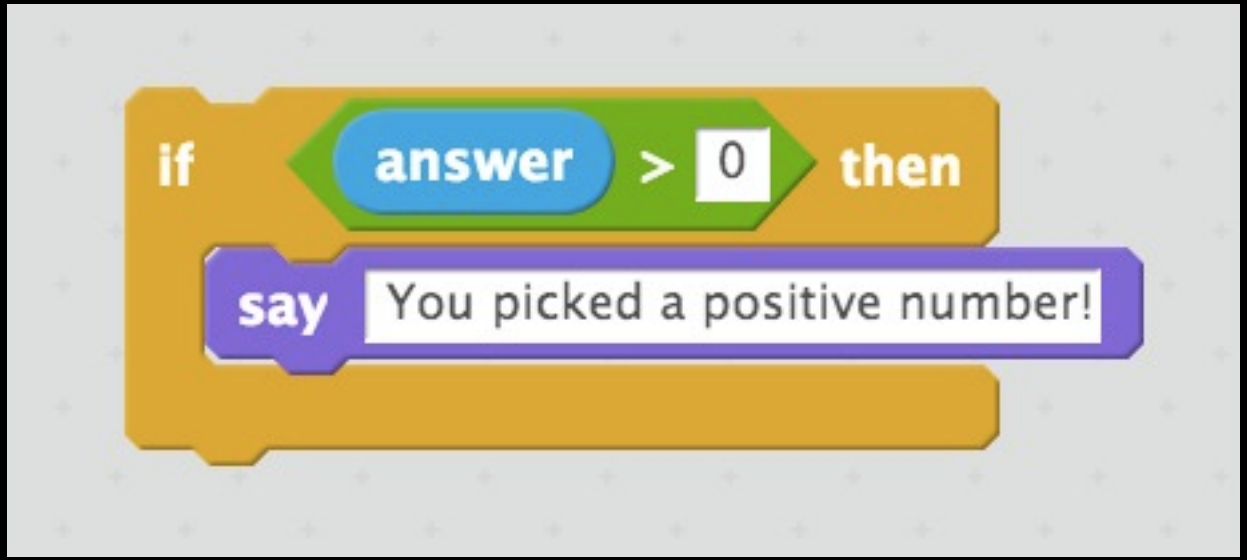
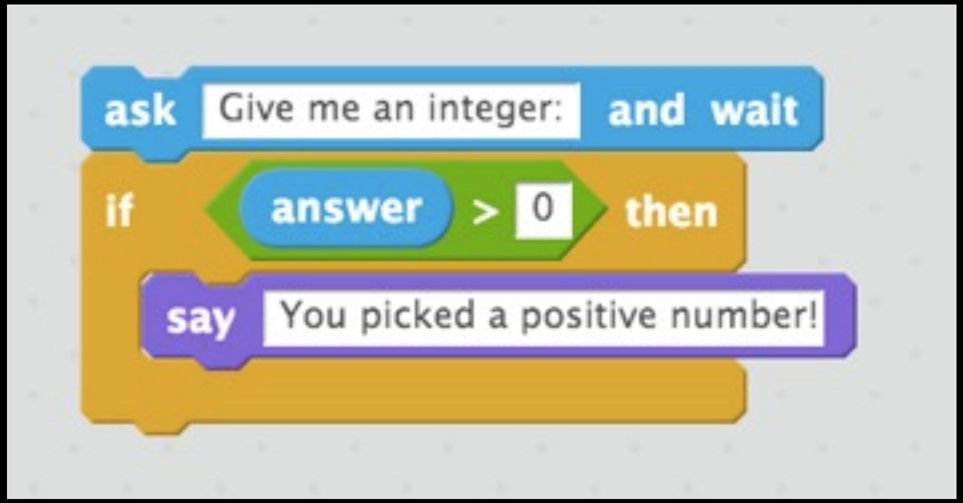
Conditions and Boolean Expressions



```
if answer > 0 then
    say You picked a positive number!
```

Logic in C



```c
#include <cs50.h>
#include <stdio.h>

int main(void)
{
    printf("Give me an integer: ");
    int answer = GetInt();

    if (answer > 0)
    {
        printf("You picked a positive number!\n");
    }
}
```

Boolean Expressions

```
answer > 0
answer < 0
answer >= 0
answer <= 0
answer == 0
answer != 0

!(answer > 0)
```

Combining Expressions

Logical AND

```
answer > 0 && answer < 5
```

Logical OR

```
answer < 0 || answer > 5
```

Combining Expressions

```c
#include <cs50.h>
#include <stdio.h>

int main(void)
{
    printf("Give me an integer: ");
    int answer = GetInt();

    if (answer > 0 && answer < 5)
    {
        printf("You picked a number between 0 and 5!\n");
    }
}
```

if ... else

```c
#include <cs50.h>
#include <stdio.h>

int main(void)
{
    printf("Give me an integer: ");
    int n = GetInt();

    if (n > 0)
    {
        printf("You picked a positive number!\n");
    }
    else
    {
        printf("You picked a negative number or 0!\n");
    }
}
```

if … else if … else

```c
#include <cs50.h>
#include <stdio.h>

int main(void)
{
    printf("Give me an integer: ");
    int n = GetInt();

    if (n > 0)
    {
        printf("You picked a positive number!\n");
    }
    else if (n < 0)
    {
        printf("You picked a negative number!\n");
    }
    else
    {
        printf("You picked 0!\n");
    }
}
```

```c
#include <cs50.h>
#include <stdio.h>

int main(void)
{
    printf("Enter your grade: ");
    int n = GetInt();

    if (n > 90)
    {
        printf("You got an A!\n");
    }
    if (n > 80)
    {
        printf("You got a B!\n");
    }
    if (n > 70)
    {
        printf("You got a C!\n");
    }
    if (n > 60)
    {
        printf("You got a D!\n");
    }
}
```

```c
#include <cs50.h>
#include <stdio.h>

int main(void)
{
    printf("Enter your grade: ");
    int n = GetInt();

    if (n > 90)
    {
        printf("You got an A!\n");
    }
    else if (n > 80)
    {
        printf("You got a B!\n");
    }
    else if (n > 70)
    {
        printf("You got a C!\n");
    }
    else if (n > 60)
    {
        printf("You got a D!\n");
    }
}
```

switch

```c
int main(void)
{
    printf("Give me an integer between 1 and 3: ");
    int n = GetInt();

    switch (n)
    {
        case 1:
            printf("You picked a low number.\n");
            break;
        case 2:
            printf("You picked a medium number.\n");
            break;
        case 3:
            printf("You picked a high number.\n");
            break;
        default:
            printf("Invalid.\n");
    }
}
```

switch

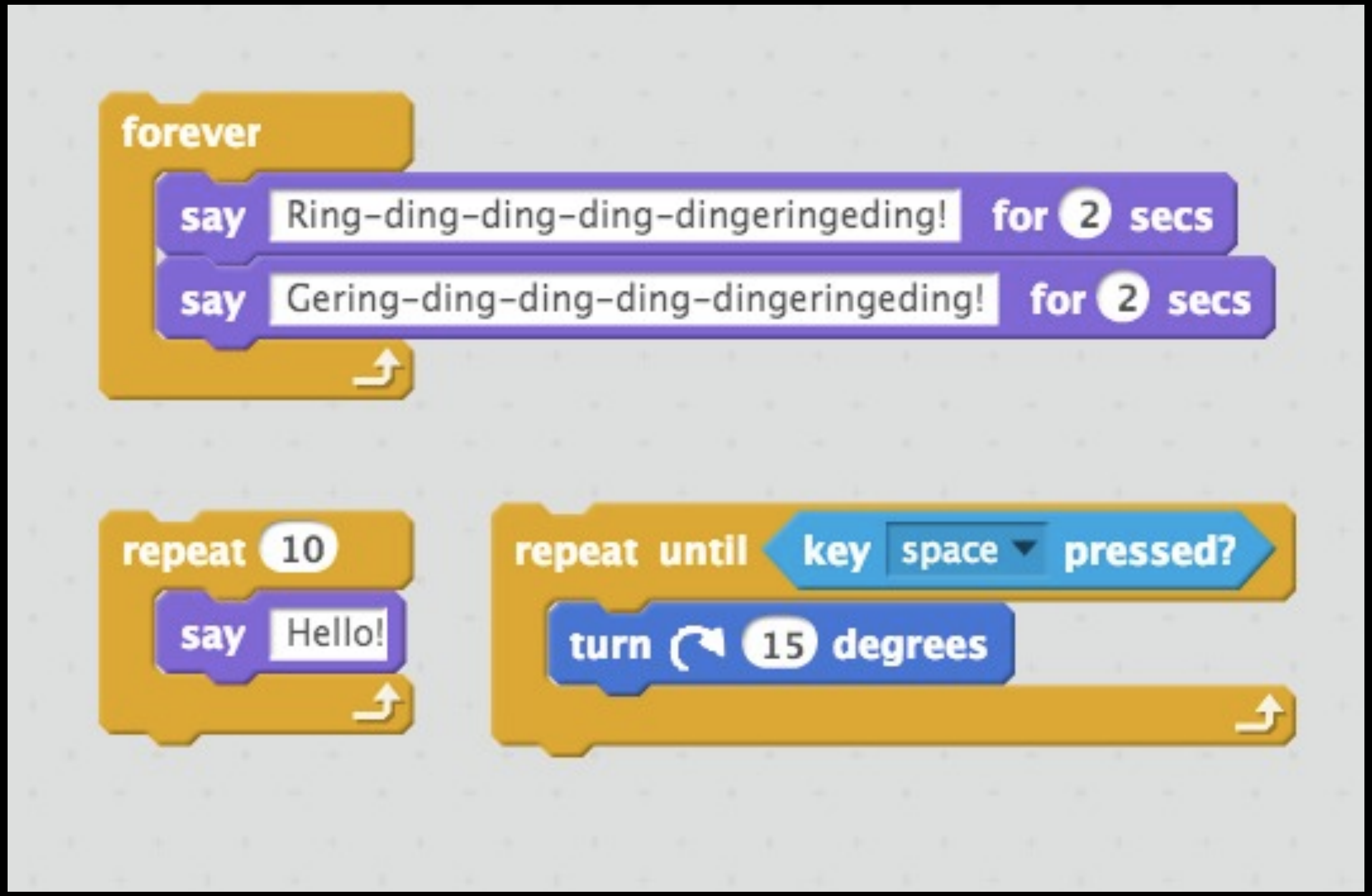
```c
int main(void)
{
    printf("Give me an integer between 1 and 3: ");
    int n = GetInt();

    switch (n)
    {
        case 1:
        case 2:
            printf("Didn't pick a high number.\n");
            break;
        case 3:
            printf("You picked a high number.\n");
            break;
        default:
            printf("Invalid.\n");
    }
}
```

