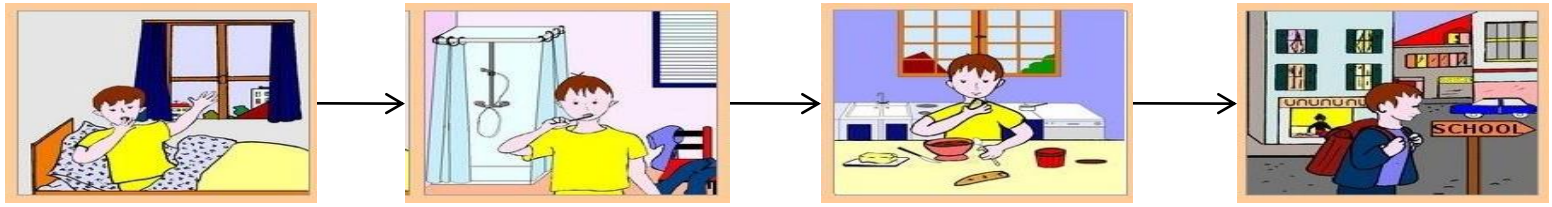
CONTROL STRUCTURES

Outline

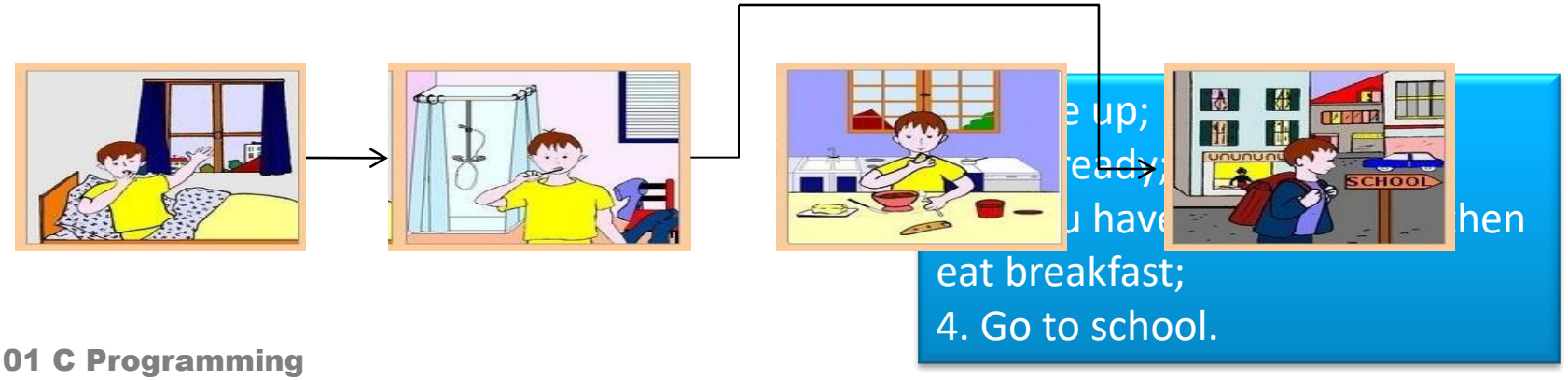
- Control structure
  - Decision Statements
    - If statement
    - If-else statement
    - Switch statement

Program

- Program is a set of instruction executed one by one.



- Depending upon the circumstances sometimes it is desirable to alter the sequence of execution of statements.

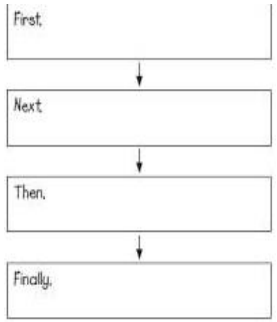


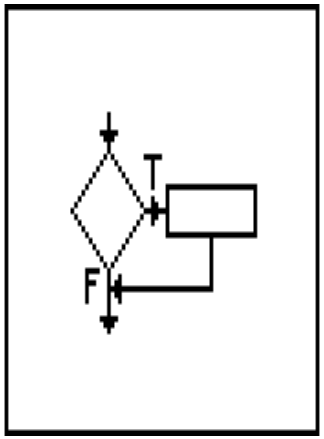
Control Statements

- The C language programs, until now follows a sequential form of execution of statements.
- C language provides statements that can alter the flow of a sequence of instructions. These statements are called *control statements*.
- These statements help to jump from one part of the program to another. The control transfer may be conditional or unconditional.

Control Structure

- A control structure refers to the way in which the programmer specifies the order of executing the statements.
- Three control structures
  - Sequence structure
    - Programs are executed sequentially by default
  - Selection structures
    - if, if…else, switch
  - Repetition structures (iteration)
    - while, do…while, for

