PDS Lab

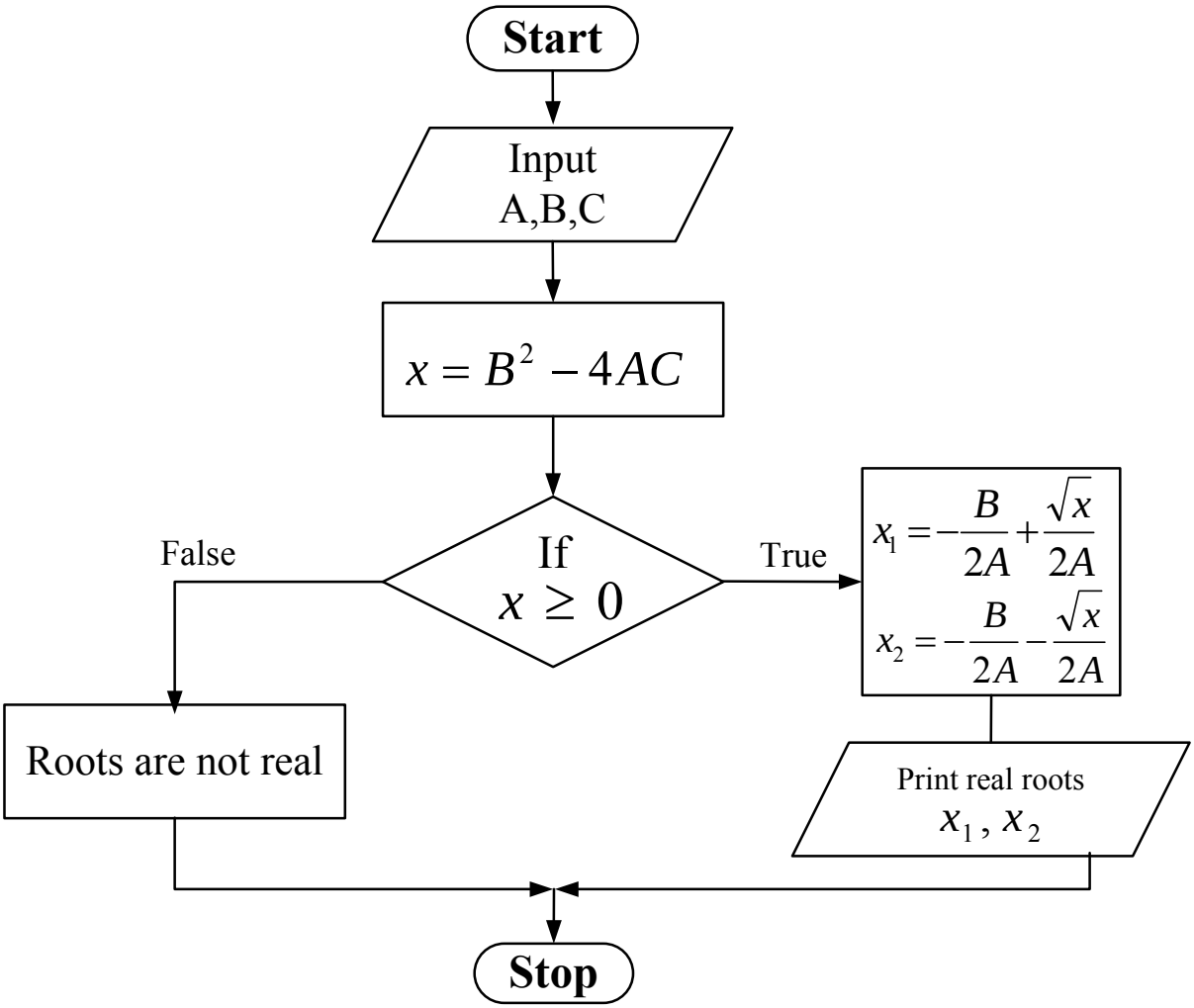
Section 16

Autumn-2017

Tutorial 3

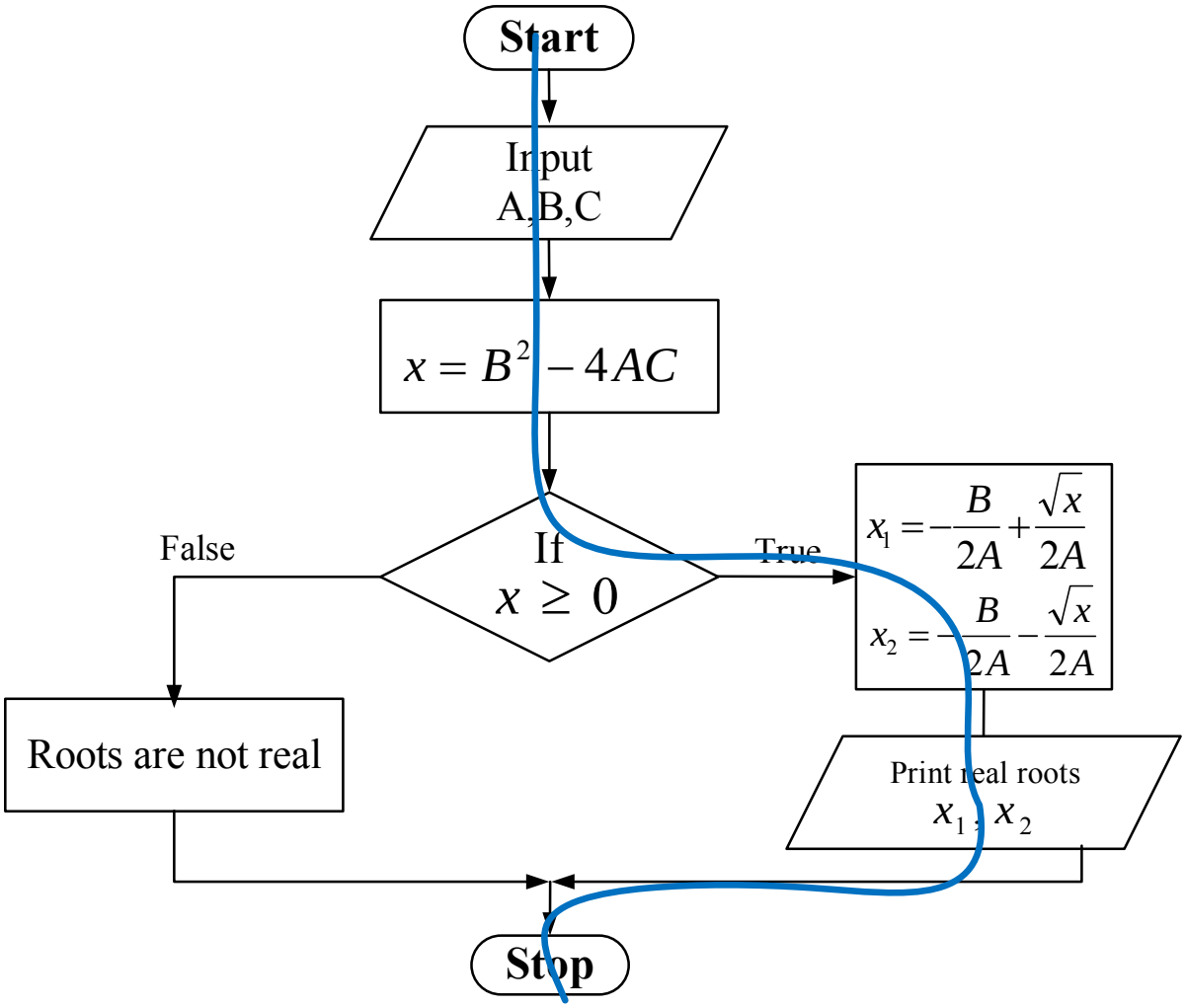
C Programming Constructs

