

Pointers

CS 113: Introduction to C

Instructor: Saikat Guha

Cornell University

Spring 2007, Lecture 3

Pointer

Pointer

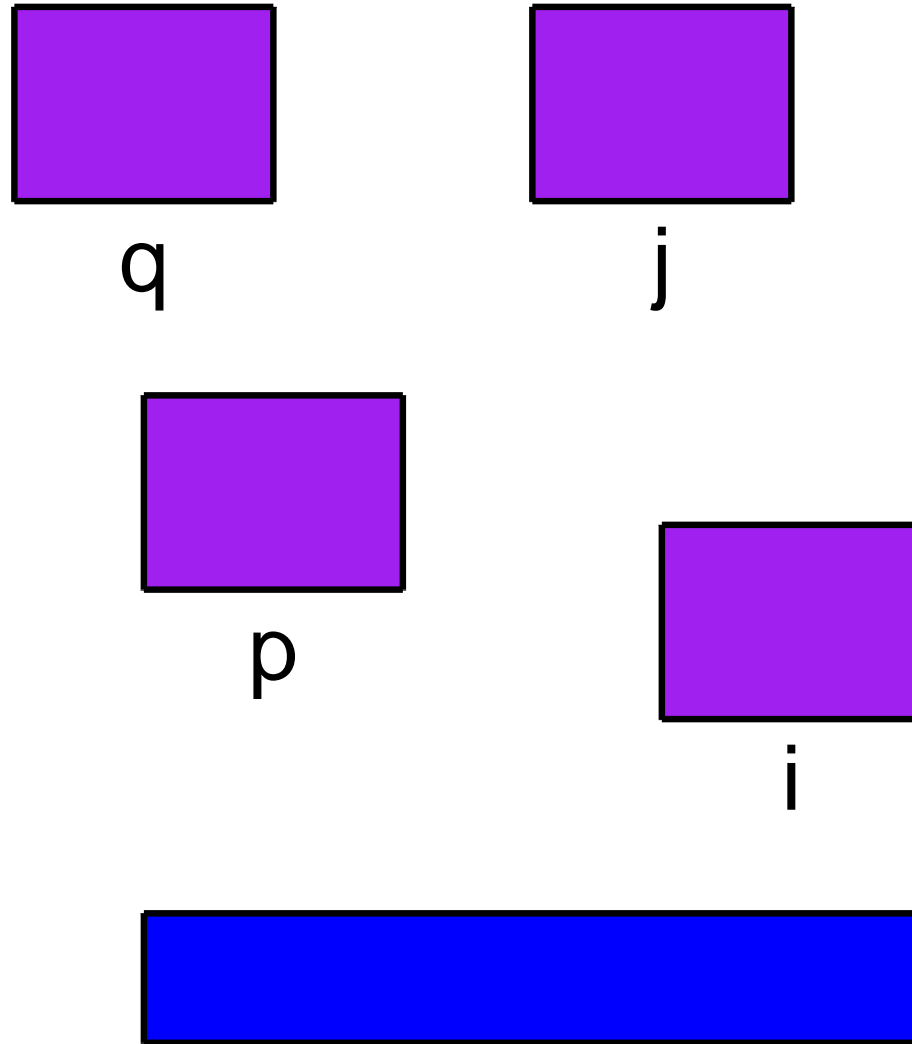
A pointer is just another variable that points to another variable. A pointer contains the memory address of the variable it points to.

