PDS Lab Section 15

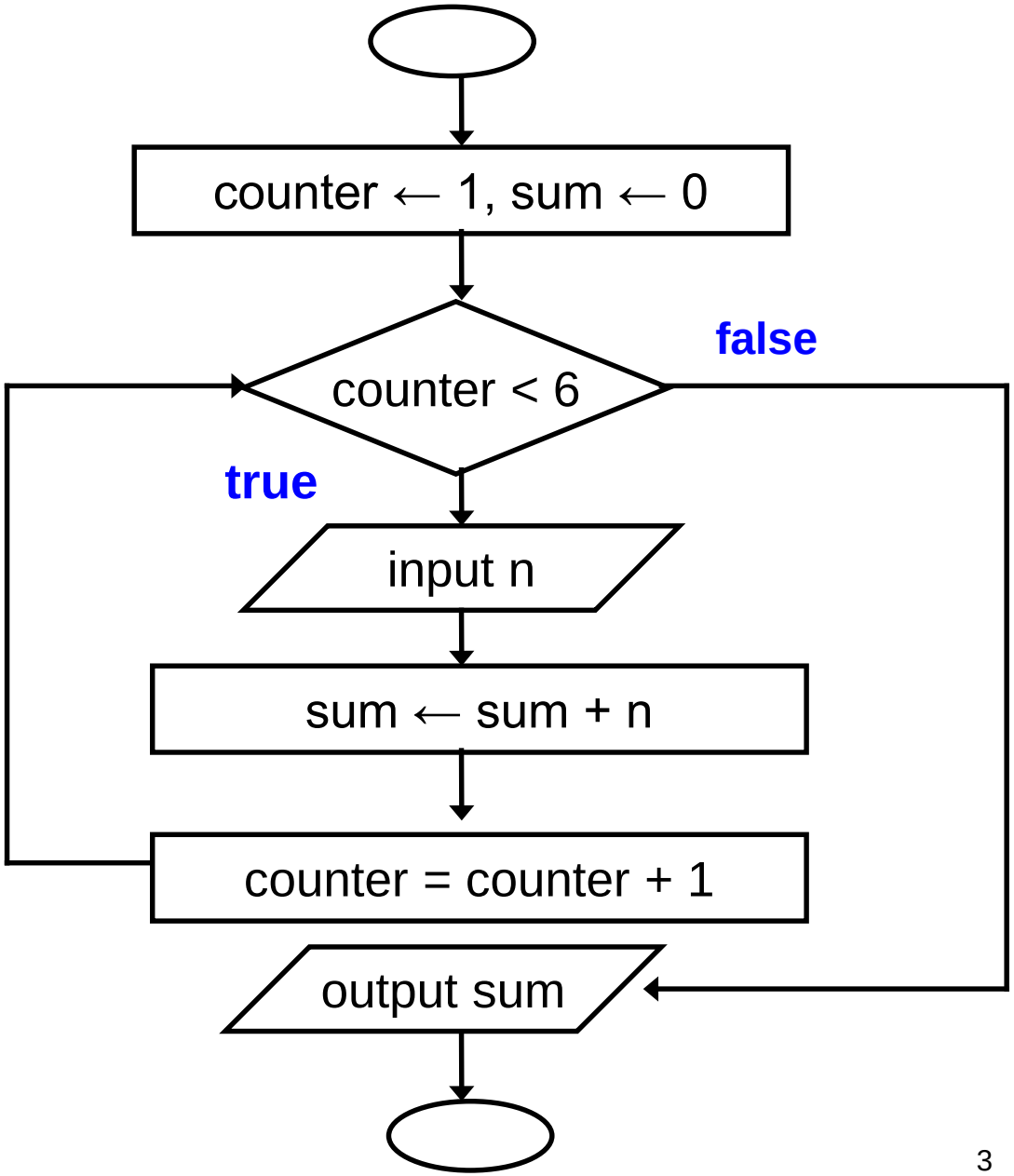
Helper slides on for-while loop

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

Example

Read 5 integers and display the their sum



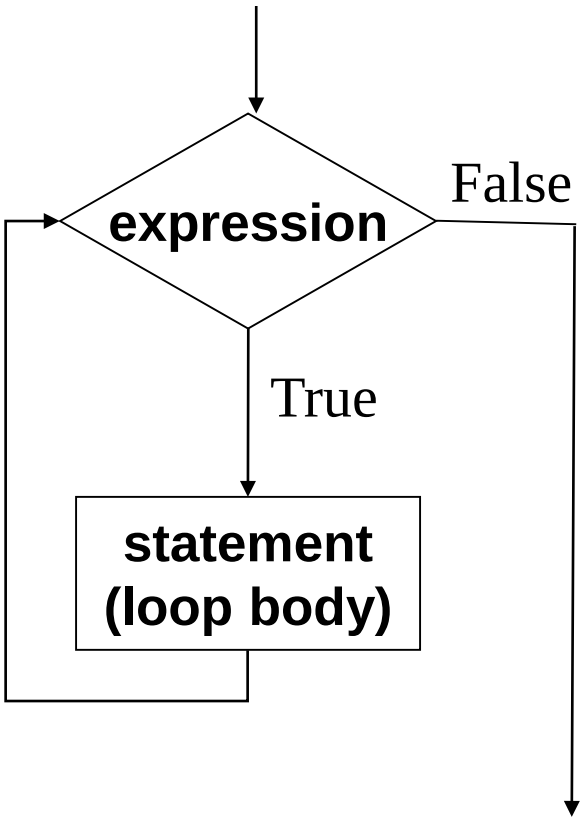
Looping: while statement

```
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.

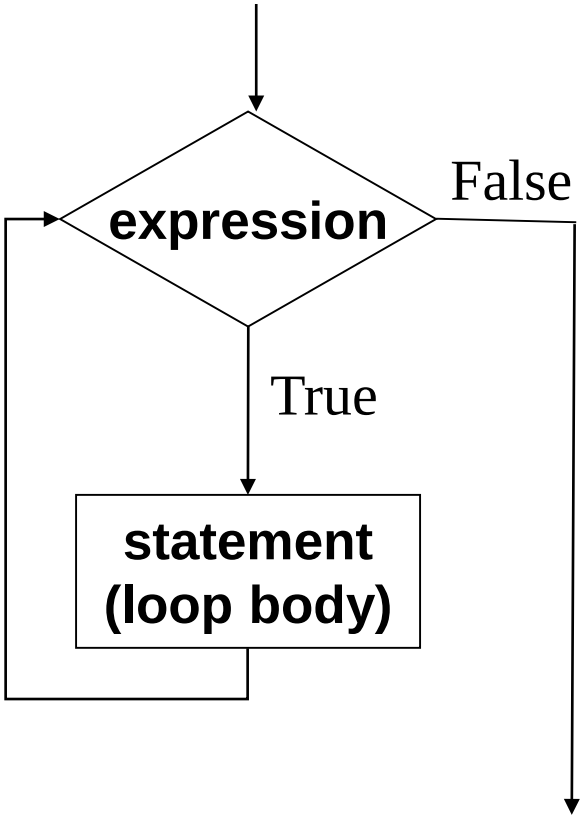
Looping: while statement

```
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 (true), the statement is executed. Then the expression is evaluated again and the same thing repeats. The loop terminates when the expression evaluates to 0 (false).

Example

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

Example

```
int  weight;
scanf(“%d”, &weight);
while ( weight > 65 ) {
    printf ("Go, exercise, ");
    printf ("then come back. \n");
    printf ("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;
}
```

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;
}
```

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;
}
```

```
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

gcd.c

Double your money

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

```
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;
}
```

Maximum of positive Numbers

```
int main() {
    double max = 0.0, next;
    printf (“Enter positive numbers, 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;
}
```

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 of the number is %d \n”, sum);
    return 0;
}
```