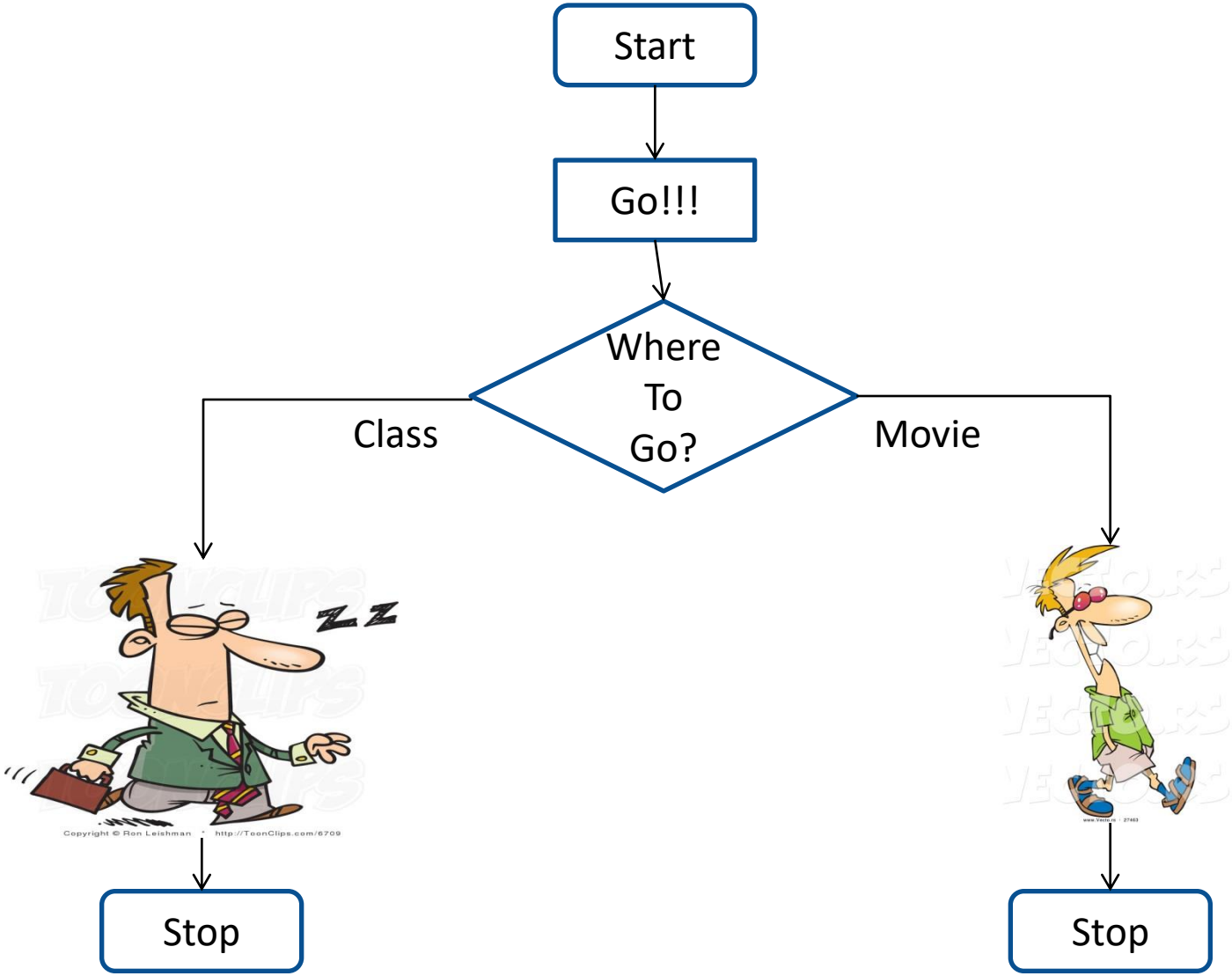




Condition Statements

- The C condition statements or the decision statements are one which checks the given condition
- Based upon the state of the condition, a sub-block is executed.
- Decision statements are the:
  - *if statement*
  - *if-else statement*
  - *switch statement*

Daily routine



if statement

If you have time?

Yes

No

Quick yak:
Quiz upon for more sequence jumping procedures followed in daily life like:

- Jumping a queue
- Not attending a class

Based on conditions

`if` Statement

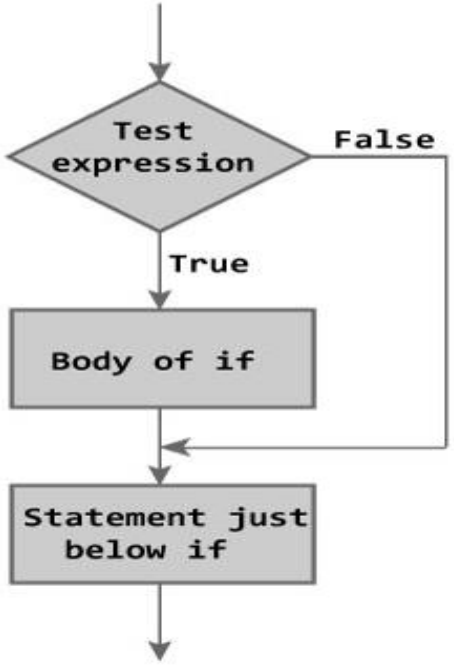
- If statement
  - It is decision making statement uses keyword *if*.
  - It allows the computer to evaluate the expression first and then, depending on whether the value is 'true' or 'false', i.e. non zero or zero it transfers the control to a particular statement.

