Imperative programming - tutorial week 2

Arnold Meijster

Dept. computer science (university of Groningen)

September 7, 2017

3.3.1 Numerical types

Given are the following declarations:

```
short a=350;
int b=3, c=10;
long d=1;
float x=391;
double y=0.71, z=0.35;
```

3.3.1 Numerical types

Given are the following declarations:

```
short a=350;
int b=3, c=10;
long d=1;
float x=391;
double y=0.71, z=0.35;
```

1. `d = a * b;`

3.3.1 Numerical types

Given are the following declarations:

```
short a=350;
int b=3, c=10;
long d=1;
float x=391;
double y=0.71, z=0.35;
```

1. `d = a * b;`
   short * int $\rightarrow$ **int,** fits in **long**

3.3.1 Numerical types

Given are the following declarations:

```
short a=350;
int b=3, c=10;
long d=1;
float x=391;
double y=0.71, z=0.35;
```

1. `d = a * b;`
   short * int $\rightarrow$ **int,** fits in **long**
   Result: d $=$ 1050

3.3.1 Numerical types

Given are the following declarations:

```
short a=350;
int b=3, c=10;
long d=1;
float x=391;
double y=0.71, z=0.35;
```

1. `d = a * b;`
   short * int $\rightarrow$ **int,** fits in **long**
   Result: d $=$ 1050
2. `c = a / b;`

3.3.1 Numerical types

Given are the following declarations:

```
short a=350;
int b=3, c=10;
long d=1;
float x=391;
double y=0.71, z=0.35;
```

1. `d = a * b;`
   short * int $\rightarrow$ **int,** fits in **long**
   Result: d $=$ 1050
2. `c = a / b;`
   short / int $\rightarrow$ **int,** fits in **int**

3.3.1 Numerical types

Given are the following declarations:

```
short a=350;
int b=3, c=10;
long d=1;
float x=391;
double y=0.71, z=0.35;
```

1. `d = a * b;`
   short * int $\rightarrow$ **int,** fits in **long**
   Result: d $=$ 1050
2. `c = a / b;`
   short / int $\rightarrow$ **int,** fits in **int**
   Result: c $=$ 116 (integer division)

3.3.1 Numerical types

Given are the following declarations:

```
short a=350;
int b=3, c=10;
long d=1;
float x=391;
double y=0.71, z=0.35;
```

1. `d = a * b;`
   short * int $\rightarrow$ **int,** fits in **long**
   Result: d $=$ 1050
2. `c = a / b;`
   short / int $\rightarrow$ **int,** fits in **int**
   Result: c $=$ 116 (integer division)
3. `y = c / b;`

3.3.1 Numerical types

Given are the following declarations:

```
short a=350;
int b=3, c=10;
long d=1;
float x=391;
double y=0.71, z=0.35;
```

1. `d = a * b;`
   short * int $\rightarrow$ **int,** fits in **long**
   Result: d $=$ 1050
2. `c = a / b;`
   short / int $\rightarrow$ **int,** fits in **int**
   Result: c $=$ 116 (integer division)
3. `y = c / b;`
   int / int $\rightarrow$ **int,** fits in **double**

3.3.1 Numerical types

Given are the following declarations:

```
short a=350;
int b=3, c=10;
long d=1;
float x=391;
double y=0.71, z=0.35;
```

1. `d = a * b;`
   short * int $\rightarrow$ **int,** fits in **long**
   Result: d $=$ 1050
2. `c = a / b;`
   short / int $\rightarrow$ **int,** fits in **int**
   Result: c $=$ 116 (integer division)
3. `y = c / b;`
   int / int $\rightarrow$ **int,** fits in **double**
   Result: y $=$ 3.0

3.3.1 Numerical types

Given are the following declarations:

```
short a=350;
int b=3, c=10;
long d=1;
float x=391;
double y=0.71, z=0.35;
```

