APS105
Winter 2012

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

Lecture 15
February 17, 2012

Today

- Even More Pointers
- Pointers and Arrays

```
if (size > 5)
{
   printf("*****\n");
   printf("The object is large!\n");
   printf("*****\n");
}
else
{
   printf("*****\n");
   printf("The object is small!\n");
   printf("*****\n");
}
```

```
printf("*****\n");
if (size > 5)
{
   printf("The object is large!\n");
}
else
{
   printf("The object is small!\n");
}
printf("*****\n");
```

```
char guess;
scanf("%c", &guess);
getchar();
...
scanf("%c", &guess);
```

Potentially Wrong

```
char guess;
scanf("%c", &guess);
char blank;
scanf("%c", &blank);
...
scanf("%c", &guess);
```

Bad Style

```
char guess;
scanf("%c", &guess);
...
scanf(" %c", &guess);
```

Midterm

- Friday March 2, in the Tutorial time slot
- Covers up to the end of pointers
- Mix of question types
- Focus on the ideas

- I'll do some review questions earlier in week
- TA will do some today

During Reading Week

- CodeLab 6 will be up
  - Due March 4
- A3 will be up
  - Due March 10

- Office hours?

The Most Common Mistake

- Using an uninitialized pointer

```
int *p;
int j = *p;
```

The Most Common Mistake

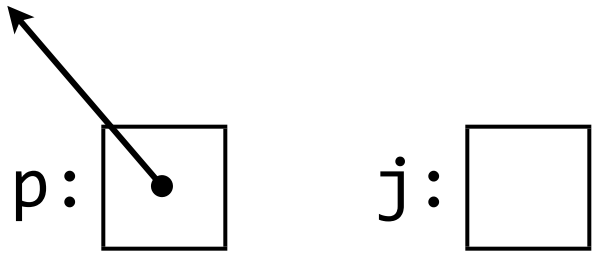
- Using an uninitialized pointer

```
int *p;
int j = *p;
```