This flowchart shows how to find the roots of a Quadratic equation $Ax^2+Bx+C = 0$

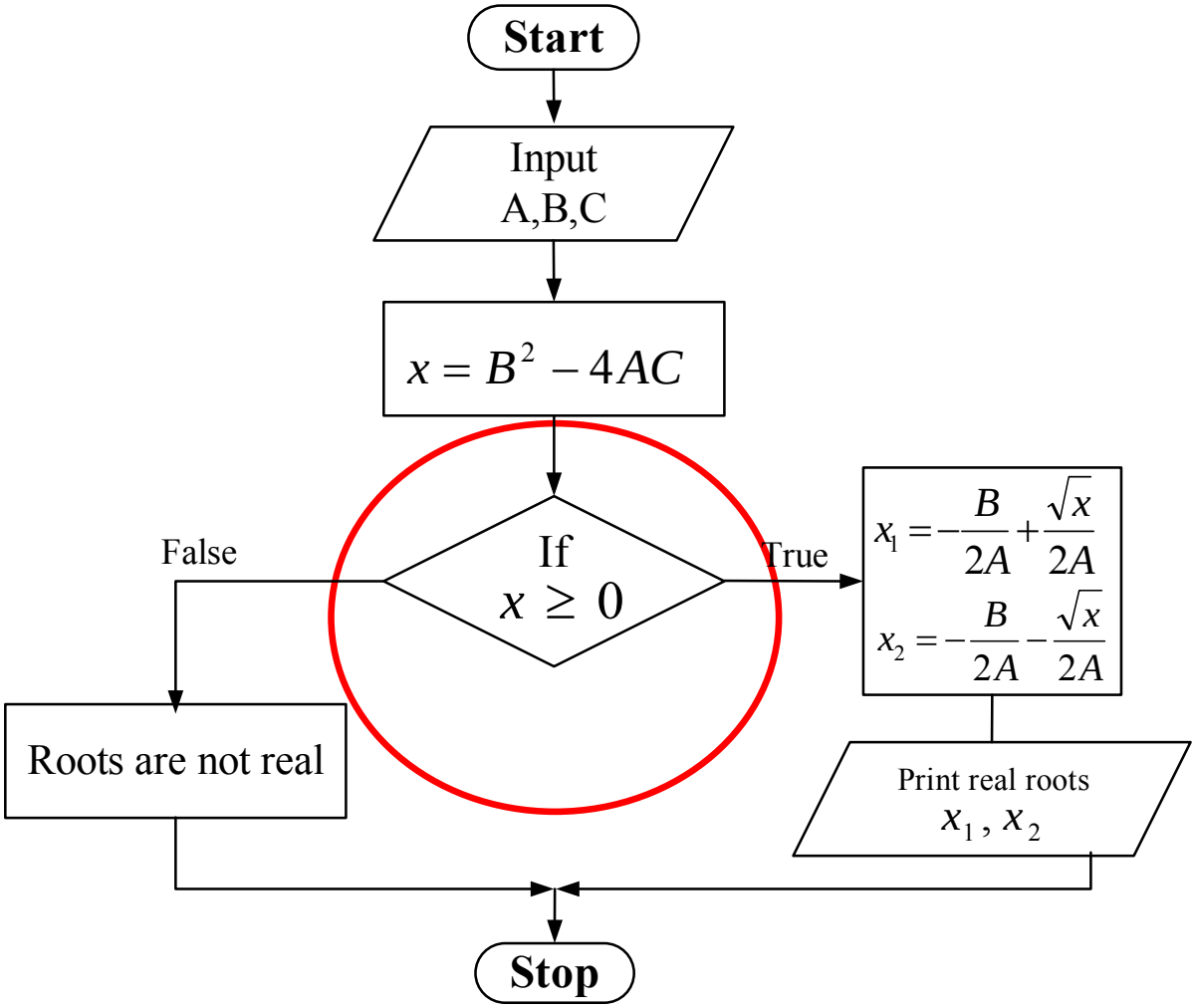


There are three things in any programming:

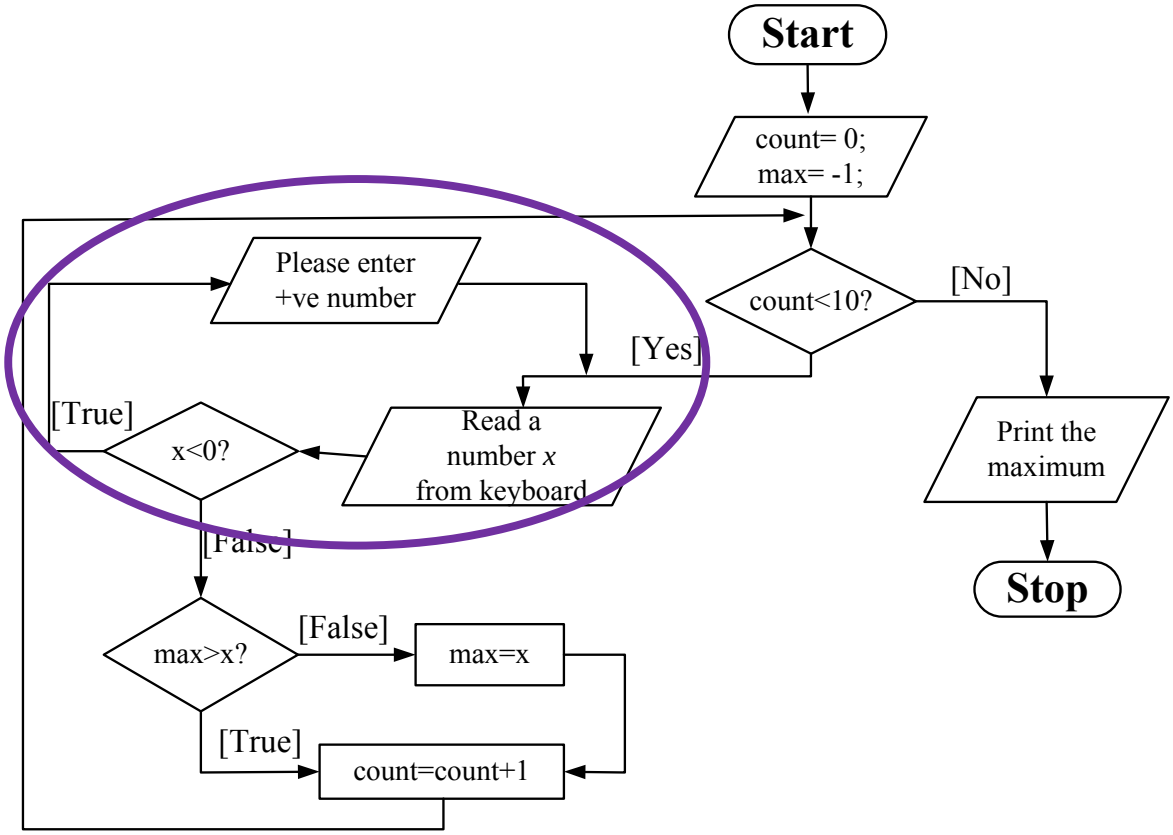
Sequence: A set of statements which would be executed one after another.



Branching: Out of many paths, follow one path

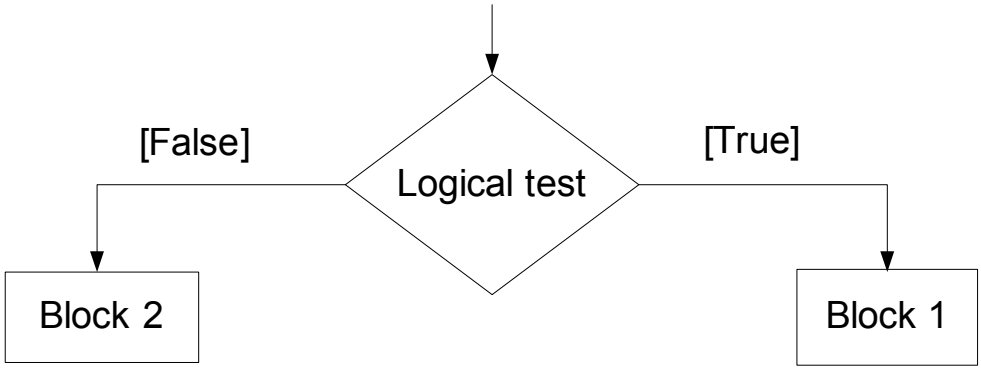


Iteration: Repeat a sequence for a number of times.



Branching

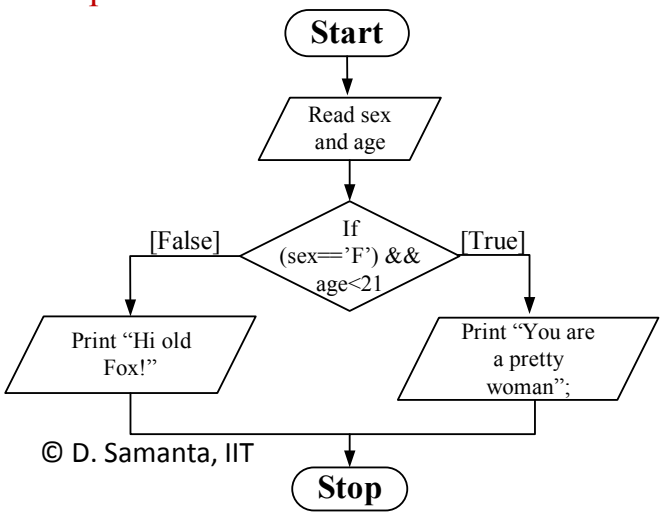
Branching (also called decision) allows different set of instructions to be executed depending on the outcome of logical test.