1. `d = a * b;`
   short * int $\rightarrow$ **int,** fits in **long**
   Result: d $=$ 1050
2. `c = a / b;`
   short / int $\rightarrow$ **int,** fits in **int**
   Result: c $=$ 116 (integer division)
3. `y = c / b;`
   int / int $\rightarrow$ **int,** fits in **double**
   Result: y $=$ 3.0
4. `y = b + x;`

3.3.1 Numerical types

Given are the following declarations:

```
short a=350;
int b=3, c=10;
long d=1;
float x=391;
double y=0.71, z=0.35;
```

1. `d = a * b;`
   short * int $\rightarrow$ **int,** fits in **long**
   Result: d $=$ 1050
2. `c = a / b;`
   short / int $\rightarrow$ **int,** fits in **int**
   Result: c $=$ 116 (integer division)
3. `y = c / b;`
   int / int $\rightarrow$ **int,** fits in **double**
   Result: y $=$ 3.0
4. `y = b + x;`
   int + float $\rightarrow$ **float,** fits in **double**

3.3.1 Numerical types

Given are the following declarations:

```
short a=350;
int b=3, c=10;
long d=1;
float x=391;
double y=0.71, z=0.35;
```

1. `d = a * b;`
   short * int $\rightarrow$ **int,** fits in **long**
   Result: d $=$ 1050
2. `c = a / b;`
   short / int $\rightarrow$ **int,** fits in **int**
   Result: c $=$ 116 (integer division)
3. `y = c / b;`
   int / int $\rightarrow$ **int,** fits in **double**
   Result: y $=$ 3.0
4. `y = b + x;`
   int + float $\rightarrow$ **float,** fits in **double**
   Result: y $=$ 394.0

3.3.1 Numerical types (Continued)

Given are the following declarations:

```
short a=350;
int b=3, c=10;
long d=1;
float x=391;
double y=0.71, z=0.35;
```

3.3.1 Numerical types (Continued)

Given are the following declarations:

```
short a=350;
int b=3, c=10;
long d=1;
float x=391;
double y=0.71, z=0.35;
```

5. `a = a + 1;`

3.3.1 Numerical types (Continued)

Given are the following declarations:

```
short a=350;
int b=3, c=10;
long d=1;
float x=391;
double y=0.71, z=0.35;
```

5. `a = a + 1;`
   short + int → int, does not fit in **short**

3.3.1 Numerical types (Continued)

Given are the following declarations:

```
short a=350;
int b=3, c=10;
long d=1;
float x=391;
double y=0.71, z=0.35;
```

5. `a = a + 1;`
   short + int → int, does not fit in **short**
   Solution: `a = (short)(a + 1);`
   Note: In practice, few programmers will cast this, although in some language (like Java) you have to.
   Result: a = 351

3.3.1 Numerical types (Continued)

Given are the following declarations:

```
short a=350;
int b=3, c=10;
long d=1;
float x=391;
double y=0.71, z=0.35;
```

5. `a = a + 1;`
   short + int → int, does not fit in **short**
   Solution: `a = (short)(a + 1);`
   Note: In practice, few programmers will cast this, although in some language (like Java) you have to.
   Result: a = 351
6. `a++;`

3.3.1 Numerical types (Continued)

Given are the following declarations:

```
short a=350;
int b=3, c=10;
long d=1;
float x=391;
double y=0.71, z=0.35;
```

5. `a = a + 1;`
   short + int → int, does not fit in **short**
   Solution: `a = (short)(a + 1);`
   Note: In practice, few programmers will cast this, although in some language (like Java) you have to.
   Result: a = 351
6. `a++;`
   The '++' operation on a short yields a short. So, this is ok.

