

# CSE- 105

# Structure Programming

**CONTROL STRUCTURE**

# Algorithm Development

- So far, we considered very simple programs (read, compute, print)
- Top-down Design
  - Start from the big picture
  - Use a process called divide-and-conquer
  - Keep dividing the problem until steps are detailed enough to convert to a program
  - Refinement with Pseudo-code (English like statements) and Flowchart (diagram, graph)
  - For Example : Area calculation problem of a circle... ..

# Pseudo-code Notation and Flowchart Symbols

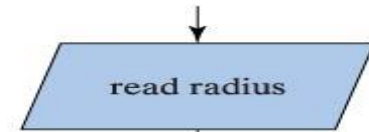
## Basic Operation

## Pseudocode Notation

## Flowchart Symbol

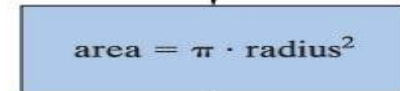
Input

read radius



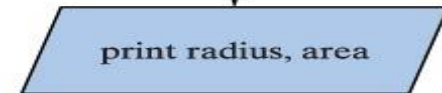
Computation

set area to  $\pi \cdot \text{radius}^2$



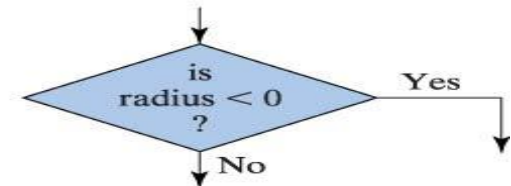
Output

print radius, area



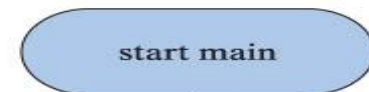
Comparisons

if radius < 0 then ...

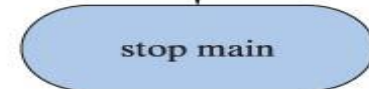


Beginning of algorithm

main:



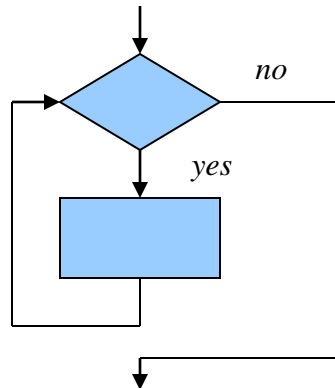
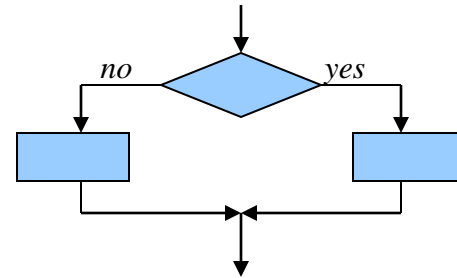
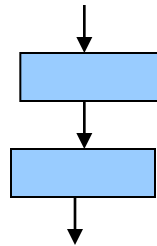
End of algorithm



# Structured Programming

Use simple control structures to organize the solution to a problem

- Sequence
- Selection
- Repetition



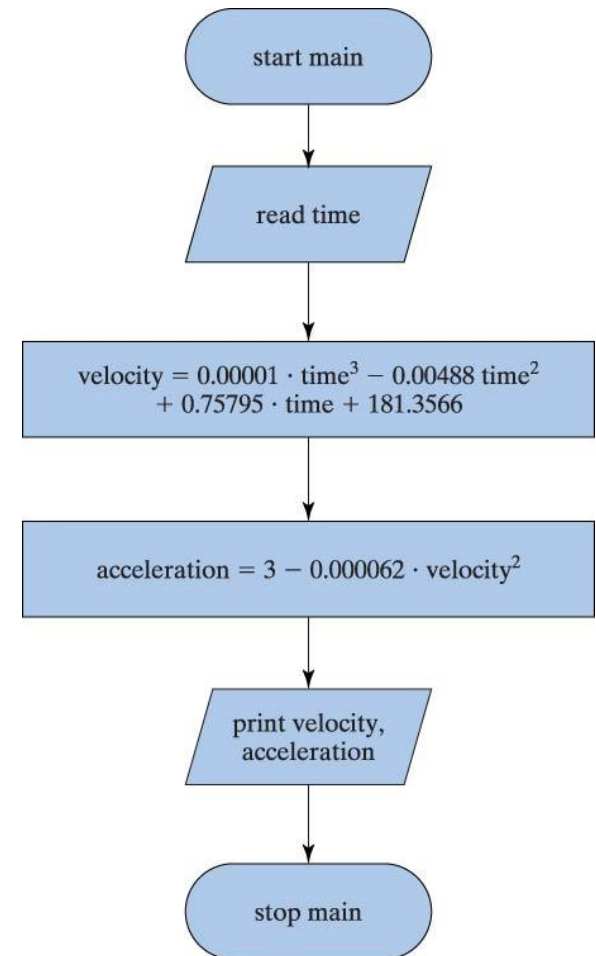
# Sequence

Write a program that takes time  
“T” as input and calculate velocity

$$V = 0.00001 \cdot T^3 - 0.00488 T^2 + 0.75795 \cdot T + 181.3566$$

and acceleration

$$A = 3 - 0.000062 \cdot V^2$$



# Sequence

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

```
main()  
{
```

```
    double time, velocity, acceleration;
```

```
    printf("Please Enter the Time \n");  
    scanf("%lf",&time);
```

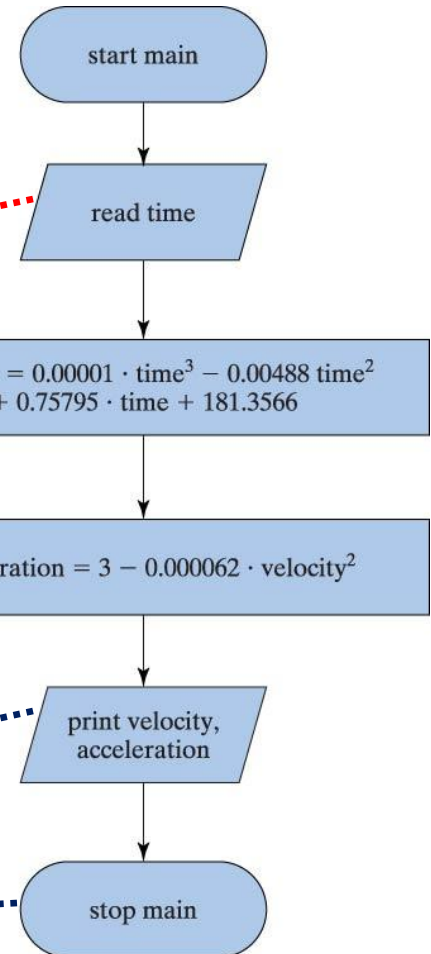
```
    velocity = (0.00001*(time*time*time))  
               - (0.00488*(time*time)) +  
               (0.75795*(time)) + 181.3566;
```

```
    acceleration = 3 - (0.000062*(velocity*velocity));
```

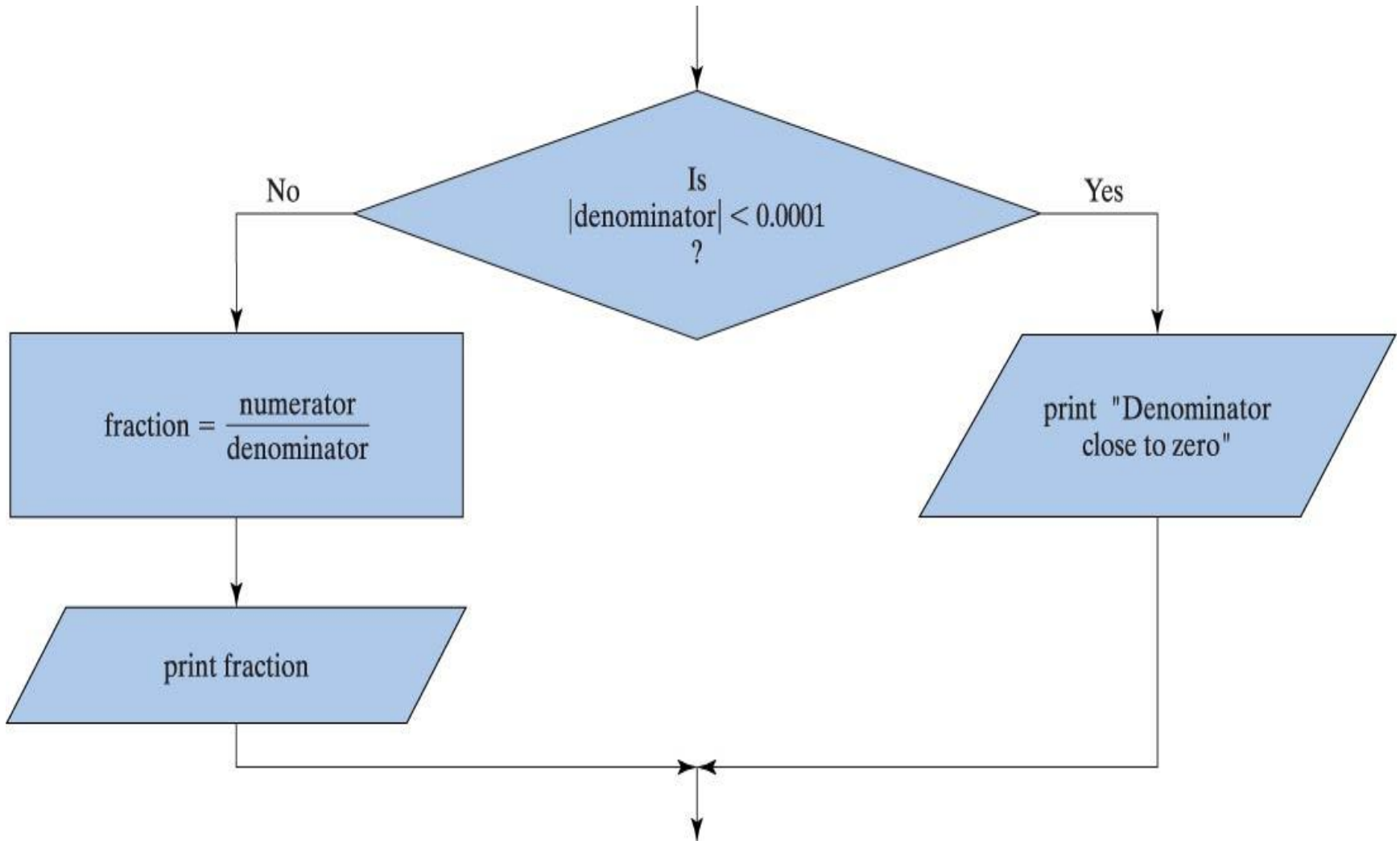
```
    printf("Velocity is %f \n",velocity);  
    printf("Acceleration is %f \n",acceleration);
```

```
    return 0;
```

