CS101 Introduction to computing

Recap of Floating Point
&
Control Flow of Program

A. Sahu and S. V .Rao

Dept of Comp. Sc. & Engg.

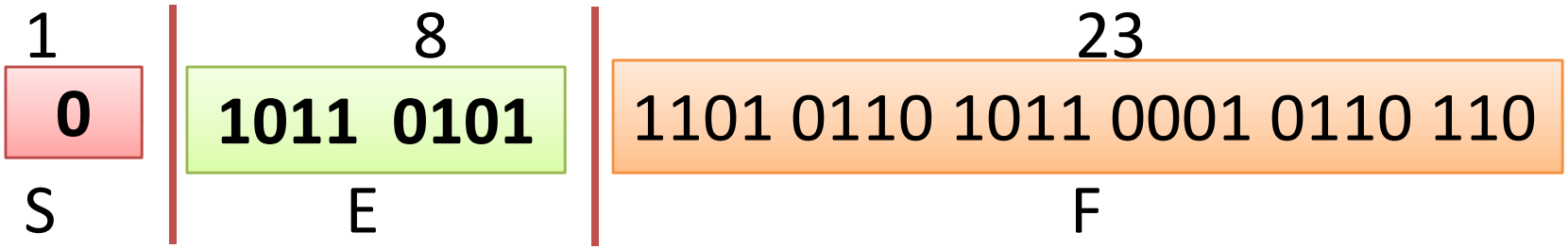
Indian Institute of Technology Guwahati

Outline

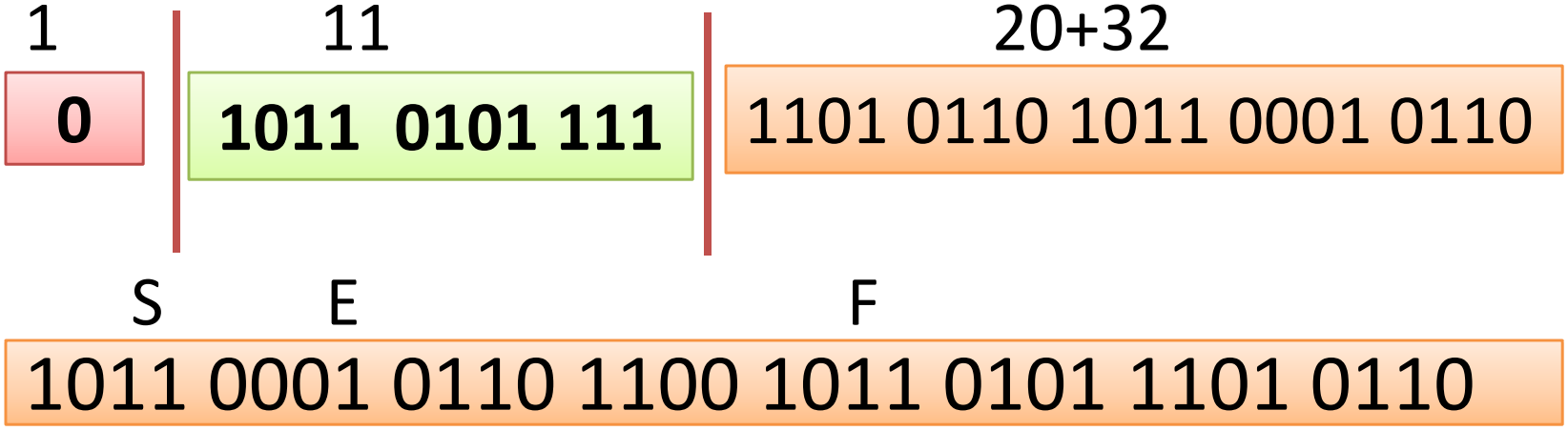
- Recap
  - Operation on Floating Point
  - Conversions and type casting in C
- Program flow control
  - If-else
  - Switch case
  - Looping : while, for, do-while
- Problem Solving