A decision can be made on any expression.
zero - `false`
nonzero - `true`
Example:
`3 < 4` is `true`

if Statement

Syntax

```
if (expression)
statement;

or

if (expression)
{
  block of statements;
}
```



if Statement

- The *if statement* has the following syntax:

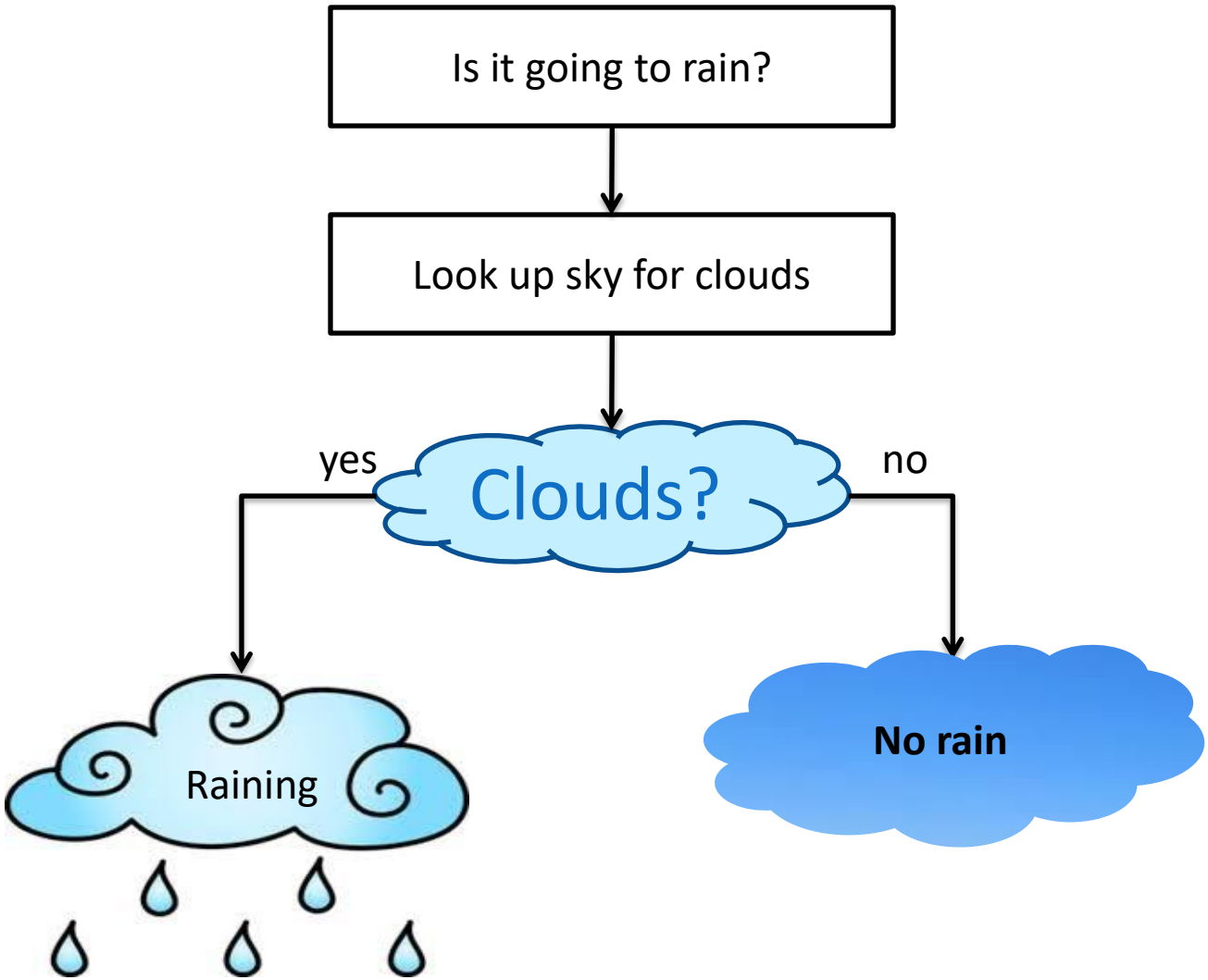
`if` is a C reserved word

The *`condition`* must be a boolean expression. It must Evaluate to either non-zero or zero.

```
if ( condition )/* no semi-colon */
    statement;
```

If the *`condition`* is non-zero, the *`statement`* is executed.
If it is zero, the *`statement`* is skipped.

Rain ???


