2 )
            printf ( "Hell was created with you in mind" ) ;
        else
            printf ( "How about mother earth !" ) ;
    }
}
```

Forms of *IF*

```
if ( condition )  
    do this ;
```

```
if ( condition )  
{  
    do this ;  
    and this ;  
}
```

```
if ( condition )  
    do this ;  
else  
    do this ;
```

```
if ( condition )  
{  
    do this ;  
    and this ;  
}  
else  
{  
    do this ;  
    and this ;  
}
```

Forms of *IF*

```
if ( condition )
    do this ;
else
{
    if ( condition )
        do this ;
    else
    {
        do this ;
        and this ;
    }
}
```

```
if ( condition )
{
    if ( condition )
        do this ;
    else
    {
        do this ;
        and this ;
    }
}
else
    do this ;
```


Word of Caution

Guess output of the program?

```
#include <stdio.h>
main()
{
    int i;
    printf("Enter Value of i");
    scanf("%d", &i);

    if (i=5)
        printf("You have entered 5\n");
    else
        printf("You have entered something other than 5\n");
}
```

Word of Caution

Enter value of i 200

o/p: You have entered 5

Enter value of i 5

o/p: You have entered 5

Enter value of i 9999

o/p: You have entered 5

Surprised?

Word of Caution: Another

Guess output of the program?

```
#include <stdio.h>
main()
{
    int i;
    printf("Enter Value of i");
    scanf("%d", &i);

    if (i==5);
        printf("You have entered 5\n");
}
```

You have entered 5

Word of Caution: Another

```
#include <stdio.h>
main()
{
    int i;
    printf("Enter Value of i");
    scanf("%d", &i);

    if (i==5)
        ;
        printf("You have entered 5\n");
}
```

Note:

- if the condition evaluates to true, the ; (null statement, which does nothing in execution gets executed.
- If the condition fails, printf() gets executed.
- In any case, printf() gets executed – fail or pass



➤ More complex Decision Making

- If a get good marks in my final year and if my GRE and TOFEL scores are good and if I get good recommendations or of I do not get a job with good prospects and if my family conditions permit me, then I would think of doing MS in US.
- How can such complex decision making be implemented in C?

Use of Logical Operators

➤ C allows usage of three logical operators, namely, `&&`, `||` and `!`.

■ These are to be read as:

- 'AND' `&&`
- 'OR' `||`
- 'NOT' `!`

➤ **Remember:** Don't use the single symbol `|` and `&`. These have a different meaning. They are bitwise operators.

Use of operators && and | |

➤ Example:

The marks obtained by a student in 5 different subjects are input through the keyboard. The student gets a division as per the following rules:

- Percentage above or equal to 60 - First division
- Percentage between 50 and 59 - Second division
- Percentage between 40 and 49 - Third division
- Percentage less than 40 - Fail

Write a program to calculate the division obtained by the student.

Solution Method 1

/ Method – I */*

main()

{

int m1, m2, m3, m4, m5, per ;

printf ("Enter marks in five subjects ") ;

scanf ("%d %d %d %d %d", &m1, &m2, &m3, &m4,
&m5) ;

per = (m1 + m2 + m3 + m4 + m5) / 5 ;

Solution Method 1

```
if ( per >= 60 )
    printf ( "First division " );
else
{
    if ( per >= 50 )
        printf ( "Second division" );
    else
    {
        if ( per >= 40 )
            printf ( "Third division" );
        else
            printf ( "Fail" );
    }
}
} /* main */
```

Use of 'Logical operators'

/ Method – II */*

main()

{

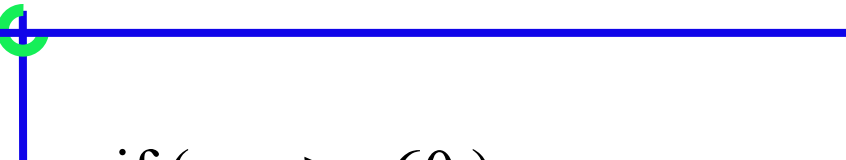
int m1, m2, m3, m4, m5, per ;

printf ("Enter marks in five subjects ") ;

scanf ("%d %d %d %d %d", &m1, &m2, &m3, &m4,
&m5) ;

per = (m1 + m2 + m3 + m4 + m5) / 5 ;

Solution Method 2