IEEE 754 standard

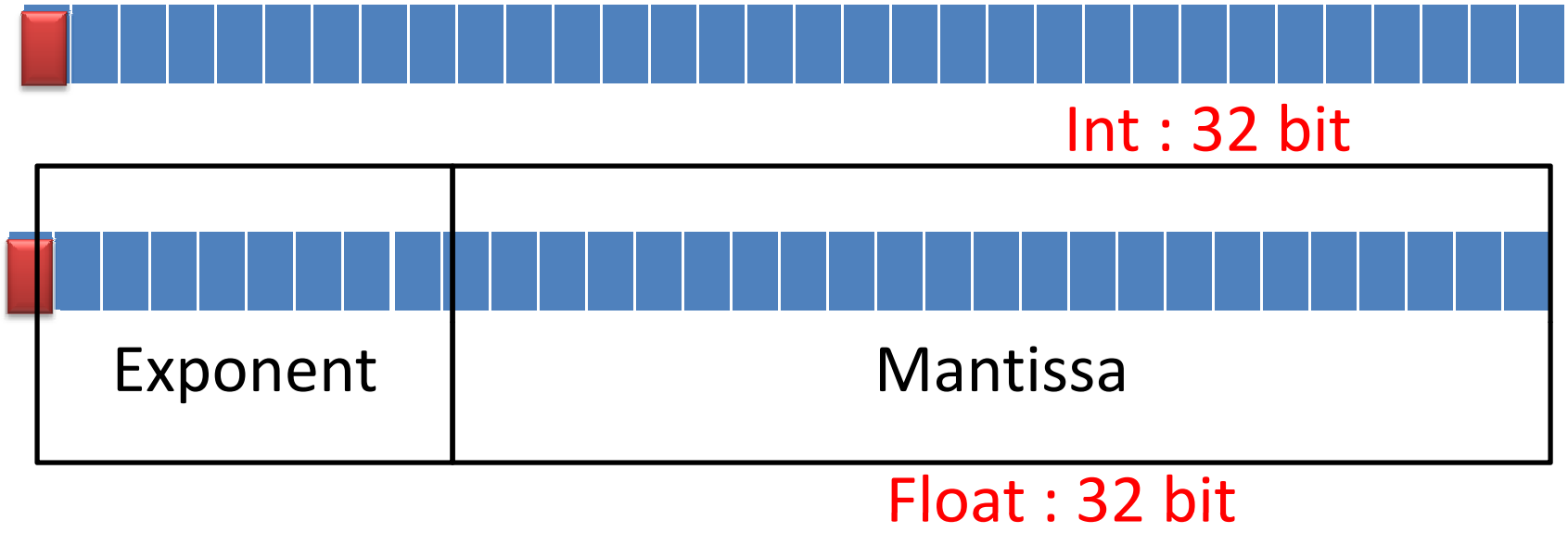
- Single precision numbers



- Double precision numbers

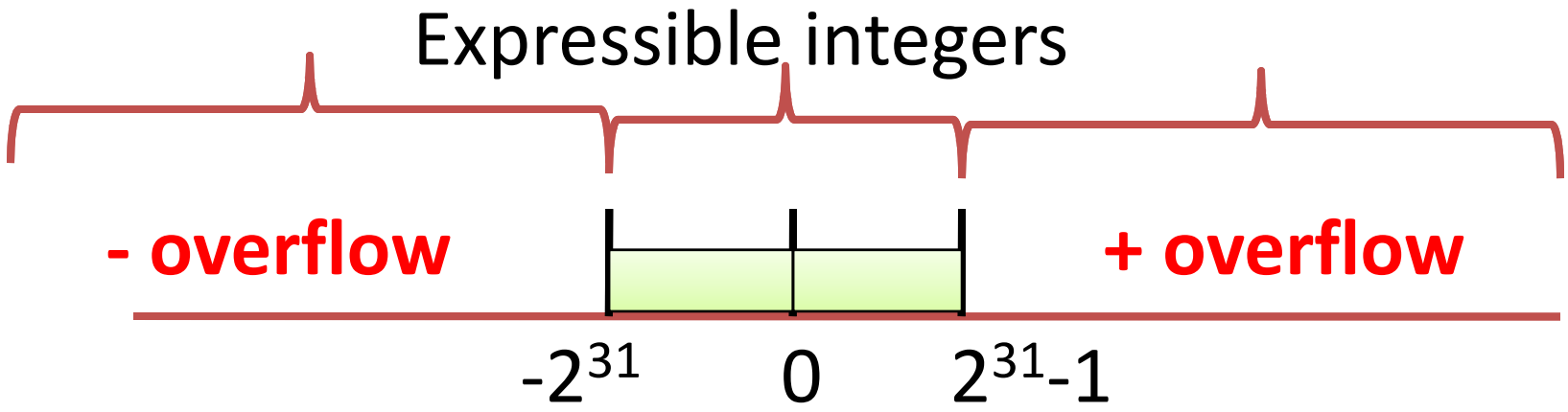


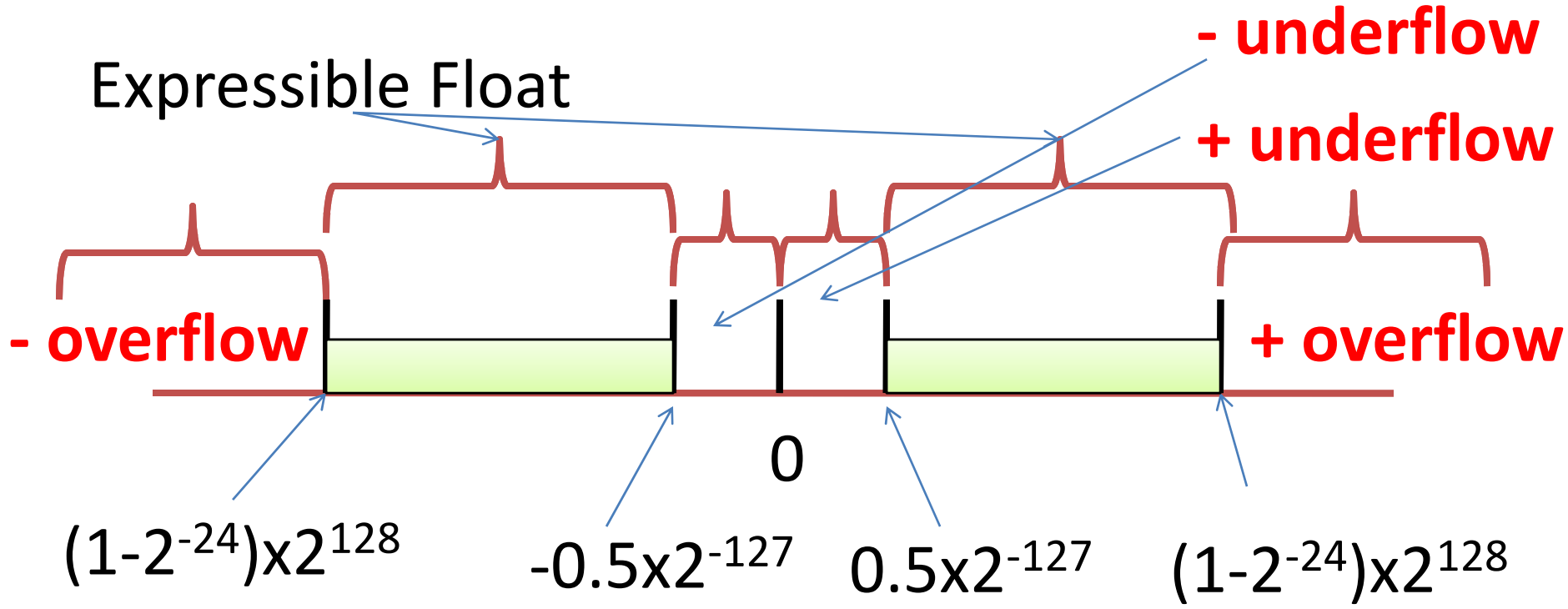
Density of int vs float



- Number of number can be represented
  - Both the cases (float, int) : 2^{32}
- Range
  - int (-2^{31} to $2^{31}-1$)
  - float Large $\pm(2 - 2^{-23}) \times 2^{127}$ **Small** $\pm 1 \times 2^{-126}$
- 50% of float numbers are **Small (**less then $\pm$**1**)

Expressible Numbers(int and float)



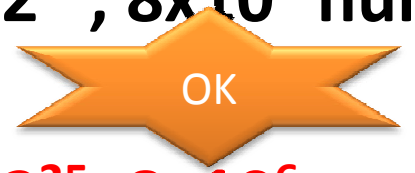


Density of 32 bit float SP

- Fraction/mantissa is 23 bit
- Number of different number can be stored for particular value of exponent
