Control Structures and Methods

An Interesting Article

“For Newcomers in Silicon Valley, the Dream of Entrepreneurship Still Lives”

http://www.nytimes.com/2012/01/25/us/silicon-valley-newcomers-are-still-dreaming-big.html

This is called the **initialization statement** and is performed before the loop starts.

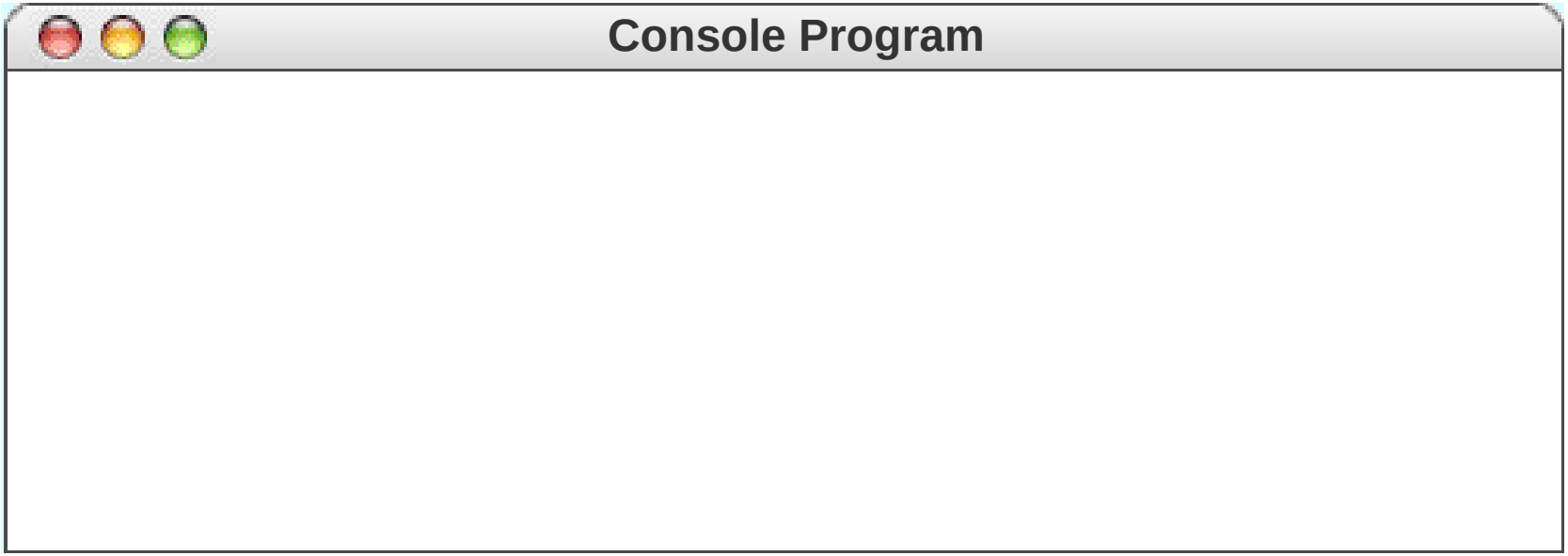
This is called the **step** or **increment** and is performed at the end of each loop iteration.

```
for (int i = 0; i < 3; i++) {
    ...
}
```

This is called the **loop condition** or **termination condition**. The loop will check whether this statement is true before each execution.

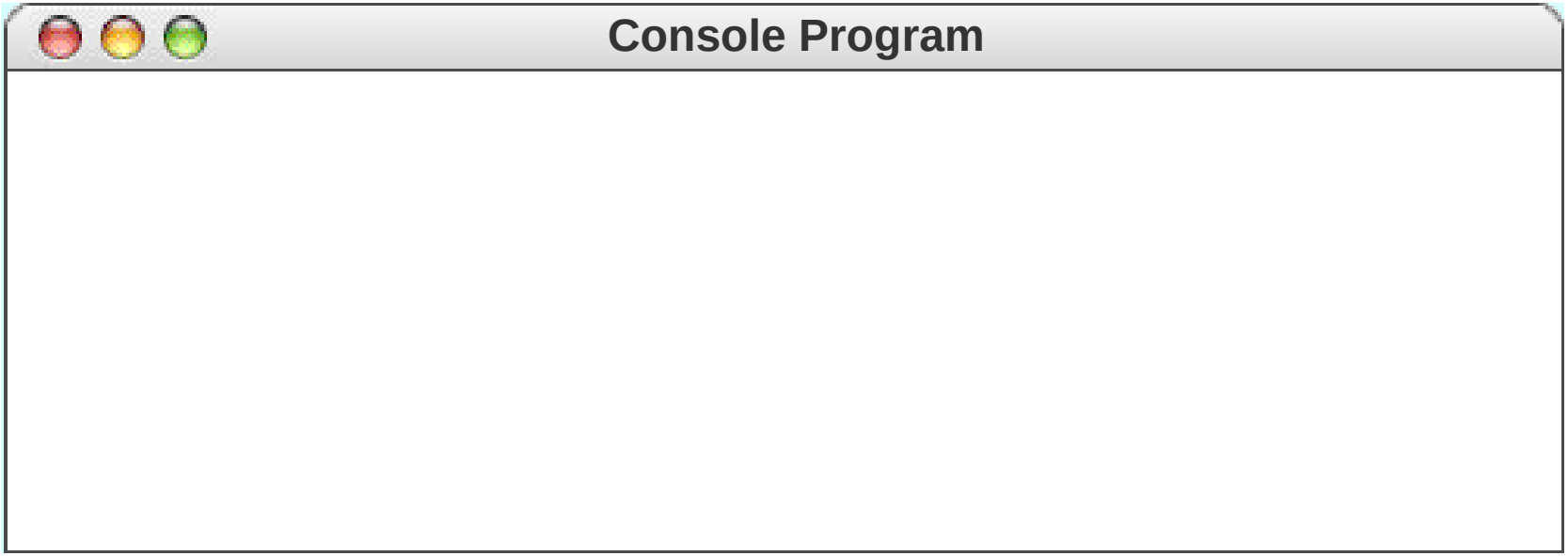
Nyan nyan nyan nyan, nyan nyan nyan
nyan nyan, nyan, nyan nyan nyan ...

```
for (int i = 0; i < 4; i++) {
    println("Nyan!");
}
```



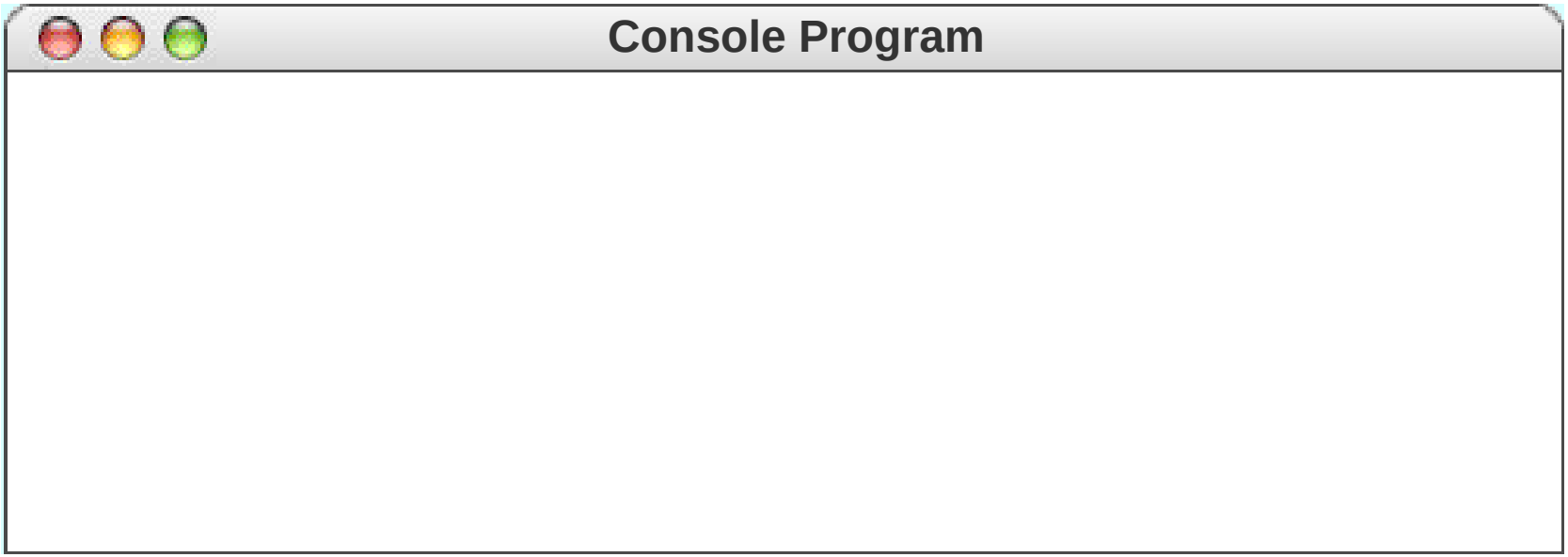
```
for (int i = 0; i < 4; i++) {
    println("Nyan!");
}
```

int i



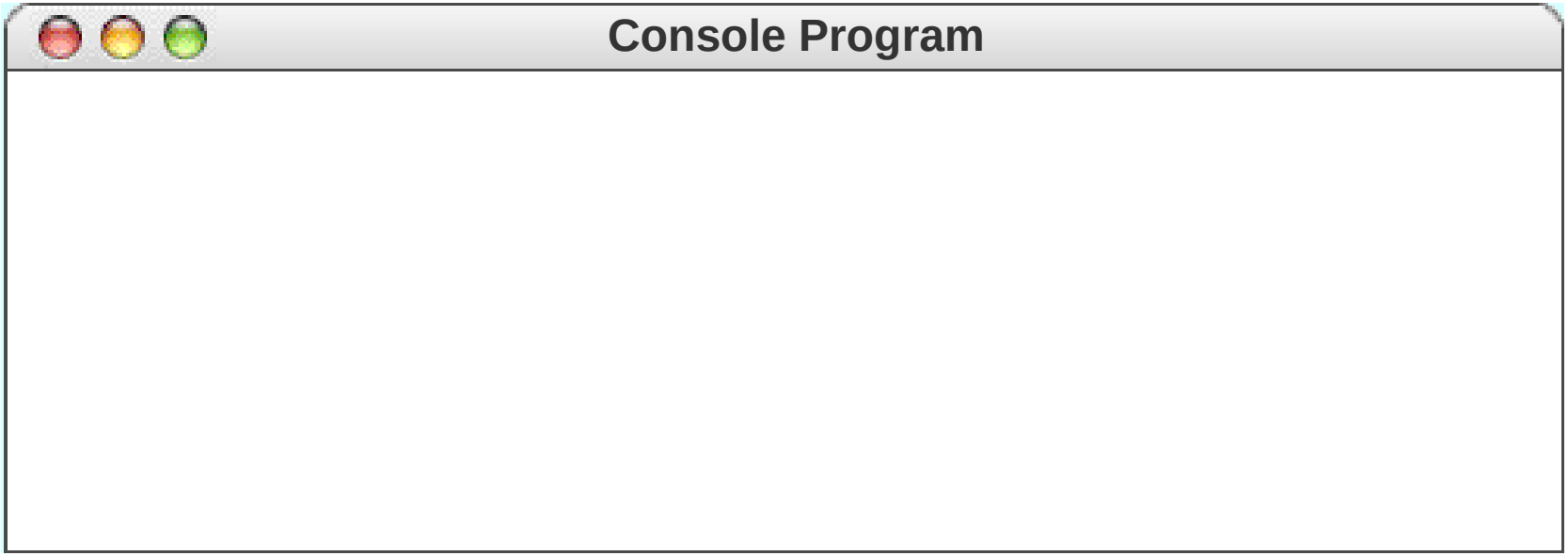
```
for (int i = 0; i < 4; i++) {
    println("Nyan!");
}
```

