CS10003: Programming & Data Structures

Dept. of Computer Science & Engineering
Indian Institute of Technology Kharagpur

Autumn 2020

Iterations and Loops – contd.

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

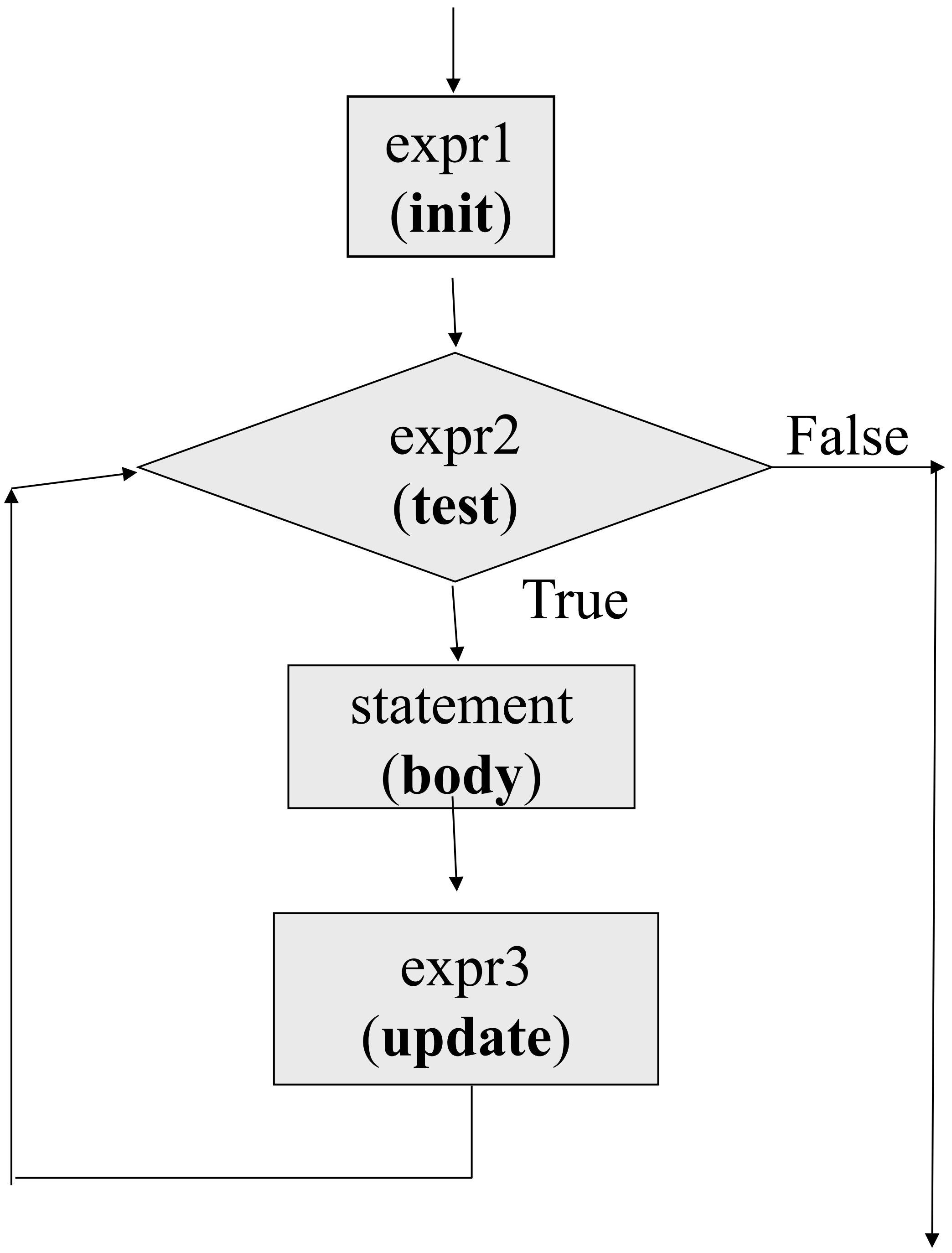
expr2 (test): test condition, loop continues if expression is non-0

