

Function Call Trace

```
public class Newton {  
    public static double sqrt(double c) {  
        double epsilon = 1e-15;  
        if (c < 0) return Double.NaN;  
        double t = c;  
        while (Math.abs(t - c/t) > epsilon * t)  
            t = (c/t + t) / 2.0;  
        return t;  
    }  
  
    public static void main(String[] args) {  
        double[] a = new double[args.length];  
        for (int i = 0; i < args.length; i++)  
            a[i] = Double.parseDouble(args[i]);  
        for (int i = 0; i < a.length; i++)  
            System.out.println(sqrt(a[i]));  
    }  
}
```

Function Call Trace

```
public class Newton {  
    public static double sqrt(double c) {  
        double epsilon = 1e-15;  
        if (c < 0) return Double.NaN;  
        double t = c;  
        while (Math.abs(t - c/t) > epsilon * t)  
            t = (c/t + t) / 2.0;  
        return t;  
    }  
    public static void main(String[] args) {  
        double[] a = new double[args.length];  
        for (int i = 0; i < args.length; i++)  
            a[i] = Double.parseDouble(args[i]);  
        for (int i = 0; i < a.length; i++)  
            System.out.println(sqrt(a[i]));  
    }  
}
```

```
% java Newton 1 2 3
```

Function Call Trace

```
public class Newton {  
    public static double sqrt(double c) {  
        double epsilon = 1e-15;  
        if (c < 0) return Double.NaN;  
        double t = c;  
        while (Math.abs(t - c/t) > epsilon * t)  
            t = (c/t + t) / 2.0;  
        return t;  
    }  
    public static void main(String[] args) {  
        double[] a = new double[args.length];  
        for (int i = 0; i < args.length; i++)  
            a[i] = Double.parseDouble(args[i]);  
        for (int i = 0; i < a.length; i++)  
            System.out.println(sqrt(a[i]));  
    }  
}
```

{ "1", "2", "3" }

args[]

{ "1", "2", "3" }

```
% java Newton 1 2 3
```

Function Call Trace

```
public class Newton {  
    public static double sqrt(double c) {  
        double epsilon = 1e-15;  
        if (c < 0) return Double.NaN;  
        double t = c;  
        while (Math.abs(t - c/t) > epsilon * t)  
            t = (c/t + t) / 2.0;  
        return t;  
    }  
    public static void main(String[] args) {  
        double[] a = new double[args.length];  
        for (int i = 0; i < args.length; i++)  
            a[i] = Double.parseDouble(args[i]);  
        for (int i = 0; i < a.length; i++)  
            System.out.println(sqrt(a[i]));  
    }  
}
```

```
% java Newton 1 2 3
```

args[]

{ "1", "2", "3" }

a[]

{ 0.0, 0.0, 0.0 }

Function Call Trace

```
public class Newton {  
    public static double sqrt(double c) {  
        double epsilon = 1e-15;  
        if (c < 0) return Double.NaN;  
        double t = c;  
        while (Math.abs(t - c/t) > epsilon * t)  
            t = (c/t + t) / 2.0;  
        return t;  
    }  
  
    public static void main(String[] args) {  
        double[] a = new double[args.length];  
        for (int i = 0; i < args.length; i++)  
            a[i] = Double.parseDouble(args[i]);  
        for (int i = 0; i < a.length; i++)  
            System.out.println(sqrt(a[i]));  
    }  
}
```

```
% java Newton 1 2 3
```

args[]

{ "1", "2", "3" }

a[]

{ 0.0, 0.0, 0.0 }

i

0

Function Call Trace

```
public class Newton {  
    public static double sqrt(double c) {  
        double epsilon = 1e-15;  
        if (c < 0) return Double.NaN;  
        double t = c;  
        while (Math.abs(t - c/t) > epsilon * t)  
            t = (c/t + t) / 2.0;  
        return t;  
    }  
  
    public static void main(String[] args) {  
        double[] a = new double[args.length];  
        for (int i = 0; i < args.length; i++)  
            a[i] = Double.parseDouble(args[i]);  
        for (int i = 0; i < a.length; i++)  
            System.out.println(sqrt(a[i]));  
    }  
}
```

```
% java Newton 1 2 3
```

args[]