int i



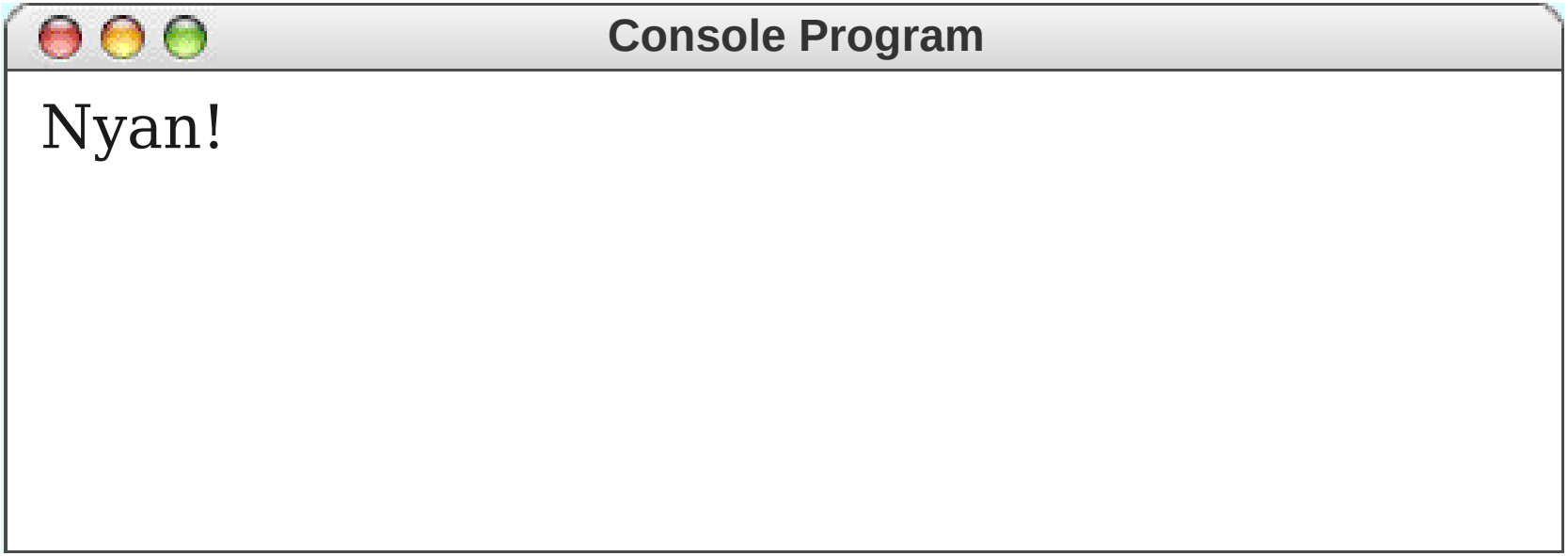
```
for (int i = 0; i < 4; i++) {
    println("Nyan!");
}
```

int i



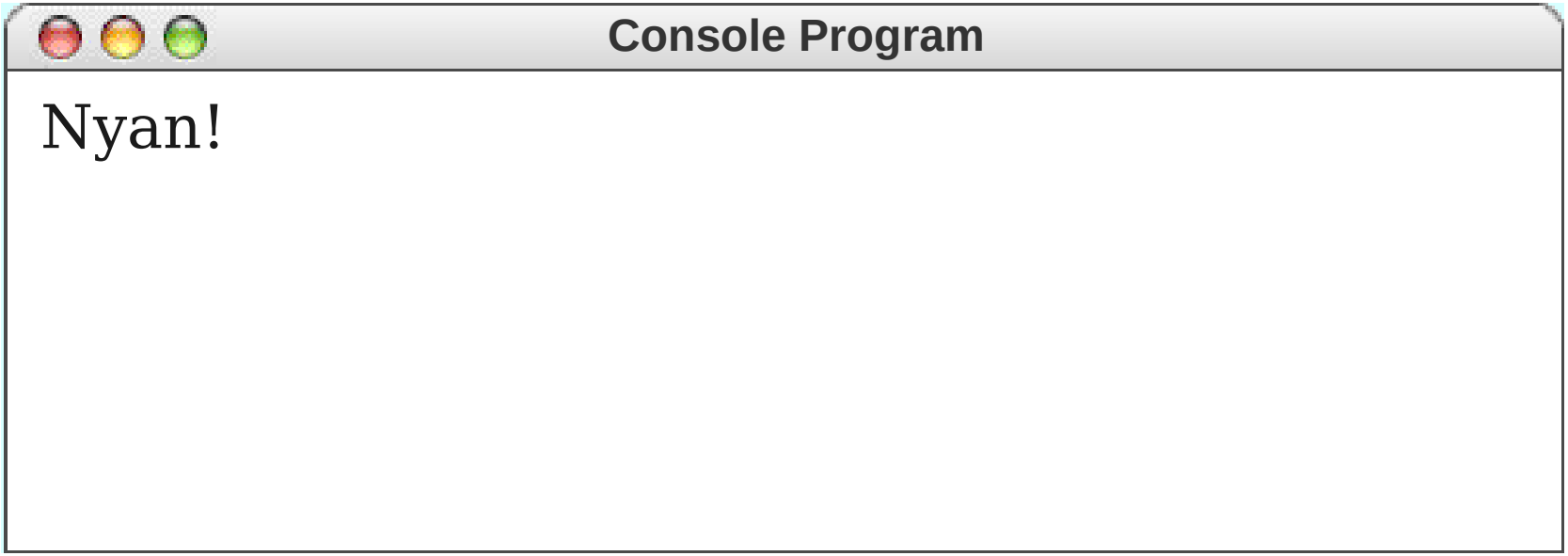
```
for (int i = 0; i < 4; i++) {
    println("Nyan!");
}
```

int i



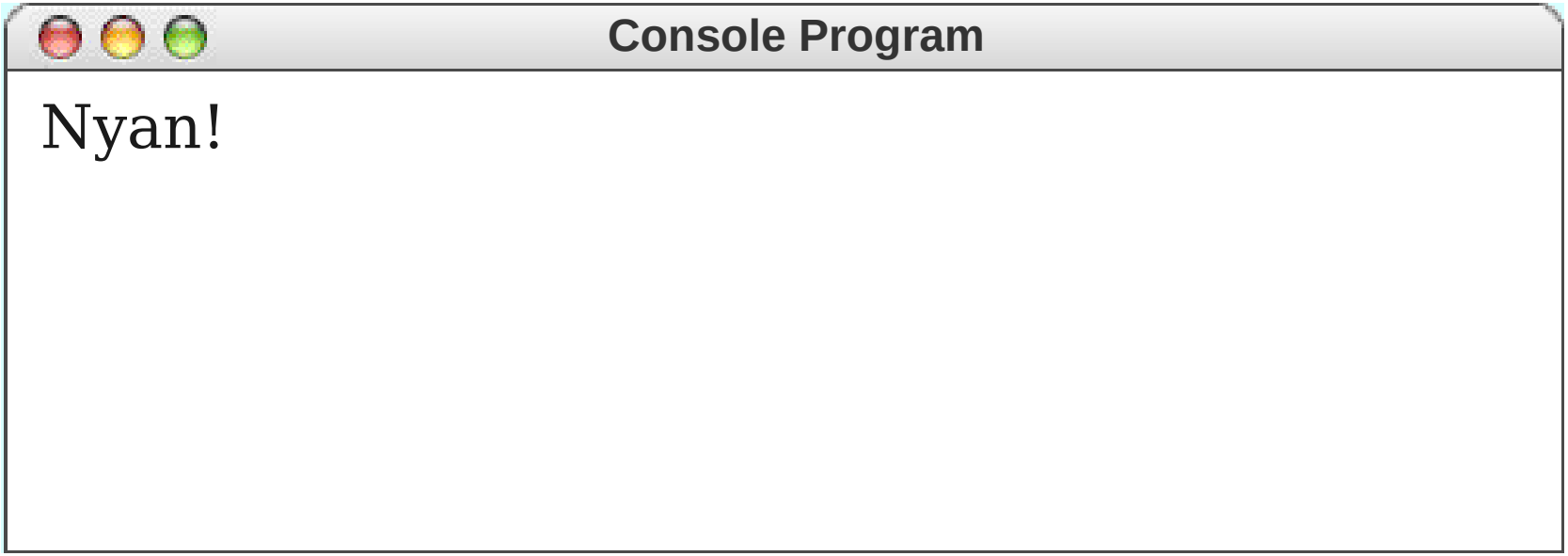
```
for (int i = 0; i < 4; i++) {
    println("Nyan!");
}
```

int i



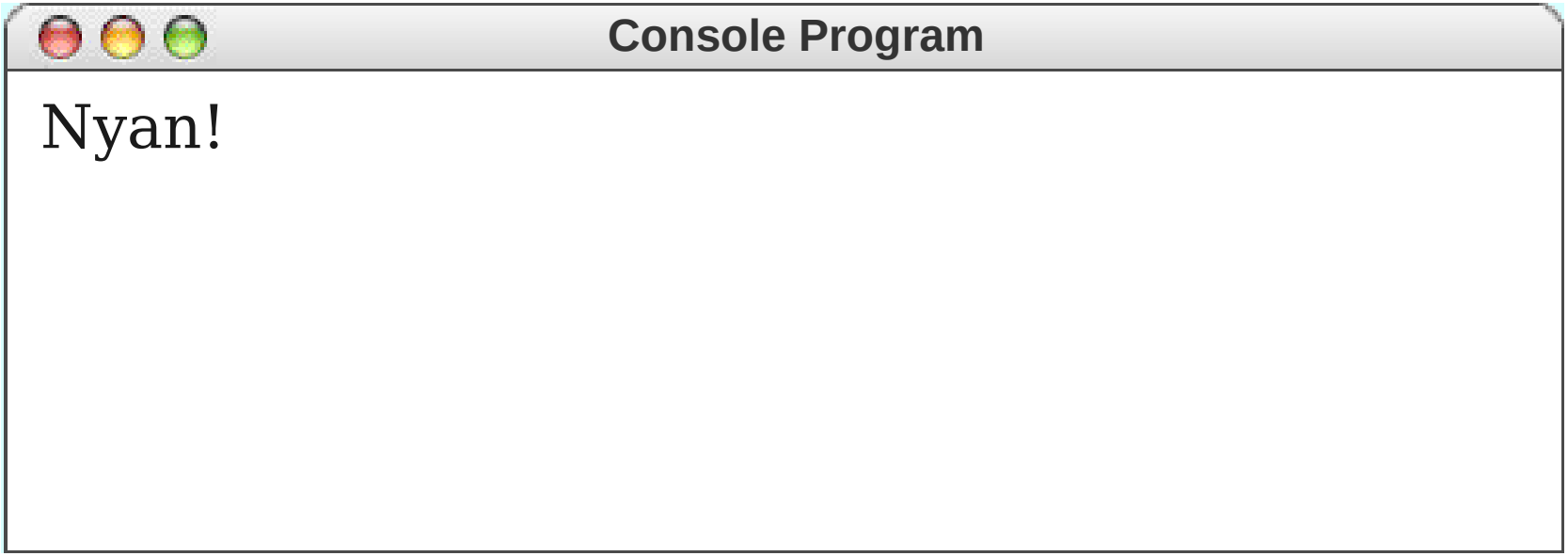
```
for (int i = 0; i < 4; i++) {
    println("Nyan!");
}
```