Ternary Operator

```c
#include <cs50.h>
#include <stdio.h>

int main(void)
{
    printf("Give me an integer: ");
    int n = GetInt();

    string s;
    if (n > 100)
    {
        s = "high";
    }
    else
    {
        s = "low";
    }

    printf("You picked a %s number!\n", s);
}
```

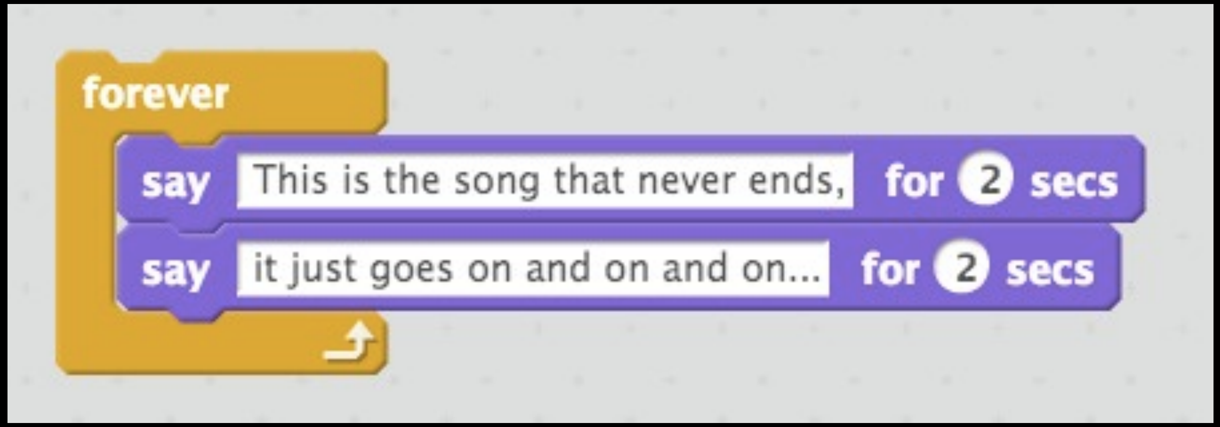
```c
#include <cs50.h>
#include <stdio.h>

int main(void)
{
    printf("Give me an integer: ");
    int n = GetInt();

    string s = (n > 100) ? "high" : "low";

    printf("You picked a %s number!\n", s);
}
```

Loops



```
forever
    say Ring-ding-ding-ding-dingeringeding! for 2 secs
    say Gering-ding-ding-ding-dingeringeding! for 2 secs

repeat 10
    say Hello!

repeat until key space pressed?
    turn ↻ 15 degrees
```

uh oh



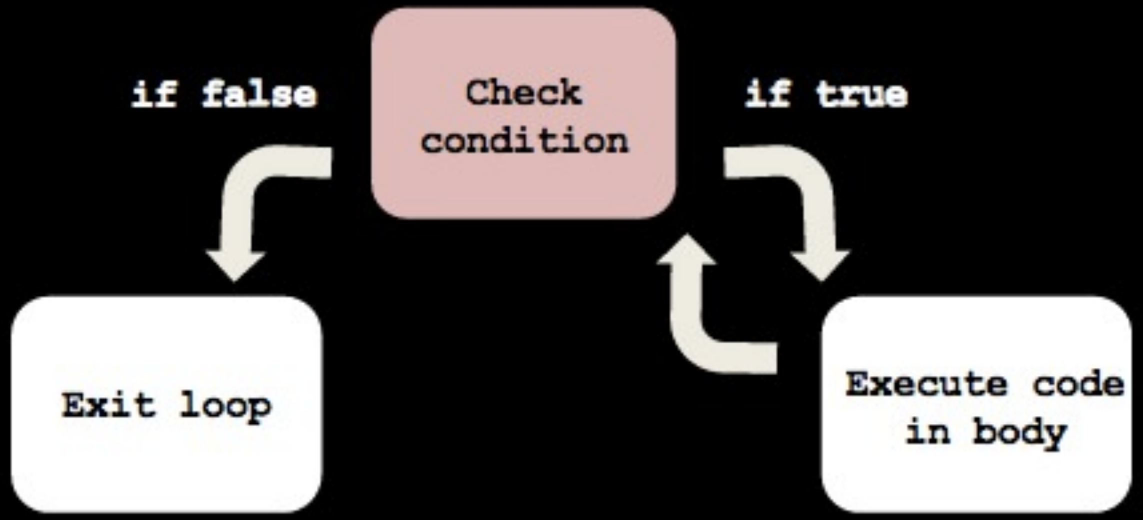
This is the song that never ends,

it just goes on and on and on...

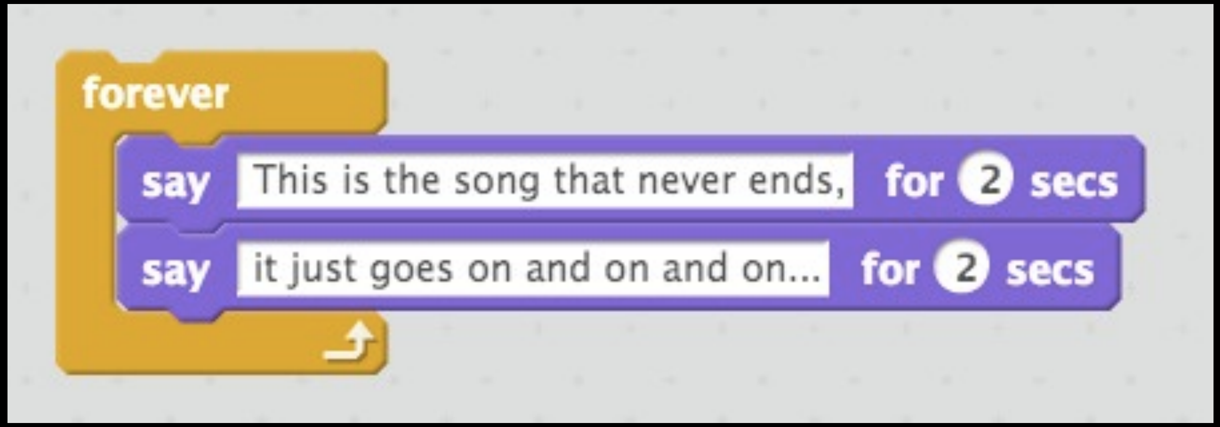
```
#include <cs50.h>
#include <stdio.h>

int main(void)
{
    printf("This is the song that never ends,\n");
    printf("it just goes on and on and on...\n");
    printf("This is the song that never ends,\n");
    printf("it just goes on and on and on...\n");
    printf("This is the song that never ends,\n");
    printf("it just goes on and on and on...\n");
    printf("This is the song that never ends,\n");
    printf("it just goes on and on and on...\n");
    // ... how do we keep going???
}
```

while loop

```
while (condition)
{
    // execute code
}
```

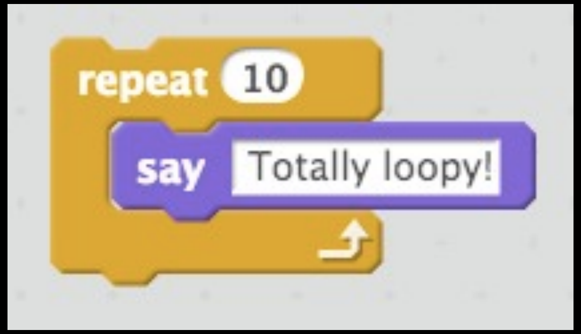


while loop


forever
say This is the song that never ends, for 2 secs
say it just goes on and on and on... for 2 secs

```c
#include <cs50.h>
#include <stdio.h>

int main(void)
{
    while (true)
    {
        printf("This is the song that never ends,\n");
        printf("it just goes on and on and on...\n");
    }
}
```

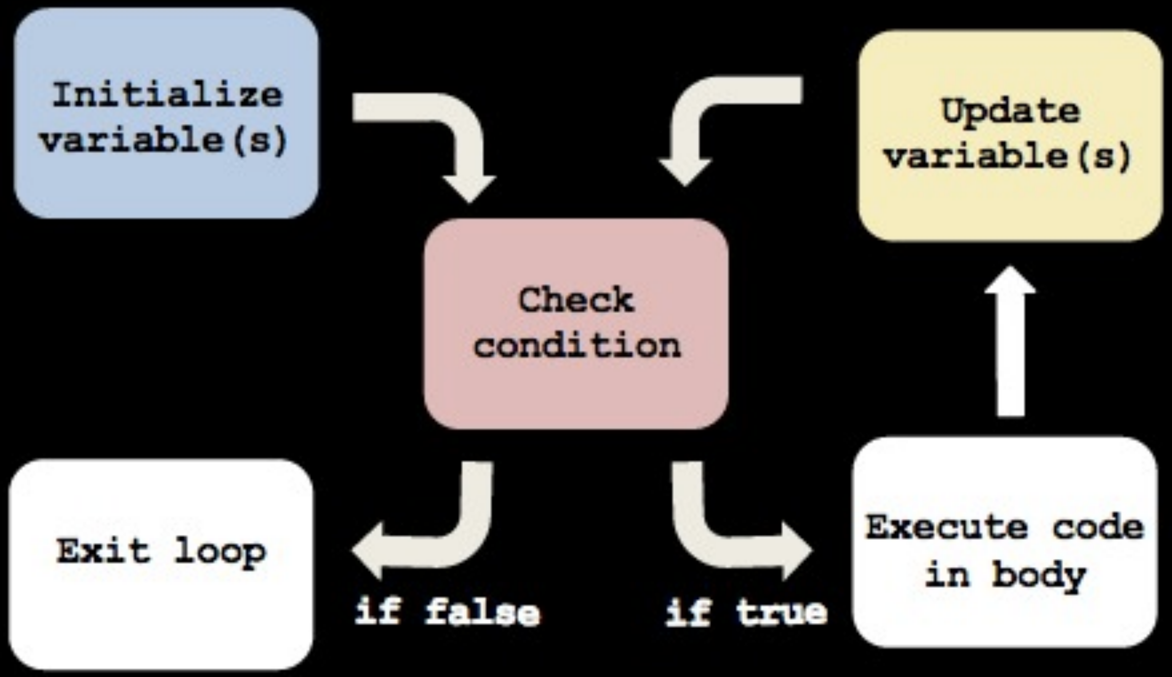
with a variable and condition



```
#include <cs50.h>
#include <stdio.h>

int main(void)
{
    int i = 10;
    while (i > 0)
    {
        printf("Totally loopy!\n");
        i--;
    }
}
```

