Pointers

CS 2022: Introduction to C

Instructor: Hussam Abu-Libdeh

(based on slides by Saikat Guha)

Fall 2011, Lecture 3

Administrivia

- Office Hours:
  Wednesdays 11:10am – 12:10pm
  4139 Upson Hall
- Announcements @
  `http://twitter.com/cs2022`

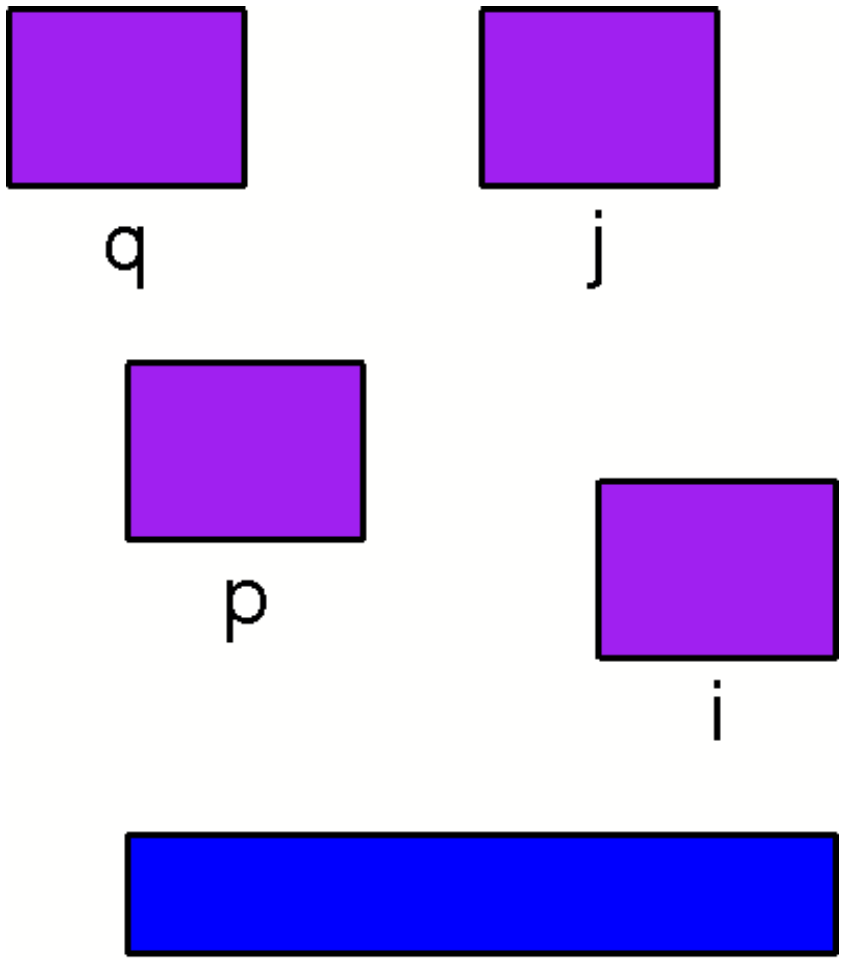
Pointer

Pointer

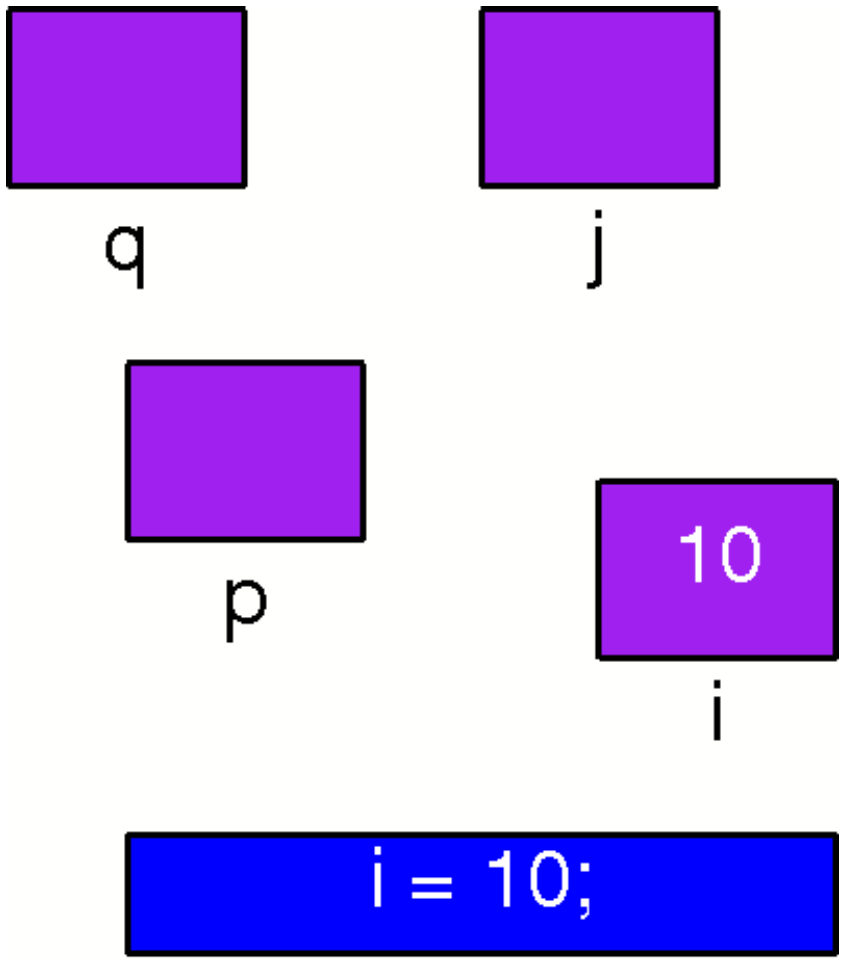
A pointer is just another variable that points to another variable. A pointer contains the memory address of the variable it points to.

```
int i;                  // Integer
int *p;                 // Pointer to integer
int **m;                // Pointer to int pointer

p = &i;                 // p now points to i
printf("%p", p);        // address of i (in p)

m = &p;                 // m now points to p
printf("%p", m);        // address of p (in m)
```