int i



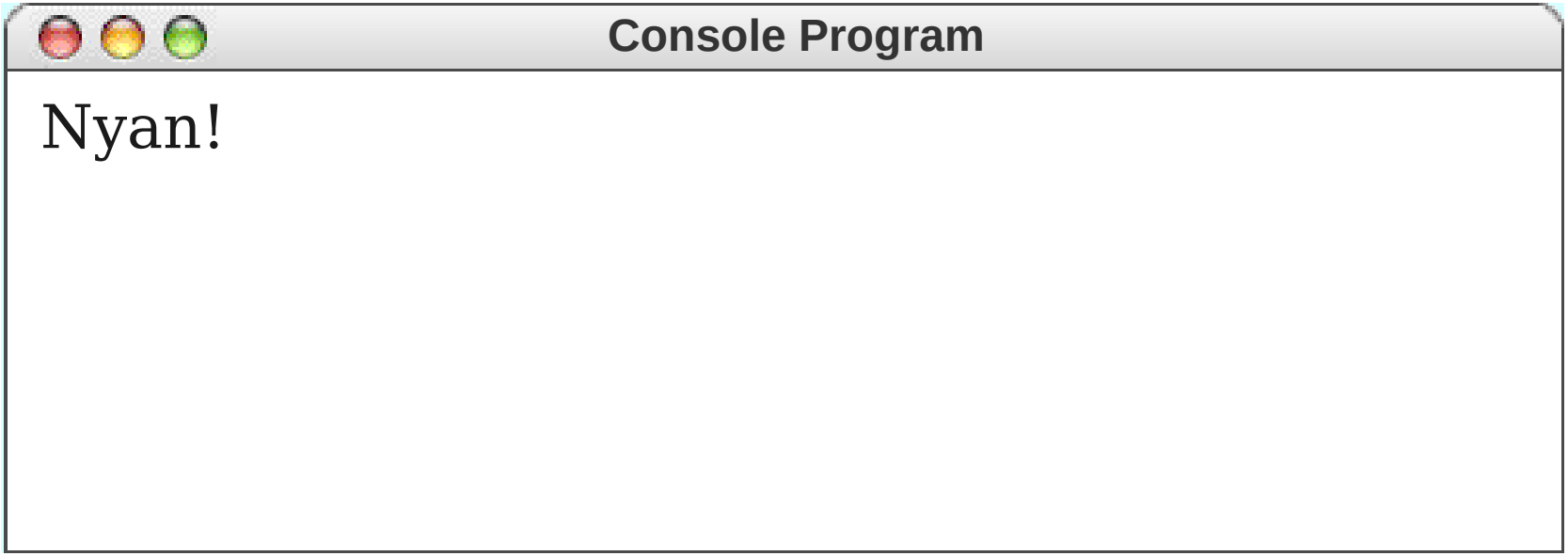
```
for (int i = 0; i < 4; i++) {
    println("Nyan!");
}
```

int i



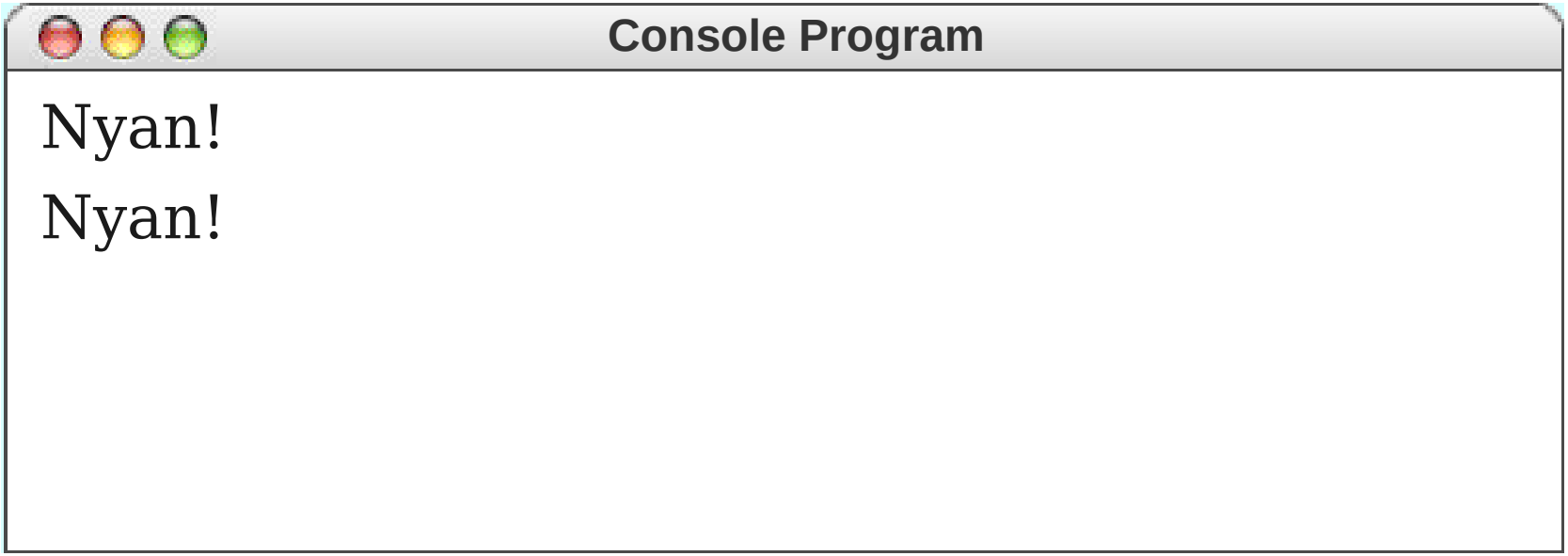
```
for (int i = 0; i < 4; i++) {
    println("Nyan!");
}
```

int i



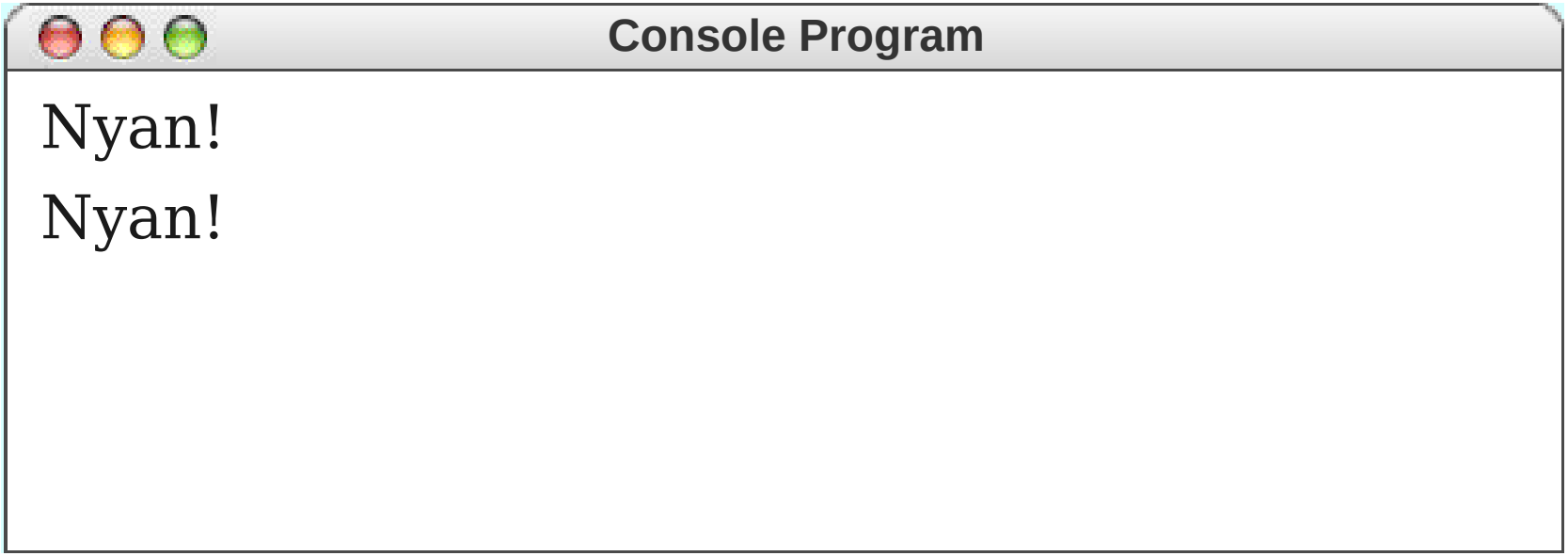
```
for (int i = 0; i < 4; i++) {
    println("Nyan!");
}
```

int i



```
for (int i = 0; i < 4; i++) {
    println("Nyan!");
}
```

int i



```
for (int i = 0; i < 4; i++) {
    println("Nyan!");
}
```

int i | 2

Console Program

Nyan!