  - Assume for exp=1, $2^{23}=8x1024x1024 \approx 8x10^6$
  - Between 1-2 we can store $8x10^6$ numbers
- Similarly
  - for exp=2, between 2-4, $8x10^6$ number of number can be stored
  - for exp=3, between 4-8, $8x10^6$ number of number can be stored
  - for exp=4, between 8-16, $8x10^6$ number of number can be stored

Density of 32 bit float SP

- Similarly
  - for exp=23, between 2^{22}-2^{23}, $8x10^6$ number of number can be stored
  - **for exp=24, between 2^{23}-2^{24}, $8x10^6$ number of number can be stored**

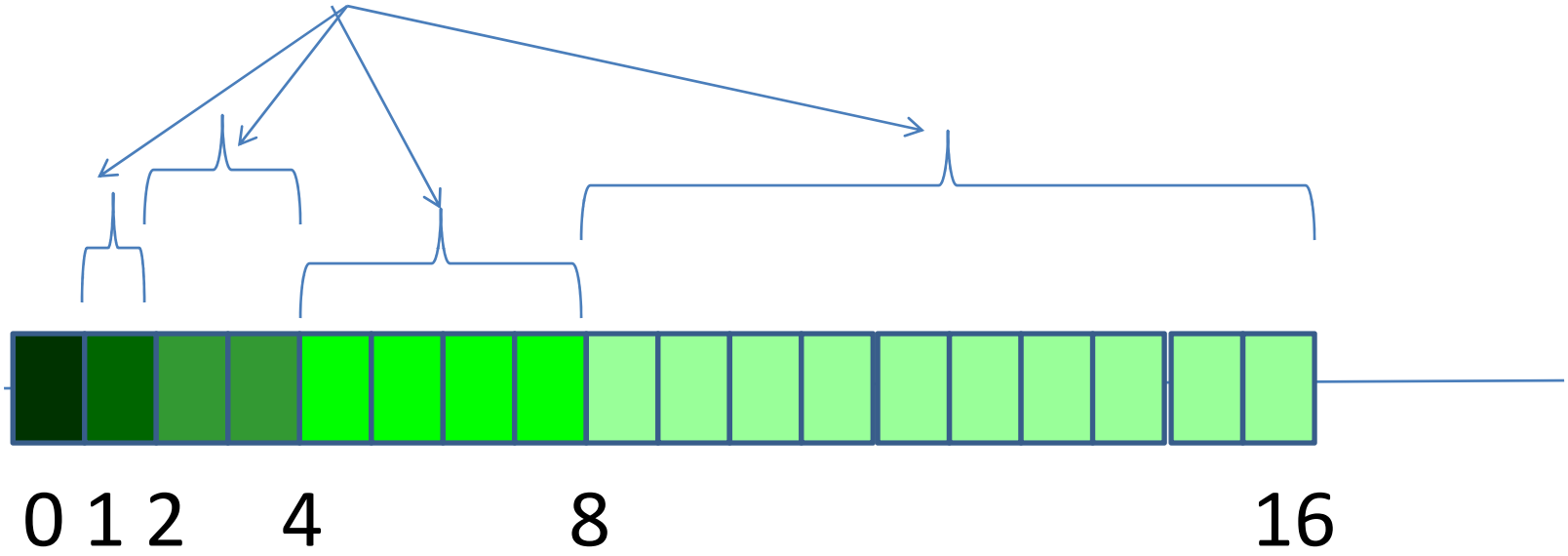


  - **for exp=25, between 2^{24}-2^{25}, $8x10^6$ number of number can be stored**
    - **2^{24}-2^{25} >8 $x10^6$**