Is it going to rain?
Look up sky for clouds
Clouds?
yes
no
Raining
No rain

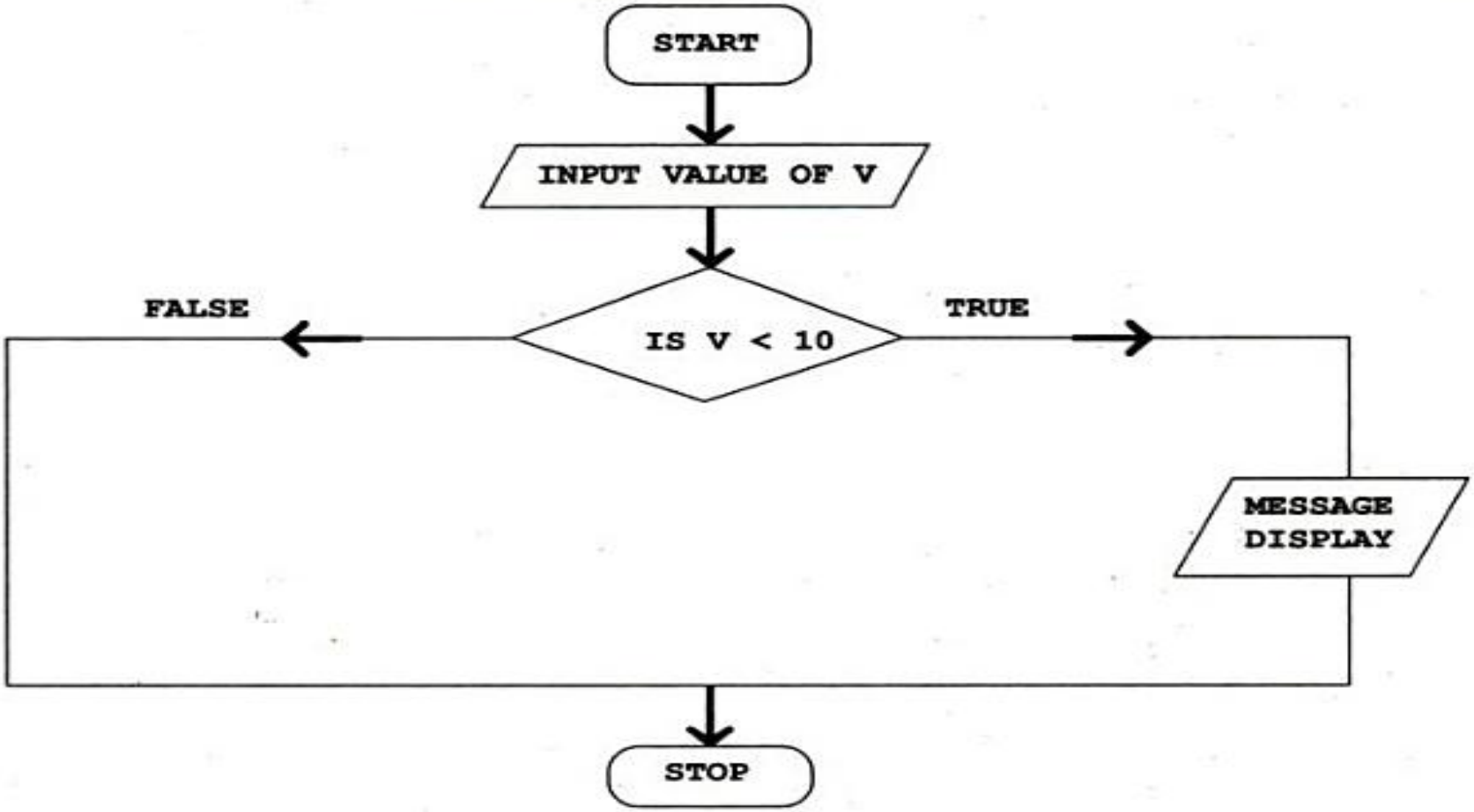
Program to check whether number is less than 10.

```
#include<stdio.h>
int main()
{
 int v;
 printf("Enter the number :");
 scanf("%d", &v);
 if(v<10)
   printf("number is less  than 10");
}
```

OUTPUT:

```
Enter the number: 6
Number is less than 10
```

Control Flow



`if..else` statement

If you have time?

Yes

No

Grab something to eat along

If..else statement

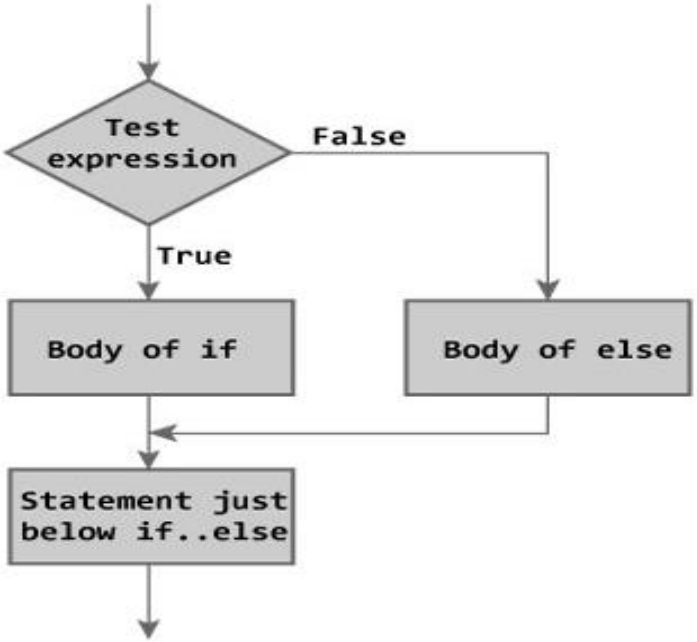
- The `if` statement executes only when the condition following if is true.
- It does nothing when the condition is false.
- The `if..else` statement takes care of the true and false conditions.

`if..else` statement

- `if..else` has two blocks.
- One block is for `if` and it is executed when condition is **non-zero**(true).
- The other block is of `else` and its executed when condition is **zero** (false).

Syntax

```
if (expression)
{
  block of statements;
}
else
{
  block of statements;
}
```



`if..else` statement

- The `else` statement cannot be used without `if`.
- No multiple `else` statements are allowed with one `if`.
- `else` statement has no expression.
- Number of `else` cannot be greater than number of `if`.

Ternary conditional operator (? :)

- C code:

```
if ( marks>= 60 )
   printf( "Pass\n");
else
   printf( "Fail\n");
```

- Same code using **ternary operator**:
  - Takes three arguments (condition, value if `true`, value if `false`)
  - Our code could be written:

```
printf("%s\n", marks>= 60 ? "Pass" : "Fail");
```

  - Or it could have been written:

```
marks >= 60 ? printf("Pass\n") :
  printf("Fail\n");
```

Program to check whether number is less than 10.