The Most Common Mistake

- Using an uninitialized pointer

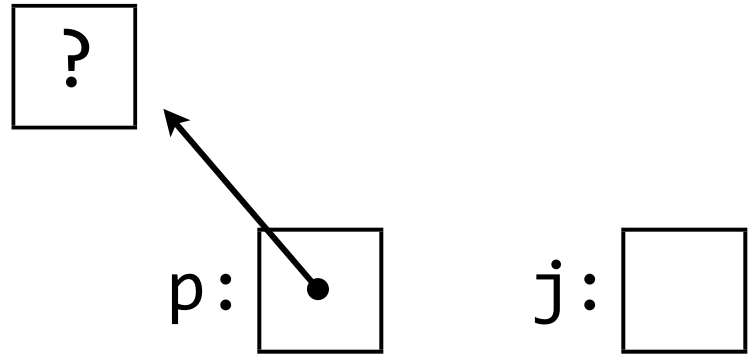
```
int *p;
int j = *p;
```



The Most Common Mistake

- Using an uninitialized pointer

```
int *p;
int j = *p;
```

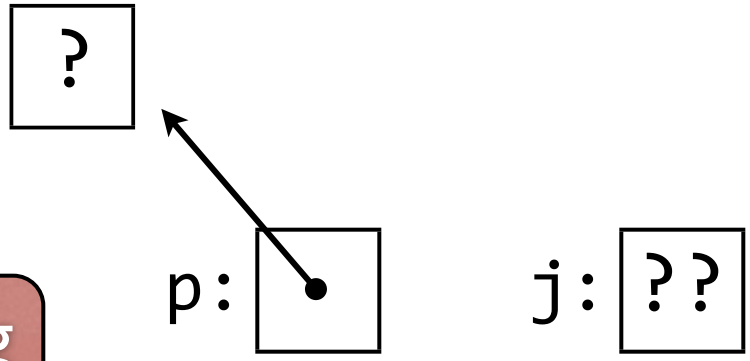


The Most Common Mistake

- Using an uninitialized pointer

```
int *p;
int j = *p;
```

Wrong

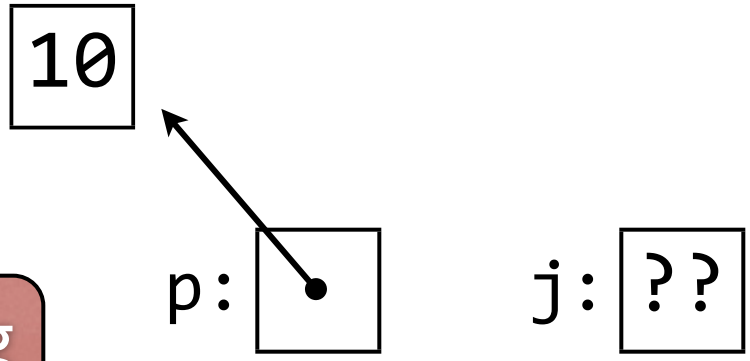


The Most Common Mistake

- Using an uninitialized pointer

```
int *p;
int j = *p;
```

Wrong




```
*p = 10;
```

Very Wrong

Pointer Fun with Binky

by Nick Parlante

This is document 104 in the Stanford CS Education Library — please see cslibrary.stanford.edu for this video, its associated documents, and other free educational materials.

Carpe Post Meridiem!

Pointers for Using Pointers

- Always remember:
  - *A pointer holds a memory address*

- Draw pictures

Compiler Warnings

```
int main(void)
{
    int i = 0;
    int *p;

    p = i;

    printf("Enter a number: ");
    scanf("%d", i);

    return 0;
}
```

Wrong

Wrong

Pointer Parameters

- Just like a normal parameter
- Make sure you call it with the right type

```
void swap(int *a, int *b);

int main(void)
{
    int i = 4;
    int j = 38;
    swap(i, j);
    swap(&i, &j);
    ...
```

Wrong

Pointer Parameters

- We can read *and write* to the original variable
- Sometimes we don't want to do that

```
void printHalf(double *p);

int main(void)
{
    double d = 4.0;

    printf("d is %g\n", d);
    printHalf(&d);
    printf("d is %g\n", d);

    return 0;
}

void printHalf(double *p)
{
    printf("****\n");
    printf("Half of d is %g \n", *p / 2.0);
    printf("****\n");
}
```

const

- Modifier for a variable that makes it read-only
- You can use it anywhere you define a variable

```
const int size = 10;
size = 12;
```

Error

```
const double price;
```

Legal, but useless

```
int position = 48;
const position;
```

Error

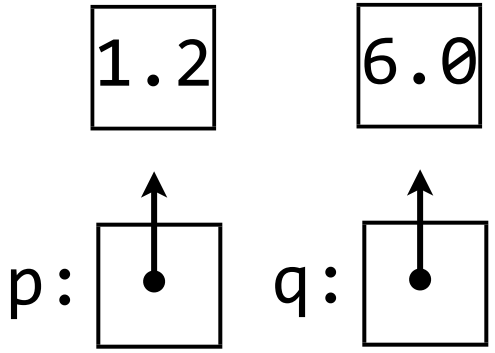
const parameters

- Most common use is parameters

p points to a const double

```
printProduct(const double *p, double *q)
{
    *p = 5.0;
}
```

Error



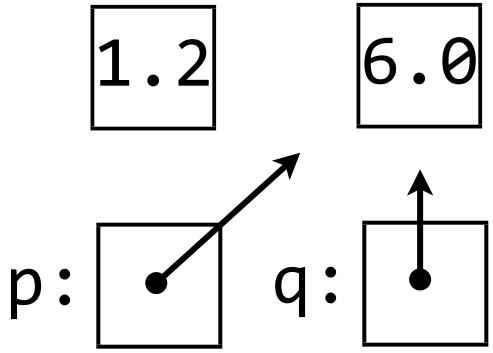
const parameters

- Most common use is parameters

p points to a const double

```
printProduct(const double *p, double *q)
{
    *p = 5.0;

     p = q;
}
```

Error



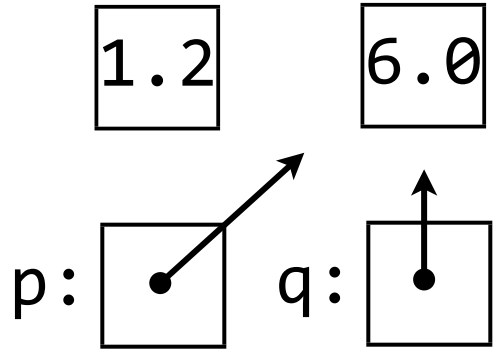
const parameters

- Most common use is parameters

p points to a const double

```
printProduct(const double *p, double *q)
{
    *p = 5.0;
     p = q;
}
```