The `if-else` statement is used to express decision.

```
if (logical test)
        block 1
else
        block 2
```

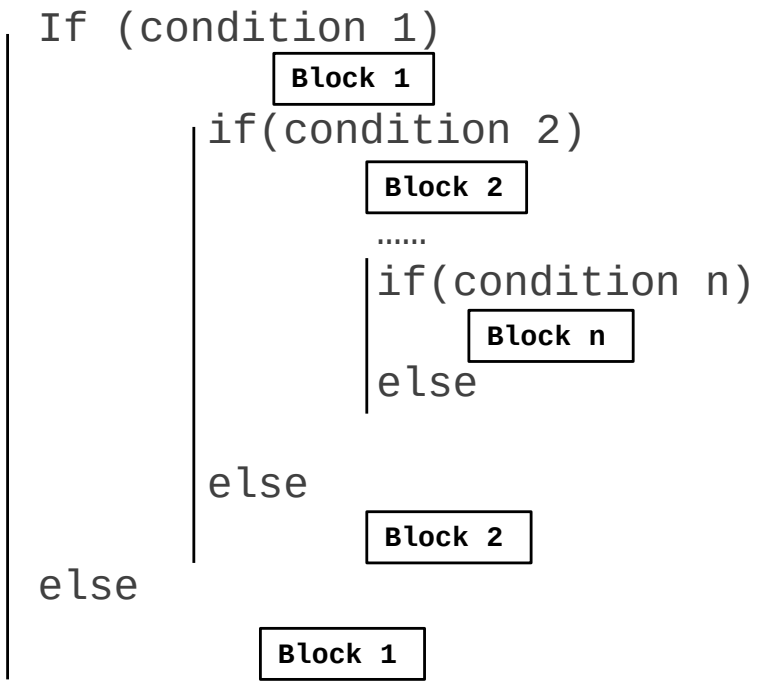
Example




```
# include<stdio.h>
char sex;
int age;
main()
{
  scanf(''%c'',&sex);
  scanf(''%d'',&age);

if((sex=='F')&&(age<21))
     printf(//.....);
 else
     printf(//.....);
}
```

Nesting of if-else statement



Problem may arise if all "*if*" statement may not have "*else*" part! Dangling else problem.

```
if (exp1)
if(exp2)
        statement1;
else
        statement2;
```

```
if (exp1)
    if(exp2){
        statement1;
    else
        statement2;
    }
```

Rule: An else clause is associated with the closest preceding unmatched *if*.

Switch : Multiple Branching

```
switch (expression) {
  case const1: statement1
  case const2: statement2
  ……
  ……
  default: statement
}
```

Switch statement is used to select a particular statement from a group of alternative statement(s).

Note: Here, `expression` should evaluate to an **`int`** value
`const1`, `const2`, …. Are integer values only

Example:

```
int letter;
switch(letter = getchar())
{
  case 'A':
     printf("First letter %c\n", letter);
     break;

  case 'Z':
     printf("Last letter %c\n", letter);
     break;

  default:
     printf("Your letter %c\n", letter);
     break;
}
```

Note: The use of `break` statement

Conditional Operator ?:

Example

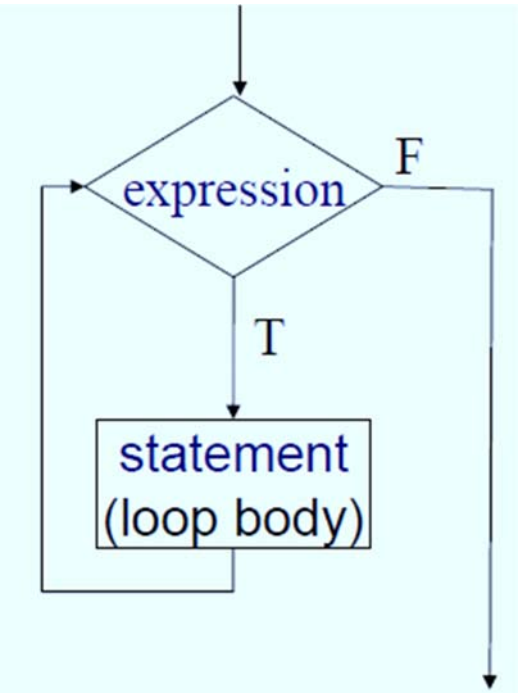
```
interest = (balance>5000)? balance×0.15 : balance×0.10;
```

This is equivalent to:

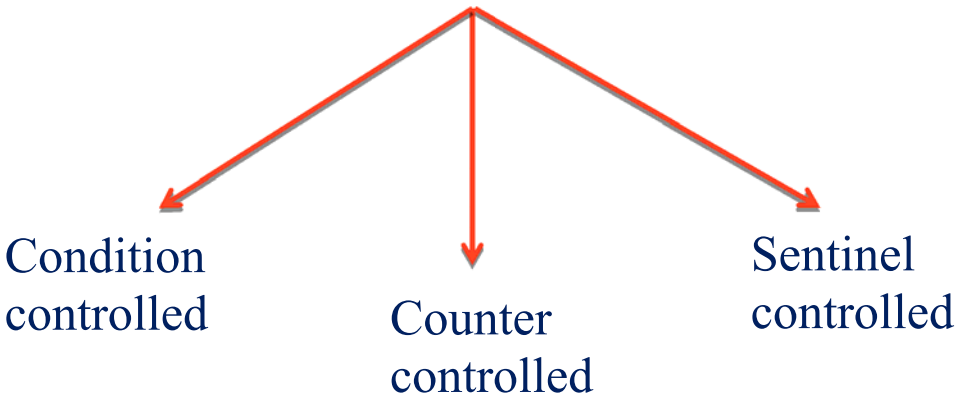
```
if =(balance>5000)
    interest= balance×0.15;
else
    interest= balance×0.10;
```

Looping

In looping (also called iteration) a group of instructions that are executed repeatedly while some condition remains true.