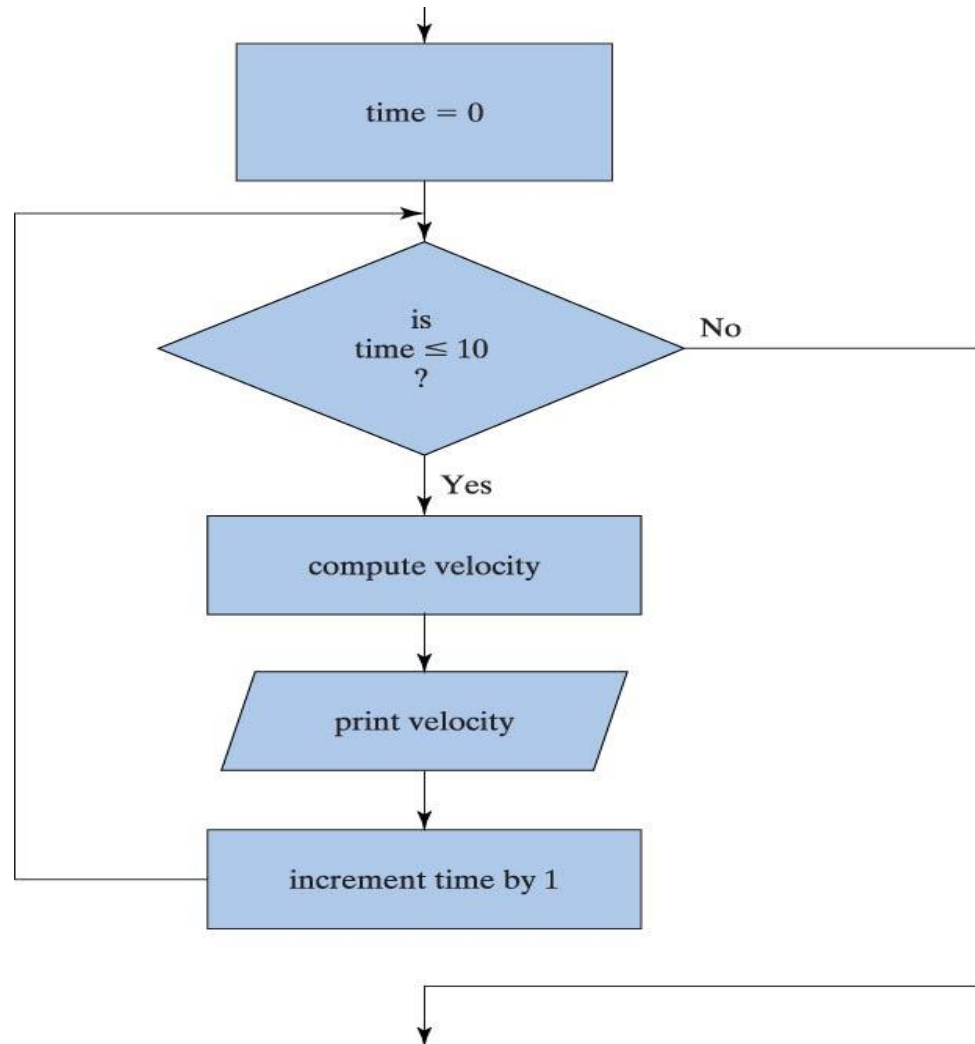
```
}
```



# Selection



# Repetition





# Extras

- Evaluation of alternative solution
  - A problem can be solved in many different ways
  - Which is the best (e.g, faster, less memory req)
- Error condition
  - Do not trust user! Check the data.  $A=b/c$ ;
  - Be clear about specifications
- Generation of Test Data
  - Test each of the error conditions
  - Program validation and verification
  - Program walkthrough

# Conditional Expressions

- Selection and repetition structures use conditions, so we will first discuss them
- A **condition** is an expression (e.g.,  $a > b$ ) that can be evaluated to be
  - TRUE (any value  $> 0$ ) or
  - FALSE (value of 0)
- Conditional Expression is composed of expressions combined with **relational** and/or **logical operators**

# Relational Operators

- `==` equality `(x == 3)`
- `!=` non equality `(y != 0)`
- `<` less than `(x < y)`
- `>` greater than `(y > 10)`
- `<=` less than equal to `(x <= 0)`
- `>=` greater than equal to `(x >= y)`

!!! **a==b** vs. **a=b** !!!

# Examples

- $A < B$
- $D = b > c;$
- If (D)  
 $A = b + c;$
- Mixing with arithmetic op
  - $X + Y \geq K/3$

4	A
2	B
4	C
3	b
4	c
2	X
1	Y
10	K

# Logical Operators

- **!**            not             $!(x==0)$
- **&&**           and             $(x>=0) \&\& (x<=10)$
- **||**           or             $(x>0) || (x<0)$

A	B	A && B	A    B	!A	!B
False	False	False	False	True	True
False	True	False	True	True	False
True	False	False	True	False	True
True	True	True	True	False	False

# Examples

- $A < B \ \&\& \ C \geq 5$
- $A + B * 2 < 5 \ \&\& \ 4 \geq A/2 \ || \ T - 2 < 10$
- $A < B < C \ \text{????}$
- $A < B < C$  is not the same as
  - $(A < B) \ \&\& \ (B < C)$

# Precedence for Arithmetic, Relational, and Logical Operators

Precedence	Operation	Associativity
1	( )	Innermost first
2	++ -- + - ! (type)	Right to left (unary)
3	* / %	Left to right
4	+ -	Left to right
5	< <= > >=	Left to right
6	== !=	Left to right
7	&&	Left to right
8		Left to right
9	= += -= *= /= %=	Right to left

# Exercise

- Assume that following variables are declared

$a = 5.5$        $b = 1.5$      $k = -3$

- Are the following true or false

$a < 10.0 + k$

$a + b \geq 6.5$

$k \neq a - b$

$!(a == 3 * b)$

$a < 10 \ \&\& \ a > 5$



# Selection Statements

- if
- if else
- switch

# if statement

- if(Boolean expression)  
statement; /\* single statement \*/
- if(Boolean expression) {  
/\* more than one statement \*/  
statement1;  
...  
statement n;  
}

# if statement - examples

- `if (x > 0)`  
    `k++;`
- `if(x > 0) {`  
    `x = sqrt(x);`  
    `k++;`  
    `}`
- `if(x > 0)`                      `/* a common mistake */`  
    `x = sqrt(x);`  
    `k++;`

# if else statement

- if(Boolean expression)  
    statement;  
else  
    statement;
- if(Boolean expression)  
    {  
        statement block  
    }  
else  
    {  
        statement block  
    }

# if else statement

- What does the following program do?
- Assume that x, y, temp are declared.

```
int x=10, y=20, temp;
```

```
if (x > y)
```

```
    temp = x;
```

```
else
```

```
    temp = y;
```

Split the statement into two separate if statements

```
if (x>y)
```

```
    temp = x;
```

