Week 2; Lecture 2

Moving to C

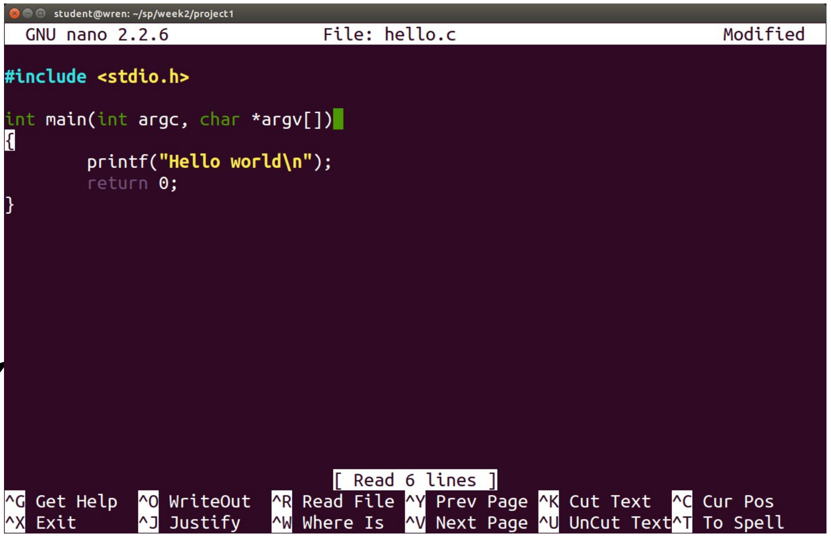
The C Language

- Why start with Scratch?
  - C Syntax is difficult
    - It obscures the behavior of the program
    - Students spend too much time learning syntax
  - C hides fewer machine details
    - You need to know a lot about what the machine is doing
  - It is harder to see what a C program is doing
    - All Scratch activities are visible

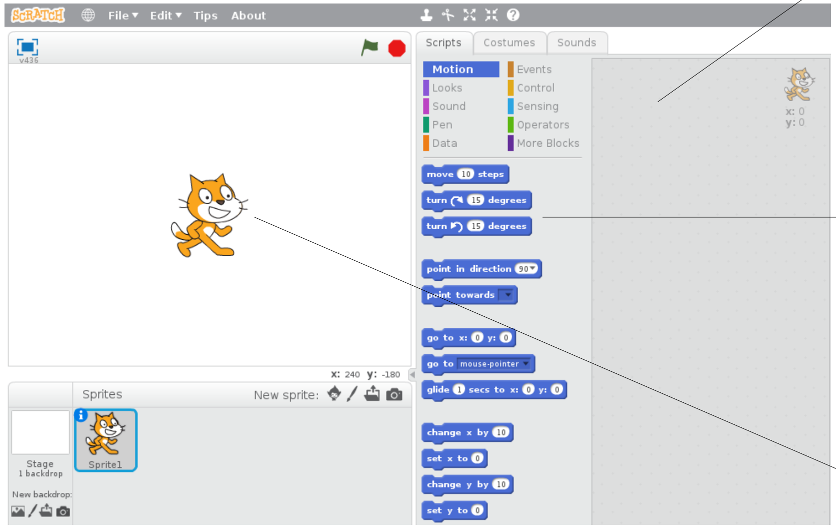
C Formatting Standards

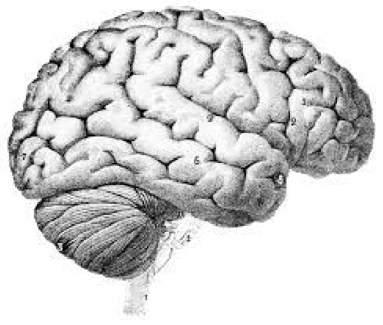
- We will be using a flexible K&R standard
  - K&R refers to Kernigan and Richie, the authors of the language.
  - They wrote the first book on C and the style they used still dominates.
  - Consistent formatting style is critical to understand other people's code.

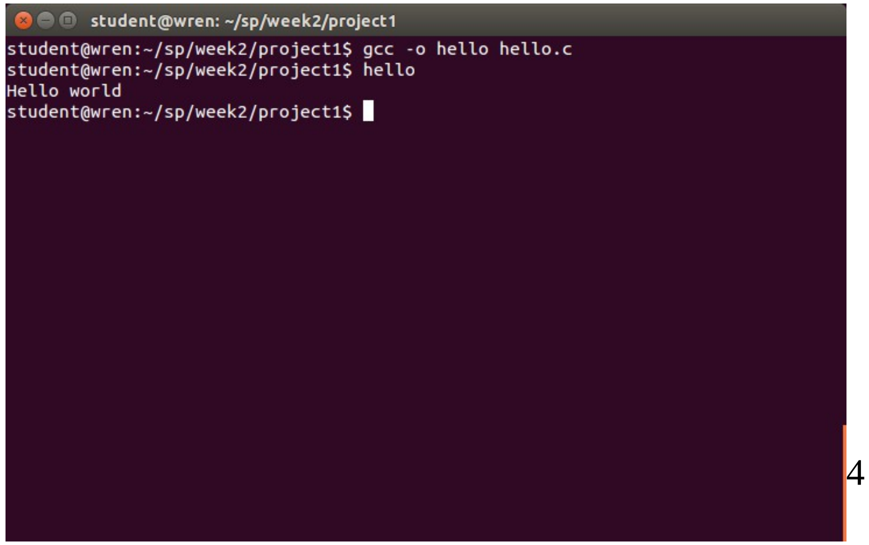
Scratch to C



```
GNU nano 2.2.6                File: hello.c                          Modified

#include <stdio.h>

int main(int argc, char *argv[])
{
        printf("Hello world\n");
        return 0;
}

                              [ Read 6 lines ]
^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text^T To Spell
```



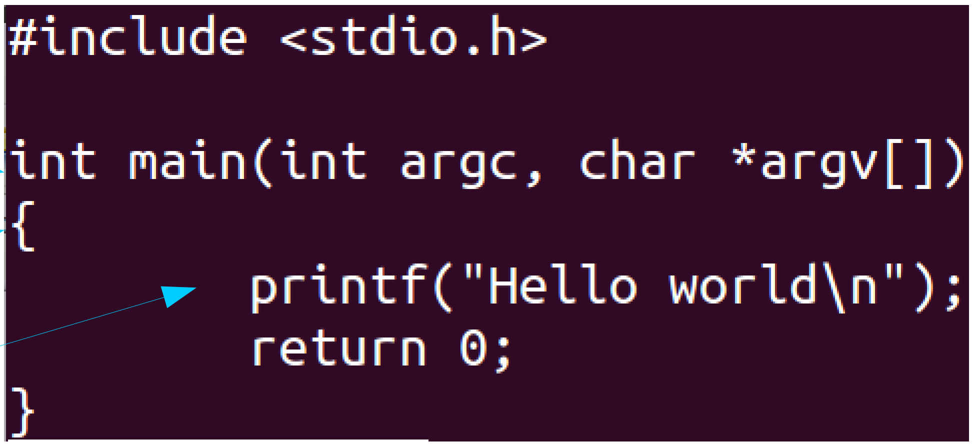




```
student@wren:~/sp/week2/project1$ gcc -o hello hello.c
student@wren:~/sp/week2/project1$ hello
Hello world
student@wren:~/sp/week2/project1$ 
```

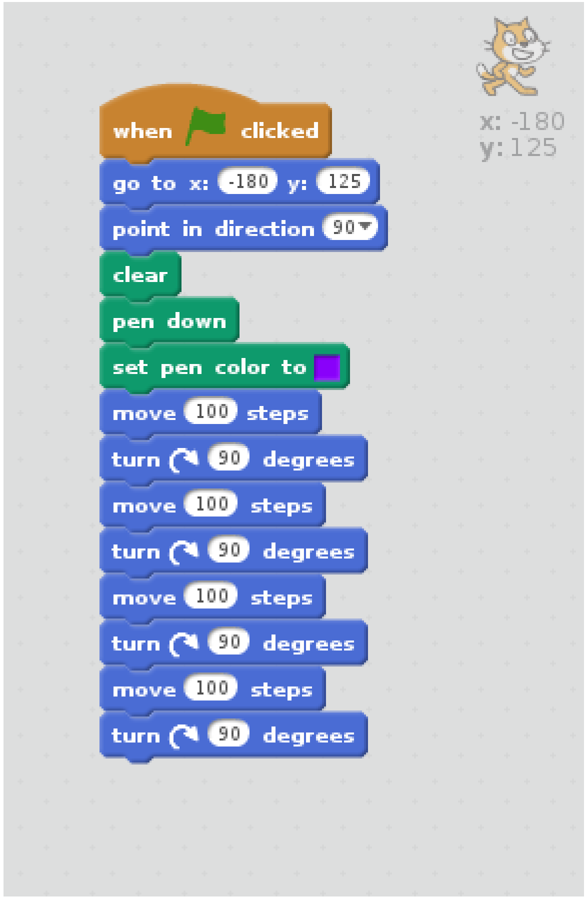
Format Rules

- Function description on separate line
- Function bracket on new line
- Indent with tab