for loop

```
for (initialization; condition; update)
{
    // execute this code
}
```



for (initialization; condition; update)

```
#include <cs50.h>
#include <stdio.h>

int main(void)
{
    int i = 10;
    while (i > 0)
    {
        printf("Totally loopy!\n");
        i--;
    }
}
```

```
#include <cs50.h>
#include <stdio.h>

int main(void)
{
    for (int i = 10; i > 0; i--)
    {
        printf("Totally loopy!\n");
    }
}
```

using the counter variable

```c
#include <cs50.h>
#include <stdio.h>

int main(void)
{
    for (int i = 10; i > 0; i--)
    {
        printf("Counting down ... %i\n", i);
    }
}
```

input validation

```c
#include <cs50.h>
#include <stdio.h>

int main(void)
{
    printf("Enter a positive number: ");
    int input = GetInt();

    while (input <= 0)
    {
        printf("Enter a positive number: ");
        input = GetInt();
    }
}
```

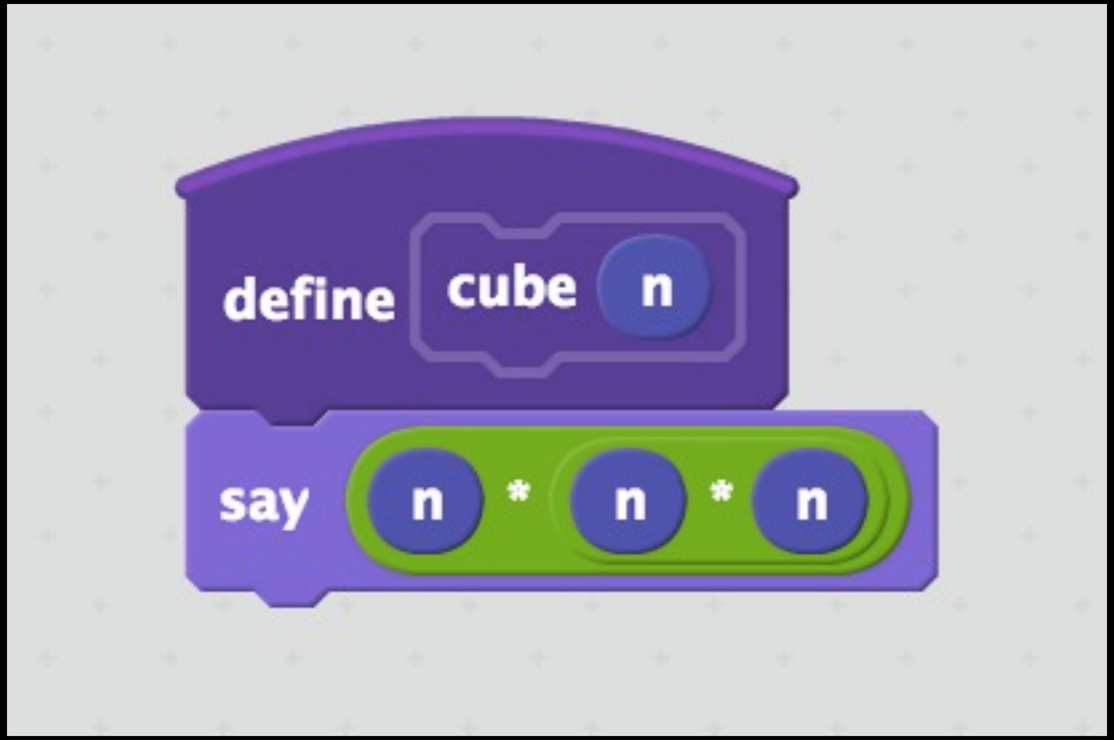
do-while loop

```c
#include <cs50.h>
#include <stdio.h>

int main(void)
{
    int input;
    do
    {
        printf("Enter a positive number: ");
        input = GetInt();
    }
    while (input <= 0);
}
```

breaking

```c
#include <cs50.h>
#include <stdio.h>

int main(void)
{
    int input;
    do
    {
       printf("Enter a positive number: ");
       input = GetInt();

       if (input > 0)
       {
           break;
       }
    }
    while (true);
}
```

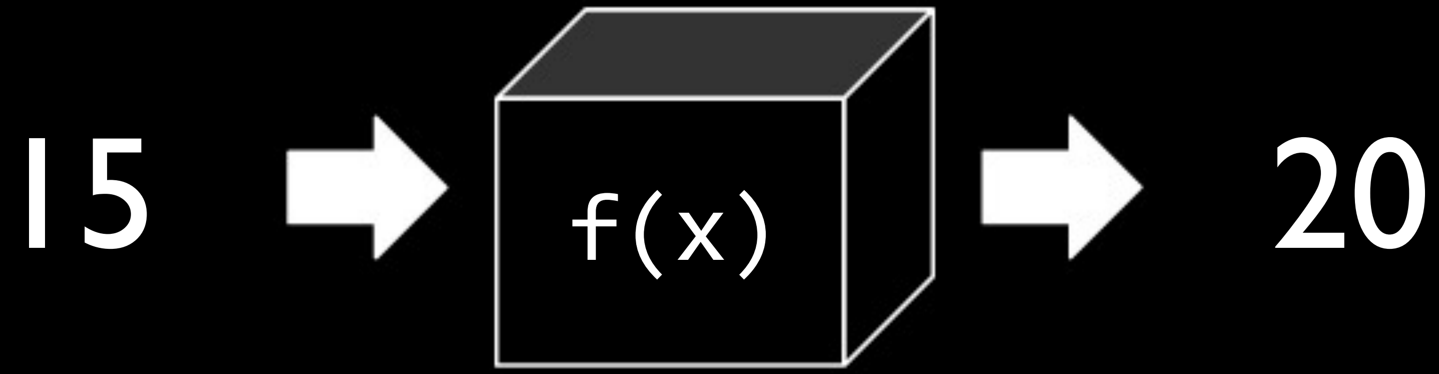
functions



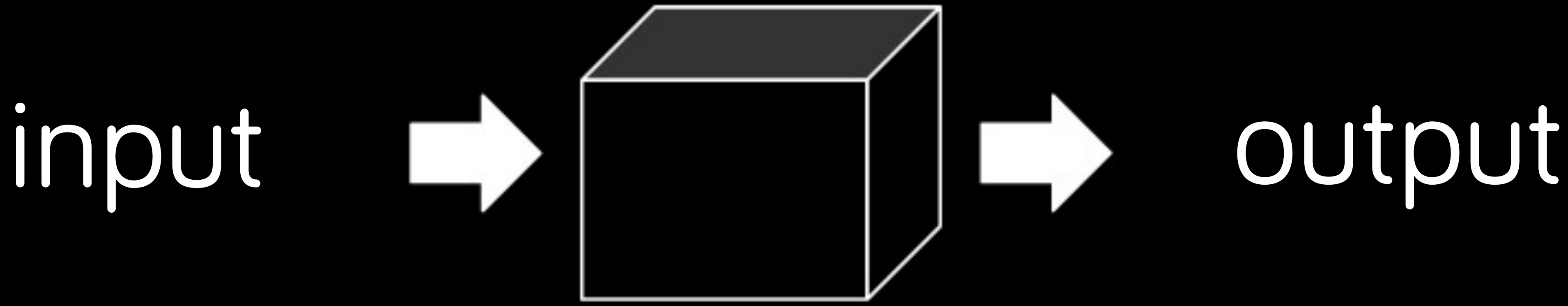
```
define cube (n)
say ((n) * ((n) * (n)))
```

functions from algebra

$$f(x) = x + 5$$



functions, more generally



organization

simplification

reusability

abstraction

```c
int quadruple(int input)
{
    return input * 4;
}

int quadruple(int input)
{
    return input << 2;
}

int quadruple(int input)
{
    // ??? no need to know!
}
```