```
if ( per >= 60 )  
    printf ( "First division" ) ;  
  
if ( ( per >= 50 ) && ( per < 60 ) )  
    printf ( "Second division" ) ;  
  
if ( ( per >= 40 ) && ( per < 50 ) )  
    printf ( "Third division" ) ;  
  
if ( per < 40 )  
    printf ( "Fail" ) ;  
} /* main */
```

The *else if* clause

```
if ( per >= 60 )
    printf ( "First division " );
else
{
    if ( per >= 50 )
        printf ( "Second division" );
    else
    {
        if ( per >= 40 )
            printf("Third division");
        else
            printf ( "Fail" );
    }
}
```

```
/* else if ladder demo */
main( )
{
    int m1, m2, m3, m4, m5, per ;
    per = ( m1+ m2 + m3 + m4+ m5 ) /5;
    if ( per >= 60 )
        printf ( "First division" );
    else if ( per >= 50 )
        printf ("Second division");
    else if ( per >= 40 )
        printf ( "Third division" );
    else
        printf ( "fail" );
}
```

The *else if* clause

```
if ( i == 2 )
    printf ( "With you..." );
else
{
    if ( j == 2 )
        printf ( "...All the time" );
}
```

```
if ( i == 2 )
    printf ( "With you..." );
else if ( j == 2 )
    printf ( "...All the time " );
```

Example – Logical Operators

Example:

A company insures its drivers in the following cases:

- If the driver is married.
- If the driver is unmarried, male & above 30 years of age.
- If the driver is unmarried, female & above 25 years of age.

In all other cases the driver is not insured. If the marital status, sex and age of the driver are the inputs, write a program to determine whether the driver is to be insured or not.

<https://ideone.com/sh7j2i>

Example – Logical Operators

```
/* Insurance of driver - using logical operators */
main( )
{
    char sex, ms ;
    int age ;
    printf ( "Enter age, sex, marital status " ) ;
    scanf ( "%d %c %c" &age, &sex, &ms ) ;

    if ( ( ms == 'M') || ( ms == 'U' && sex == 'M' && age
> 30 ) || ( ms == 'U' && sex == 'F' && age > 25 ) )
        printf ( "Driver is insured" ) ;

    else
        printf ( "Driver is not insured" ) ;

}
```

Example – Logical Operators

<https://ideone.com/BH7NhK>

Write a program to calculate the salary as per the following table:

Gender	Years of Service	Qualifications	Salary
Male	≥ 10	Post-Graduate	15000
	≥ 10	Graduate	10000
	< 10	Post-Graduate	10000
	< 10	Graduate	7000
Female	≥ 10	Post-Graduate	12000
	≥ 10	Graduate	9000
	< 10	Post-Graduate	10000
	< 10	Graduate	6000

The ! Operator

➤ This operator reverses the result of the expression it operates on.

- if the expression evaluates to a non-zero value, then applying ! operator to it results into a 0. Vice versa
- if the expression evaluates to zero then on applying ! operator to it makes it 1, a non-zero value.

- $!(y < 10)$

- “not y less than 10”

- $y \geq 10$.

if (!flag) is same as if (flag == 0)

Hierarchy of Operators

Operators	Type
!	Logical NOT
* / %	Arithmetic and modulus
+ -	Arithmetic
< > <= >=	Relational
== !=	Relational
&&	Logical AND
	Logical OR
=	Assignment

Summarize Logical Operators

Operands		Results			
x	y	!x	!y	x && y	x y
0	0	1	1	0	0
0	non-zero	1	0	0	1
non-zero	0	0	1	0	1
non-zero	non-zero	0	0	1	1

The Conditional Operators

➤ Also known as ternary operator

➤ A short form of if-then-else

■ *expression 1 ? expression 2 : expression 3*

■ “if expression 1 is true (that is, if its value is non-zero), then the value returned will be expression 2, otherwise the value returned will be expression 3”.

```
int x, y ;  
scanf ( "%d", &x ) ;  
y = ( x > 5 ? 3 : 4 ) ;
```

```
if ( x > 5 )  
    y = 3 ;  
else  
    y = 4 ;
```

More on Conditional Operators

```
char a ;  
int y ;  
scanf ( "%c", &a ) ;  
y = ( a >= 65 && a <= 90 ? 1 : 0 ) ;
```

1 would be assigned to y if `a >= 65 && a <= 90` evaluates to true, otherwise 0 would be assigned.

- It's not necessary that the conditional operators should be used only in arithmetic statements.

