:: show-addrs.c ::

```
#include <stdio.h>

/* show how the & operator can be applied to any varname */

struct person {
    int age;
    char name[30];
    char city[30];
};

int main()
{
    int x;
    char c;
    float a;
    char l[10];
    struct person p = { 12, "Lee", "Boston" };

    /* simple vars */
    printf("x is at %lu, c is at %lu, a is at %lu\n",
           (unsigned long)&x, (unsigned long)&c, (unsigned long)&a);

    /* arrays */
    printf("l is at %lu, l[2] is at %lu, l[100] is at %lu\n",
           (unsigned long)&l, (unsigned long)&l[2], (unsigned long)&l[100]);

    /* structs */
    printf("p is at %lu, p.age is at %lu, p.name is at %lu, p.city is at %lu\n",
           (unsigned long)&p, (unsigned long)&p.age,
           (unsigned long)&p.name, (unsigned long)&p.city);
}
```

:: assign-ptrs.c ::

```
/* assign-ptrs.c - show pointer assignment */

main()
{
    int x; /* an int */
    int *p; /* a pointer to an int */
    int *q; /* another ptr to an int */

    p=& x; /* store address of x in p */
    x=1 2; /* store a value in x */

    // added by class 2013
    printf("x is %d, p is %lu, q is %lu, *p = %d\n",
           x, (unsigned long)p, (unsigned long)q, *p);
    printf("about to set *q to *p...\n");
    *q = *p;

    q = p; /* copy a pointer value */
    printf("x is %d, p is %lu, q is %lu, *p = %d\n",
           x, (unsigned long)p, (unsigned long)q, *p);

    x = 100;
    x += 4;
    printf("x is %d, p is %lu, q is %lu, *p = %d\n",
           x, (unsigned long)p, (unsigned long)q, *p);

    *p = 200;
    *q += 3;
    printf("x is %d, p is %lu, q is %lu, *p = %d\n",
           x, (unsigned long)p, (unsigned long)q, *p);

    printf(" about to store 12 in location 200...\n");
    q = 200; /* the compiler will complain, but do it */
    *q = 12; /* stand back. */
}
#find_pals.c

```c
#include <stdio.h>

/*
 * find_pals -- find palindromes in input
 */

#define LEN 1000

int is_a_pal(char[]);

int main()
{
    char line[LEN];
    while( fgets(line, LEN, stdin) ){
        if ( is_a_pal(line) ){
            printf("%s", line);
        }
    }
}

/* returns 1 if arg is a palindrome, 0 if not */
/* note: "" IS a palindrome */
/* TODO: make it case insensitive */

int is_a_pal(char s[])
{
    char *left, *right;
    left = s[0];
    /* find end of string */
    for(right=&s[0]; *right != '\n' && *right != '\0' ; right++ )
    {
        right--;
    }
    /* do the check for palindrome */
    while( left < right ){
        if ( *left != *right )
        {
            return 0;
        }
        left++;
        right--;
    }
    return 1;
}
```

#ptr1.c

```c
#include <stdio.h>

// ptr1.c -- practice using pointers
// predict output AND draw memory diagram

int f(int*, int);

int main()
{
    int x,y;
    int *p, *q;
    y = 2;
    x = 3;
    p = &x;
    q = p;
    *q = *p - 2;
    *p = y;
    x = f(&y, *p);
    printf("%d %d %d %d\n", x, y, *p, *q);
    return 0;
}

int f(int* t, int u)
{
    *t = 2 * u;
    return 1 + *t;
}
```
------------ ptr2.c -----------
#include <stdio.h>

int main()
{
    int x, y;
    int *p, *q;
    int **z;

    y = 2;
    x = 3;
    p = &x;

    q = &y;
    z = &p;

    *z = 12;
    **z = 12;
    printf("%d\n", **z);
}

------------ ptr3.c -----------
#include <stdio.h>

// ptr3.c -- more practice using pointers
// predict output AND draw memory diagram

void f(char *, char *);
void show(char n[]);

int main()
{
    char n[] = "abcdefg";
    char *p, *q;

    show(n);
    p = &n[2];
    q = p + 1;
    f(p, q);
    show(n);
    f(p-1, q+1);
    show(n);
    return 0;
}

void show(char a[])
{
    int i;
    for(i=0; a[i] != '\0'; i++)
    {
        printf("%c ", a[i]);
        putchar('\n');
    }
}

void f(char* p1, char *p2)
{
    char temp;
    temp = *p1;
    *p1 = *p2;
    *p2 = temp;
}
```c
#include <stdio.h>

// ptr4.c -- more practice using pointers
// predict output AND draw memory diagram

int main()
{
    int a[5] = {0, 7, 2, 8, 3};
    int *p, *q;
    int **z;
    int m;
    int i;
    p = &a[2];
    q = p+1;
    z = &p;
    m = *p + *q;
    ++(*z);
    ++(*p);
    printf("%d %d %d\n", m, **z, *q);
    for(i=0; i<5; i++)
        printf("%d\n", a[i]);
    return 0;
}

int main()
{
    int y, x;
    int *p, *q;
    int **z;
    y = 2;
    x = 3;
    p = &x;
    q = p;
    *q = *p - 2;
    *p = y;
    printf("%d\n", x + *p + **q);
    printf("p and q are %lu and %lu\n", (unsigned long)p, (unsigned long)q);
    printf("about to store a long in a pointer...\n");
    p = (int *) 4;
    printf("p and q are %lu and %lu\n", (unsigned long)p, (unsigned long)q);
    printf("The value stored in location 4 is %d\n", *p);
    return 0;
}
```
```c
#include <stdio.h>
#define LEN 100

unsigned strlen_i(char []);
unsigned strlen_p(char []);
char *strcpy_p(char [], char []);

main()
{
    char s1[LEN], s2[LEN];
    int len;

    printf("enter a string: ");
    fgets(s1, LEN, stdin);
    len = strlen_i(s1);
    printf("The length of %s is %d\n", s1, len);
    strcpy_p(s2, s1);
    printf("The length of s2 = %s is %d\n", s2, strlen_p(s2));
    printf("that string is %s\n", strcpy_p(s2, "wow"));
}

unsigned strlen_i(char a[])
{
    unsigned pos = 0;
    while( a[pos] != '\0' )
    {
        pos++;
        return pos;
    }
}

unsigned strlen_p(char a[])
{
    char *p ; /* define a pointer */
    p = a;
    p = &a[0]; /* point to item 0 */
    while( *p != '\0' ) /* point to nul? */
    {
        p++;
        /* no: increment */
        return p - a; /* compute distance */
    }
}

unsigned strlen_p2(char a[])
{
    char *p ; /* define a pointer */
    p = a;
    while( *p++ ) /* point to nul? */
    {
        return --p - a;
    }
}
```
unsigned strlen_p3(char a[])  
{  
    char *p ; /* define a pointer */  
    int count=0;  

    p = a;  
    p = &a[0]; /* point to item 0 */  
    while( *p++ ) /* point to null */  
    {  
        count++;  
    }  
    return count;  
}  

char *strcpy_p(char dest[], char src[])  
{  
    char *dp = dest, *sp = src;  
    while( *dp = *sp ){  
        dp++;  
        sp++;  
    }  
    return dest;  
}  

char *strcpy_p2(char dest[], char src[])  
{  
    char *dp = dest, *sp = src;  

    while( *dp++ = *sp++ )  
    {  
    }  
    return dest;  
}