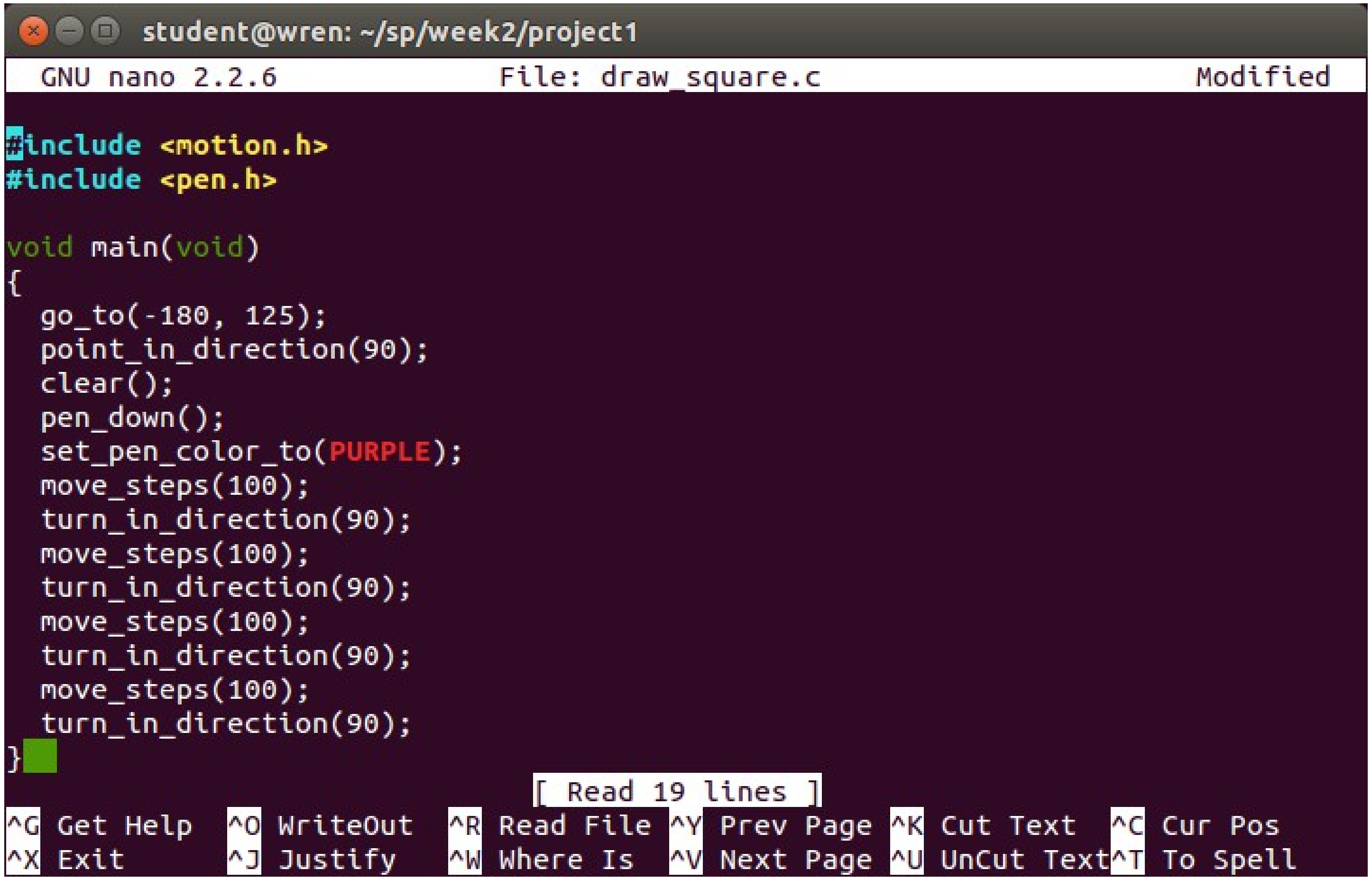


```
#include <stdio.h>

int main(int argc, char *argv[])
{
	printf("Hello world\n");
	return 0;
}
```

Draw a Square 1



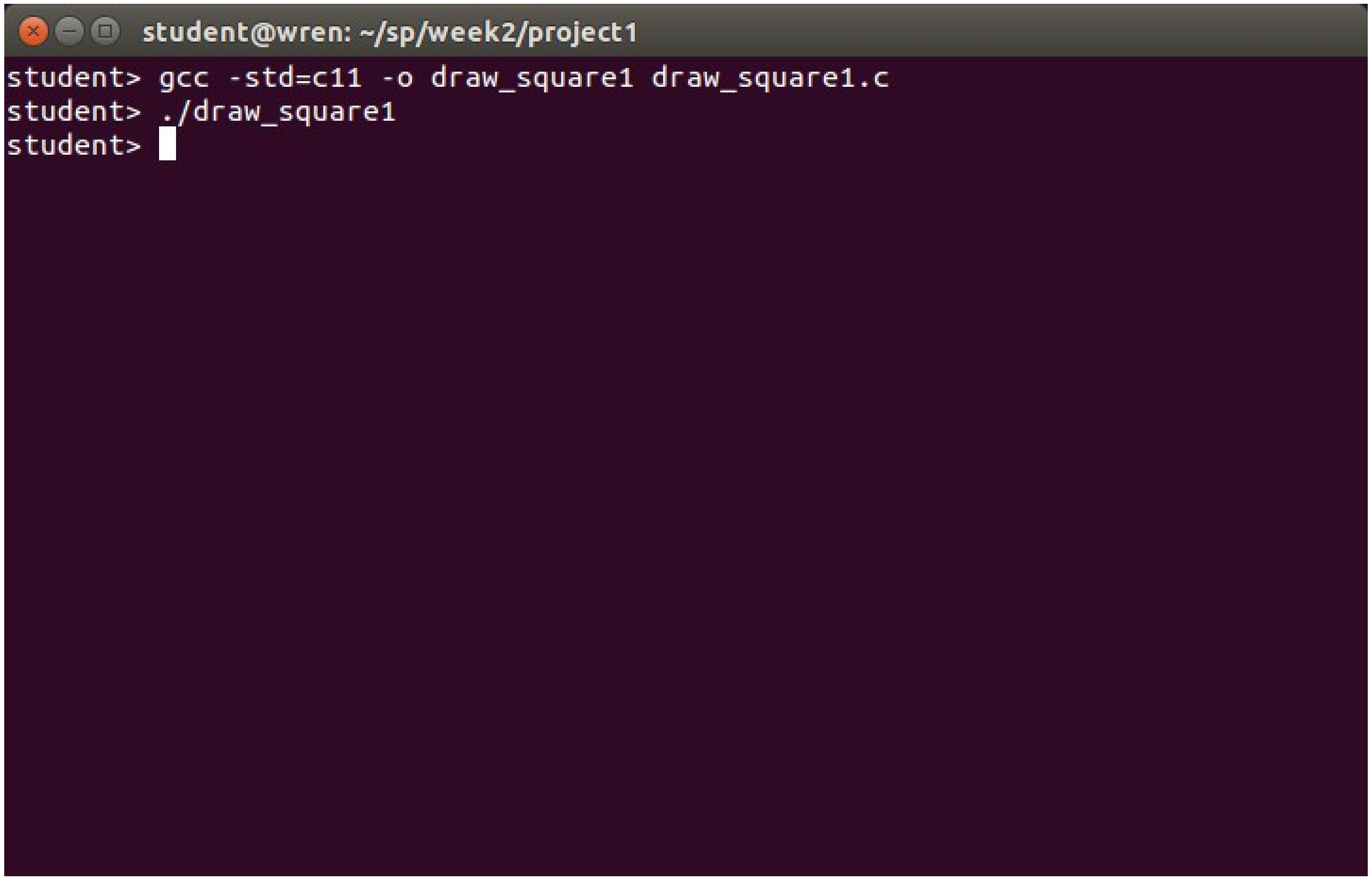
```
when [flag] clicked
go to x: -180 y: 125
point in direction 90
clear
pen down
set pen color to
move 100 steps
turn 90 degrees
move 100 steps
turn 90 degrees
move 100 steps
turn 90 degrees
move 100 steps
turn 90 degrees
```



```
student@wren: ~/sp/week2/project1
  GNU nano 2.2.6              File: draw_square.c                      Modified

#include <motion.h>
#include <pen.h>

void main(void)
{
  go_to(-180, 125);
  point_in_direction(90);
  clear();
  pen_down();
  set_pen_color_to(PURPLE);
  move_steps(100);
  turn_in_direction(90);
  move_steps(100);
  turn_in_direction(90);
  move_steps(100);
  turn_in_direction(90);
  move_steps(100);
  turn_in_direction(90);
}
                               [ Read 19 lines ]
^G Get Help   ^O WriteOut   ^R Read File  ^Y Prev Page  ^K Cut Text   ^C Cur Pos
^X Exit       ^J Justify    ^W Where Is   ^V Next Page  ^U UnCut Text ^T To Spell
```