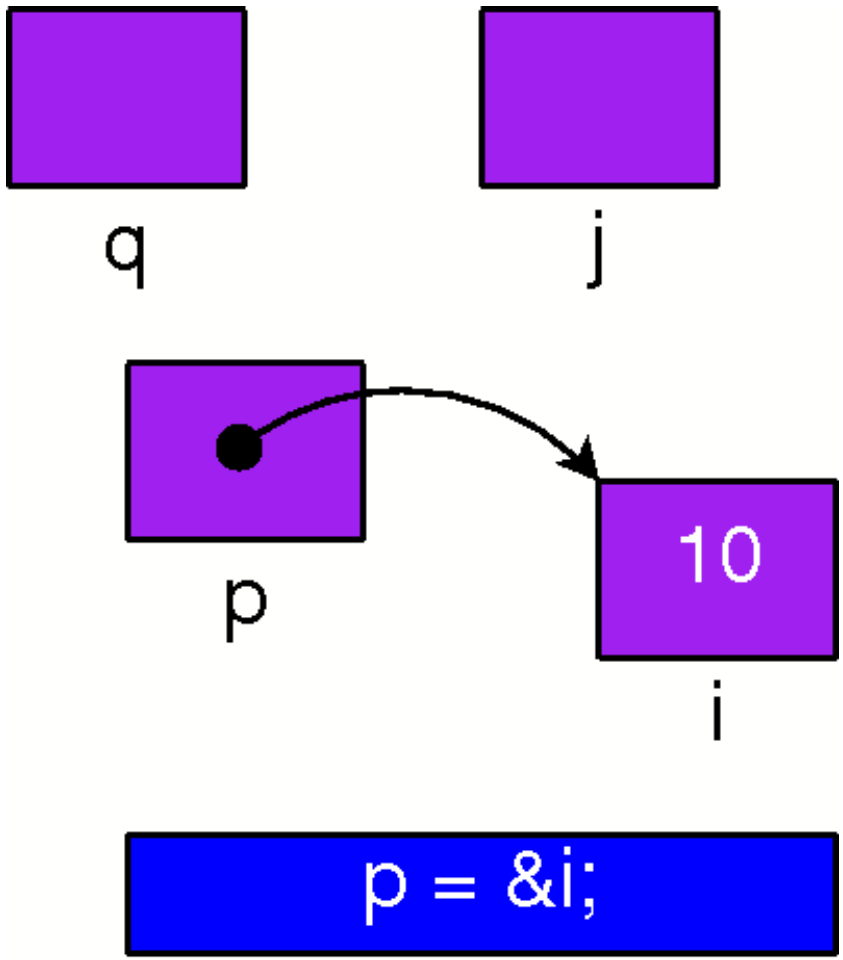
Pointers



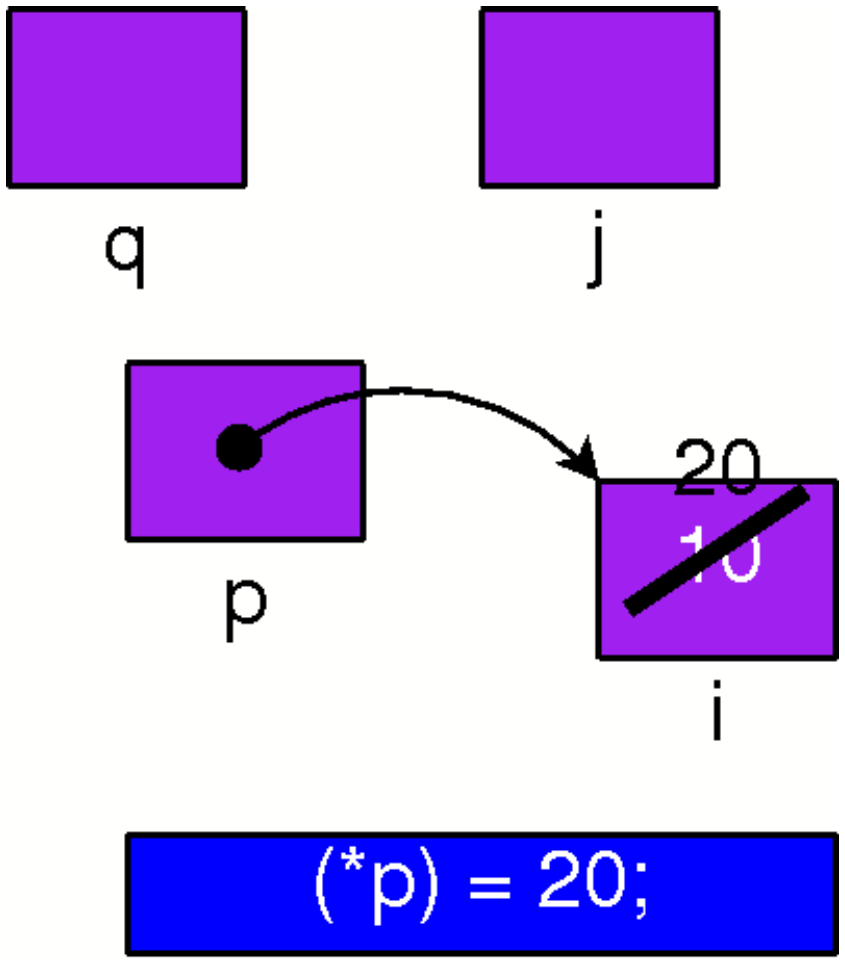
Pointers



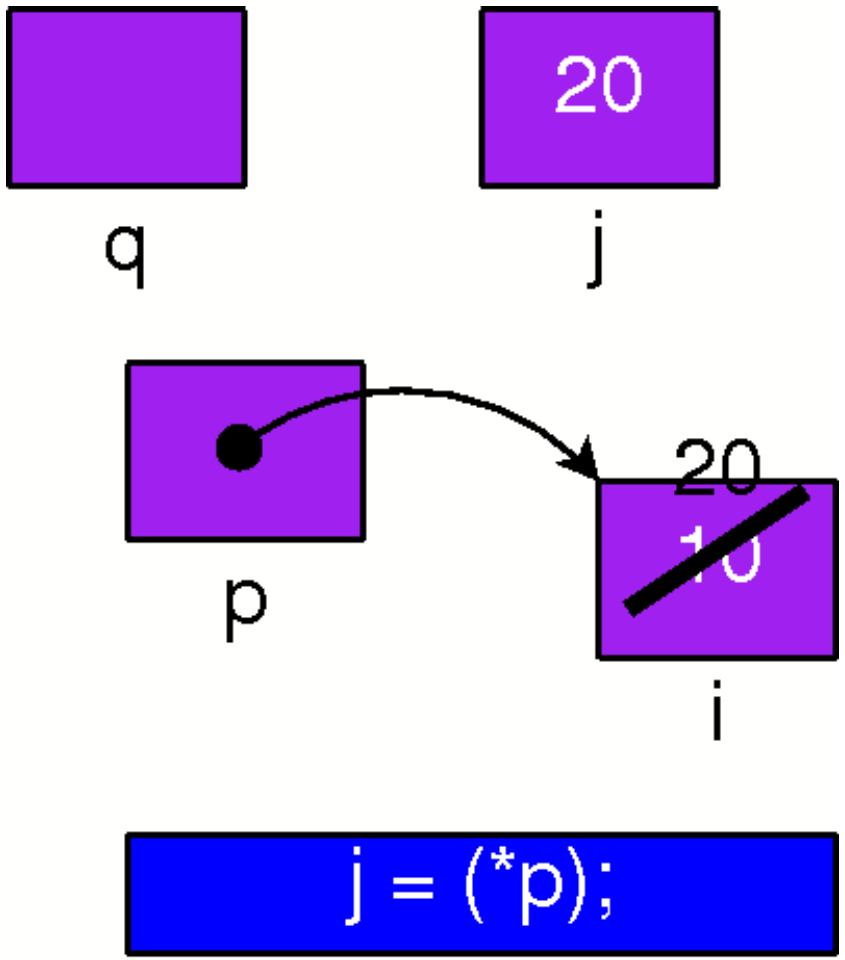
Pointers



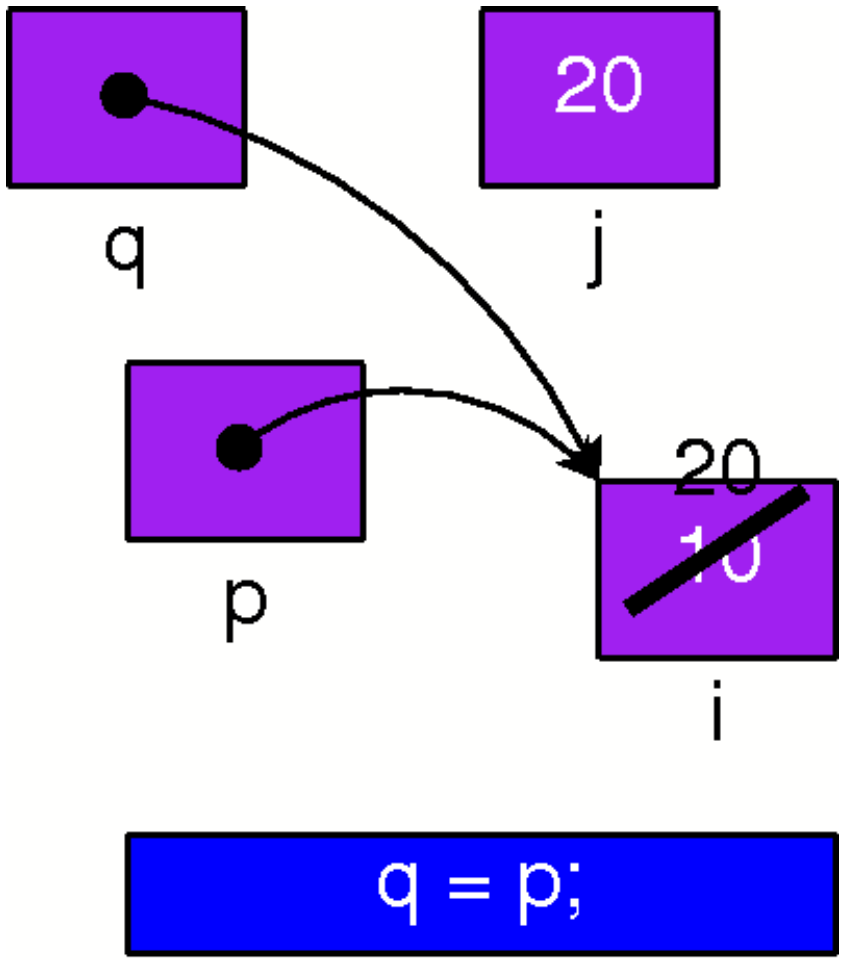
Pointers



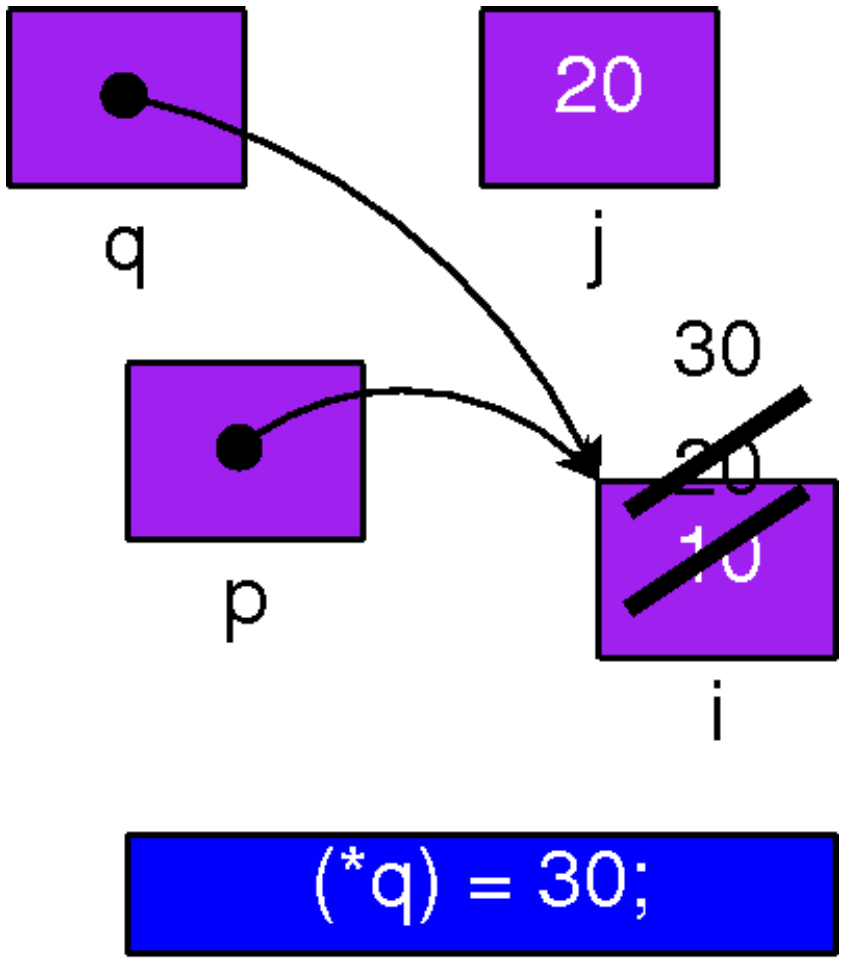
Pointers



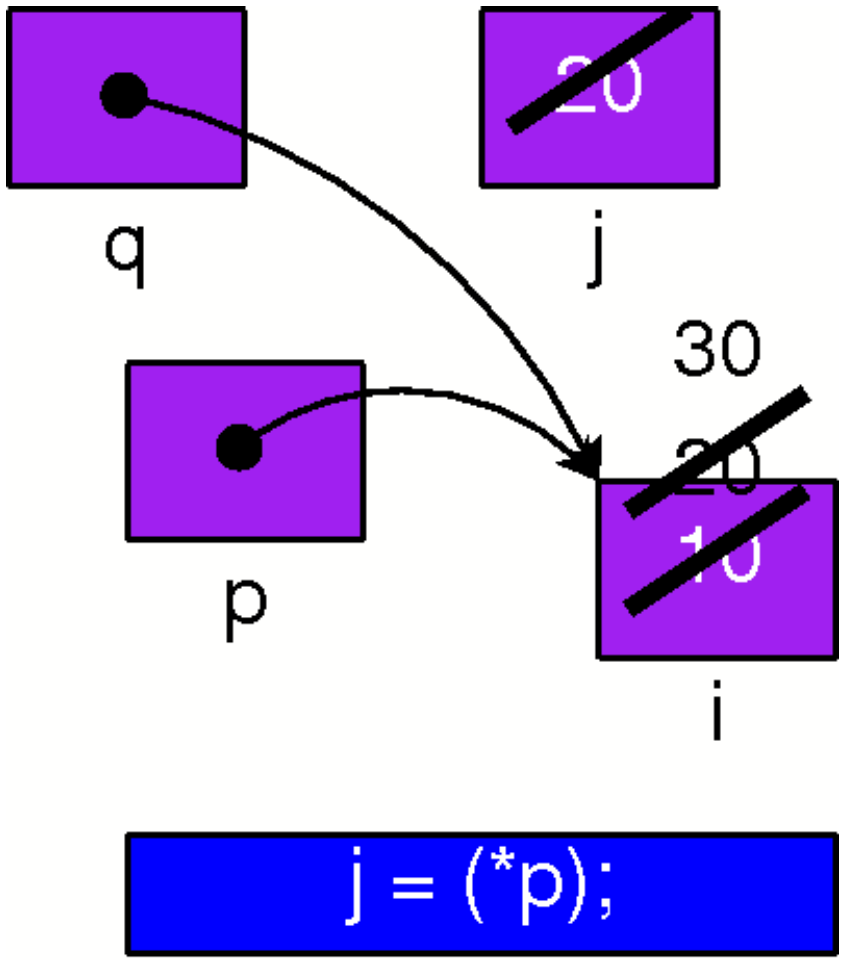
Pointers



Pointers



Pointers



swap1.c: Swap

```
#include <stdio.h>
int main() {
    int a, b;
    int *p, *q;

