Lecture 6

ASCII Code

- Each character is assigned a unique integer value (code) between 32 and 127
- The code of a character is represented by an 8-bit unit.
  - Since an 8-bit unit can hold a total of 2^8=256 values and the computer character set is much smaller than that, some values of this 8-bit unit do not correspond to visible characters
- But not a good idea to remember exact ASCII codes while programming. Use the facts that
  - C stores characters as integers
  - Ascii codes of some important characters are contiguous (digits, lowercase alphabets, uppercase alphabets)

Decimal	Hex	Binary	Character	Decimal	Hex	Binary	Character
32	20	00100000	SPACE	80	50	01010000	P
33	21	00100001	!	81	51	01010001	Q
34	22	00100010	"	82	52	01010010	R
35	23	00100011	#	83	53	01010011	S
36	24	00100100	$	84	54	01010100	T
37	25	00100101	%	85	55	01010101	U
38	26	00100110	&	86	56	01010110	V
39	27	00100111	'	87	57	01010111	W
40	28	00101000	(	88	58	01011000	X
41	29	00101001	)	89	59	01011001	Y
42	2a	00101010	*	90	5a	01011010	Z
43	2b	00101011	+	91	5b	01011011	[
44	2c	00101100	,	92	5c	01011100	\
45	2d	00101101	-	93	5d	01011101	]
46	2e	00101110	.	94	5e	01011110	^
47	2f	00101111	/	95	5f	01011111	_
48	30	00110000	0	96	60	01100000	`
49	31	00110001	1	97	61	01100001	a
50	32	00110010	2	98	62	01100010	b

51	33	00110011	3	99	63	01100011	c
52	34	00110100	4	100	64	01100100	d
53	35	00110101	5	101	65	01100101	e
54	36	00110110	6	102	66	01100110	f
55	37	00110111	7	103	67	01100111	g
56	38	00111000	8	104	68	01101000	h
57	39	00111001	9	105	69	01101001	i
58	3a	00111010	:	106	6a	01101010	j
59	3b	00111011	;	107	6b	01101011	k
60	3c	00111100	<	108	6c	01101100	l
61	3d	00111101	=	109	6d	01101101	m
62	3e	00111110	>	110	6e	01101110	n
63	3f	00111111	?	111	6f	01101111	o
64	40	01000000	@	112	70	01110000	p
65	41	01000001	A	113	71	01110001	q
66	42	01000010	B	114	72	01110010	r
67	43	01000011	C	115	73	01110011	s
68	44	01000100	D	116	74	01110100	t
69	45	01000101	E	117	75	01110101	u
70	46	01000110	F	118	76	01110110	v

71	47	01000111	G		119	77	01110111	w
72	48	01001000	H		120	78	01111000	x
73	49	01001001	I		121	79	01111001	y
74	4a	01001010	J		122	7a	01111010	z
75	4b	01001011	K		123	7b	01111011	{
76	4c	01001100	L		124	7c	01111100	|
77	4d	01001101	M		125	7d	01111101	}
78	4e	01001110	N		126	7e	01111110	~
79	4f	01001111	O		127	7f	01111111	DELETE

Quiz…

Expression	Value?
‘9’ >= ‘0’	1 (true)
‘a’ < ‘e’	1 (true)
‘Z’ == ‘z’	0 (false)
‘a’ <= ‘A’	0 (false)

Example: checking if a character is a lowercase alphabet

```
int main()
{   /* Read a character and display whether it is lower case or upper case */
        char c1;
        scanf("%c", &c1);
      /* the ascii code of c1 must lie between the
         ascii codes of 'a' and 'z' */
        if (c1 >= 'a' && c1<= 'z')
            printf("%c is a lowercase alphabet\n", c1);
        else printf("%c is not a lowercase alphabet\n", c1);
        return 0;
}
```

Example: converting a character from lowercase to uppercase

```
int main()
{
        char c1;
        scanf("%c", &c1);
        /* convert to uppercase if lowercase, else leave as it is */
        if (c1 >= 'a' && c1<= 'z')
        /* since ascii codes of uppercase letters are contiguous, the
           uppercase version of c1 will be as far away from the ascii code
           of 'A' as it is from the ascii code of 'a' */
        c1 = 'A' + (c1 – 'a');
        printf(("The letter is %c\n", c1);
        return 0;
}
```

Exercise

- Write a program that:
  - When the user enters a or A, displays “First letter”
  - When the user enters z or Z, displays “last letter”.
  - For any other letter entered by the user it displays “middle letter”.

Switching with char type

```
char letter;
scanf(“%c”, &letter);
switch ( letter ) {
    case 'A':
        printf ("First letter \n");
        break;
    case 'Z':
        printf ("Last letter \n");
        break;
    default :
        printf ("Middle letter \n");
}
```

Will print this statement for all letters other than A or Z

Switching with char type

```
char letter;
scanf(“%c”, &letter);
switch ( letter ) {
    case 'A’:
    case ‘a’:
        printf ("First letter \n");
        break;
    case 'Z’:
    case ‘z’:
        printf ("Last letter \n");
        break;
    default :
        printf ("Middle letter \n");
}
```

Switching with char type

```
char letter;
scanf(“%c”, &letter);
switch ( letter ) {
    case 'A':
        printf ("First letter \n");
        break;
    case 'Z':
        printf ("Last letter \n");
        break;
    default :
        printf ("Middle letter \n");
}
```

Will print this statement for all letters other than A or Z

Another Example

```
switch (choice = getchar()) {
    case ‘r’ :
    case ‘R’: printf(“Red”);
                break;
    case ‘b’ :
    case ‘B’ : printf(“Blue”);
               break;
    case ‘g’ :
    case ‘G’: printf(“Green”);
               break;
    default: printf(“Black”);
}
```

Another Example

```
switch (choice = getchar()) {
    case ‘r’ :
    case ‘R’: printf(“Red”);
              break;
    case ‘b’ :
    case ‘B’ : printf(“Blue”);
               break;
    case ‘g’ :
    case ‘G’: printf(“Green”);
               break;
    default: printf(“Black”);
}
```

Since there isn’t a break statement here, the control passes to the next statement (printf) without checking the next condition.