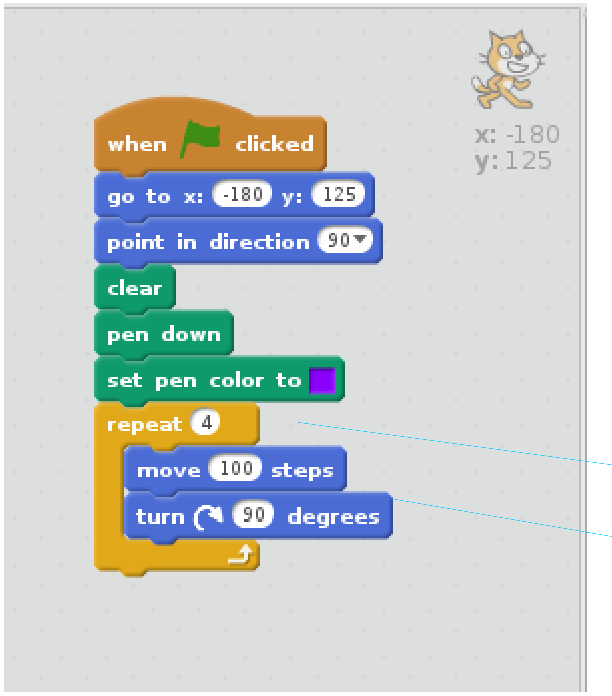
Running the Program

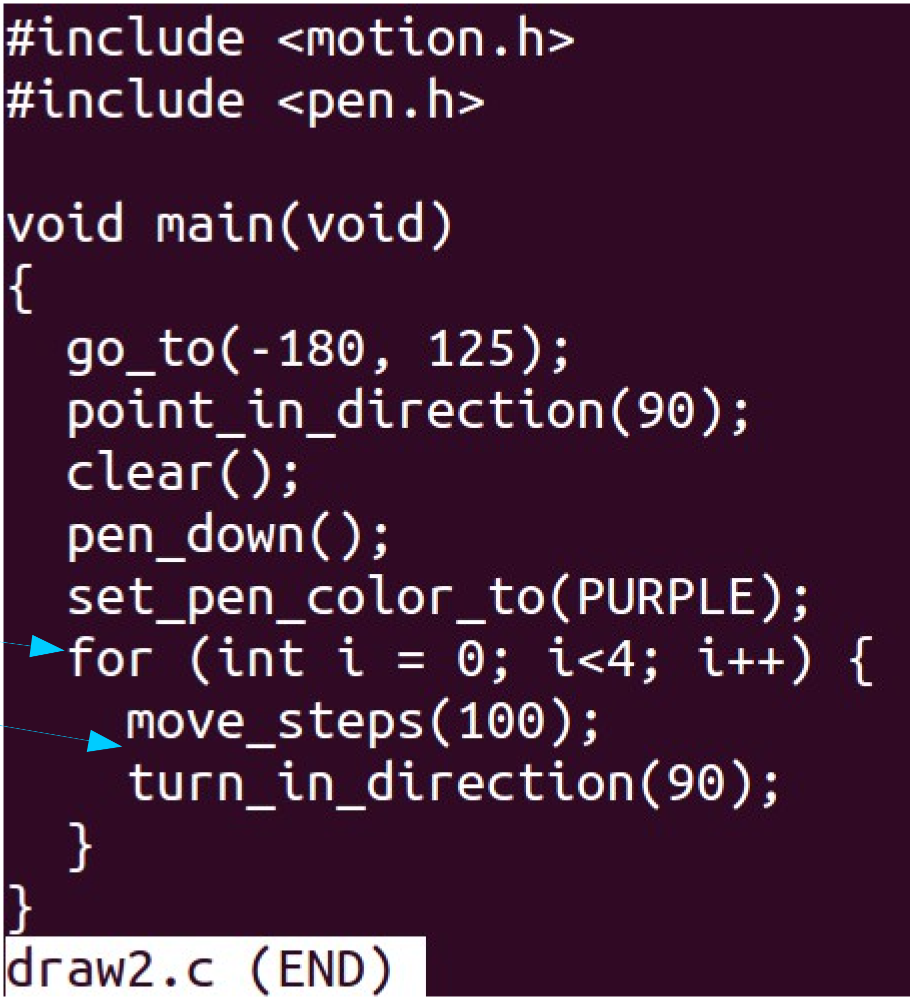
when clicked



```
student> gcc -std=c11 -o draw_square1 draw_square1.c
student> ./draw_square1
student> 
```

Draw a Square 2

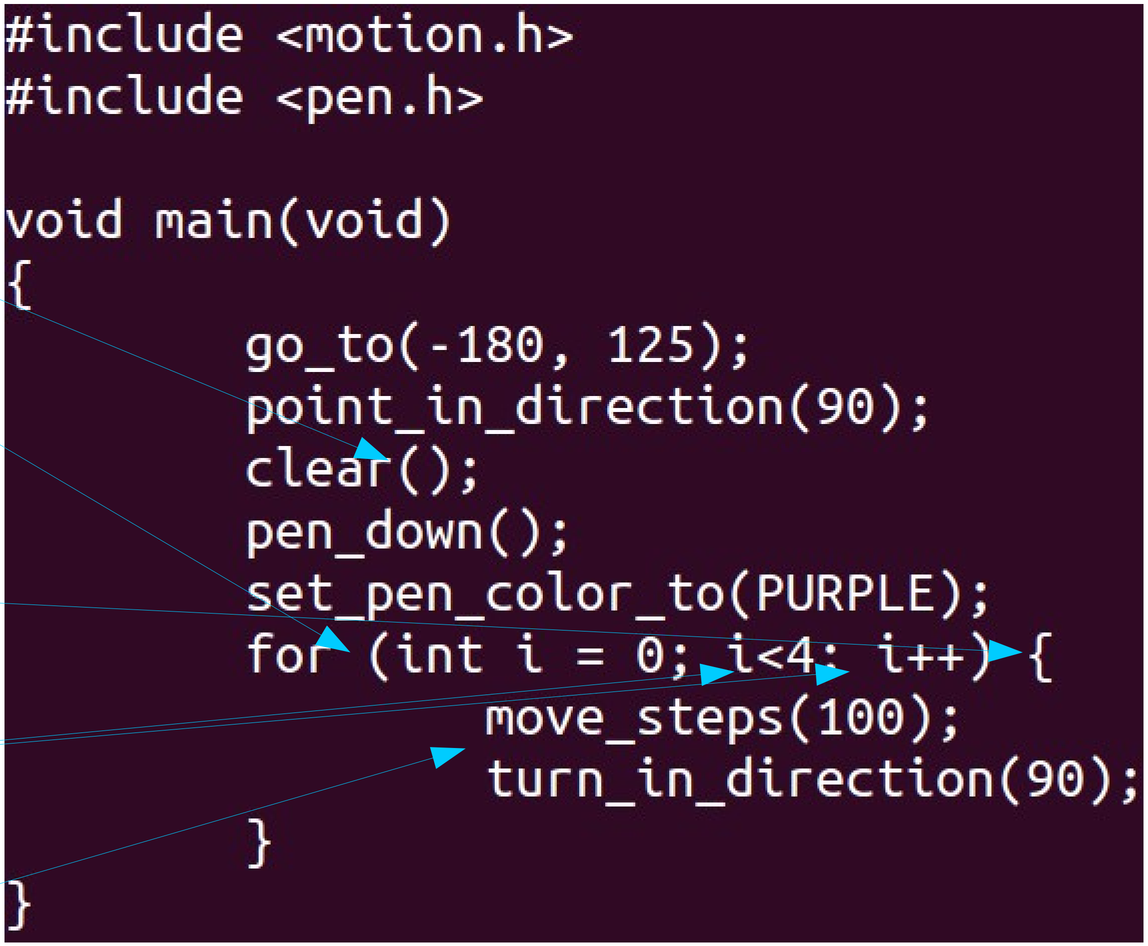




```
#include <motion.h>
#include <pen.h>

void main(void)
{
  go_to(-180, 125);
  point_in_direction(90);
  clear();
  pen_down();
  set_pen_color_to(PURPLE);
  for (int i = 0; i<4; i++) {
    move_steps(100);
    turn_in_direction(90);
  }
}
draw2.c (END)
```