digit-sum.c

Looping: for Statement

- Most commonly used looping structure in C

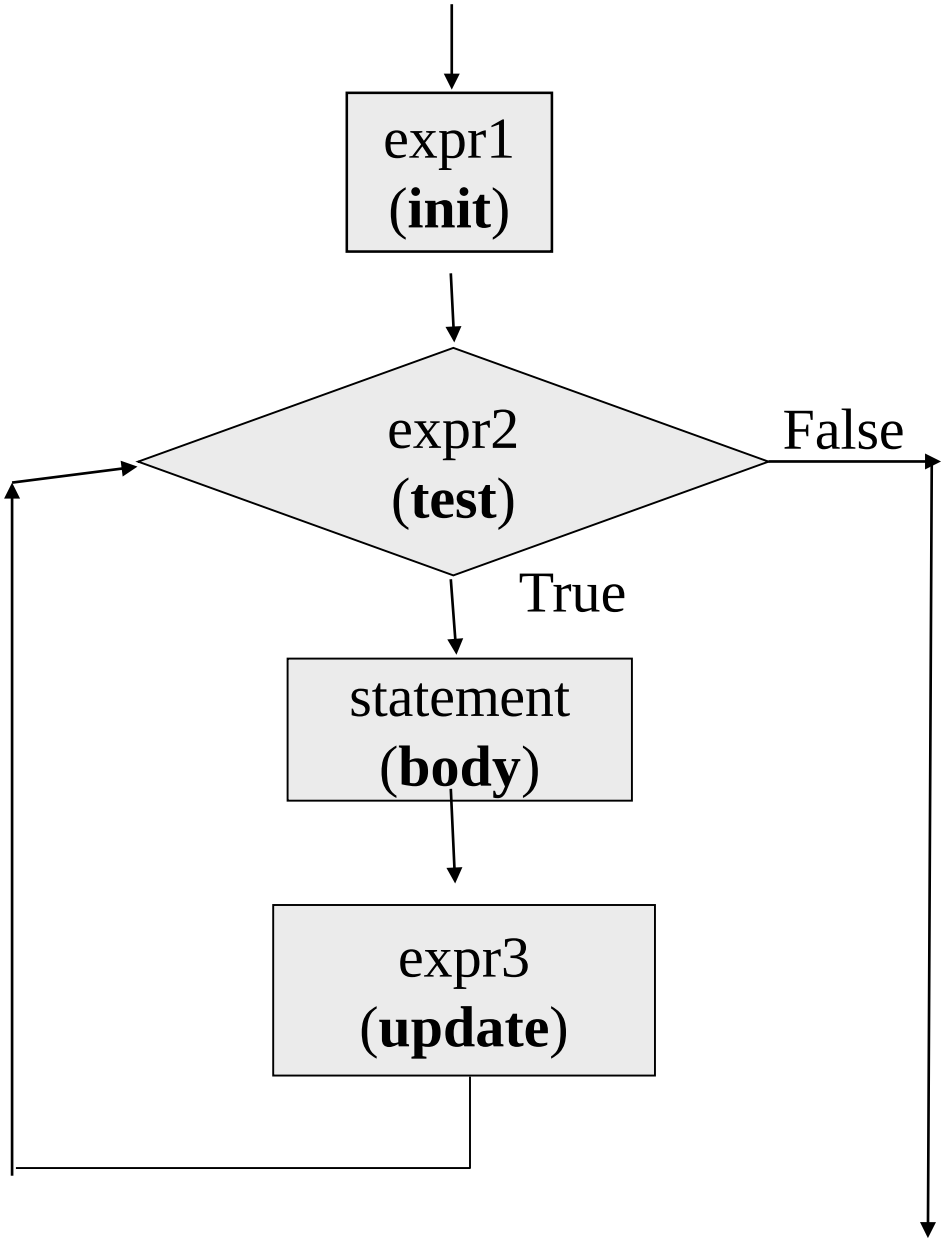
```
for ( expr1; expr2; expr3)
    statement;

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


expr1
(**init**)
expr2
(**test**)
False
True
statement
(**body**)
expr3
(**update**)

Example: Computing 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;
}
```

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;
}
```

eseries-1.c

Equivalence of for and while

Whatever you can do with a for loop, you can also do with a while loop, and vice-versa

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

Same as

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

Sum of first N Natural Numbers (with both for and while loops)

```
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;
}
```

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
  - Program proceeds with statement after for structure

Some common errors to watch for

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

The ; ends the loop, so the statement is now out of the loop, not what you wanted.

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

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

This will never terminate. Why?

Nested Loops: Printing a 2-D Figure

- How would you print the following pattern?

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

repeat 3 times
    print a row of 5 *'s

repeat 5 times
    print *

Nested Loops

```
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;
}
```

outer loop

inner loop

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");
}
```

2d-figure.c

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");
}
```

Some simple problems to practice

- Read in a positive integer n. Then read in n positive integers and print the difference between the largest and smallest numbers
- Read in a positive integer n. Then read in n integers and find the sum of their absolute values
- Read in a positive integer n. Then read in n floating point numbers and print their average
- Read in a positive integer n. Then read n integers and print the number of integers less than 0 and number of integers >= 0 (keep two counts)
- Read in two positive integers n and m and print all common factors of n and m
- Read in an integer n. Then find and print the sum of the odd digits in n.
- Do all of the above with both for and while loops.