  - …
  - for exp=127, between 2^{126}-2^{127}, $8x10^6$ number of number can be stored

Density of 32 bit float SP

- $2^{23}=8x1024x1024 \approx 8x10^6$



Floating Point in C

- C Guarantees Two/Three Levels

`float`	single precision
`double`	double precision
`long double`	quad precision

Mathematical Properties of FP Add

- Compare to those of Abelian Group
  - Closed under addition? YES
    - But may generate infinity or NaN
  - Commutative? YES
  - Associative? NO
    - Overflow and inexactness of rounding
  - 0 is additive identity? YES
  - Every element has additive inverse ALMOST
    - Except for infinities & NaNs
- Monotonicity
  - $a \geq b \Rightarrow a+c \geq b+c$? ALMOST
    - Except for infinities & NaNs

Math. Properties of FP Mult

- Compare to Commutative Ring
  - Closed under multiplication? YES
    - But may generate infinity or NaN
  - Multiplication Commutative? YES
  - Multiplication is Associative? NO
    - Possibility of overflow, inexactness of rounding
  - 1 is multiplicative identity? YES
  - Multiplication distributes over addition? NO
    - Possibility of overflow, inexactness of rounding
- Monotonicity
  - $a \geq b \;\&\; c \geq 0 \Rightarrow a * c \geq b * c$? ALMOST
    - Except for infinities & NaNs

Type Casting

- Some languages are strictly type, addition of two different type of object
  - Result an compilation error (Example ML)
  - You can not add **a mango** and **an apple**
- In C we have :
  - auto up gradation and demotion (Implicit type casting) of type
  - Explicit type casting

```
int I;  float  F;
I=F;    //auto demotion
F=I;   //auto promotion
I=(int) F;  //manual demotion
F=(float) I; //manual promotion
```

Type Casting Floating Point

- Casting between `int`, `float`, and `double` changes numeric values
- `double` or `float` to `int`
  - Truncates fractional part
  - Like rounding toward zero
  - Not defined when out of range
    - Generally saturates to TMin or TMax
- `int` to `double`
  - Exact conversion, as long as int has ≤ 53 bit word size
- `int` to `float`
  - Will round according to rounding mode

Answers to Floating Point Puzzles

```
int x = …;

float f = …;

double d = …;
```

Assume neither `d` nor `f` is NAN

- `x==(int)(float)x` Comparison Result?
- `x==(int)(double)x` Comparison Result?
- `f==(float)(double)f` Comparison Result?
- `d==(float)d` Comparison Result?
- `f==-(-f);` Comparison Result?

Answers to Floating Point Puzzles

```
int x = …;

float f = …;

double d = …;
```

Assume neither `d` nor `f` is NAN

- `x==(int)(float)x` — No: 24 bit Fraction (1+23)
- `x==(int)(double)x` — Yes: 53 bit Fraction(1+52)
- `f==(float)(double)f` — Yes: increases precision
- `d==(float)d` — No: loses precision
- `f==-(-f);` — Yes: Just change sign bit

Answers to Floating Point Puzzles

```
int x = …;

float f = …;

double d = …;
```

Assume neither `d` nor `f` is NAN

- `2/3 == 2/3.0` Comparison Result?
- `d < 0.0` ⇒ `((d*2)<0.0)` Comparison Result?
- `d > f` ⇒ `-f > -d` Comparison Result?
- `d * d >= 0.0` Comparison Result?
- `(d+f)-d == f` Comparison Result?

Answers to Floating Point Puzzles

```
int x = …;
float f = …;
double d = …;
```

Assume neither `d` nor `f` is NAN

- `2/3 == 2/3.0` — No: 2/3 == 0
- `d < 0.0 ⇒ ((d*2) < 0.0)` — Yes!
- `d > f ⇒ -f > -d` — Yes!
- `d * d >= 0.0` — Yes!
- `(d+f)-d == f` — No: Not associative