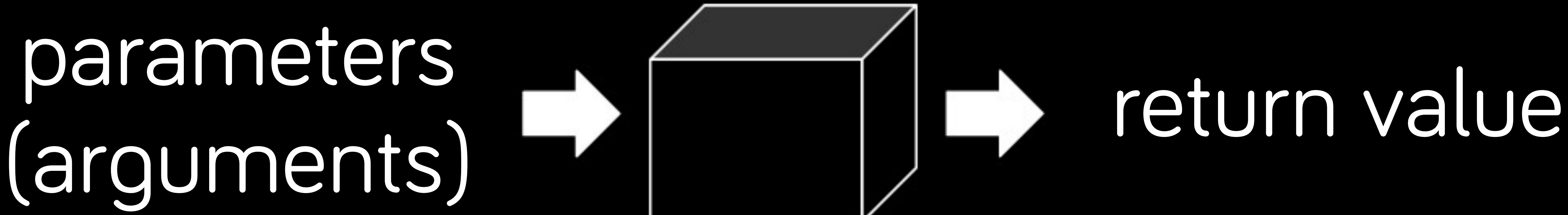
abstraction

```c
#include <stdio.h>

int main(void)
{
    int x = 2;
    int y = 3;
    int z = 4;
    x = x * 3;  ←
    y = y * 3;  ←  fix all three
    z = z * 3;  ←
}
```

```c
#include <stdio.h>

int cube(int input);

int main(void)
{
    int x = 2;
    int y = 3;
    int z = 4;
    x = cube(x);
    y = cube(y);
    z = cube(z);
}

int cube(int input)
{
    return input * 3;
}
```

fix just once! →

we call these



function definition

```
int cube(int input)
{
    int output = input * input * input;
    return output;
}
```

function header

```
int cube(int input)
{
    int output = input * input * input;
    return output;
}
```

function header

return type

name

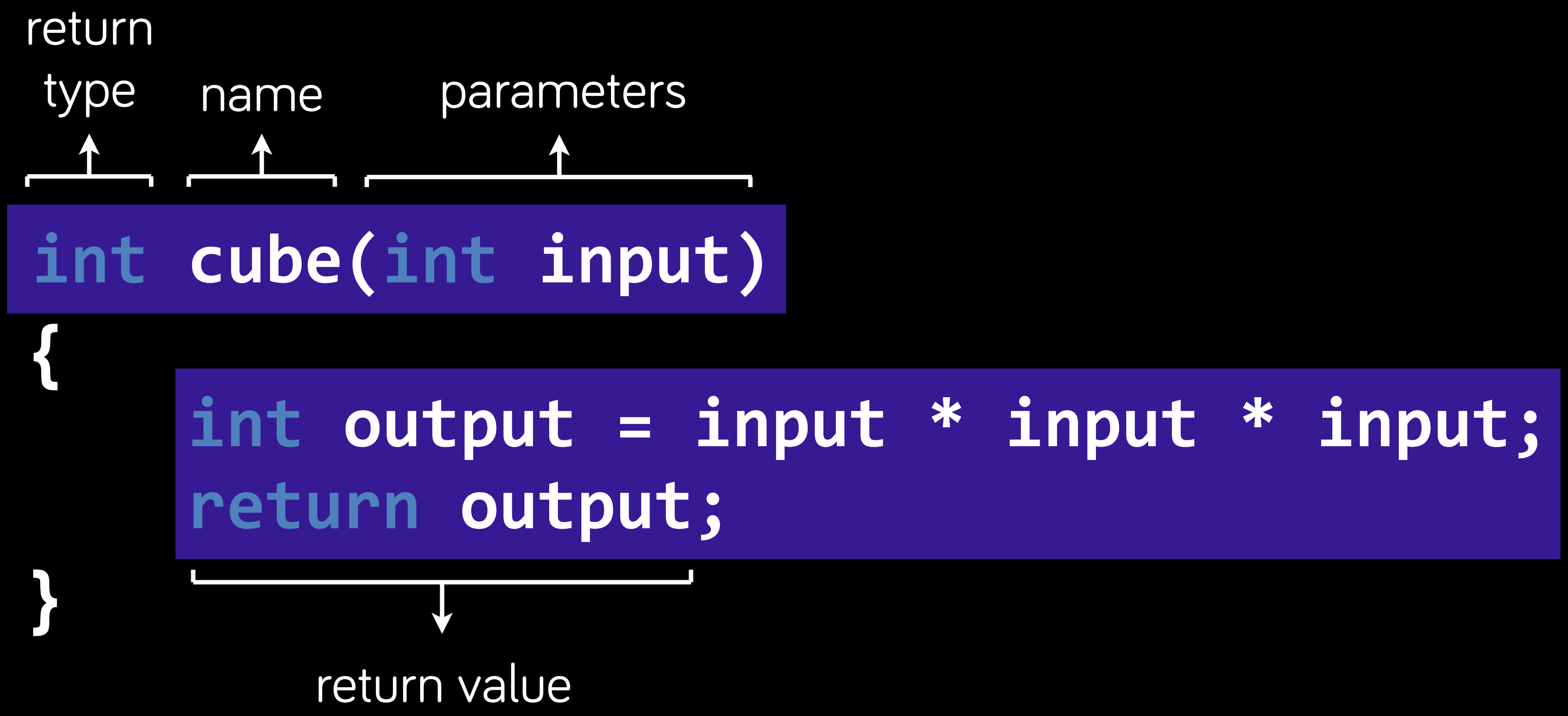
parameters

```
int cube(int input)
{
     int output = input * input * input;
     return output;
}
```

function body

```
int cube(int input)
{
    int output = input * input * input;
    return output;
}
```

return value



```
int cube(int input)
{
    int output = input * input * input;
    return output;
}
```

using a function

```c
#include <stdio.h>

int cube(int input)
{
    int output = input * input * input;
    return output;
}

int main(void)
{
    int x = 2;
    printf("x is %i\n", x);
    x = cube(x);
    printf("x is %i\n", x);
}
```

oops, order matters!

```c
#include <stdio.h>

int main(void)
{
    int x = 2;
    printf("x is %i\n", x);
    x = cube(x);
    printf("x is %i\n", x);
}

int cube(int input)
{
    int output = input * input * input;
    return output;
}
```

function prototype

```
#include <stdio.h>

int cube(int input);

int main(void)
{
    int x = 2;
    printf("x is %i\n", x);
    x = cube(x);
    printf("x is %i\n", x);
}

int cube(int input)
{
    int output = input * input * input;
    return output;
}
```

parameter vs argument

parameter

```c
int cube(int input)
{
    int output = input * input * input;
    return output;
}
```

argument

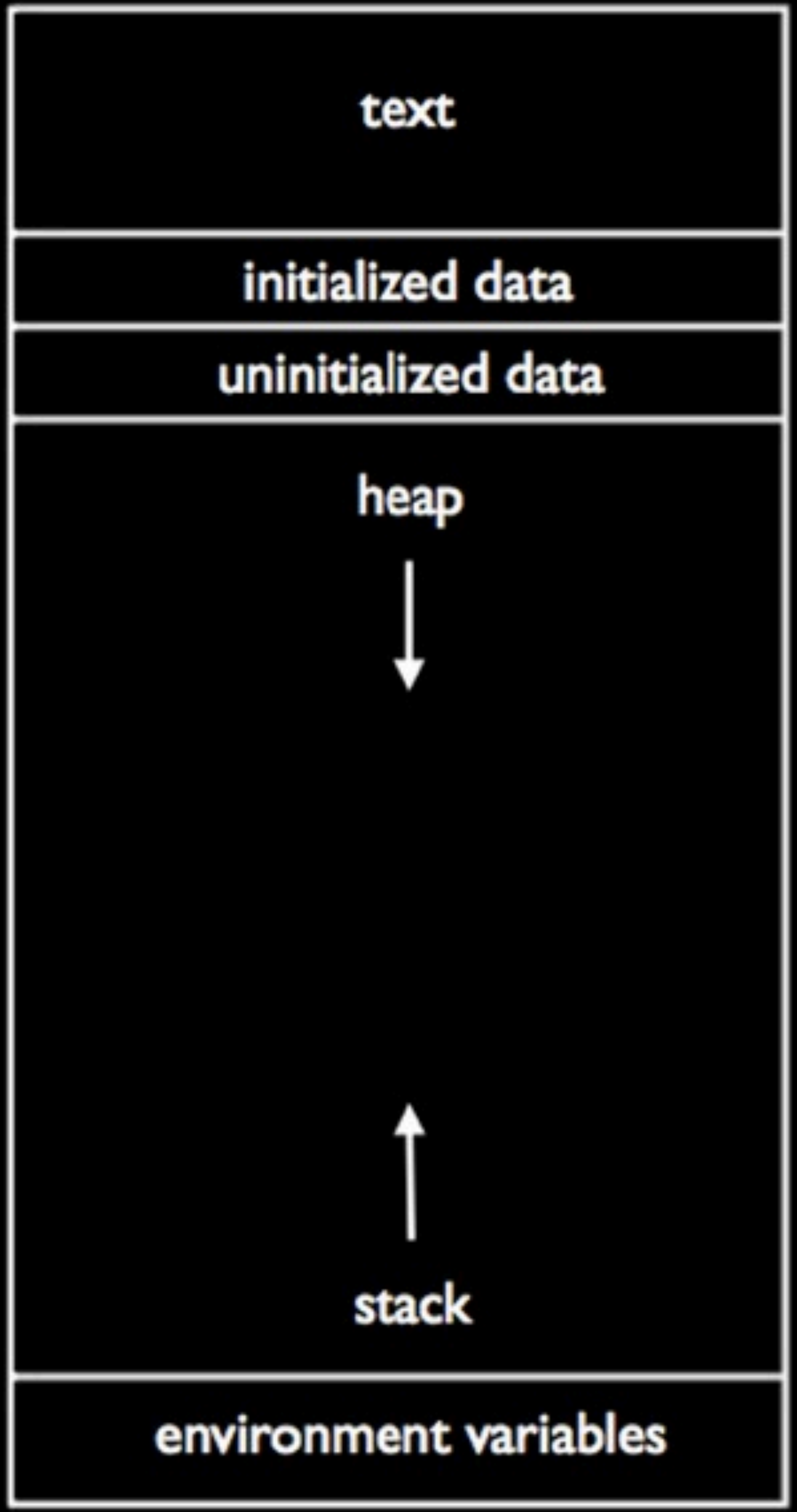
```c
int main(void)
{
    int x = 2;
    x = cube(x);
    printf("x is %i\n", x);
}
```