3.3.1 Numerical types (Continued)

Given are the following declarations:

```
short a=350;
int b=3, c=10;
long d=1;
float x=391;
double y=0.71, z=0.35;
```

5. `a = a + 1;`
   short + int $\rightarrow$ **int,** does not fit in **short**
   Solution: `a = (short)(a + 1);`
   Note: In practice, few programmers will cast this, although in some language (like Java) you have to.
   Result: a = 351
6. `a++;`
   The '++' operation on a short yields a short. So, this is ok.
   Result: a = 351;

3.3.1 Numerical types (Continued)

Given are the following declarations:

```
short a=350;
int b=3, c=10;
long d=1;
float x=391;
double y=0.71, z=0.35;
```

5. `a = a + 1;`
   short + int $\rightarrow$ **int,** does not fit in **short**
   Solution: `a = (short)(a + 1);`
   Note: In practice, few programmers will cast this, although in some language (like Java) you have to.
   Result: a = 351
6. `a++;`
   The '++' operation on a short yields a short. So, this is ok.
   Result: a = 351;
7. `b = a + 1`

3.3.1 Numerical types (Continued)

Given are the following declarations:

```
short a=350;
int b=3, c=10;
long d=1;
float x=391;
double y=0.71, z=0.35;
```

5. `a = a + 1;`
   short + int → int, does not fit in **short**
   Solution: `a = (short)(a + 1);`
   Note: In practice, few programmers will cast this, although in some language (like Java) you have to.
   Result: a = 351
6. `a++;`
   The '++' operation on a short yields a short. So, this is ok.
   Result: a = 351;
7. `b = a + 1`
   short + int → int, fits in **int**

3.3.1 Numerical types (Continued)

Given are the following declarations:

```
short a=350;
int b=3, c=10;
long d=1;
float x=391;
double y=0.71, z=0.35;
```

5. `a = a + 1;`
   short + int $\rightarrow$ **int,** does not fit in **short**
   Solution: `a = (short)(a + 1);`
   Note: In practice, few programmers will cast this, although in some language (like Java) you have to.
   Result: a = 351
6. `a++;`
   The '++' operation on a short yields a short. So, this is ok.
   Result: a = 351;
7. `b = a + 1`
   short + int $\rightarrow$ **int,** fits in **int**
   Result: b = 351

3.3.1 Numerical types (Continued)

Given are the following declarations:

```
short a=350;
int b=3, c=10;
long d=1;
float x=391;
double y=0.71, z=0.35;
```

8. `d = 100 * (x - y);`

3.3.1 Numerical types (Continued)

Given are the following declarations:

```
short a=350;
int b=3, c=10;
long d=1;
float x=391;
double y=0.71, z=0.35;
```

8. `d = 100 * (x - y);`
   int * (float - double) $\rightarrow$ **int * double** $\rightarrow$ **double,** does not fit in **long**

3.3.1 Numerical types (Continued)

Given are the following declarations:

```
short a=350;
int b=3, c=10;
long d=1;
float x=391;
double y=0.71, z=0.35;
```

8. `d = 100 * (x - y);`
   int * (float - double) $\rightarrow$ **int * double** $\rightarrow$ **double,** does not fit in **long**
   Solution 1: `d = (long)(100 * (x-y));` Result: d = 39029
   Solution 2: `d = 100 * (long)(x-y);` Result: d = 39000

3.3.1 Numerical types (Continued)

Given are the following declarations:

```
short a=350;
int b=3, c=10;
long d=1;
float x=391;
double y=0.71, z=0.35;
```

8. `d = 100 * (x - y);`
   int * (float - double) $\rightarrow$ **int * double** $\rightarrow$ **double,** does not fit in **long**
   Solution 1: `d = (long)(100 * (x-y));` Result: d = 39029
   Solution 2: `d = 100 * (long)(x-y);` Result: d = 39000
9. `x = (float)(a / b);`

3.3.1 Numerical types (Continued)

Given are the following declarations:

```
short a=350;
int b=3, c=10;
long d=1;
float x=391;
double y=0.71, z=0.35;
```