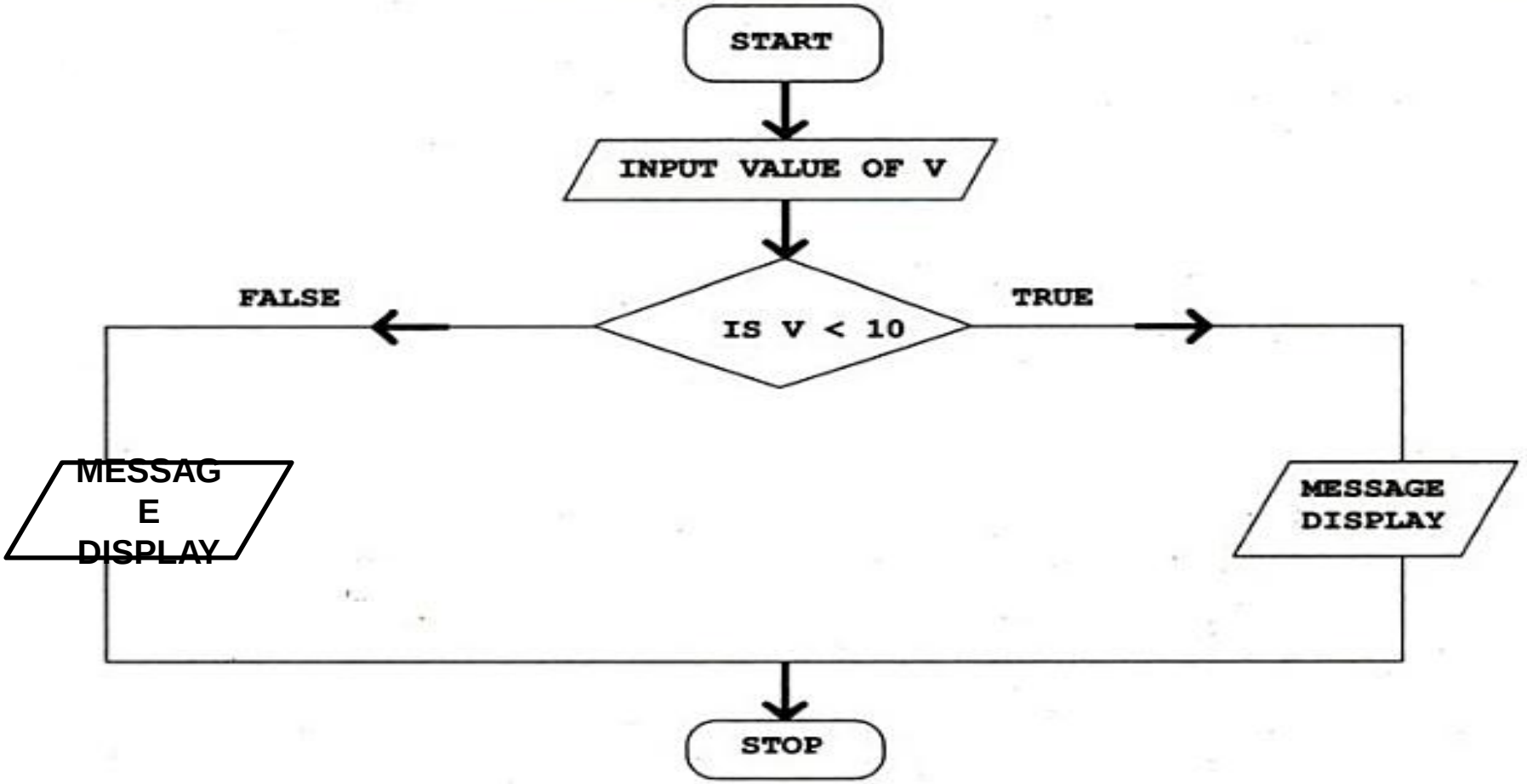
```
#include<stdio.h>
void main()
{
 int a;
 printf("Enter the number :");
 scanf("%d", &v);
 if(v<10)
   printf("number is less than 10");
 else
   printf("number is greater than 10");
}
```

```
Enter the number: 7
Number is less than 10
```

or

```
Enter the number: 100
Number is greater than 10
```

Control Flow



Nested *if..else*

- In `if..else` statement, else block is executed by default after failure of if condition.
- The nested `if...else` statement is used when program requires more than one test expression.
- Test for multiple cases by placing `if…else` selection statements inside `if…else` selection statement.
- This kind of nesting will be unlimited.