```
int i ;  
scanf ( "%d", &i ) ;  
( i == 1 ? printf ( "Amit" ) : printf ( "All and sundry" ) ) ;
```

More on Conditional Operators

The conditional operators can be nested as shown below.

```
int big, a, b, c ;  
big = ( a > b ? ( a > c ? 3: 4 ) : ( b > c ? 6: 8 ) ) ;
```

Check the following statement:

```
a > b ? g = a : g = b ;
```

- Error: 'Lvalue Required'.
- Can be removed by adding parenthesis
 - *a > b ? g = a : (g = b) ;*
- In absence of parentheses the compiler believes that **b** is being assigned to the result of the expression to the left of second =. Hence it reports an error.

Exercise

➤ What will be output of the following program

```
# include <stdio.h>
```

```
main( )
```

```
{
```

```
    int a = 300, b, c ;
```

```
    if ( a >= 400 )
```

```
        b = 300 ;
```

```
    c = 200 ;
```

```
    printf ( "\n%d %d", b, c ) ;
```

```
}
```

B will contain some garbage value and c will be equal to 200.

Exercise

➤ What will be output of the following program

```
# include <stdio.h>
```

```
main( )
```

```
{
```

```
    int a = 500, b, c ;
```

```
    if ( a >= 400 )
```

```
        b = 300 ;
```

```
    c = 200 ;
```

```
    printf ( "\n%d %d", b, c ) ;
```

```
}
```

300, 200

Exercise

➤ What will be output of the following program

```
# include <stdio.h>
```

```
main( )
```

```
{
```

```
    int x =3;
```

```
    float y = 3.0;
```

```
    if ( x == y )
```

```
        printf ("x and y are equal\n") ;
```

```
    else
```

```
        printf("x and y are not equal\n");
```

```
}
```

x and y are equal

Exercise

➤ What will be output of the following program

```
# include <stdio.h>
```

```
main( )
```

```
{
```

```
    int x =3, y, z;
```

```
    y = x = 10;
```

```
    z = x < 10;
```

```
    printf (“x = %d, y = %d, z = %d\n”, x, y, z) ;
```

```
}
```

10, 10, 0 (since $x < 10$ turns to be FALSE it is replaced by 0)

Exercise

➤ What will be output of the following program

```
# include <stdio.h>
```

```
main( )
```

```
{
```

```
    int x = 10, y = 20, z = 5, i;
```

```
    i = x < y < z;
```

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

```
}
```

Since $x < y$ turns to be TRUE it is replaced by 1. Then $1 < z$ is compared and to be TRUE. The 1 is assigned to i.

Exercise

➤ What will be output of the following program

```
#include <stdio.h>

int X=40;

main()
{
    int X=20;
    printf("%d\n", X);
}
```

Whenever there is conflict between a local variable and global variable, the local variable gets priority. (x = 20)

Exercise

➤ What will be output of the following program