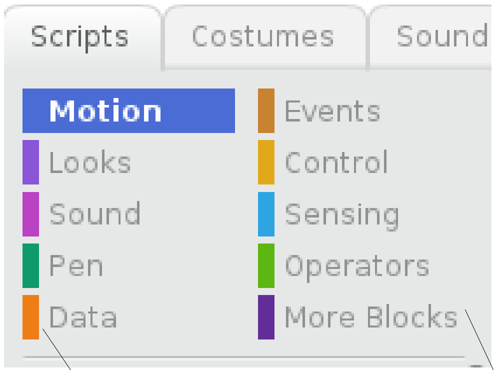
Format Rules

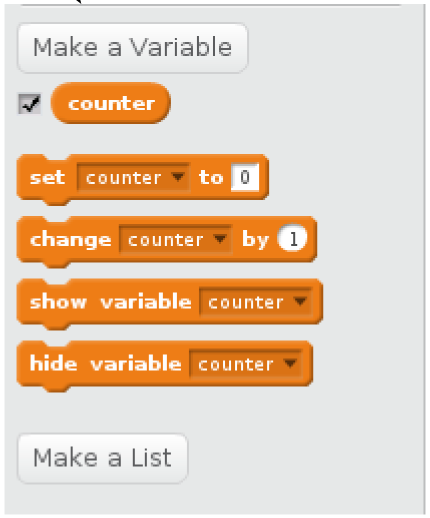
- No space after function name
- Space after for
- Bracket on same line
- Space after semi-colon
- Indent two tabs

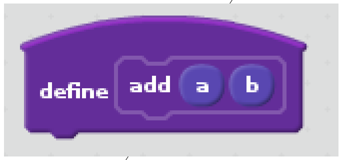


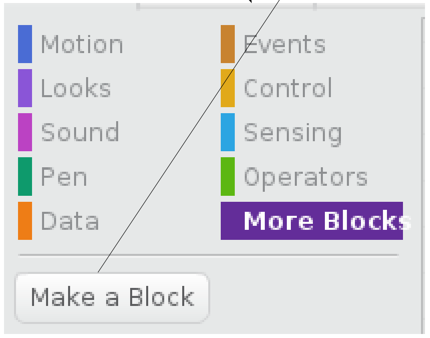
```
#include <motion.h>
#include <pen.h>

void main(void)
{
		go_to(-180, 125);
		point_in_direction(90);
		clear();
		pen_down();
		set_pen_color_to(PURPLE);
		for (int i = 0; i<4; i++) {
				move_steps(100);
				turn_in_direction(90);
		}
}
```