{ "1", "2", "3" }

a[]

{ 1.0, 0.0, 0.0 }

i

0

Function Call Trace

```
public class Newton {  
    public static double sqrt(double c) {  
        double epsilon = 1e-15;  
        if (c < 0) return Double.NaN;  
        double t = c;  
        while (Math.abs(t - c/t) > epsilon * t)  
            t = (c/t + t) / 2.0;  
        return t;  
    }  
  
    public static void main(String[] args) {  
        double[] a = new double[args.length];  
        for (int i = 0; i < args.length; i++)  
            a[i] = Double.parseDouble(args[i]);  
        for (int i = 0; i < a.length; i++)  
            System.out.println(sqrt(a[i]));  
    }  
}
```

```
% java Newton 1 2 3
```

args[]

{ "1", "2", "3" }

a[]

{ 1.0, 0.0, 0.0 }

i

1

Function Call Trace

```
public class Newton {  
    public static double sqrt(double c) {  
        double epsilon = 1e-15;  
        if (c < 0) return Double.NaN;  
        double t = c;  
        while (Math.abs(t - c/t) > epsilon * t)  
            t = (c/t + t) / 2.0;  
        return t;  
    }  
  
    public static void main(String[] args) {  
        double[] a = new double[args.length];  
        for (int i = 0; i < args.length; i++)  
            a[i] = Double.parseDouble(args[i]);  
        for (int i = 0; i < a.length; i++)  
            System.out.println(sqrt(a[i]));  
    }  
}
```

```
% java Newton 1 2 3
```

args[]

{ "1", "2", "3" }

a[]

{ 1.0, 2.0, 0.0 }

i

1

Function Call Trace

```
public class Newton {  
    public static double sqrt(double c) {  
        double epsilon = 1e-15;  
        if (c < 0) return Double.NaN;  
        double t = c;  
        while (Math.abs(t - c/t) > epsilon * t)  
            t = (c/t + t) / 2.0;  
        return t;  
    }  
    public static void main(String[] args) {  
        double[] a = new double[args.length];  
        for (int i = 0; i < args.length; i++)  
            a[i] = Double.parseDouble(args[i]);  
        for (int i = 0; i < a.length; i++)  
            System.out.println(sqrt(a[i]));  
    }  
}
```

```
% java Newton 1 2 3
```

args[]

{ "1", "2", "3" }

a[]

{ 1.0, 2.0, 0.0 }

i

2

Function Call Trace

```
public class Newton {  
    public static double sqrt(double c) {  
        double epsilon = 1e-15;  
        if (c < 0) return Double.NaN;  
        double t = c;  
        while (Math.abs(t - c/t) > epsilon * t)  
            t = (c/t + t) / 2.0;  
        return t;  
    }  
    public static void main(String[] args) {  
        double[] a = new double[args.length];  
        for (int i = 0; i < args.length; i++)  
            a[i] = Double.parseDouble(args[i]);  
        for (int i = 0; i < a.length; i++)  
            System.out.println(sqrt(a[i]));  
    }  
}
```

```
% java Newton 1 2 3
```

args[]

{ "1", "2", "3" }

a[]

{ 1.0, 2.0, 3.0 }

i

2

Function Call Trace

```
public class Newton {  
    public static double sqrt(double c) {  
        double epsilon = 1e-15;  
        if (c < 0) return Double.NaN;  
        double t = c;  
        while (Math.abs(t - c/t) > epsilon * t)  
            t = (c/t + t) / 2.0;  
        return t;  
    }  
    public static void main(String[] args) {  
        double[] a = new double[args.length];  
        for (int i = 0; i < args.length; i++)  
            a[i] = Double.parseDouble(args[i]);  
        for (int i = 0; i < a.length; i++)  
            System.out.println(sqrt(a[i]));  
    }  
}
```

```
% java Newton 1 2 3
```

args[]

{ "1", "2", "3" }

a[]