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

```
    temp = y;
```

# Exercise

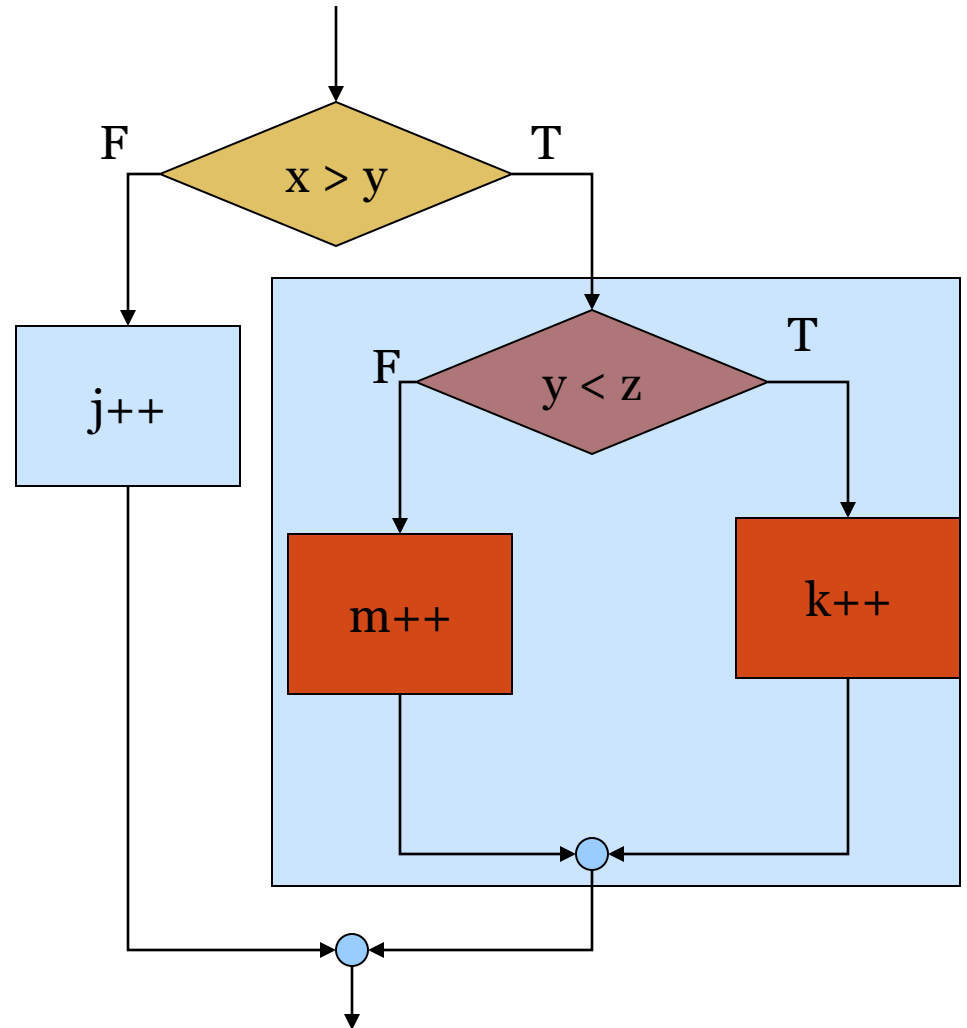
- Write an if-else statement to find both the maximum and minimum of two numbers.
- Assume that x, y, min, max are declared.

```
if (x>y) {  
    max = x;  
    min = y;}  
else {  
    max = y;  
    min = x;}
```

Ex:      x = 10, y = 5  
          x = 3,  y = 4  
          x = 6,  y = 6

# nested if-else

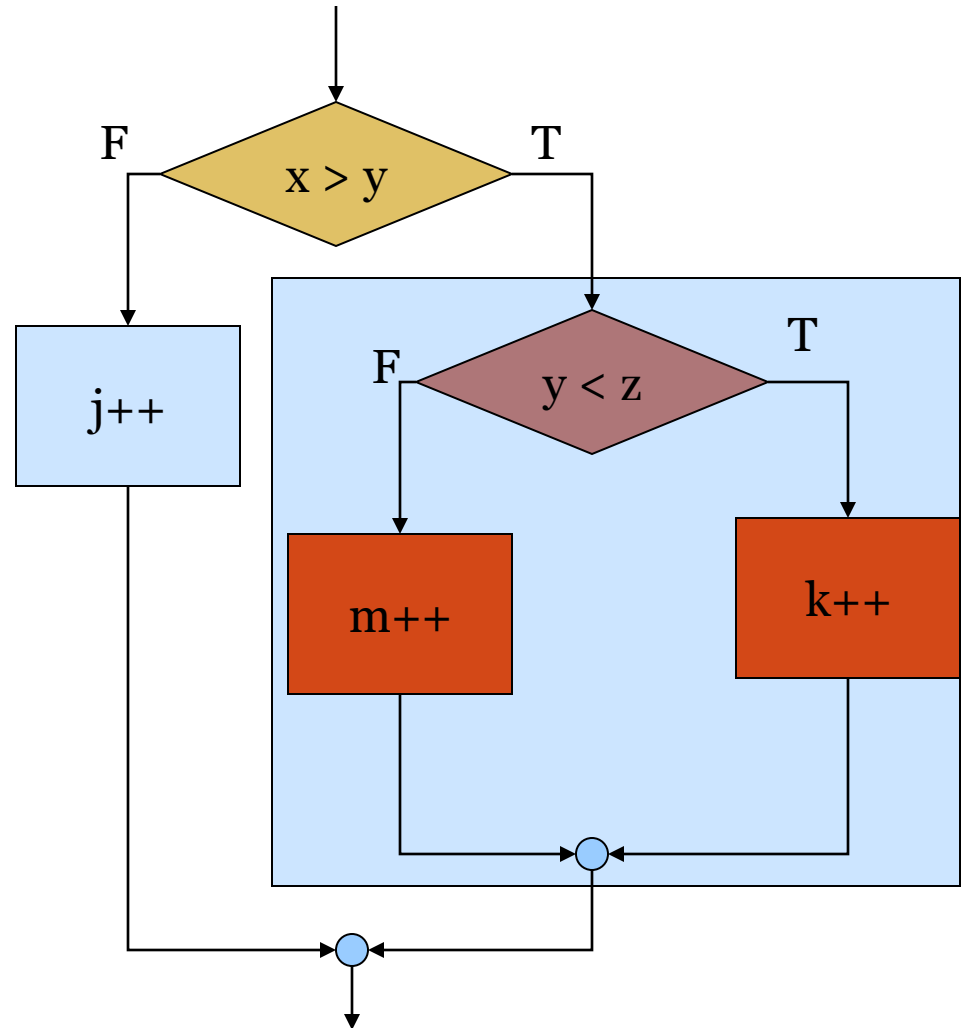
```
if(x > y) {  
    if(y < z) {  
        k++;  
    } else {  
        m++;  
    }  
} else {  
    j++;  
}
```



# Exercise

What are the values of  $j$ ,  $k$  and  $m$ , Where,  
 $\text{int } x=9, y=7, z=2, k=0, m=0, j=0;$

```
if(x > y) {  
    if(y < z) {  
        k++;  
    } else {  
        m++;  
    }  
} else {  
    j++;  
}
```





# Exercise

- Given a score and the following grading scale write a program to find the corresponding grade.

90-100	A
80-89	B
70-79	C
60-69	D
0-59	F

# Solution-1

```
if ((score >= 90) && (score <= 100))  
    grade = 'A';  
else if ((score >= 80) && (score <= 89))  
    grade = 'B';  
else if ((score >= 70) && (score <= 79))  
    grade = 'C';  
else if ((score >= 60) && (score <= 69))  
    grade = 'D';  
else if ((score >= 0) && (score <= 59))  
    grade = 'F';  
else  
    printf("Invalid Score\n");
```

# Solution-2

```
if ((score >= 0) && (score <= 100))  
    if (score >= 90)  
        grade = 'A';  
    else if (score >= 80)  
        grade = 'B';  
    else if (score >= 70)  
        grade = 'C';  
    else if (score >= 60)  
        grade = 'D';  
    else  
        grade = 'F';  
else  
    printf("Invalid Score\n");
```

# Exercise: Find the value of **a**

```
int a = 750;  
if (a > 0)
```

```
    if (a >= 1000)
```

```
        a = 0;
```

```
    else
```

```
        if (a < 500)
```

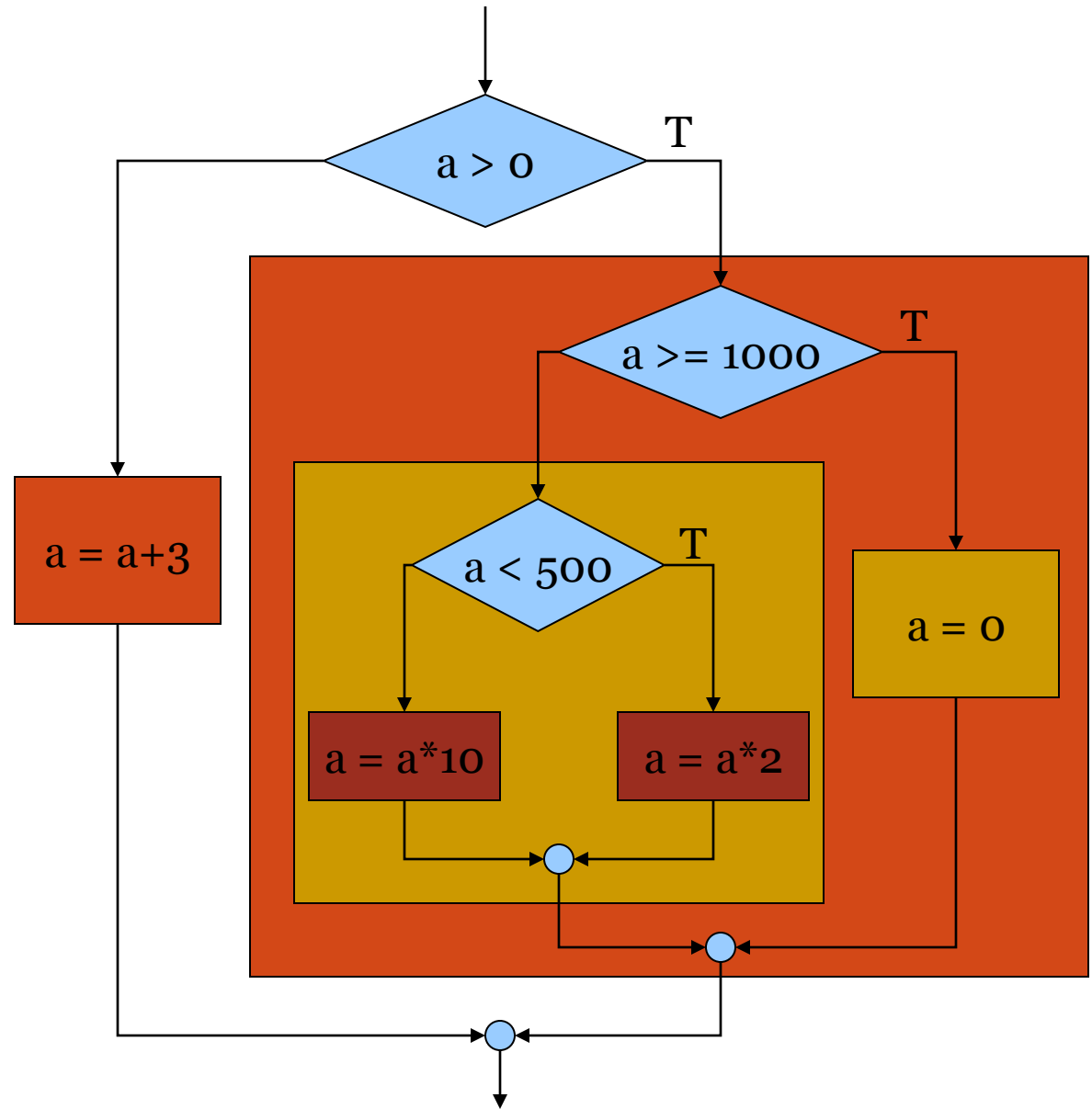
```
            a = a * 2;
```

```
        else
```

```
            a = a * 10;
```