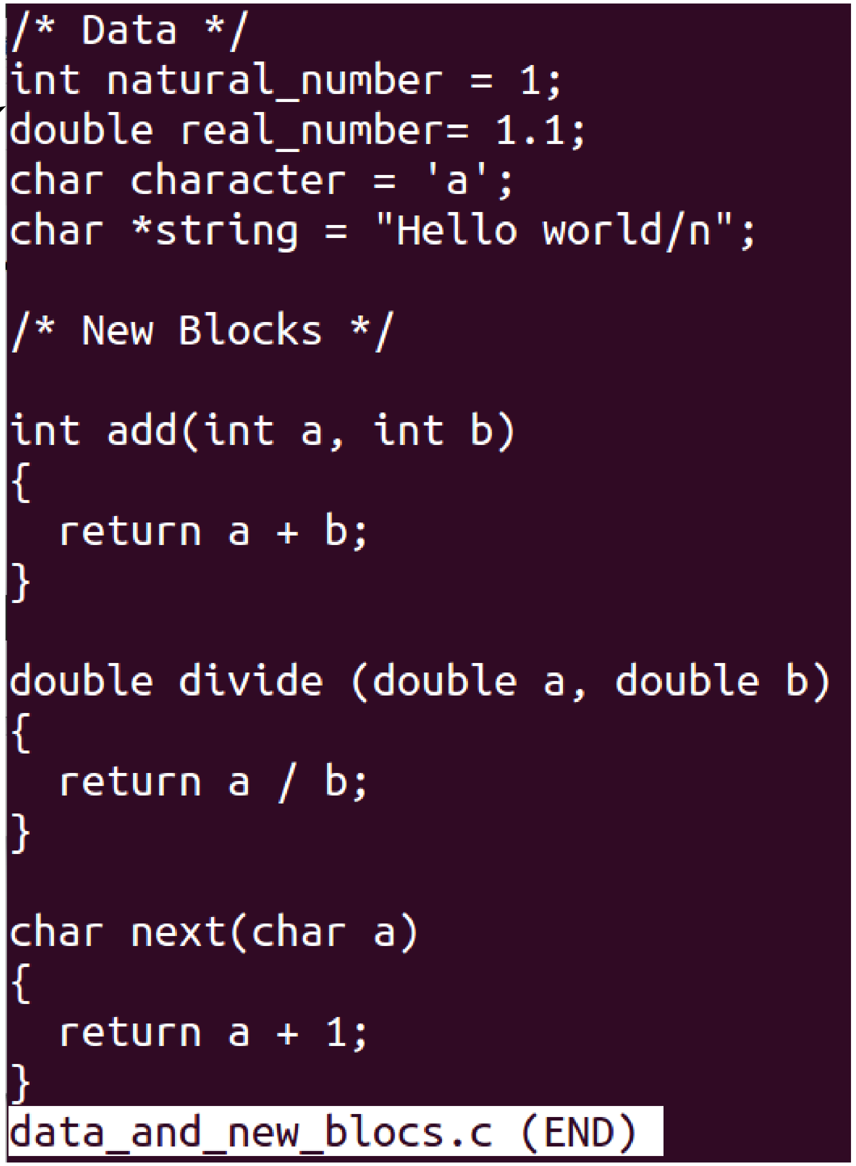
Data and New Blocks





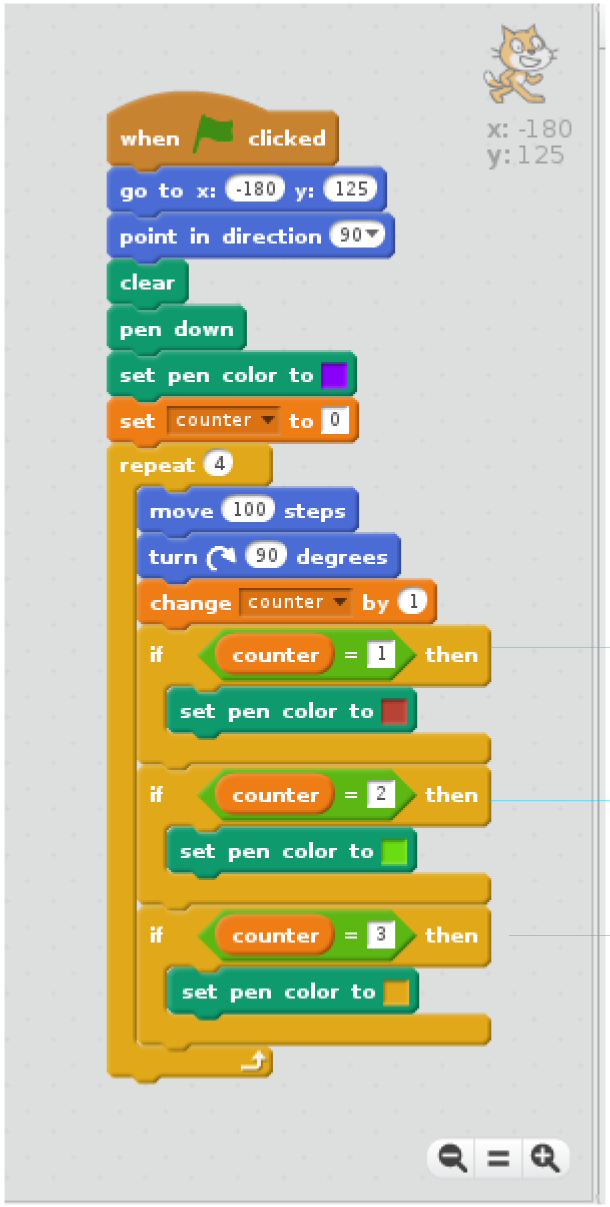






```
/* Data */
int natural_number = 1;
double real_number= 1.1;
char character = 'a';
char *string = "Hello world/n";

/* New Blocks */

int add(int a, int b)
{
  return a + b;
}

double divide (double a, double b)
{
  return a / b;
}

char next(char a)
{
  return a + 1;
}
data_and_new_blocs.c (END)
```

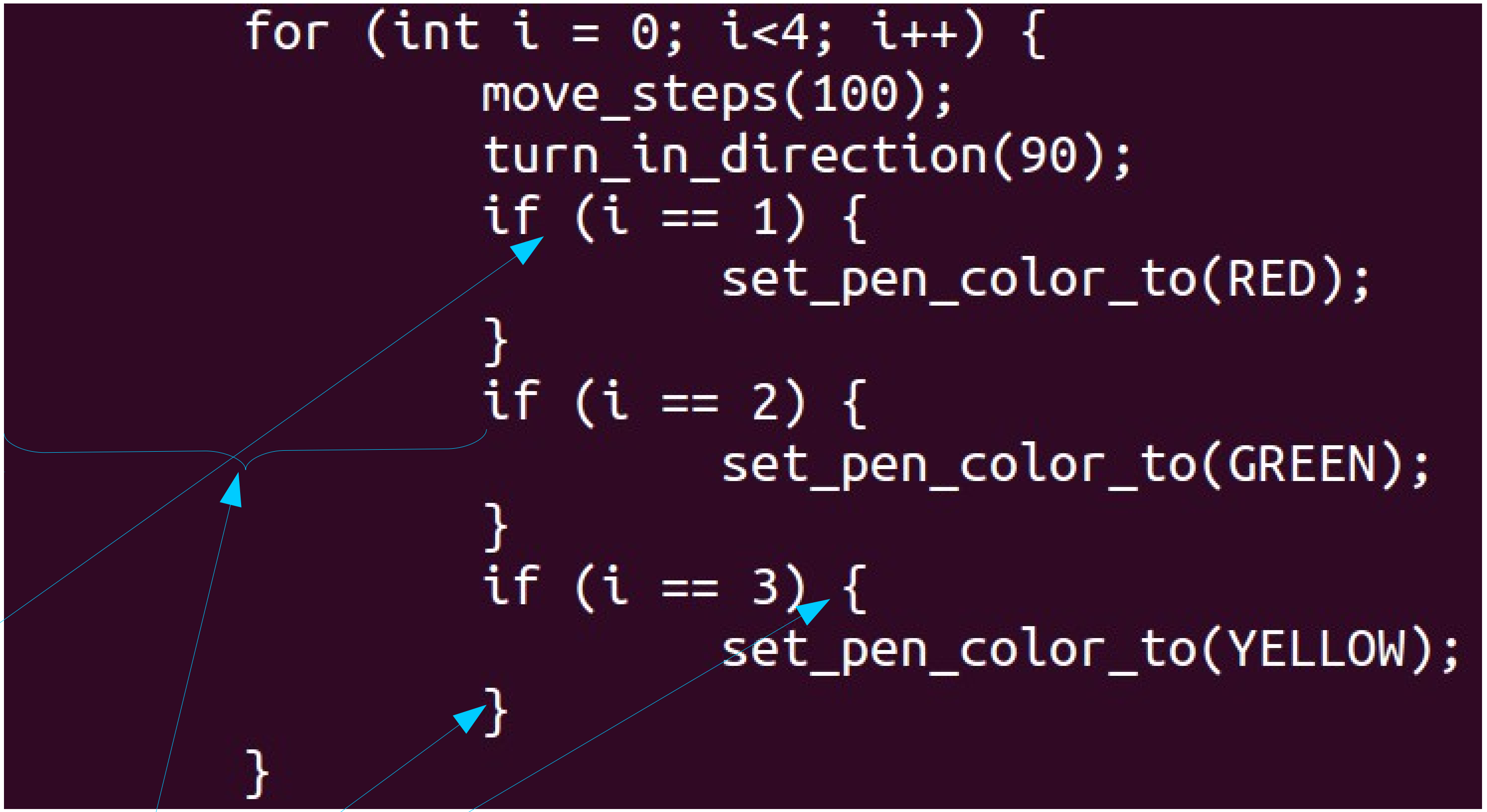
Draw a Square 3



```
when 🏴 clicked
go to x: -180 y: 125
point in direction 90
clear
pen down
set pen color to 🟪
set counter to 0
repeat 4
    move 100 steps
    turn ↻ 90 degrees
    change counter by 1
    if counter = 1 then
        set pen color to 🟥
    if counter = 2 then
        set pen color to 🟩
    if counter = 3 then
        set pen color to 🟨
```

```
#include <motion.h>
#include <pen.h>

void main(void)
{
        go_to(-180, 125);
        point_in_direction(90);
        clear();
        pen_down();
        set_pen_color_to(PURPLE);
        for (int i = 0; i<4; i++) {
                move_steps(100);
                turn_in_direction(90);
                if (i == 1) {
                        set_pen_color_to(RED);
                }
                if (i == 2) {
                        set_pen_color_to(GREEN);
                }
                if (i == 3) {
                        set_pen_color_to(YELLOW);
                }
        }
}
```

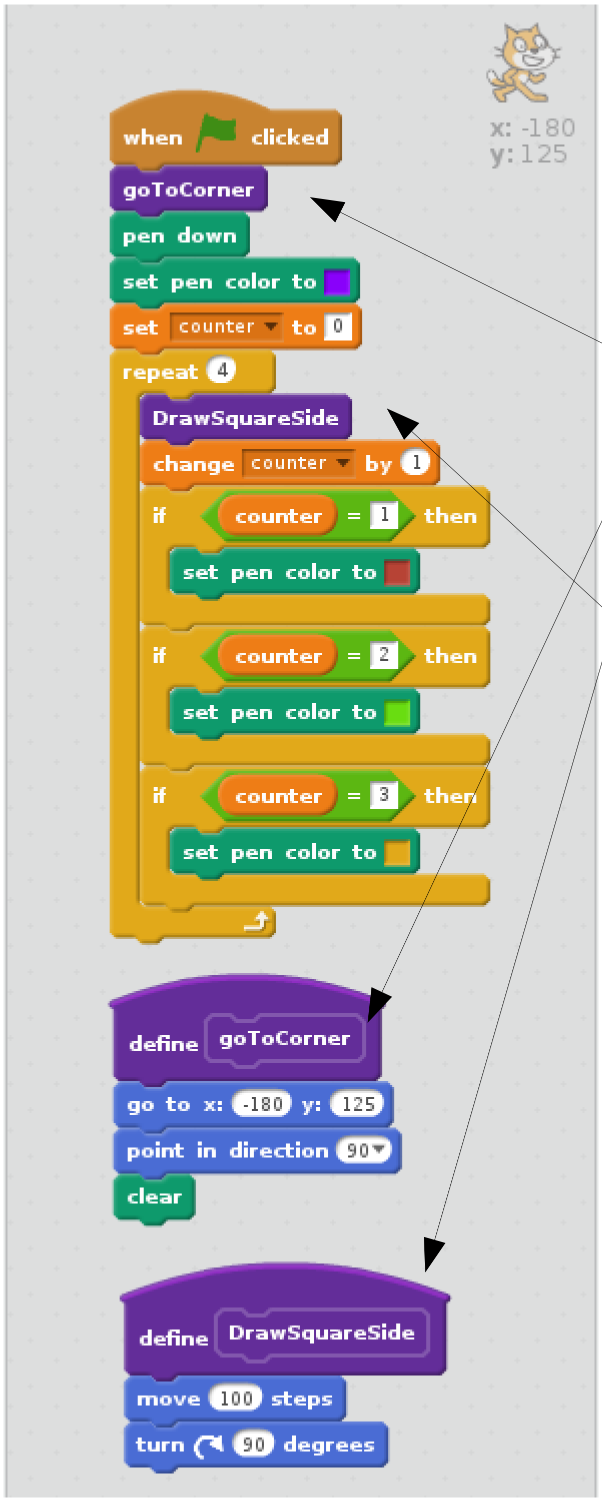
Format Rules



```
	for (int i = 0; i<4; i++) {
			move_steps(100);
			turn_in_direction(90);
			if (i == 1) {
				set_pen_color_to(RED);
			}
			if (i == 2) {
				set_pen_color_to(GREEN);
			}
			if (i == 3) {
				set_pen_color_to(YELLOW);
			}
	}
```