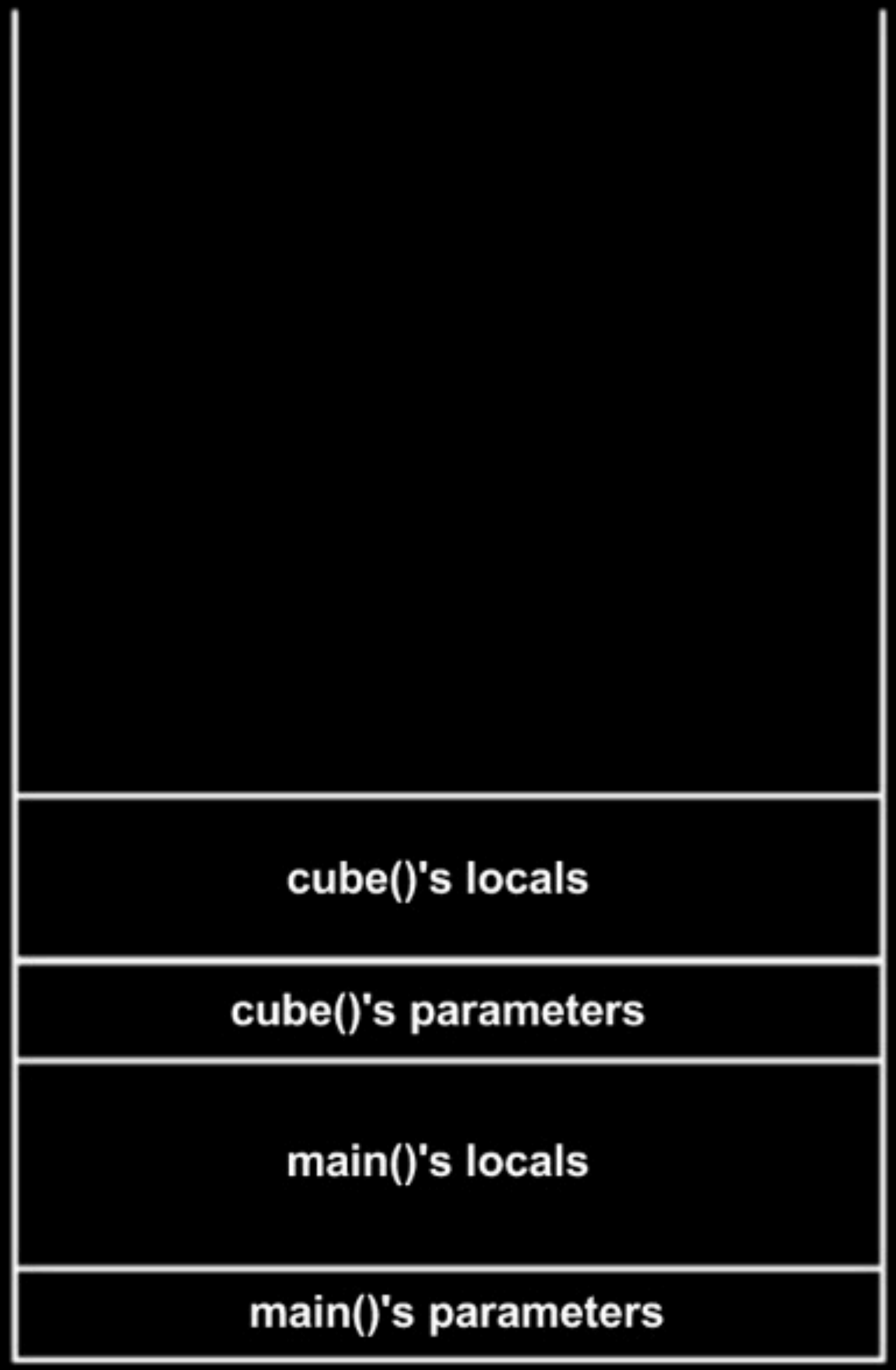
side effects

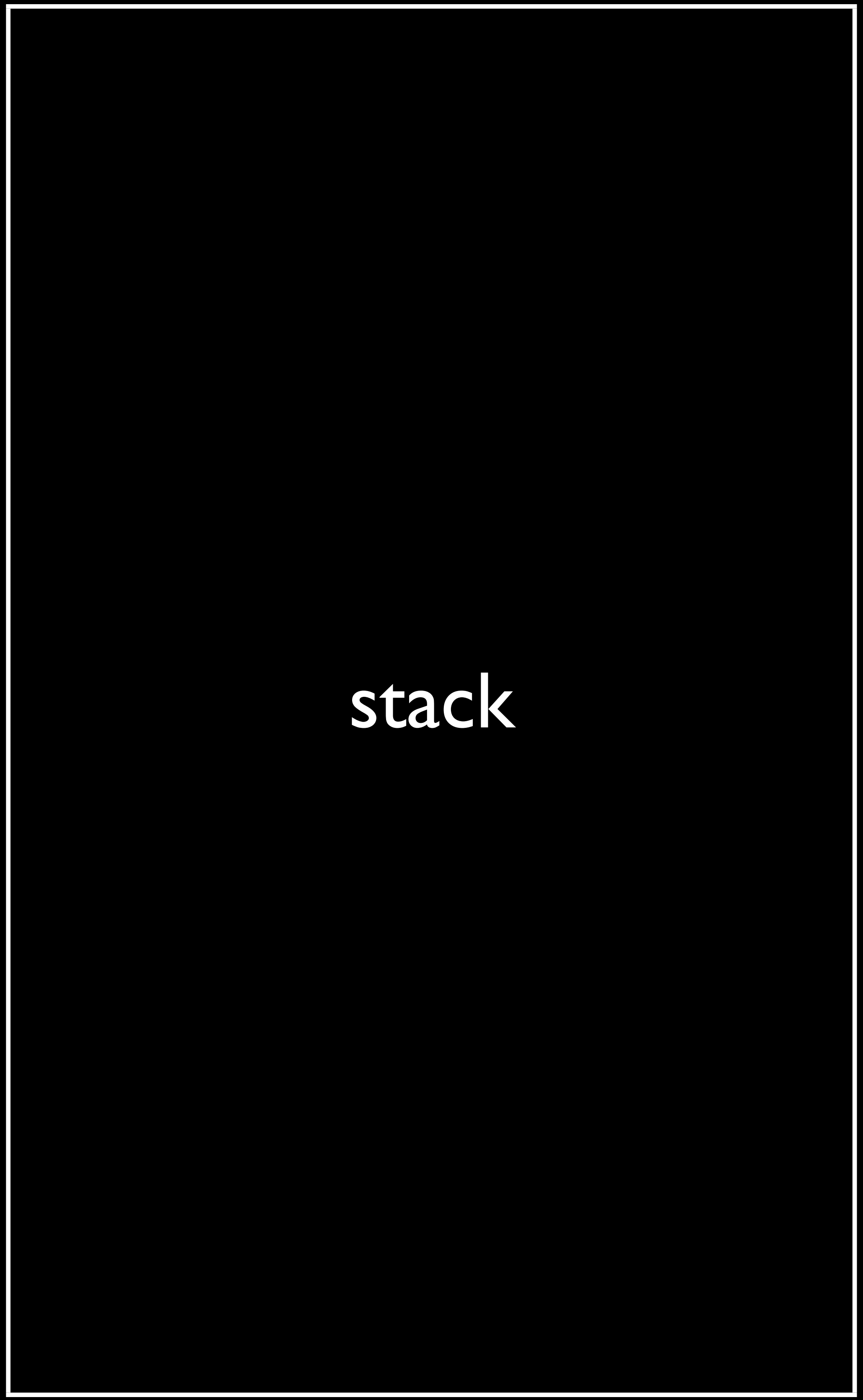
```
#include <stdio.h>

int main(void)
{
    while (true)
    {
        sing();
    }
}

void sing()
{
    printf("This is the song that never ends,\n");
    printf("it just goes on and on and on...\n");
}
```

what does this program do?

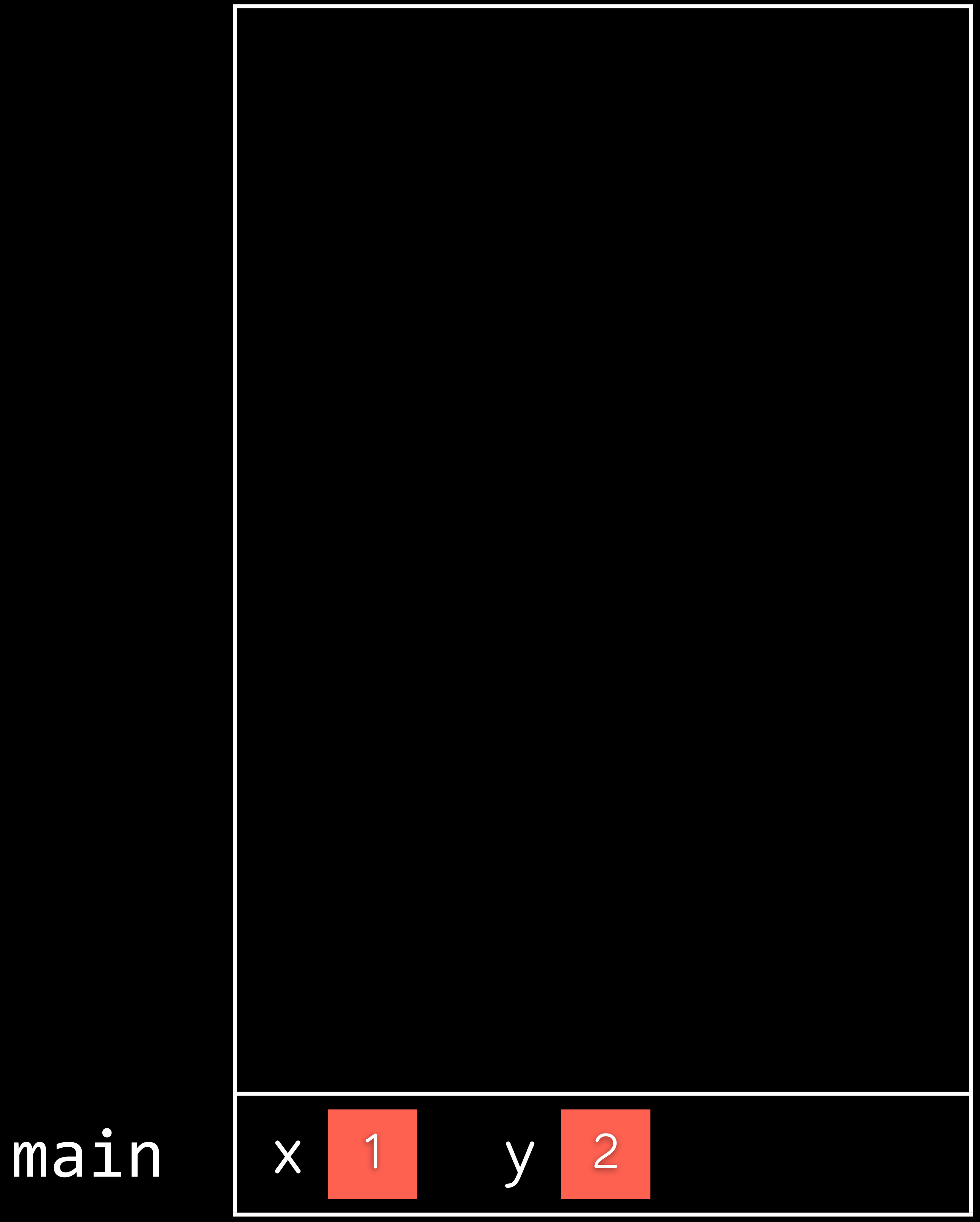
```
#include <stdio.h>
void swap(int a, int b);

int main(void)
{
    int x = 1;
    int y = 2;
    swap(x, y);
    printf("x is %i\n", x);
    printf("y is %i\n", y);
}

void swap(int a, int b)
{
    int tmp = a;
    a = b;
    b = tmp;
}
```