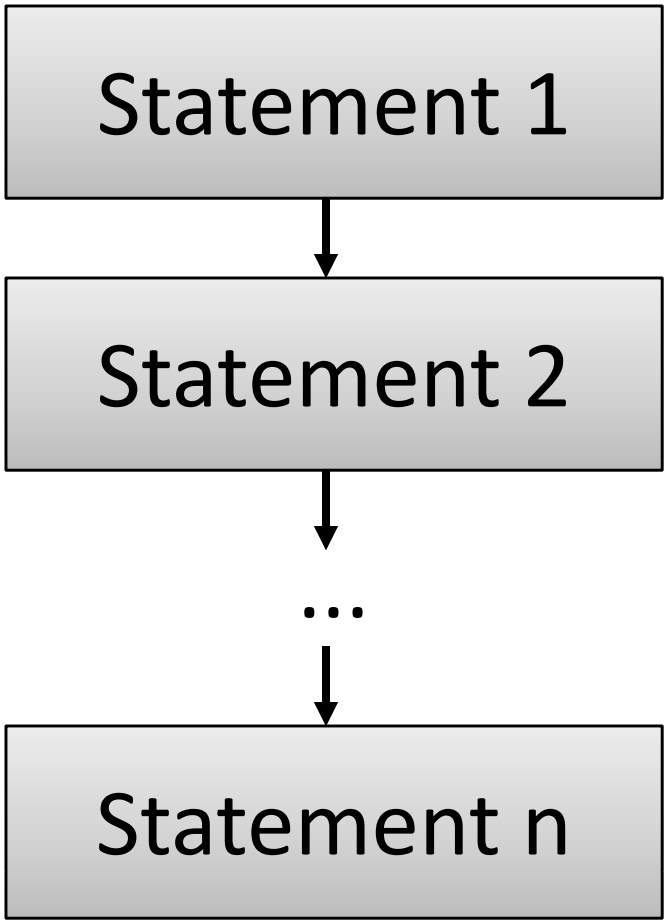
Control Flow of Program

if-else, if-goto, switch-case
&
for, while, do-while

Structured Programming

- All programs can be written in terms of only three control structures
  - **Sequence, selection and repetition**
- The **sequence** structure
  - Unless otherwise directed, the statements are executed in the order in which they are written.
- **The selection structure**
  - Used to choose among alternative courses of action.
- **The repetition structur**e
  - Allows an action to be repeated while some condition remains true.