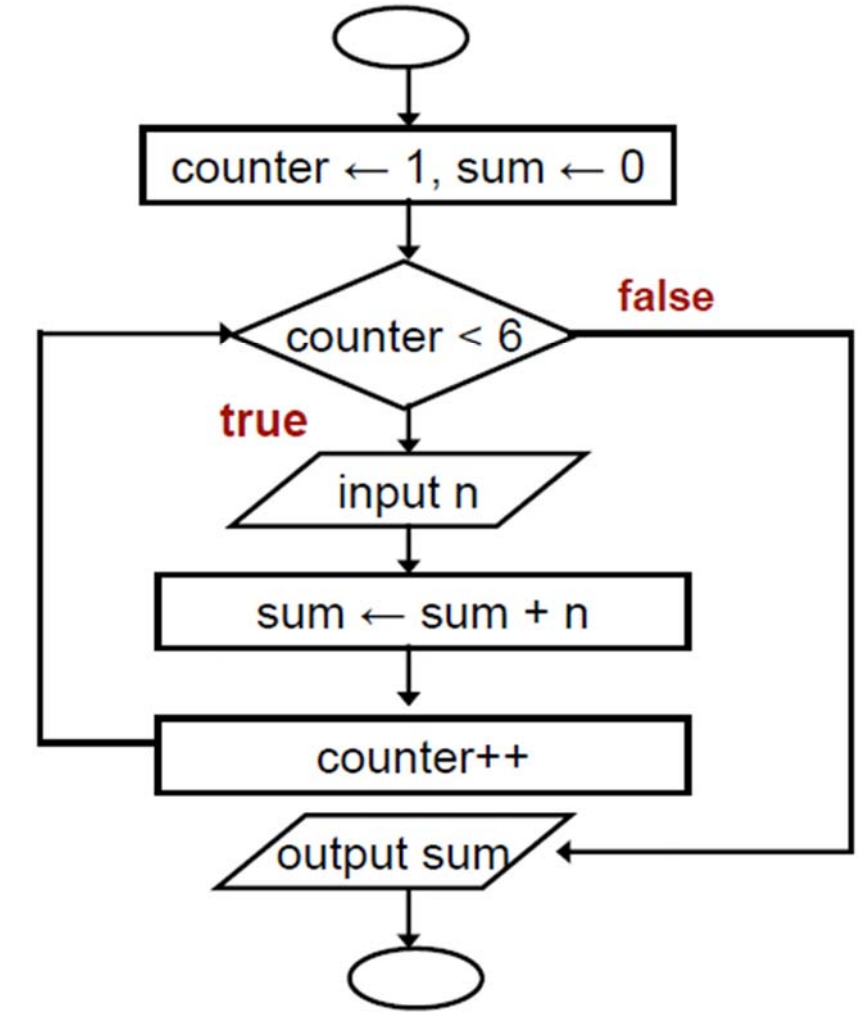


There are three ways to control a looping:



Example 1: **Counter controlled**

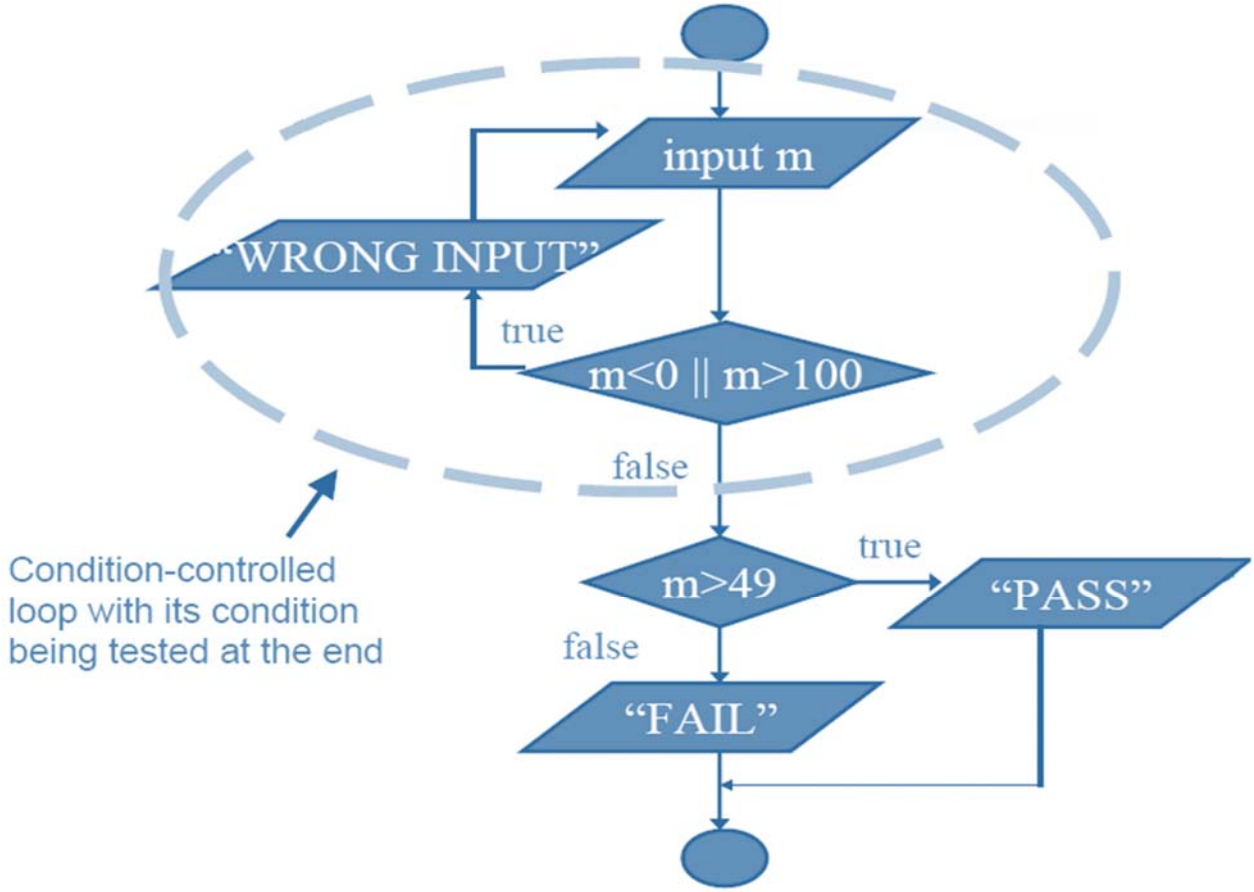
Read 5 integers and display the value of their summation.