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

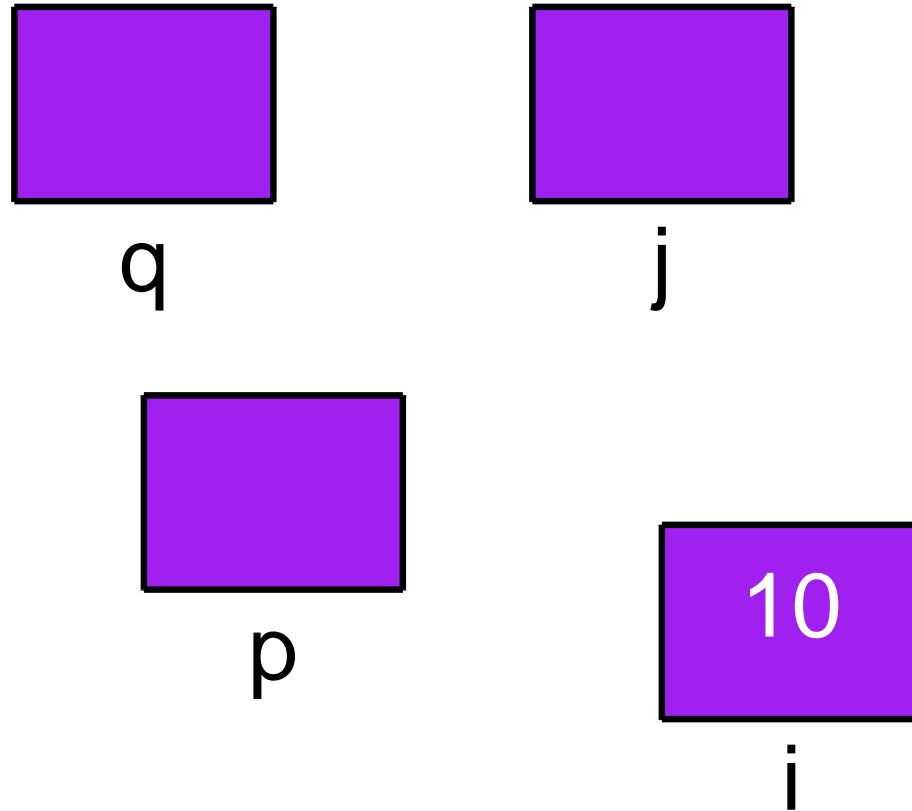
p = &i;         // p now points to i
printf("%p", p); // address of i (in p)

