

CS-259

Data Structures with Java

Prime Numbers and Prime Factors



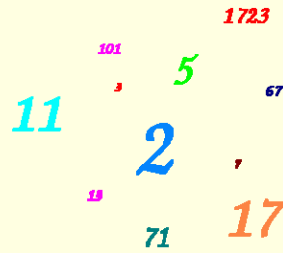
Instructor: **Joel Castellanos**

e-mail: joel@unm.edu

web: <http://cs.unm.edu/~joel/>

Course website:

<http://cs.unm.edu/~joel/cs259/>



8/29/2009

Prime Numbers

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

Can you spot a pattern?

2

Is n Prime? – Facts

- If n is not divisible by any number smaller than n , then n is prime.
- To prove n is not prime, only one factor needs to be found.
- If n is not prime, then there exists a pair of integers, a and b such that $n=a \times b$.
- If $n=a \times b$, then either a or b must be less than or equal to the square root of n .
- Therefore, starting from 2 and counting up, if no factors have been found from 2 through \sqrt{n} , then n must be prime.

3

Is n Prime? – The Algorithm

1. Given n , calculate the largest possible first factor:
 $\text{largestPossibleFirstPrime} = \sqrt{n}$
2. Let $\text{testDivisor} = 2$.
3. If testDivisor is greater than $\text{largestPossibleFirstPrime}$, then stop checking because n is prime.
4. If testDivisor is a factor of n , then n is not prime.
5. If testDivisor does not a factor of n , then increment testDivisor and return to step 3.

4

Is n Prime? By for Loop

```
public static void main(String[] args)
{
    int n = 64;
    int largestPossibleFirstPrime = (int)Math.sqrt(n);
    boolean foundDivisor = false;

    for (int k=2; k <= largestPossibleFirstPrime; k++)
    {
        System.out.println("k="+k);
        if (n % k == 0)
        {
            foundDivisor = true;
        }
    }
    if (foundDivisor)
    {
        System.out.println(
            n + " is NOT Prime.");
    }
    else System.out.println(n + " is Prime.");
}
}
```

```
k=2
k=3
k=4
k=5
k=6
k=7
k=8
64 is NOT Prime.
```

5

Is n Prime? By while Loop

```
public static void main(String[] args)
{
    int n = 64;
    int largestPossibleFirstPrime = (int)Math.sqrt(n);
    boolean foundDivisor = false;
    int k=2;
    while(k <= largestPossibleFirstPrime)
    {
        System.out.println("k="+k);
        if (n % k == 0)
        {
            foundDivisor = true;
            break;
        }
        k++;
    }
    if (foundDivisor)
    {
        System.out.println(n + " is NOT Prime.");
    }
    else System.out.println(n + " is Prime.");
}
}
```

k is created and initialized outside the loop, tested in the while statement and incremented inside the loop.

6

for Loop verses while Loop

```
public static void main(String[] args)
{ for (int i=0; i<12; i=i+3)
  { System.out.print("i=" + i + ", ");
  }
  System.out.println();

  int k=0;
  while (k<12)
  { System.out.print("k=" + k + ", ");
    k=k+3;
  }
}
```

for is more compact.
while is more general.

Output:
i=0, i=3, i=6, i=9,
k=0, k=3, k=6, k=9,

7

Prime Factors

9 → 3•3

100 → 10•10 → (5•2)•(5•2)

13 → 13

420 → 10•42 → (5•2)•(7•6) → 5•2•7•(2•3)

100 → 2•50 → 2•(2•25) → 2•2•(5•5)

The above is a non-systematic method.

8

Finding Prime Factors - Algorithm

1. Let n be the number of which to find the prime factors.
2. Start with 2 as a test divisor.
3. If the test divisor is greater than the square root of n , then the test divisor is either 1 or prime, and we are done.
4. If the remainder of n divided by the test divisor is zero, then:
 - a) The test divisor is a prime factor of n . Print it.
 - b) Replace n with n divided by the test divisor.
 - c) If the new n equals one, then all the divisors have been found.
5. If the remainder of n divided by the test divisor not zero, then:
 - a) Increment the test divisor.
6. Loop to step 3.

9

Prime Factors: Example $n=100$

- Start with $n=100$, and $\text{testDivisor} = 2$;
- 2 divides 100, so print 2 and set $n=100/2=50$.
- 2 divides 50, so print 2 and set $n=50/2=25$.
- 2 does not divide 25, so increment testDivisor to 3.
- 3 does not divide 25, so increment testDivisor to 4.
- 4 does not divide 25, so increment testDivisor to 5.
- 5 divides 25, so print 5, and set $n=25/5=5$.
- 5 divides 5, so print 5, and set $n=5/5=1$.
- Since n is equal to 1, stop.

10

Lab 2: Prime Factors

- Write a Java program that hard-codes a positive integer value `final int ORIGINAL_NUMBER`, and uses the given prime factor algorithm to correctly print that `ORIGINAL_NUMBER` is prime, or to print its prime factors.
- All repeated factors must be printed.
- The prime factors must be printed in order from smallest to largest.
- Your code must work for all positive integer values of `ORIGINAL_NUMBER` from 1 through 2 billion.
- Your output must include the value of `ORIGINAL_NUMBER` and must be clearly represented.

11

Lab 2: Grading Rubric

- Total points: 20
- Your code will be run with 12 test cases of unknown legal values of `ORIGINAL_NUMBER`. You will receive one point for each correct (that matches the specs) result .
- Proper adherence to the CS-259 coding standard, including comments, is worth 5 points. You start with those 5 points and each error levies a -1 up to the maximum of 5 points.
- 3 Points: your code is efficient (i.e. does not retest many values, ...), and your code is easy to read.
- -2 points for each compiler warning.
- Due Date: Midnight on Monday, Aug 31.

12