Control Structures

Announcements

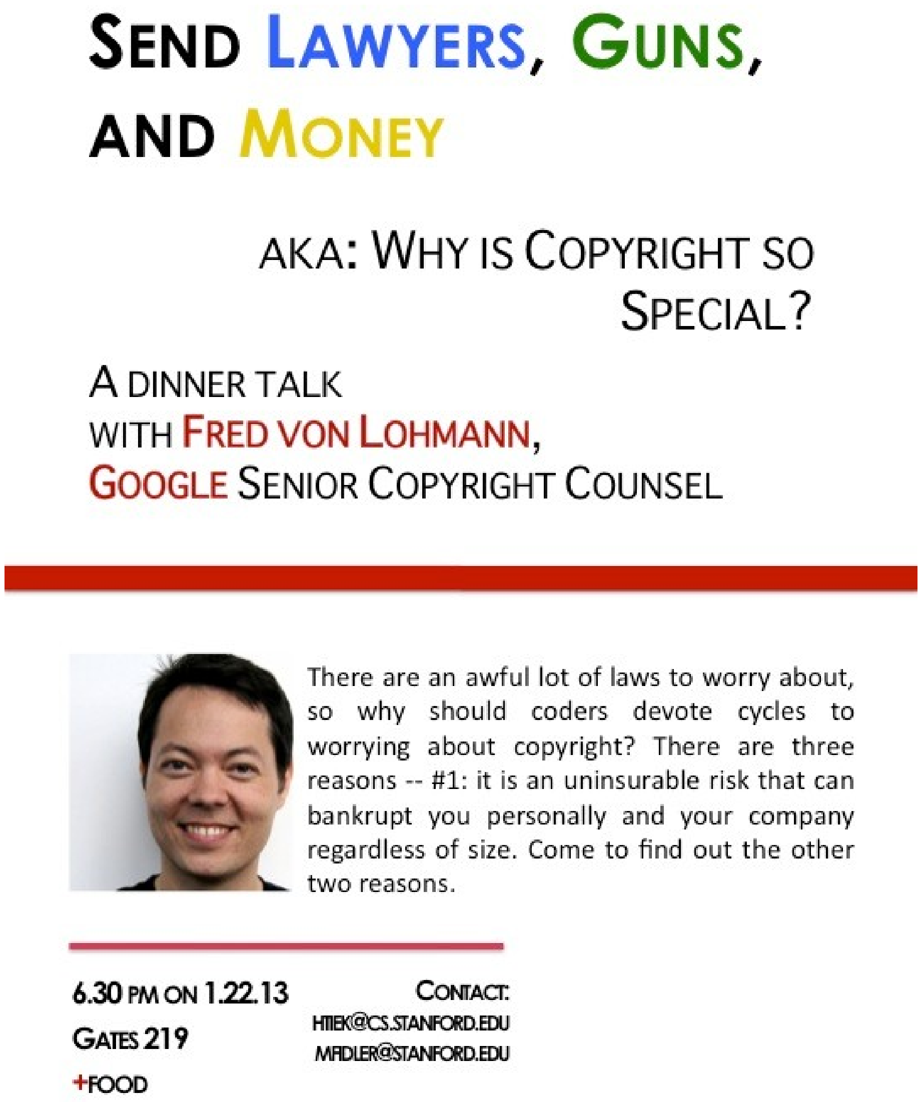
- Programming Assignment #1 due right now.
    - Due on next Wednesday if using a late day.
- Email due on Sunday night.
- Programming Assignment #2 out today, due Wednesday, January 30 at 3:15PM.
    - Play around with Java statements and control structures!
    - Make some pretty pictures!
    - Explore your creative potential!

Friday Four Square!
Today at 4:15PM, Outside Gates

Casual Dinner for Women in CS

- Next **Thursday, January 24** in Gates 219 at 6:00PM.
- Good food, great company, and everyone is invited!
- RSVP through email link (sent out yesterday).

Upcoming Talk



SEND LAWYERS, GUNS, AND MONEY

AKA: WHY IS COPYRIGHT SO SPECIAL?

A DINNER TALK
WITH FRED VON LOHMANN,
GOOGLE SENIOR COPYRIGHT COUNSEL

There are an awful lot of laws to worry about, so why should coders devote cycles to worrying about copyright? There are three reasons -- #1: it is an uninsurable risk that can bankrupt you personally and your company regardless of size. Come to find out the other two reasons.

6.30 PM ON 1.22.13
GATES 219
+FOOD

CONTACT:
HTIEK@CS.STANFORD.EDU
MFIDLER@STANFORD.EDU

- Fred von Lohmann, Senior Copyright Counsel, will be giving a talk next **Tuesday** at **6:30PM** in **Gates 219**.
- Everyone is welcome!

Recap From Last Time

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

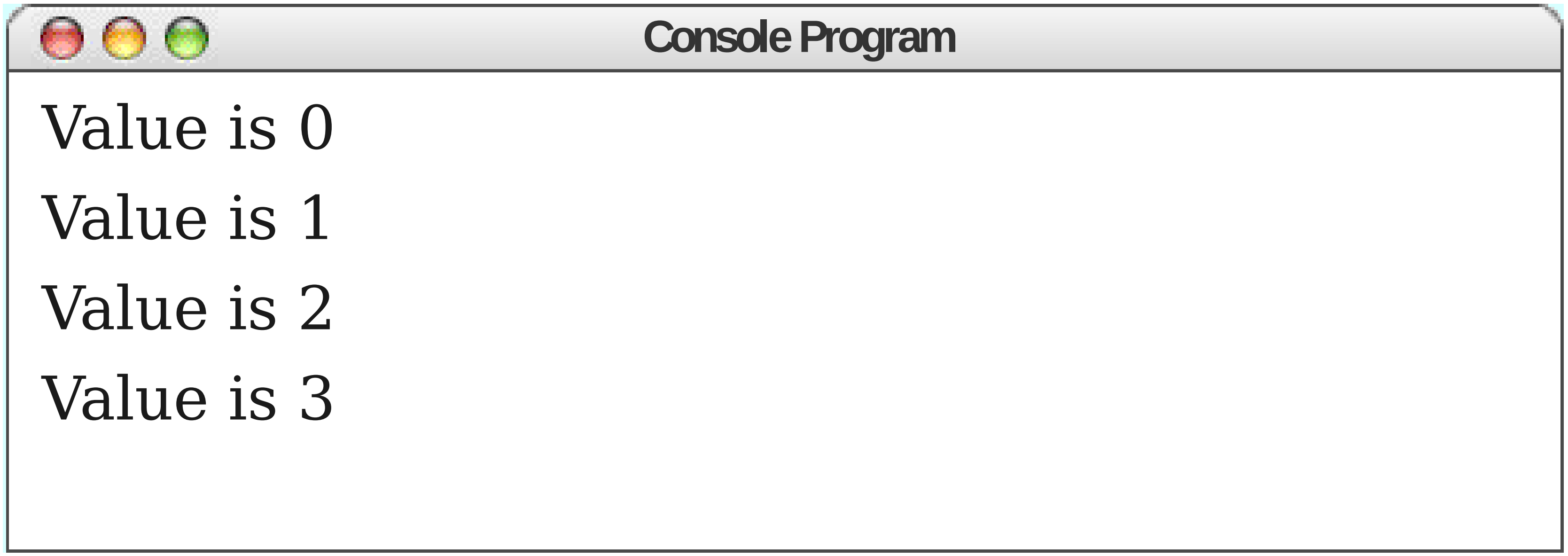
This is called the step or increment and is performed at the end of each loop iteration.

```
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.

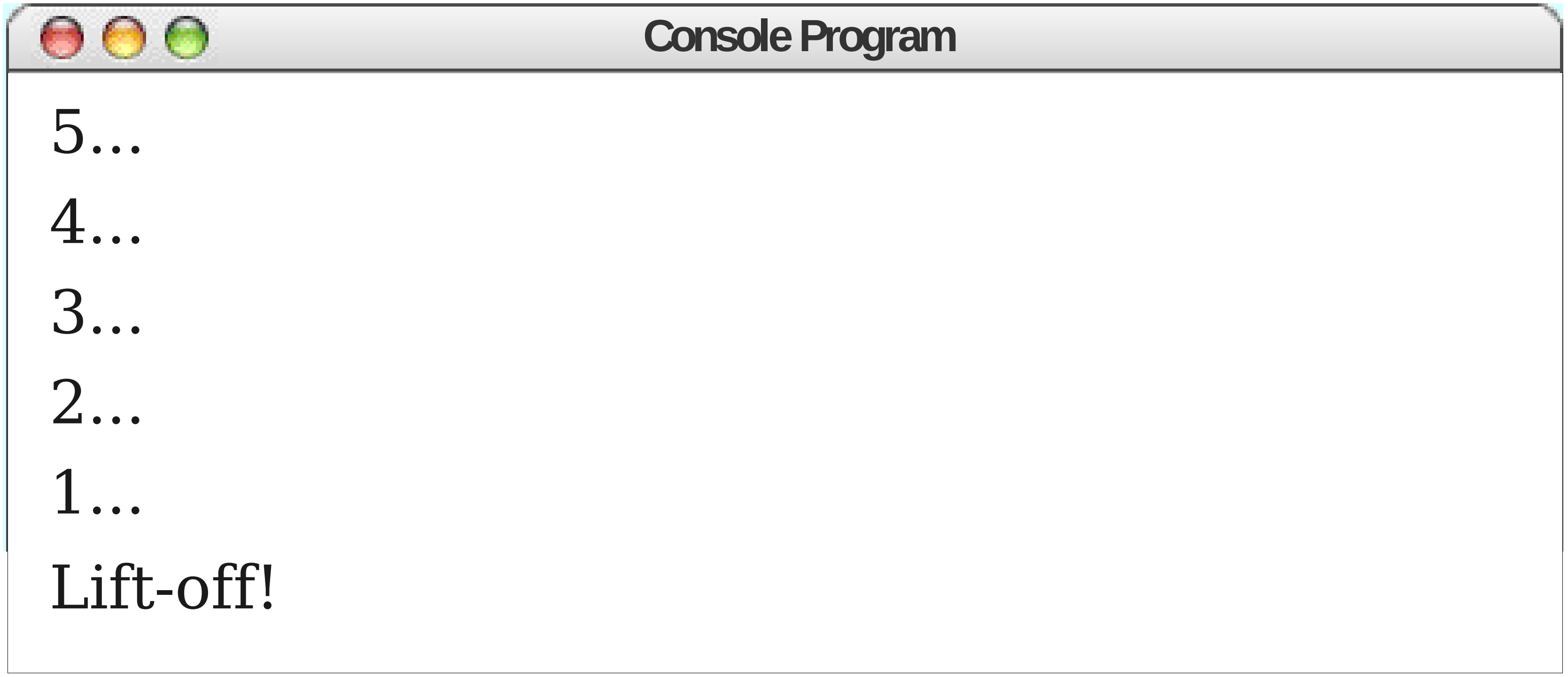
Accessing the Loop Counter

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



```
Value is 0
Value is 1
Value is 2
Value is 3
```

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



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

Video: Apollo 11 Launch

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!");
```

Control Statements

```
for
if
while
```

Control Statements

```
for
if
while
```

`if` statement

```
if (condition) {
… statements to run if condition holds …
}
```

```java
public void run() {
    /* Do the launch countdown! */
    for (int i = 30; i > 0; i--) {
        println("T-" + i + " seconds.");

        /* Specific mission commands. */
        if (i == 15) {
            println("Guidance is internal.");
        }
        if (i == 9) {
            println("Ignition sequence start.");
        }
    }

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

```
public void run() {
    /* Do the launch countdown! */
    for (int i = 30; i > 0; i--) {
        println("T-" + i + " seconds.");

        /* Specific mission commands. */
        if (i == 15) {
            println("Guidance is internal.");
        }
        if (i == 9) {
            println("Ignition sequence start.");
        }
    }

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

```
public void run() {
    /* Do the launch countdown! */
    for (int i = 30; i > 0; i--) {
        println("T-" + i + " seconds.");

        /* Specific mission commands. */
        if (i == 15) {
            println("Guidance is internal.");
        }
        if (i == 9) {
            println("Ignition sequence start.");
        }
    }

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

```
public void run() {
    /* Do the launch countdown! */
    for (int i = 30; i > 0; i--) {
        println("T-" + i + " seconds.");

        /* Specific mission commands. */
        if (i == 15) {
            println("Guidance is internal.");
        }
        if (i == 9) {
            println("Ignition sequence start.");
        }
    }

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

Magic Numbers

- A **magic number** is a number written in a piece of code whose meaning cannot easily be deduced from context.

```
double weight = 9.8 * (mass - 14.3);
```

- Magic numbers are a Bad Thing; they make code harder to read.

Constants

- Not all variables actually *vary*.
- A **constant** is a name for a value that never changes.
- Syntax (defined outside of any method):

```
private static final type name = value;
```

- By convention, constants are named in **`UPPER_CASE_WITH_UNDERSCORES`** to differentiate them from variables.