Evaluating expressions

```
int main () {
    int operand1, operand2;
    int result = 0;
    char operation ;
    /* Get the input values */
    printf ("Enter operand1 :");
    scanf("%d",&operand1) ;
    printf ("Enter operation :");
    scanf ("\n%c",&operation);
    printf ("Enter operand 2 :");
    scanf ("%d", &operand2);
    switch (operation)   {
    case '+' :
        result=operand1+operand2;
        break;
    case '-' :
        result=operand1-operand2;
        break;
    case '*' :
        result=operand1*operand2;
        break;
    case '/' :
        if (operand2 !=0)
            result=operand1/operand2;
        else
            printf("Divide by 0 error");
        break;
    default:
        printf("Invalid operation\n");
        return;
    }
    printf ("The answer is %d\n",result);
    return 0;
}
```

Practice Problems

1. Read in 3 integers and print a message if any one of them is equal to the sum of the other two.

2. Read in the coordinates of two points and print the equation of the line joining them in y = mx +c form.

3. Read in the coordinates of 3 points in 2-d plane and check if they are collinear. Print a suitable message.

4. Read in the coordinates of a point, and the center and radius of a circle. Check and print if the point is inside or outside the circle.

5. Read in the coefficients a, b, c of the quadratic equation $ax^2 + bx + c = 0$, and print its roots nicely (for imaginary roots, print in x + iy form)

6. Suppose the digits 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 are mapped to the lowercase letters a, b, c, d, e, f, g, h, i, j respectively. Read in a single digit integer as a character (using %c in scanf) and print its corresponding lowercase letter. Do this both using switch and without using switch (two programs). Do not use any ascii code value directly.

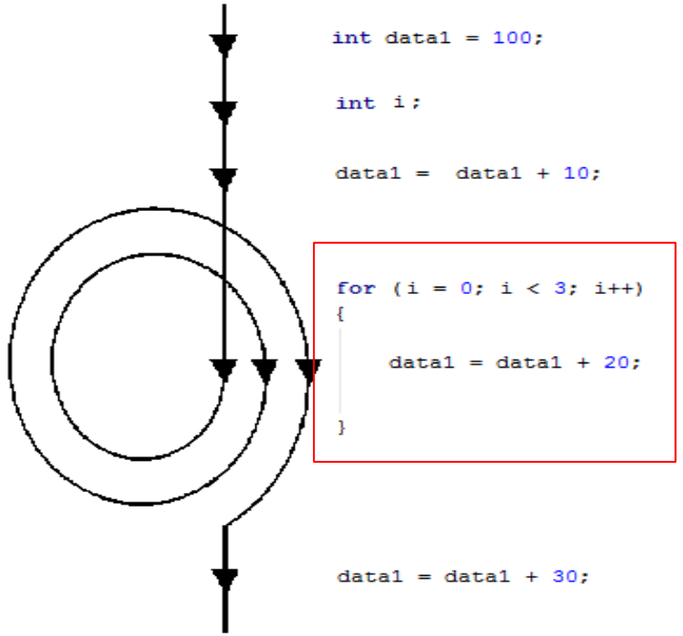
7. Suppose that you have to print the grades of a student, with >= 90 marks getting EX, 80-89 getting A, 70-79 getting B, 60-69 getting C, 50-59 getting D, 35-49 getting P and <30 getting F. Read in the marks of a student and print his/her grade.