8. `d = 100 * (x - y);`
   int * (float - double) $\rightarrow$ **int * double** $\rightarrow$ **double,** does not fit in **long**
   Solution 1: `d = (long)(100 * (x-y));` Result: d = 39029
   Solution 2: `d = 100 * (long)(x-y);` Result: d = 39000
9. `x = (float)(a / b);`
   (float)(short / int) $\rightarrow$ **(float)(int)** $\rightarrow$ **float,** fits in **float**

3.3.1 Numerical types (Continued)

Given are the following declarations:

```
short a=350;
int b=3, c=10;
long d=1;
float x=391;
double y=0.71, z=0.35;
```

8. `d = 100 * (x - y);`
   int * (float - double) $\rightarrow$ **int * double** $\rightarrow$ **double,** does not fit in **long**
   Solution 1: `d = (long)(100 * (x-y));` Result: d = 39029
   Solution 2: `d = 100 * (long)(x-y);` Result: d = 39000
9. `x = (float)(a / b);`
   (float)(short / int) $\rightarrow$ **(float)(int)** $\rightarrow$ **float,** fits in **float**
   Result: x = 116.0

3.3.1 Numerical types (Continued)

Given are the following declarations:

```
short a=350;
int b=3, c=10;
long d=1;
float x=391;
double y=0.71, z=0.35;
```

8. `d = 100 * (x - y);`
   int * (float - double) $\rightarrow$ **int * double** $\rightarrow$ **double,** does not fit in **long**
   Solution 1: `d = (long)(100 * (x-y));` Result: d = 39029
   Solution 2: `d = 100 * (long)(x-y);` Result: d = 39000
9. `x = (float)(a / b);`
   (float)(short / int) $\rightarrow$ **(float)(int)** $\rightarrow$ **float,** fits in **float**
   Result: x = 116.0
10. `z = (float)a / c;`

3.3.1 Numerical types (Continued)

Given are the following declarations:

```
short a=350;
int b=3, c=10;
long d=1;
float x=391;
double y=0.71, z=0.35;
```

8. `d = 100 * (x - y);`
   int * (float - double) $\rightarrow$ **int * double** $\rightarrow$ **double,** does not fit in **long**
   Solution 1: `d = (long)(100 * (x-y));` Result: d = 39029
   Solution 2: `d = 100 * (long)(x-y);` Result: d = 39000
9. `x = (float)(a / b);`
   (float)(short / int) $\rightarrow$ **(float)(int)** $\rightarrow$ **float,** fits in **float**
   Result: x = 116.0
10. `z = (float)a / c;`
    (float)short / int $\rightarrow$ **float / int** $\rightarrow$ **float,** fits in **double**

3.3.1 Numerical types (Continued)

Given are the following declarations:

```
short a=350;
int b=3, c=10;
long d=1;
float x=391;
double y=0.71, z=0.35;
```

8. `d = 100 * (x - y);`
   int * (float - double) $\rightarrow$ **int * double** $\rightarrow$ **double,** does not fit in **long**
   Solution 1: `d = (long)(100 * (x-y));` Result: d = 39029
   Solution 2: `d = 100 * (long)(x-y);` Result: d = 39000
9. `x = (float)(a / b);`
   (float)(short / int) $\rightarrow$ **(float)(int)** $\rightarrow$ **float,** fits in **float**
   Result: x = 116.0
10. `z = (float)a / c;`
    (float)short / int $\rightarrow$ **float / int** $\rightarrow$ **float,** fits in **double**
    Result: z = 35.0;

3.3.1 Numerical types (Continued)

Given are the following declarations:

```
short a=350;
int b=3, c=10;
long d=1;
float x=391;
double y=0.71, z=0.35;
```