I'M SORRY, DR. WEINER. WE'RE JUST NOT THAT IMPRESSED.
$W_c = 2$

The Mathematics Council roundly rejected my proposed Weiner's Constant.

(This DESPITE the fact that you can average any two numbers by dividing by Weiner's Constant.)

(This DESPITE the fact that you can average any two numbers by dividing by Weiner's Constant.)

General Rules for Constants

- You can usually use 0 and 1 in loops without needing constants.
- When computing averages, it's fine to just use the number 2.
- Anything more complex than this should probably be made into a constant.

Or `else`

```
if (condition) {
… statements to run if condition holds …
} else {
… statements to run if condition doesn't hold …
}
```

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(" (╯°□°）╯︵ ┻━┻ ");
}
```

Control Statements

```
for
if
while
```

Control Statements

```
for
if
while
```

The **`while`** Loop

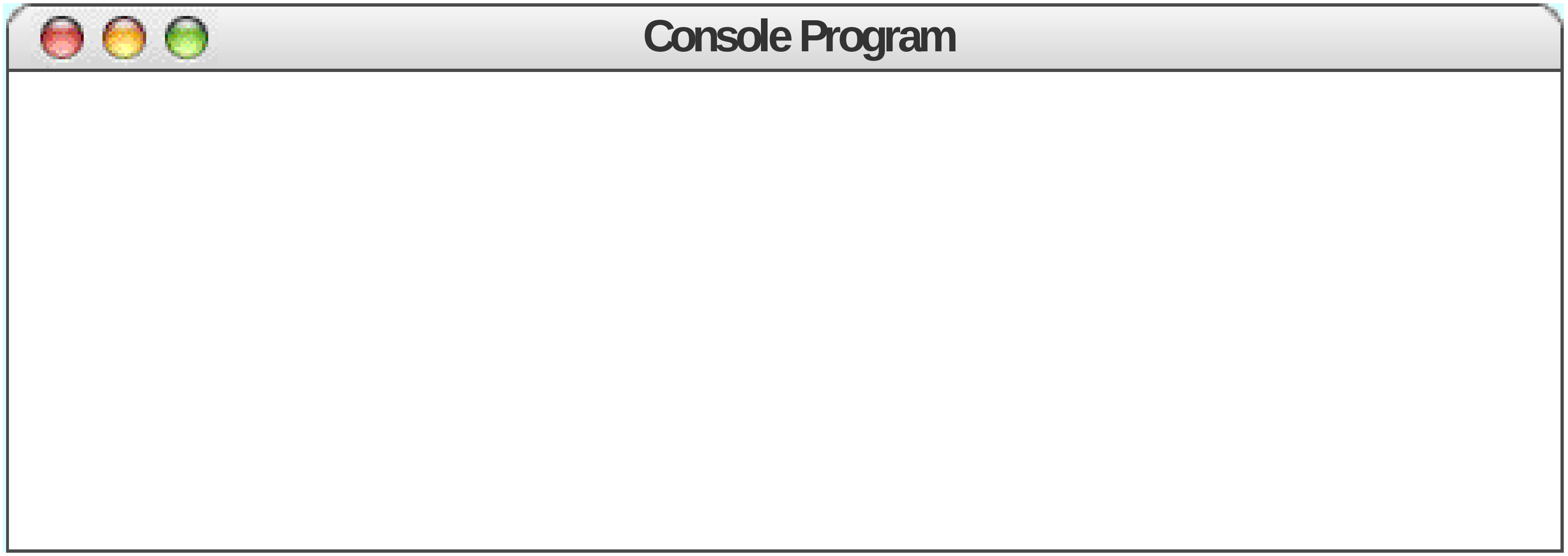
```
while (condition) {
      ... statements ...
}
```

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

`while` loop

Example:

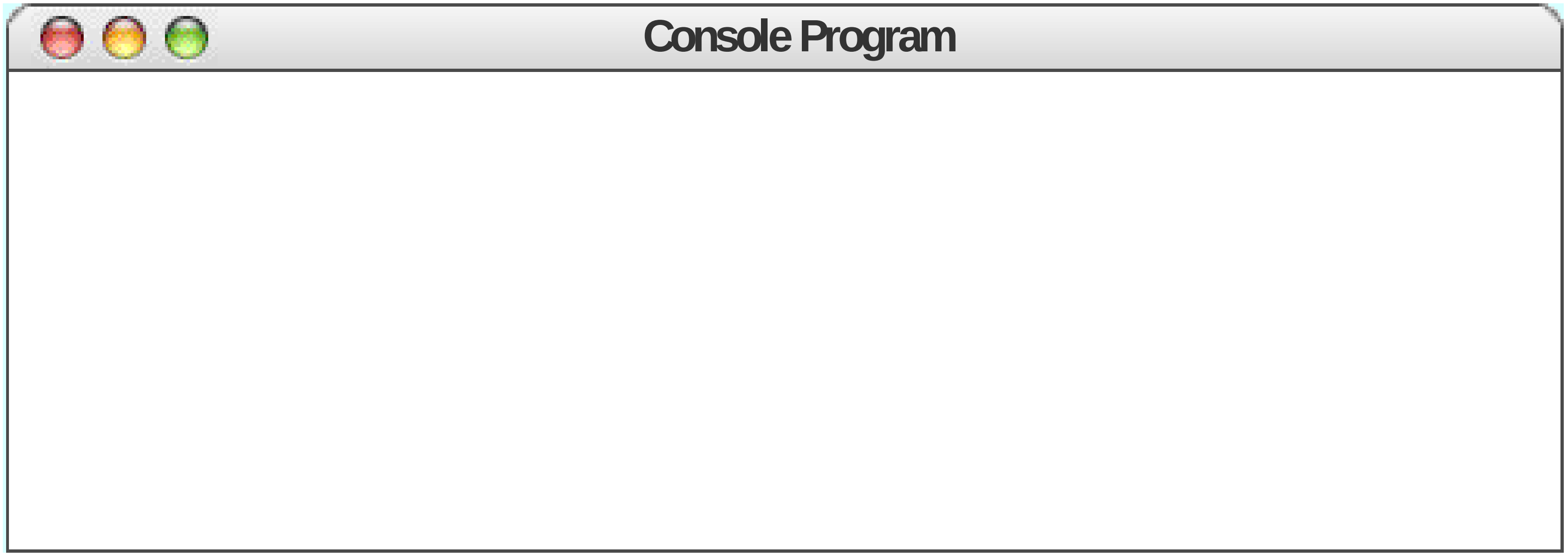
```
int x = 15;
while (x > 1) {
    x /= 2;
    println(x);
}
```



`while` loop

Example:

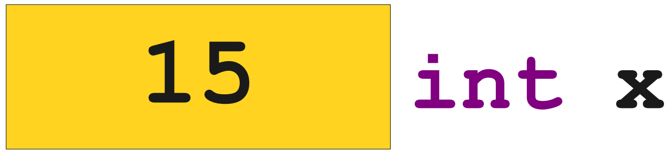
```
int x = 15;
while (x > 1) {
    x /= 2;
    println(x);
}
```

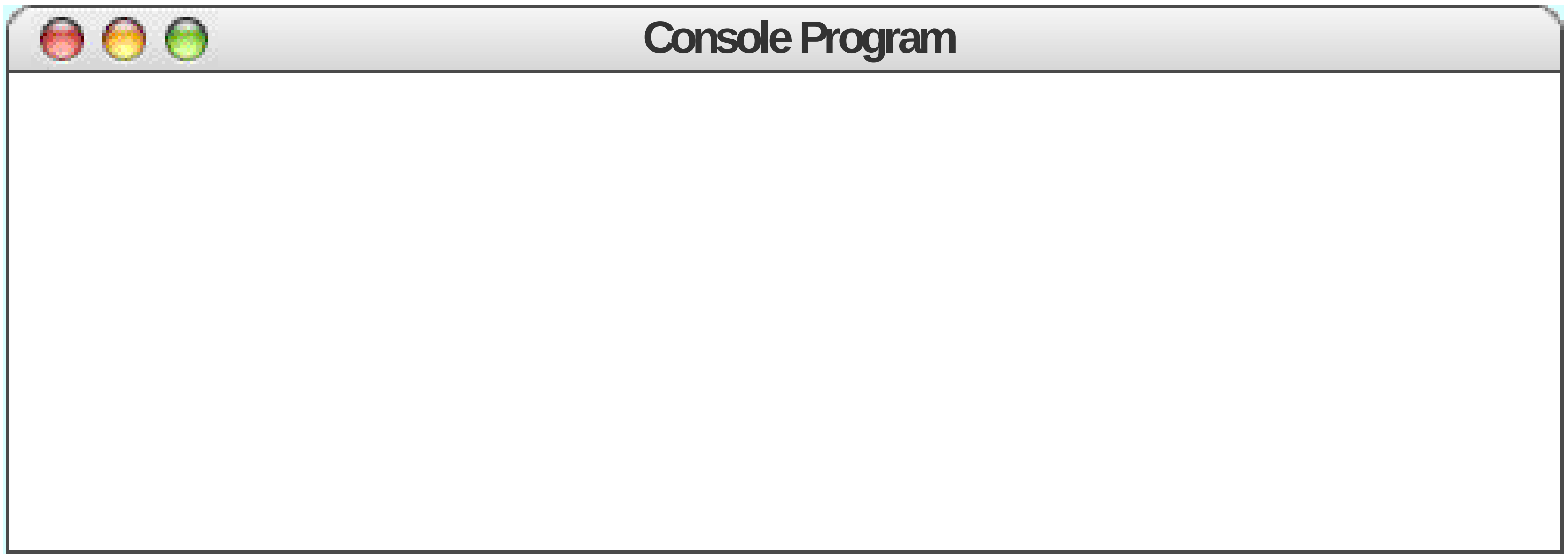


`while` loop

Example:

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

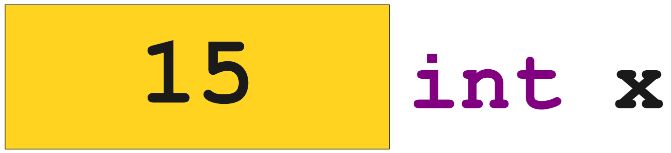


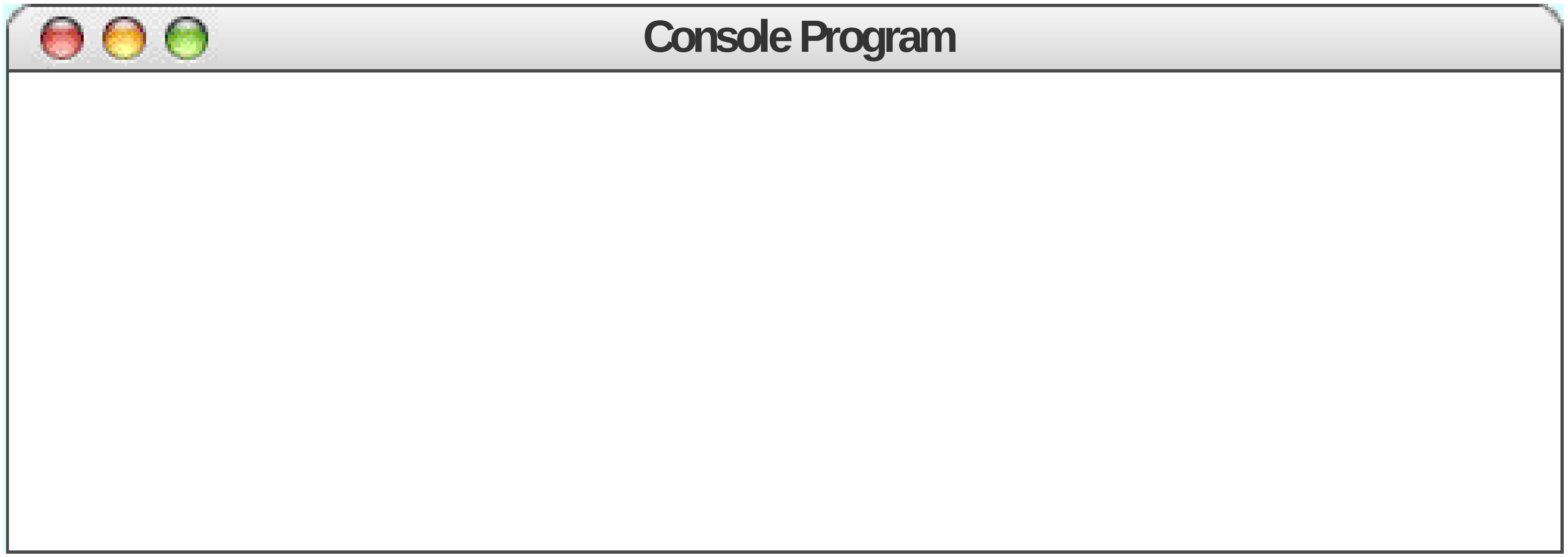


`while` loop

Example:

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

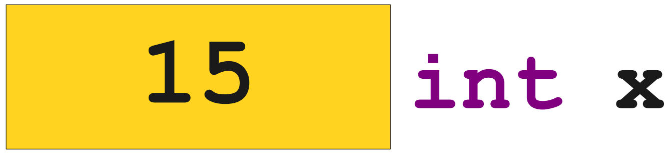


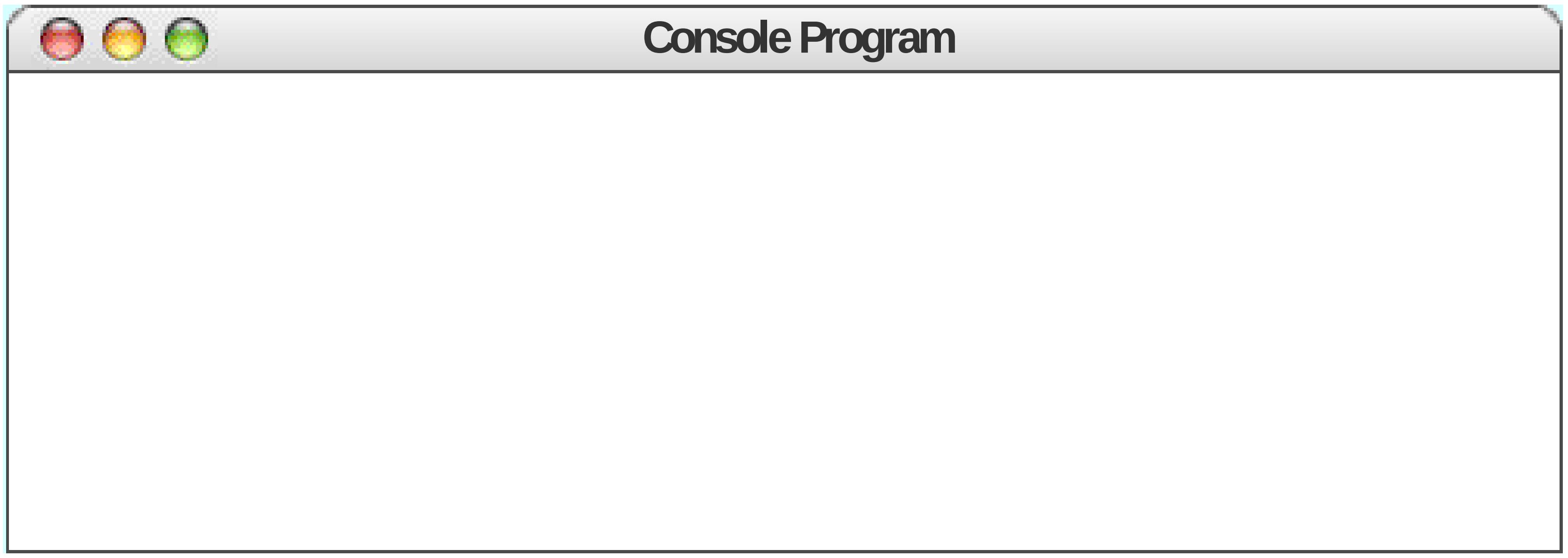


while loop

Example:

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