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

Pointers

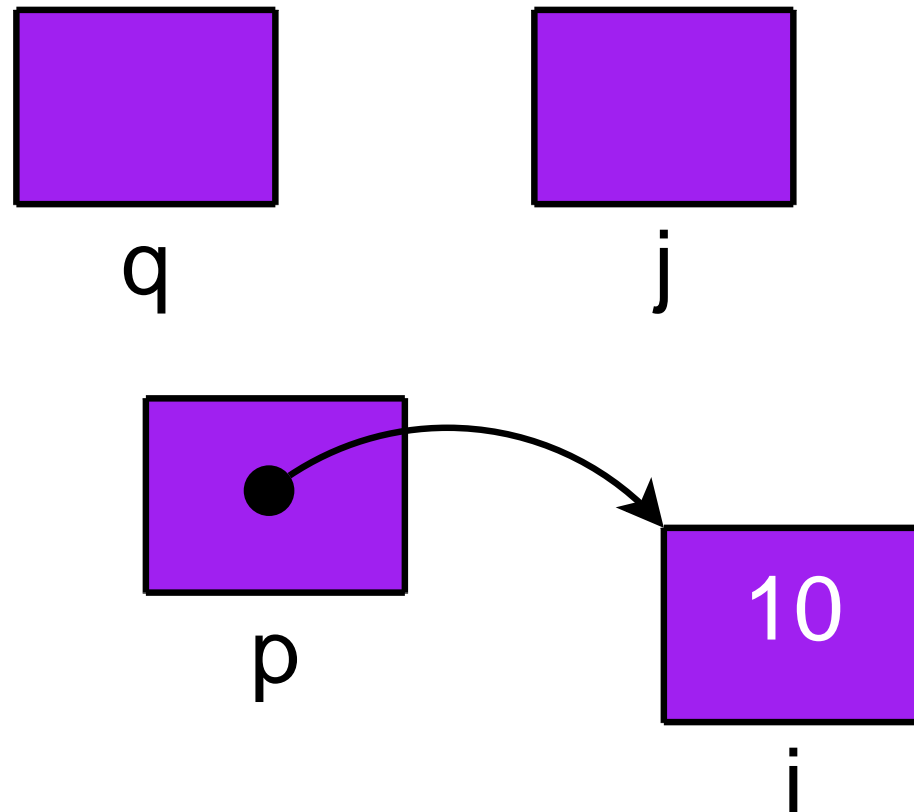


Pointers



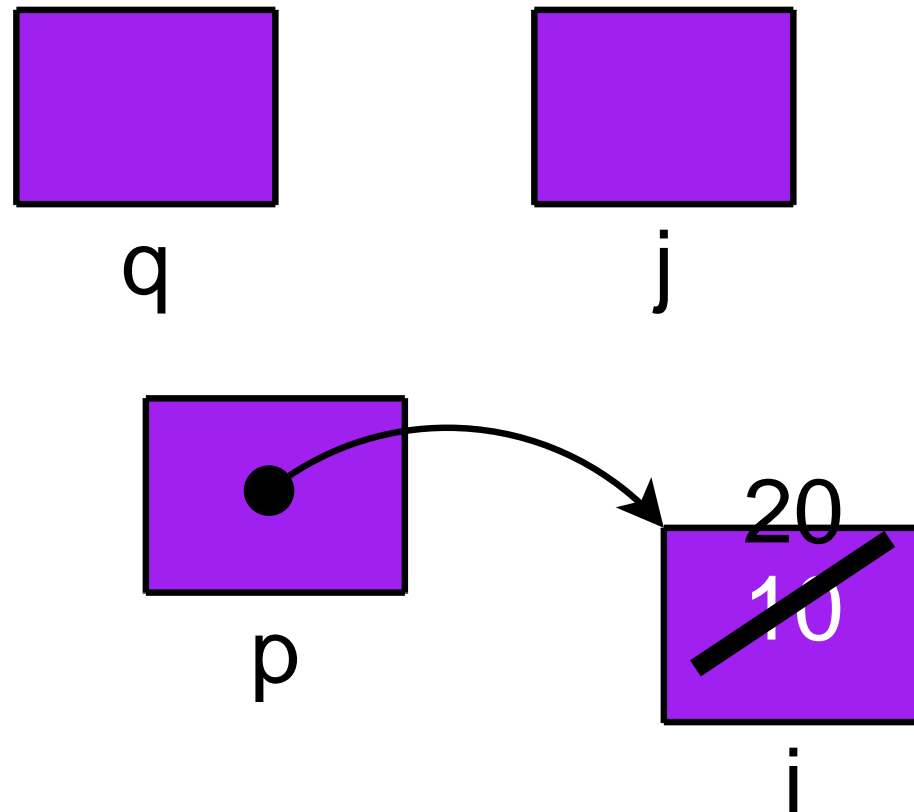
```
i = 10;
```

Pointers



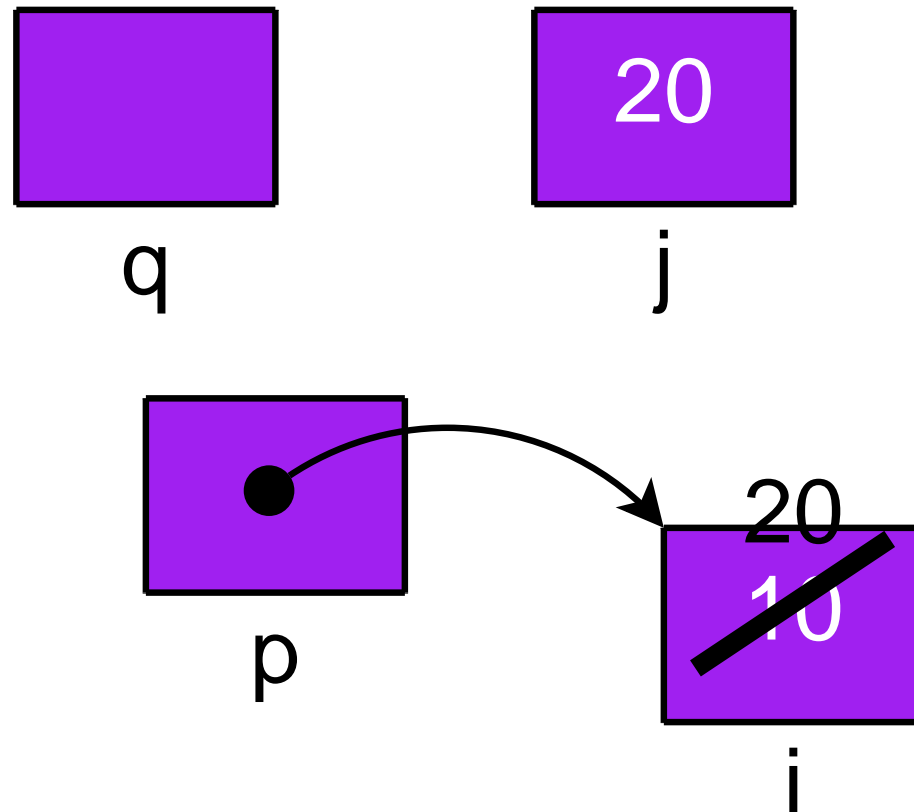
```
p = &i;
```

Pointers



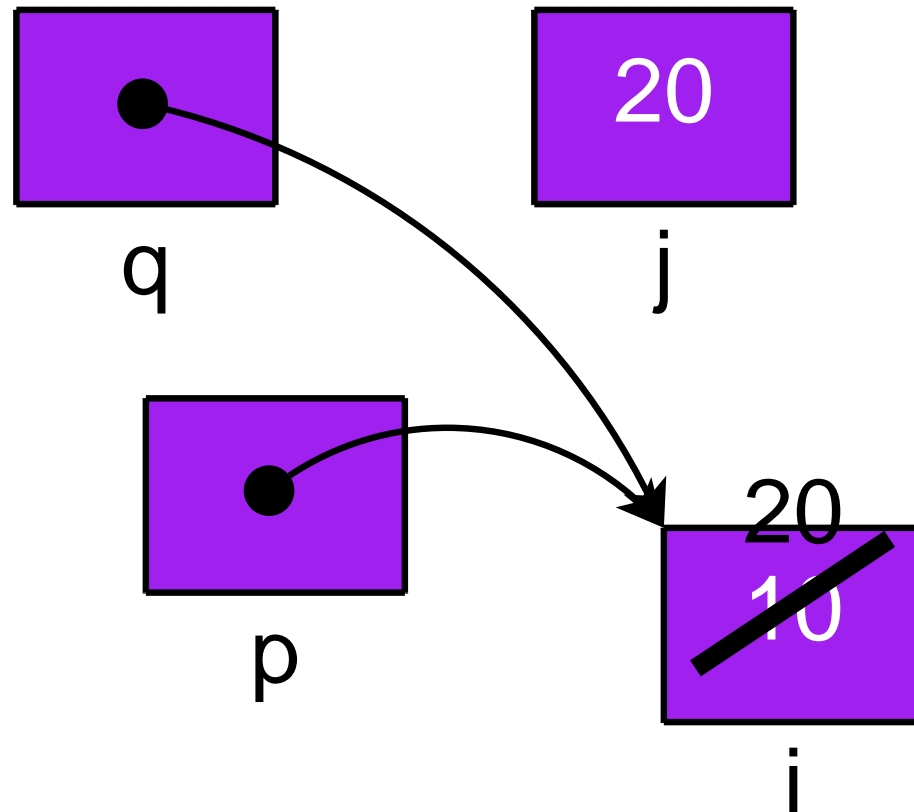