Lecture 7

Looping

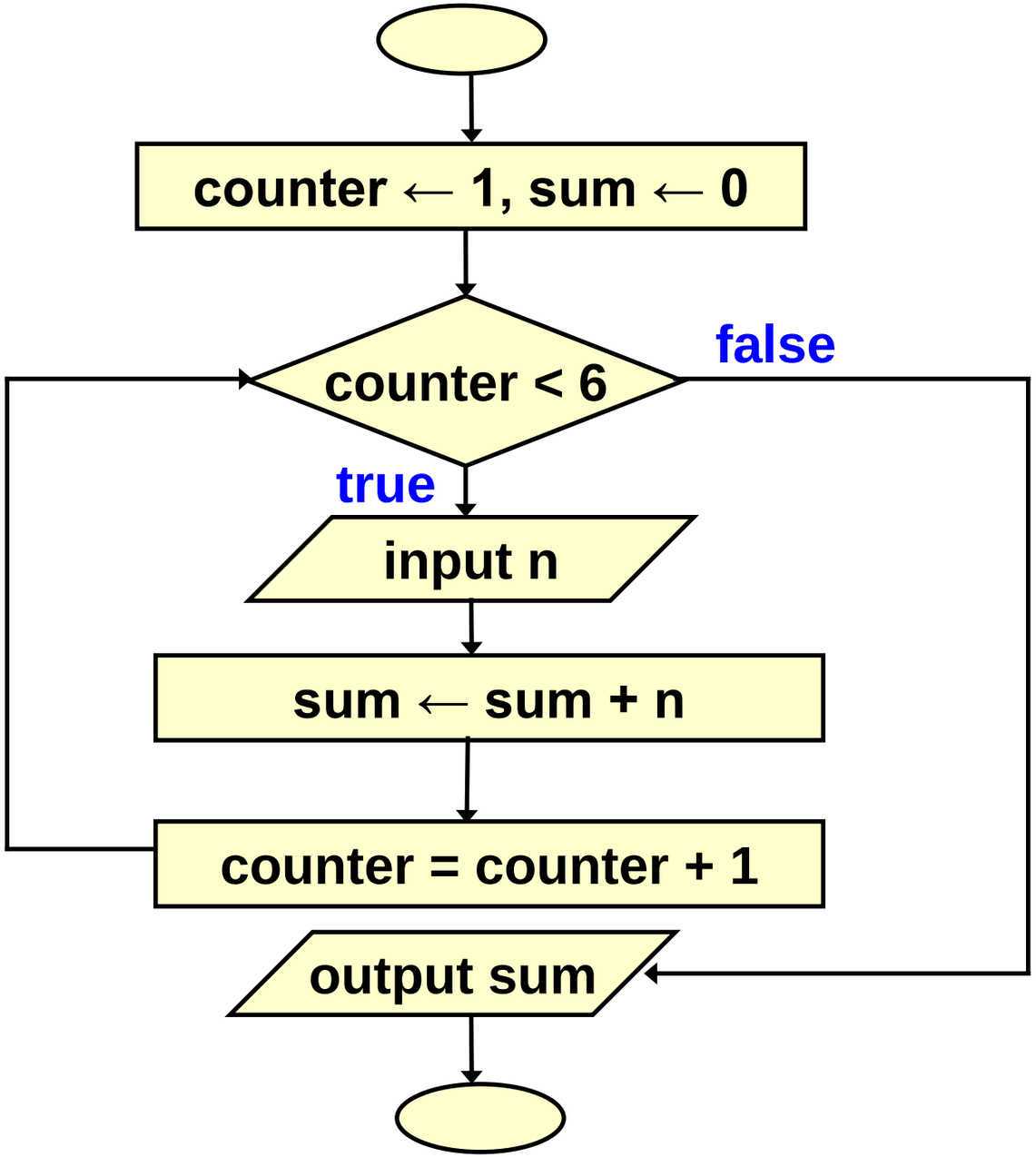
Loops

- Group of statements that are executed repeatedly while some condition remains true
- Each execution of the group of statements is called an iteration of the loop
- Examples:
  - Keep on dividing a number by two and display the remainder until the number becomes 1 or 0.
  - Multiply a number with itself n times.
  - Keep on reading a number from key board and adding, until the user enters 0.



Example

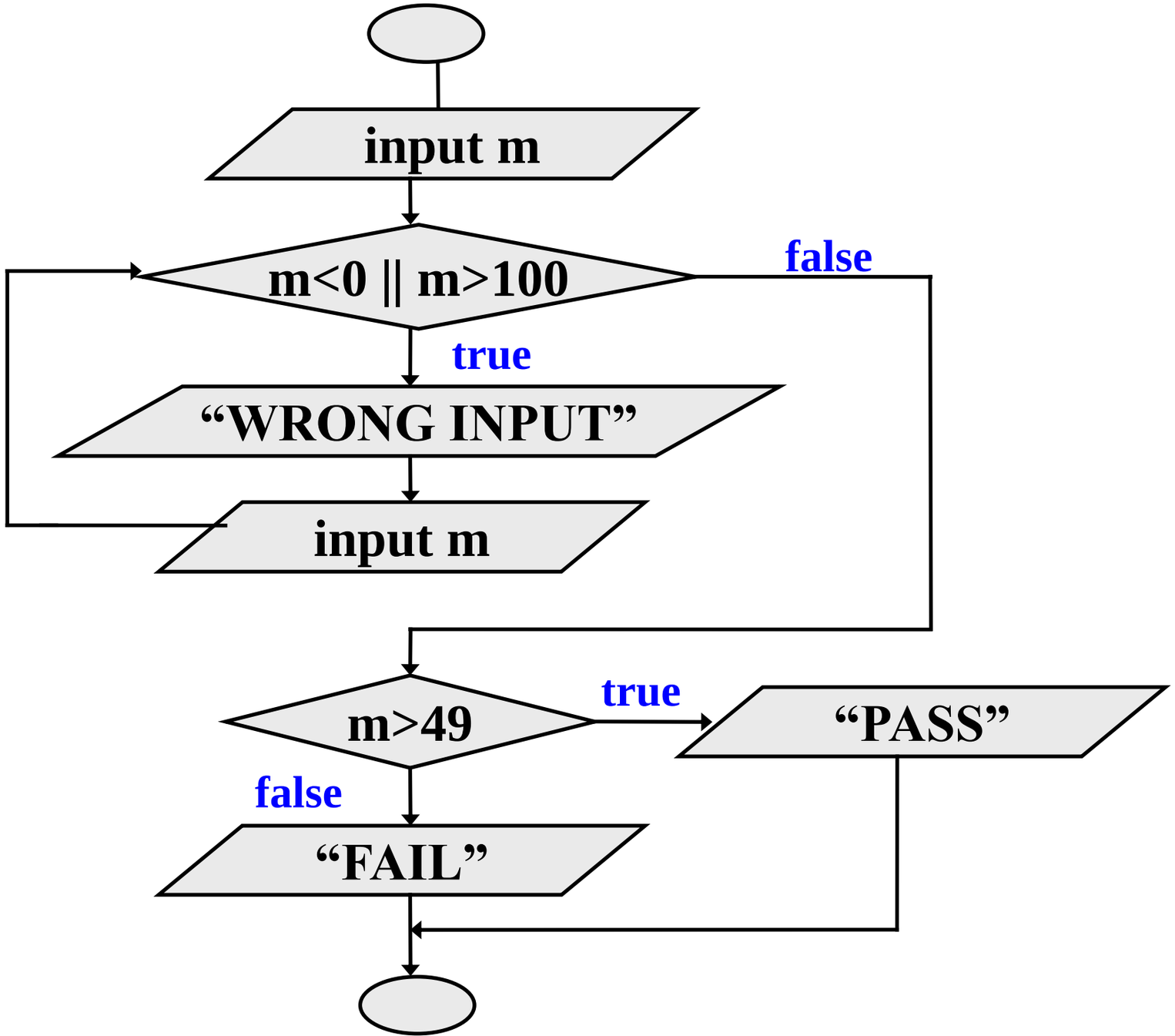
Read 5 integers and display the their sum