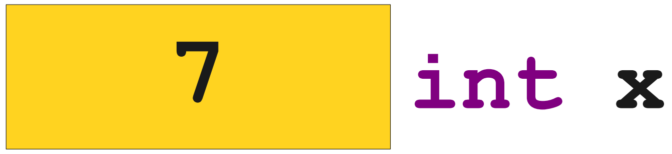


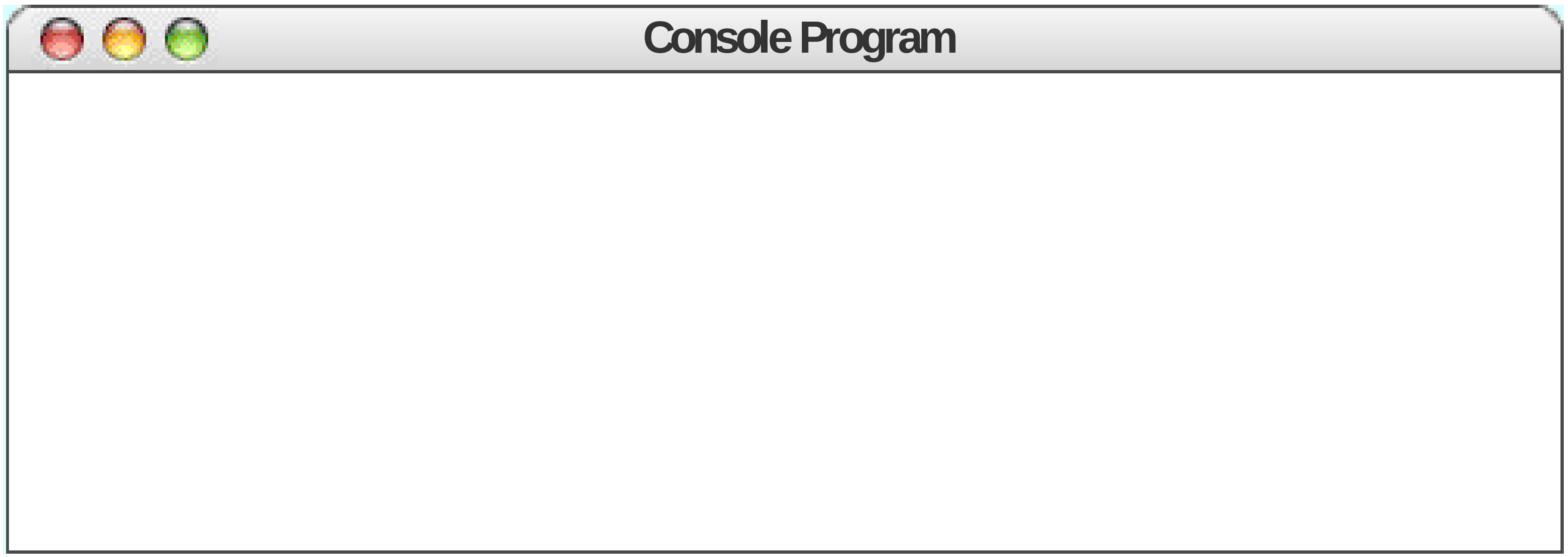


`while` loop

Example:

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

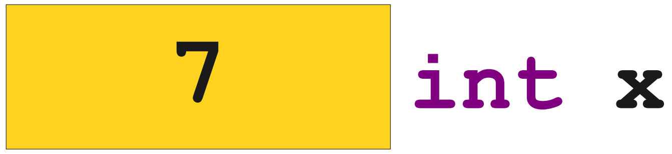


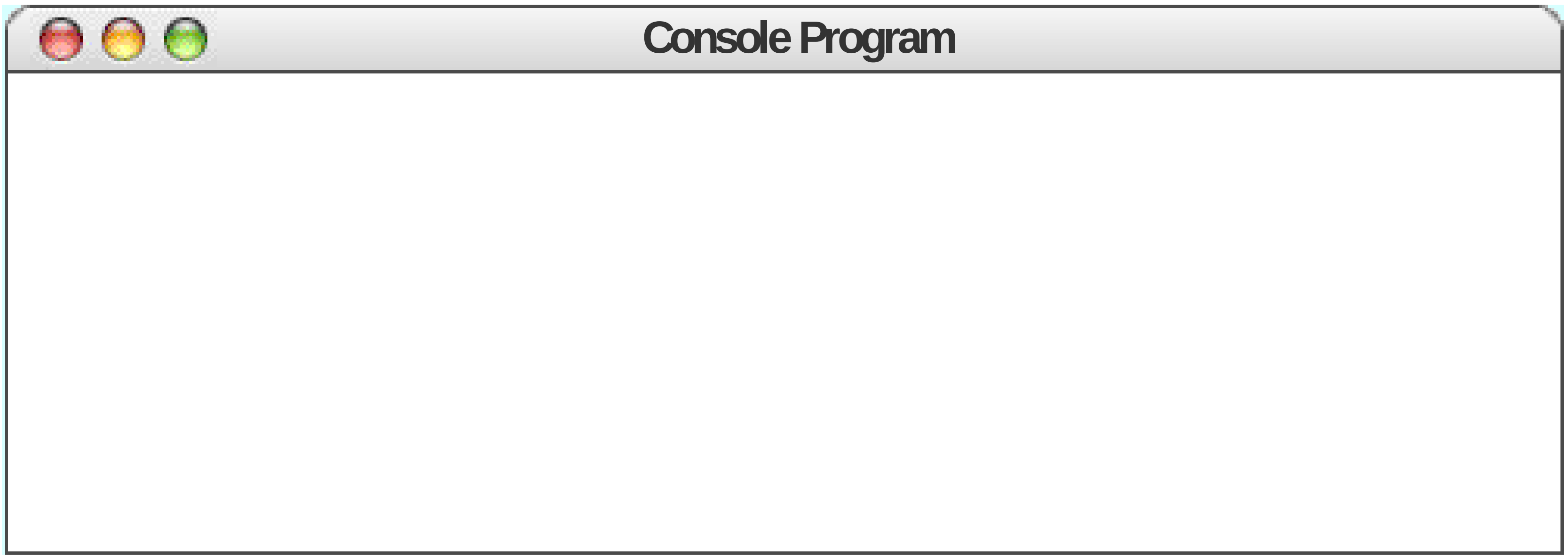


while loop

Example:

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

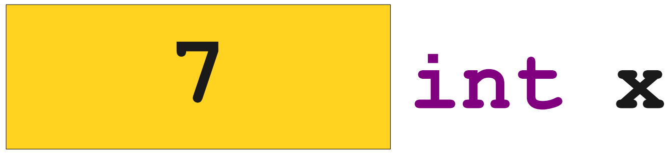


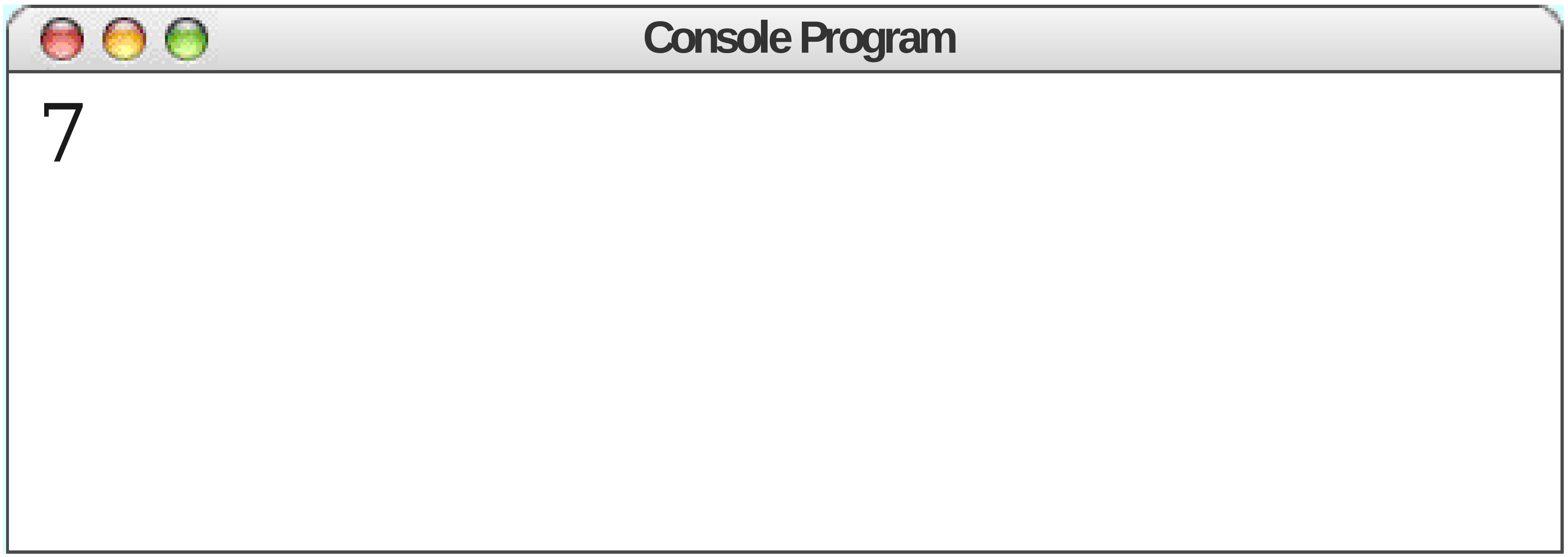


`while` loop

Example:

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



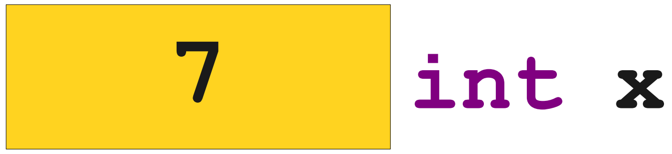


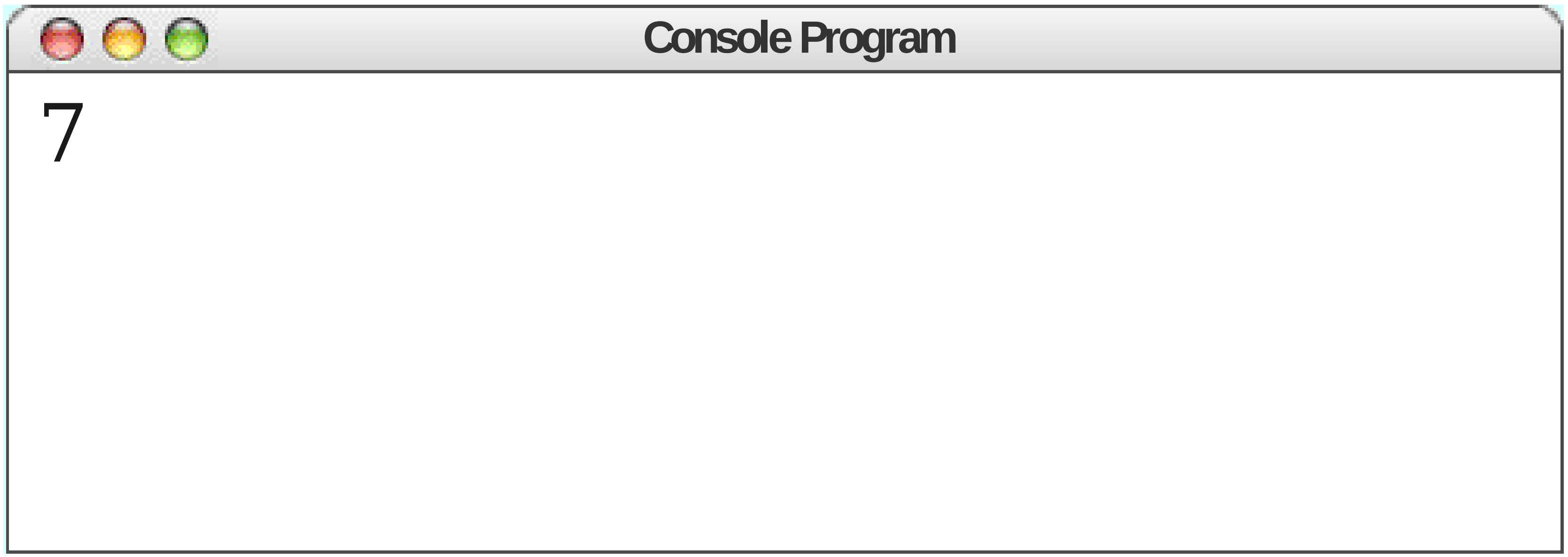
```
7
```

`while` loop

Example:

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



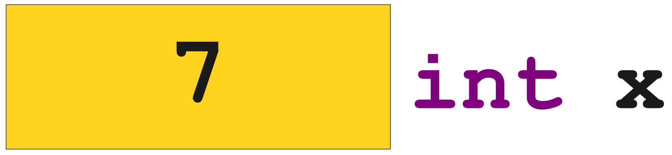


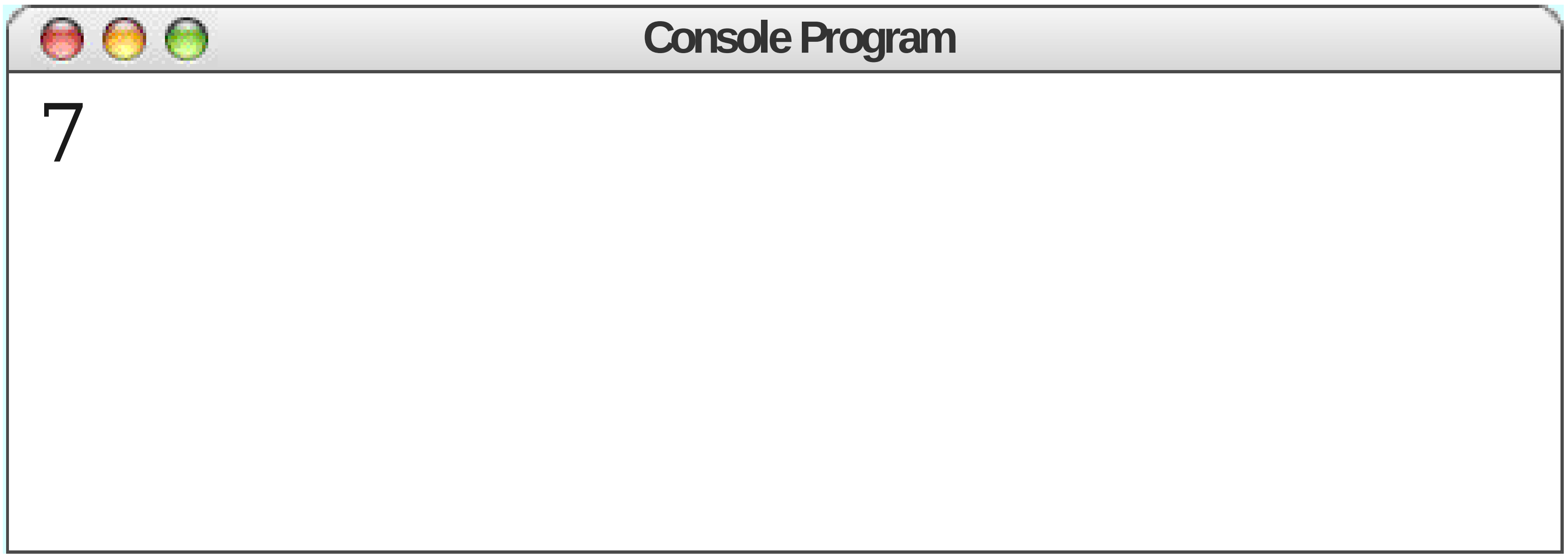
```
7
```

`while` loop

Example:

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



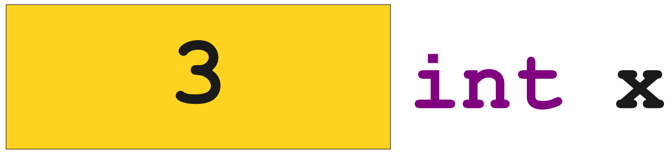