Error



Wrong

```
printProduct(double * const p, double *q)
```

p is const and points to a double

`const` constants

- You can also use `const` for constants

```
#define MAX_VALUE 4
```

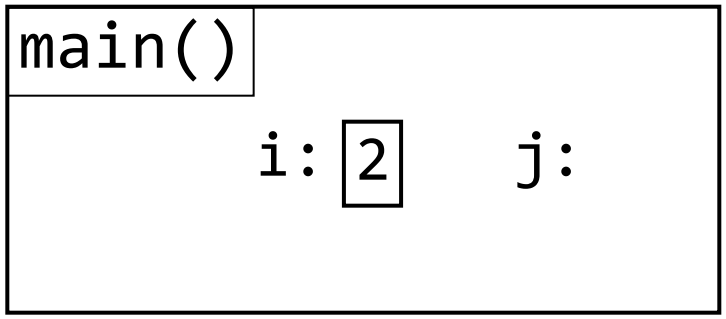
```
const int MAX_VALUE = 4;
```

Returning Pointers

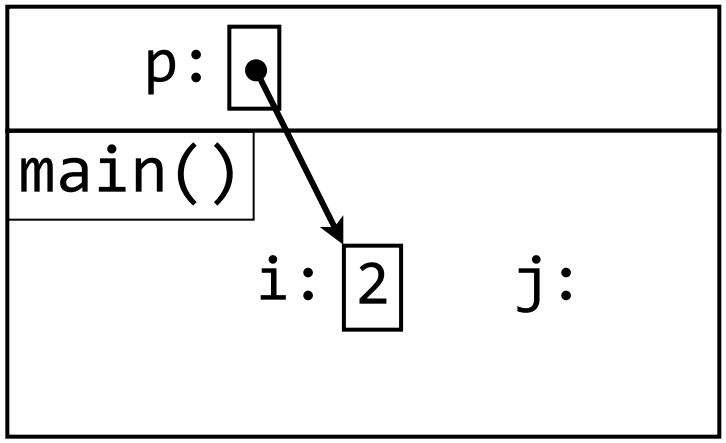
```
int *max(int *p, int *q)
{
    if (*p > *q)
    {
        return p;
    }
    else
    {
        return q;
    }
}
```

```
int main(void)
{

    int i = 8;
    int j = 9;

    int *r;

    r = max(&i, &j);

    printf("%d\n", *r);

    return 0;
}
```

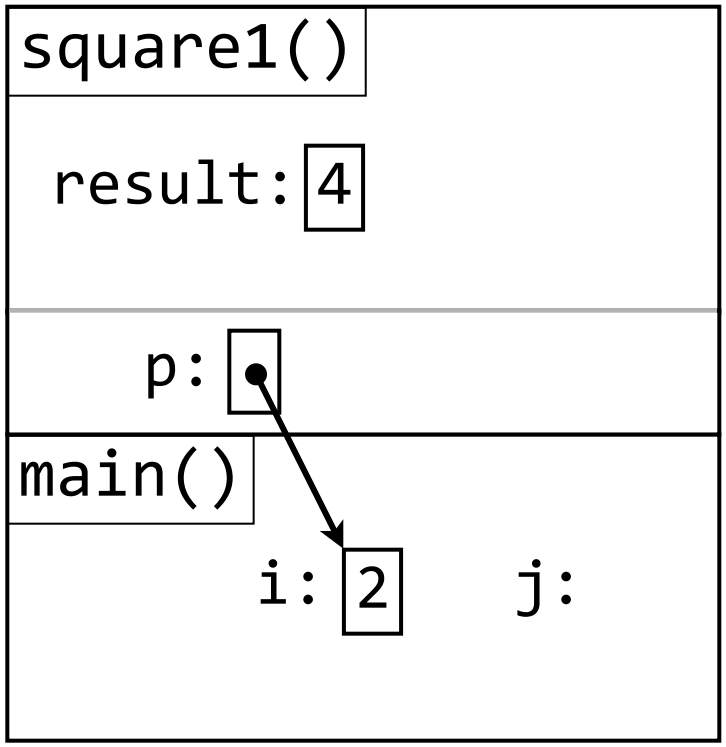
```
int square1(int *p)
{
  int result = *p * *p;
  return result;
}

int main(void)
{
  int i = 2;
  int j = square1(&i);
  printf("%d", j);

  return 0;
}
```