{ 1.0, 2.0, 3.0 }

i

3

← goes out
of scope

Function Call Trace

```
public class Newton {  
    public static double sqrt(double c) {  
        double epsilon = 1e-15;  
        if (c < 0) return Double.NaN;  
        double t = c;  
        while (Math.abs(t - c/t) > epsilon * t)  
            t = (c/t + t) / 2.0;  
        return t;  
    }  
  
    public static void main(String[] args) {  
        double[] a = new double[args.length];  
        for (int i = 0; i < args.length; i++)  
            a[i] = Double.parseDouble(args[i]);  
        for (int i = 0; i < a.length; i++)  
            System.out.println(sqrt(a[i]));  
    }  
}
```

```
% java Newton 1 2 3
```

args[]

{ "1", "2", "3" }

a[]

{ 1.0, 2.0, 3.0 }

i

0

Function Call Trace

```
public class Newton {  
    public static double sqrt(double c) {  
        double epsilon = 1e-15;  
        if (c < 0) return Double.NaN;  
        double t = c;  
        while (Math.abs(t - c/t) > epsilon * t)  
            t = (c/t + t) / 2.0;  
        return t;  
    }  
    public static void main(String[] args) {  
        double[] a = new double[args.length];  
        for (int i = 0; i < args.length; i++)  
            a[i] = Double.parseDouble(args[i]);  
        for (int i = 0; i < a.length; i++)  
            System.out.println(sqrt(a[i]));  
    }  
}
```

1.0 ←

1.0

args[]

{ "1", "2", "3" }

a[]

{ 1.0, 2.0, 3.0 }

i

0

```
% java Newton 1 2 3
```

Function Call Trace

```
public class Newton {  
    public static double sqrt(double c) {  
        double epsilon = 1e-15;  
        if (c < 0) return Double.NaN;  
        double t = c;  
        while (Math.abs(t - c/t) > epsilon * t)  
            t = (c/t + t) / 2.0;  
        return t;  
    }  
    public static void main(String[] args) {  
        double[] a = new double[args.length];  
        for (int i = 0; i < args.length; i++)  
            a[i] = Double.parseDouble(args[i]);  
        for (int i = 0; i < a.length; i++)  
            System.out.println(sqrt(a[i]));  
    }  
}
```

1.0 ←

1.0

c

1.0

args[]

{ "1", "2", "3" }

a[]

{ 1.0, 2.0, 3.0 }

i

0

```
% java Newton 1 2 3
```

1.0

Function Call Trace

```
public class Newton {  
    public static double sqrt(double c) {  
        double epsilon = 1e-15;  
        if (c < 0) return Double.NaN;  
        double t = c;  
        while (Math.abs(t - c/t) > epsilon * t)  
            t = (c/t + t) / 2.0;  
        return t;  
    }  
    public static void main(String[] args) {  
        double[] a = new double[args.length];  
        for (int i = 0; i < args.length; i++)  
            a[i] = Double.parseDouble(args[i]);  
        for (int i = 0; i < a.length; i++)  
            System.out.println(sqrt(a[i]));  
    }  
}
```

c

1.0

args[]

{ "1", "2", "3" }

a[]

{ 1.0, 2.0, 3.0 }

i

0

```
% java Newton 1 2 3
```

Function Call Trace

```
public class Newton {  
    public static double sqrt(double c) {  
        double epsilon = 1e-15;  
        if (c < 0) return Double.NaN;  
        double t = c;  
        while (Math.abs(t - c/t) > epsilon * t)  
            t = (c/t + t) / 2.0;  
        return t;  
    }  
    public static void main(String[] args) {  
        double[] a = new double[args.length];  
        for (int i = 0; i < args.length; i++)  
            a[i] = Double.parseDouble(args[i]);  
        for (int i = 0; i < a.length; i++)  
            System.out.println(sqrt(a[i]));  
    }  
}
```

```
% java Newton 1 2 3
```

c

1.0

args[]

{ "1", "2", "3" }

a[]

