APS105
Winter 2012

Jonathan Deber
jdeber -at- cs -dot- toronto -dot- edu

Lecture 12
February 10, 2012

Today

- Function/scope odds and ends
- Built-in functions

- Assignment 2

return

- Functions with a return type must have `return` *expression*
- Function immediately exits

```
double sum(double x, double y)
{
    double sum = x + y;
    return sum;
    sum += x;
    return sum;
}
```

} Never executed

```
int max(int a, int b)
{
  if (a > b)
  {
    return a;
  }
  else
  {
    return b;
  }
}
```

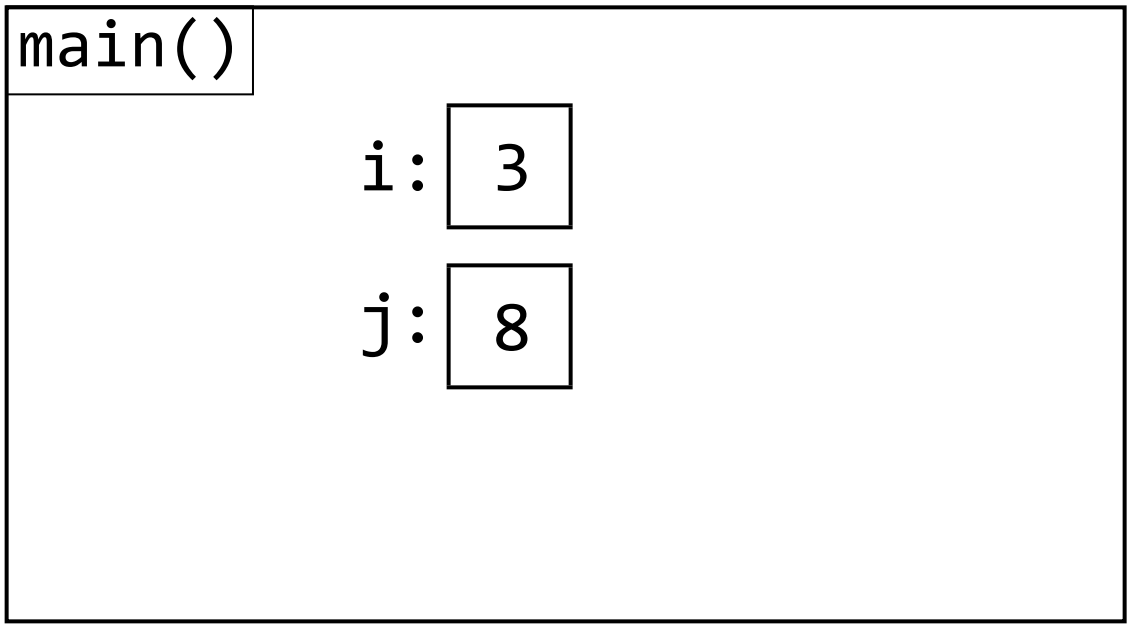
```
int max(int a, int b)
{
  int result;
  if (a > b)
  {
    result = a;
  }
  else
  {
    result = b;
  }
  return result;
}
```

```
int max(int a, int b)
{
  if (a > b)
  {
    return a;
  }
  else
  {
    return b;
  }
}
```

Error

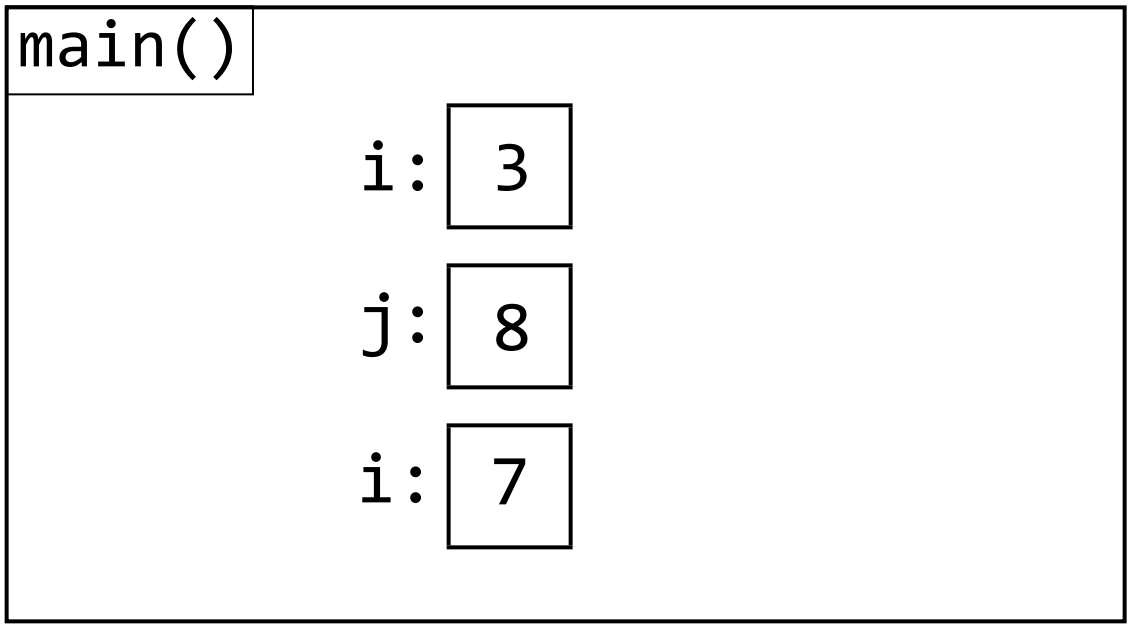
```
int max(int a, int b)
{
  if (a > b)
  {
    return a;
  }
  else if (a < b)
  {
    return b;
  }
}
```

```c
#define N 5


int main(void)
{
    int i = 3;
    int j = 8;

    printf("%d\n", i);









    printf("%d\n", i);

    return 0;
}
```