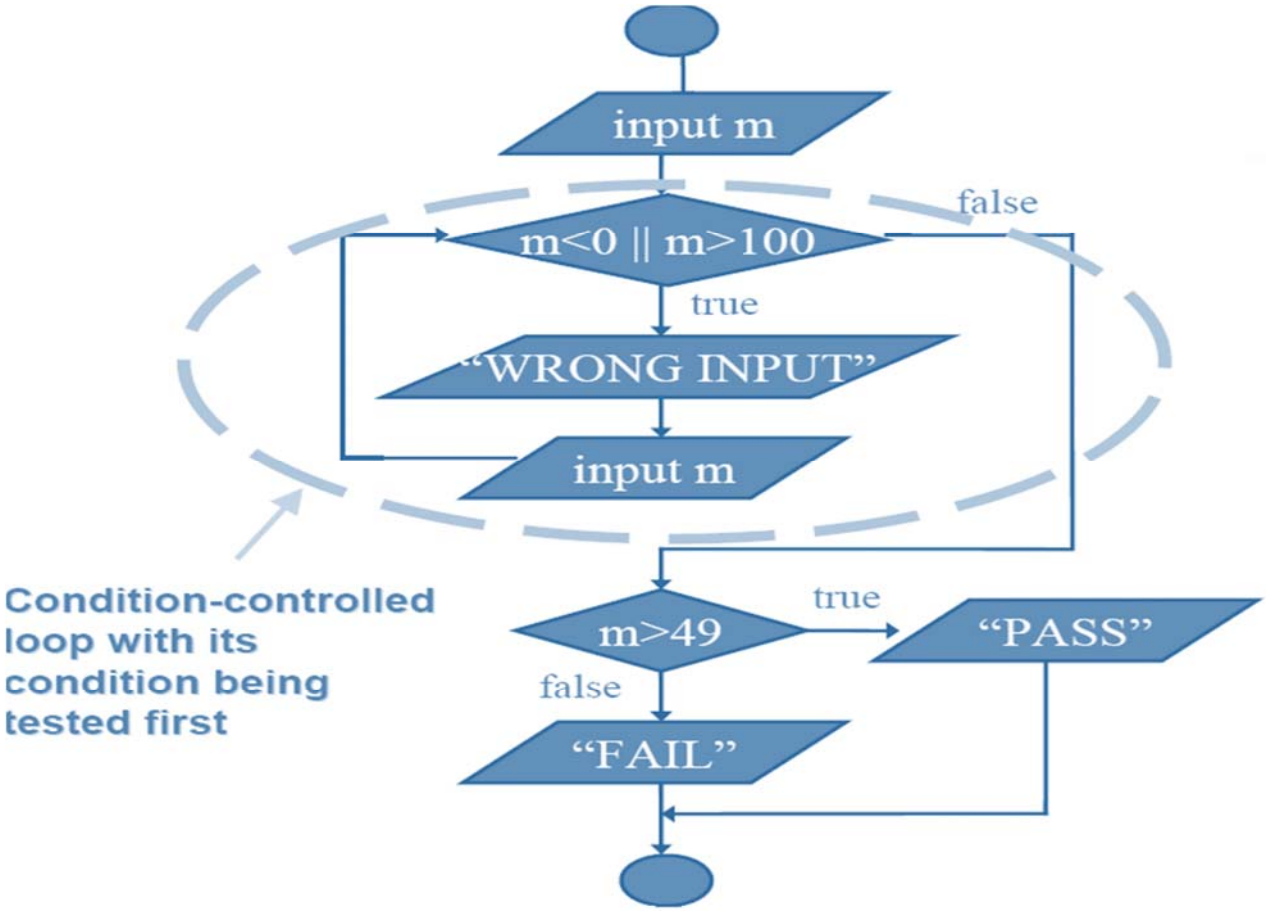


Example 2: **Condition controlled**

Given an exam marks as input, display the appropriate message based on the rules below:

- If marks is greater than 49, display "PASS", otherwise display "FAIL"
- However, for input outside the 0-100 range, display "WRONG INPUT" and prompt the user to input again until a valid input is entered.





Example 3: **Sentinel controlled**

Read a number of positive integers and display the summation and average of these integers.