`(*p) = 20;`

Pointers

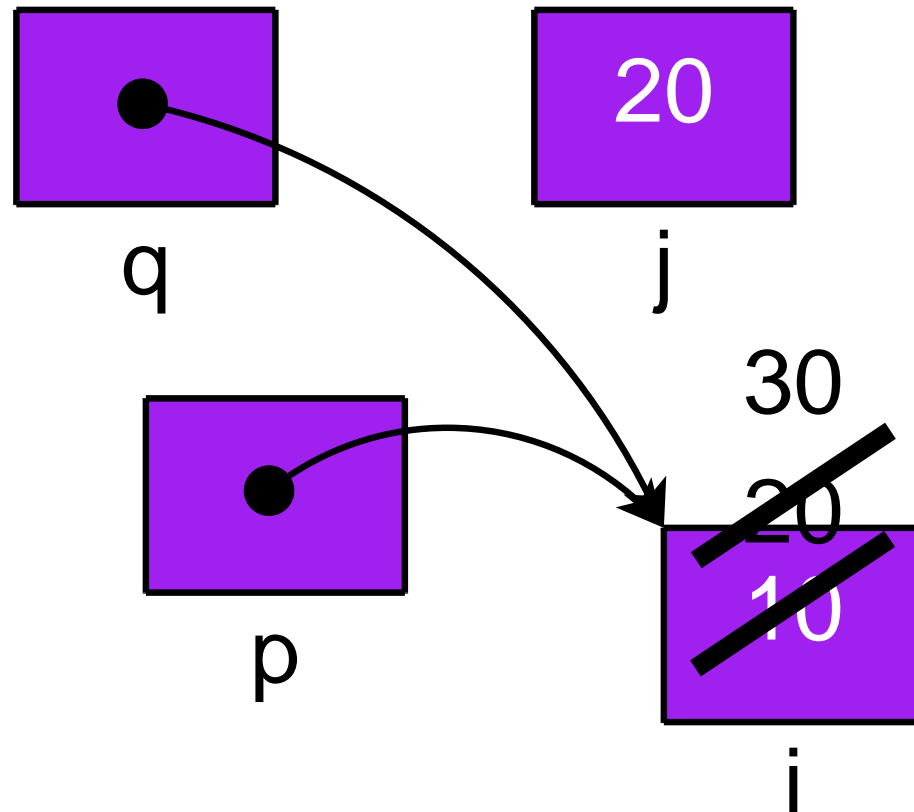


```
j = (*p);
```

Pointers

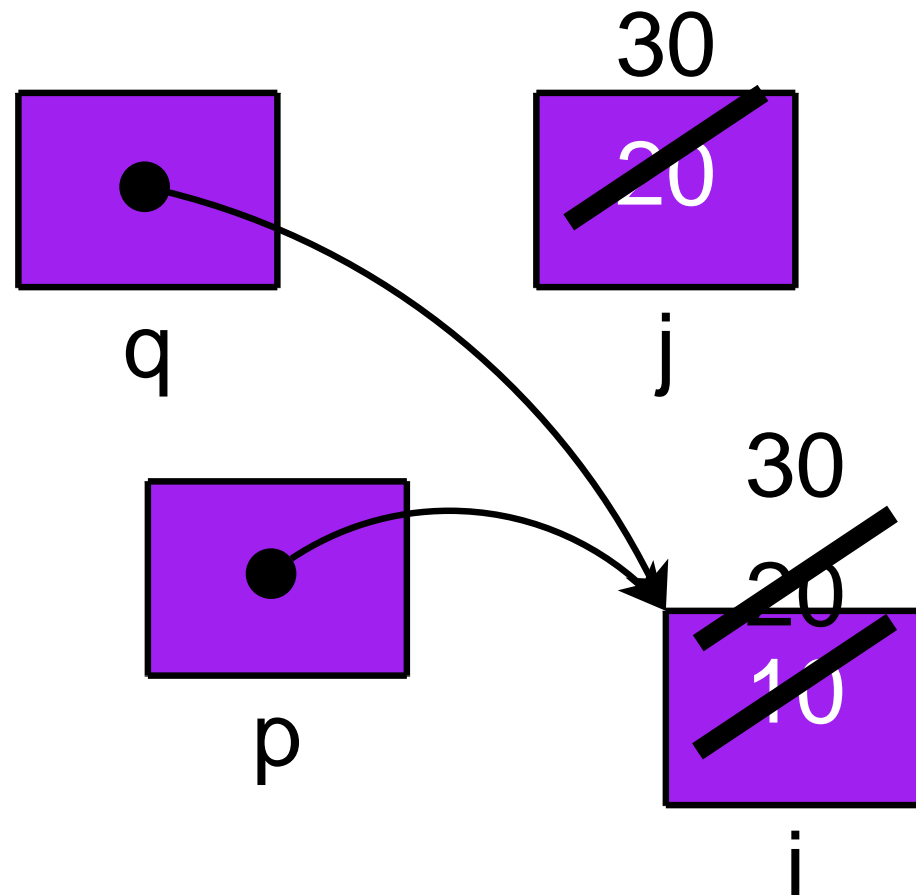


Pointers



```
(*q) = 30;
```

Pointers



```
j = (*p);
```

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