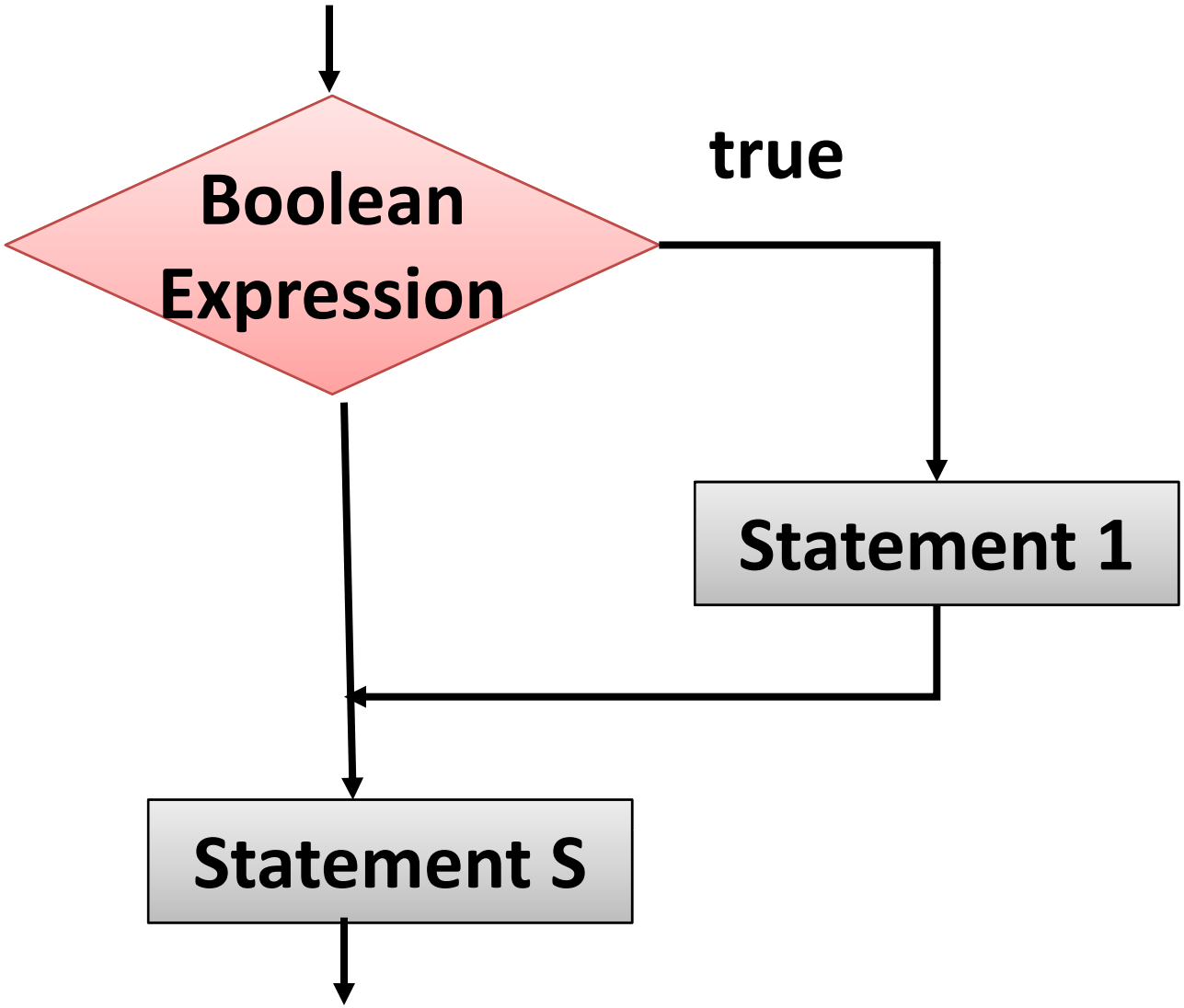
Sequential Execution



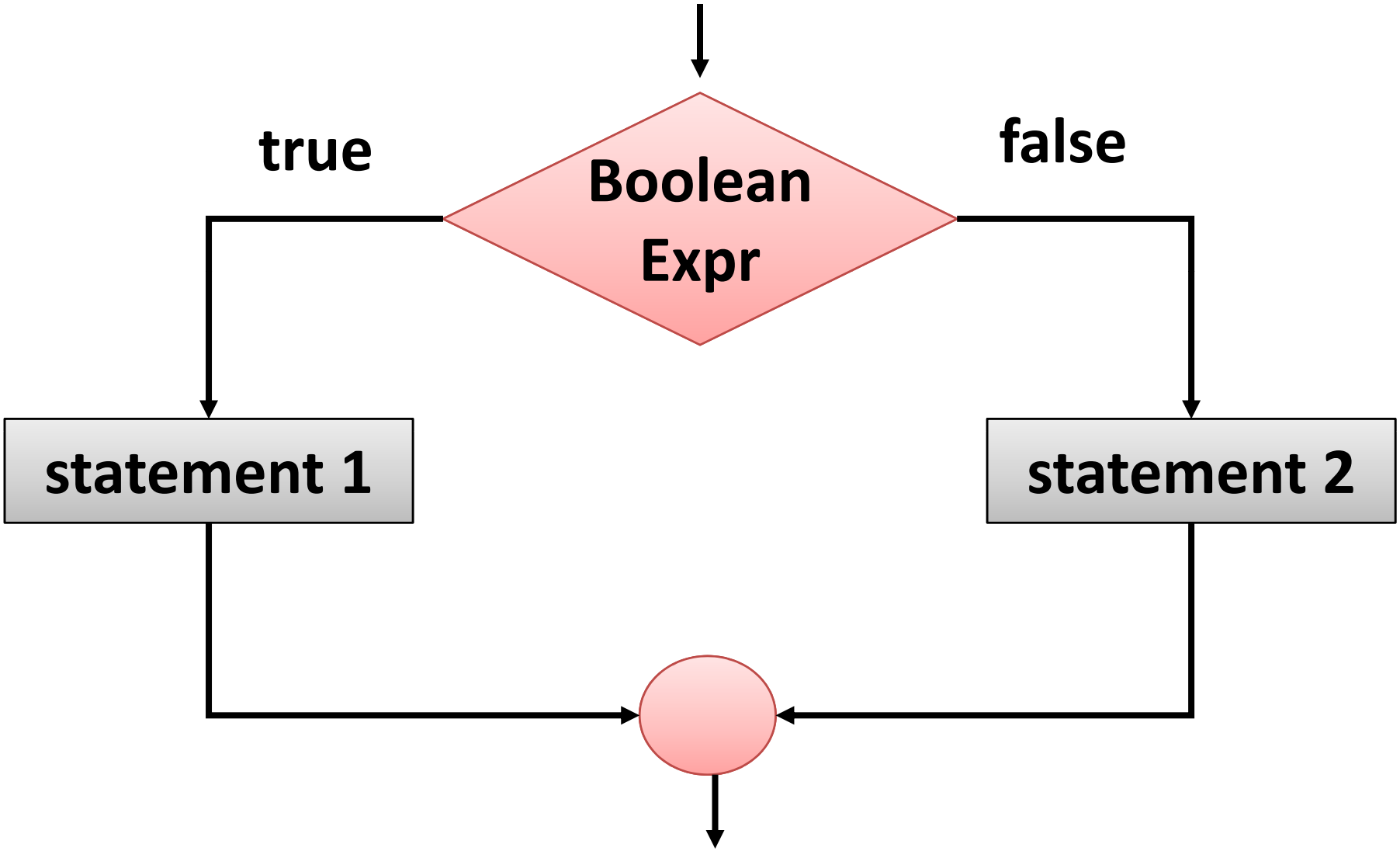
Compute the resonant frequency of an RLC circuit

```c
#include <stdio.h>
#include <math.h>
int main() {
  double l, c, omega, f;
  printf("Enter inductance in mH: ");        //S1
  scanf("%lf", &l);                          //S2
  printf("Enter capacitance in microF: ");   //S4
  scanf("%lf", &c);                          //S5
  omega = 1.0/sqrt((l.0/1000)*(c/1000000)); //S6
  f = omega / (2 * M_PI);                    //S7
  printf("Resonant freq: %.2f\n", f);        //S8
  return 0;                                  //S9
}
```

Selective Execution : Flow chart (only if)



Selective Execution : Flow chart



Selection: the if-else statement

```
if   ( condition ){
      statement(s)/*if clause */
}
else {
      statement(s)/*else clause */
}
```

Nesting of if-else Statements

```
if ( condition1 )
{
     statement(s)
}
else if ( condition2 )
{
    statement(s)
}
. . . /* more else clauses may be here */
else
{
    statement(s)  /* the default case */
}
```

Bad Example : 2 if 1 else

```
if ( n > 0 )
    if ( a > b  )
        z=a;
else
      z=b;
```

```
if ( n > 0 )
  if ( a > b )
      z=a;
  else
      z=b;
```

```
if ( n > 0 )
{
  if (a> b)
      z=a;
}
else
      z=b;
```

Indentation will not ensure result :

else match with closest if

Code of Red box behaves like Code of Green box

In Assembly language: No if-else

- Assembly language
  - No support for [if else, *No for loop, No while loop]*
  - All higher construct get implemented