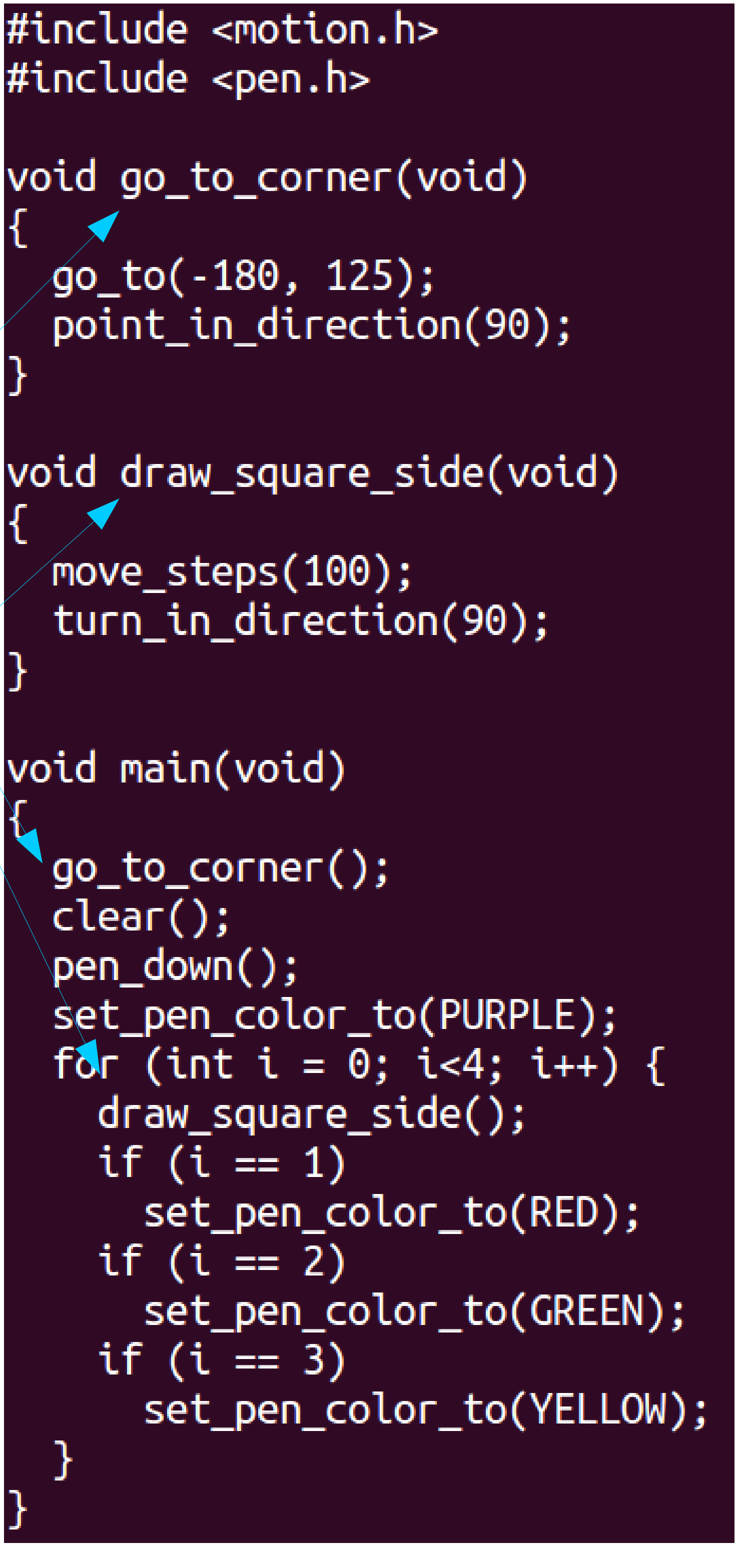
- Space
- Indent three tabs
- Use brackets

a Square 4



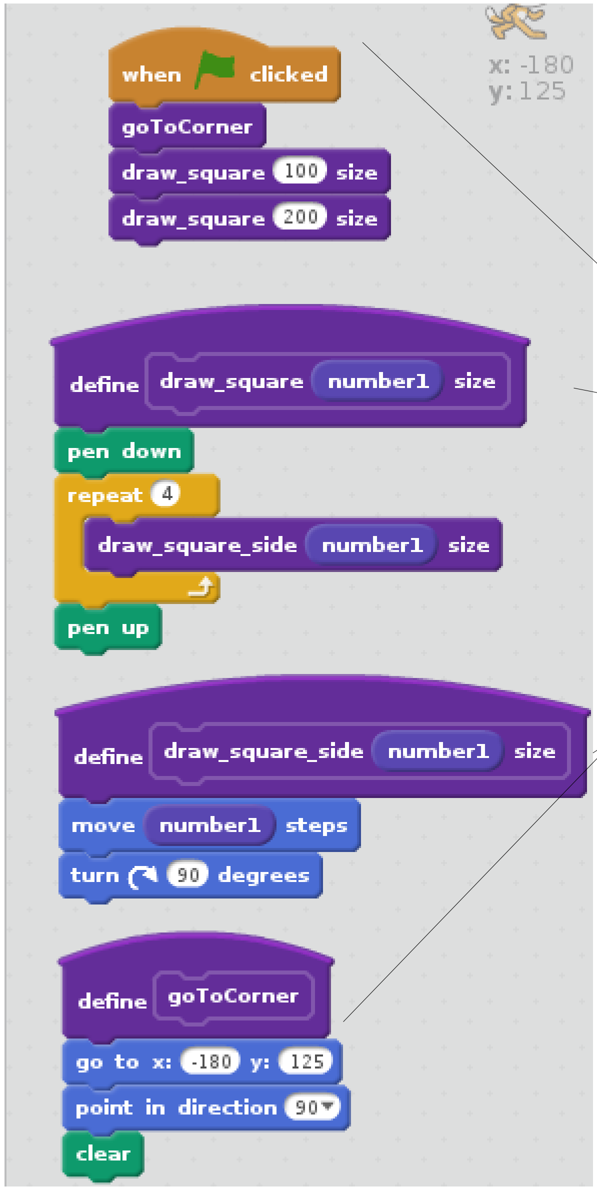
```
x: -180
y: 125

when 🏁 clicked
goToCorner
pen down
set pen color to 🟪
set counter to 0
repeat 4
    DrawSquareSide
    change counter by 1
    if counter = 1 then
        set pen color to 🟥
    if counter = 2 then
        set pen color to 🟩
    if counter = 3 then
        set pen color to 🟨

define goToCorner
go to x: -180 y: 125
point in direction 90
clear

define DrawSquareSide
move 100 steps
turn ↻ 90 degrees
```

- Call goToCorner
- Define goToCorner
- Call drawSquareSide
- Define drawSquareSide