Nested `if..else`

Syntax

```
if ( condition 1 )
{
    block of statements;
}
else if  ( condition 2 )
{
    block of statements;
}
else {
    block of statements;
}
```

Program to check whether number is less than 10.

```
#include<stdio.h>
void main()
{
 int a;
 printf("Enter the number :");
 scanf("%d", &v);
 if(v<10){
   printf("number is less than 10");
 }
 else if(v<100){
   printf("number is less than 100");
 }
}
```

```
Enter the number: 1
Number is less than 10
```

or

```
Enter the number: 56
Number is less than 100
```

Forms of `if`

The **if** statement can take any of the following forms:

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

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

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

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

Program to print grades of students marks.

```
#include<stdio.h>
void main( )
{
 float marks;
 scanf("%f", &marks);
 if (marks>90){
          printf("Grade A");
        }
 else  if (marks>80)
{
           printf("Grade B");
     }
 else  if(marks>70)
{
      printf("Grade C");
     }
 else if (marks >60)
{
      printf("Grade D");
     }
}
```

```
66.70
Grade D
```

or

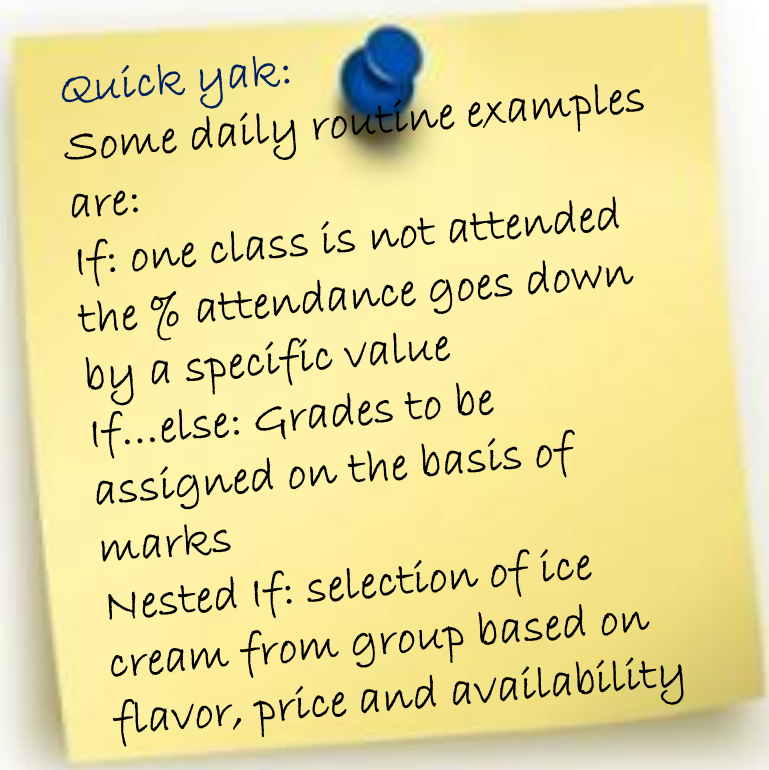
```
78.00
Grade C
```

Forms of `if`

Decision control statements	Syntax
`if`	`if (condition)` `  Statements;`

`if…else`	`if (condition)`
	`  Statement1;`
	`  Statement2;}`
	`else {`
	`  Statement3;`
	`  Statement4;}`

Nested `if`	`if (condition1`
	`  Statement1;}`
	`else`
	`if(condition2){`
	`  Statement2;}`
	`else`
	`  Statement 3;`



else part is executed.