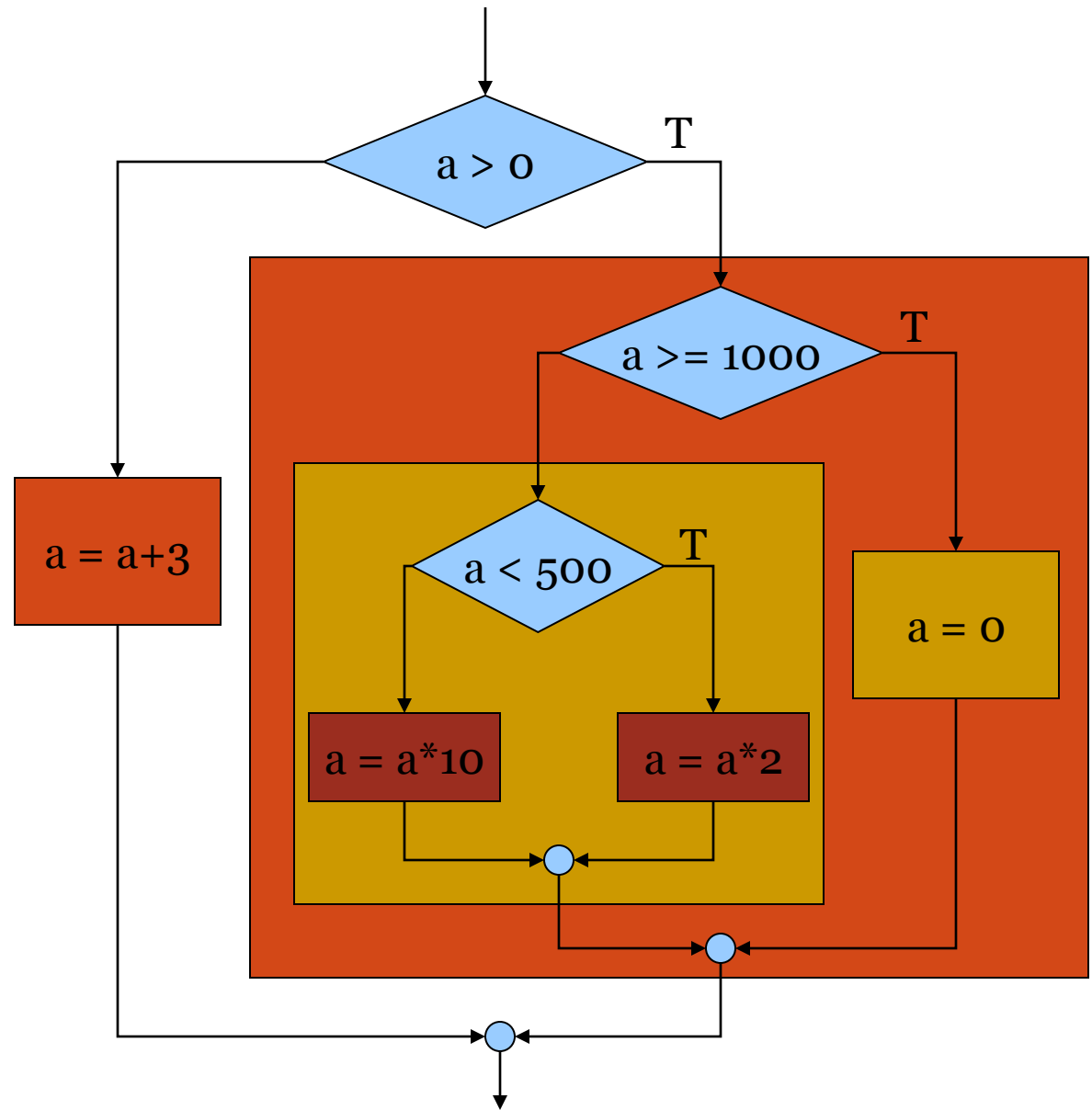
```
    else
```

```
        a = a + 3;
```



# Exercise: Find the value of **a**

```
int a = 750;  
if (a > 0) {  
    if (a >= 1000) {  
        a = 0;  
    } else {  
        if (a < 500) {  
            a = a * 2;  
        } else {  
            a = a * 10;  
        }  
    }  
} else {  
    a = a + 3;  
}
```



# Exercise: which task takes more time

- Suppose we have two tasks A and B
  - A takes  $A_h$  hours,  $A_m$  minutes, and  $A_s$  seconds
  - B takes  $B_h$  hours,  $B_m$  minutes, and  $B_s$  seconds
- Write if-else statements to print out which task takes more time?

# Indentation

```
int a = 750;  
if (a>0)  
    if (a >= 1000)  
        a = 0;  
    else  
        if (a <500)  
            a *= 2;  
        else  
            a *= 10;  
    else  
        a += 3;
```



Good

```
int a = 750;  
if (a>0)  
if (a >= 1000)  
    a = 0;  
else  
if (a <500)  
    a *= 2;  
else  
    a *= 10;  
else  
    a += 3;
```



Not good

# Exercise

- What is the output of the following program

```
int a = 5, b = 3;
```

```
if (a>10)
    a = 50;
    b = 20;

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



```
if (a>10)
    a = 50;
b = 20;

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

```
if (a>10) {
    a = 50;
    b = 20;
}
printf(" a = %d, b = %d\n",a, b);
```



```
if (a>10) {
    a = 50;
    b = 20;
}
printf(" a = %d, b = %d\n",a, b);
```



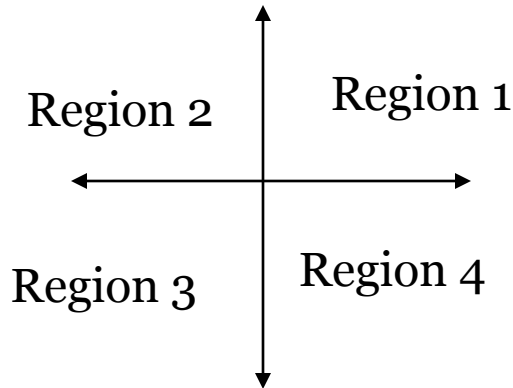
# More selection examples

# Max, Min, Median

- Write a program that reads 3 integer numbers a, b and c from user and computes minimum, median and maximum of the numbers.
- Example:
  - $a = 2, b = 5, c = 3$ 
    - ✦ minimum = 2, maximum = 5, median = 3
  - $a = 2, b = 2, c = 3$ 
    - ✦ minimum = 2, maximum = 3, median = 2

