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

```c
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

\[ \text{q} \quad \text{j} \]
\[ \text{p} \quad \text{i} \]
Pointers

\[ i = 10; \]
Pointers

```
p = &i;
```
Pointers

\[ (*p) = 20; \]
Pointers

j = (*p);
Pointers

Pointers

CS 2022, Fall 2009, Lecture 3
Pointers

\[(\ast q) = 30;\]
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

```c
#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);

    a = b; b = a;
    printf("After: %d, %d, %d, %d",
           a, b, *p, *q);

    return 0;
}
```c
#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);
    X: __ __ __ __ __ __

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

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

    return 0;
}
```
Pointer Arithmetic
Pointer Arithmetic

Diagram showing pointer arithmetic with a pointer `p` and an index `i`. The expression `*(p+1)` is indicated.
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.
#include <stdlib.h>

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

    *p = 42;
    return 0;
}

Nameless Variables
A poor man’s array

```c
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

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

int main() {
    int a, b;

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