{ 1.0, 2.0, 3.0 }

i

0

Function Call Trace

```
public class Newton {  
    public static double sqrt(double c) {  
        double epsilon = 1e-15;  
        if (c < 0) return Double.NaN;  
        double t = c;  
        while (Math.abs(t - c/t) > epsilon * t)  
            t = (c/t + t) / 2.0;  
        return t;  
    }  
    public static void main(String[] args) {  
        double[] a = new double[args.length];  
        for (int i = 0; i < args.length; i++)  
            a[i] = Double.parseDouble(args[i]);  
        for (int i = 0; i < a.length; i++)  
            System.out.println(sqrt(a[i]));  
    }  
}
```

c	t
1.0	1.0

args[]
{ "1", "2", "3" }

a[]
{ 1.0, 2.0, 3.0 }

i
0

```
% java Newton 1 2 3
```

Function Call Trace

```
public class Newton {  
    public static double sqrt(double c) {  
        double epsilon = 1e-15;  
        if (c < 0) return Double.NaN;  
        double t = c;  
        while (Math.abs(t - c/t) > epsilon * t)  
            t = (c/t + t) / 2.0;  
        return t;  
    }  
    public static void main(String[] args) {  
        double[] a = new double[args.length];  
        for (int i = 0; i < args.length; i++)  
            a[i] = Double.parseDouble(args[i]);  
        for (int i = 0; i < a.length; i++)  
            System.out.println(sqrt(a[i]));  
    }  
}
```

c	t
1.0	1.0

args[]
{ "1", "2", "3" }

a[]
{ 1.0, 2.0, 3.0 }

i
0

```
% java Newton 1 2 3
```

Function Call Trace

```
public class Newton {  
    public static double sqrt(double c) {  
        double epsilon = 1e-15;  
        if (c < 0) return Double.NaN;  
        double t = c;  
        while (Math.abs(t - c/t) > epsilon * t)  
            t = (c/t + t) / 2.0;  
        return t;  
    }  
    public static void main(String[] args) {  
        double[] a = new double[args.length];  
        for (int i = 0; i < args.length; i++)  
            a[i] = Double.parseDouble(args[i]);  
        for (int i = 0; i < a.length; i++)  
            System.out.println(sqrt(a[i]));  
    }  
}
```

1.0 -----

1.0 ←-----

c	t
1.0	1.0

args[]
{ "1", "2", "3" }

a[]
{ 1.0, 2.0, 3.0 }

i
0

```
% java Newton 1 2 3
```

Function Call Trace

```
public class Newton {  
    public static double sqrt(double c) {  
        double epsilon = 1e-15;  
        if (c < 0) return Double.NaN;  
        double t = c;  
        while (Math.abs(t - c/t) > epsilon * t)  
            t = (c/t + t) / 2.0;  
        return t;  
    }  
    public static void main(String[] args) {  
        double[] a = new double[args.length];  
        for (int i = 0; i < args.length; i++)  
            a[i] = Double.parseDouble(args[i]);  
        for (int i = 0; i < a.length; i++)  
            System.out.println(sqrt(a[i]));  
    }  
}
```

args[]

{ "1", "2", "3" }

a[]

{ 1.0, 2.0, 3.0 }

i

0

1.0

```
% java Newton 1 2 3  
1.0
```

Function Call Trace

```
public class Newton {  
    public static double sqrt(double c) {  
        double epsilon = 1e-15;  
        if (c < 0) return Double.NaN;  
        double t = c;  
        while (Math.abs(t - c/t) > epsilon * t)  
            t = (c/t + t) / 2.0;  
        return t;  
    }  
  
    public static void main(String[] args) {  
        double[] a = new double[args.length];  
        for (int i = 0; i < args.length; i++)  
            a[i] = Double.parseDouble(args[i]);  
        for (int i = 0; i < a.length; i++)  
            System.out.println(sqrt(a[i]));  
    }  
}
```

```
% java Newton 1 2 3  
1.0
```

args[]

{ "1", "2", "3" }

a[]