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

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

zardoz.c: Zardoz

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

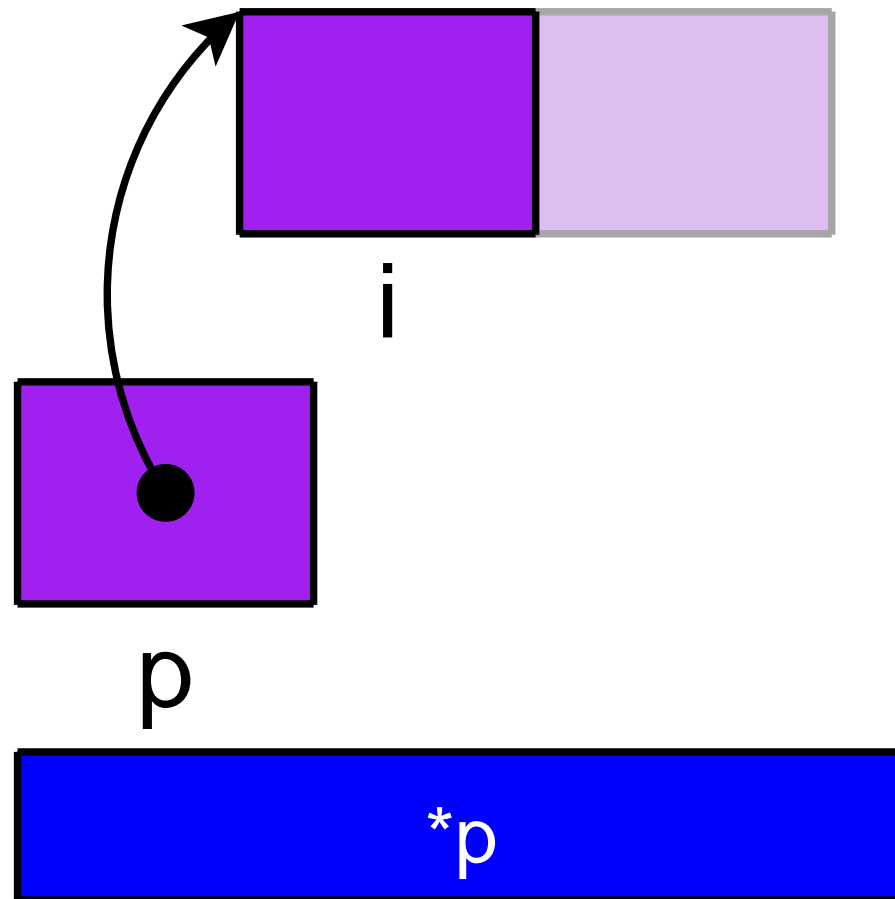
    printf("X: %d %d %d %d %d %d",
           **m, **n, *p, *q, a, b);