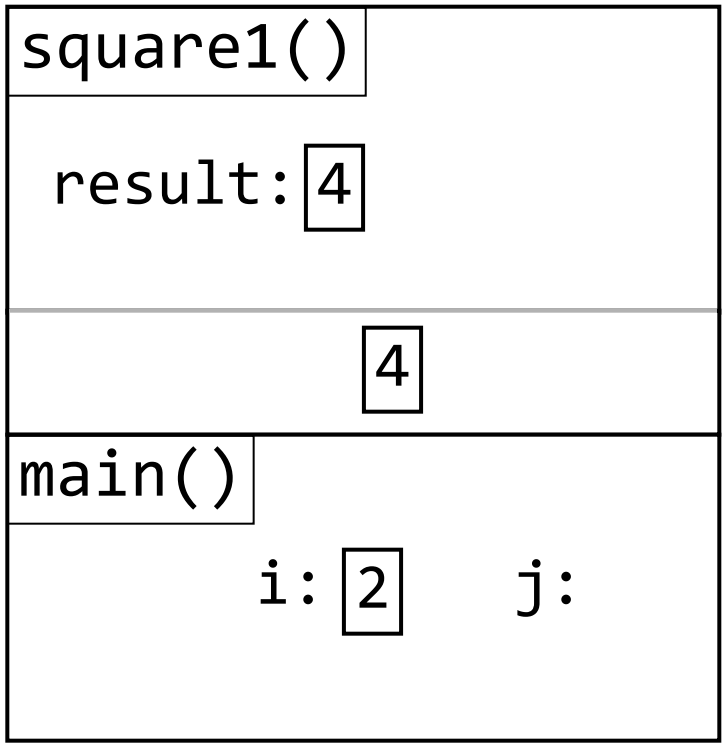
```
int square1(int *p)
{
  int result = *p * *p;
  return result;
}

int main(void)
{
  int i = 2;
  int j = square1(&i);
  printf("%d", j);

  return 0;
}
```



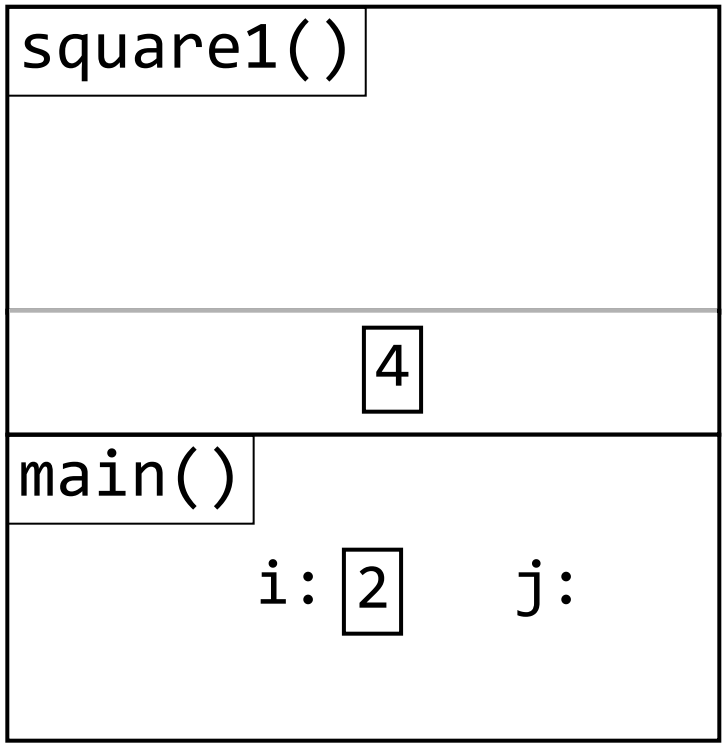
```
int square1(int *p)
{
  int result = *p * *p;
  return result;
}

int main(void)
{
  int i = 2;
  int j = square1(&i);
  printf("%d", j);

  return 0;
}
```



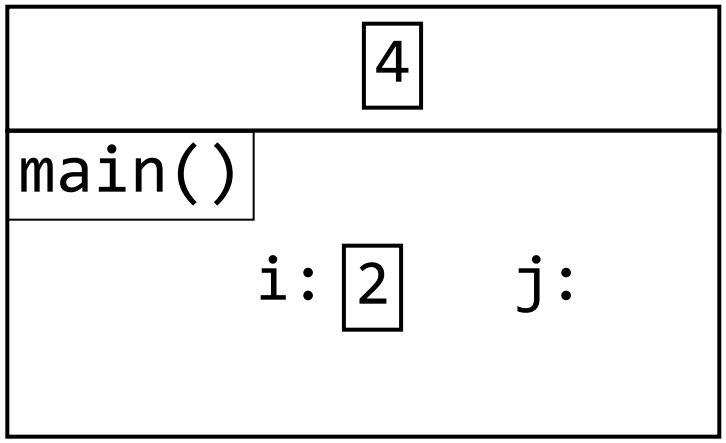
```
int square1(int *p)
{
  int result = *p * *p;
  return result;
}

int main(void)
{
  int i = 2;
  int j = square1(&i);
  printf("%d", j);

  return 0;
}
```