8. `d = 100 * (x - y);`
   int * (float - double) $\rightarrow$ **int * double** $\rightarrow$ **double,** does not fit in **long**
   Solution 1: `d = (long)(100 * (x-y));` Result: d = 39029
   Solution 2: `d = 100 * (long)(x-y);` Result: d = 39000
9. `x = (float)(a / b);`
   (float)(short / int) $\rightarrow$ **(float)(int)** $\rightarrow$ **float,** fits in **float**
   Result: x = 116.0
10. `z = (float)a / c;`
    (float)short / int $\rightarrow$ **float / int** $\rightarrow$ **float,** fits in **double**
    Result: z = 35.0;
11. `a = (int)y + x;`

3.3.1 Numerical types (Continued)

Given are the following declarations:

```
short a=350;
int b=3, c=10;
long d=1;
float x=391;
double y=0.71, z=0.35;
```

8. `d = 100 * (x - y);`
   int * (float - double) → **int * double** → **double,** does not fit in **long**
   Solution 1: `d = (long)(100 * (x-y));` Result: d = 39029
   Solution 2: `d = 100 * (long)(x-y);` Result: d = 39000
9. `x = (float)(a / b);`
   (float)(short / int) → **(float)(int)** → **float,** fits in **float**
   Result: x = 116.0
10. `z = (float)a / c;`
    (float)short / int → **float / int** → **float,** fits in **double**
    Result: z = 35.0;
11. `a = (int)y + x;`
    (int)double + float → **int + float** → **float,** does not fit in **short**

3.3.1 Numerical types (Continued)

Given are the following declarations:

```
short a=350;
int b=3, c=10;
long d=1;
float x=391;
double y=0.71, z=0.35;
```

8. `d = 100 * (x - y);`
   int * (float - double) $\rightarrow$ **int * double** $\rightarrow$ **double,** does not fit in **long**
   Solution 1: `d = (long)(100 * (x-y));` Result: d = 39029
   Solution 2: `d = 100 * (long)(x-y);` Result: d = 39000
9. `x = (float)(a / b);`
   (float)(short / int) $\rightarrow$ **(float)(int)** $\rightarrow$ **float,** fits in **float**
   Result: x = 116.0
10. `z = (float)a / c;`
    (float)short / int $\rightarrow$ **float / int** $\rightarrow$ **float,** fits in **double**
    Result: z = 35.0;
11. `a = (int)y + x;`
    (int)double + float $\rightarrow$ **int + float** $\rightarrow$ **float,** does not fit in **short**
    Solution: `a = (short)(y + x);` Result: a = 391

3.3.1 Numerical types (Continued)

Given are the following declarations:

```
short a=350;
int b=3, c=10;
long d=1;
float x=391;
double y=0.71, z=0.35;
```

12. `a = (int)(y + x);`

3.3.1 Numerical types (Continued)

Given are the following declarations:

```
short a=350;
int b=3, c=10;
long d=1;
float x=391;
double y=0.71, z=0.35;
```

12. `a = (int)(y + x);`
    (int)(double + float) $\rightarrow$ **(int)double** $\rightarrow$ **int,** does not fit in **short**

3.3.1 Numerical types (Continued)

Given are the following declarations:

```
short a=350;
int b=3, c=10;
long d=1;
float x=391;
double y=0.71, z=0.35;
```

12. `a = (int)(y + x);`
    (int)(double + float) $\rightarrow$ **(int)double** $\rightarrow$ **int,** does not fit in **short**
    Solution: `a = (short)(y + x);`
    Result: a = 391

3.3.1 Numerical types (Continued)

Given are the following declarations:

```
short a=350;
int b=3, c=10;
long d=1;
float x=391;
double y=0.71, z=0.35;
```

12. `a = (int)(y + x);`
    (int)(double + float) $\rightarrow$ **(int)double** $\rightarrow$ **int,** does not fit in **short**
    Solution: `a = (short)(y + x);`
    Result: a = 391
13. `a = (short)(y + x);`

3.3.1 Numerical types (Continued)

Given are the following declarations:

```
short a=350;
int b=3, c=10;
long d=1;
float x=391;
double y=0.71, z=0.35;
```