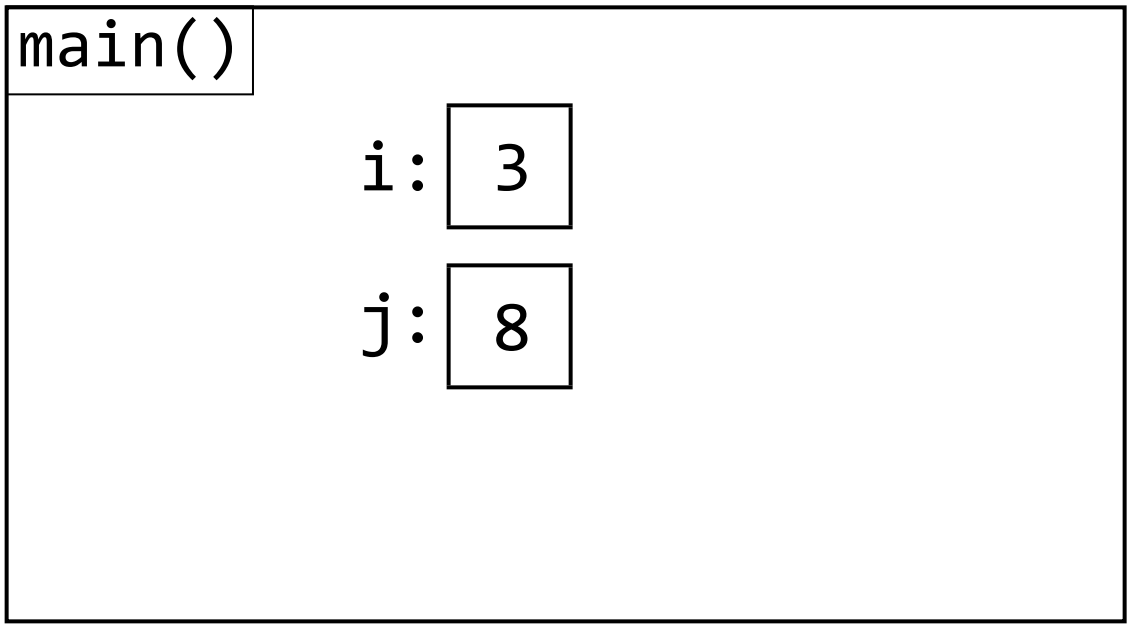


```
#define N 5


int main(void)
{
    int i = 3;
    int j = 8;

    printf("%d\n", i);
    int i = 7;


    printf("%d\n", i);

    return 0;
}
```