text
initialized data
uninitialized data
heap
stack
environment variables

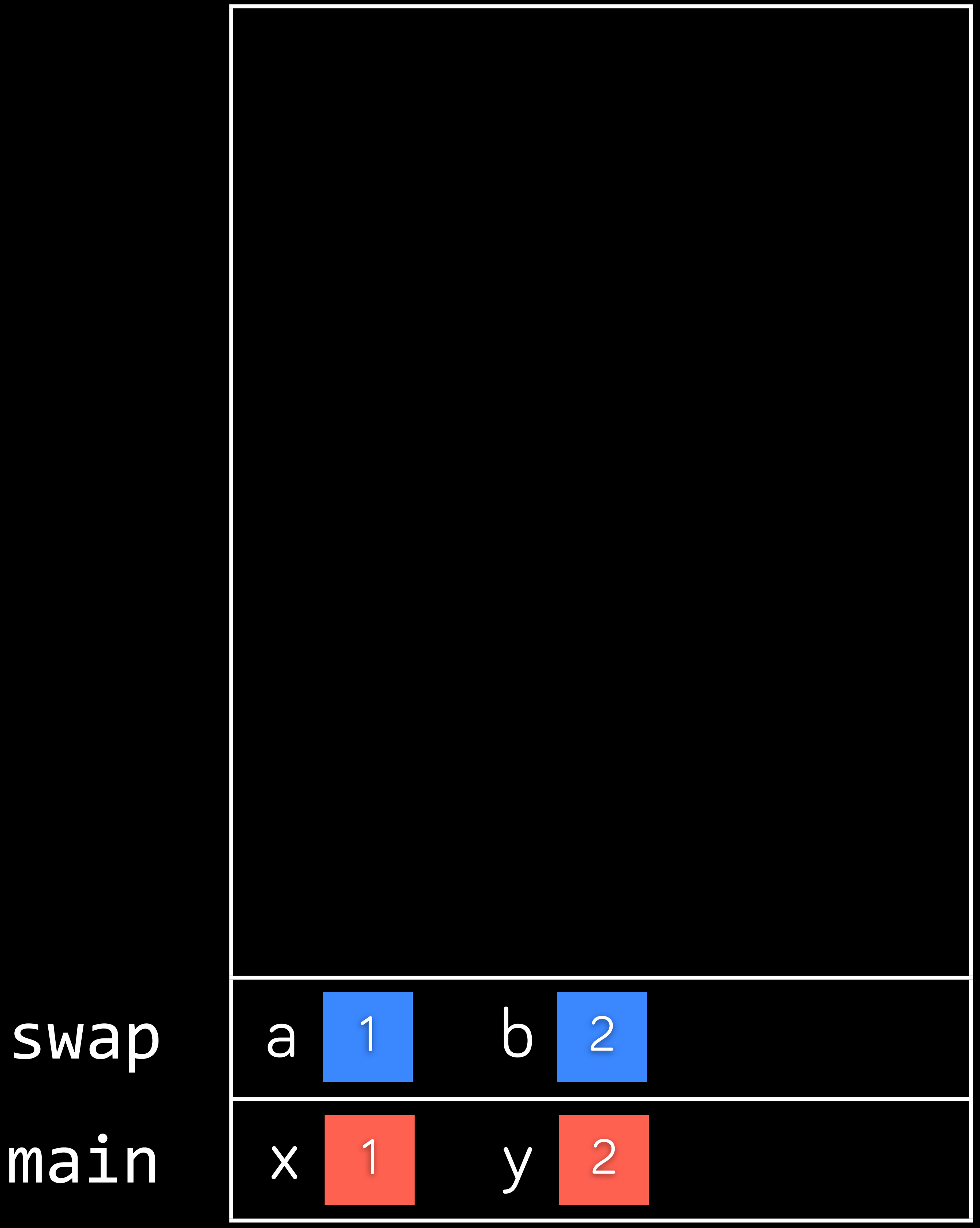

cube()'s locals
cube()'s parameters
main()'s locals
main()'s parameters



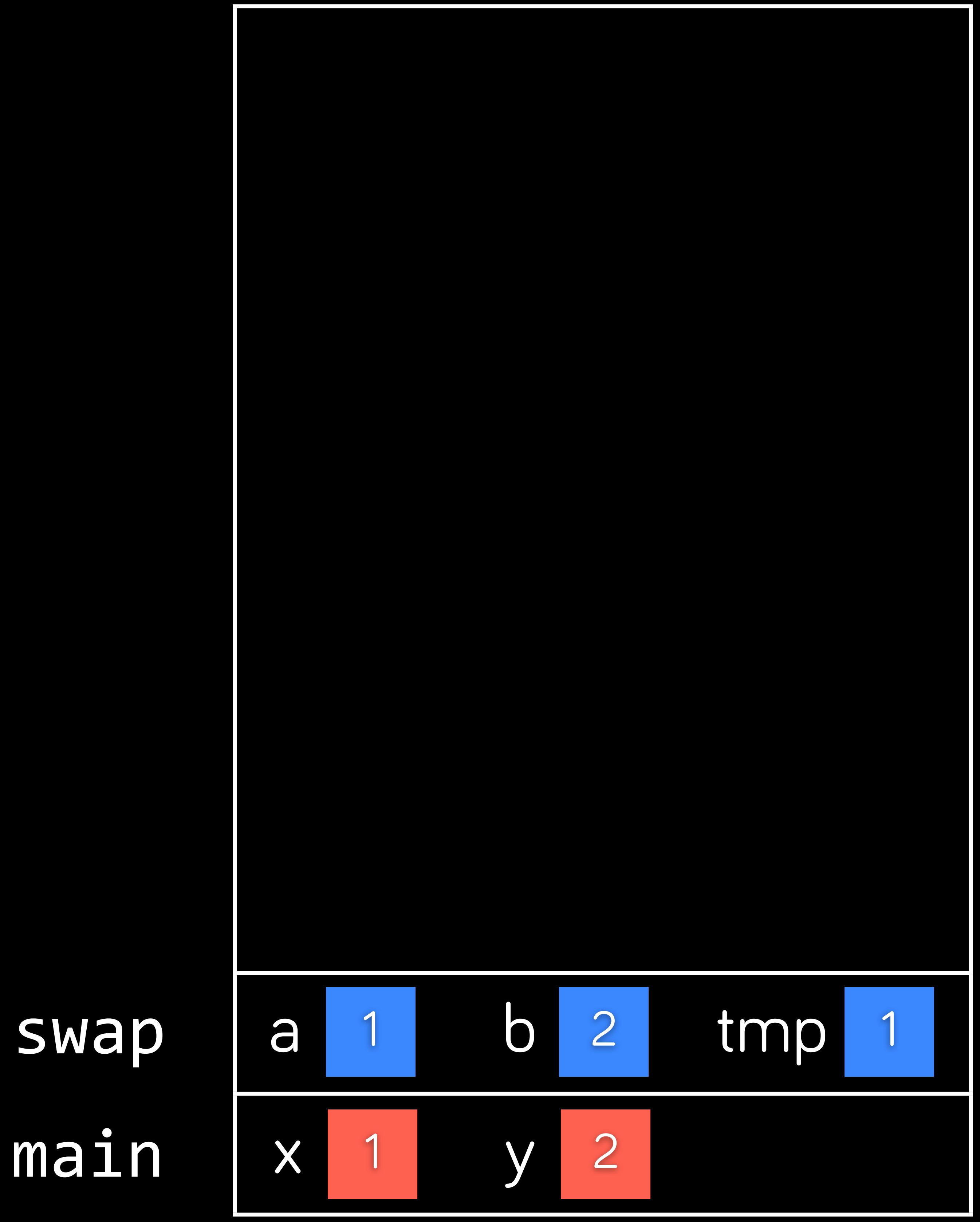
```c
#include <stdio.h>
void swap(int a, int b);

int main(void)
{
    int x = 1;
    int y = 2;
    swap(x, y);
    printf("x is %i\n", x);
    printf("y is %i\n", y);
}

void swap(int a, int b)
{
    int tmp = a;
    a = b;
    b = tmp;
}
```