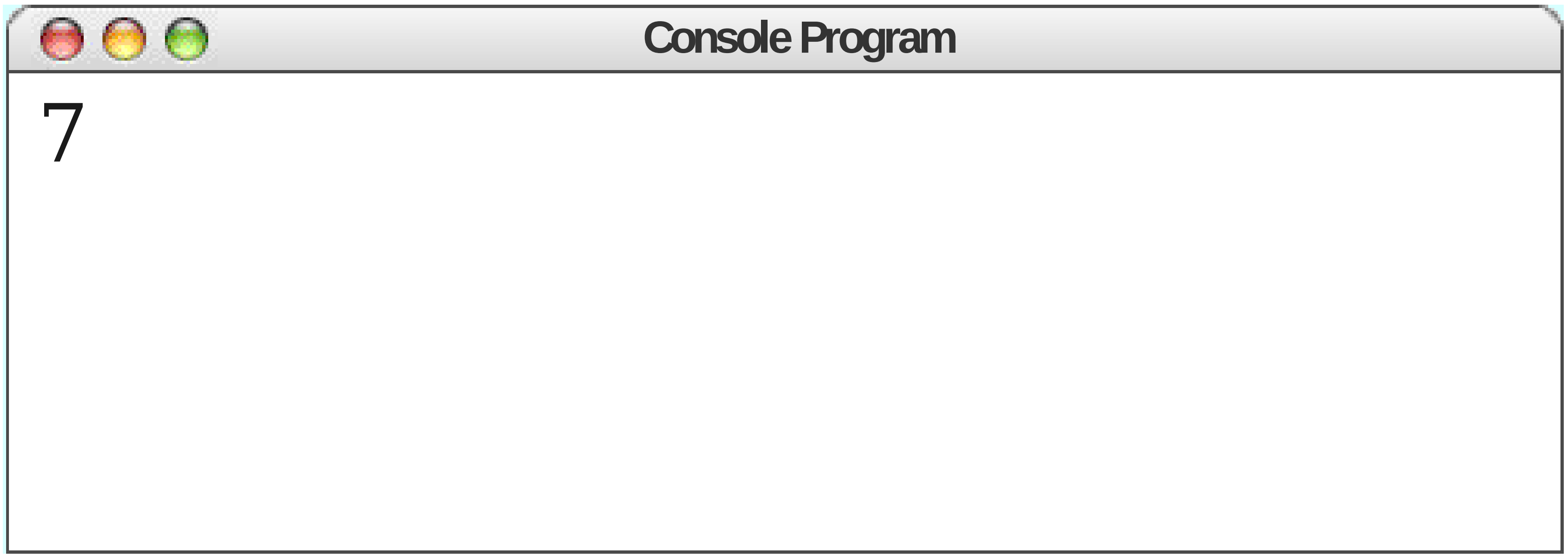
```
7
```

`while` loop

Example:

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





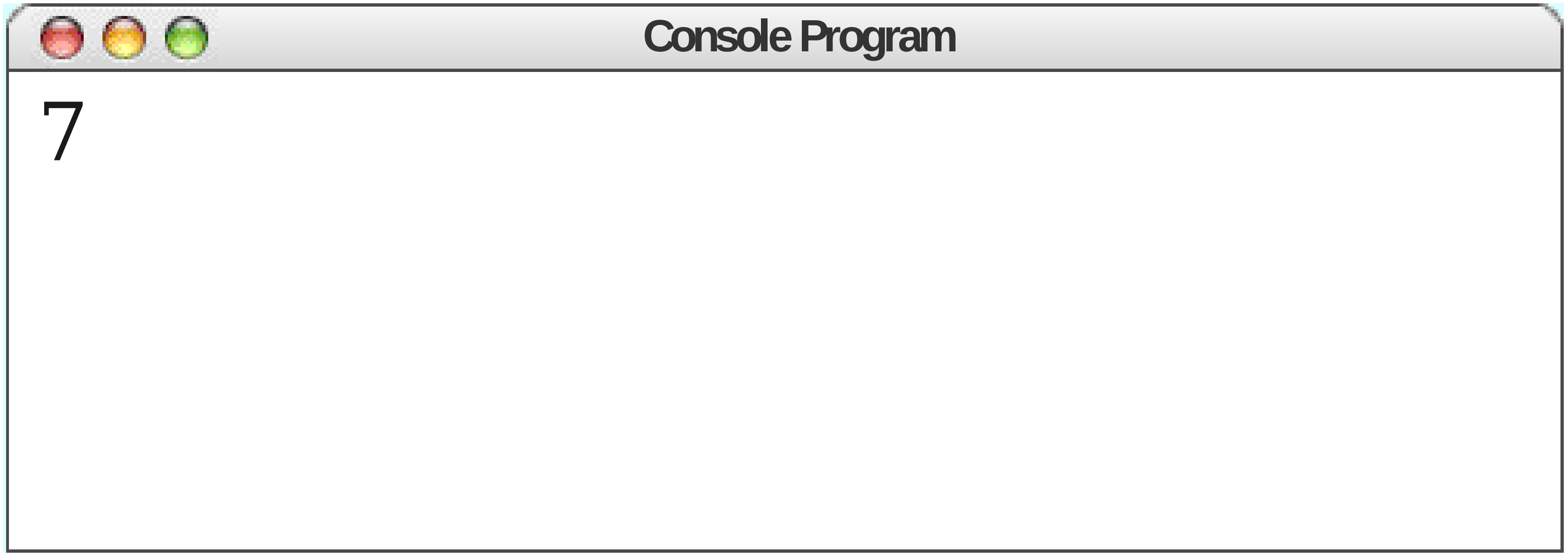
```
7
```

`while` loop

Example:

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

3 `int x`

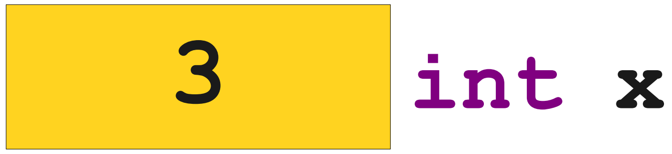

Console Program

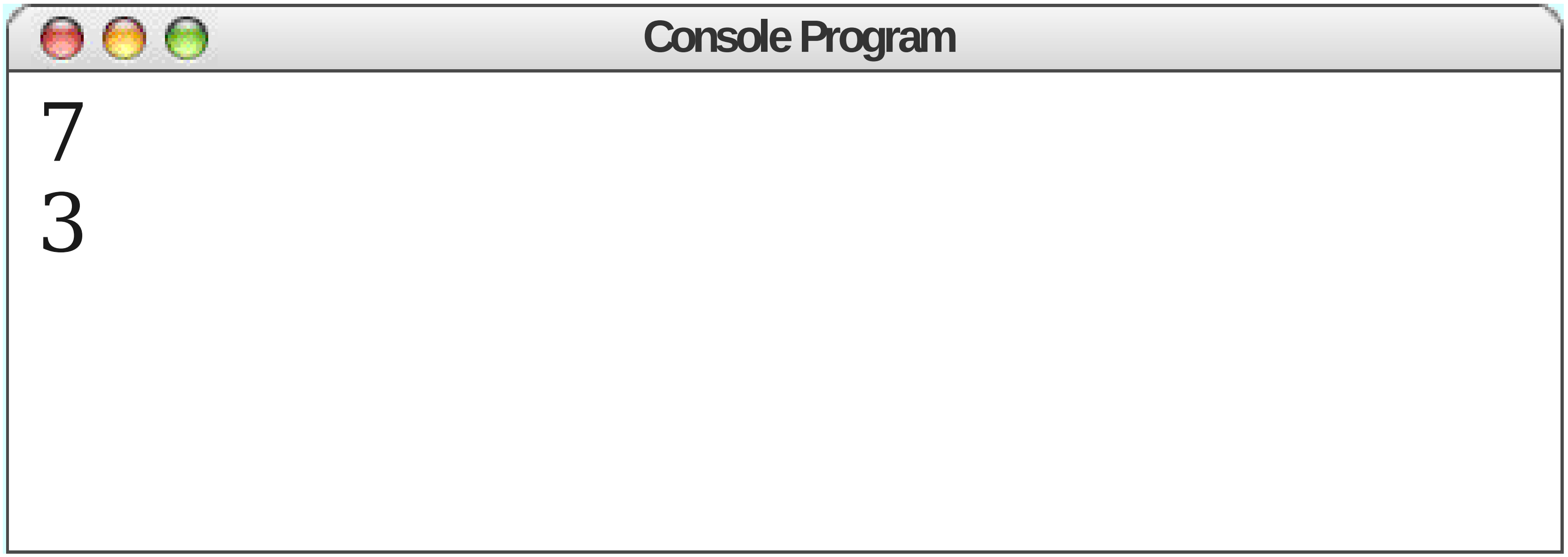
```
7
```

`while` loop

Example:

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



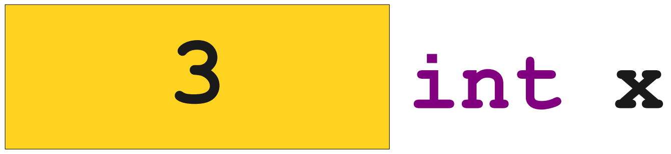


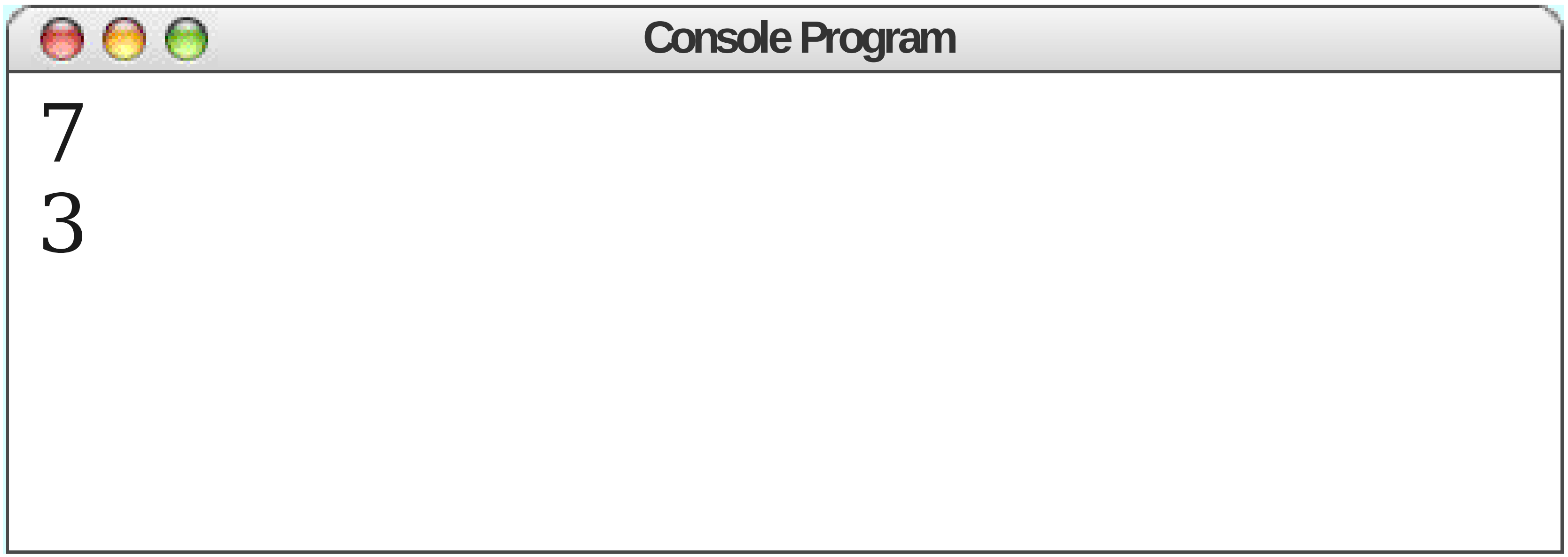
```
7
3
```

`while` loop

Example:

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



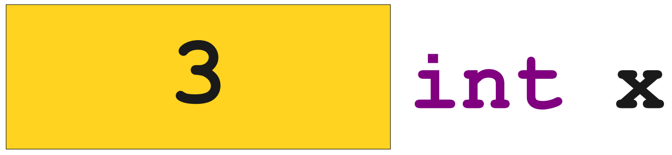


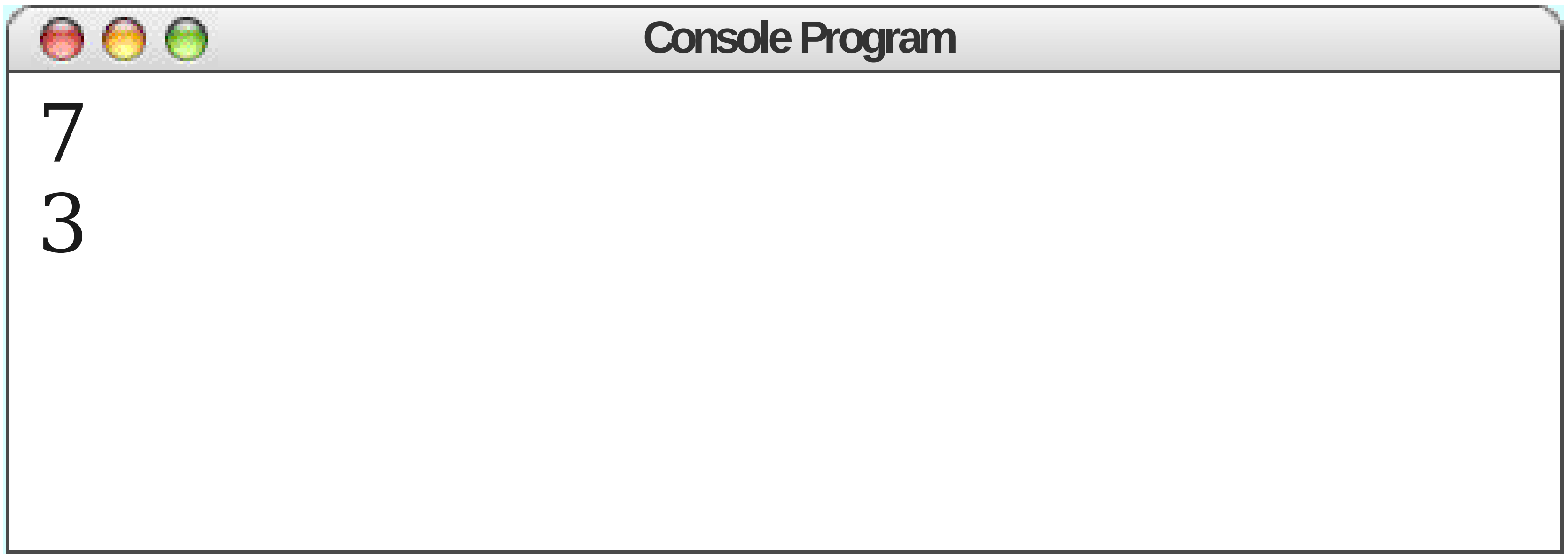
```
7
3
```

while loop

Example:

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





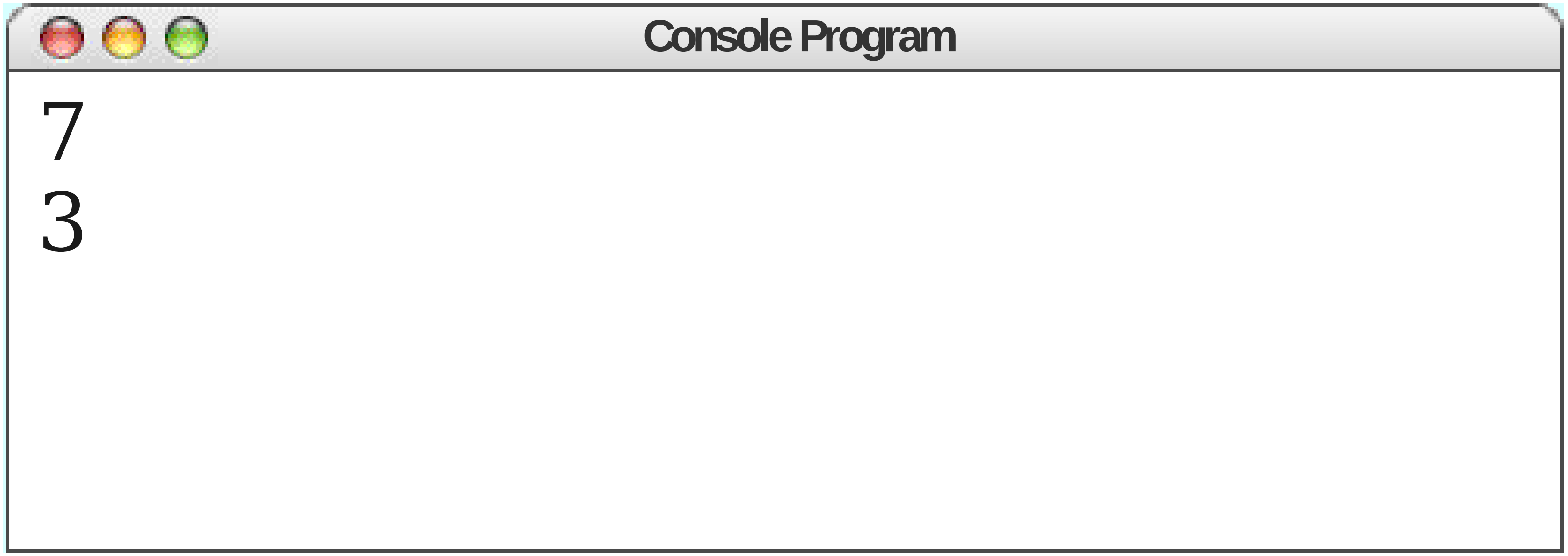
```
7
3
```

`while` loop