Nyan!

```
for (int i = 0; i < 4; i++) {
    println("Nyan!");
}
```

int i | 2

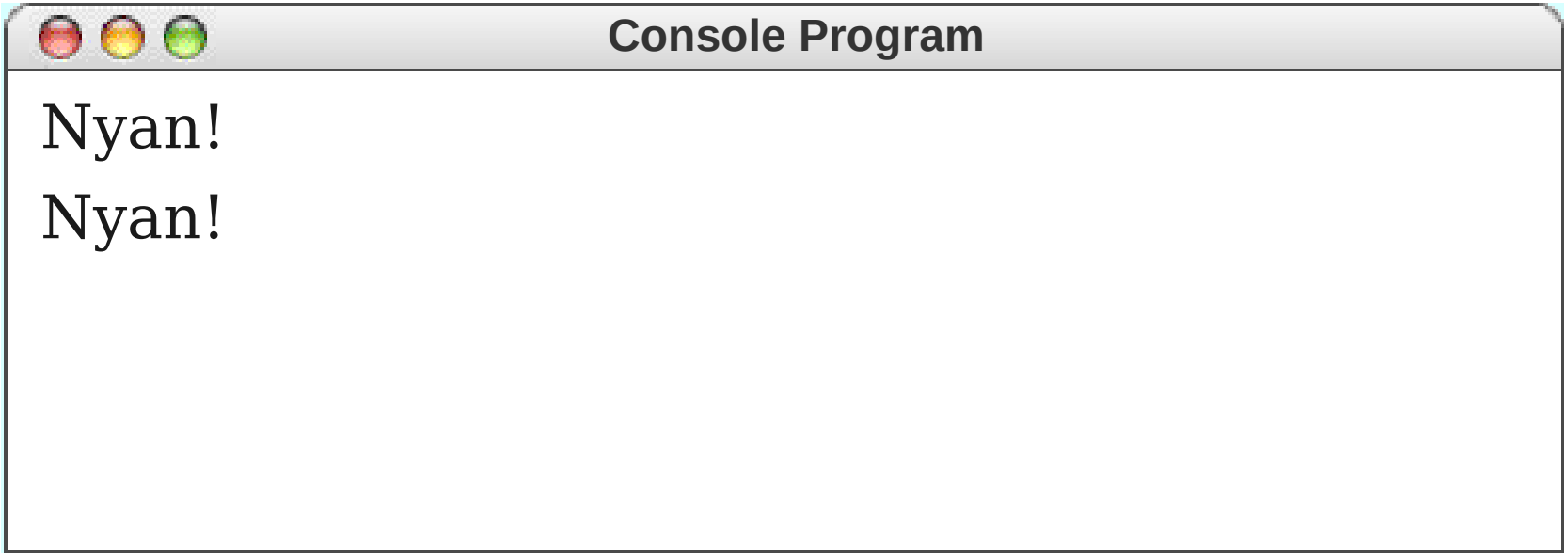
Console Program

Nyan!

Nyan!

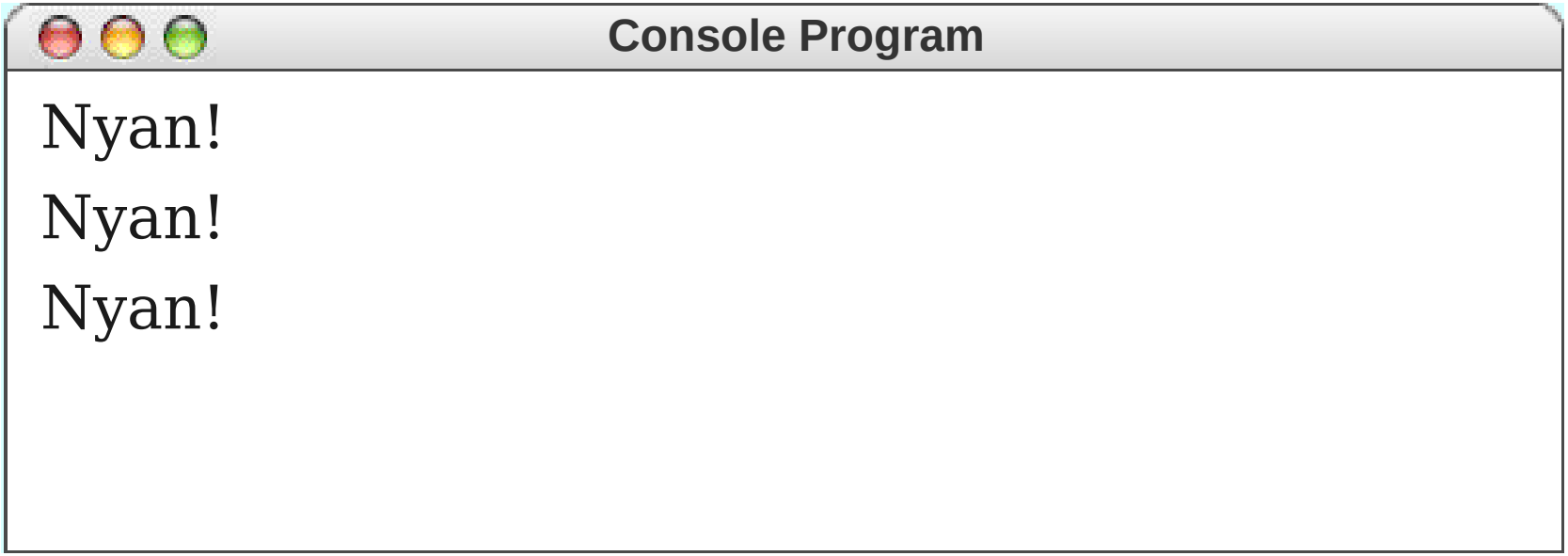
```
for (int i = 0; i < 4; i++) {
    println("Nyan!");
}
```

int i | 2



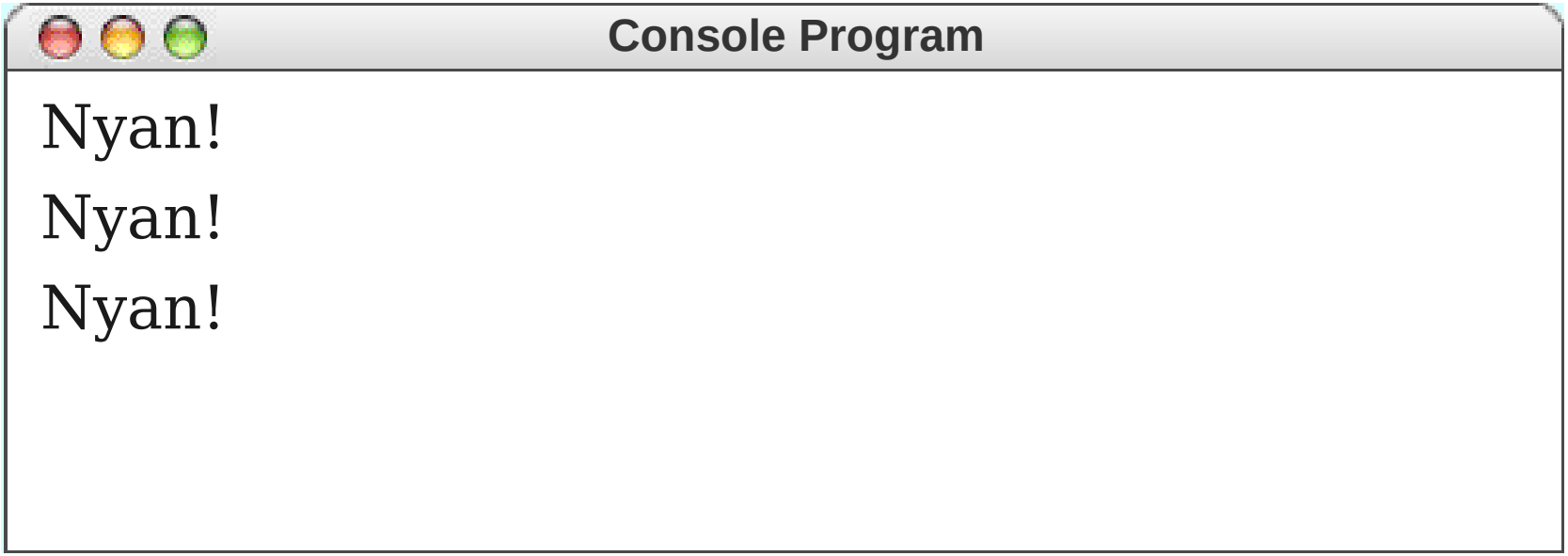
```
for (int i = 0; i < 4; i++) {
    println("Nyan!");
}
```

int i | 2



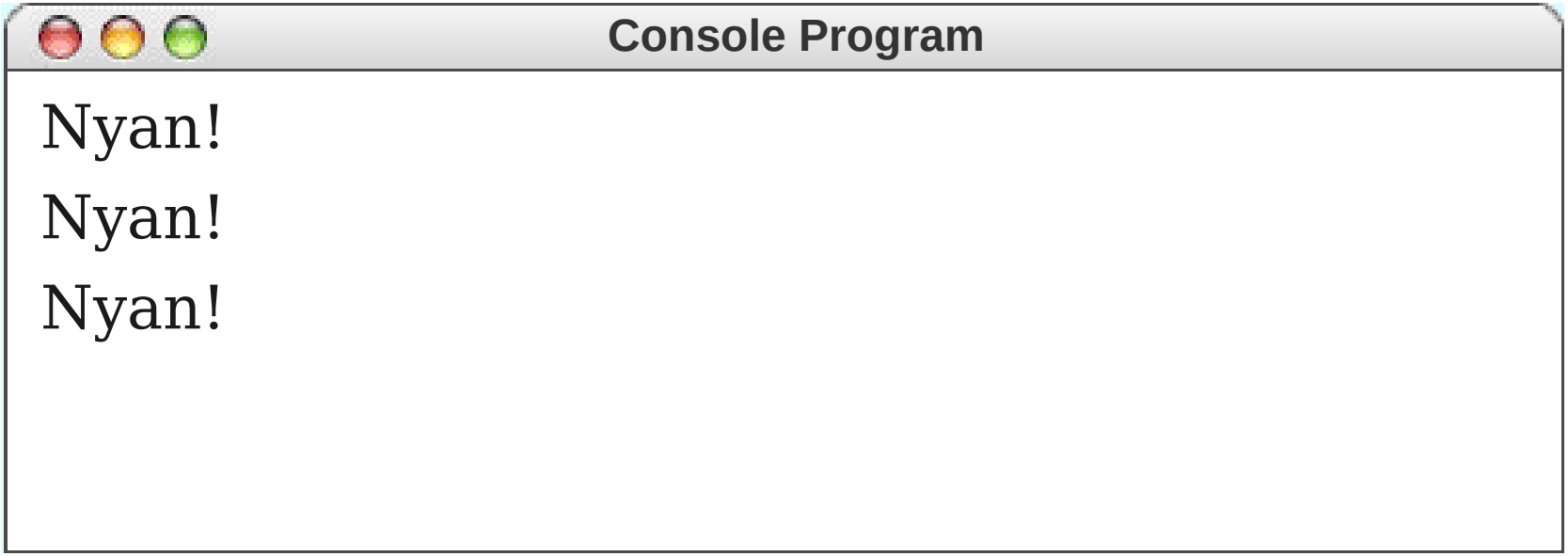
```
for (int i = 0; i < 4; i++) {
    println("Nyan!");
}
```