Error

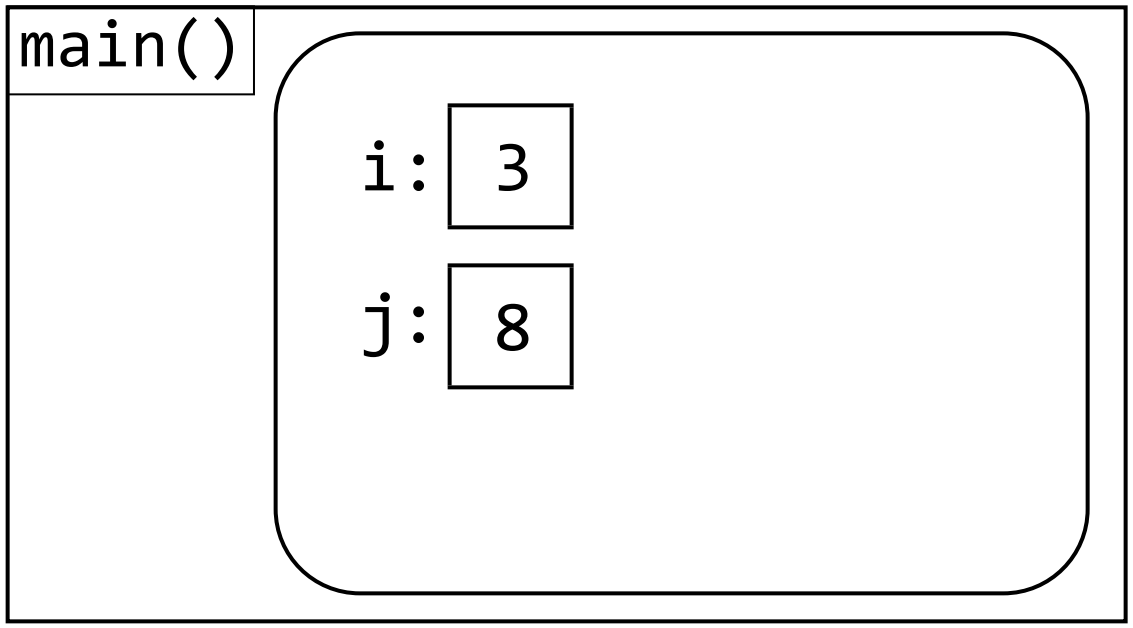


```c
#define N 5


int main(void)
{
    int i = 3;
    int j = 8;

    printf("%d\n", i);

    for (int i = 0; i < N; i++)
    {
        printf("* %d * \n", i);
    }


    printf("%d\n", i);

    return 0;
}
```



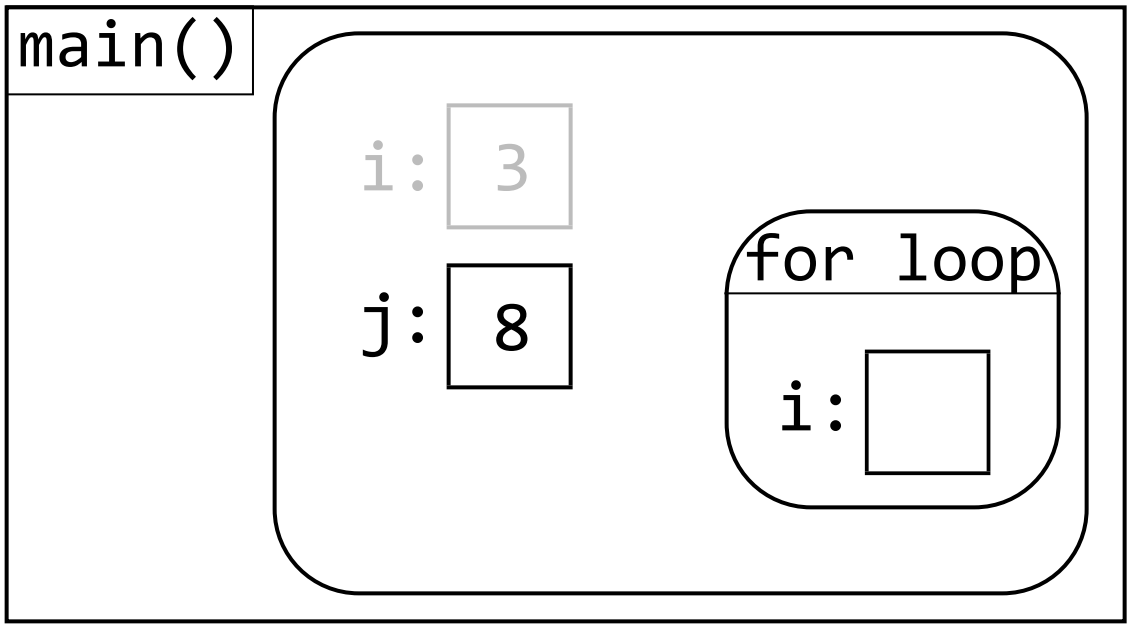

```
3
* 0 *
* 1 *
* 2 *
* 3 *
* 4 *
3
```

```c
#define N 5


int main(void)
{
    int i = 3;
    int j = 8;

    printf("%d\n", i);

    for (int i = 0; i < N; i++)
    {
        printf("* %d * \n", i);
    }


    printf("%d\n", i);

    return 0;
}
```