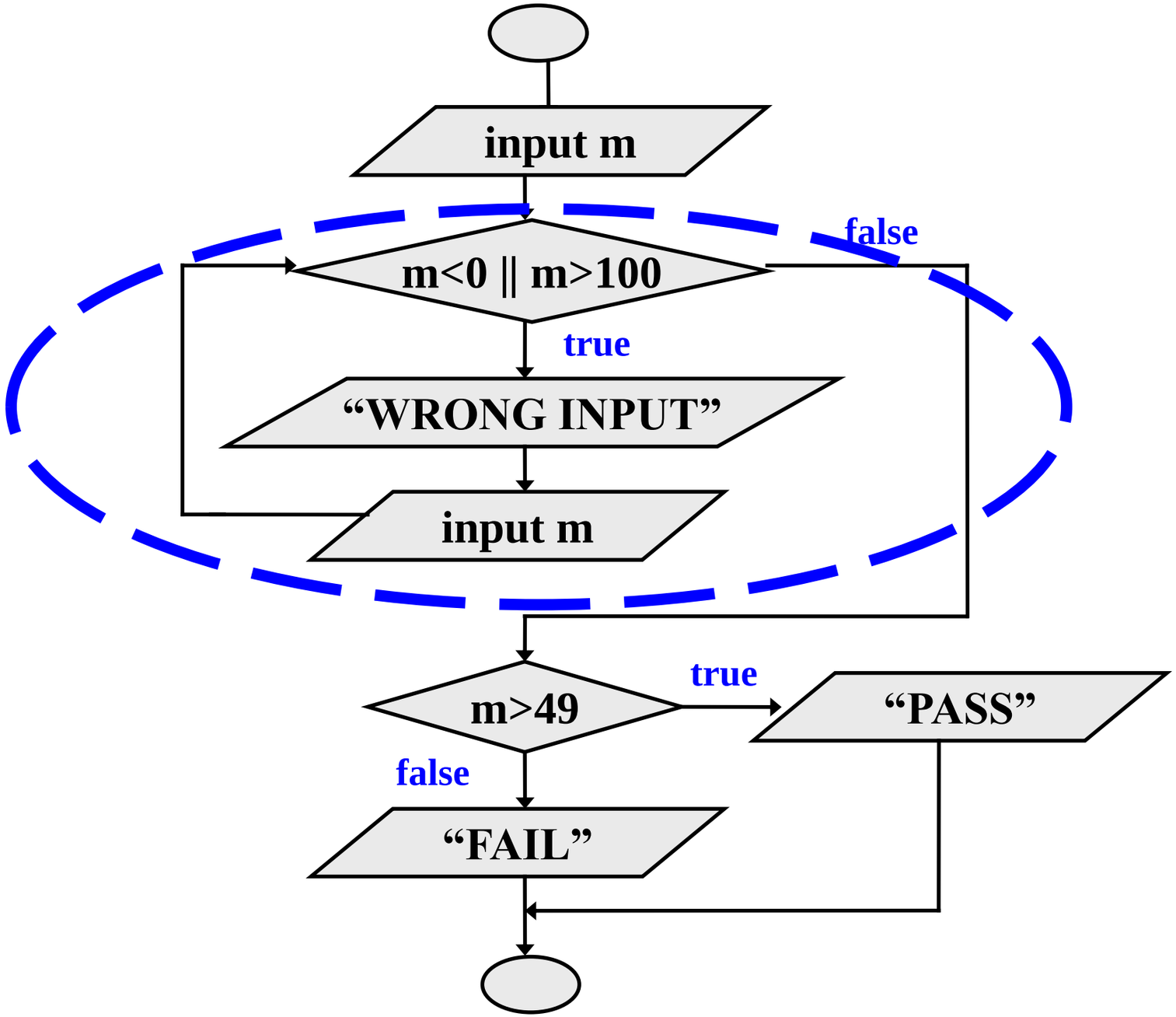


Example

Read 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…


input m
m<0 || m>100
false
true
“WRONG INPUT”
input m
m>49
true
“PASS”
false
“FAIL”


input m
m<0 || m>100
false
true
“WRONG INPUT”
input m
m>49
true
“PASS”
false
“FAIL”

Types of Loops

- Loops are controlled by boolean expressions
- C has three kinds of loops:
  - ***while loop***
  - ***do loop***
  - ***for loop***

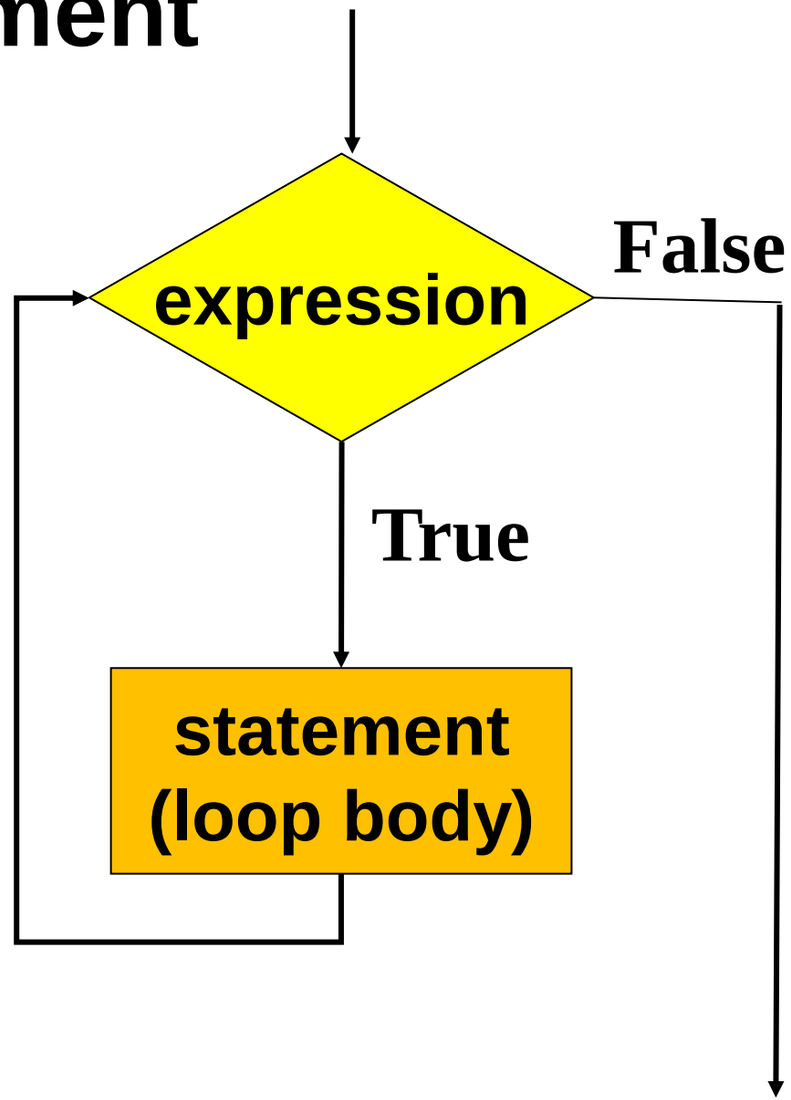
Looping: while statement

```
while (expression)
      statement;

while (expression) {
   Block of statements;
}
```

The condition to be tested is a logical expression enclosed in parentheses. The expression is evaluated, and if its value is non-zero, the statement is executed. Then the expression is evaluated again and the same thing repeats. The loop terminates when the expression evaluates to 0.