    a = 10; b = 20;
    p = &a; q = &b;
    printf("Before: %d, %d, %d, %d",
        a, b, *p, *q);

    __ = __;
    __ = __;

    printf("After: %d, %d, %d, %d",
        a, b, *p, *q);
    return 0;
}
```

```
Before:  10, 20, 10, 20
After:  10, 20, 20, 10
```

swap1.c: Swap

```
#include <stdio.h>
int main() {
    int a, b;
    int *p, *q;

    a = 10; b = 20;
    p = &a; q = &b;
    printf("Before: %d, %d, %d, %d",
        a, b, *p, *q);

    p = &b;
    q = &a;

    printf("After: %d, %d, %d, %d",
        a, b, *p, *q);
    return 0;
}
```

```
Before:  10, 20, 10, 20
After:  10, 20, 20, 10
```

swap2.c: Swap

```
#include <stdio.h>
int main() {
    int a, b;
    int *p, *q;

    a = 10; b = 20;
    p = &a; q = &b;
    printf("Before: %d, %d, %d, %d",
        a, b, *p, *q);

    __ = __;
    __ = __;

    printf("After: %d, %d, %d, %d",
        a, b, *p, *q);
    return 0;
}
```

```
Before:  10, 20, 10, 20
After:  20, 10, 20, 10
```

swap2.c: Swap

```
#include <stdio.h>
int main() {
    int a, b;
    int *p, *q;

    a = 10; b = 20;
    p = &a; q = &b;
    printf("Before: %d, %d, %d, %d",
        a, b, *p, *q);

    a  = 20;
    b  = 10;

    printf("After: %d, %d, %d, %d",
        a, b, *p, *q);
    return 0;
}
```

```
Before:  10, 20, 10, 20
After:  20, 10, 20, 10
```