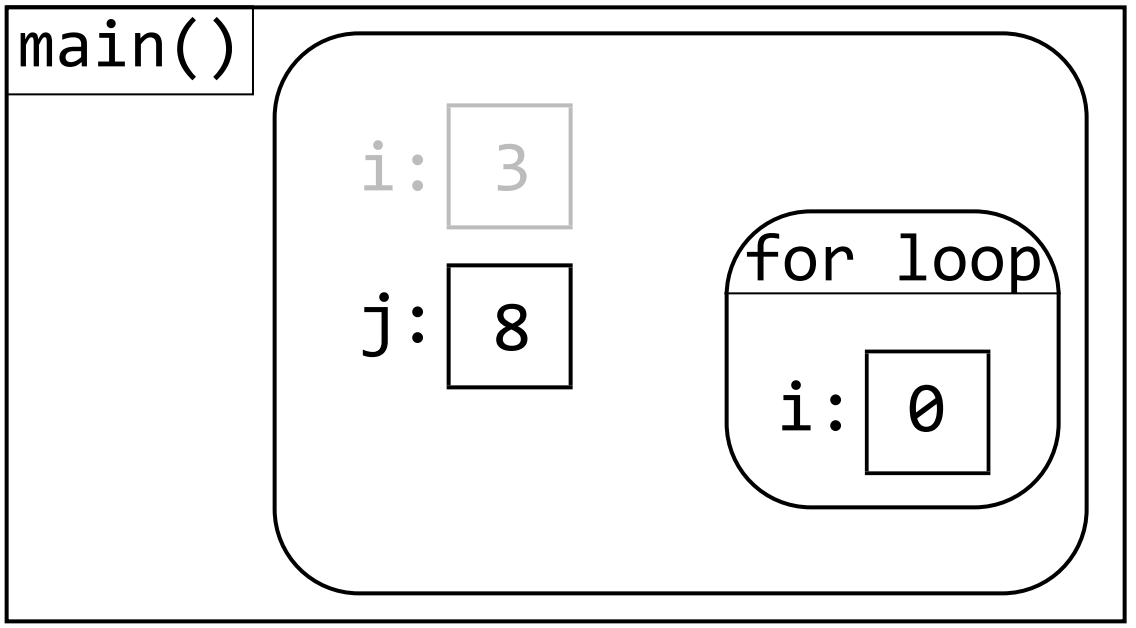

```
3
* 0 *
* 1 *
* 2 *
* 3 *
* 4 *
3
```

```c
#define N 5


int main(void)
{
    int i = 3;
    int j = 8;

    printf("%d\n", i);

    for (int i = 0; i < N; i++)
    {
        printf("* %d * \n", i);
    }


    printf("%d\n", i);

    return 0;
}
```



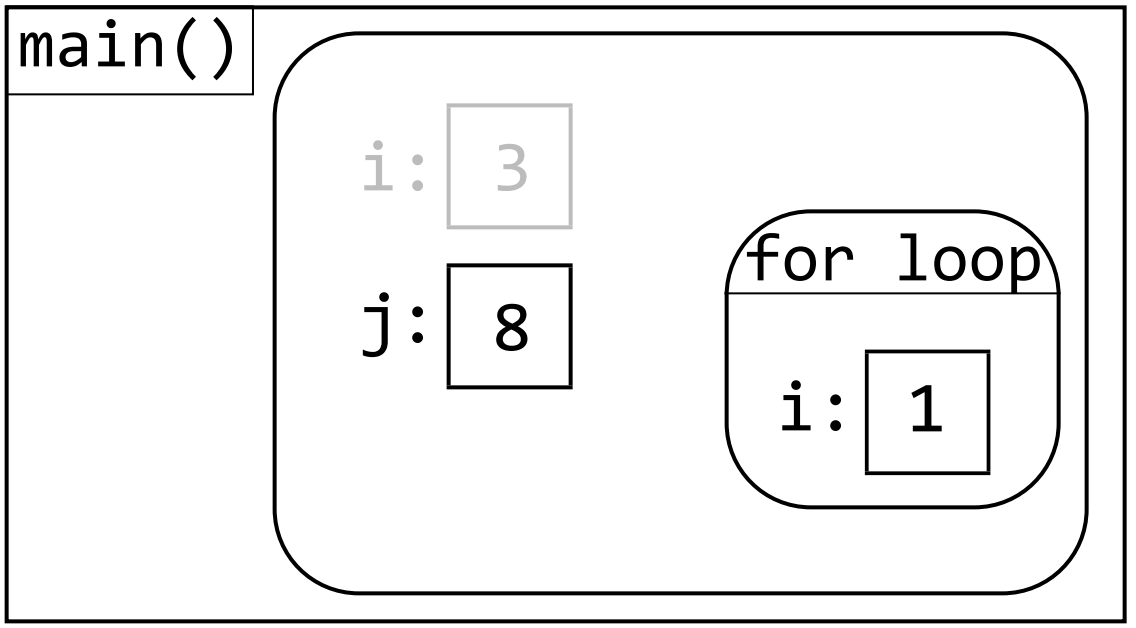

```
3
* 0 *
* 1 *
* 2 *
* 3 *
* 4 *
3
```

```c
#define N 5


int main(void)
{
    int i = 3;
    int j = 8;

    printf("%d\n", i);

    for (int i = 0; i < N; i++)
    {
        printf("* %d * \n", i);
    }


    printf("%d\n", i);

    return 0;
}
```