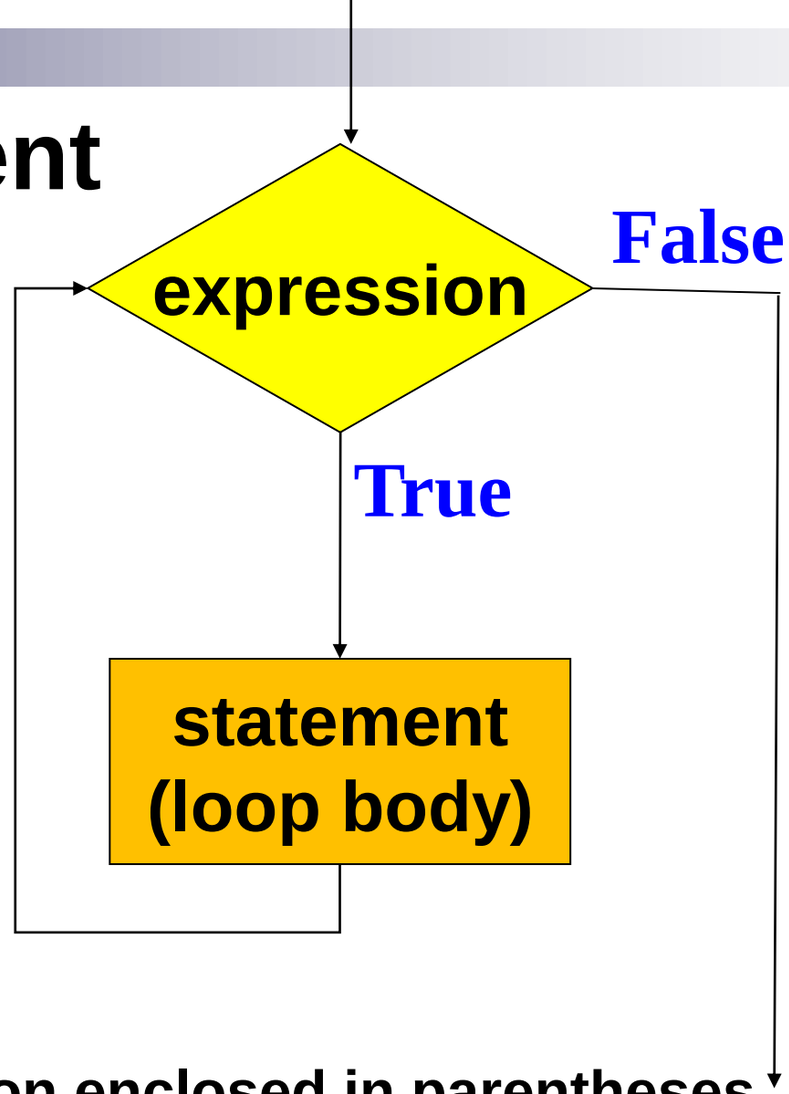
Looping: while statement

```
while (expression)
    statement;

while (expression) {
   Block of statements;
}
```



Looping: while statement

```
while (expression)
        statement;

while (expression) {
     Block of statements;
}
```



The condition to be tested is any expression enclosed in parentheses. The expression is evaluated, and if its value is non-zero, the statement is executed. Then the expression is evaluated again and the same thing repeats. The loop terminates when the expression evaluates to 0.

Example

```
int main(){
    int i = 1, n;
    scanf("%d", &n);
    while (i <= n)  {
        printf ("Line no : %d\n",i);
        i = i + 1;
    }
}
```

Output

```
4
Line no : 1
Line no : 2
Line no : 3
Line no : 4
```

Example: Exercise Till Lose Weight

```
int  weight;
scanf("%d", &weight);
printf ("Weight is: %d\n", weight);
while ( weight > 65 ) {
    printf ("Go, exercise, ");
    printf ("then come back… \n");
    printf ("Measure and Enter your weight: ");
    scanf ("%d", &weight);
 }
```

Sum of first N natural numbers

```
int main() {
      int N, count, sum;
      scanf ("%d", &N) ;
      sum = 0;
      count = 1;
      while (count <= N)  {
            sum = sum + count;
            count = count + 1;
      }
      printf ("Sum = %d\n", sum) ;
      return 0;
}
```

Output

```
9
Sum of first 9 numbers = 45
```

SUM = $1^2 + 2^2 + 3^2 + \ldots + N^2$

```
int main() {
      int N, count, sum;
      scanf ("%d", &N) ;
      sum = 0;
      count = 1;
      while (count <= N)  {
            sum = sum + count * count;
            count = count + 1;
      }
      printf ("Sum = %d\n", sum) ;
      return 0;
}
```

- Suppose your Rs 10000 is earning interest at 1% per month. How many months for your money to double?

```
int main() {
   double my_money = 10000.0;
   int n=0;
   while (my_money < 20000.0) {
       my_money = my_money * 1.01;
       n++;
   }
  printf (“My money will double in %d months.\n”,n);
  return 0;
}
```

Time to double your money in bank…

```
int main() {
    double max = 0.0, next;
    printf (“Enter positive numbers only, end with 0 or a
     negative number\n”);
    scanf(“%lf”, &next);
    while (next > 0)  {
        if (next > max)  max = next;
        scanf(“%lf”, &next);
    }
    printf (“The maximum number is %lf\n”, max) ;
    return 0;
}
```

Maximum of positive numbers entered

Output

```
Enter positive numbers only, end with 0 or a negative number
45
32
7
5
0
The maximum number is 45.000000
```

Find the sum of digits of a number

```
int main(){
    int   n, sum=0;
    scanf ("%d", &n);
    while (n != 0)  {
          sum = sum + (n % 10);
          n = n / 10;
    }
    printf ("The sum of digits is %d \n", sum);
     return 0;
}
```

Output

```
573254
The sum of digits is 26
```

Compute GCD of two numbers

```
int main() {
    int  A, B, temp;
    scanf ("%d %d", &A, &B);
    if  (A > B)  {
        temp = A;  A = B;  B = temp;
   }
    while ((B % A) != 0)  {
            temp = B % A;
            B = A;
            A = temp;
    }
    printf ("The GCD is %d", A);
   return 0;
}
```

Euclid's Algorithm

```
12 ) 45 ( 3
     36
      9 ) 12 ( 1
          9
          3 ) 9 ( 3
              9
              0
```

Initial: A=12, B=45
Iteration 1: temp=9, B=12,A=9
Iteration 2: temp=3, B=9, A=3
B % A = 0 ➔ GCD is 3

Exercise

- Write a program to determine the most significant digit of the value stored in an integer variable num.