int i | 2



```
for (int i = 0; i < 4; i++) {
    println("Nyan!");
}
```

int i | 3



```
for (int i = 0; i < 4; i++) {
    println("Nyan!");
}
```

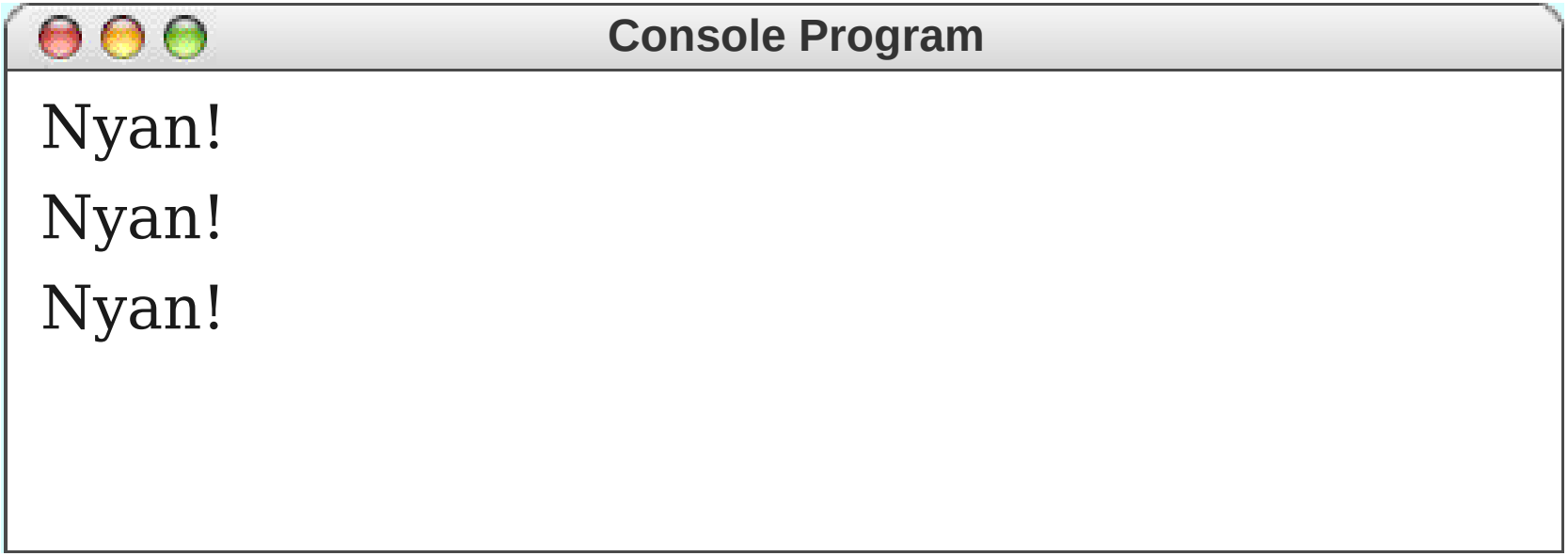
int i | 3

Console Program

Nyan!
Nyan!
Nyan!

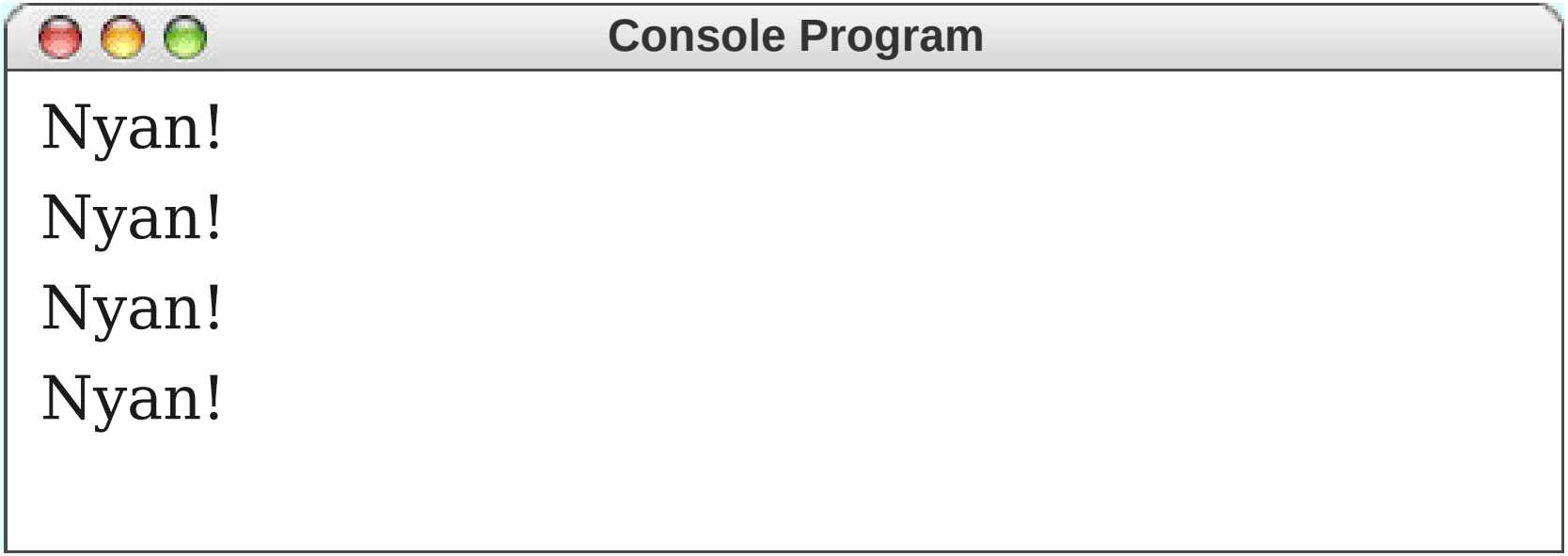
```
for (int i = 0; i < 4; i++) {
    println("Nyan!");
}
```

int i | 3



```
for (int i = 0; i < 4; i++) {
    println("Nyan!");
}
```

int i | 3



```
for (int i = 0; i < 4; i++) {
    println("Nyan!");
}
```

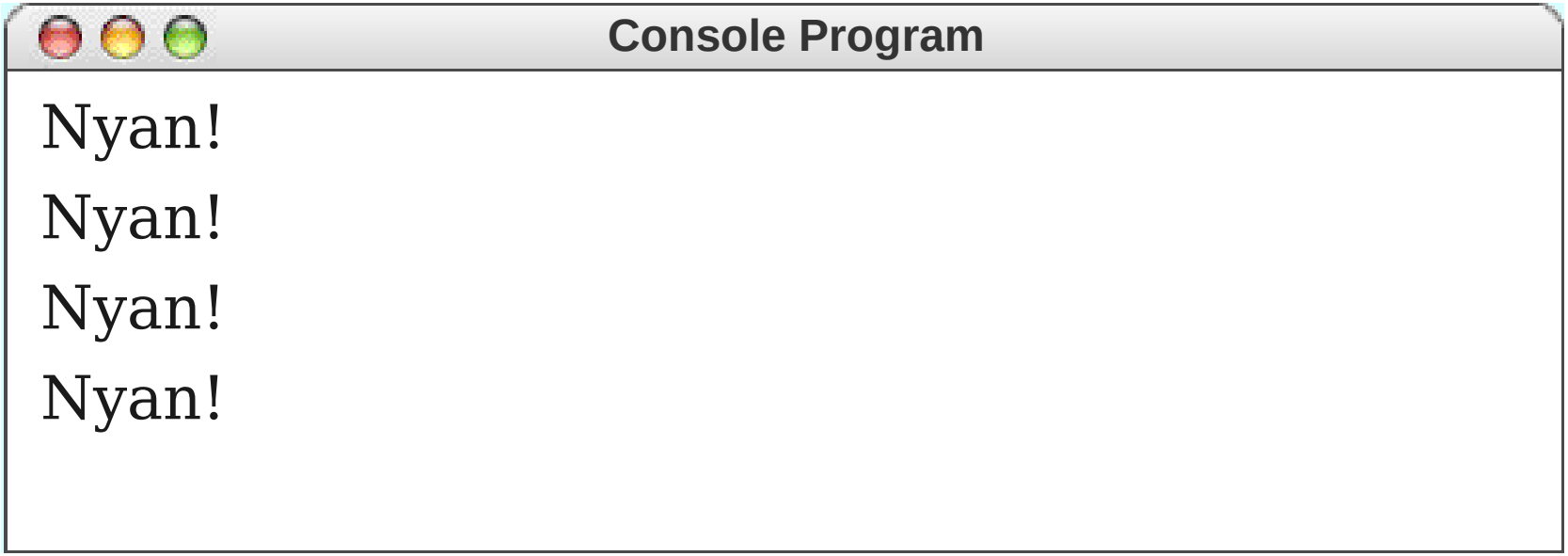
int i | 3

Console Program

```
Nyan!
Nyan!
Nyan!
Nyan!
```

```
for (int i = 0; i < 4; i++) {
    println("Nyan!");
}
```