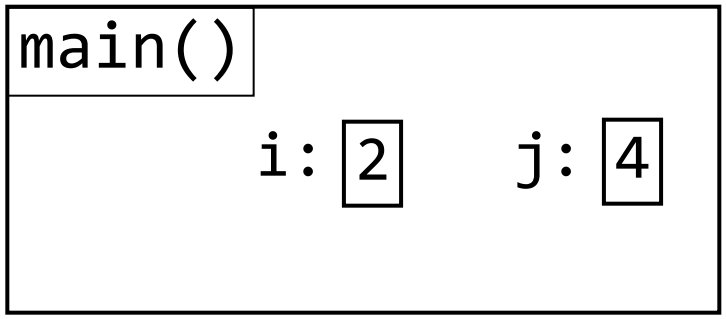
```
int square1(int *p)
{
  int result = *p * *p;
  return result;
}

int main(void)
{
  int i = 2;
  int j = square1(&i);
  printf("%d", j);

  return 0;
}
```



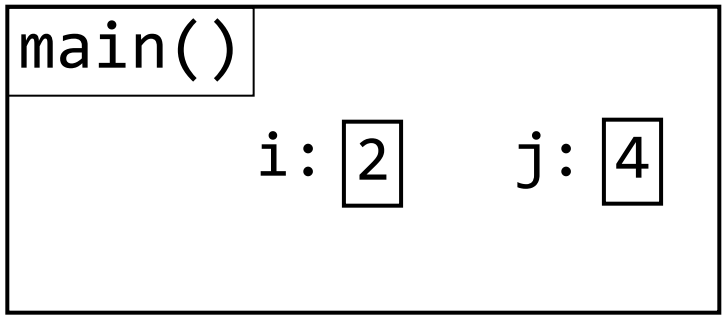
```
int square1(int *p)
{
  int result = *p * *p;
  return result;
}

int main(void)
{
  int i = 2;
  int j = square1(&i);
  printf("%d", j);

  return 0;
}
```



```
int square1(int *p)
{
  int result = *p * *p;
  return result;
}

int main(void)
{
  int i = 2;
  int j = square1(&i);
  printf("%d", j);

  return 0;
}
```

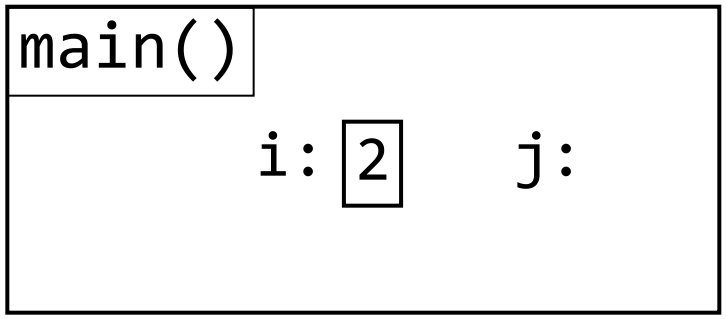


```
int square1(int *p)
{
  int result = *p * *p;
  return result;
}

int main(void)
{
  int i = 2;
  int j = square1(&i);
  printf("%d", j);
                    4
  return 0;
}
```

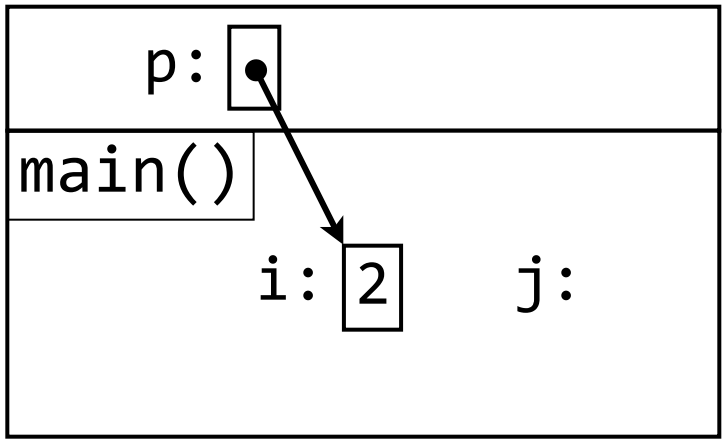


```
int *square2(int *p)
{
  int result = *p * *p;
  return &result;
}

int main(void)
{
  int i = 2;
  int *j = square2(&i);
  printf("%d", *j);

  return 0;
}
```