# Region in a plane

- Write a program that reads a point  $(x, y)$  from user and prints its region



For example

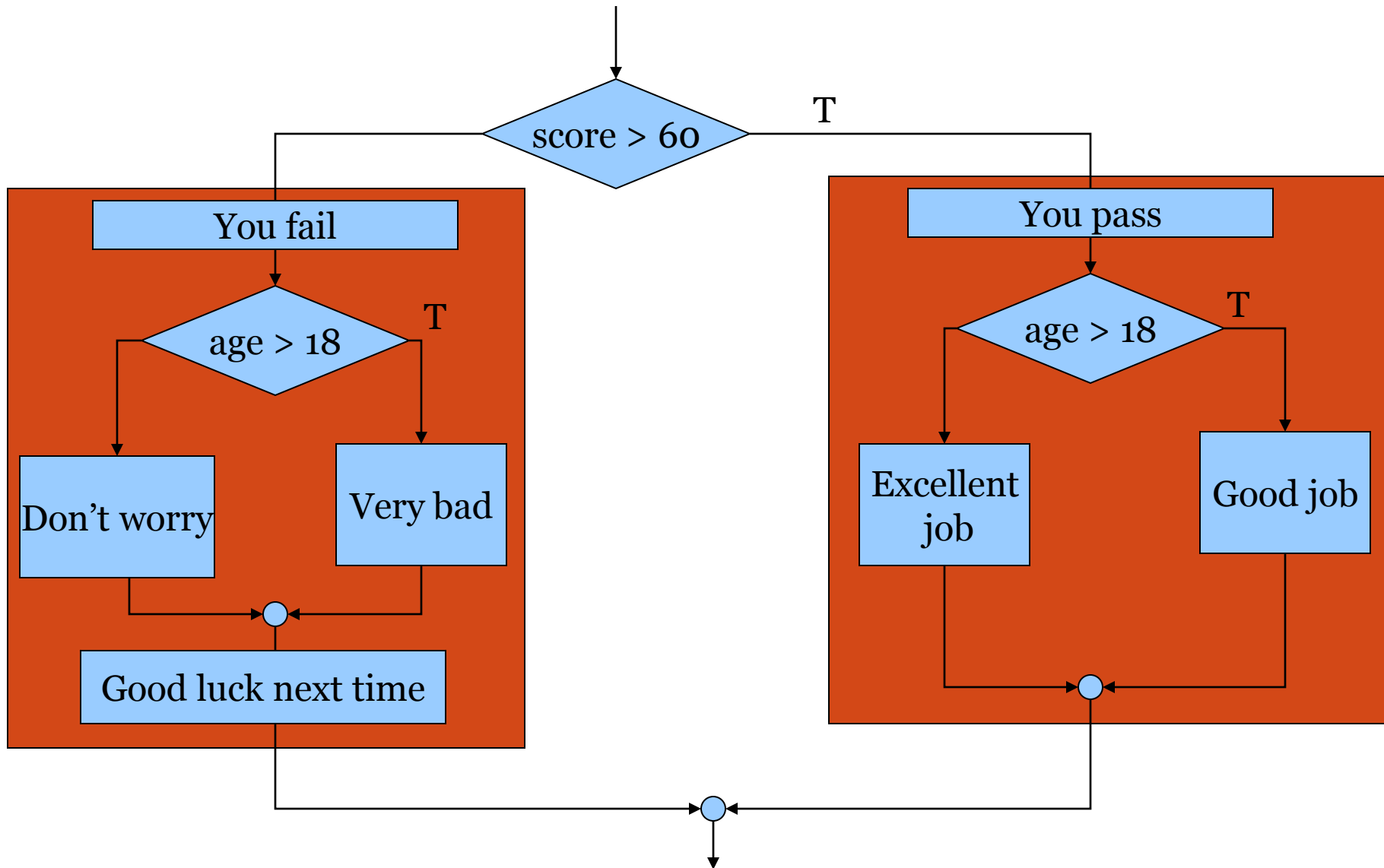
Enter x, y: 3 -1

This point is in Region 4

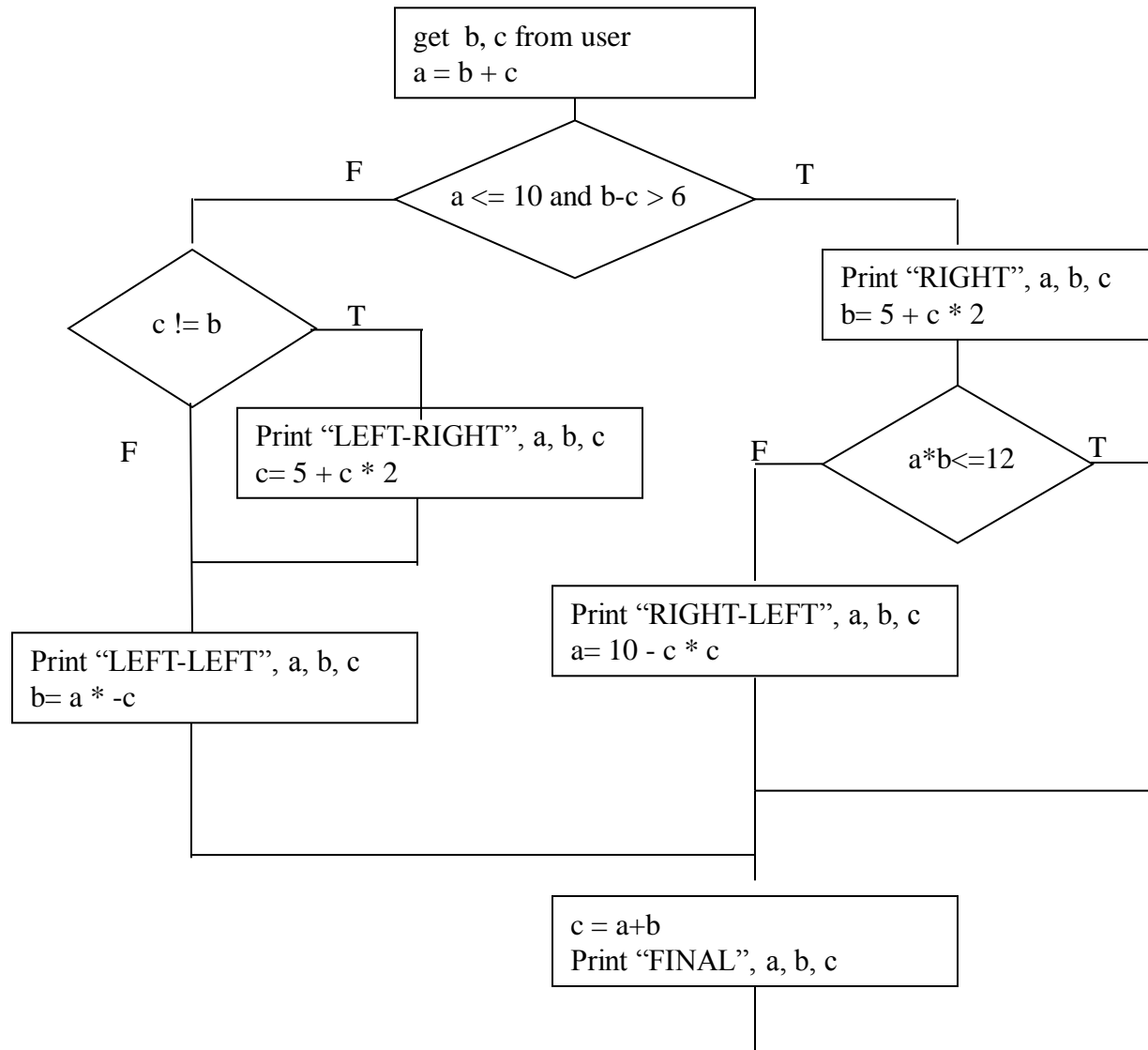
Enter x, y: -1 -5

This point is in region 3

# Write if-else statement



```
if (score > 60) {  
    printf("You Pass\n");  
    if (age > 18) {  
        printf("Good job \n");  
    } else {  
        printf("Excellent job\n");  
    }  
} else {  
    printf("You Fail\n");  
    if (age > 18) {  
        printf(" Very bad \n");  
    } else {  
        printf(" Don't worry \n");  
    }  
    printf(" Good luck next time \n");  
}
```

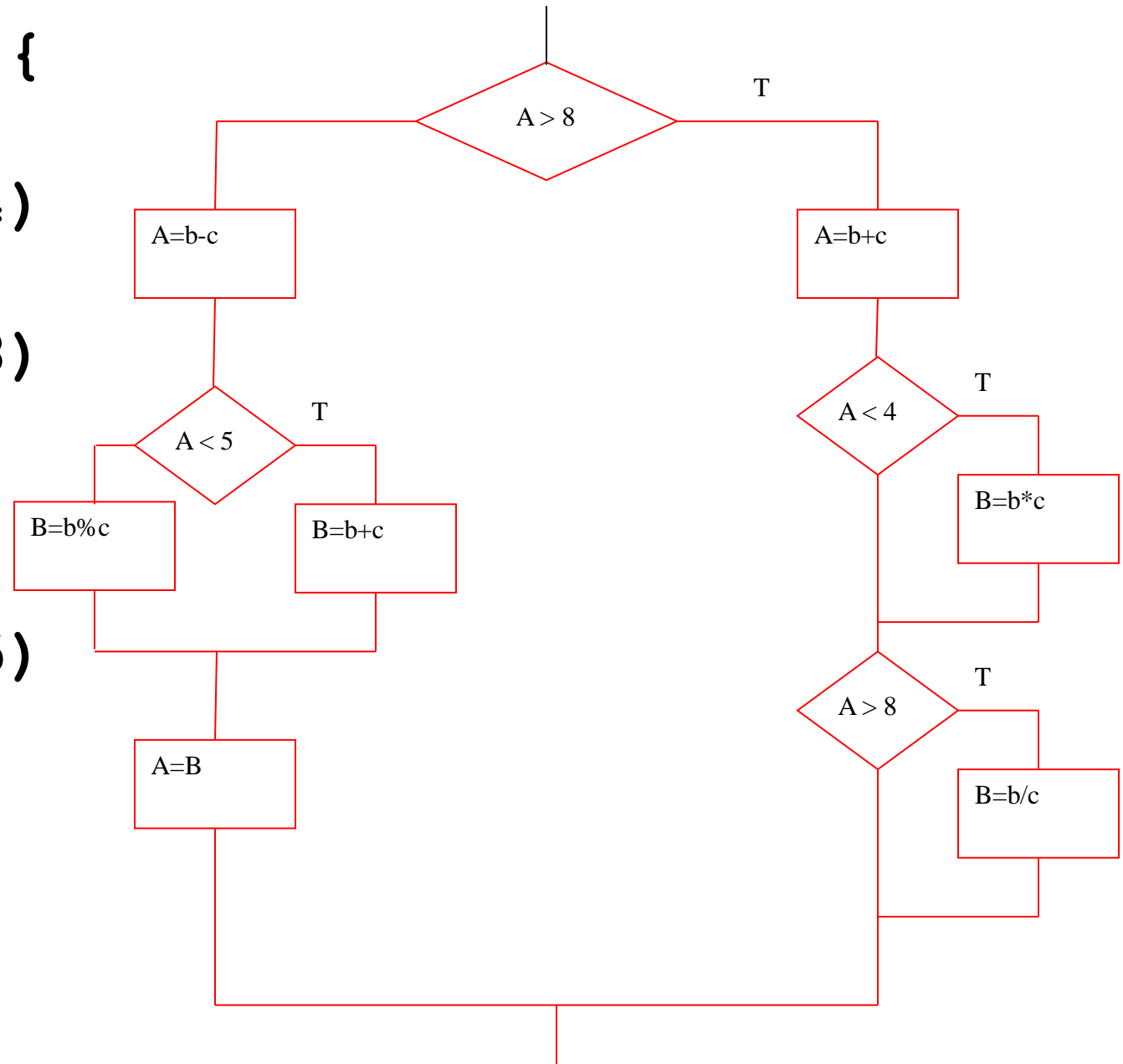


Print "RIGHT", a, b, c means  
`printf("RIGHT a=%lf b=%lf c=%lf \n",a, b, c);`

```
a=b+c;
if (a<=10 && b-c>6) {
    printf("RIGHT a=%lf b=%lf c=%lf \n", a, b, c);
    b=5+c*2;
    if (a*b<=12) {
    } else {
        printf("RIGHT-LEFT a=%lf b=%lf c=%lf \n",a, b, c);
        a=10-c*c;
    }
} else {
    if (c != b) {
        printf("LEFT-RIGHT a=%lf b=%lf c=%lf \n",a, b, c);
        c=5+c*2;
    }
    printf("LEFT-LEFT a=%lf b=%lf c=%lf \n",a, b, c);
    b=a*-c;
}
c=a+b;
printf("Final a=%lf b=%lf c=%lf \n",a, b, c);
```