swap3.c: Swap

```
#include <stdio.h>
int main() {
    int a, b;
    int *p, *q;

    a = 10; b = 20;
    p = &a; q = &b;
    printf("Before: %d, %d, %d, %d",
        a, b, *p, *q);

    __ = __; __ = __;
    __ = __; __ = __;

    printf("After: %d, %d, %d, %d",
        a, b, *p, *q);
    return 0;
}
```

```
Before:  10, 20, 10, 20
After:  20, 10, 10, 20
```

swap3.c: Swap

```
#include <stdio.h>
int main() {
    int a, b;
    int *p, *q;

    a = 10; b = 20;
    p = &a; q = &b;
    printf("Before: %d, %d, %d, %d",
        a, b, *p, *q);

    a  = 20; b  = 10;
    p  = &b; q  = &a;

    printf("After: %d, %d, %d, %d",
        a, b, *p, *q);
    return 0;
}
```

```
Before:  10, 20, 10, 20
After:  20, 10, 10, 20
```

Pointers to Pointers!

```
#include <stdio.h>
int main() {
    int a = 10, b = 20;
    int *p = &a, *q = &b;
    int **m = &p, **n = &q;

    printf("X: %d %d %d %d %d %d\n",
        **m, **n, *p, *q, a, b);

    *m = *n; m = n;
    *m = &a; n = &p;
    **n = 30;

    printf("Y: %d %d %d %d %d %d\n",
        **m, **n, *p, *q, a, b);
    return 0;
}
```

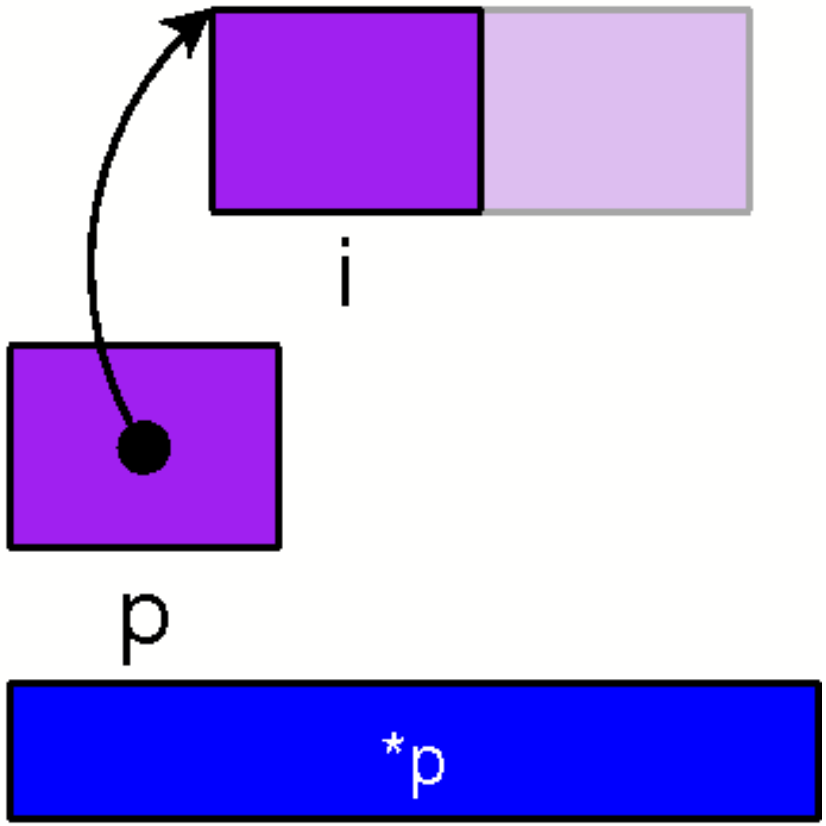
```
X: -- -- -- -- -- --
Y: -- -- -- -- -- --
```

Pointers to Pointers!