```
#include <stdio.h>
```

```
main( )
```

```
{
```

```
    int i = 65 ;
```

```
    char j = 'A' ;
```

```
    if ( i == j )
```

```
        printf ( "C is WOW" ) ;
```

```
    else
```

```
        printf( "C is a headache" ) ;
```

```
}
```

C is WOW

Exercise

➤ What will be output of the following program

```
#include <stdio.h>
```

```
main( )
```

```
{
```

```
    float a = 12.25, b = 12.52 ;
```

```
    if ( a = b )
```

```
        printf ( "\\na and b are equal" ) ;
```

```
}
```

Exercise

➤ What will be output of the following program

```
#include <stdio.h>
```

```
main( )
```

```
{
```

```
    int j = 10, k = 12 ;
```

```
    if ( k >= j )
```

```
    {
```

```
        {
```

```
            k = j ;
```

```
            j = k ;
```

```
        }
```

```
    }
```

```
}
```

Exercise

➤ What will be output of the following program

```
#include <stdio.h>
```

```
main( )
```

```
{
```

```
    if ( 'X' < 'x' )
```

```
        printf ( "\nascii value of X is smaller than  
that of x" ) ;
```

```
}
```


Exercise

➤ What will be output of the following program

```
#include <stdio.h>
```

```
main( )
```

```
{
```

```
    int x = 10 ;
```

```
    if ( x >= 2 ) then
```

```
        printf ( "\n%d", x ) ;
```

```
}
```

Exercise

➤ What will be output of the following program

```
#include <stdio.h>
```

```
main( )
```

```
{
```

```
    int x = 10, y = 15 ;
```

```
    if ( x % 2 = y % 3 )
```

```
        printf ( "\nCarpathians" ) ;
```

```
}
```

Exercise

➤ What will be output of the following program

```
#include <stdio.h>
```

```
main( )
```

```
{
```

```
    int x = 30 , y = 40 ;
```

```
    if ( x == y )
```

```
        printf( "x is equal to y" ) ;
```

```
    elseif ( x > y )
```


```
        printf( "x is greater than y" ) ;
```

```
    elseif ( x < y )
```

```
        printf( "x is less than y" ) ;
```

```
}
```

Exercise

- 
- Any integer is input through the keyword.
Write a program to find out whether it is an odd or even number.

<https://ideone.com/FMV8bS>

Exercise: Guess the Output

```
main( )
{
    int i = 4, z = 12 ;

    if ( i = 5 || z > 50 )
        printf ( "\nDean of students affairs" ) ;
    else
        printf ( "\nDosa" ) ;
}
```

Exercise: Guess the Output

```
main( )
{
    int i = 4, j = -1, k = 0, w, x, y, z ;

    w = i || j || k ;
    x = i && j && k ;
    y = i || j && k ;
    z = i && j || k ;

    printf ( "\nw = %d x = %d y = %d z = %d",
w, x, y, z ) ;
}
```

Exercise: Guess the Output

```
main( )
{
    int x = 20 , y = 40 , z = 45 ;
    if ( x > y && x > z )
        printf( "x is big" ) ;
    else if ( y > x && y > z )
        printf( "y is big" ) ;
    else if ( z > x && z > y )
        printf( "z is big" ) ;
}
```

Exercise: Guess the Output

```
main( )
{
    char spy = 'a', password = 'z' ;
    if ( spy == 'a' or password == 'z' )
        printf ( "\nAll the birds are safe in the
nest" ) ;
}
```


Exercise: Guess the Output

```
main( )
{
    int x = 2;
    if ( x == 2 && x != 0 ) ;
    {
        printf ( "\nHi" ) ;
        printf( "\nHello" ) ;
    }
    else
        printf( "Bye" ) ;
}
```

Exercise: Guess the Output

```
main( )  
{  
    int i = 10, j ;  
    i >= 5 ? j = 10 : j = 15 ;  
    printf ( "\n%d %d", i, j ) ;  
}
```

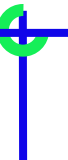
Exercise: Guess the Output

```
main( )  
{  
    int a = 5 , b = 6 ;  
    ( a == b ? printf( "%d",a) ) ;  
}
```

Case Control Instruction

- In real life we are often faced with situations where we are required to make a choice between a number of alternatives rather than only one or two.
- C provides a special control statement that allows us to handle such cases effectively; rather than using a series of if statements.

Decision Using Switch

- 
- The control statement that allows us to make a decision from the number of choices is called a **switch**, or more correctly a **switch-case-default**, since these three keywords go together to make up the control statement