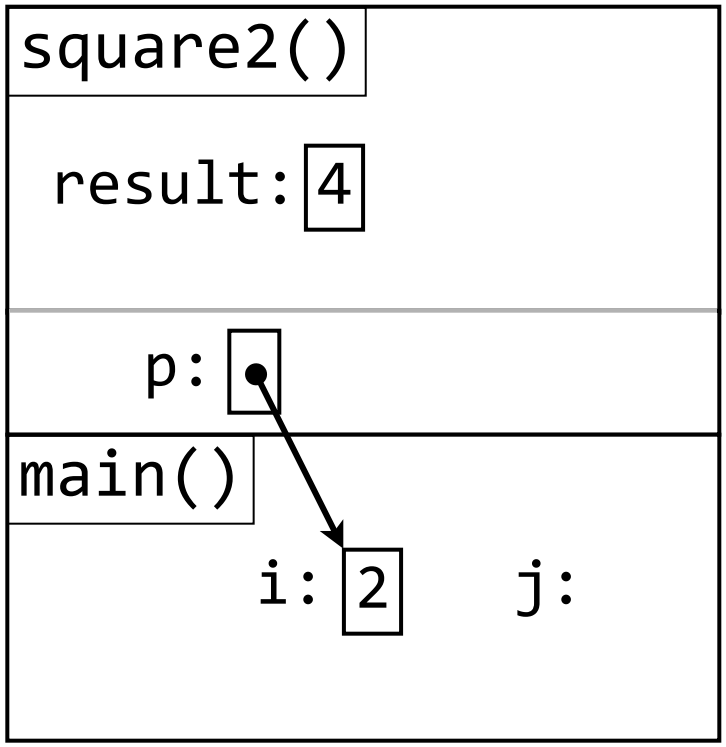
```
int *square2(int *p)
{
  int result = *p * *p;
  return &result;
}

int main(void)
{
  int i = 2;
  int *j = square2(&i);
  printf("%d", *j);

  return 0;
}
```



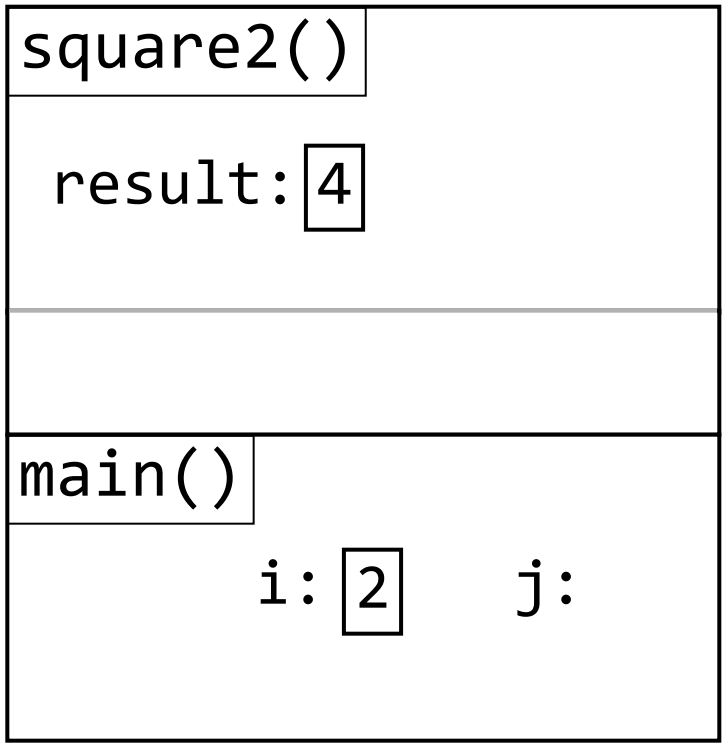
```
int *square2(int *p)
{
  int result = *p * *p;
  return &result;
}

int main(void)
{
  int i = 2;
  int *j = square2(&i);
  printf("%d", *j);

  return 0;
}
```