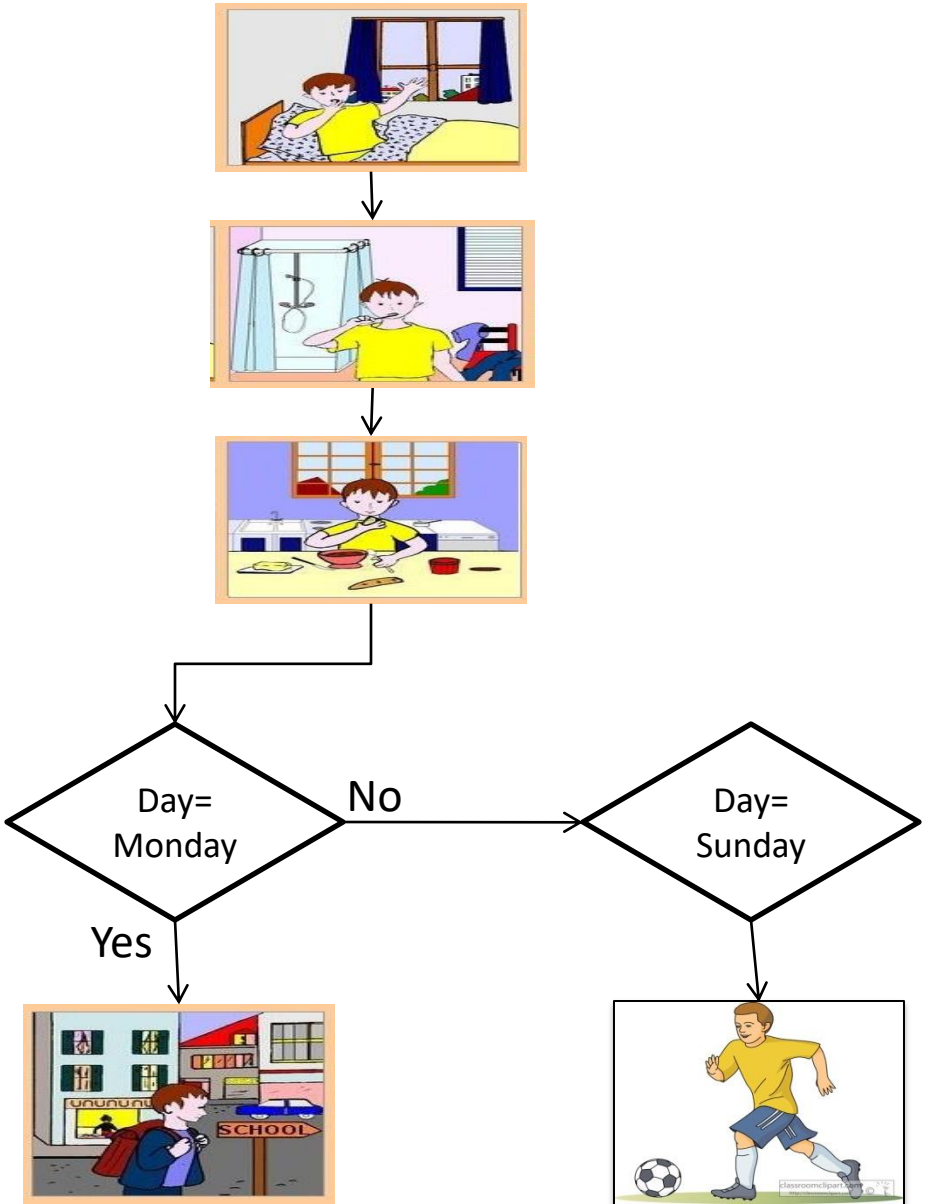
`break` statement

- *`break`* is a **keyword**.
- `break` allows the programmer to *terminate the loop*.
- A `break` statement causes control to transfer to the next statement after the loop or block.
- The `break` statement can be used in nested loops. If we use `break` in the innermost loop then the control of the program is terminated only from the innermost loop.

How break statement works?

```
while (test Expression)
{
    // codes
    if (condition for break)
    {
        break;
    }
    // codes
}
```

`switch` Statement



`switch` Statement

- The control statement that allows to make a decision from the number of choices is called switch.
- Also called switch-case-default.
- The switch statement provides another way to decide which statement to execute next.
- *The switch statement evaluates an expression, then attempts to match the result to one of several possible cases.*
- Each case contains a value and a list of statements.
- The flow of control transfers to statement associated with the first case value that matches.

- When a **break** statement is reached, the switch terminates, and the flow of control jumps to the next line following the switch statement.
- Not every case needs to contain a **break**. If no **break** appears, the flow of control will *fall through* to subsequent cases until a break is reached.
- A **switch** statement can have an optional default case, which must appear at the end of the switch. The default case can be used for performing a task when none of the cases is true. No **break** is needed in the default case.

switch Statement

Syntax

```
switch (expression)
{
case constant1:
        statements;
        break;
case constant2:
        statements;
        break;
case constant3:
        statements;
        break;
default:
        statements;
}
```

switch and case are reserved words

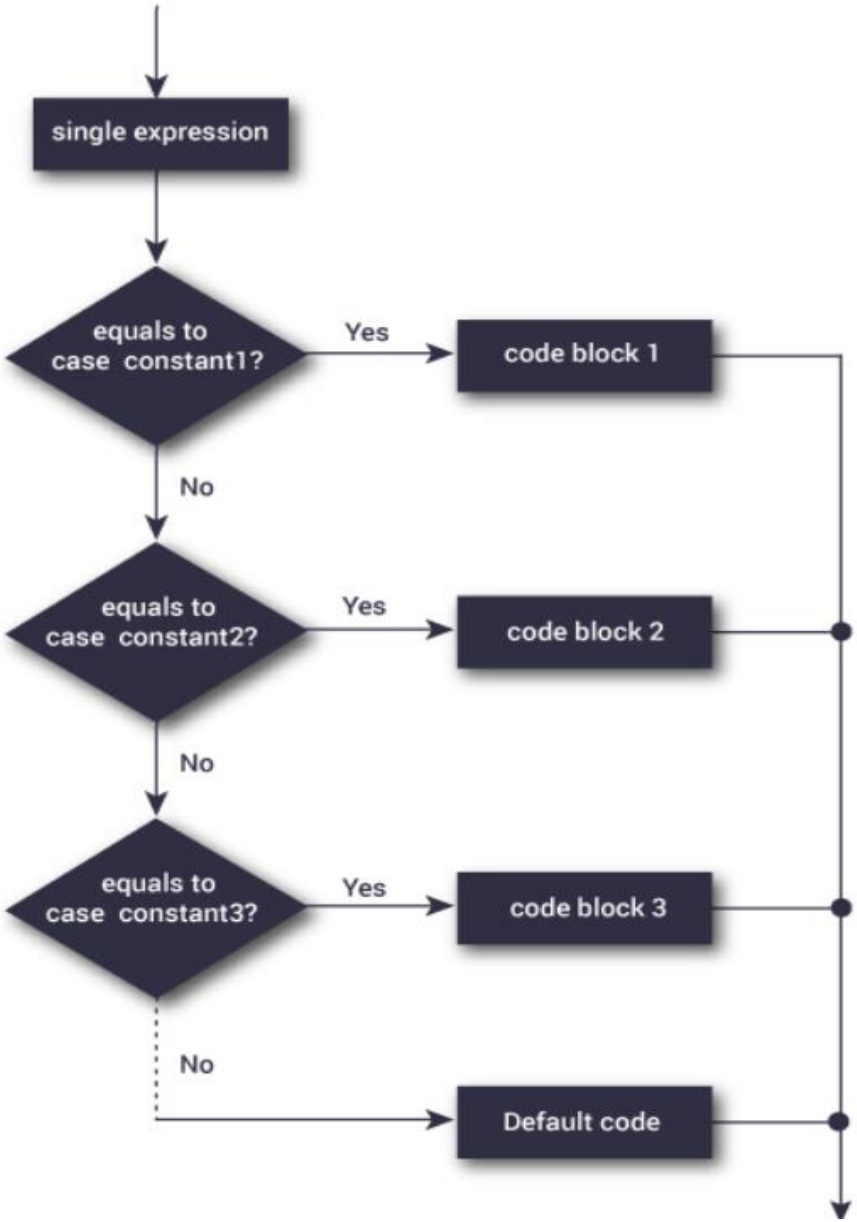
```
switch ( expression )
{
   case value1 :
      statement-list1
   case value2 :
      statement-list2
   case value3 :
      statement-list3
   case  ...

}
```