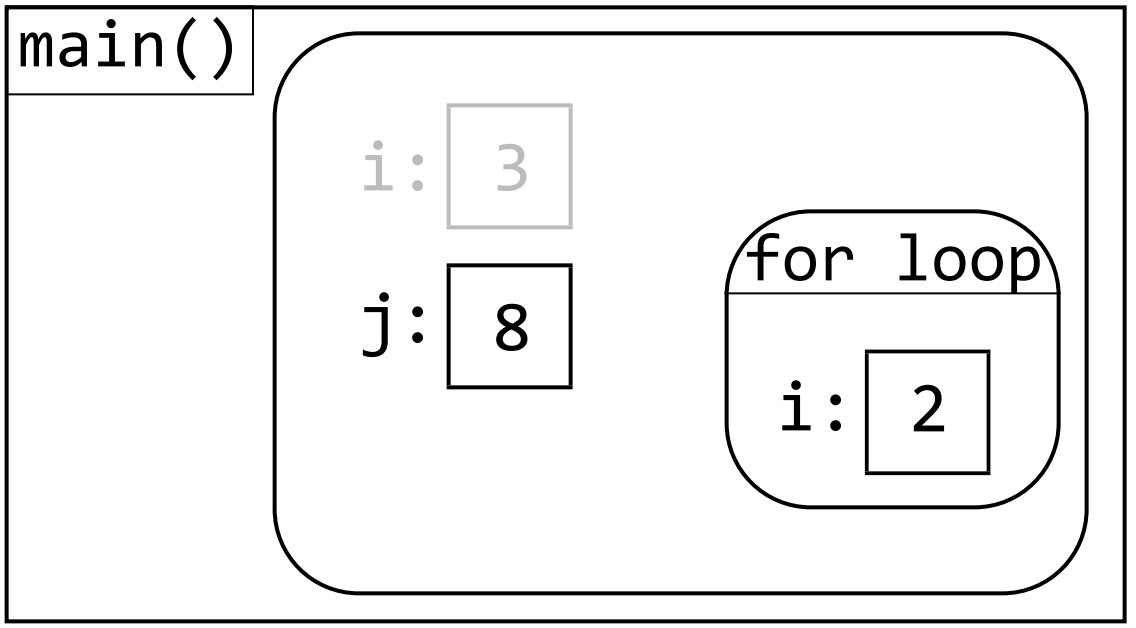

```
3
* 0 *
* 1 *
* 2 *
* 3 *
* 4 *
3
```

```c
#define N 5


int main(void)
{
    int i = 3;
    int j = 8;

    printf("%d\n", i);

    for (int i = 0; i < N; i++)
    {
        printf("* %d * \n", i);
    }


    printf("%d\n", i);

    return 0;
}
```