int i | 4



```
for (int i = 0; i < 4; i++) {
    println("Nyan!");
}
```

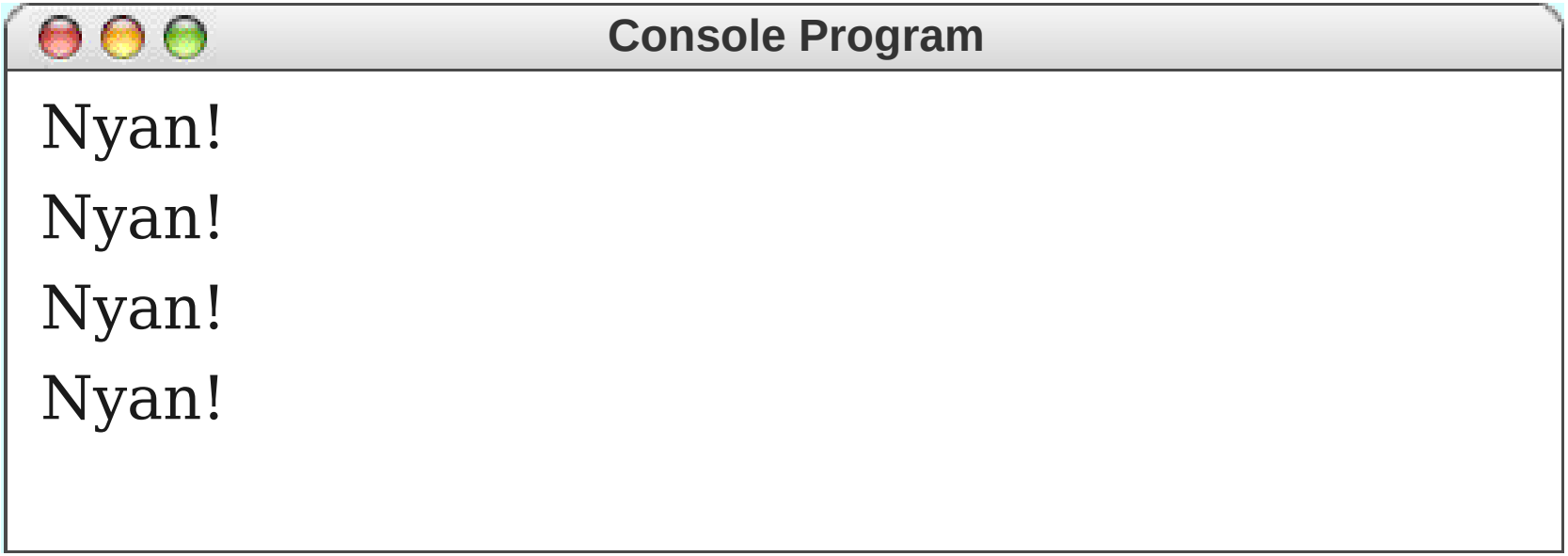
int i | 4

Console Program

```
Nyan!
Nyan!
Nyan!
Nyan!
```

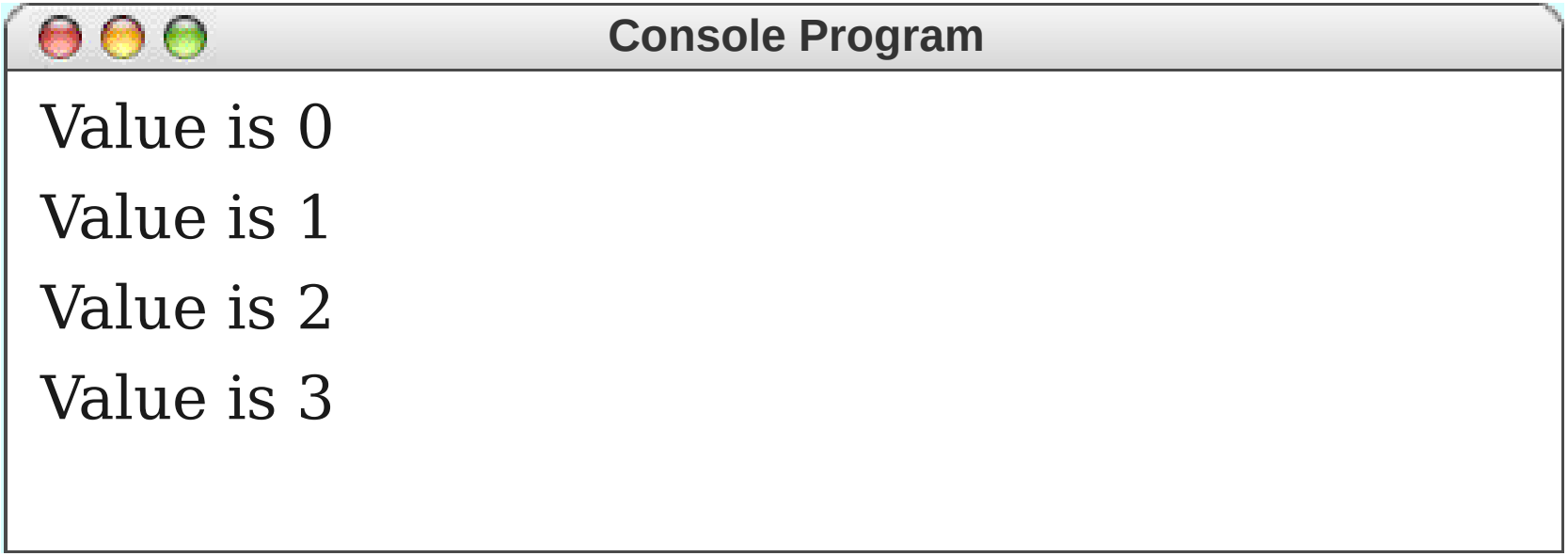
```
for (int i = 0; i < 4; i++) {
    println("Nyan!");
}
```

int i | 4



Accessing the Loop Counter

```
for (int i = 0; i < 4; i++) {
    println("Value is " + i);
}
```



```
for (int i = 5; i > 0; i--) {
    println(i + "...");
}
println("Lift-off!");
```

Console Program

```
5...
4...
3...
2...
1...
Lift-off!
```

http://www.youtube.com/watch?v=zGNryrsT7OI

T-15 Seconds: Guidance is Internal
T-9 Seconds: Ignition Sequence Start
T-0 Seconds: All Engines Running

```
for (int i = 30; i > 0; i--) {
    println("T-" + i + "...");
}
println("Lift-off!");
```

`if` statement

- **General form:** `if (`*condition*`) {`
  statements
  `}`

  Any boolean condition/variable

```
if (first > second)              // can omit braces
 println("First is bigger");     // if one statement

if (first > second) {
  println("Brace yourself...");
  println("...for an embrace!");
}
```

- Use braces with `if` with more than one statement
- Good idea to use braces (block) even if there is only one statement in the `if`

```java
private static final int COUNTDOWN_START = 30;
private static final int GUIDANCE_START = 15;
private static final int IGNITION_START = 9;

public void run() {
    /* Do the launch countdown! */
    for (int i = COUNTDOWN_START; i > 0; i--) {
        println("T-" + i + " seconds.");

        /* Specific mission commands. */
        if (i == GUIDANCE_START) {
            println("Guidance is internal.");
        }
        if (i == IGNITION_START) {
            println("Ignition sequence start.");
        }
    }

    println("All engines running. Lift-off!");
}
```

```
private static final int COUNTDOWN_START = 30;
private static final int GUIDANCE_START = 15;
private static final int IGNITION_START = 9;

public void run() {
    /* Do the launch countdown! */
    for (int i = COUNTDOWN_START; i > 0; i--) {
        println("T-" + i + " seconds.");

        /* Specific mission commands. */
        if (i == GUIDANCE_START) {
            println("Guidance is internal.");
        }
        if (i == IGNITION_START) {
            println("Ignition sequence start.");
        }
    }

    println("All engines running. Lift-off!");
}
```

`if-else` statement

- **General form:** `if (`*condition*`) {`
  statements
  `} else {`
  statements
  `}`

```
if ((num % 2) == 0) {
  println("num is even");
} else {
  println("num is odd");
  println("and so are you");
}
```

Cascading `if`

```
if (score >= 90) {
  println(" AWWWW YEAHHHHH ");
} else if (score >= 80) {
  println(" <(^_^)> ");
} else if (score >= 70) {
  println(" :-| ");
} else if (score >= 60) {
  println(" ಠ_ಠ ");
} else {
  println(" (ﾉ°□° )ﾉ︵┻━┻ ");
}
```

Control Structures in Karel

```
for
if
while
```

Control Structures in Karel

```
for
if
while
```

The **`while`** Loop

```
while (condition) {
        … statements …
}
```

- Checks ***condition*** before each iteration and executes ***statements*** if true.
- Does **not** check ***condition*** in the middle of the loop.

`while` loop

Example:

```
int x = 15;
while (x > 1) {
    x /= 2;
    println(x);
}
```

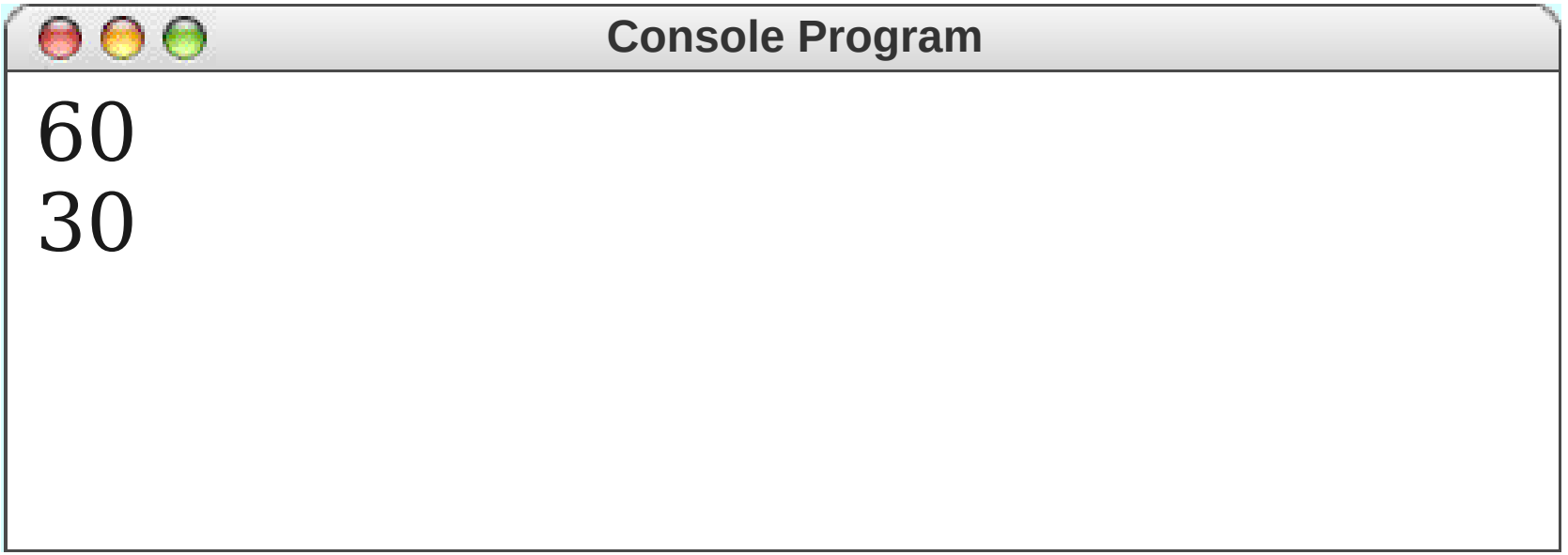
Console Program

```
7
3
1
```

break-ing out of a Loop

- The **break** statement immediately exits a loop.

```
int x = 120;
while (x > 1) {
    x /= 2;
    if (x % 2 == 1)
        break;
    println(x);
}
```



Looping Forever

- Recall: **`while`** loops iterate as long as their condition evaluates to **`true`**.
- A loop of the form **`while (true)`** will loop forever (or until a **`break`** is executed).

```
while (true) {
        …
}
```

The “Loop-and-a-Half” Idiom

- Often you will need to
  - read a value from the user,
  - decide whether to continue, and if so
  - process the value.
- The **loop-and-a-half idiom** can be used:

```
while (true) {
    /* … get a value from the user … */
    if (condition)
        break;

    /* … process the value … */
}
```