```
switch ( integer expression )
```

```
{
```

```
    case constant 1 :
```

```
        do this ;
```

```
    case constant 2 :
```

```
        do this ;
```

```
    case constant 3 :
```

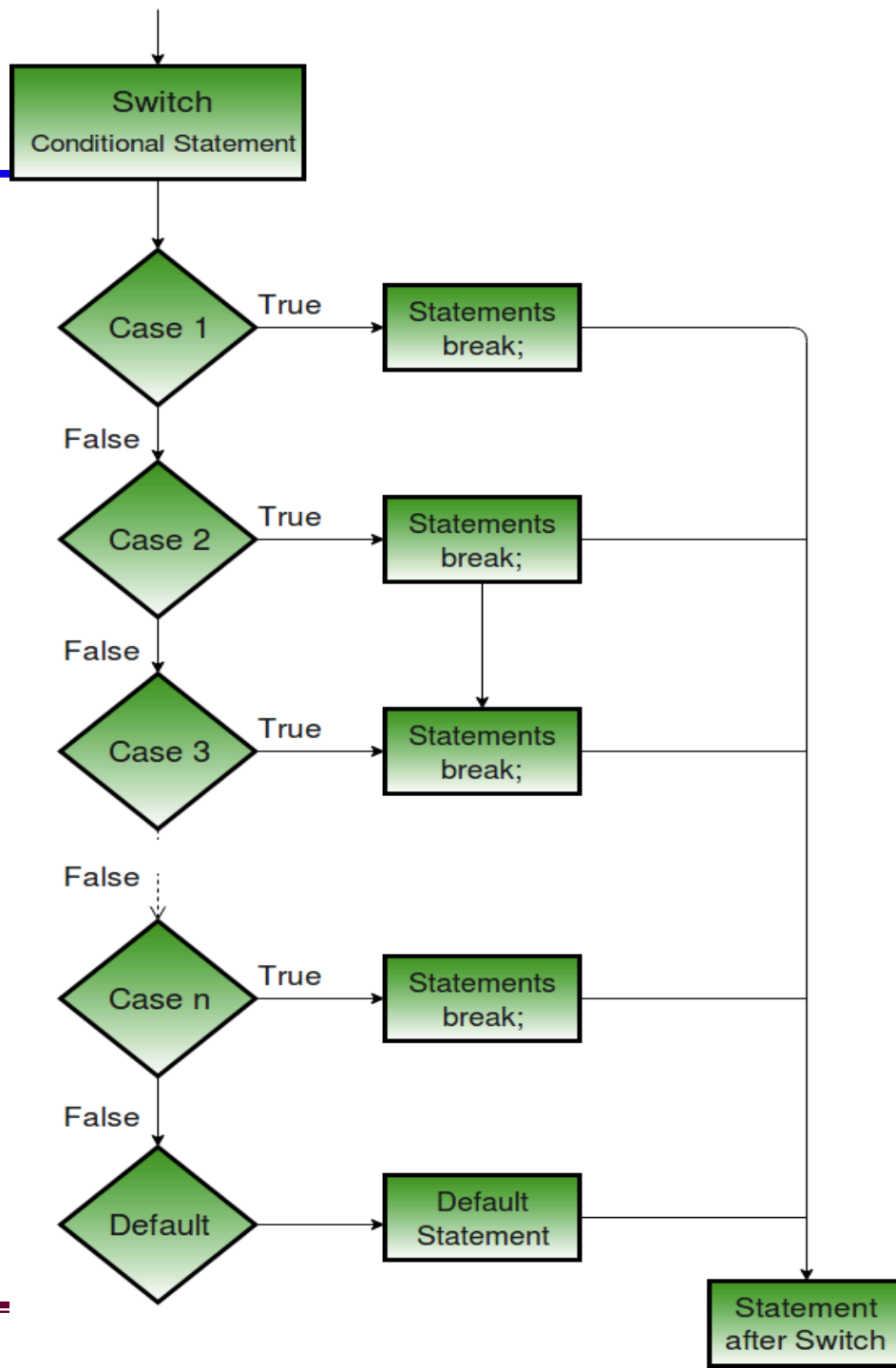
```
        do this ;
```


```
    default :
```


```
        do this ;
```

```
}
```

Decision Using Switch



- 
- The integer expression following the keyword switch is any C expression that will yield an integer value.
 - It could be an integer constant like 1, 2 or 3, or an expression that evaluates to an integer.
 - The keyword case is followed by an integer or a character constant.
 - Each constant in each case must be different from all the others.
 - The “do this” lines in the above form of switch represent any valid C statement.

- 
- What happens when we run a program containing a switch?
- First, the integer expression following the keyword switch is evaluated.
 - The value it gives is then matched, one by one, against the constant values that follow the case statements.
 - When a match is found, the program executes the statements following that case, and all subsequent case and default statements as well.
 - If no match is found with any of the case statements, only the statements following the default are executed



main()

{

int i = 2 ;

switch (i)

{

case 1 :

printf ("I am in case 1 \n") ;

case 2 :

printf ("I am in case 2 \n") ;

case 3 :

printf ("I am in case 3 \n") ;

default :

printf ("I am in default \n") ;

}

}

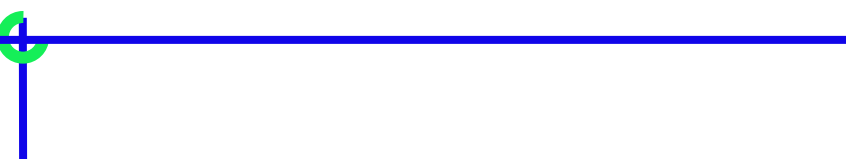
The output of this program would be

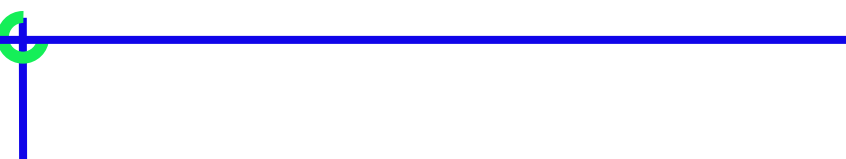
I am in case 2


I am in case 3

I am in default

Surprised!!

- 
- The output is definitely not what we expected!
 - We didn't expect the second and third line in the above output.
 - The program prints case 2 and 3 and the default case. Well, yes.
 - We said the switch executes the case where a match is found and all the subsequent cases and the default as well.

- 
- If you want that only case 2 should get executed, it is upto you to get out of the switch then and there by using a break statement.