Example:

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

$\boxed{1}$ `int x`



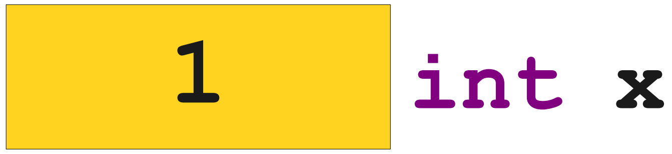
```
7
3
```

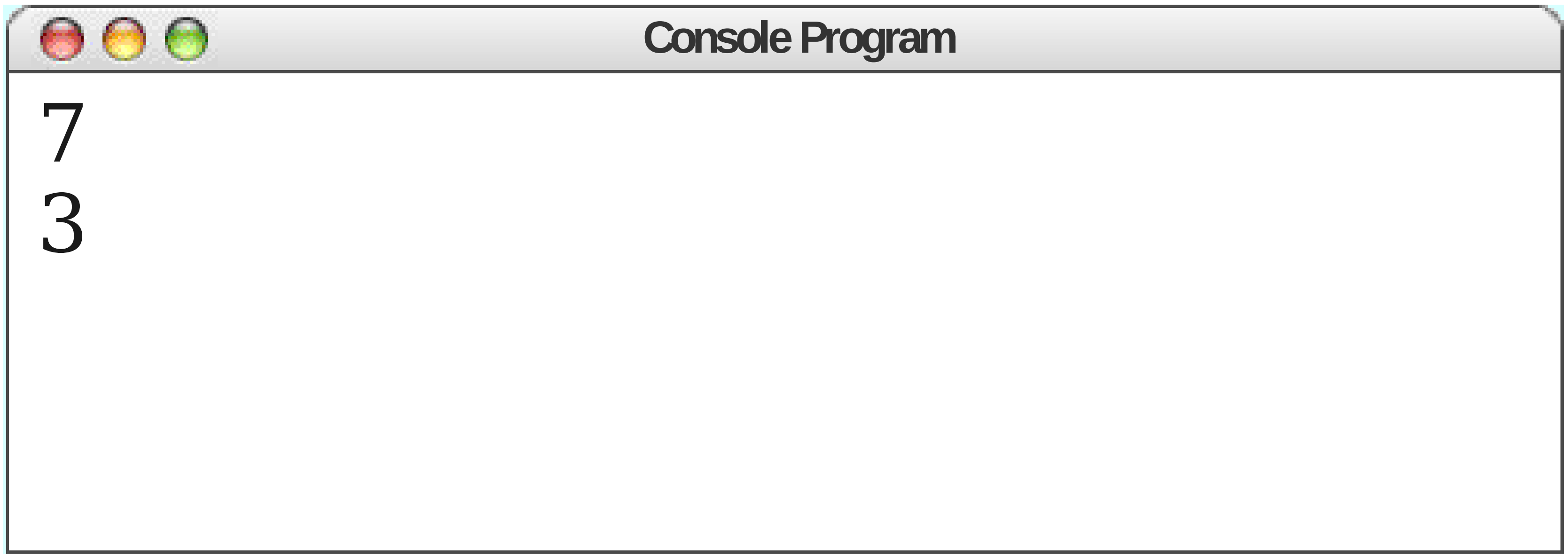
Based on slides by Mehran Sahami

`while` loop

Example:

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



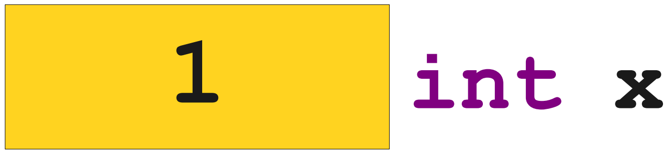


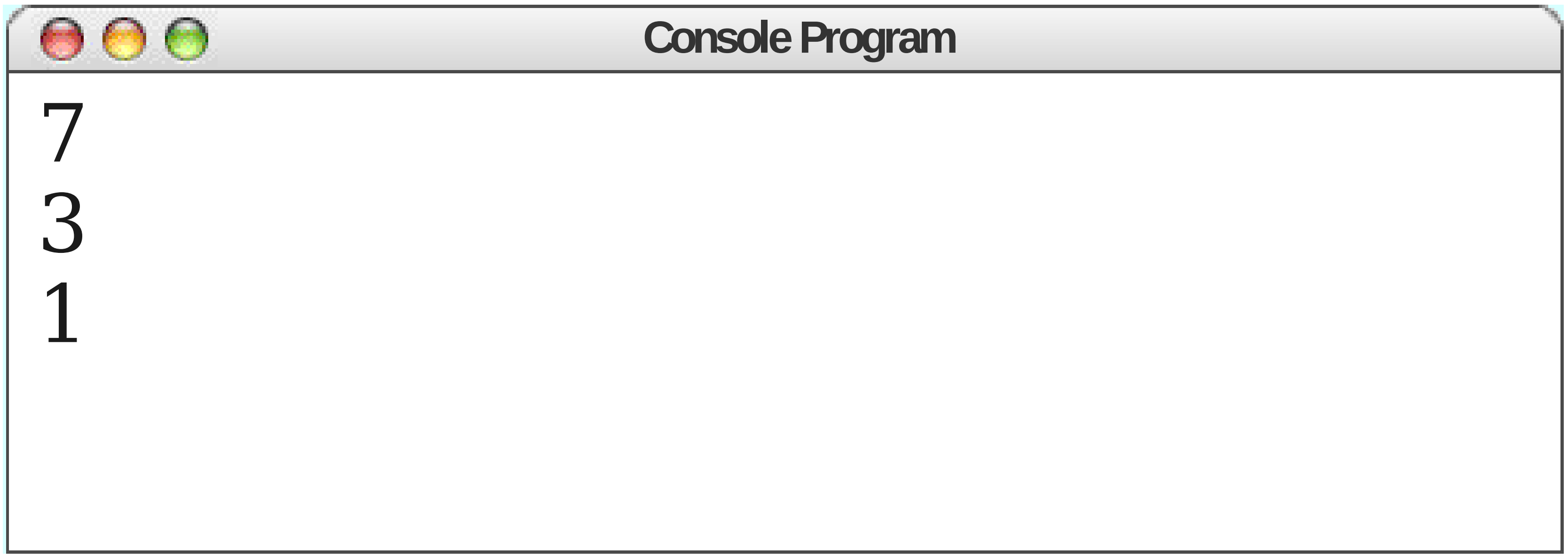
```
7
3
```

while loop

Example:

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



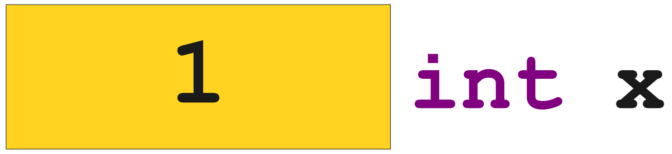


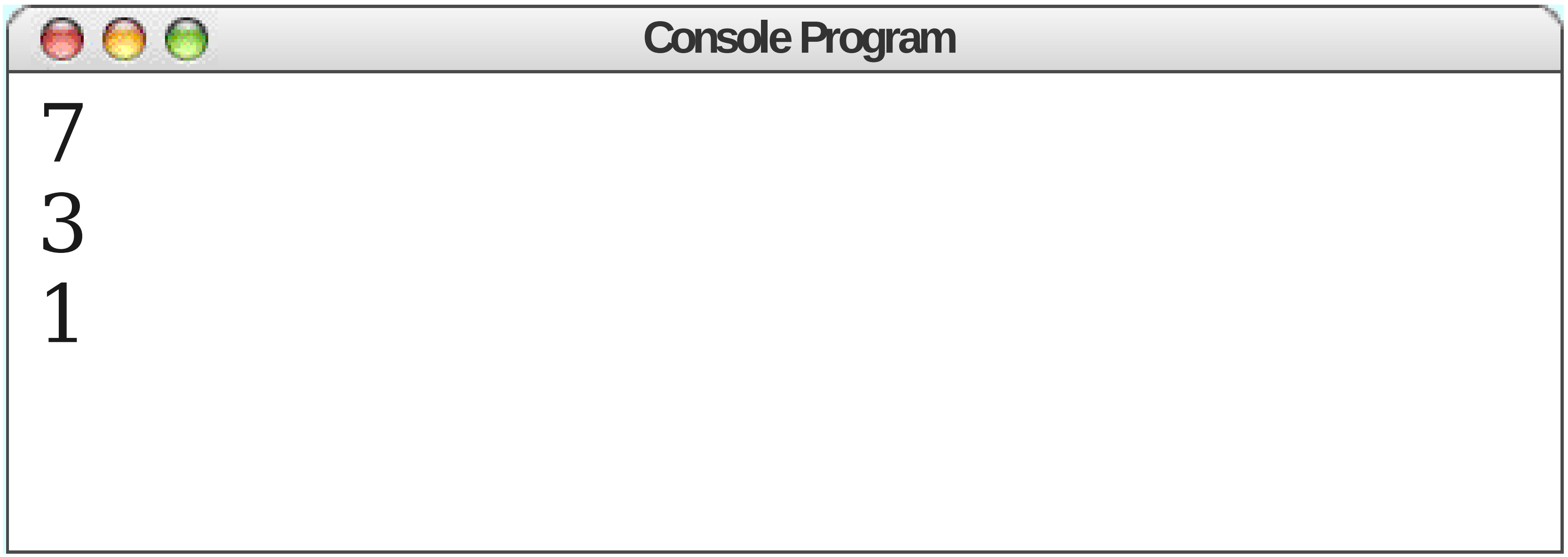
```
7
3
1
```

`while` loop

Example:

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



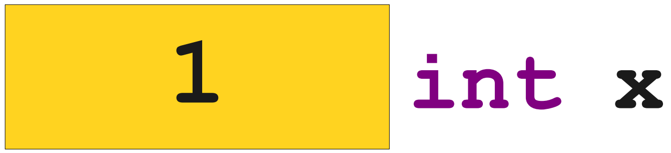


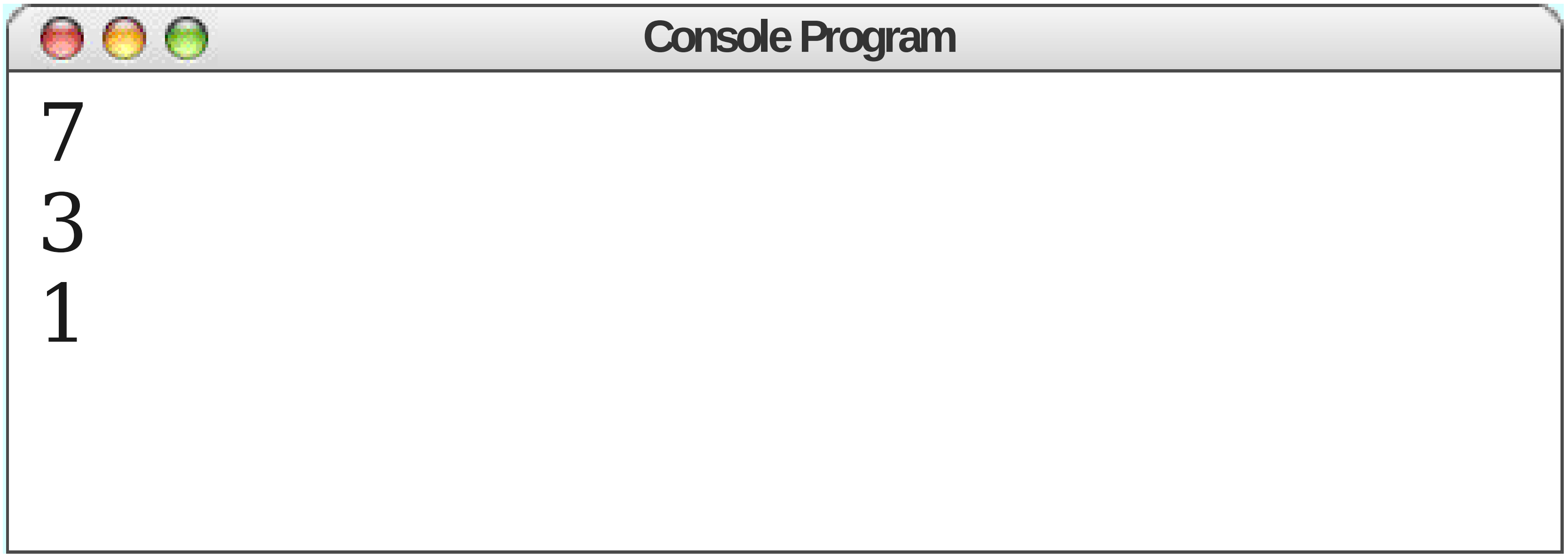
```
7
3
1
```

`while` loop

Example:

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



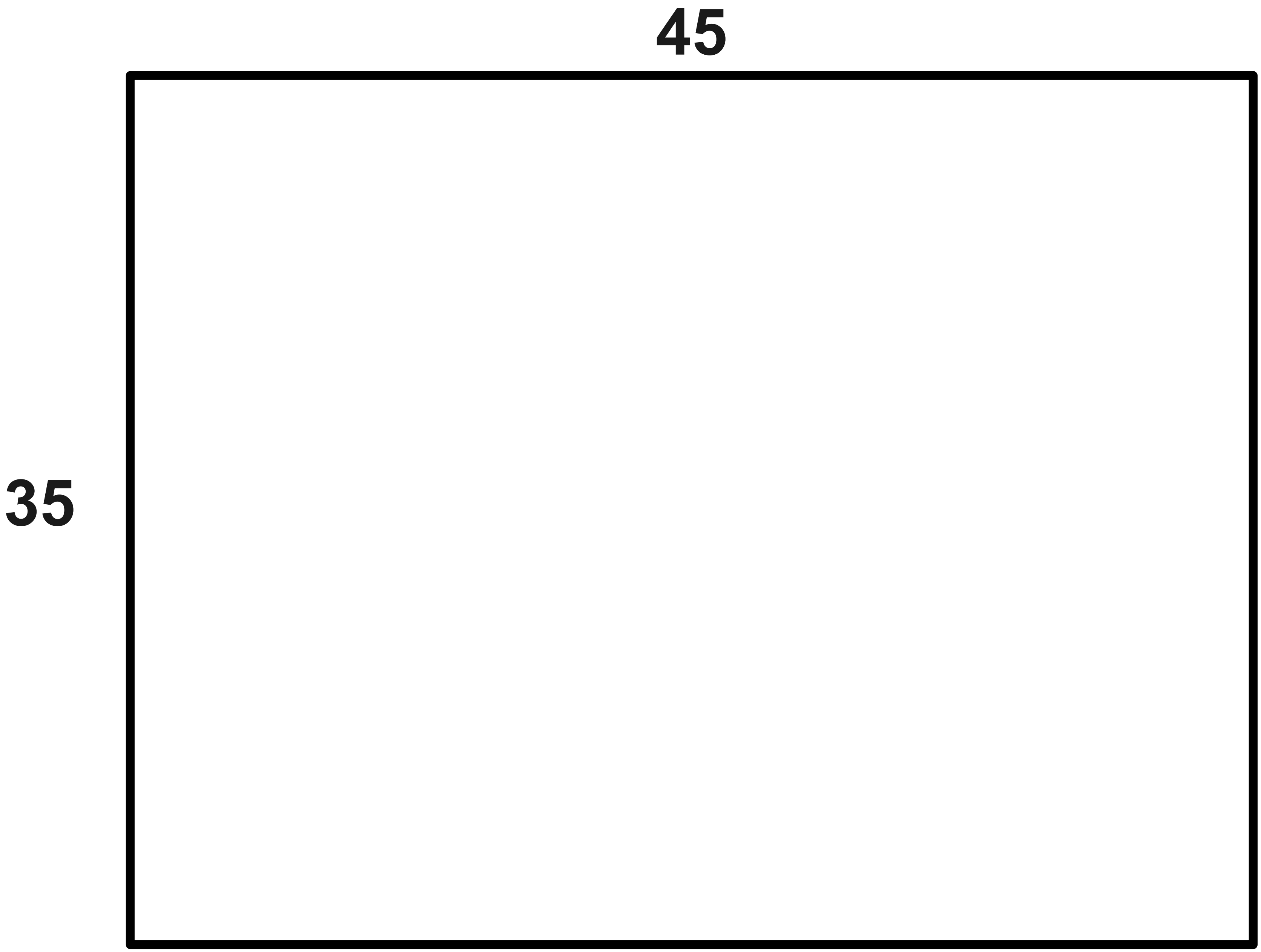