{ 1.0, 2.0, 3.0 }

i

1

Function Call Trace

```
public class Newton {  
    public static double sqrt(double c) {  
        double epsilon = 1e-15;  
        if (c < 0) return Double.NaN;  
        double t = c;  
        while (Math.abs(t - c/t) > epsilon * t)  
            t = (c/t + t) / 2.0;  
        return t;  
    }  
    public static void main(String[] args) {  
        double[] a = new double[args.length];  
        for (int i = 0; i < args.length; i++)  
            a[i] = Double.parseDouble(args[i]);  
        for (int i = 0; i < a.length; i++)  
            System.out.println(sqrt(a[i]));  
    }  
}
```

2.0 ←

2.0

c
2.0

args[]
{ "1", "2", "3" }

a[]
{ 1.0, 2.0, 3.0 }

i
1

```
% java Newton 1 2 3  
1.0
```

Function Call Trace

```
public class Newton {  
    public static double sqrt(double c) {  
        double epsilon = 1e-15;  
        if (c < 0) return Double.NaN;  
        double t = c;  
        while (Math.abs(t - c/t) > epsilon * t)  
            t = (c/t + t) / 2.0;  
        return t;  
    }  
    public static void main(String[] args) {  
        double[] a = new double[args.length];  
        for (int i = 0; i < args.length; i++)  
            a[i] = Double.parseDouble(args[i]);  
        for (int i = 0; i < a.length; i++)  
            System.out.println(sqrt(a[i]));  
    }  
}
```

c

2.0

args[]

{ "1", "2", "3" }

a[]

{ 1.0, 2.0, 3.0 }

i

1

```
% java Newton 1 2 3  
1.0
```

Function Call Trace

```
public class Newton {  
    public static double sqrt(double c) {  
        double epsilon = 1e-15;  
        if (c < 0) return Double.NaN;  
        double t = c;  
        while (Math.abs(t - c/t) > epsilon * t)  
            t = (c/t + t) / 2.0;  
        return t;  
    }  
    public static void main(String[] args) {  
        double[] a = new double[args.length];  
        for (int i = 0; i < args.length; i++)  
            a[i] = Double.parseDouble(args[i]);  
        for (int i = 0; i < a.length; i++)  
            System.out.println(sqrt(a[i]));  
    }  
}
```

c

2.0

args[]

{ "1", "2", "3" }

a[]

{ 1.0, 2.0, 3.0 }

i

1

```
% java Newton 1 2 3  
1.0
```


Function Call Trace

```
public class Newton {  
    public static double sqrt(double c) {  
        double epsilon = 1e-15;  
        if (c < 0) return Double.NaN;  
        double t = c;  
        while (Math.abs(t - c/t) > epsilon * t)  
            t = (c/t + t) / 2.0;  
        return t;  
    }  
    public static void main(String[] args) {  
        double[] a = new double[args.length];  
        for (int i = 0; i < args.length; i++)  
            a[i] = Double.parseDouble(args[i]);  
        for (int i = 0; i < a.length; i++)  
            System.out.println(sqrt(a[i]));  
    }  
}
```

c	t
2.0	2.0

args[]
{ "1", "2", "3" }

a[]
{ 1.0, 2.0, 3.0 }

i
1

```
% java Newton 1 2 3  
1.0
```

Function Call Trace

```
public class Newton {  
    public static double sqrt(double c) {  
        double epsilon = 1e-15;  
        if (c < 0) return Double.NaN;  
        double t = c;  
        while (Math.abs(t - c/t) > epsilon * t)  
            t = (c/t + t) / 2.0;  
        return t;  
    }  
    public static void main(String[] args) {  
        double[] a = new double[args.length];  
        for (int i = 0; i < args.length; i++)  
            a[i] = Double.parseDouble(args[i]);  
        for (int i = 0; i < a.length; i++)  
            System.out.println(sqrt(a[i]));  
    }  
}
```