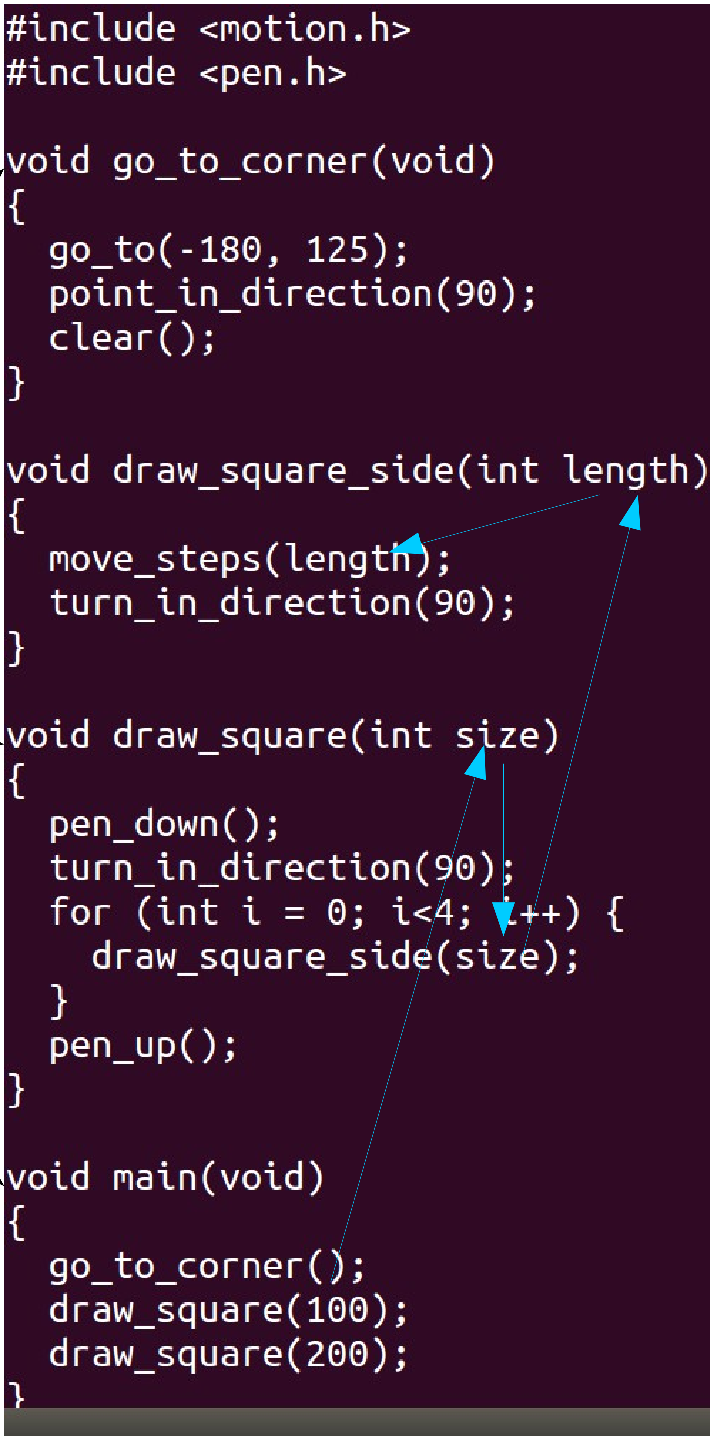


```
#include <motion.h>
#include <pen.h>

void go_to_corner(void)
{
  go_to(-180, 125);
  point_in_direction(90);
}

void draw_square_side(void)
{
  move_steps(100);
  turn_in_direction(90);
}

void main(void)
{
  go_to_corner();
  clear();
  pen_down();
  set_pen_color_to(PURPLE);
  for (int i = 0; i<4; i++) {
    draw_square_side();
    if (i == 1)
      set_pen_color_to(RED);
    if (i == 2)
      set_pen_color_to(GREEN);
    if (i == 3)
      set_pen_color_to(YELLOW);
  }
}
```

Draw Squares

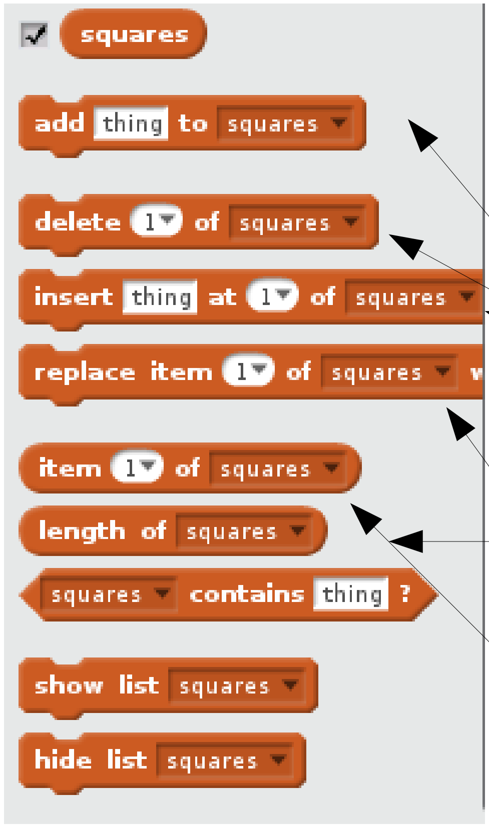


```
when [flag] clicked
goToCorner
draw_square (100) size
draw_square (200) size

define draw_square (number1) size
pen down
repeat (4)
    draw_square_side (number1) size
pen up

define draw_square_side (number1) size
move (number1) steps
turn ↻ (90) degrees

define goToCorner
go to x: (-180) y: (125)
point in direction (90▾)
clear
```



```
#include <motion.h>
#include <pen.h>

void go_to_corner(void)
{
  go_to(-180, 125);
  point_in_direction(90);
  clear();
}

void draw_square_side(int length)
{
  move_steps(length);
  turn_in_direction(90);
}

void draw_square(int size)
{
  pen_down();
  turn_in_direction(90);
  for (int i = 0; i<4; i++) {
    draw_square_side(size);
  }
  pen_up();
}

void main(void)
{
  go_to_corner();
  draw_square(100);
  draw_square(200);
}
```

Draw Squares 2

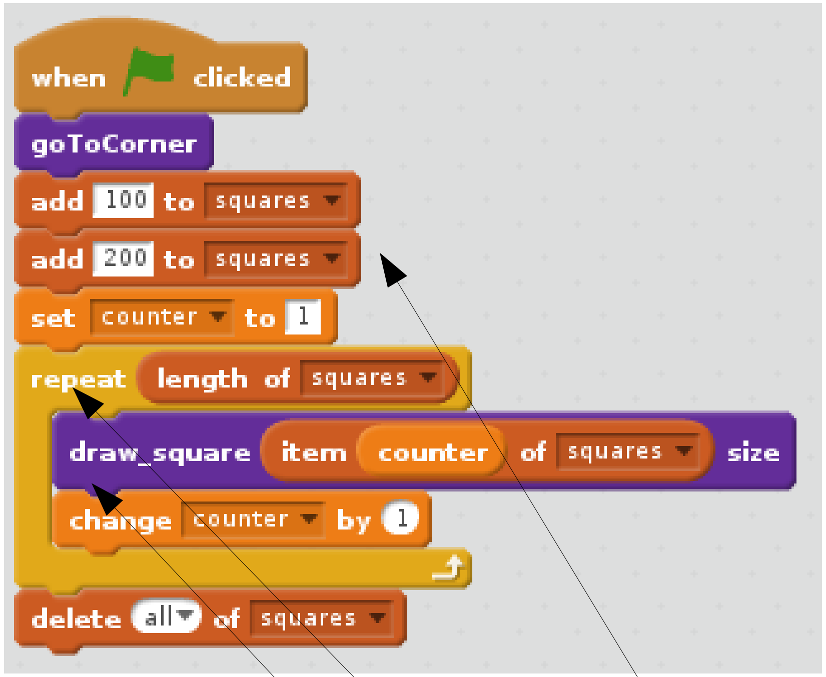


- In C
  - Indicate list contents
  - Indicate max size of list
  - Cannot add, delete, or insert
  - No way to find out length
  - Can
    - Replace
    - Access item