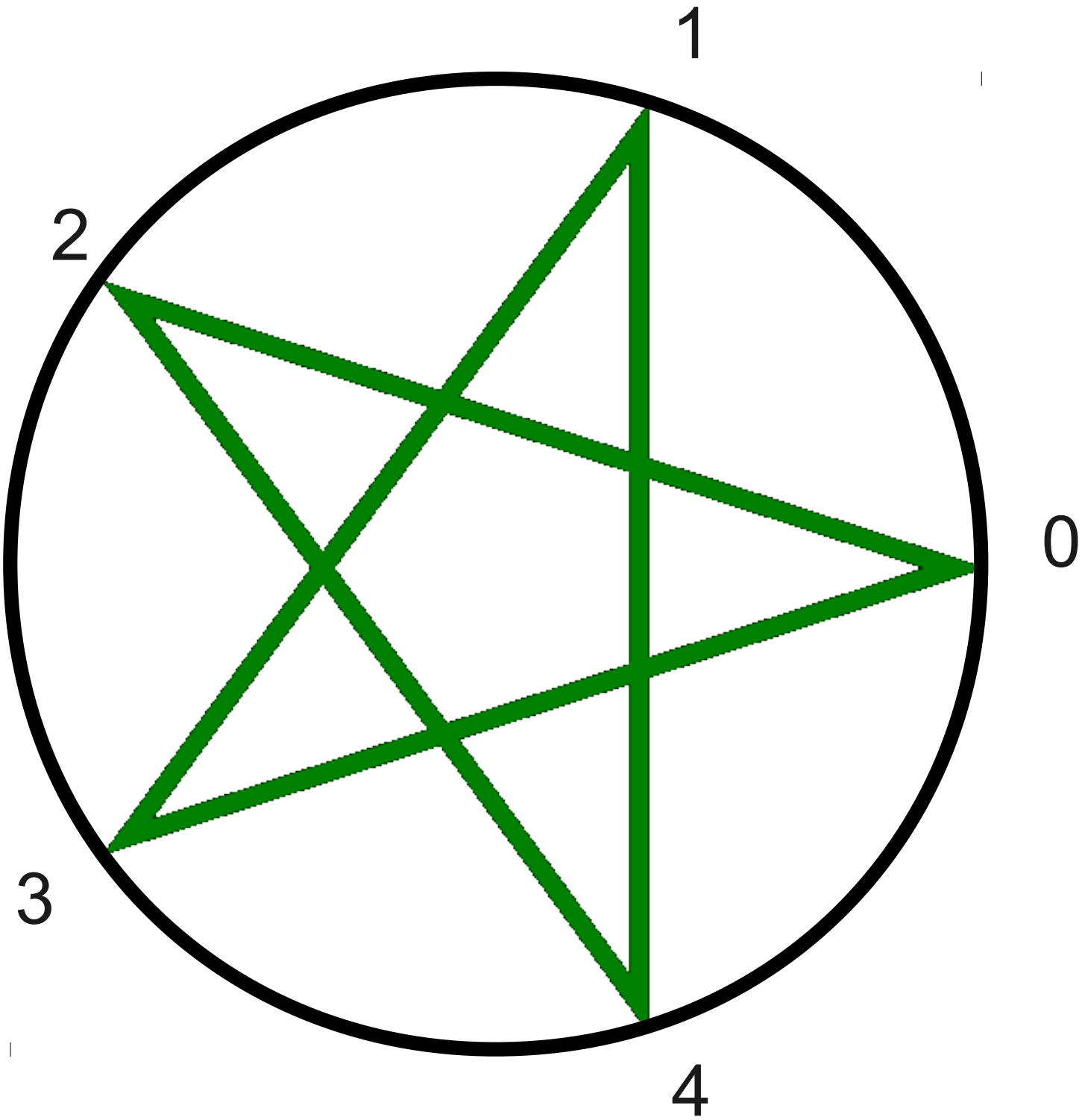
`for` versus `while`

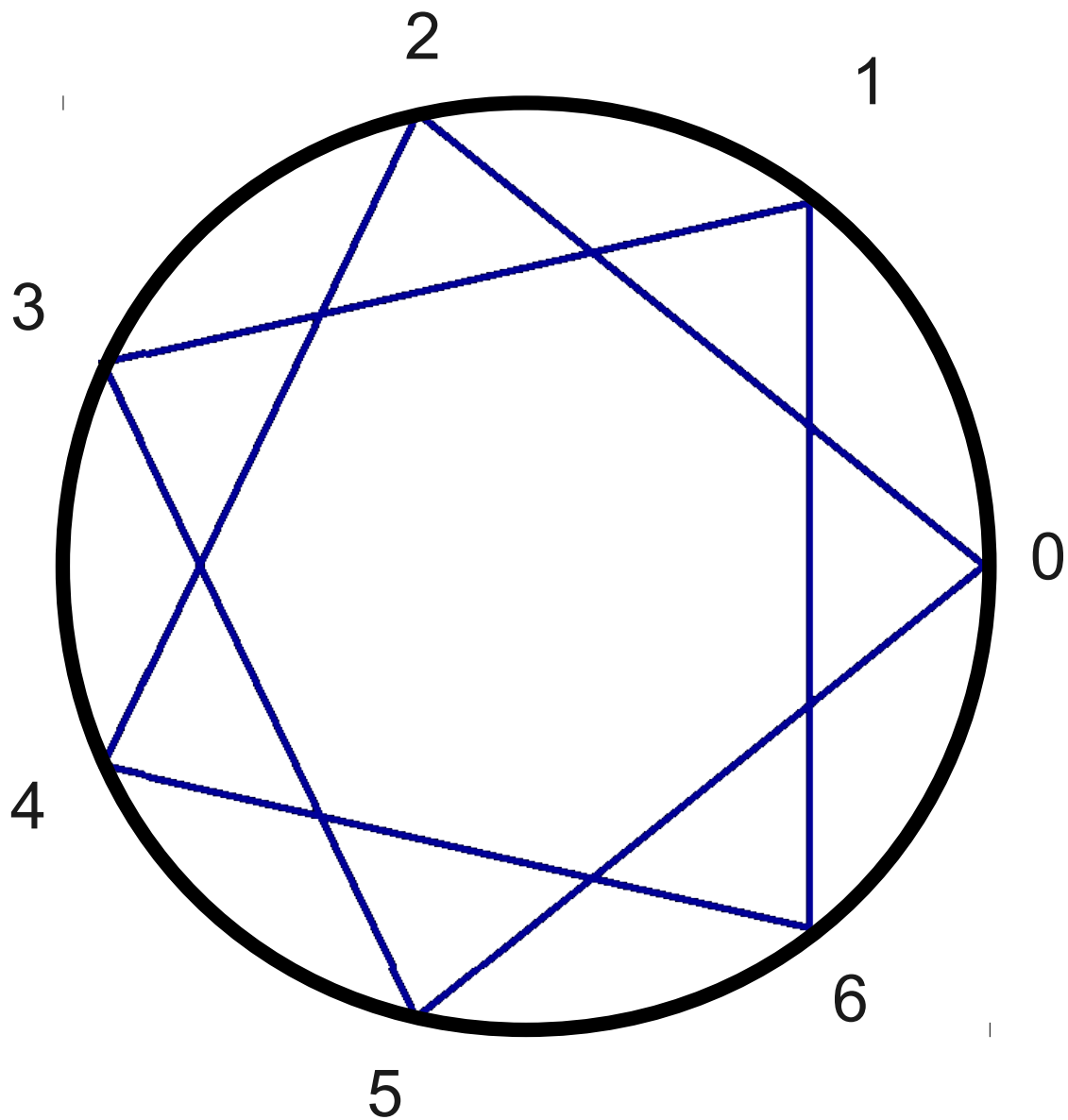
```
for (init ; test ; step) {
    statements
}
```

- **`for`** loop used for *definite* iteration.
- Generally, we know how many times we want to iterate.

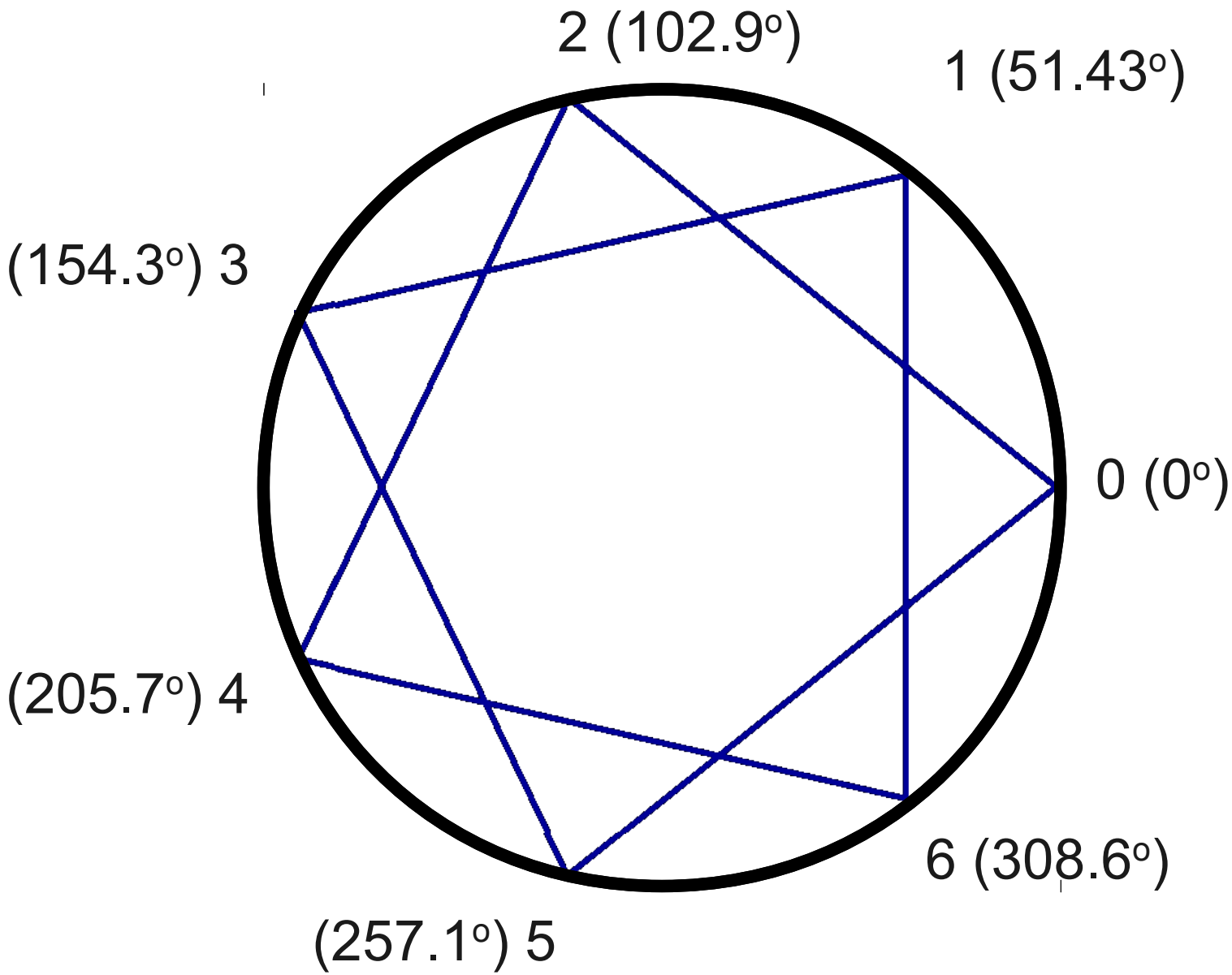
```
init
while ( test ) {
    statements
    step
}
```

- **`while`** loop used for *indefinite* iteration.
- Generally, don't know how many times to iterate beforehand.