c	t
2.0	2.0

args[]
{ "1", "2", "3" }

a[]
{ 1.0, 2.0, 3.0 }

i
1

```
% java Newton 1 2 3  
1.0
```

Function Call Trace

```
public class Newton {  
    public static double sqrt(double c) {  
        double epsilon = 1e-15;  
        if (c < 0) return Double.NaN;  
        double t = c;  
        while (Math.abs(t - c/t) > epsilon * t)  
            t = (c/t + t) / 2.0;  
        return t;  
    }  
    public static void main(String[] args) {  
        double[] a = new double[args.length];  
        for (int i = 0; i < args.length; i++)  
            a[i] = Double.parseDouble(args[i]);  
        for (int i = 0; i < a.length; i++)  
            System.out.println(sqrt(a[i]));  
    }  
}
```

c	t
2.0	1.41

args[]
{ "1", "2", "3" }

a[]
{ 1.0, 2.0, 3.0 }

i
1

```
% java Newton 1 2 3  
1.0
```

Function Call Trace

```
public class Newton {  
    public static double sqrt(double c) {  
        double epsilon = 1e-15;  
        if (c < 0) return Double.NaN;  
        double t = c;  
        while (Math.abs(t - c/t) > epsilon * t)  
            t = (c/t + t) / 2.0;  
        return t;  
    }  
    public static void main(String[] args) {  
        double[] a = new double[args.length];  
        for (int i = 0; i < args.length; i++)  
            a[i] = Double.parseDouble(args[i]);  
        for (int i = 0; i < a.length; i++)  
            System.out.println(sqrt(a[i]));  
    }  
}
```

c	t
2.0	1.41

args[]
{ "1", "2", "3" }

a[]
{ 1.0, 2.0, 3.0 }

i
1

```
% java Newton 1 2 3  
1.0
```

Function Call Trace

```
public class Newton {  
    public static double sqrt(double c) {  
        double epsilon = 1e-15;  
        if (c < 0) return Double.NaN;  
        double t = c;  
        while (Math.abs(t - c/t) > epsilon * t)  
            t = (c/t + t) / 2.0;  
        return t;  
    }  
    public static void main(String[] args) {  
        double[] a = new double[args.length];  
        for (int i = 0; i < args.length; i++)  
            a[i] = Double.parseDouble(args[i]);  
        for (int i = 0; i < a.length; i++)  
            System.out.println(sqrt(a[i]));  
    }  
}
```

1.414213562373095 -----

1.414213562373095 ←-----

```
% java Newton 1 2 3  
1.0
```

c	t
2.0	1.41

args[]
{ "1", "2", "3" }

a[]
{ 1.0, 2.0, 3.0 }

i
1

Function Call Trace

```
public class Newton {  
    public static double sqrt(double c) {  
        double epsilon = 1e-15;  
        if (c < 0) return Double.NaN;  
        double t = c;  
        while (Math.abs(t - c/t) > epsilon * t)  
            t = (c/t + t) / 2.0;  
        return t;  
    }  
  
    public static void main(String[] args) {  
        double[] a = new double[args.length];  
        for (int i = 0; i < args.length; i++)  
            a[i] = Double.parseDouble(args[i]);  
        for (int i = 0; i < a.length; i++)  
            System.out.println(sqrt(a[i]));  
    }  
}
```

args[]

{ "1", "2", "3" }

a[]

{ 1.0, 2.0, 3.0 }

i

1

1.414213562373095

```
% java Newton 1 2 3  
1.0  
1.414213562373095
```

Function Call Trace

```
public class Newton {  
    public static double sqrt(double c) {  
        double epsilon = 1e-15;  
        if (c < 0) return Double.NaN;  
        double t = c;  
        while (Math.abs(t - c/t) > epsilon * t)  
            t = (c/t + t) / 2.0;  
        return t;  
    }  
  
    public static void main(String[] args) {  
        double[] a = new double[args.length];  
        for (int i = 0; i < args.length; i++)  
            a[i] = Double.parseDouble(args[i]);  
        for (int i = 0; i < a.length; i++)  
            System.out.println(sqrt(a[i]));  
    }  
}
```

```
% java Newton 1 2 3  
1.0  
1.414213562373095
```

args[]

{ "1", "2", "3" }

a[]