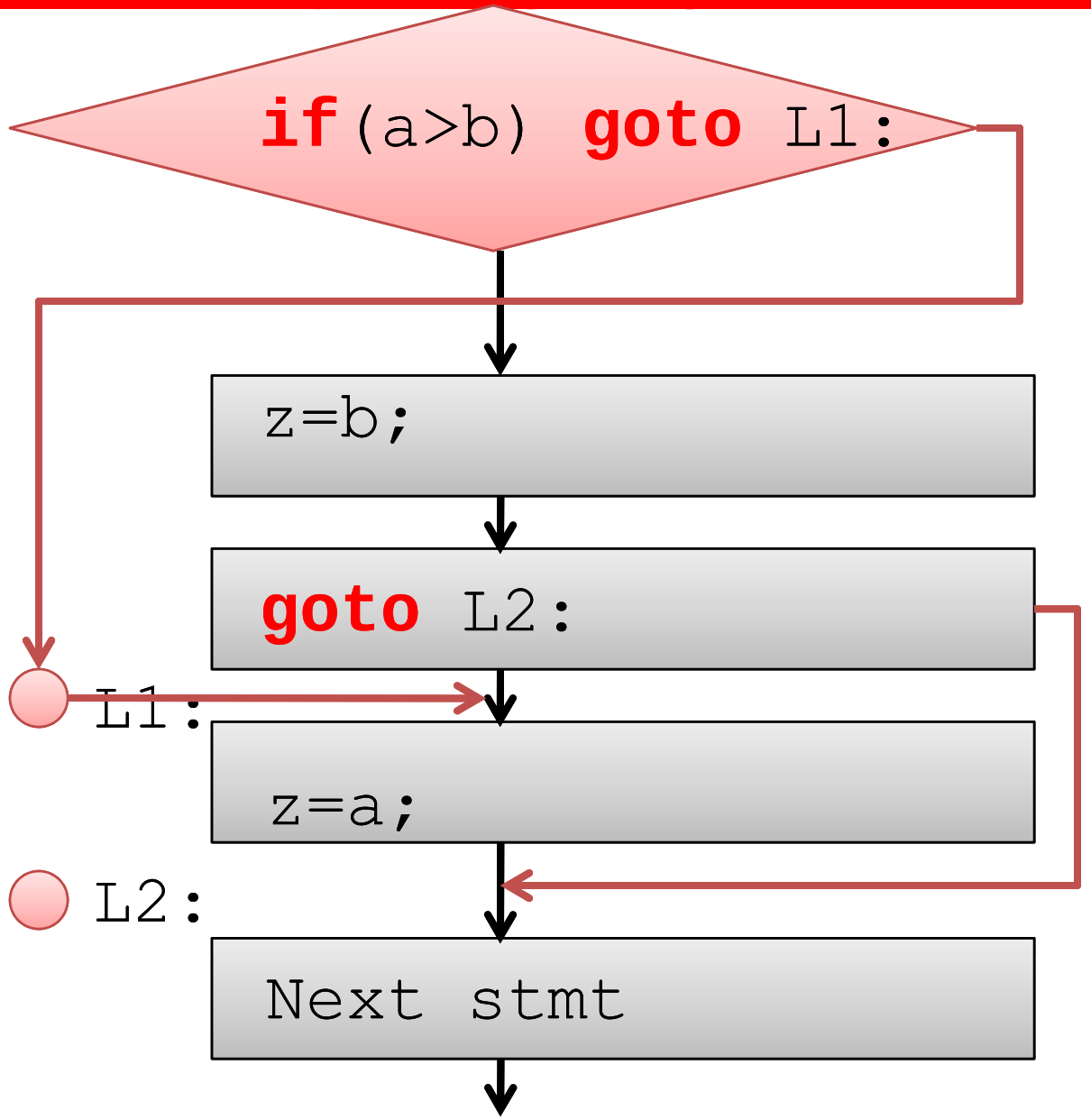
    using **if** and **goto** statement

    goto statement uses **Label**
- **If else** get converted to **if goto**

```
if(a> b )
     z=a;
else z=b;
NextStmt;
```

```
     if(a>b) goto L1:
     z=b;
     goto L2:
L1:  z=a;
L2:  Next stmt
```