- For example:
  - If num=457138, your program should display 4.

Find the most significant digit of a number

```
int main(){
    int   n, msdigit=0;
    scanf ("%d", &n);
    while (n != 0)  {
          msdigit = n % 10;
          n = n / 10;
    }
    printf ("Most significant digit is %d \n", msdigit);
     return 0;
}
```

Lecture 8

Looping: for Statement

- Most commonly used looping structure in C

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

for ( expr1; expr2; expr3){
  Block of statements;
}
```

expr1 (init) : initialize parameters

expr2 (test): test condition, loop continues if expression is non-0

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

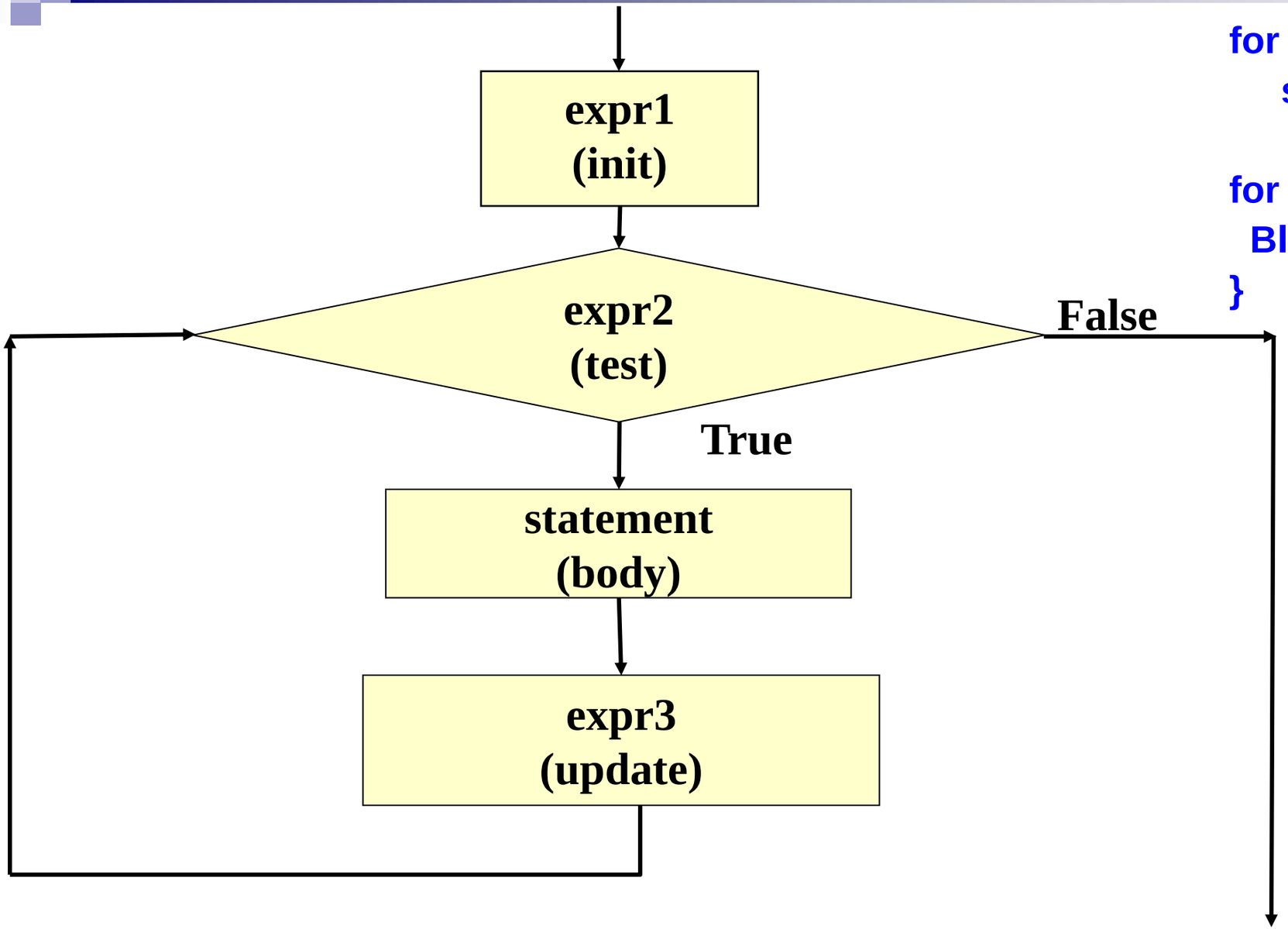
statement (body): body of loop

For Loop

For loop has three parts:

- ***initial value*** of the control variable.
- **condition** that tests whether control variable has reached the desired value.
- ***increment* (or *decrement*)** the control variable.

```
for ( initialization; loop condition; loop update )
{
   // loop body
}
```




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

for ( expr1; expr2; expr3){
  Block of statements;
}
```

First Example: Display 1 to 10

```
int counter =1;              /* initialization */

while (counter <= 10) {     /* repetition condition*/
     printf( "%d\n", counter );
    ++counter;               /* increment *
}
```

While Loop Version

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

For Loop version

Example 2: Compute Factorial

```
int main () {
    int N, count, prod;
    scanf (“%d”, &N) ;
    prod = 1;
    for (count = 1;count <= N; ++count)
        prod = prod * count;
    printf (“Factorial = %d\n”, prod) ;
    return 0;
}
```

Output

```
7
Factorial = 5040
```

Computing e^x series up to N terms

```
int main () {
      float x, term, sum;
      int n, count;
      scanf ("%f", &x);
      scanf ("%d", &n);
      term = 1.0; sum = 0;
      for (count = 1; count <= n; ++count)  {
            sum += term;
            term *= x/count;
       }
      printf ("%f\n", sum);
      return 0;
}
```

$$= 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$$

Output

```
2.3
10
The series sum is 7.506626
```

Computing e^x series up to 4 decimal places

```
int main() {
    float x, term, sum;
    int cnt;
    scanf (“%f”, &x) ;
    term = 1.0; sum = 0;
    for (cnt = 1; term >= 0.0001; ++cnt)  {
        sum += term;
        term *= x/cnt;
     }
     printf (“%f\n”, sum);
     return 0;
}
```

The comma operator

- We can give several statements separated by commas in an expression.