# Another if-else $\rightarrow$ flowchart

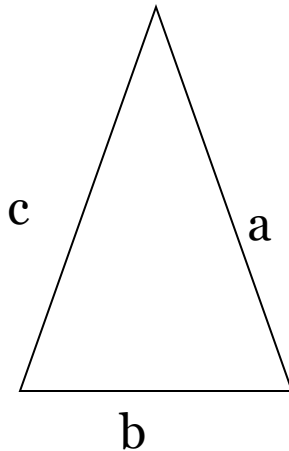
```
if( A > 8) {  
    A=b+c;  
    if (A < 4)  
        B=b*c;  
    if (A > 8)  
        B=b/c;  
} else {  
    A=b-c;  
    if (A < 5)  
        B=b+c;  
    else  
        B=b%c;  
    A=B;  
}
```





# Triangle inequality

- Suppose we want to check if we can make a triangle using  $a$ ,  $b$ ,  $c$



$$|a-b| \leq c \quad |a-c| \leq b \quad |b-c| \leq a$$

$$a+b \geq c \quad a+c \geq b \quad b+c \geq a$$

# Charge for money transfer

- Suppose you transfer \$N and bank's charge occurs as follows.

$$\text{cost} = \begin{cases} \$10 & \text{if } N \leq \$500 \\ \$10 + 2\% \text{ of } N & \text{if } 500 < N \leq 1000 \\ \$15 + 0.1\% \text{ of } N & \text{if } 1000 < N < 10000 \\ \$30 & \text{Otherwise} \end{cases}$$

- Write a program that reads N and computes cost

# Spell out a number in text using if-else and switch

- Write a program that reads a number between 1 and 999 from user and spells out it in English.

For example:

- 453           → Four hundred fifty three
- 37            → Thirty seven
- 204           → Two hundred four

# Loop (Repetition) Structures

# Problem: Conversion table degrees $\rightarrow$ radians



radians = degrees \* PI / 180;

## Degrees to Radians

0	0.000000
10	0.174533
20	0.349066
30	0.523599
...	
340	5.934120
350	6.108653
360	6.283186

# Sequential Solution

```
degrees = ???  
radians = degrees*PI/180;  
printf("%d %f \n", degrees, radians);
```



Not a good solution

```
#include <stdio.h>  
#define PI 3.141593  
  
int main(void)  
{  
    int degrees=0;  
    double radians;  
  
    printf("Degrees to Radians \n");  
  
    degrees = 0;  
    radians = degrees*PI/180;  
    printf("%d %f \n", degrees, radians);  
  
    degrees = 10;  
    radians = degrees*PI/180;  
    printf("%d %f \n", degrees, radians);  
  
    degrees = 20;  
    radians = degrees*PI/180;  
    printf("%d %f \n", degrees, radians);  
  
    ...  
    degrees = 360;  
    radians = degrees*PI/180;  
    printf("%d %f \n", degrees, radians);  
}
```

# Loop Solution

```
degrees = ???  
radians = degrees*PI/180;  
printf("%d %f \n", degrees, radians);
```

```
#include <stdio.h>  
#define PI 3.141593  
  
int main(void)  
{  
    int degrees=0;  
    double radians;  
  
    printf("Degrees to Radians \n");  
  
    while (degrees <= 360) {  
  
        radians = degrees*PI/180;  
        printf("%d %f \n", degrees, radians);  
        degrees += 10;  
    }  
}
```

# Loop (Repetition) Structures

- while statement
- do while statement
- for statement
- Two new statements used with loops
  - break and continue



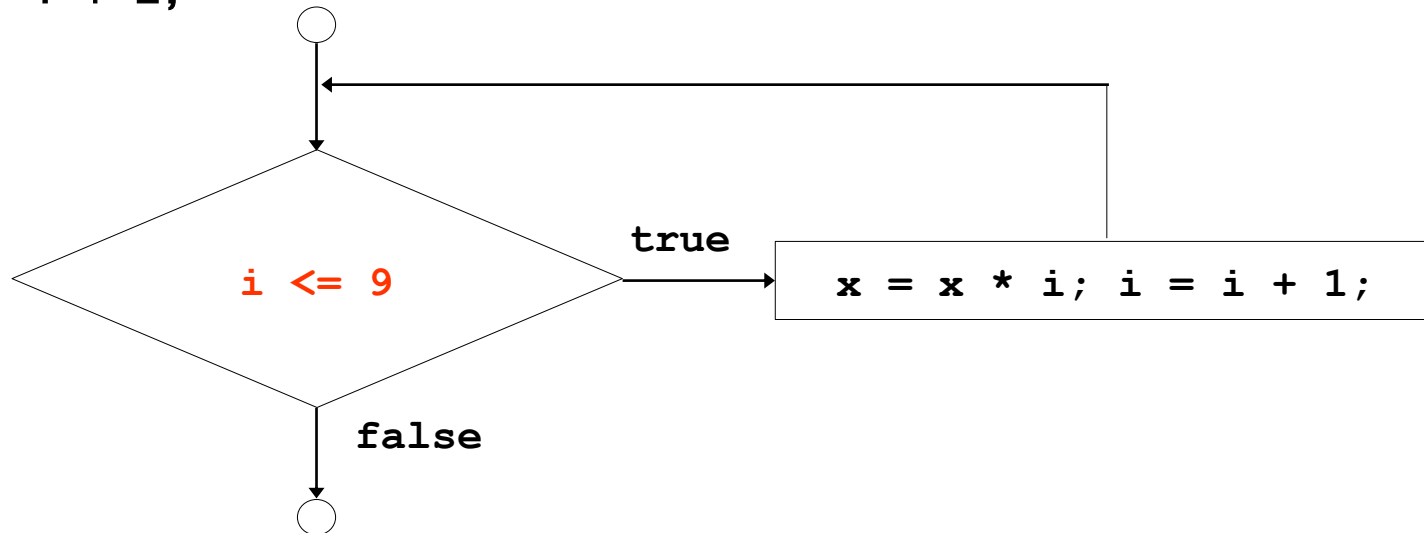
# while statement

- `while(expression)`  
    `statement;`
- `while(expression) {`  
    `statement;`  
    `statement;`  
    `.`  
    `}`

# The while Control Structure

Example:

```
x = 1; i = 2;  
while (i <= 9) {  
    x = x * i;  
    i = i + 1;  
}
```



# Example

```
#include <stdio.h>
#define PI 3.141593

int main(void)
{
    int degrees=0;
    double radians;

    printf("Degrees to Radians \n");
    while (degrees <= 360)
    {
        radians = degrees*PI/180;
        printf("%6i %9.6f \n", degrees, radians);
        degrees += 10;
    }
    return 0;
}
```

# do while

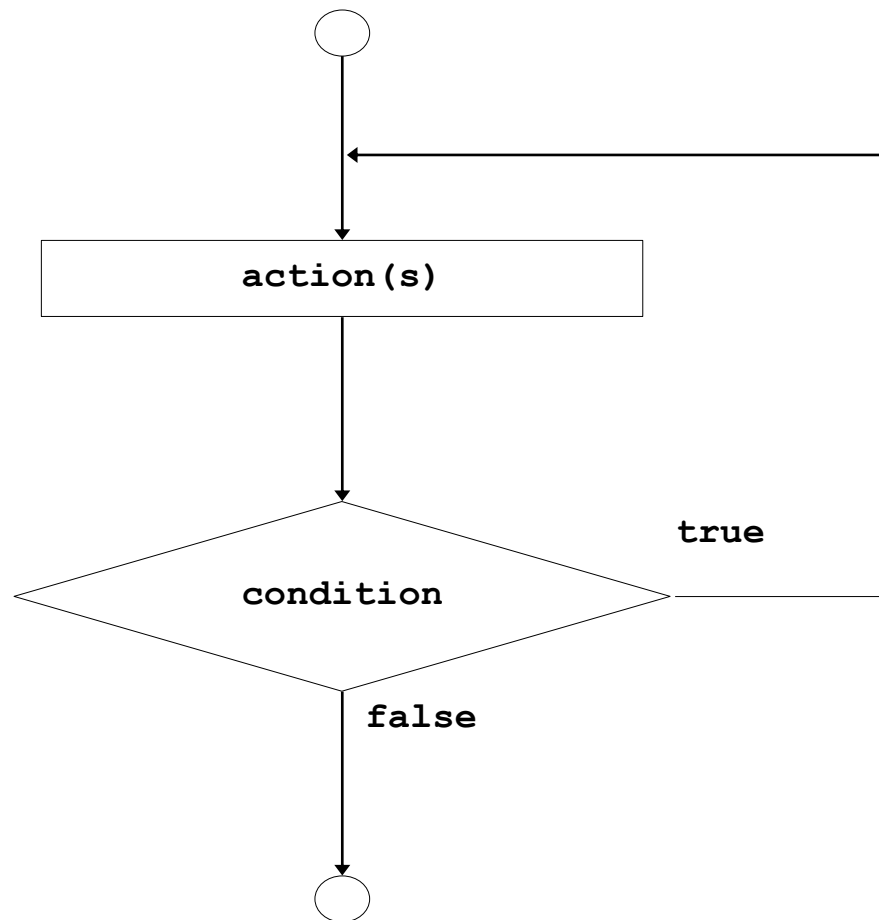
- do  
    statement;  
while(expression);