In Assembly language: No if-else



Multi-way if else : switch case

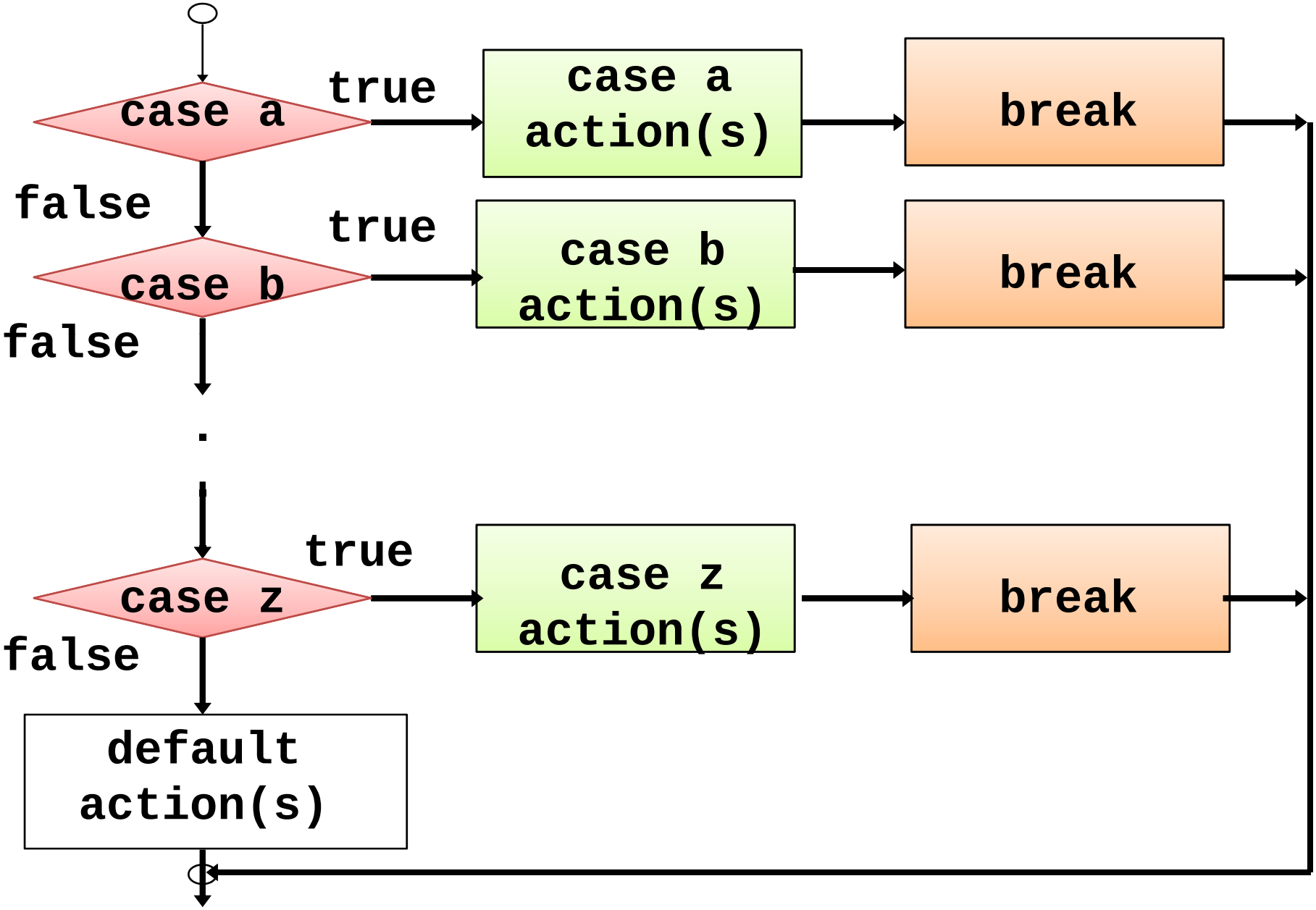
- If-else : two way, if part and else part
- To make it multi-way: nested if-else
  - Confusing, lengthy
- C language provide
  - Switch case
  - Multi way selection
  - Range multi-way selection

The switch Multiple-Selection Structure

- **`switch`**
  - Useful when expression is tested for multiple values
  - Consists of a series of **`case`** labels and an optional **`default`** case
  - **`break`** is (almost always) necessary

```
switch (<expression>){
  case <Value1> :
     <Action/Stmts for Value1>; break;
  case <Value2> :
     <Action/Stmts for Value2>; break;
  . . .
  default: <Action/Stmts for DefaultValue>;
           break;
}
```

Flowchart of Switch Statement



Multiway Switch Selection example

```
int main(){//simple calculator
 int a=50,b=10, R;
 char choice;
 printf("Enter choice");
 scanf("%c",&choice);

 switch (choice){
  case 'a' : R=a+b; printf("R=%d",R); break;
  case 's' : R=a-b; printf("R=%d",R); break;
  case 'm' : R=a*b; printf("R=%d",R); break;
  case 'd' : R=a/b; printf("R=%d",R); break;
  default  : printf("Wrong choice") ; break;
  }
 return 0;
}
```

Multiway Switch Selection example

```
int main(){//simple calculator
 int a=50,b=10, R;
 char choice;
 printf("Enter choice");
 scanf("%c",&choice);

 switch (choice){
  case 'a' : R=a+b; printf("R=%d",R); break;
  case 's' : R=a-b; printf("R=%d",R); break;
  case 'm' : R=a*b; printf("R=%d",R); break;
  case 'd' : R=a/b; printf("R=%d",R); break;
  default  : printf("Wrong choice") ; break;
  }
 return 0;
}
```

Multiway Switch Selection example

```
switch (choice){
 case 'A' : // no break, work for both A & a
            // next statement automatically
            // get executed
 case 'a' : R=a+b; printf("R=%d",R); break;
 case 'S' :
 case 's' : R=a-b; printf("R=%d",R); break;
 case 'M' :
 case 'm' : R=a*b; printf("R=%d",R); break;
 case 'D' :
 case 'd' : R=a/b; printf("R=%d",R); break;
 default  : printf("Wrong choice") ; break;
 }
```

Range Multiway Switch Selection example

```
int x;
scanf("%d",&x);
switch (x){
 case 1 ... 20:// 1 space three dots space 20
    printf("You entered >=1 and <=20");
    break;
 case 21 ... 30 :
     printf("You entered >=21 and <=30");
    break;
 default  :
    printf("You entered < 1 and >31") ;
    break;
 }
```

Syntax = case <low_range> ... <high_range> :

Thanks