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

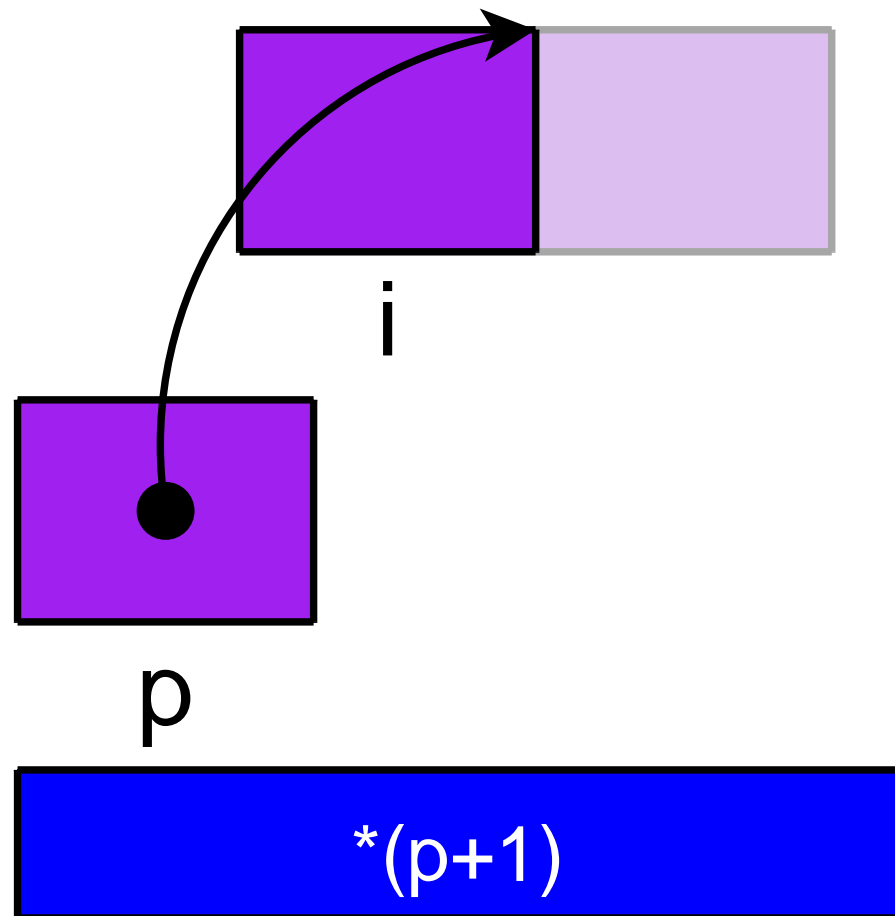
    printf("Y: %d %d %d %d %d %d",
           **m, **n, *p, *q, a, b);
    return 0;
}
```

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

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);  
}
```