```
7
3
1
```

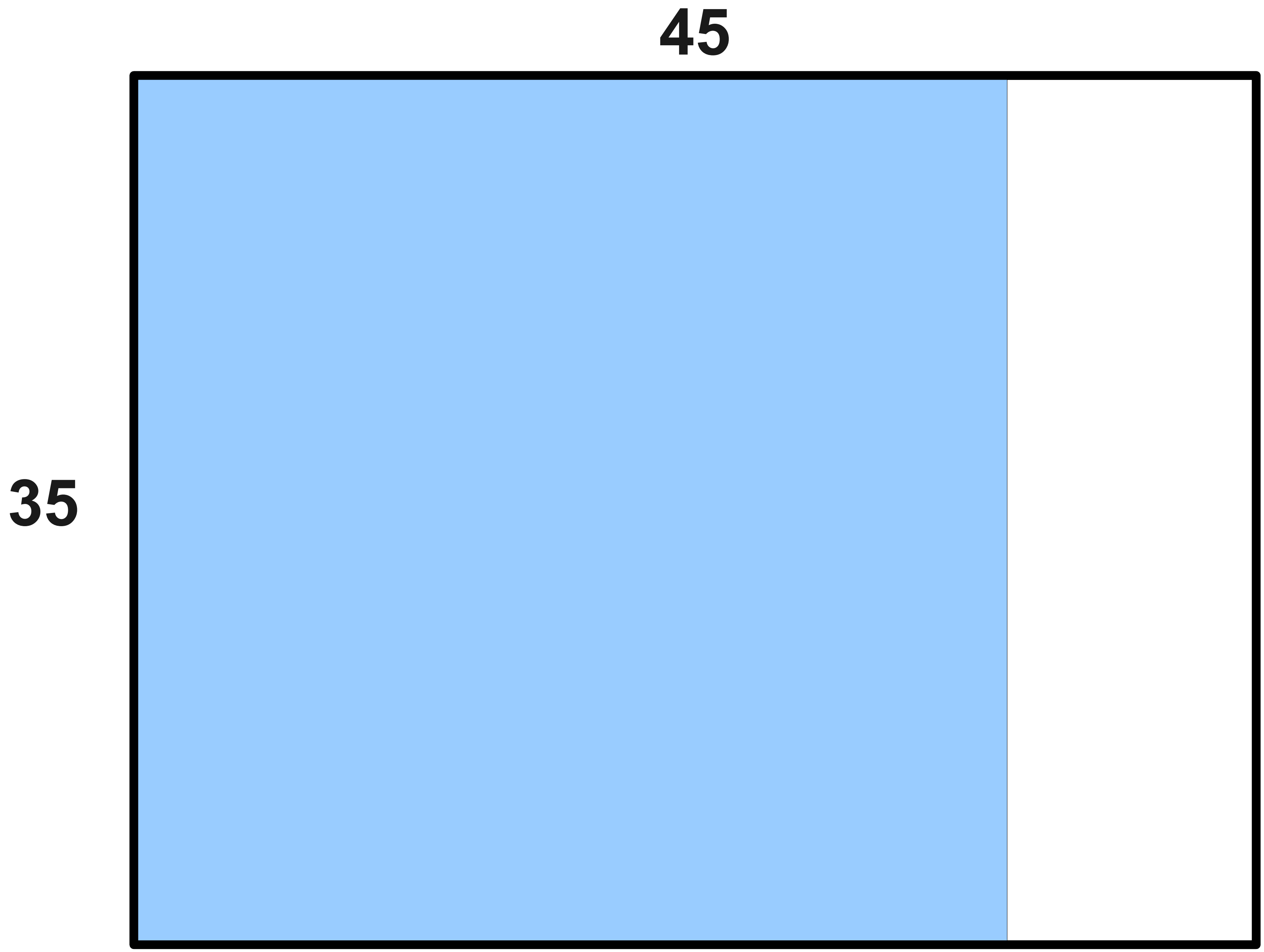
Greatest Common Divisors

- Given two integers a and b, the **greatest common divisor** (or *gcd*) of a and b is the largest number that divides a and b.
- Examples:
  - The *gcd* of 12 and 8 is 4.
  - The *gcd* of 100 and 10 is 10.
  - The *gcd* of 137 and 42 is 1.

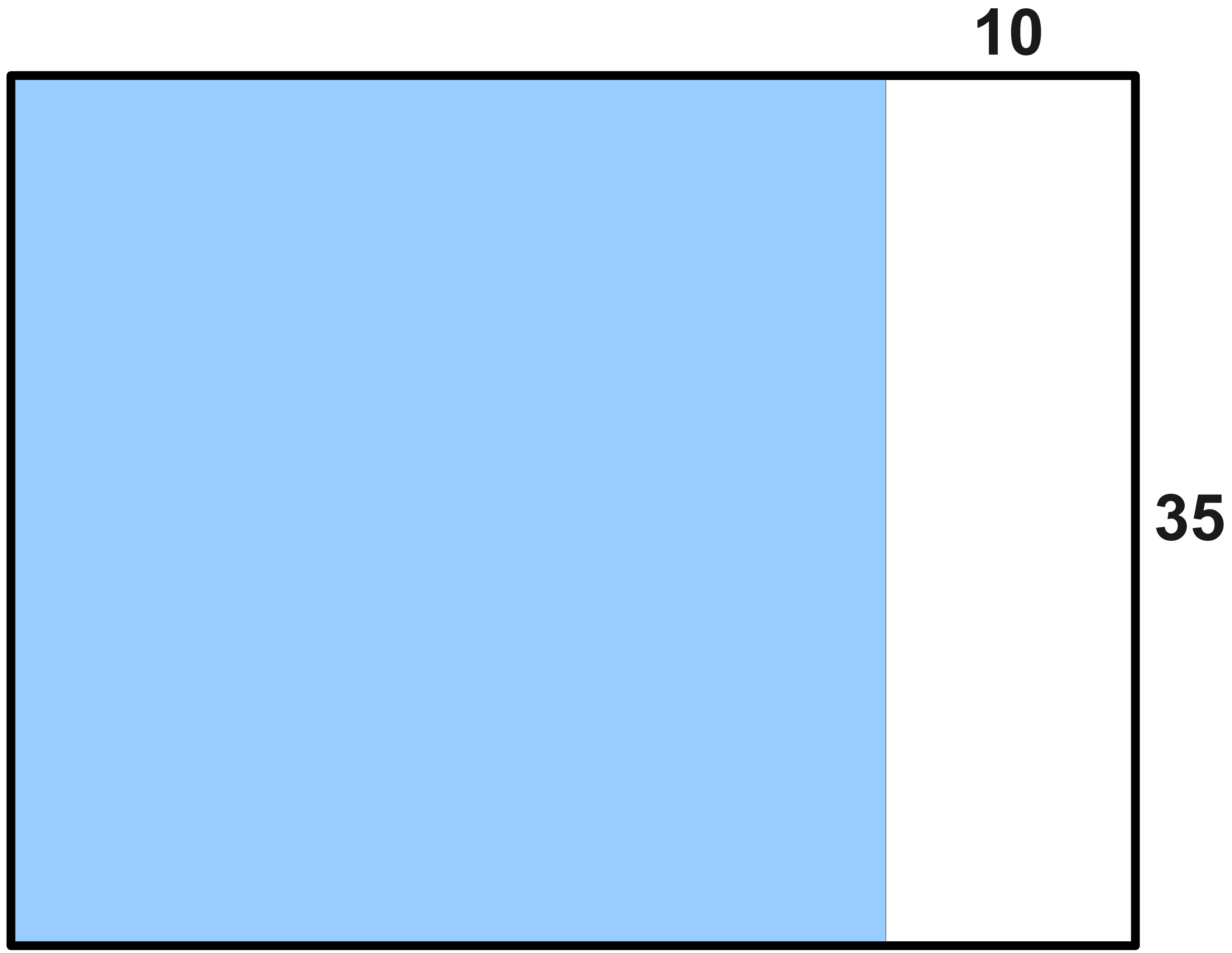
Euclid's Algorithm



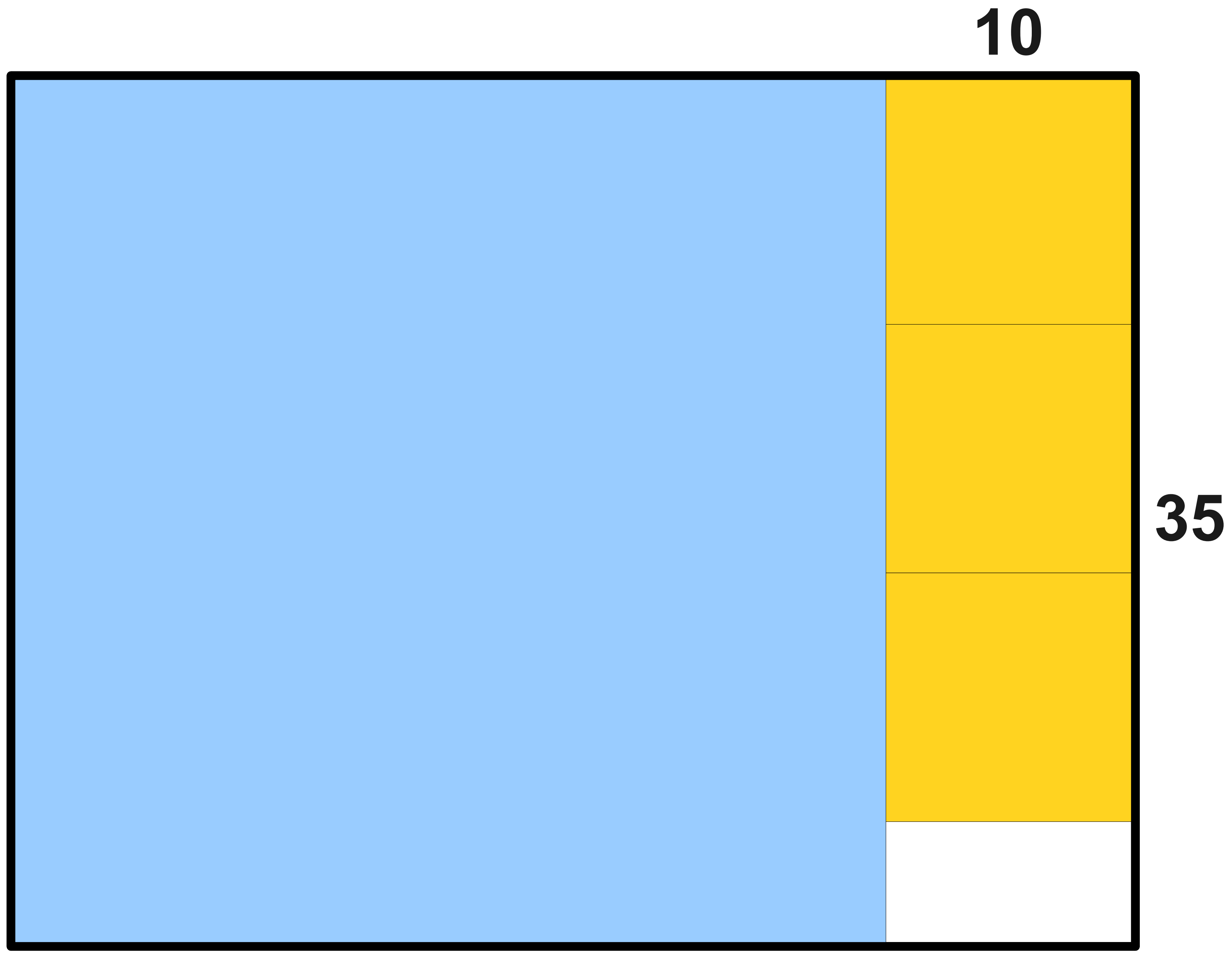
Euclid's Algorithm



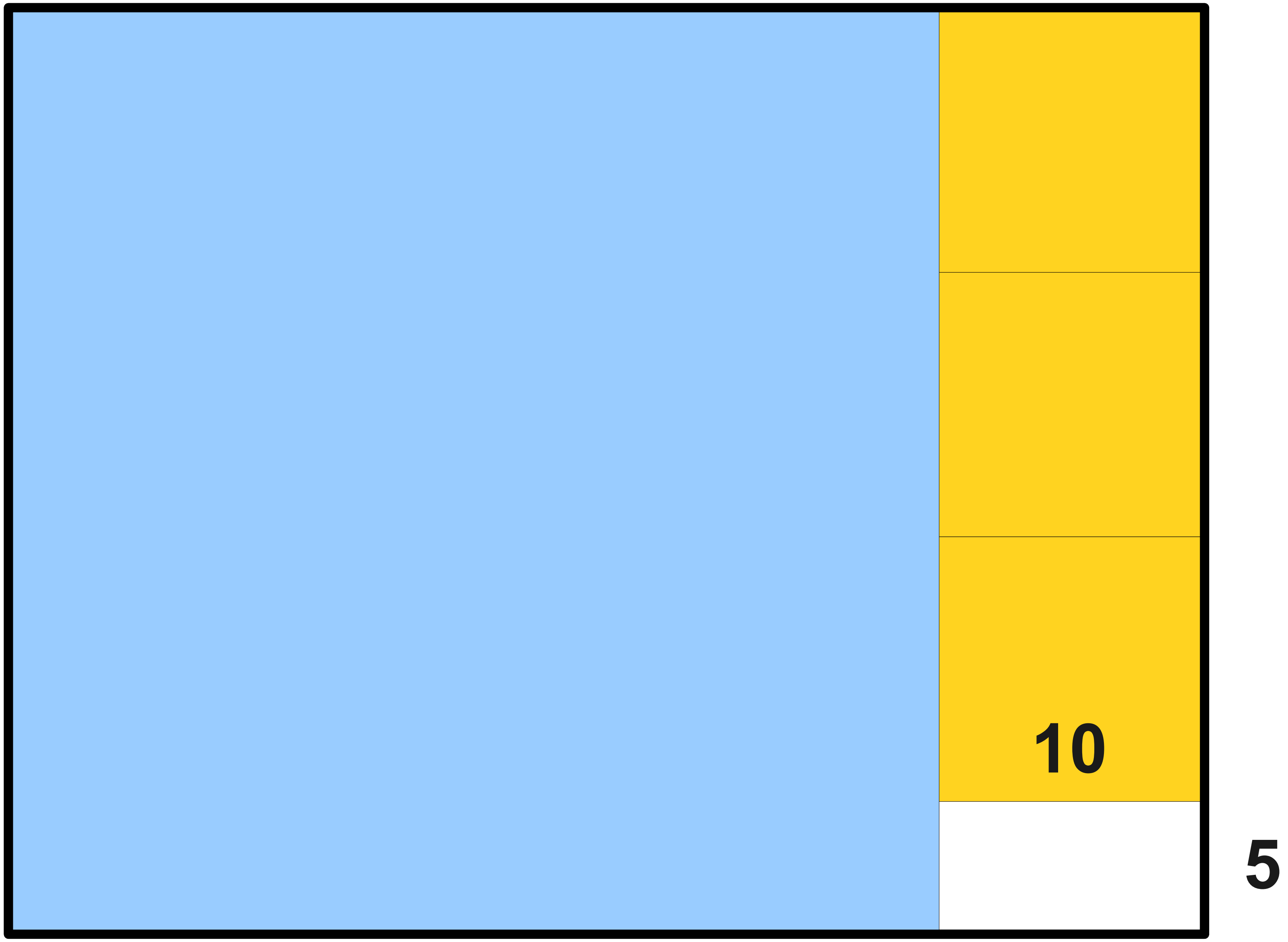
Euclid's Algorithm



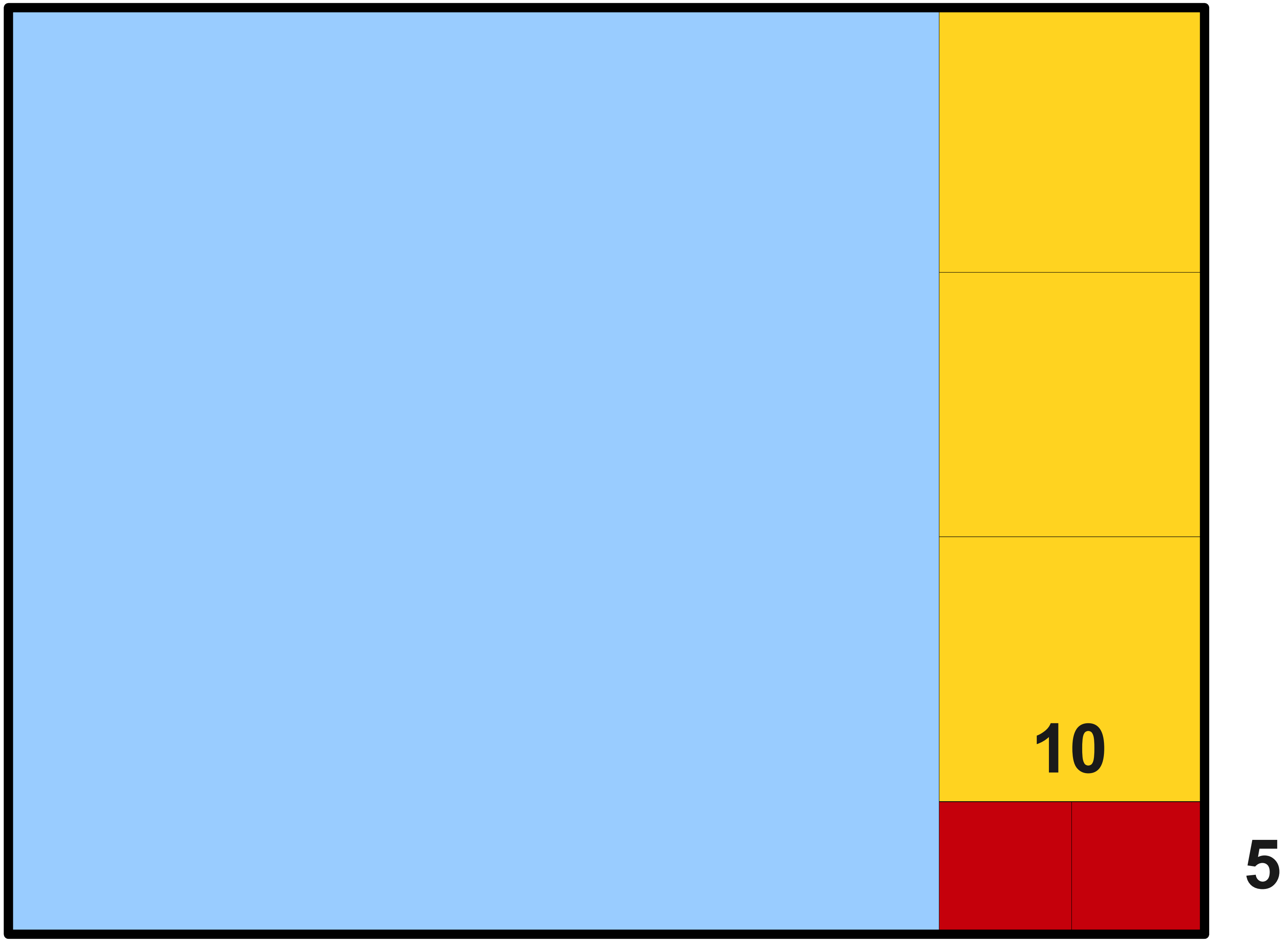
Euclid's Algorithm



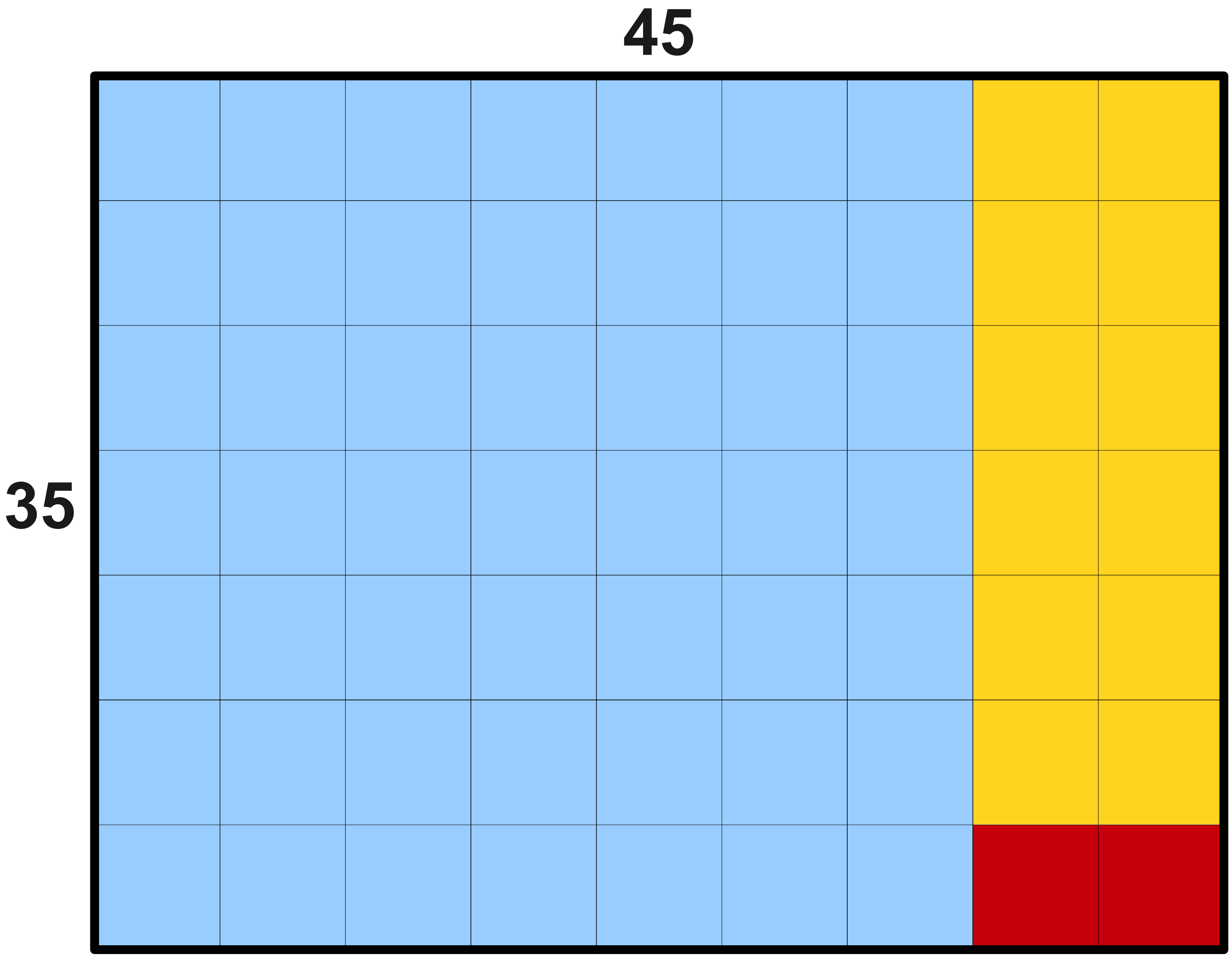
Euclid's Algorithm



Euclid's Algorithm



Euclid's Algorithm



Euclid's Algorithm

- To compute the *gcd* of a and b:
  - If $b = 0$, the *gcd* is a.
  - Otherwise:
    - Divide a by b and obtain the remainder r.
    - Set a to be b and b to be r.
    - Repeat.
- This procedure was known to the Greeks as **anthyphairesis**; it's almost always referred to now as **Euclid's algorithm**.
- It is one of the oldest algorithms still in use today.

Looping Forever

- `while` loops iterate as long as their condition evaluates to `true`.
- A loop of the form `while (true)` will loop forever (unless something stops it).

```
while (true) {
        …
}
```

Video: NyanCat

```
while (true) {
  println("Nyan!");
}
```

Getting Out of Loops

- If you want to immediately bail out of a loop, you can use the `break` statement.
- It is common to see `while (true)` paired with a `break` statement.
- Intuition: Loop forever until the body of the loop decides it's time to leave.

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.
- Technique: The **loop-and-a-half idiom**:

```
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.

Next Time

- **Object-Oriented Programming**
  - How are modern programs structured?
- **Programming with Graphics**
  - Letting your inner artist run wild!
  - The collage model.
  - Geometric figures.