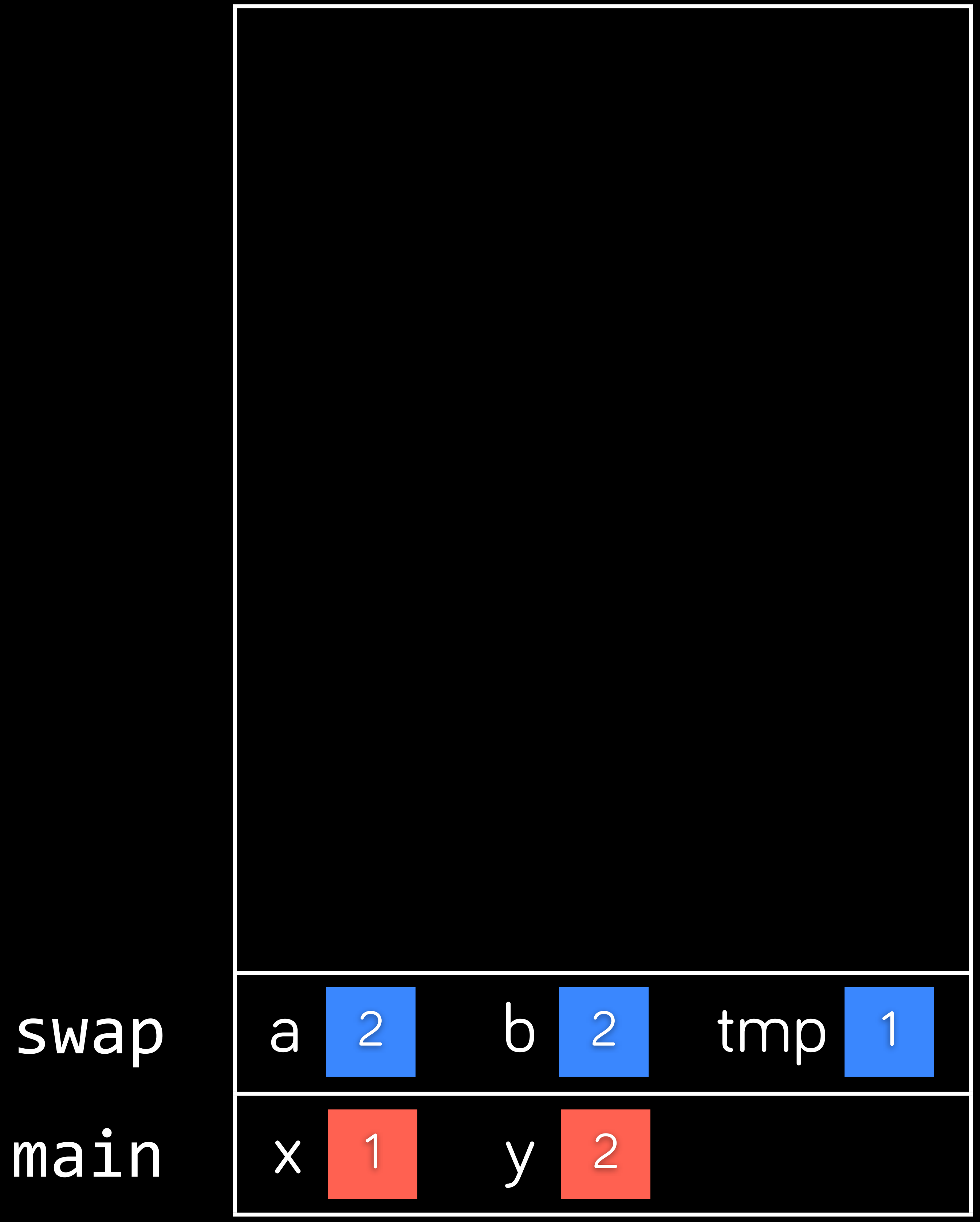
```
#include <stdio.h>
void swap(int a, int b);

int main(void)
{
    int x = 1;
    int y = 2;
    swap(x, y);
    printf("x is %i\n", x);
    printf("y is %i\n", y);
}

void swap(int a, int b)
{
    int tmp = a;
    a = b;
    b = tmp;
}
```



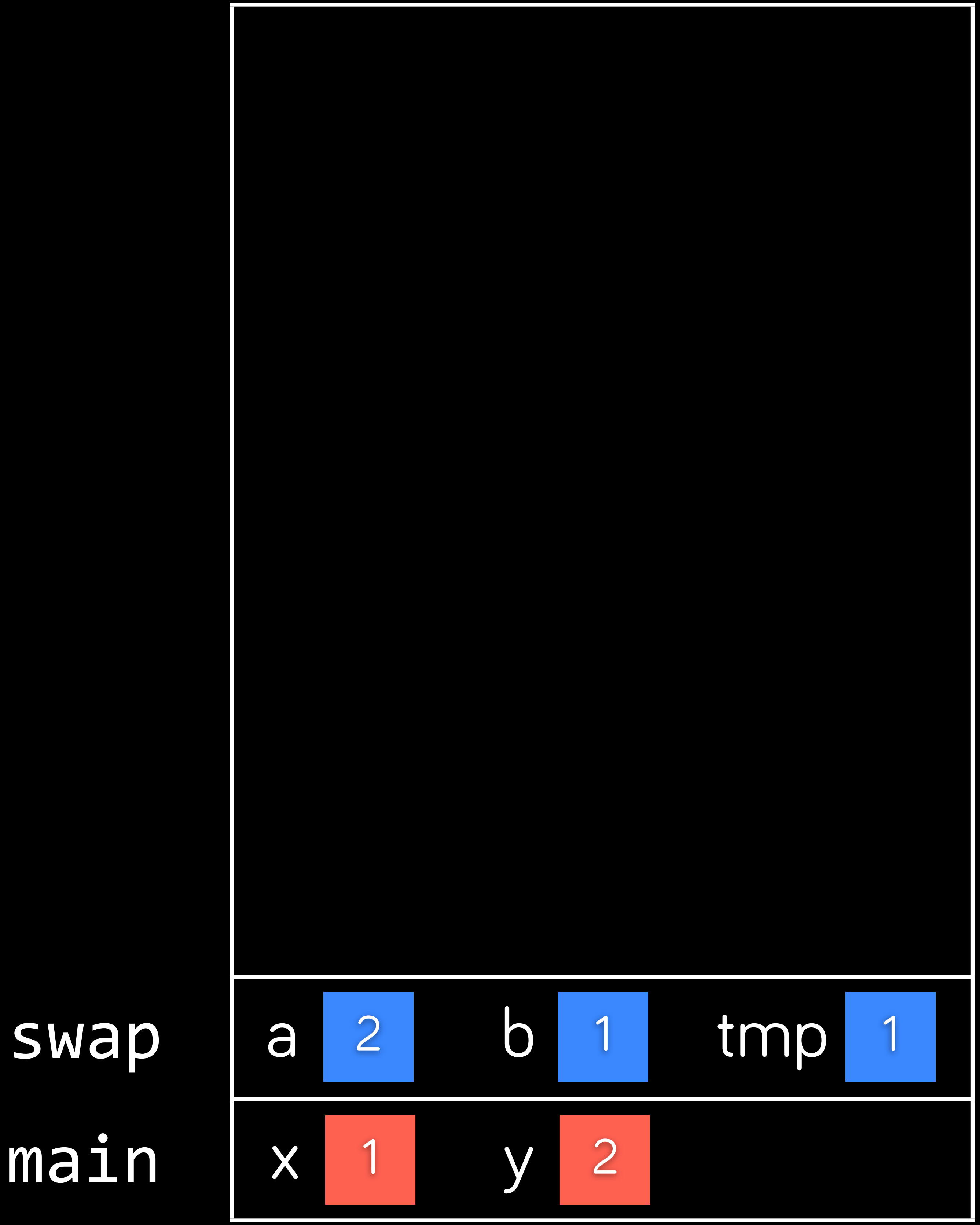
```
#include <stdio.h>
void swap(int a, int b);

int main(void)
{
    int x = 1;
    int y = 2;
    swap(x, y);
    printf("x is %i\n", x);
    printf("y is %i\n", y);
}

void swap(int a, int b)
{
    int tmp = a;
    a = b;
    b = tmp;
}
```



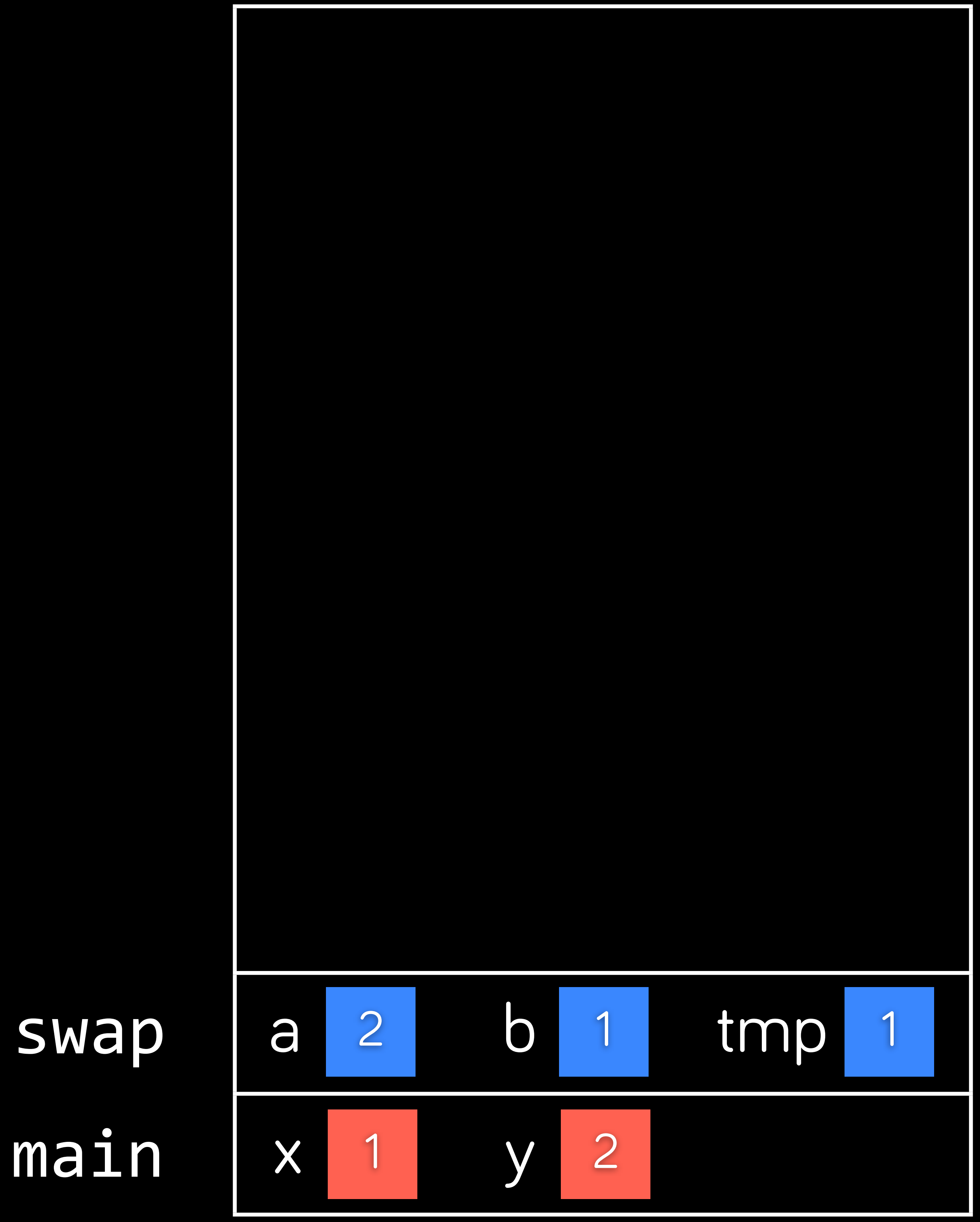
```
#include <stdio.h>
void swap(int a, int b);

int main(void)
{
    int x = 1;
    int y = 2;
    swap(x, y);
    printf("x is %i\n", x);
    printf("y is %i\n", y);
}

void swap(int a, int b)
{
    int tmp = a;
──→ a = b;
    b = tmp;
}
```