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

statement (body): body of loop

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

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



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;
}
```

Computing e^x series up to 4 decimal places

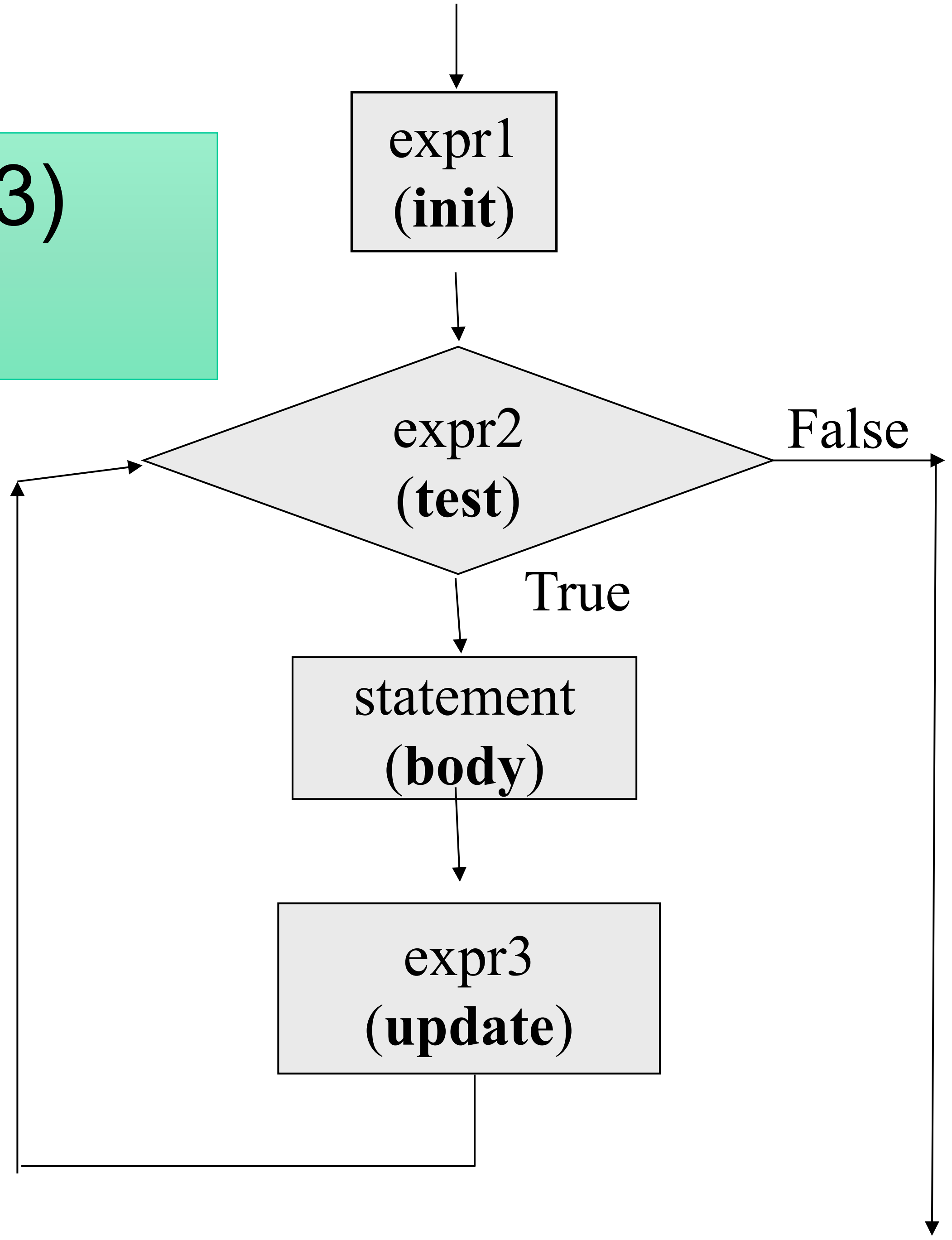
```
int main () {
     float x, term, sum;
     int cnt;
     scanf (“%f”, &x) ;
     term = 1.0; sum = 0;
     for (cnt = 1; term >= 0.0001; ++cnt)  {
          sum += term;
          term *= x/cnt;
      }
      printf (“%f\n”, sum) ;
      return 0;
}
```

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;
}
```

Advanced expression in *for* structure

Arithmetic expressions

Initialization, loop-continuation, and increment can contain arithmetic expressions.

e.g. Let `x = 2` and `y = 10`

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

is equivalent to

```
for ( j = 2; j <= 80; j += 5 )
```

"Increment" may be negative (decrement)