If *expression* matches *value2*, control jumps to here

Rules of using `switch case`

1. Case label must be unique
2. Case label must end with colon
3. Case label must have constant expression
4. Case label must be of integer, character type like case 2, case 1+1, case ‘a’
5. Case label should not be floating point
6. Default can be placed anywhere in switch
7. Multiple cases cannot use same expression
8. Relational operators are not allowed in switch
9. Nesting of switch is allowed.
10. Variables are not allowed in switch case label.

Switch workflow



Syntax error in `switch` statement

```
switch(pt){
      case count:
       printf("%d", count);
       break;
      case 1<8:
       printf("A point");
       break;
      case 2.5:
       printf("A line");
       break;
      case 3 + 7.7:
       printf("A triangle");
      case 3 + 7.7:
       printf("A triangle");
       break;
      case count+5:
       printf("A pentagon");
       break;
}
```

Variable cannot be used as label

Relational operators are not allowed

Floating point number cannot be used

Floating point number cannot be used and same expression cannot be used

constant expression should be used

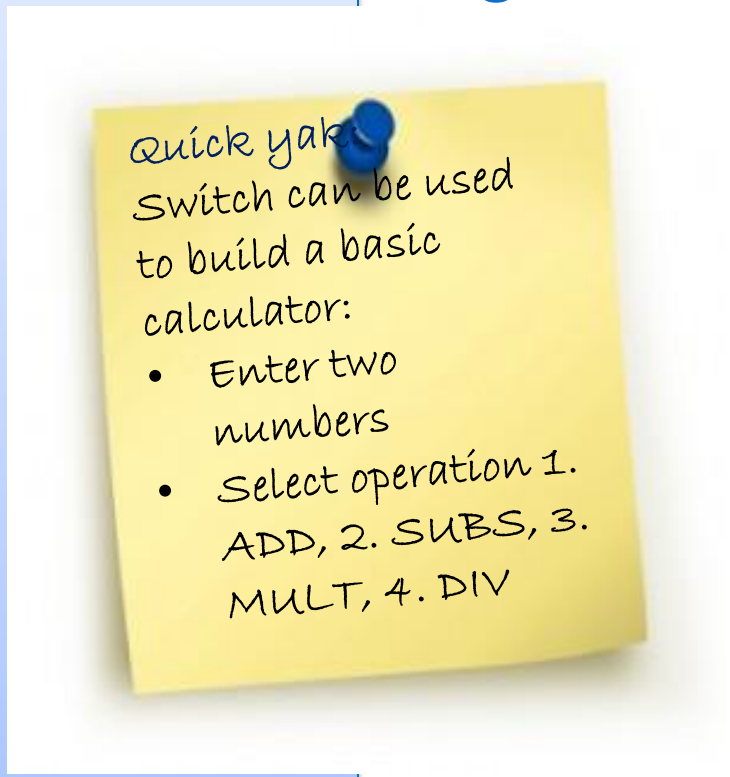
Program to show switch statement in geometry

```
#include<stdio.h>
void main()
{
   int pt;
   printf("Enter the number of nodes:");
   scanf("%d", &pt);
   switch(pt){
     case 0:
      printf("\nNo Geometry");
     break;
     case 1:
      printf("\nA point");
     break;
     case 2:
      printf("\nA line");
     break;
     case 3:
      printf("\nA triangle");
     break;
     case 4:
      printf("\nA rectangle");
     break;
     case 5:
      printf("\nA pentagon");
     break;
    default:
     printf("Invalid input");
    }
}
```

```
Enter the number of nodes: 2
A line
```

Program to in

```
#include<stdio.h>
void main()
{
   int key;
   printf("Press 1 to turn left.");
   printf("Press 2 to turn right.");
   printf("Press 3 to increase speed.");
   printf("Press 4 for break: ");
   scanf("%d", &key);
   switch(key){
     case 1:
       printf("\nTurn left");
     break;
     case 2:
       printf("\nTurn right");
     break;
     case 3:
       printf("\nIncrease speed");
     break;
     case 4:
      printf("\nBreak");
     break;
     default:
       printf("Invalid input");
   }
```




```
Press 1 to turn left.
Press 2 to turn right.
Press 3 to increase speed.
Press 4 for break: 4
Break
```

Difference between switch and if:

- *if* statements can evaluate float conditions; *switch* statements cannot evaluate float conditions.
- *if* statement can evaluate relational operators, *switch* statement cannot evaluate relational operators i.e. they are not allowed in switch statement.

Next Class: Loop control and Jump statements

cse101@lpu.co.in