- A negative or zero input indicates the end of input process.

- Input Example:

  30 16 42 −9

  Sentinel Value

- Output Example:

  $Sum = 88$

  $Average = 29.33$

While Statement

The "while" statement is used to carry out looping operations, in which a group of statements is executed repeatedly, as long as some condition remains satisfied.

```
while (condition)
    statement (s);
```

Example 4: Syntax of while statement

```
while (condition) {
    statement_1;
     ...
     statement_N;
}
```

```
while (i < n) {
    printf ("Line no : d.\n",i);
  i++;
}
```

Note:

- The while-loop will not be entered if the loop-control expression evaluates to false (zero) even before the first iteration.
- `break` can be used to come out of the while loop.

Example 5: `while` statement with `break`

```
float weight;
int flag = 1;

printf ("Enter your weight: ");
scanf ("%f", &weight);

while ( weight > 65.0 ) {
      printf ("Go, exercise, ");
      printf ("then come back. \n");

      printf ("Do you want to continue?");
      scanf ("%d", &flag);

      if (!flag)
         break;
}
```

Nested `while` Statement

How would you print the following diagram?

```
* * * * *
* * * * *
* * * * *
```

repeat 3 times
print a row of 5 stars

repeat 5 times
print *

```
#define ROWS 3
#define COLS 5
…
row =1;
while (row <= ROWS) {
/* print a row of 5 *'s */
   row++;
}
```

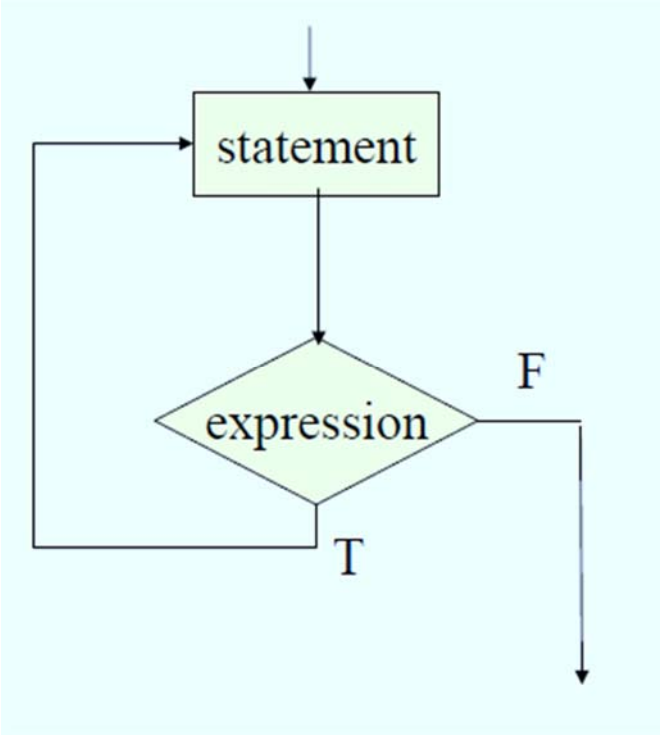
```
row=1;
while (row <= ROWS) {
    col=1;
    while (col <= COLS) {
        printf ("*");
        col++;
    }
    printf("\n");
    row++;
}
```

outer loop

inner loop

do-while Statement

do *statement* while (*expression*)




```
main ()
{
   int digit=0;
   do
     printf("%d\n",digit++);
   while (digit <= 9) ;
}
```

`for` Statement

The "for" statement is the most commonly used looping structure in C.

General syntax:

```
for ( expr1; expr2; expr3) statement;
```

expr1 (init) : initialize parameters

expr2 (test): test condition, loop continues if satisfied

expr3 (update): used to alter the value of the parameters after each iteration

statement (body): body of the loop

```
for (expr1;expr2;expr3) {
   statement
}
```

Example

```
int i;
for (i=1; i < 100; i += 2;) {
   printf("%d\t",i);
}
```

The comma operator

We can give several statements separated by commas in place of "expression1", "expression2", and "expression3".

Example

```
for (fact=1, i=1; i<=10; i++)
fact = fact × i;
```

```
for (sum=0, i=1; i<=N; i++)
sum = sum + i × i;
```

Specifying "Infinite Loop"

```
while (1) {
 statements
}
```

```
for (; ;)
{
 statements
}
```

```
do {
 statements
} while (1);
```

The break Statement

Break out of the loop { }

- can be used with
  - while
  - do while
  - for
  - switch

Example

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

   while ( i<=10 ) /* Run loop for 10 times */
   {
     fact = fact * i;
     if ( fact > 100 ) {
       printf("Factorial of %d above 100", i);
       break;
       /* break out of the while loop */
     }
   i++ ;
  }
  return 1;
}
```

- does not work with
  `if`
  `else`

  - Causes immediate exit from a while, do/while, for or switch structure.

  - Program execution continues with the first statement after the structure.

The `continue` Statement

- Skips the remaining statements in the body of a *while*, *for* or *do/while* structure.
  - Proceeds with the next iteration of the loop.

- while and do/while
  - Loop-continuation test is evaluated immediately after the continue statement is executed.

- for structure
  - *expression3* is evaluated, then *expression2* is evaluated.

Example

```
/* a program segment to calculate 10!
```

```
fact = 1; i = 1;

while (1) {
    fact = fact × i;
    i ++;
       if(i < 10)
         continue;    /* not done yet! Go to loop */
       else
         break;   /* Done! Break the loop
}
```

Tutorial Problems

Problem 1

Draw the flowchart of the following C program

```
# include <stdio.h>
main()
{
    int a, b, c;
    scanf("%d %d %d ", &a, &b, &c);

    if a ≥ b && a ≥ c
        printf ("\n The largest number is %d", a );

    if b ≥ a && b ≥ c
        printf ("\n The largest number is %d", b );

    if c ≥ a && c ≥ b
        printf ("\n The largest number is %d", c );
}
```

Problem 2

Are the following two code fragments give same result?

Code 1:

```
if (section==16)
    printf(''you are in section 16'');
```

Code 2:

```
if (section=16)
    printf(''you are in section 16'');
```

Problem 3

What will be the output of the following code segment? You should assume suitable values of the variables and then give your answer.