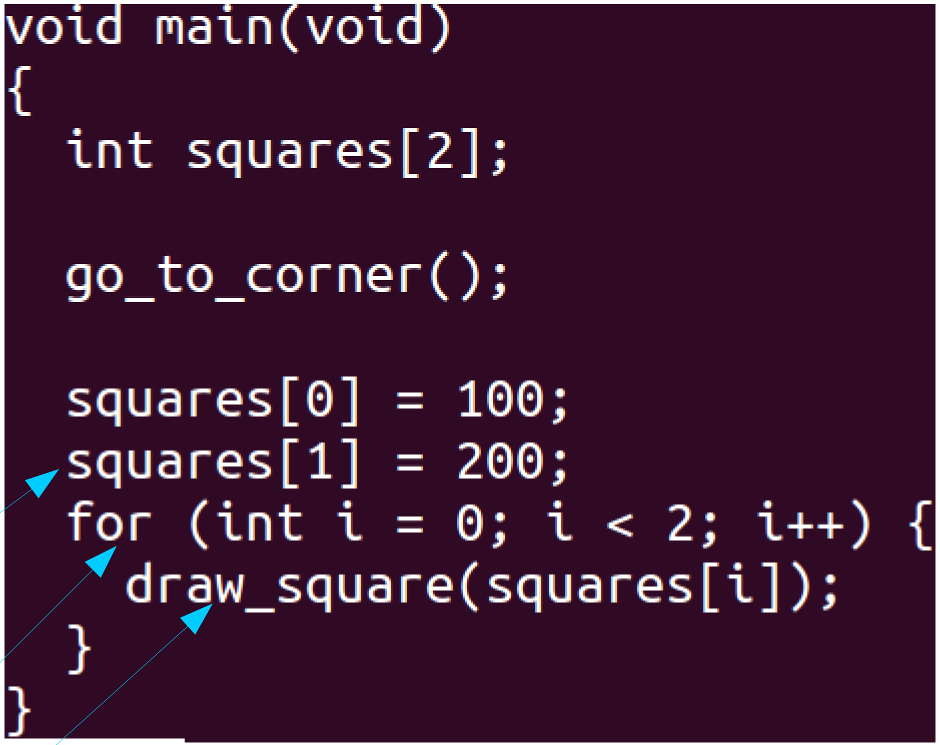
```
int squares[2];
```

```
squares[0] = 123;

squares[1] + squares[2];
```

Draw Squares 2



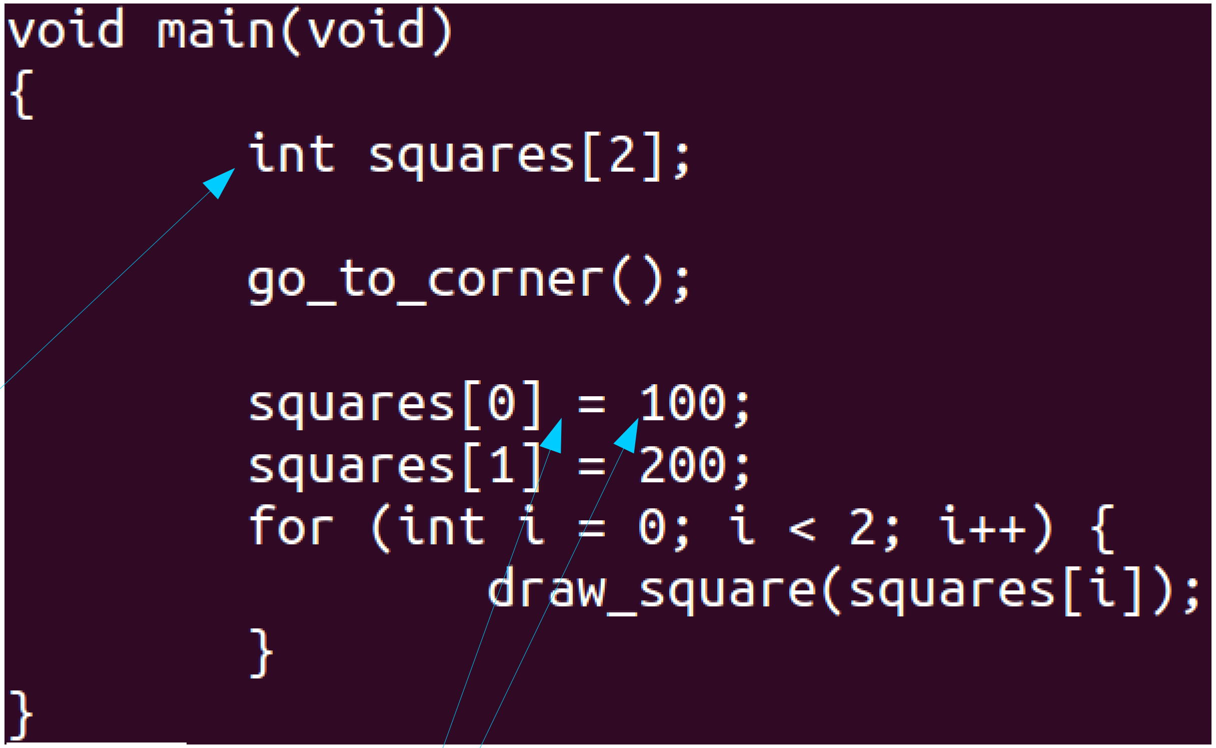
```
when [flag] clicked
goToCorner
add 100 to squares
add 200 to squares
set counter to 1
repeat length of squares
    draw_square item counter of squares size
    change counter by 1
delete all of squares
```



```
void main(void)
{
  int squares[2];

  go_to_corner();

  squares[0] = 100;
  squares[1] = 200;
  for (int i = 0; i < 2; i++) {
    draw_square(squares[i]);
  }
}
```

- Initialize
  - Squares ← {100, 200}
  - Counter ← 1
- Loop through Squares
- Drawing a new square each time

Format Rules

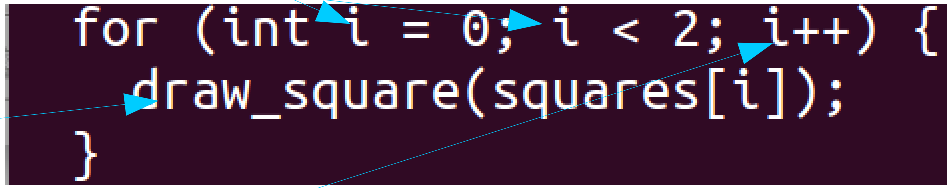


```
void main(void)
{
        int squares[2];

        go_to_corner();

        squares[0] = 100;
        squares[1] = 200;
        for (int i = 0; i < 2; i++) {
                draw_square(squares[i]);
        }
}
```

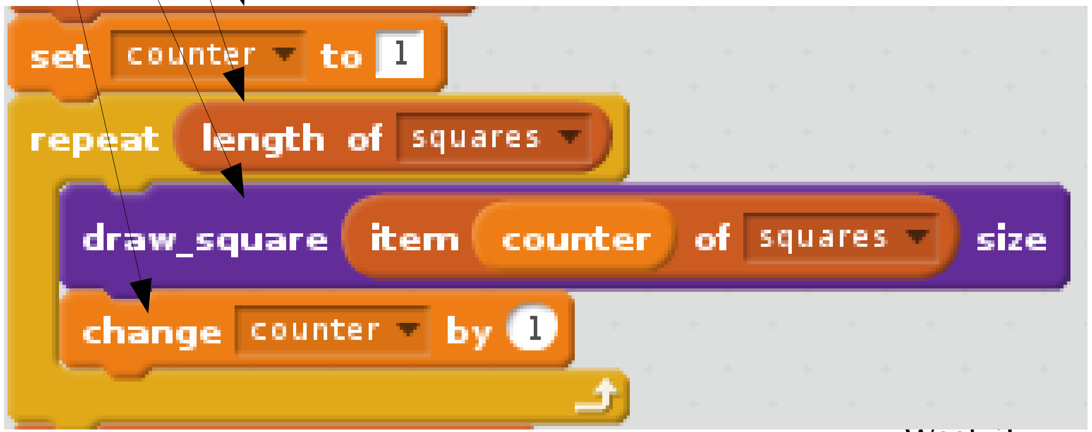
- One variable per line
- Space around = sign in assignment

Looping through a list

- Initialize counter
- Repeat to end of list
- Action
- Increment counter



```
for (int i = 0; i < 2; i++) {
  draw_square(squares[i]);
}
```



```
set counter to 1
repeat (length of squares)
  draw_square (item (counter) of squares) size
  change counter by 1
```