If loop continuation condition initially false

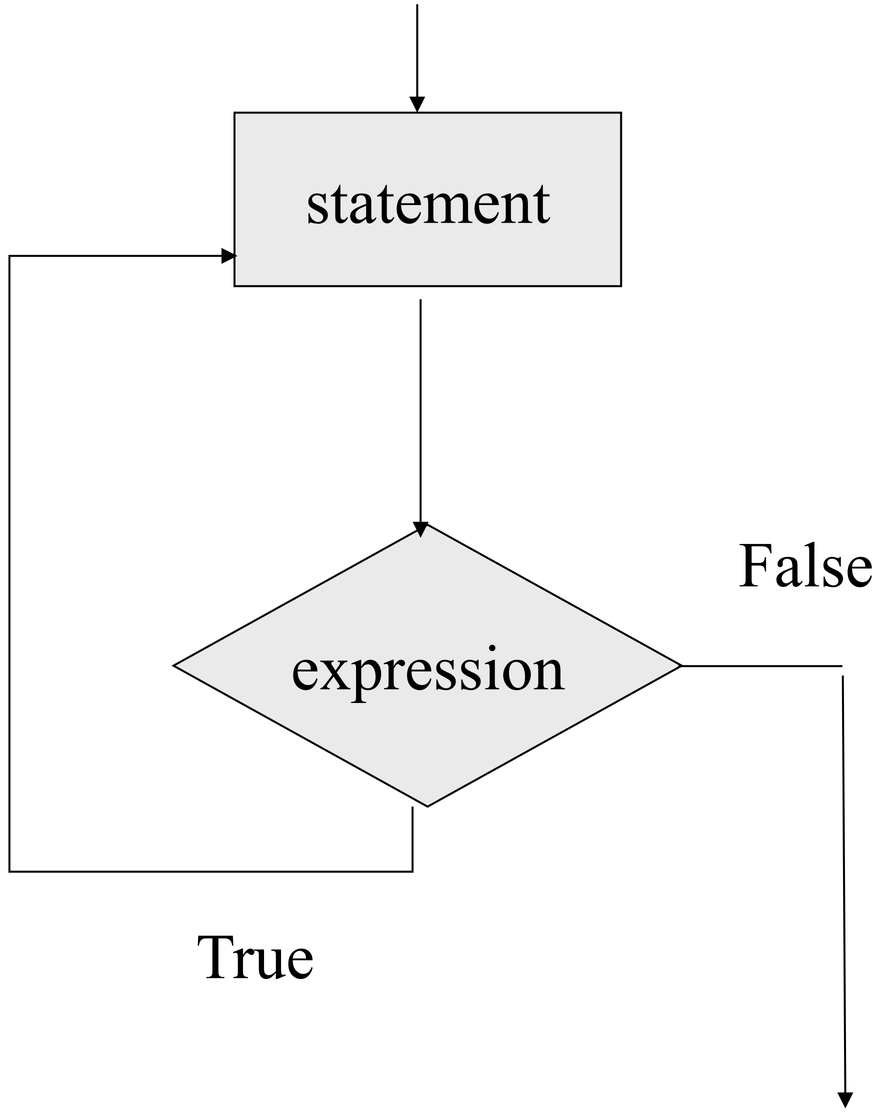
Body of `for` structure not performed

Control proceeds with statement after `for` structure

Looping: do-while statement

```
do
    statement;
while (expression);

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



Example

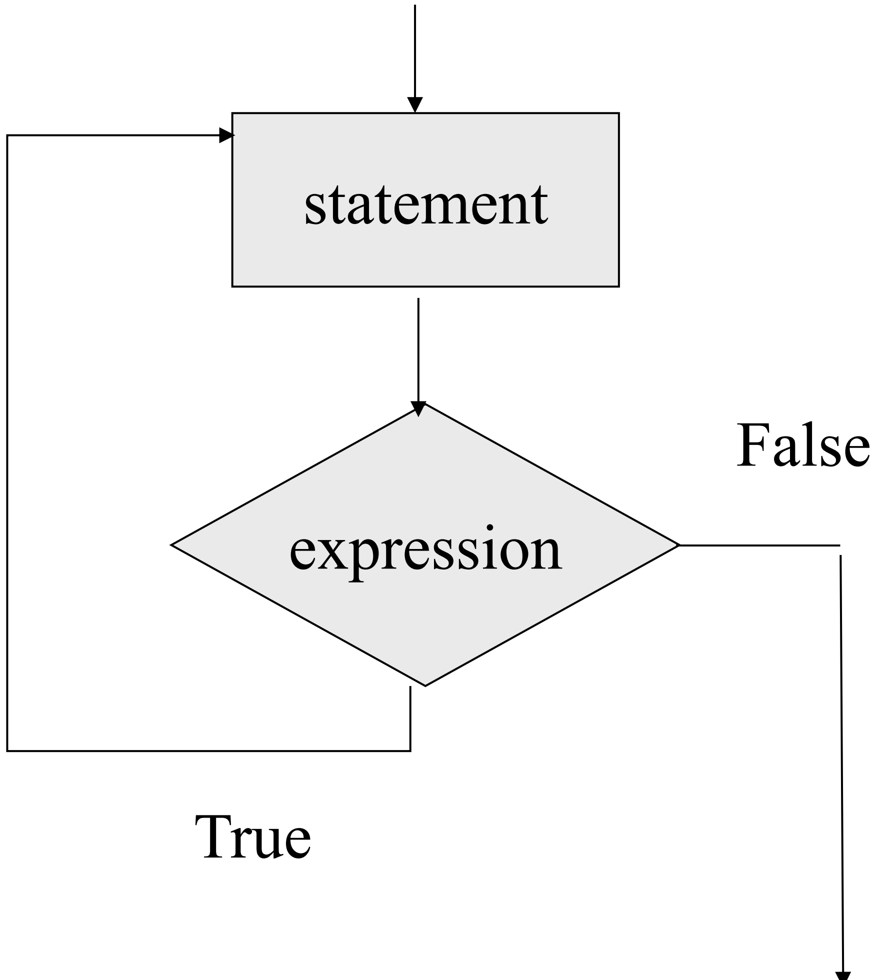
Problem: Prompt user to input “month” value, keep prompting until a correct value of month is given as input

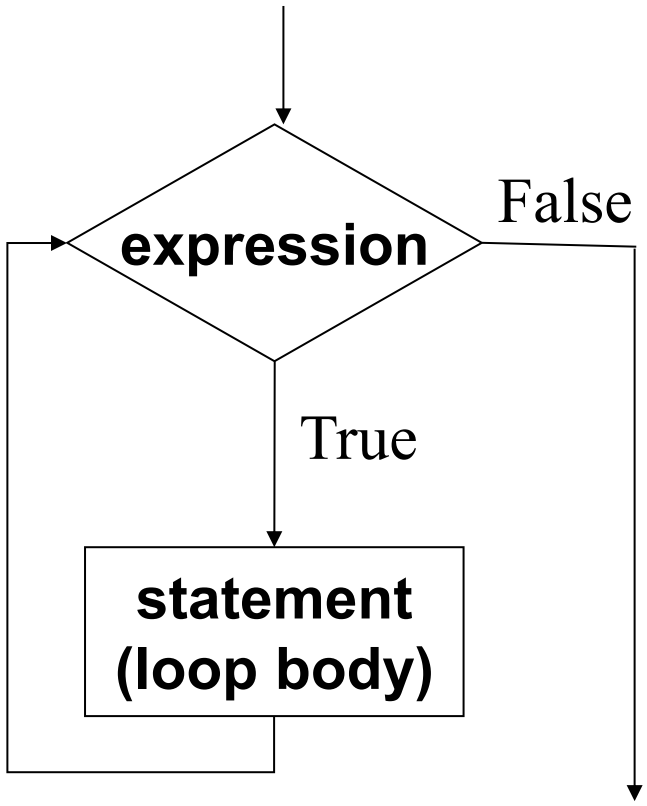
```
do {
     printf (“Please input month {1-12}”);
     scanf (“%d”, &month);
} while ((month < 1) || (month > 12));
```

Comparison between do-while and while

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

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





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;
}
```

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;
}
```

Sentinel-Controlled Loop

Receive 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: A set of integers ending with a negative integer or a zero

Output: Summation and Average of these integers

Input Example:

30 16 42 -9

Sentinel Value

Output Example:

Sum = 88

Average = 29.33

Specifying “Infinite Loop”

```
count=1;
while(1) {
    printf(“Count=%d”,count);
    count++;
}
```

```
count=1;
do {
    printf(“Count=%d”,count);
    count++;
} while(1);
```

```
count=1;
for(;;) {
    printf(“Count=%d”,count);
    count++;
}
```

```
for(count=1;;count++) {
    printf(“Count=%d”,count);
}
```

Specifying "Infinite Loop"

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

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

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

break Statement

Break out of the loop { }

- can use with
  - *while, do while, for, switch*
- does not work with
  - *if {}*
  - *else {}*

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

Program execution continues with the first statement after the structure

Common uses of the break statement

- Escape early from a loop
- Skip the remainder of a switch structure

Break from “Infinite Loop”

```
count=1;
while(1) {
    printf(“Count=%d”,count);
    count++;
    if(count>100)
        break;
}
```

```
count=1;
do {
    printf(“Count=%d”,count);
    count++;
    if(count>100)
        break;
} while(1);
```

```
count=1;
for(;;) {
    printf(“Count=%d”,count);
    count++;
    if(count>100)
        break;
}
```

```
for(count=1;;count++) {
    printf(“Count=%d”,count);
    if(count>100)
        break;
}
```

Thank You!