12. `a = (int)(y + x);`
    (int)(double + float) $\rightarrow$ **(int)double** $\rightarrow$ **int,** does not fit in **short**
    Solution: `a = (short)(y + x);`
    Result: a = 391
13. `a = (short)(y + x);`
    (short)(double + float) $\rightarrow$ **(short)double** $\rightarrow$ **short,** fits in **short**

3.3.1 Numerical types (Continued)

Given are the following declarations:

```
short a=350;
int b=3, c=10;
long d=1;
float x=391;
double y=0.71, z=0.35;
```

12. `a = (int)(y + x);`
    (int)(double + float) $\rightarrow$ **(int)double** $\rightarrow$ **int,** does not fit in **short**
    Solution: `a = (short)(y + x);`
    Result: a = 391
13. `a = (short)(y + x);`
    (short)(double + float) $\rightarrow$ **(short)double** $\rightarrow$ **short,** fits in **short**
    Result: a = 391

3.3.1 Numerical types (Continued)

Given are the following declarations:

```
short a=350;
int b=3, c=10;
long d=1;
float x=391;
double y=0.71, z=0.35;
```

12. `a = (int)(y + x);`
    (int)(double + float) → **(int)double** → **int,** does not fit in **short**
    Solution: `a = (short)(y + x);`
    Result: a = 391
13. `a = (short)(y + x);`
    (short)(double + float) → **(short)double** → **short,** fits in **short**
    Result: a = 391
14. `c = (int)a / z;`

3.3.1 Numerical types (Continued)

Given are the following declarations:

```
short a=350;
int b=3, c=10;
long d=1;
float x=391;
double y=0.71, z=0.35;
```

12. `a = (int)(y + x);`
    (int)(double + float) → **(int)double** → **int,** does not fit in **short**
    Solution: `a = (short)(y + x);`
    Result: a = 391
13. `a = (short)(y + x);`
    (short)(double + float) → **(short)double** → **short,** fits in **short**
    Result: a = 391
14. `c = (int)a / z;`
    (int)short / double → **int / double** → **double,** does not fit in **int**

3.3.1 Numerical types (Continued)

Given are the following declarations:

```
short a=350;
int b=3, c=10;
long d=1;
float x=391;
double y=0.71, z=0.35;
```

12. `a = (int)(y + x);`
    (int)(double + float) → **(int)double** → **int,** does not fit in **short**
    Solution: `a = (short)(y + x);`
    Result: a = 391
13. `a = (short)(y + x);`
    (short)(double + float) → **(short)double** → **short,** fits in **short**
    Result: a = 391
14. `c = (int)a / z;`
    (int)short / double → **int / double** → **double,** does not fit in **int**
    Solution: `c = (int)(a/z);`
    Result: c = 1000

3.3.1 Numerical types (Continued)

Given are the following declarations:

```
short a=350;
int b=3, c=10;
long d=1;
float x=391;
double y=0.71, z=0.35;
```

15. `c = a / (int)z;`

3.3.1 Numerical types (Continued)

Given are the following declarations:

```
short a=350;
int b=3, c=10;
long d=1;
float x=391;
double y=0.71, z=0.35;
```

15. `c = a / (int)z;`
    short / (int)double $\rightarrow$ **short / int** $\rightarrow$ **int,** fits in **int**

3.3.1 Numerical types (Continued)

Given are the following declarations:

```
short a=350;
int b=3, c=10;
long d=1;
float x=391;
double y=0.71, z=0.35;
```

15. `c = a / (int)z;`
    short / (int)double $\rightarrow$ **short / int** $\rightarrow$ **int,** fits in **int**
    Crashes with "Floating point exception". We divide by zero, since `(int)0.35 == 0`.