```
main( )
```

```
{
```

```
int i = 2 ;
```

```
switch ( i )
```

```
{
```

```
case 1 :
```

```
    printf ( "I am in case 1 \n" ) ;
```

```
    break ;
```

```
case 2 :
```

```
    printf ( "I am in case 2 \n" ) ;
```

```
    break ;
```

```
case 3 :
```

```
    printf ( "I am in case 3 \n" ) ;
```


```
    break ;
```

```
default :
```

```
    printf ( "I am in default \n" ) ; } }
```

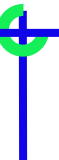
The output of this program would be:

I am in case 2

- 
- The previous example shows how this is done.
 - Note that there is no need for a break statement after the default, since the control comes out of the switch anyway

Few tips about usage of switch

- The program in previous slides may give you an impression that cases in a switch must be arranged in ascending order – 1,2,3, and default.
- In fact, you can put the cases in any order



```
main( )
```

```
{
```

```
int i = 22 ;
```

```
switch ( i )
```

```
{
```

```
case 121 :
```

```
    printf ( "I am in case 121 \n" ) ;
```

```
    break ;
```

```
case 7 :
```

```
    printf ( "I am in case 7 \n" ) ;
```

```
    break ;
```

```
case 22 :
```

```
    printf ( "I am in case 22 \n" ) ;
```

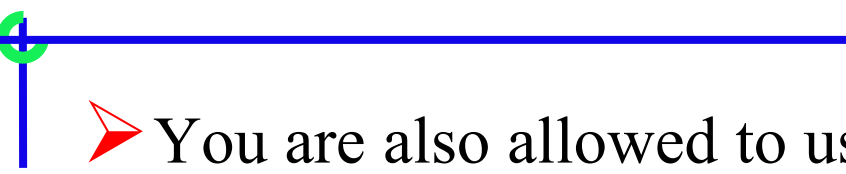
```
    break ;
```

```
default :
```


```
    printf ( "I am in default \n" ) ; } }
```

The output of this program would be:

I am in case 22



➤ You are also allowed to use char values in case and switch as shown in the following program



```
main( )
```

```
{
```

```
char i = 'x' ;
```

```
switch ( i )
```

```
{
```

```
case 'v' :
```

```
    printf ( "I am in case v \n" ) ;
```

```
    break ;
```

```
case 'a' :
```

```
    printf ( "I am in case a \n" ) ;
```

```
    break ;
```

```
case 'x' :
```

```
    printf ( "I am in case x \n" ) ;
```

```
    break ;
```


```
default :
```


```
    printf ( "I am in default \n" ) ; } }
```

The output of this program would be:

I am in case x

In fact here when we use 'v', 'a', 'x' they are actually replaced by the ASCII values (118, 97, 120) of these character constants

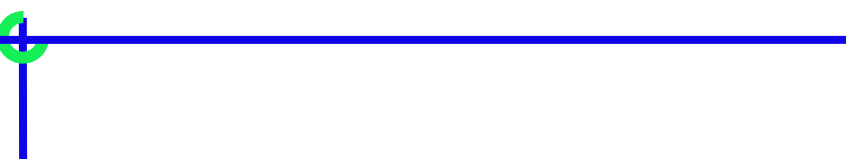
- 
- At times we may want to execute a common set of statements for multiple cases.
 - How this can be done is shown in the following example

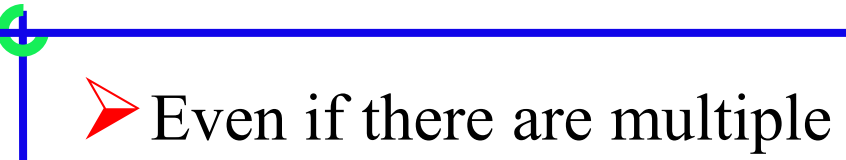