```
z = 0;
if(n>0)
if(a>b)
z=a;
else z=b;
printf(``Yahoo! %d'', z);
```

Case 1: n = 5, a = 9, b = 6
Case 2: n = -5, a = 9, b = 6
Case 3 n = 5, a = 6, b = 9

Problem 4

Your program asks a user to type y or Y to select Yes and n or N to select No. Which of the following codes is correct to show the option selected by the user?

Code 1

```
c = getchar();
if(c==`y')&&(c==`Y') printf(``Yes \n'');
else printf(``No \n'');
```

Code 2

```
c = getchar();
if(c!=`n')||(c!=`N') printf(``Yes \n'');
else printf(``No \n'');
```

Problem 5

Write simplified equivalent code for the following.

```
x = ((a>10)&&(b<5))?a+b:0;
```

```
marks>60 ? printf("passed \n"): printf("fail \n");
```

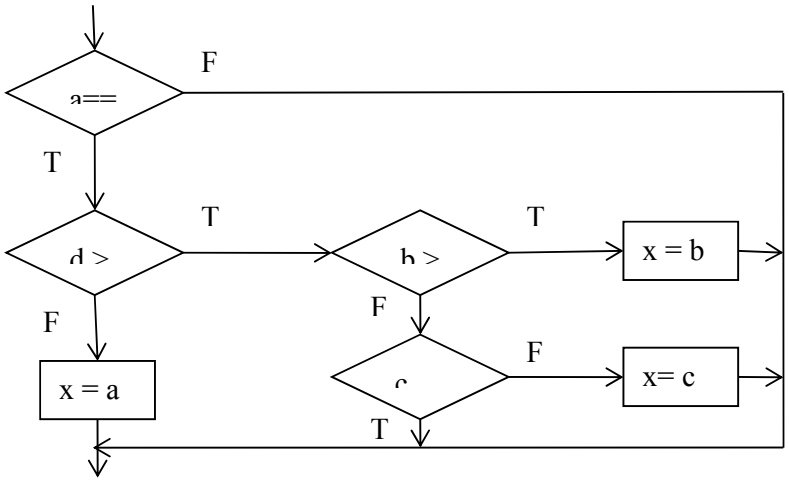
Problem 6

Write c-program to convert a grade given a marks.

$Ex: marks \geq 90$
$A: 80 \leq marks < 90$
$C: 70 \leq marks < 80$
$C: 60 \leq marks < 70$
$D: 50 \leq marks < 60$
$P: 35 \leq marks < 50$
$F: marks < 35$

Problem 7

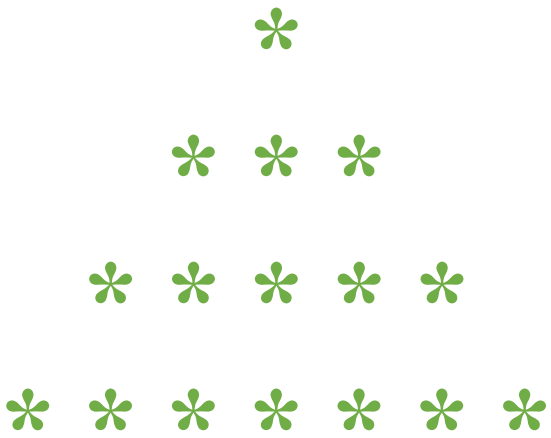
Which of the following nested if statements are logically equivalent to the flow chart below?



(A) `if(a==b)if(d>c) if(b>c)x=b; else if (c==a);else x=c; else x=a;`
(B) `if(a==b)if(d>c) if(b>c)x=b; else if (c==a); x=c; else x=a;`
(C) `if(a==b)if(d>c) if(b>c)x=b; else if (c!=a) x=c; else x=a;`
(D) `if(a==b)if(d>c) if(b>c)x=b; else if (c!=a) x=c; else; else x=a;`
(E) None of the above.

Problem 8

How would you print the following Pascal triangle?



Problem 9

Let `n`, `i` and `sum` be int variables. The user enters a positive value of `n`. Which of the following program segments prints the largest value of sum?

```
sum = 0; i = 1; while (++i < n) sum += i;
printf("%d", sum);

sum = 0; i = 1; while (i++ < n) sum += i;
printf("%d", sum);

for (sum = 0, i = 1; i < n; i++) sum += i;
printf("%d", sum);

for (sum = 0, i = 1; i <= n; ++i) sum += i;
printf("%d", sum);
```

Problem 10

Consider the program segment.

```
int sum = 0;
int i = 0;
while (i < 5)
{
  sum = sum + i;
  i++;
}
printf(``%d\n'',sum);
```

Suppose we replace the while loop in the segment above with a for loop. Which of the following for loops will result in the same value of `sum` printing out?

A.
```
for (int i = 0; i <= 5; i++)
  sum = sum + i;
```

B.
```
for (int i = 1; i <= 5; i++)
  sum = sum + i;
```

C.
```
for (int i = 1; i < 5; i++)
  sum = sum + i;
```

D.
```
for (int i = 2; i < 5; i++)
  sum = sum + i;
```

E.
```
for (int i = 1; i < 6; i++)
  sum = sum + i;
```


Important links:

http://cse.iitkgp.ac.in/~dsamanta/courses/pds/index.html