- do {  
    statement1;  
    statement2;  
} while(expression);

**S** note - the expression is tested *after* the statement(s) are executed, so statements are executed *at least once*.

## 4.8 The do...while Repetition Statement

- Flowchart of the do...while repetition statement



# Example

```
#include <stdio.h>
#define PI 3.141593

int main(void)
{
    int degrees=0;
    double radians;

    printf("Degrees to Radians \n");
    do
    {
        radians = degrees*PI/180;
        printf("%6i %9.6f \n",degrees,radians);
        degrees += 10;
    } while (degrees <= 360);
    return 0;
}
```

# for statement

- for(*initialization; test; increment or decrement*)  
statement;
- for(*initialization; test; increment or decrement*)  
{  
statement;  
statement;  
.  
}

# Example

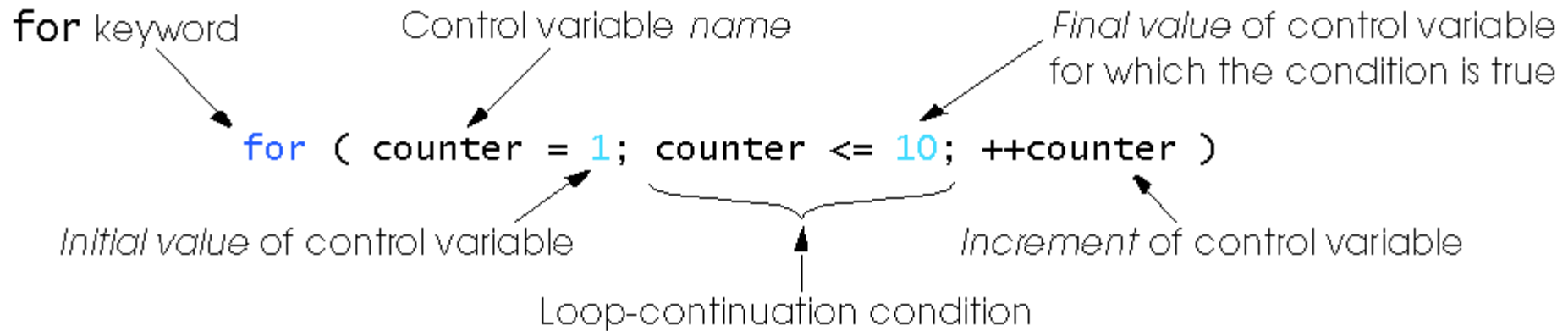
```
#include <stdio.h>
#define PI 3.141593

int main(void)
{
    int degrees;
    double radians;

    printf("Degrees to Radians \n");
    for (degrees=0; degrees<=360; degrees+=10)
    {
        radians = degrees*PI/180;
        printf("%6i %9.6f \n", degrees, radians);
    }
    return 0;
}
```



## 4.4 The for Repetition Statement



## 4.4 The for Repetition Statement

- Format when using for loops

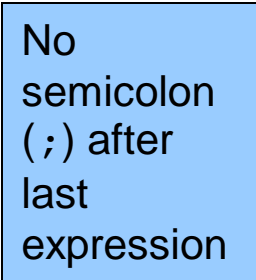
*for ( initialization; loopContinuationTest; increment )  
statement*

- Example:

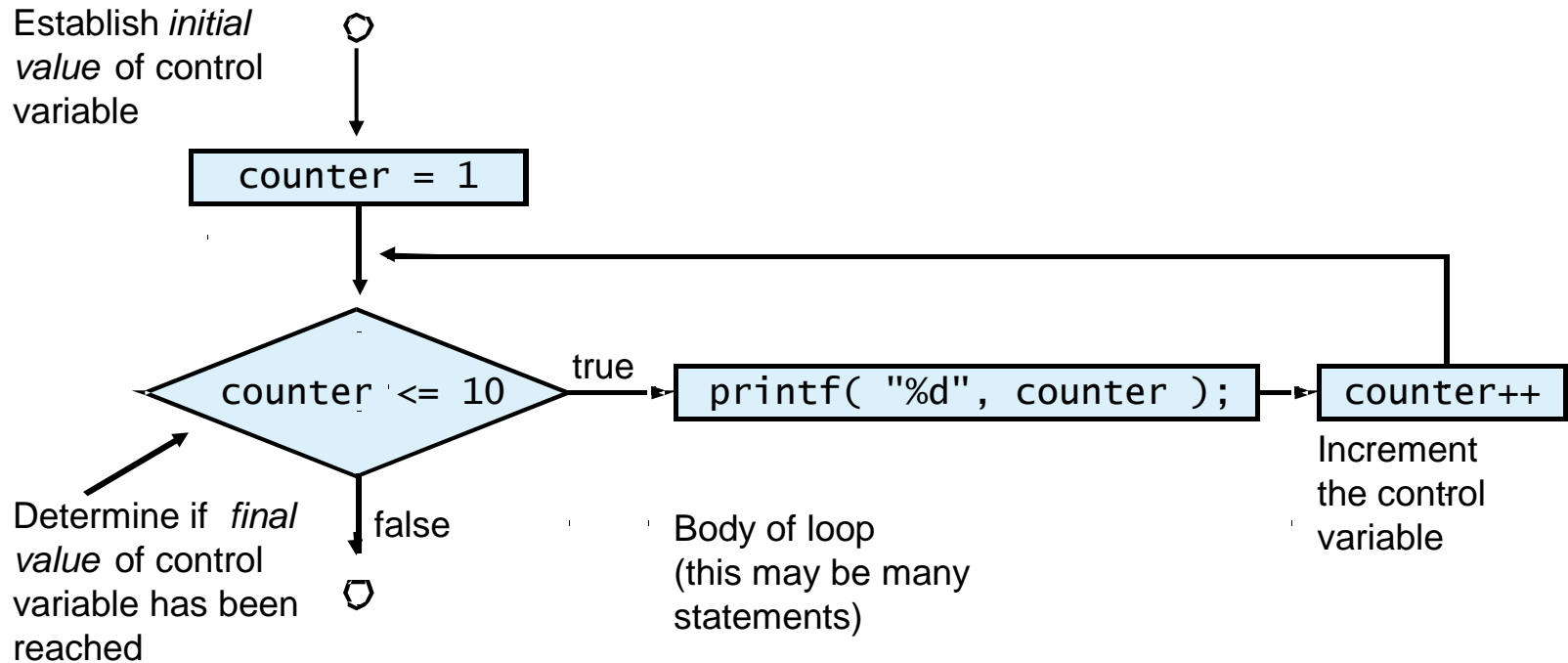
```
for(counter = 1; counter <= 10; counter++ )  
    printf( "%d\n", counter );
```

- Prints the integers from one to ten

No  
semicolon  
(;) after  
last  
expression



# Flow Chart of the Example **for** Loop



# The for Repetition Statement

- For loops can usually be rewritten as while loops:

```
initialization;  
while ( loopContinuationTest ) {  
    statement;  
    increment;  
}
```

- Initialization and increment

- Can be more than one, comma-separated lists of statements
- Can even add the counter variable declaration in *initialization*
- Example:

```
for (int i = 0, j = 0;  j + i <= 10; j++, i++)  
    printf( "%d\n", j + i );
```

# Examples

```
int sum = 0;  
for(int i=1; i < 7; i+=2)  
    sum = sum + i;
```



```
int fact = 1;  
for(int n=5; n > 1; n--)  
    fact = fact * n;
```

<del>?</del> <del>1</del> <del>3</del> <del>5</del> 7	i
<del>0</del> <del>1</del> <del>4</del> 9	sum
5	n
1	fact

# Exercise

Determine the number of times that each of the following for loops are executed.

```
for (k=3; k<=10; k++) {  
    statements;  
}
```

```
for (k=3; k<=10; ++k) {  
    statements;  
}
```

```
for(count=-2; count<=5; count++) {  
    statements;  
}
```

$$\left\lfloor \frac{\textit{final} - \textit{initial}}{\textit{increment}} \right\rfloor + 1$$

# Example

- What will be the output of the following program, also show how values of variables change in the memory.

```
int sum1, sum2, k;
sum1 = 0;
sum2 = 0;
for( k = 1; k < 5; k++) {
    if( k % 2 == 0)
        sum1 =sum1 + k;
    else
        sum2 = sum2 + k;
}
printf("sum1 is %d\n", sum1);
printf("sum2 is %d\n", sum2);
```

0	2	6	sum1		
0	1	4	sum2		
1	2	3	4	5	k

```
sum1 is 6
sum2 is 4
```

# break statement

- **break;**
  - terminates loop
  - execution continues with the first statement following the loop

```
sum = 0;
for (k=1; k<=5; k++) {
    scanf("%lf",&x);
    if (x > 10.0)
        break;
    sum +=x;
}
printf("Sum = %f \n",sum);
```