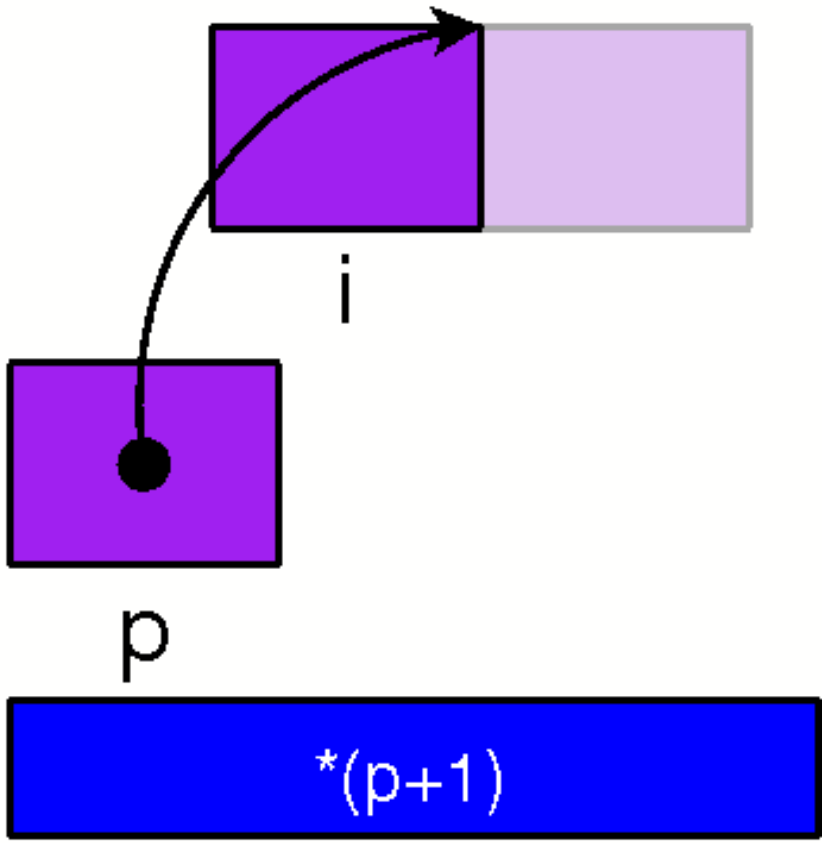
```
#include <stdio.h>
int main() {
    int a = 10, b = 20;
    int *p = &a, *q = &b;
    int **m = &p, **n = &q;

    printf("X: %d %d %d %d %d %d\n",
        **m, **n, *p, *q, a, b);

    *m = *n; m = n;
    *m = &a; n = &p;
    **n = 30;

    printf("Y: %d %d %d %d %d %d\n",
        **m, **n, *p, *q, a, b);
    return 0;
}
```

```
X: 10 20 10 20 10 20
Y: 10 30 30 10 10 30
```

Pointer Arithmetic



Pointer Arithmetic



Memory in C

Variables

- Independent variables are a figment of your imagination.
- When in C, think of memory cells. Each memory cell has an integer address.
- You can access any memory cell at any time from any function.
- Variable names are simply shortcuts for your convenience.

Nameless Variables

```
#include <stdlib.h>

int main() {
    int *p = (int *)malloc(sizeof(int));

    *p = 42;
    return 0;
}
```

A poor man's array

```
int * newarray(int siz) {
    return (int *)malloc(siz * sizeof(int));
}

void set(int *arr, int idx, int val) {
   *(arr+idx)  = val;
}

int get(int *arr, int idx) {
    return *(arr + idx);
}
```

Multiple Return Values

```
void getab(int *a, int *b) {
    *a = 10;
    *b = 20;
}

int main() {
    int a, b;

    getab(&a, &b);
}
```

Pointers Recap

- `int *ptr;`
- Pointers are variables that store memory addresses of other variables
- Type of variable pointed to depends on type of pointer:
  - `int *ptr` points to an integer variable
  - `char *ptr` points to a character variable
  - Can cast between pointer types:
    `my_int_ptr = (int *) my_other_ptr;`
  - `void *ptr` has an unspecified type; must be cast to a type before used

Pointers Recap

- Two main operations:
  - `*` *dereference*: gets the value at the memory location stored in a pointer
  - `&` *address of*: gets the address of a variable
  - `int *my_ptr = &my_var;`
- Pointer arithmetic: directly manipulate a pointer's content to access other memory locations
  - **Use with caution!**: can crash your program due to bad memory accesses
  - However, it is useful in accessing and manipulating data structures
- Pointers to pointers
  - `int **my_2d_array;`