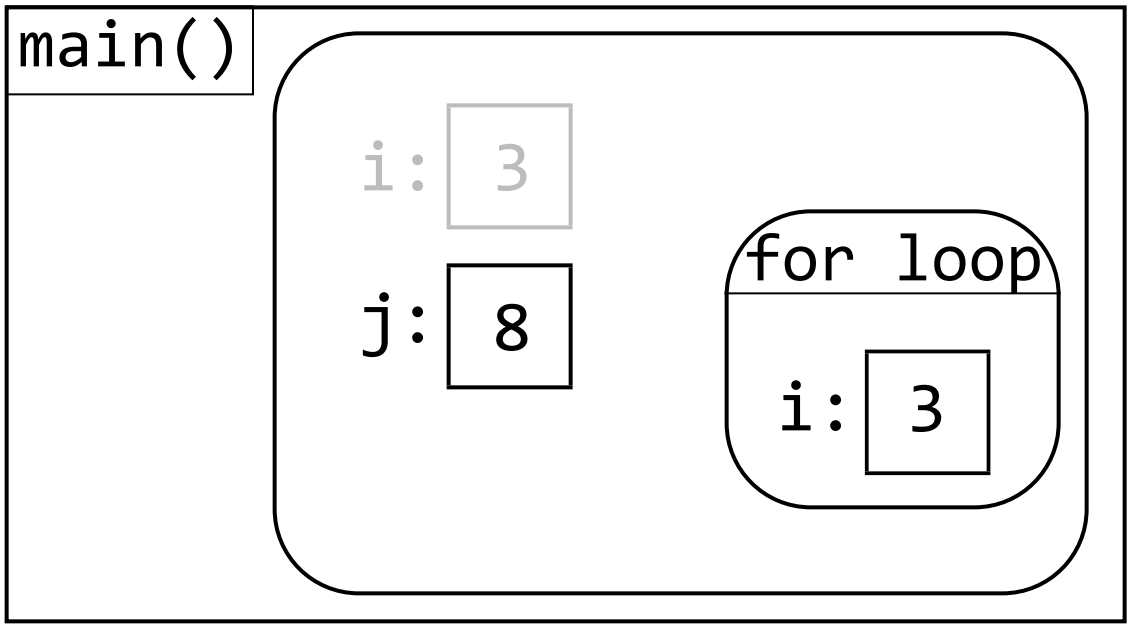

```
3
* 0 *
* 1 *
* 2 *
* 3 *
* 4 *
3
```

```c
#define N 5


int main(void)
{
    int i = 3;
    int j = 8;

    printf("%d\n", i);

    for (int i = 0; i < N; i++)
    {
        printf("* %d * \n", i);
    }


    printf("%d\n", i);

    return 0;
}
```



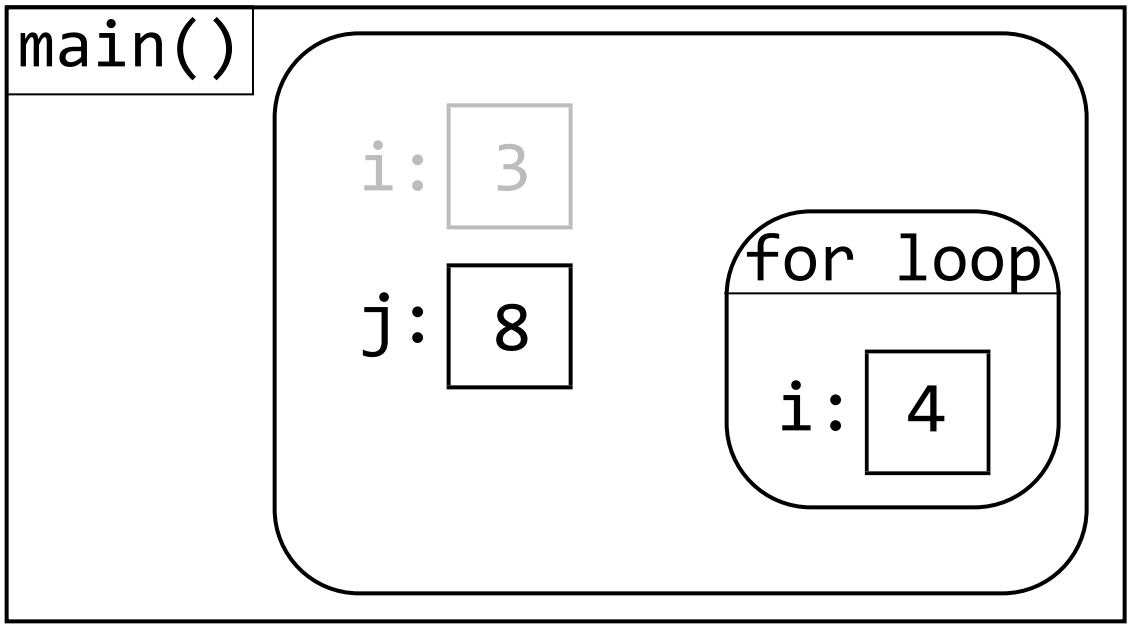

```
3
* 0 *
* 1 *
* 2 *
* 3 *
* 4 *
3
```

```c
#define N 5


int main(void)
{
    int i = 3;
    int j = 8;

    printf("%d\n", i);

    for (int i = 0; i < N; i++)
    {
        printf("* %d * \n", i);
    }


    printf("%d\n", i);

    return 0;
}
```