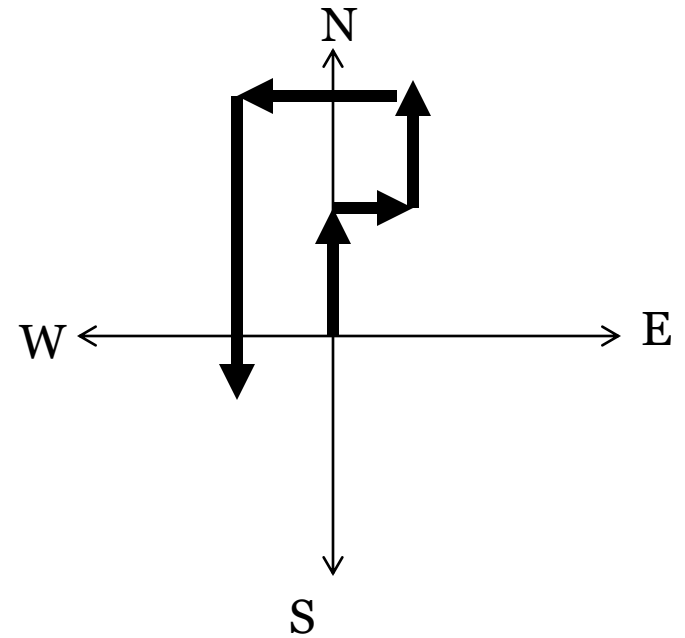
# continue statement

- **continue;**
  - forces next iteration of the loop, skipping any remaining statements in the loop

```
sum = 0;
for (k=1; k<=5; k++) {
    scanf("%lf",&x);
    if (x > 10.0)
        continue;
    sum +=x;
}
printf("Sum = %f \n",sum);
```

# Example: A man walks

- Suppose a man (say, A) stands at  $(0, 0)$  and waits for user to give him the direction and distance to go.
- User may enter N E W S for north, east, west, south, and any value for distance.
- When user enters 0 as direction, stop and print out the location where the man stopped



```
float x=0, y=0;
char direction;
float mile;
while (1) {
    printf("Please input the direction as N,S,E,W (o to exit): ");
    scanf("%c", &direction);    fflush(stdin);
    if (direction=='o'){ /*stop input, get out of the loop */
        break;
    }
    if (direction!='N' && direction!='S' && direction!='E' && direction!='W') {
        printf("Invalid direction, re-enter \n");
        continue;
    }
    printf("Please input the mile in %c direction: ", direction);
    scanf ("%f",&mile);  fflush(stdin);
    if (direction == 'N'){           /*in north, compute the y*/
        y+=mile;
    } else if (direction == 'E'){     /*in east, compute the x*/
        x+=mile;
    } else if (direction == 'W'){     /*in west, compute the x*/
        x-=mile;
    } else if (direction == 'S'){     /*in south, compute the y*/
        y-=mile;
    }
}
printf("\nCurrent position of A: (%4.2f,%4.2f)\n",x,y);    /* output A's location */
```

# Example: what will be the output

```
int main()
{
    int a, b, c;
    a=5;
    while(a > 2) {
        for (b = a ; b < 2 * a ; b++ ) {
            c = a + b;
            if (c < 8) continue;
            if (c > 11) break;
            printf( "a = %d  b = %d  c = %d \n", a, b, c);
        } /* end of for-loop */
        a--;
    } /* end of while loop */
}
```

a = 5	b = 5	c = 10
a = 5	b = 6	c = 11
a = 4	b = 4	c = 8
a = 4	b = 5	c = 9
a = 4	b = 6	c = 10
a = 4	b = 7	c = 11
a = 3	b = 5	c = 8

# More loop examples

# Exercise

- What is the output of the following program?

```
for (i=1; i<=5; i++) {  
    for (j=1; j<=4; j++) {  
        printf("*");  
    }  
    printf("\n");  
}
```

Output

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

# Exercise

- What is the output of the following program?

```
for (i=1; i<=5; i++) {  
    for (j=1; j<=i; j++) {  
        printf("*");  
    }  
    printf("\n");  
}
```

Output

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

Example: **nested loops** to generate the following output

i=1 \*  
i=2 + +  
i=3 \* \* \*  
i=4 + + + +  
i=5 \* \* \* \* \*

```
int i, j;  
for(i=1; i <= 5; i++){  
    printf("i=%d ", i);  
    for(j=1; j <= i; j++) {  
        if (i % 2 == 0)  
            printf("+ ");  
        else  
            printf("* ");  
    }  
    printf("\n");  
}
```



Exercise: Modify the following program to produce the output.

```
for (i=A; i<=B; i++) {  
    for (j=C; j<=D; j++) {  
        printf("*");  
    }  
    printf("\n");  
}
```

Output

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

# Exercise

- Write a program using loop statements to produce the output.

Output

\*

\*\*

\*\*\*

\*\*\*\*

\*\*\*\*\*

# For vs. while loop

Convert the following for loop to while loop

```
for( i=5; i<10; i++) {  
    printf(" i = %d \n", i);  
}
```

```
i=5;  
while(i<10) {  
    printf(" i = %d \n", i);  
    i++;  
}
```

# For vs. while loop : Convert the following for loop to while loop

```
for( i=5; i<10; i++) {  
    printf("AAA %d \n", i);  
    if (i % 2==0) continue;  
    pritntf("BBB %d \n", i);  
}
```

```
i=5;  
while(i<10) {  
    printf("AAA %d \n", i);  
    if (i % 2==0) {  
        i++;  
        continue;  
    }  
    pritntf("BBB %d \n", i);  
    i++;  
}
```

# Compute $x^y$ when $y$ is integer

- Suppose we don't have  $\text{pow}(x,y)$  and  $y$  is integer, write a loop to compute  $x^y$

```
Enter x, y  
  
res=1;  
  
for(i=1; i<=y; i++){  
    res = res * x;  
}
```

## Exercise: sum

- Write a program to compute the following

$$\sum_{i=1}^n i = 1 + 2 + 3 + \dots + n$$

$$total = 2 + 4 + 6 + \dots + 2n$$

Enter n

```
total=0;
```

```
for(i=1; i<=n; i++)
```

```
    total = total + i ;
```

```
print total
```

Enter n

```
total=0;
```

```
for(i=1; i<=n; i++)
```

```
    total = total + 2 * i ;
```

```
print total
```

## Exercise: sum

- Write a program to compute the following

$$\sum_{i=0}^m x^i = x^0 + x^1 + x^2 + x^3 + x^4 + \cdots + x^m$$

Enter x and m

```
total=0;
```

```
for(i=0; i<=m; i++)
```

```
    total = total + pow(x, i);
```

```
print total
```

Enter x and m

```
total=0; sofarx=1;
```

```
for(i=0; i<=m; i++) {
```

```
    total = total +sofarx;
```

```
    sofarx = sofarx * x;
```

```
}
```

```
print total
```

## Exercise: ln 2

- Write a program to compute the following

$$\ln 2 = \frac{1}{1} - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \frac{1}{5} - \frac{1}{6} + \frac{1}{7} - \dots \pm \frac{1}{n}$$

```
Enter n
ln2=0;
for(i=1; i<=n; i++)
    if ( i % 2 == 0)
        ln2 = ln2 - 1.0 / i;
    else
        ln2 = ln2 + 1.0 / i;
print total
```



## Exercise: $e^x$

- Write C program that reads the value of  $x$  and  $n$  from the keyboard and then approximately computes the value of  $e^x$  using the following formula:

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots + \frac{x^n}{n!}$$

- Then compare your approximate result to the one returned by `exp(x)` in C library, and print out whether your approximation is higher or lower.

```
int    i, n;
double x, ex;
double powx, fact;

printf("Enter the value of x and n : ");
scanf("%lf %d",&x, &n);

/* Write a loop to compute e^x using the above formula */
ex=1.0;    fact=1.0;    powx=1.0;
for(i=1; i<=n; i++){
    powx = powx * x;
    fact = fact * i;
    ex = ex + powx / fact;
}
printf("Approx value of e^x is %lf when n=%d\n",ex, n);

/* Check if ex is higher/lower than exp(x) in math lib.*/
if(ex < exp(x))
    printf("ex est is lower than exp(x)=%lf\n",exp(x));
else if (ex > exp(x))
    printf("ex est is higher than exp(x)=%lf\n",exp(x));
else
    printf("ex est is the same as exp(x)\n");
```

## Exercise: sin x

- Compute sin x using

$$\sin x = \frac{x}{1!} - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \dots - (-1)^n \frac{x^{2n+1}}{(2n+1)!}$$

```
Get n
total=0; powx=x; factx=1;
for(i=0; i <= n; i++){
    k= 2*n+1;
    if (i%2==0) total= total - powx/factx;
    else total= total + powx/factx;
    powx= powx * x * x;
    factx = factx * k * (k-1);
}
Print total;
```

# Example

- Write a program that prints in two columns n even numbers starting from 2, and a running sum of those values. For example suppose user enters 5 for n, then the program should generate the following table:

Enter n (the number of even numbers): 5

Value	Sum
-------	-----

2	2
---	---

4	6
---	---

6	12
---	----

8	20
---	----

10	30
----	----

```
#include <stdio.h>
int main(void)
{
    /*  Declare variables.  */
    int n;
    int sum, i;

    printf("Enter  n ");
    scanf("%d", &n);

    printf("Value \t Sum\n");
    sum = 0;
    for(i=1; i <=n; i++){
        sum = sum + 2*i;
        printf("%d \t %d\n", 2*i, sum);
    }
    return 0;
}
```