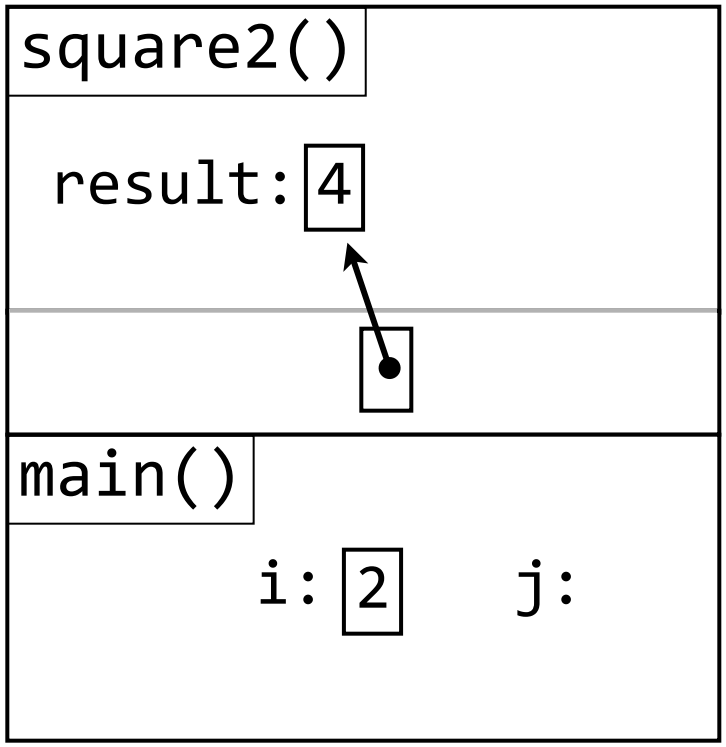
```
int *square2(int *p)
{
  int result = *p * *p;
  return &result;
}

int main(void)
{
  int i = 2;
  int *j = square2(&i);
  printf("%d", *j);

  return 0;
}
```



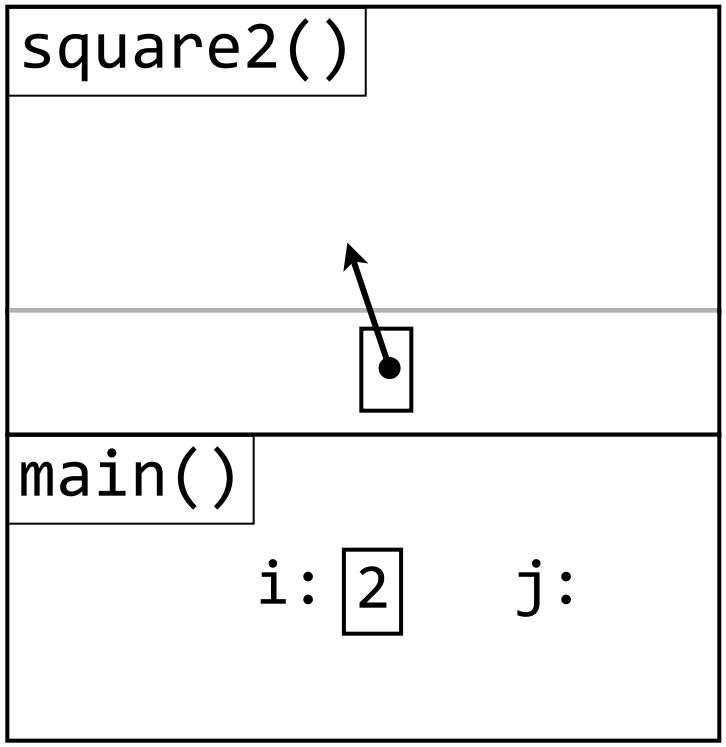
```
int *square2(int *p)
{
  int result = *p * *p;
  return &result;
}

int main(void)
{
  int i = 2;
  int *j = square2(&i);
  printf("%d", *j);

  return 0;
}
```