1
2
0
3
4

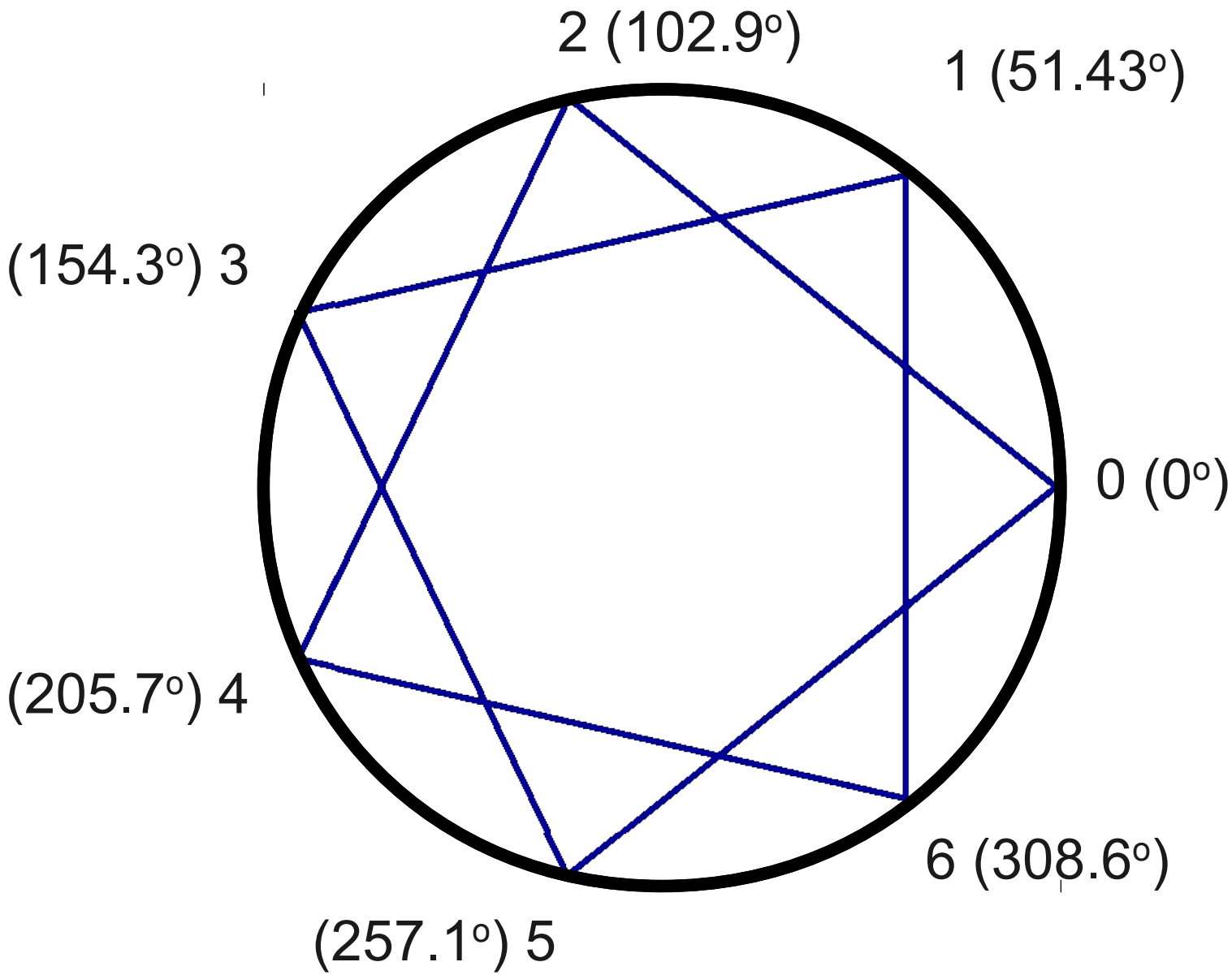


Each point k is connected to point k + 2, after wrapping around.



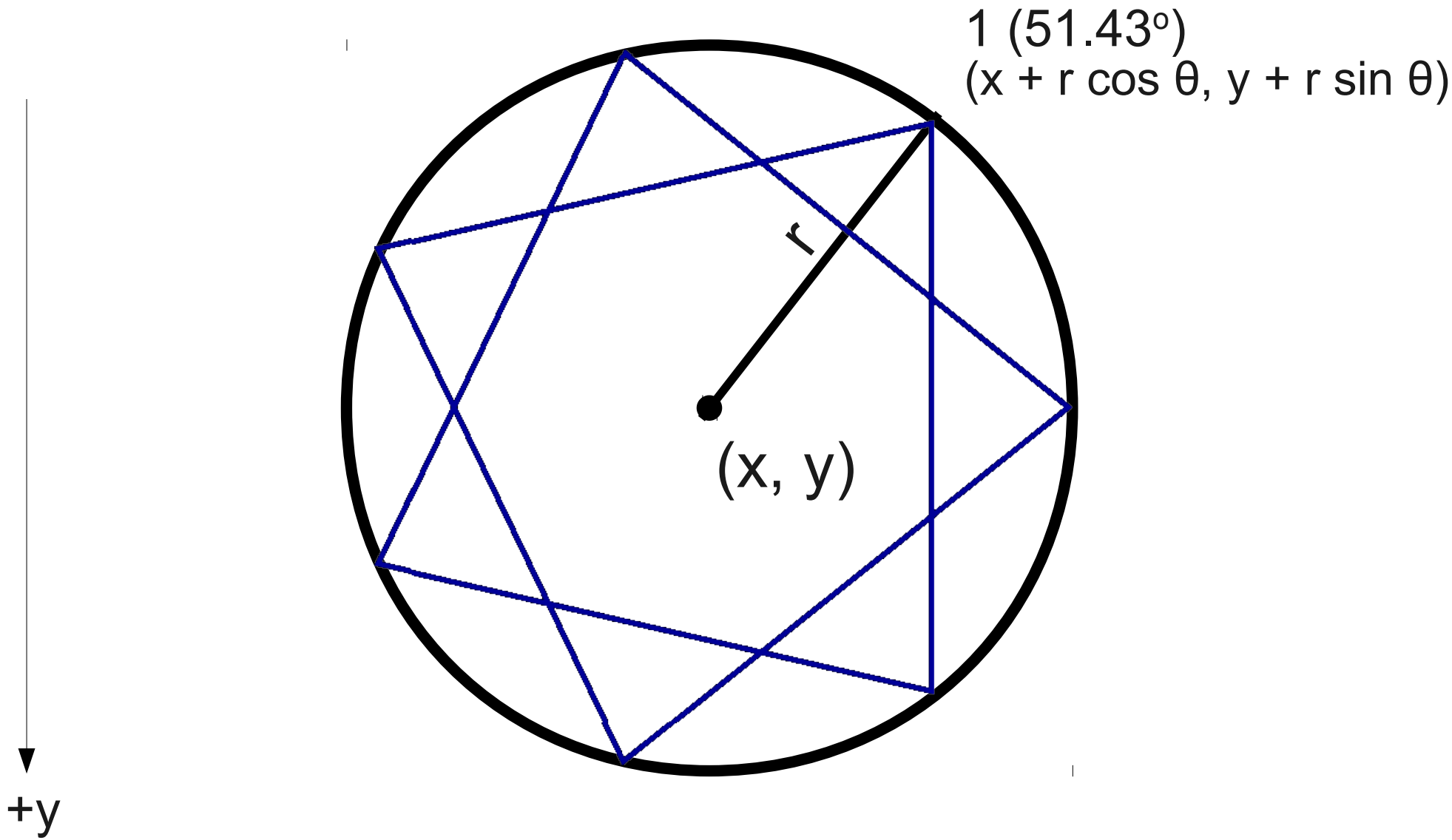
Each point k is connected to point k + 2, after wrapping around.

Point k is at $\frac{k}{numSides} \times 360^{\circ}$



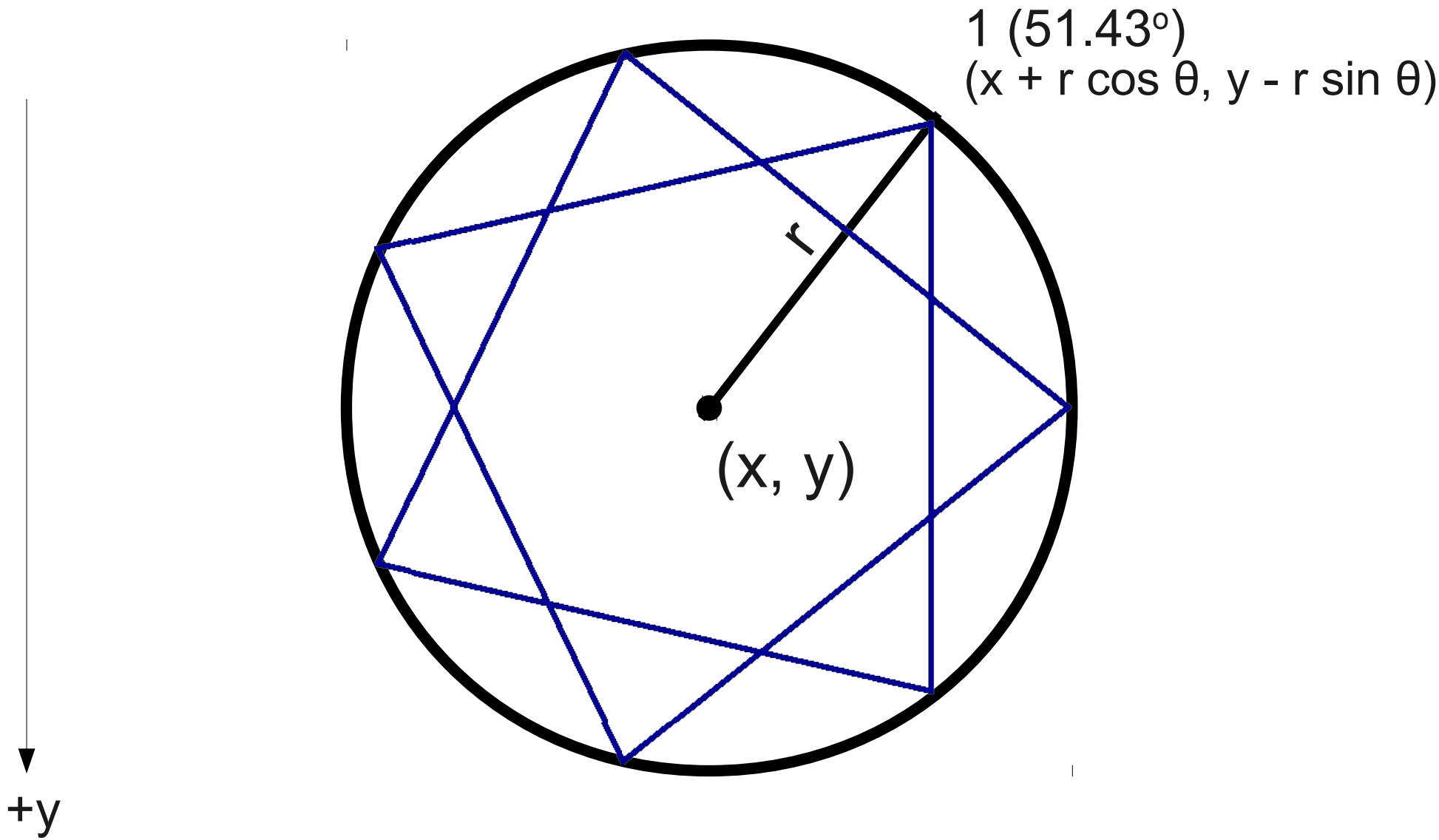
Each point k is connected to point k + 2, after wrapping around.

Point k is at $\frac{k}{numSides} \times 360^{\circ}$



Each point k is connected to point k + 2, after wrapping around.

Point k is at $\frac{k}{numSides} \times 360^{\circ}$



Each point k is connected to point k + 2, after wrapping around.

Point k is at $\frac{k}{numSides} \times 360^{\circ}$

Passing Parameters

- A method can accept **parameters** when it is called.
- Syntax:

```
private void name(parameters) {
    /* … method body … */
}
```

- The values of the parameters inside the method are set when the method is called.
- The values of the parameters can vary between calls.