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

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

Equivalence of for and while

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

Same as

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

Sum of first N Natural Numbers

```
int main () {
    int N, count, sum;
    scanf (“%d”, &N) ;
    sum = 0;
    count = 1;
    while (count <= N)  {
        sum = sum + count;
        count = count + 1;
    }
    printf (“%d\n”, sum) ;
    return 0;
}
```

```
int main () {
    int N, count, sum;
    scanf (“%d”, &N) ;
    sum = 0;
    for (count=1; count <= N; ++count)
        sum = sum + count;
    printf (“%d\n”, sum) ;
    return 0;
}
```

Exercise

- Convert while Loop into for loop

```
int volume = 25;
int barrelSize = 200;
while(volume < barrelSize) {
        printf("The barrel is not full.\n");
        volume = volume +25;
}
```

Some observations on for

- Initialization, loop-continuation test, and update can contain arithmetic expressions

  for **(k = x; k <= 4 * x * y; k += y / x)**

- Update may be negative (decrement)

  for (digit = 9; digit >= 0; --digit)

- If loop continuation test is initially 0 (false)
  - Body of for structure not performed
    - No statement executed

```
for (count=1; 0; ++count)
        sum = sum + count;
```

  - Program proceeds with statement after for structure

Programming Exercise

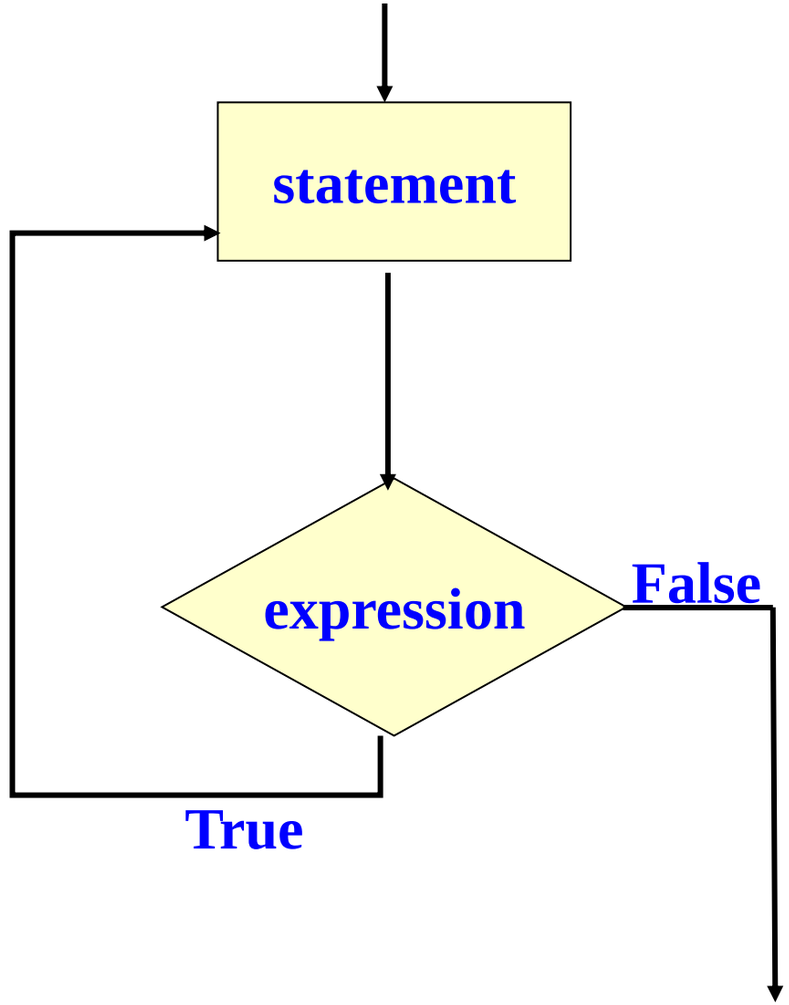
- Display all even numbers from 0 to 20

```
int i;
for(i=0;i<20;i+=2)
    printf("%d\n", i);
```

Looping: do-while statement

```
do
   statement;
while (expression);
```

```
do {
   Block of
    statements;
} while (expression);
```



Example

Problem: Prompt user to input “month” value, keep prompting until a valid month value is given as input…

```
do {
        printf (“Please input month {1-12}”);
        scanf (“%d”, &month);
} while ((month < 1) || (month > 12));
```

Decimal to binary conversion (prints binary in reverse order)

```
int main() {
     int  dec;
     scanf ("%d", &dec);
     do{
          printf ("%2d",  (dec % 2));
          dec = dec / 2;
     }  while (dec != 0);
     printf ("\n");
      return 0;
}
```

Output

```
277
101010001
```

Echo characters typed on screen until end of line

```
int main () {
      char echo ;
      do {
           scanf (“%c”, &echo);
             printf (“%c”,echo);
       }  while (echo != ‘\n’) ;
       return 0;
}
```

Output

```
This is a test line
This is a test line
```

Specifying “Infinite Loop”

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

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

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

The break Statement

- Break out of the loop body { }
  - can use with while, do while, for, switch
  - 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

An Example

```
int main() {
    int  fact, i;
    fact = 1;  i = 1;
    while  ( i<10 )    {    /* run loop –break when fact >100*/
            fact = fact * i;
            if ( fact > 100 )  {
                    printf ("Factorial of %d  above 100", i);
                    break;  /* break out of the while loop */
            }
            ++i;
    }
    return 0;
}
```

Test if a number is prime or not

```
int main() {
    int  n, i=2;
    scanf (“%d”, &n);
    limit = sqrt(n);
    for (i = 2, i <= limit; i++)  {
        if (n % i == 0)  {
            printf (“%d is not a prime \n”, n);
            break;
        }
    }
    if (i > limit) printf (“%d is a prime \n”, n);
    return 0;
}
```