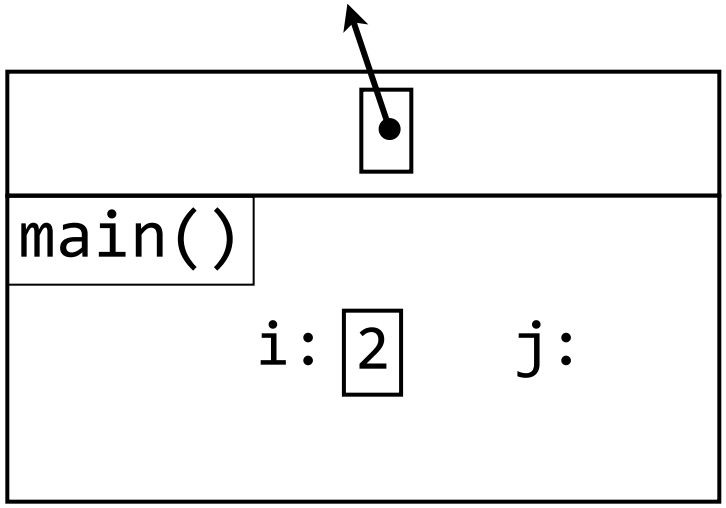
```
int *square2(int *p)
{
  int result = *p * *p;
  return &result;
}

int main(void)
{
  int i = 2;
  int *j = square2(&i);
  printf("%d", *j);

  return 0;
}
```



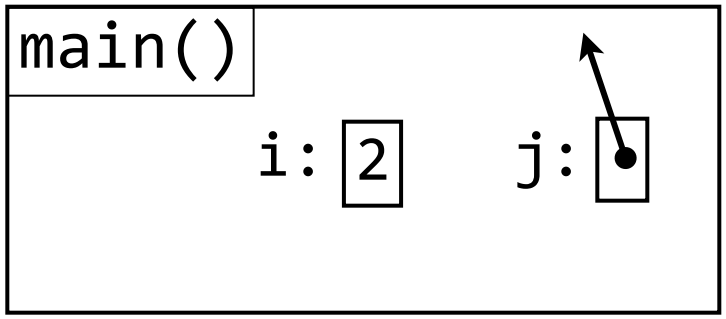
```
int *square2(int *p)
{
  int result = *p * *p;
  return &result;
}

int main(void)
{
  int i = 2;
  int *j = square2(&i);
  printf("%d", *j);

  return 0;
}
```



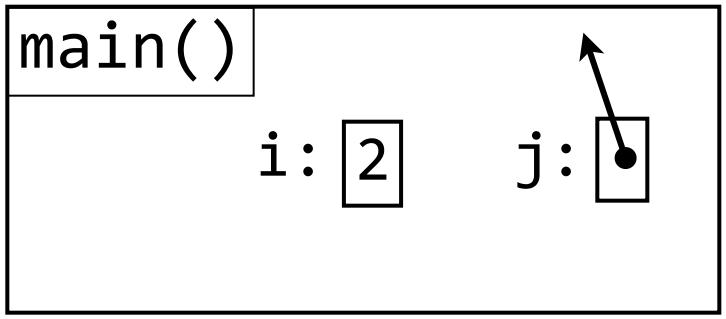
```
int *square2(int *p)
{
  int result = *p * *p;
  return &result;
}

int main(void)
{
  int i = 2;
  int *j = square2(&i);
  printf("%d", *j);

  return 0;
}
```



```
int *square2(int *p)
{
  int result = *p * *p;
  return &result;
}

int main(void)
{
  int i = 2;
  int *j = square2(&i);
  printf("%d", *j);

  return 0;
}
```



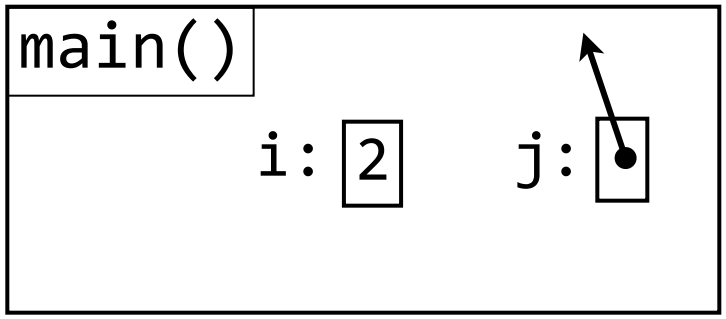
```
int *square2(int *p)
{
  int result = *p * *p;
  return &result;
}

int main(void)
{
  int i = 2;
  int *j = square2(&i);
  printf("%d", *j);
                  ????
  return 0;
}
```



```
int *square2(int *p)
{
  int result = *p * *p;
  return &result;
}

int main(void)
{
  int i = 2;
  int *j = square2(&i);
  printf("%d", *j);

  return 0;
}
```

Wrong

????



Pointing to Nothing

- You can't follow a pointer that points at something that doesn't exist anymore!
  - (Well, you can, but it's undefined)
  - Same as using an uninitialized pointer

```
int *p;
*p = 42;
```

- Common source of this bug is via `return` from a function call