```
main( )
{
char ch ;


printf ( "Enter any of the alphabet a, b, or c " ) ;
scanf ( "%c", &ch ) ;
switch ( ch )
{ case 'a' :
case 'A' :
    printf ( "a as in ashar" ) ;
    break ;
case 'b' :
case 'B' :
    printf ( "b as in brain" ) ;
    break ;
```

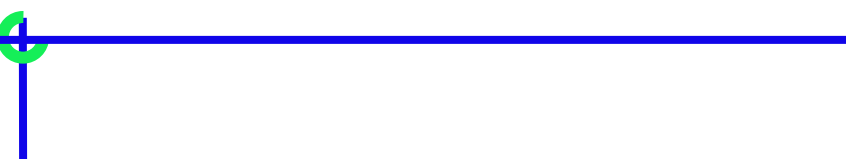
```
case 'c' :
case 'C' :
    printf ( "c as in cookie" ) ;
    break ;
default :
    printf ( "wish you knew what are
alphabets" ) ;
}
}
```

- 
- Here, we are making use of the fact that once a case is satisfied the control simply falls through the case till it doesn't encounter a break statement.
 - That is why if an alphabet a is entered the case 'a' is satisfied and since there are no statements to be executed in this case the control automatically reaches the next case i.e. case 'A' and executes all the statements in this case.

- 
- Even if there are multiple statements to be executed in each case there is no need to enclose them within a pair of braces (unlike if, and else).
 - Every statement in a switch must belong to some case or the other. If a statement doesn't belong to any case the compiler won't report an error. However, the statement would never get executed.

- 
- If we have no default case, then the program simply falls through the entire switch and continues with the next instruction (if any,) that follows the closing brace of switch.
 - The break statement when used in a switch takes the control outside the switch.
 - However, use of continue will not take the control to the beginning of switch as one is likely to believe.

- 
- In principle, a switch may occur within another, but in practice it is rarely done. Such statements would be called nested switch statements.
 - The switch statement is very useful while writing menu driven programs.



➤ We can check the value of any expression in a switch.

- Thus the following switch statements are legal.

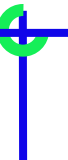
- `switch (i + j * k)`

- `switch (23 + 45 % 4 * k)`

- `switch (a < 4 && b > 7)`

- Expressions can also be used in cases provided they are constant expressions.

- Thus case `3 + 7` is correct, however, case `a + b` is incorrect.

- 
- Is switch a replacement for if? Yes or No
 - Yes, because it offers a better way of writing programs as compared to if, and no because in certain situations we are left with no choice but to use if.
 - The disadvantage of switch is that one cannot have a case in a switch which looks like:
 - `case i <= 20 :`
 - The advantage of switch over if is that it leads to a more structured program and the level of indentation is manageable, more so if there are multiple statements within each case of a switch.

switch Versus if-else Ladder

- There are some things that you simply cannot do with a switch.
- These are:
 - A. A float expression cannot be tested using a switch
 - B. Cases can never have variable expressions (for example it is wrong to say case `a + 3` :)
 - C. Multiple cases cannot use same expressions.
- Thus the following switch is illegal:



```
switch ( a )
```

```
{
```

```
case 3 :
```

```
...
```

```
case 1 + 2 :
```

```
...
```

```
}
```



What would be the output

```
main( )
{
    char suite = 3 ;
    switch ( suite )
    {
        case 1 :
            printf ( "\nDiamond" ) ;
        case 2 :
            printf ( "\nSpade" ) ;
        default :
            printf ( "\nHeart" ) ;
    }
    printf ( "\nI thought one wears a suite" ) ;
}
```



What would be the output

```
main( )
{
    int k, j = 2 ;
    switch ( k = j + 1 )
    {
        case 0 :
            printf ( "\nTailor" ) ;
        case 1 :
            printf ( "\nTutor" ) ;
        case 2 :
            printf ( "\nTramp" ) ;
        default :
            printf ( "\nPure Simple Egghead!" ) ;
    }
}
```



What would be the output