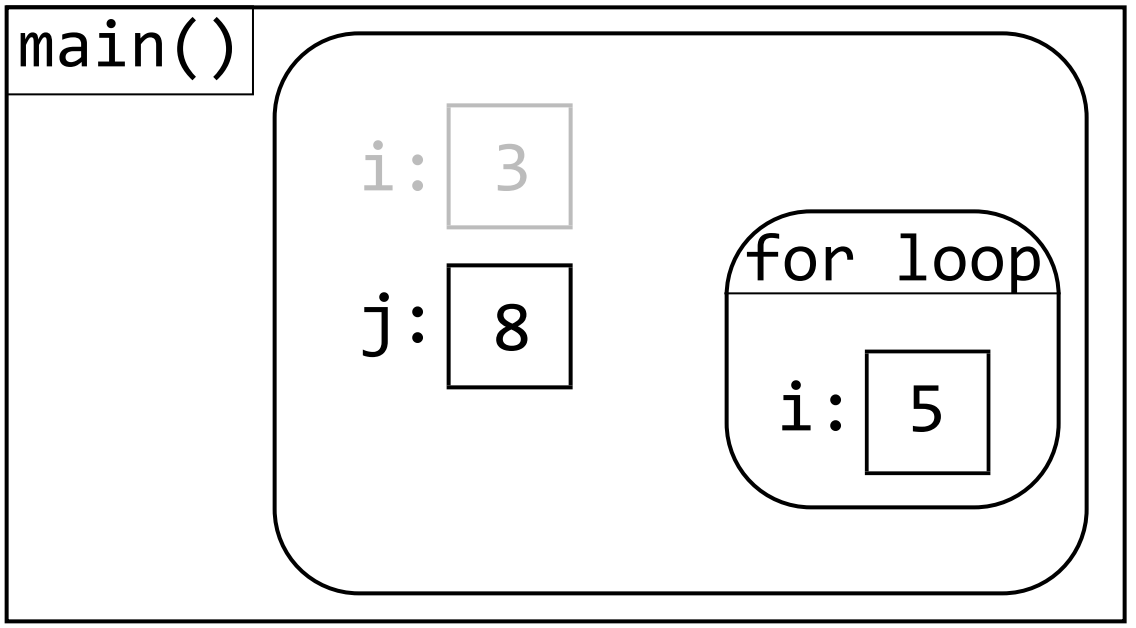

```
3
* 0 *
* 1 *
* 2 *
* 3 *
* 4 *
3
```

```c
#define N 5


int main(void)
{
    int i = 3;
    int j = 8;

    printf("%d\n", i);

    for (int i = 0; i < N; i++)
    {
        printf("* %d * \n", i);
    }


    printf("%d\n", i);

    return 0;
}
```



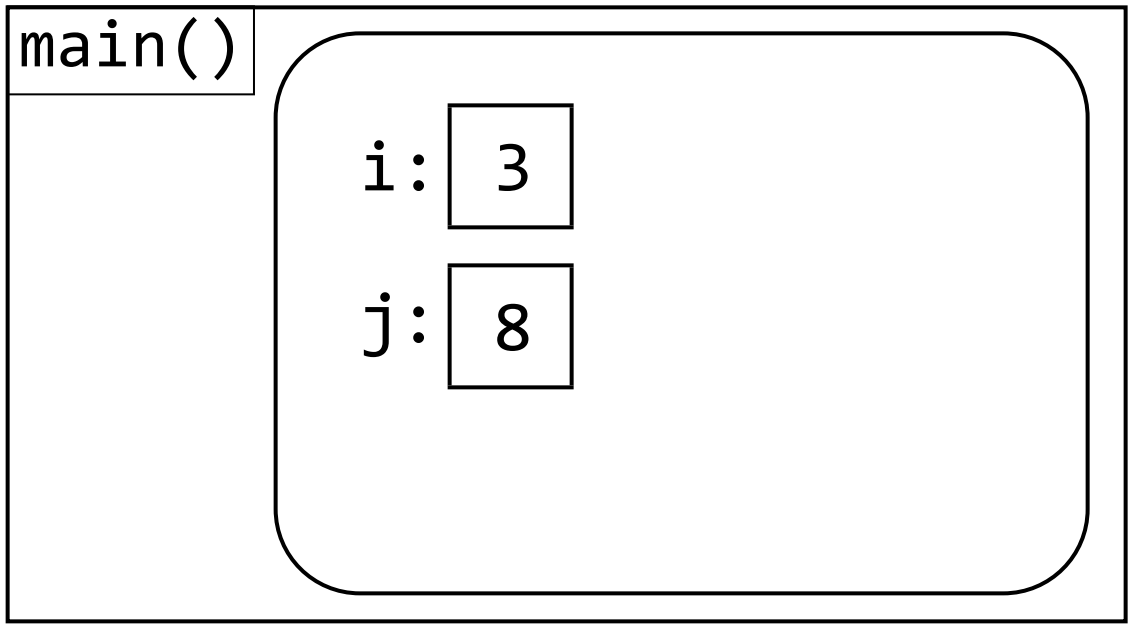

```
3
* 0 *
* 1 *
* 2 *
* 3 *
* 4 *
3
```

```c
#define N 5


int main(void)
{
    int i = 3;
    int j = 8;

    printf("%d\n", i);

    for (int i = 0; i < N; i++)
    {
        printf("* %d * \n", i);
    }


    printf("%d\n", i);

    return 0;
}
```




```
3
* 0 *
* 1 *
* 2 *
* 3 *
* 4 *
3
```

```c
#define N 5


int main(void)
{
    int i = 3;
    int j = 8;

    printf("%d\n", i);

    for (int i = 0; i < N; i++)
    {
        printf("* %d * \n", i);
    }


    printf("%d\n", i);

    return 0;
}
```




```
3
* 0 *
* 1 *
* 2 *
* 3 *
* 4 *
3
```