{ 1.0, 2.0, 3.0 }

i

2

Function Call Trace

```
public class Newton {  
    public static double sqrt(double c) {  
        double epsilon = 1e-15;  
        if (c < 0) return Double.NaN;  
        double t = c;  
        while (Math.abs(t - c/t) > epsilon * t)  
            t = (c/t + t) / 2.0;  
        return t;  
    }  
    public static void main(String[] args) {  
        double[] a = new double[args.length];  
        for (int i = 0; i < args.length; i++)  
            a[i] = Double.parseDouble(args[i]);  
        for (int i = 0; i < a.length; i++)  
            System.out.println(sqrt(a[i]));  
    }  
}
```

3.0 ←

3.0

args[]

{ "1", "2", "3" }

a[]

{ 1.0, 2.0, 3.0 }

i

2

3.0

```
% java Newton 1 2 3  
1.0  
1.414213562373095
```


Function Call Trace

```
public class Newton {  
    public static double sqrt(double c) {  
        double epsilon = 1e-15;  
        if (c < 0) return Double.NaN;  
        double t = c;  
        while (Math.abs(t - c/t) > epsilon * t)  
            t = (c/t + t) / 2.0;  
        return t;  
    }  
    public static void main(String[] args) {  
        double[] a = new double[args.length];  
        for (int i = 0; i < args.length; i++)  
            a[i] = Double.parseDouble(args[i]);  
        for (int i = 0; i < a.length; i++)  
            System.out.println(sqrt(a[i]));  
    }  
}
```

1.7320508075688772 -----

1.7320508075688772 ←---

```
% java Newton 1 2 3  
1.0  
1.414213562373095
```

c

3.0

args[]

{ "1", "2", "3" }

a[]

{ 1.0, 2.0, 3.0 }

i

2

Function Call Trace

```
public class Newton {  
    public static double sqrt(double c) {  
        double epsilon = 1e-15;  
        if (c < 0) return Double.NaN;  
        double t = c;  
        while (Math.abs(t - c/t) > epsilon * t)  
            t = (c/t + t) / 2.0;  
        return t;  
    }  
    public static void main(String[] args) {  
        double[] a = new double[args.length];  
        for (int i = 0; i < args.length; i++)  
            a[i] = Double.parseDouble(args[i]);  
        for (int i = 0; i < a.length; i++)  
            System.out.println(sqrt(a[i]));  
    }  
}
```

```
% java Newton 1 2 3  
1.0  
1.414213562373095  
1.7320508075688772
```

args[]

{ "1", "2", "3" }

a[]

{ 1.0, 2.0, 3.0 }

i

2

Function Call Trace

```
public class Newton {  
    public static double sqrt(double c) {  
        double epsilon = 1e-15;  
        if (c < 0) return Double.NaN;  
        double t = c;  
        while (Math.abs(t - c/t) > epsilon * t)  
            t = (c/t + t) / 2.0;  
        return t;  
    }  
  
    public static void main(String[] args) {  
        double[] a = new double[args.length];  
        for (int i = 0; i < args.length; i++)  
            a[i] = Double.parseDouble(args[i]);  
        for (int i = 0; i < a.length; i++)  
            System.out.println(sqrt(a[i]));  
    }  
}
```

```
% java Newton 1 2 3  
1.0  
1.414213562373095  
1.7320508075688772
```

args[]

{ "1", "2", "3" }

a[]

{ 1.0, 2.0, 3.0 }

i

3

Function Call Trace

```
public class Newton {  
    public static double sqrt(double c) {  
        double epsilon = 1e-15;  
        if (c < 0) return Double.NaN;  
        double t = c;  
        while (Math.abs(t - c/t) > epsilon * t)  
            t = (c/t + t) / 2.0;  
        return t;  
    }  
  
    public static void main(String[] args) {  
        double[] a = new double[args.length];  
        for (int i = 0; i < args.length; i++)  
            a[i] = Double.parseDouble(args[i]);  
        for (int i = 0; i < a.length; i++)  
            System.out.println(sqrt(a[i]));  
    }  
}
```

args[]

{ "1", "2", "3" }

a[]

{ 1.0, 2.0, 3.0 }

```
% java Newton 1 2 3  
1.0  
1.414213562373095  
1.7320508075688772
```