3.3.1 Numerical types (Continued)

Given are the following declarations:

```
short a=350;
int b=3, c=10;
long d=1;
float x=391;
double y=0.71, z=0.35;
```

15. `c = a / (int)z;`
    short / (int)double $\rightarrow$ **short / int** $\rightarrow$ **int,** fits in **int**
    Crashes with "Floating point exception". We divide by zero, since `(int)0.35 == 0`.
    Solution: `c = (int)(a / z);`
    Result: c = 1000

3.3.1 Numerical types (Continued)

Given are the following declarations:

```
short a=350;
int b=3, c=10;
long d=1;
float x=391;
double y=0.71, z=0.35;
```

15. `c = a / (int)z;`
    short / (int)double $\rightarrow$ **short / int** $\rightarrow$ **int,** fits in **int**
    Crashes with "Floating point exception". We divide by zero, since `(int)0.35 == 0`.
    Solution: `c = (int)(a / z);`
    Result: c = 1000
16. `c = (int)(a / z);`

3.3.1 Numerical types (Continued)

Given are the following declarations:

```
short a=350;
int b=3, c=10;
long d=1;
float x=391;
double y=0.71, z=0.35;
```

15. `c = a / (int)z;`
    short / (int)double $\rightarrow$ **short / int** $\rightarrow$ **int,** fits in **int**
    Crashes with "Floating point exception". We divide by zero, since `(int)0.35 == 0`.
    Solution: `c = (int)(a / z);`
    Result: c = 1000

16. `c = (int)(a / z);`
    (int) (short / double) $\rightarrow$ **(int) double** $\rightarrow$ **int,** fits in **int**

3.3.1 Numerical types (Continued)

Given are the following declarations:

```
short a=350;
int b=3, c=10;
long d=1;
float x=391;
double y=0.71, z=0.35;
```

15. `c = a / (int)z;`
    short / (int)double $\rightarrow$ **short / int** $\rightarrow$ **int,** fits in **int**
    Crashes with "Floating point exception". We divide by zero, since `(int)0.35 == 0`.
    Solution: `c = (int)(a / z);`
    Result: c = 1000

16. `c = (int)(a / z);`
    (int) (short / double) $\rightarrow$ **(int) double** $\rightarrow$ **int,** fits in **int**
    Result: c = 1000;

3.3.2.1 Divisors

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

int main() {
  int number, divisor;
  printf("Please type a positive integer: ");
  scanf("%d", &number);

  printf("The divisors of %d are:", number);
  for (divisor = 1; divisor <= number/2; divisor++) {
    if (number % divisor == 0) {
      printf(" %d", divisor);
    }
  }
  /* each number divides itself */
  printf(" %d\n", number);
  return 0;
}
```

3.3.2.2 Exponentiation

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

int main() {
  int g, m, power;
  printf("g (>=0): ");
  scanf("%d", &g);
  printf("m (>=0): ");
  scanf("%d", &m);
  power = 1;
  while (m > 0) {
    /* loop invariant: power*exponentiation(g,m)==X */
    power = power*g;
    m--;
  }
  printf("exponentiation(g,m) = %d\n", power);
  return 0;
}
```

3.3.2.3 Integer logarithm

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

int main() {
  int g, x, power, exponent;
  printf("base g(>=2): ");
  scanf("%d", &g);
  printf("integer x(>=1): ");
  scanf("%d", &x);

  power = 1;
  exponent = 0;  /* invariant: power == exponentiation
      (g,m) */
  while (power*g <= x) {
    power = power*g;
    exponent++;
  }
  printf("intlog(%d) = %d\n", x, exponent);

  return 0;
}
```

3.3.2.4 The inverse of the factorial function

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

int main() {
  int n, count, fac;
  scanf("%d", &n);

  count = 1;
  fac = 1;
  while (fac*(count+1) <= n) {
    /* loop invariant: fac == count! */
    count++;
    fac = fac*count;
  }
  printf("caf(%d) == %d\n", n, count);

  return 0;
}
```