```
main( )
```

```
{  
    int k ;  
    float j = 2.0 ;  
    switch ( k = j + 1 )  
    {  
        case 3 :  
            printf ( "\nTrapped" ) ;  
            break ;  
        default :  
            printf ( "\nCaught!" ) ;  
    }  
}
```



What would be the output

```
main( )
{
    int ch = 'a' + 'b' ;
    switch ( ch )
    {
        case 'a' :
        case 'b' :
            printf ( "\nYou entered b" ) ;
        case 'A' :
            printf ( "\na as in ashar" ) ;
        case 'b' + 'a' :
            printf ( "\nYou entered a and b" ) ;
    }
}
```



```
main( )
```

```
{
```

```
    int temp ;
```

```
    scanf ( "%d", &temp ) ;
```

```
    switch ( temp )
```

```
    {
```

```
        case ( temp <= 20 ) :
```

```
            printf ( "\nOooooooooohhhh! Damn cool!" ) ;
```

```
        case ( temp > 20 && temp <= 30 ) :
```

```
            printf ( "\nRain rain here again!" ) ;
```

```
        case ( temp > 30 && temp <= 40 ) :
```

```
            printf ( "\nWish I am on Everest" ) ;
```

```
        default :
```

```
            printf ( "\nGood old nagpur weather" ) ;
```

```
    }
```

```
}
```



```
main( )
```

```
{
```

```
    float a = 3.5 ;
```

```
    switch ( a )
```

```
    {
```

```
        case 0.5 :
```

```
            printf ( "\nThe art of C" ) ;
```

```
        break ;
```

```
        case 1.5 :
```

```
            printf ( "\nThe spirit of C" ) ;
```

```
        break ;
```

```
        case 2.5 :
```

```
            printf ( "\nSee through C" ) ;
```

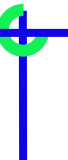
```
        break ;
```

```
        case 3.5 :
```

```
            printf ( "\nSimply c" ) ;
```

```
    }
```

```
}
```



```
main( )
{
    int a = 3, b = 4, c ;
    c = b - a ;
    switch ( c )
    {
        case 1 || 2 :
            printf ( "God give me an opportunity to change
things" ) ;
            break ;

        case a || b :
            printf ( "God give me an opportunity to
run my show" ) ;
            break ;
    }
}
```



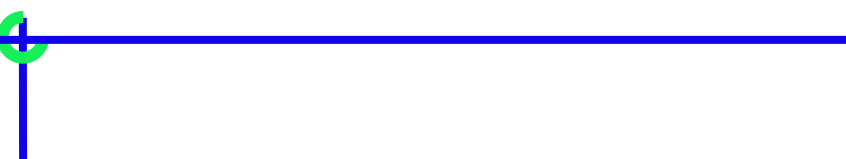
➤ Write a menu driven program which has following options:

- 1. Sum of all the numbers
- 2. Prime or not
- 3. Odd or even

Make use of switch statement.

<https://ideone.com/CP6cYU>

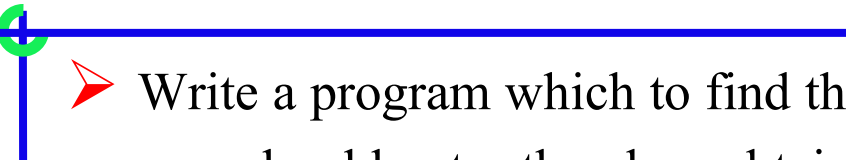
Exercise?



Write a program that takes name of animals (dog, cat, mouse) as input, and subsequently print sound they produce while barking.

- Create using if-else
- Create using switch-case-default

Home Assignment

- 
- Write a program which to find the grace marks for a student using switch. The user should enter the class obtained by the student and the number of subjects he has failed in.
- – If the student gets first class and the number of subjects he failed in is greater than 3, then he does not get any grace. If the number of subjects he failed in is less than or equal to 3 then the grace is of 5 marks per subject.
 - – If the student gets second class and the number of subjects he failed in is greater than 2, then he does not get any grace. If the number of subjects he failed in is less than or equal to 2 then the grace is of 4 marks per subject.
 - – If the student gets third class and the number of subjects he failed in is greater than 1, then he does not get any grace. If the number of subjects he failed in is equal to 1 then the grace is of 5 marks per subject

Solve using both if-else & switch-case-default