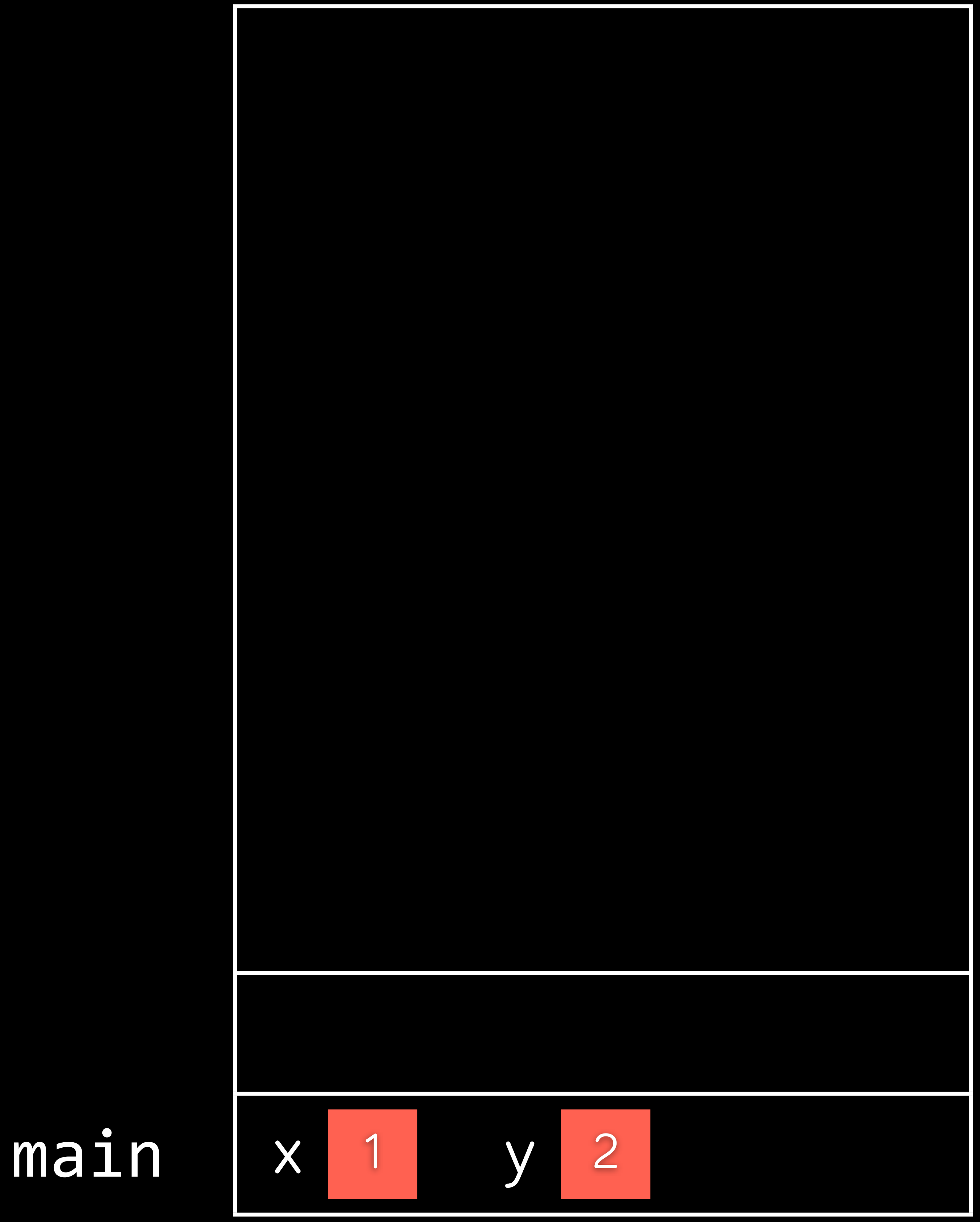
```
#include <stdio.h>
void swap(int a, int b);

int main(void)
{
    int x = 1;
    int y = 2;
    swap(x, y);
    printf("x is %i\n", x);
    printf("y is %i\n", y);
}

void swap(int a, int b)
{
    int tmp = a;
    a = b;
→   b = tmp;
}
```



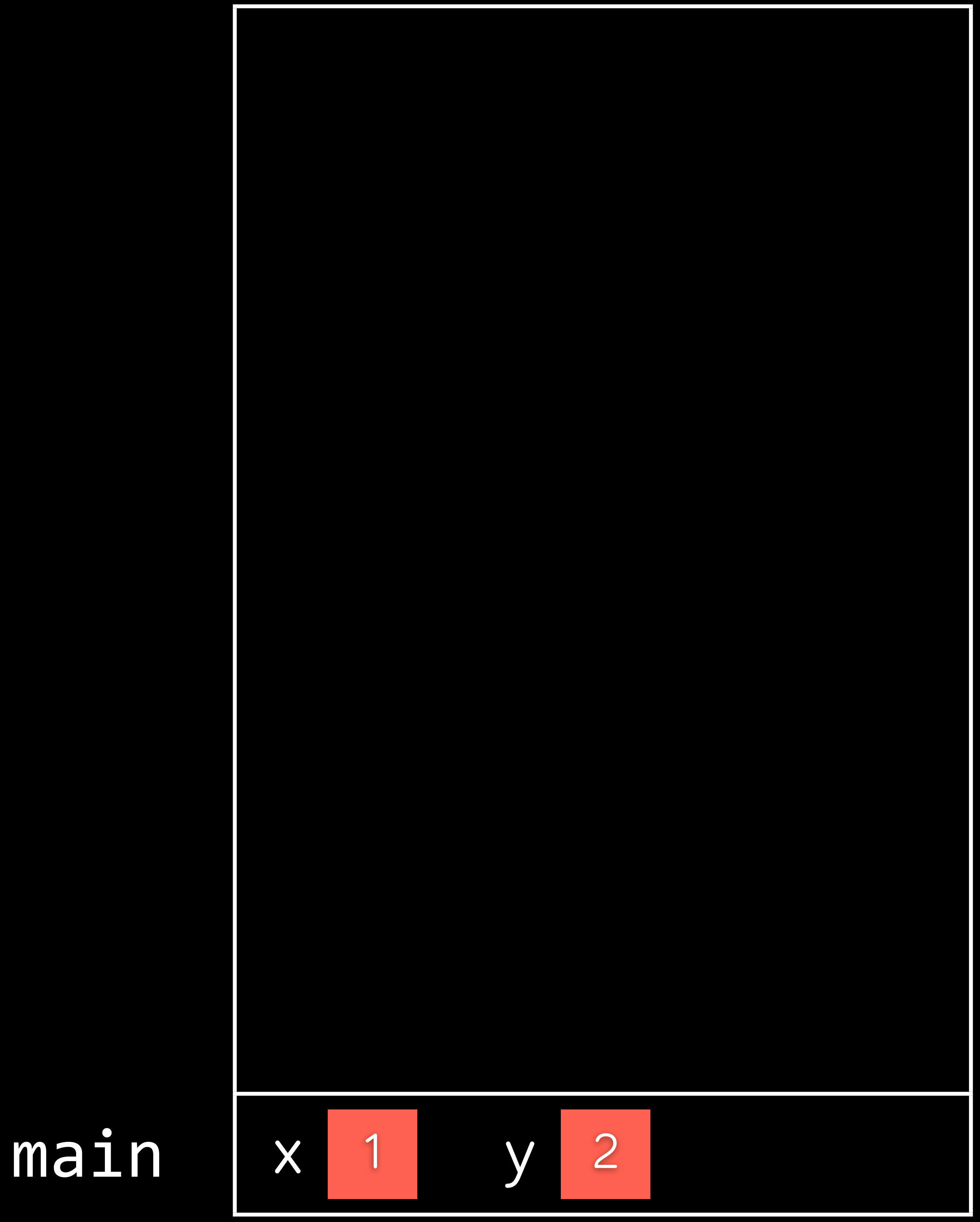
```
#include <stdio.h>
void swap(int a, int b);

int main(void)
{
    int x = 1;
    int y = 2;
    swap(x, y);
    printf("x is %i\n", x);
    printf("y is %i\n", y);
}

void swap(int a, int b)
{
    int tmp = a;
    a = b;
    b = tmp;
}
```



```c
#include <stdio.h>
void swap(int a, int b);

int main(void)
{
    int x = 1;
    int y = 2;
    swap(x, y);
    printf("x is %i\n", x);
    printf("y is %i\n", y);
}

void swap(int a, int b)
{
    int tmp = a;
    a = b;
    b = tmp;
}
```



```c
#include <stdio.h>
void swap(int a, int b);

int main(void)
{
    int x = 1;
    int y = 2;
    swap(x, y);
    printf("x is %i\n", x);
    printf("y is %i\n", y);
}

void swap(int a, int b)
{
    int tmp = a;
    a = b;
    b = tmp;
}
```



```
#include <stdio.h>
void swap(int a, int b);

int main(void)
{
    int x = 1;
    int y = 2;
    swap(x, y);
    printf("x is %i\n", x);
    printf("y is %i\n", y);
}

void swap(int a, int b)
{
    int tmp = a;
    a = b;
    b = tmp;
}
```

"passing by value"