Shadowing

- A child block may redeclare a variable that was declared in its parent
- You lose access to the parent's version for the duration of the child block
- You rarely want to do this

From Problem To Algorithm

Instructions

Add the numbers 2 and 3.
Print out the result.

Program Design
(human)

I want a program that will show me what you get when you add 2 and 3.

Programming
(human)

```
int sum = 2 + 3;
printf("%d", sum);
```

Compiling
(computer)

Machine Code

```
11111110111011011111101011001111
00000001000000000000000000000111
10000000000000000000000000000011
00000000000000000000000000000010
00000000000000000000000000001011
00000000000000000000011000000000
00000000000000000000000010000101
00000000000000000000000000000000
00000000000000000000000000011001
00000000000000000000000001001000
```

Initialize the canvas
Initialize the pen

main `while` loop

Print out the canvas
Print out the bottom coords
Ask for a valid command
If it's 'Q', stop
If it's 'U', raise the pen
If it's 'D', lower the pen
If it's a direction command
  Ask for a valid distance
  Calculate the new x position
  Calculate the new y position
  If the pen is down, draw a line
  Update the pen's position
  Draw the pen

```
void clearCanvas(...);

void printCanvas(...);
void printBotCoordinates(void);
char getValidCommand(void);

bool isDirectionCommand(...);
int getValidDistance(void);
int calculateXPosition(...);
int calculateYPosition(...);
void drawLine(...);
```

Pseudocode

Initialize the canvas

```
for (each row)
{
    for (each column)
    {
        set that element to BLANK_SYMBOL
    }
}
```

Pseudocode

Calculate the new x position

```
if (we're heading East)
{
    add distance to our current position
    if (we're off the right edge of the canvas)
    {
        our new position is at the right edge
    }
}
else if (we're heading West)
{
    subtract distance from our current position
    if (we're off the left edge of the canvas)
    {
        our new position is at the left edge
    }
}
```

Let's Look at A2...

Built-in Functions

C Library

- Libraries are collections of functions and constants

  `stdio.h` `stdbool.h`

- Less work
- Fewer bugs
- Faster

`math.h`

- `math.h` is the math library
- `#include <math.h>`
- Provides some useful functions
- Might provide some constants

math.h constants

- M_PI is π
- M_E is e
- Not part of the C standard

```
#define M_PI 3.14159265358979323846264 /* pi */

#include <math.h>

...

double area = M_PI * r * r;
```

math.h functions

```
double cos(double x);
double sin(double x);
double tan(double x);
double exp(double x);
double log(double x);
double log10(double x);
double sqrt(double x);
double pow(double x, double y);
double ceil(double x);
double floor(double x);
double rint(double x);
double fabs(double x);
```

e^x

$\log_e(x)$ a.k.a $\ln(x)$

$\log_{10}(x)$

$\sqrt{x}$

x^y

$\lceil x \rceil$

$\lfloor x \rfloor$

C99

$|x|$

many more

`-lm` flag

- Need to use the `-lm` flag
  - `-l` is "link", `m` is "math"

```
$ gcc -lm math.c
$ gcc -Wall -std=c99 -lm -o math math.c
```

What's Wrong With `sin()`?

- We don't actually know what it does
- Before using a function, we need to read the documentation
- It gives us a contract
  - We give it some form of input
  - It does something and/or gives us back some form of output

Advanced search
Language tools

Google Search I'm Feeling Lucky

Advertising Programs Business Solutions About Google **Go to Google Canada**

man

- The UNIX manual system

First match: `$ man` *command*

C library only: `$ man 3` *command*

`math.h` errors

- Complicated, and we won't deal with it
- You can assume that all input to math functions will be valid

```
double d = sqrt(-16);
```