Another Way

```
int main() {
     int  n, i = 2, flag = 0;
    double limit;
     scanf (“%d”, &n);
    limit = sqrt(n);
     while (i <= limit)  {
            if (n % i == 0)  {
                   printf (“%d is not a prime \n”, n);
                   flag = 1; break;
            }
            i = i + 1;
     }
     if (flag == 0) printf (“%d is a prime \n”, n);
    return 0;
}
```

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
  - Loop-continuation test is evaluated immediately after the continue statement is executed
- for loop
  - expr3 is evaluated, then expr2 is evaluated

Example with break and continue: Add positive numbers until a 0 is typed, but ignore any negative numbers typed

```
int main() {
    int sum = 0, next;
     while (1)  {
            scanf(“%d”, &next);
            if (next < 0)  continue;
            else if (next == 0) break;
            sum = sum + next;
     }
    printf (“Sum = %d\n”, sum) ;
    return 0;
}
```

Output

```
10
-20
30
40
-5
10
0
Sum = 90
```

Loops: Some Common Mistakes

```
while (sum <= NUM) ;
    sum = sum+2;
```

```
for (i=0; i<=NUM; ++i);
    sum = sum+i;
```

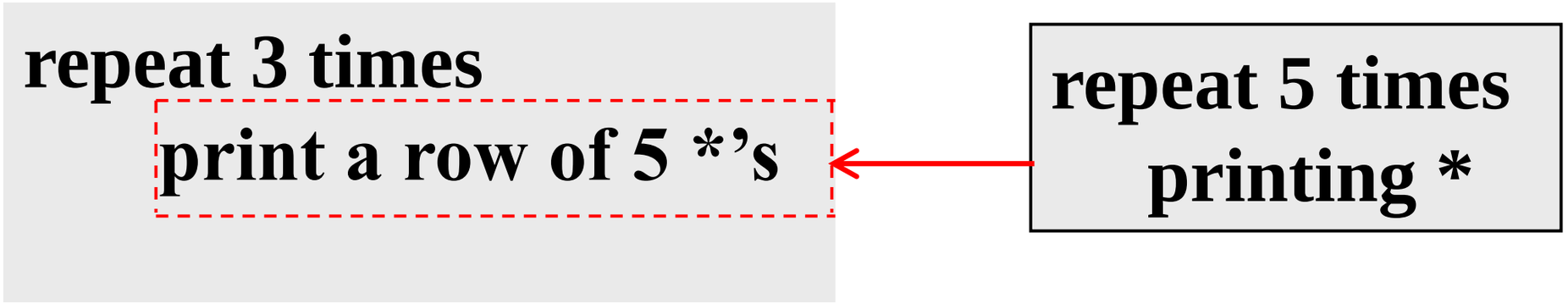
```
for (i=1; i!=10; i=i+2)
    sum = sum+i;
```

```
double x;
for (x=0.0; x != 2.0; x=x+0.2)
    printf("%.18f\n", x);
```

Nested Loops: Printing a 2-D Figure

- How would you print the following diagram?

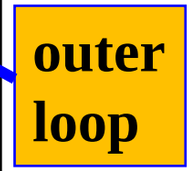
```
*****
*****
*****
```

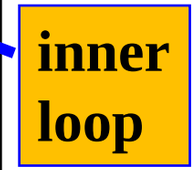


Display pattern: Configurable number of rows and columns

```
const int ROWS = 3;
const int COLS = 5;
...
row = 1;
while (row <= ROWS) {
 /* print a row of 5 *'s */
       ...
      ++row;
}
```

```
row = 1;
while (row <= ROWS) {
      /* print a row of 5 *'s */
      col = 1;
      while (col <= COLS) {
            printf ("* ");
            col++;
      }
      printf("\n");
      ++row;
}
```





2-D Figure: with for loop

Print

```
* * * * *

* * * * *

* * * * *
```

```
const int ROWS = 3;
const int COLS = 5;
....
for (row=1; row<=ROWS; ++row) {
    for (col=1; col<=COLS; ++col) {
        printf(“* ”);
    }
    printf(“\n”);
}
```

Another 2-D Figure

Print

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

```
const int ROWS = 5;
....
int row, col;
for (row=1; row<=ROWS; ++row) {
    for (col=1; col<=row; ++col) {
        printf("* ");
    }
    printf("\n");
}
```

Yet Another One

Print

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

```
const int ROWS = 5;
....
int row, col;
for (row=0; row<ROWS; ++row)  {
      for (col=1; col<=row; ++col)
            printf("  ");
      for (col=1; col<=ROWS-row; ++col)
            printf("* ");
      printf ("\n");
}
```

break and continue with nested loops

- For nested loops, break and continue are matched with the nearest loops (for, while, do-while)
- Example:

```
while (i < n) {
    for (k=1; k < m; ++k) {
        if  (k % i == 0) break;
    }
    i = i + 1;
}
```

Breaks to here

Example

```
int main()
{
      int low, high, desired, i, flag = 0;
      scanf("%d %d %d", &low, &high, &desired);
      i = low;
    while (i < high) {
        for (j = i+1; j <= high; ++j) {
             if  (j % i == desired) {
                 flag = 1;
               break;
            }
        }
        if (flag == 1) break;
        i = i + 1;
     }
      return 0;
}
```

Breaks to here

Breaks to here

Practice Problems (do each with both for and while loops separately)

1. Read in an integer N. Then print the sum of the squares of the first N natural numbers
2. Read in an integer N. Then read in N numbers and print their maximum and second maximum (do not use arrays even if you know it)
3. Read in an integer N. Then read in N numbers and print the number of integers between 0 and 10 (including both), between 11 and 20, and > 20. (do not use arrays even if you know it)
4. Repeat 3, but this time print the average of the numbers in each range.
5. Read in a positive integer N. If the user enters a negative integer or 0, print a message asking the user to enter the integer again. When the user enters a positive integer N finally, find the sum of the logarithmic series ($\log_e(1+x)$) upto the first N terms
6. Read in an integer N. Then read in integers, and find the sum of the first N positive integers read. Ignore any negative integers or 0 read (so you may actually read in more than N integers, just find the sum with only the positive integers and stop when N such positive integers are read)
7. Read in characters until the '\n' character is typed. Count and print the number of lowercase letters, the number of uppercase letters, and the number of digits entered.