Pointers

CS 113: Introduction to C

Instructor: Saikat Guha

Cornell University

Fall 2006, Lecture 3
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)
```
i = 10;
Pointers

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

```c
(*p) = 20;
```
j = (*p);
q = p;

Diagram:

- q is pointing to 20
- j is pointing to 20
- p is pointing to 10
\[(*q) = 30;\]
j = (*p);
```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: 10, 20, 20, 10
#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
#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;
}
#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: __ __ __ __ __ __
    Y: __ __ __ __ __ __

    *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;
}
Pointer Arithmetic

Diagram:

- `p` points to an object.
- `i` is an integer.
- The expression `*p` refers to the value stored at the memory location pointed to by `p`.

Note: The diagram illustrates basic pointer arithmetic concepts, including dereferencing a pointer (`*p`), and how pointers can be used to access memory locations.
Pointer Arithmetic

i

Pointers

*(p+1)
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;
}
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);
}
```
void getab(int *a, int *b) {
    *a = 10;
    *b = 20;
}

int main() {
    int a, b;

    getab(&a, &b);
}