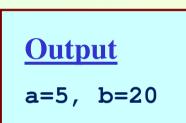
Passing Pointers to a Function

- Pointers are often passed to a function as arguments.
 - Allows data items within the calling program to be accessed by the function, altered, and then returned to the calling program in altered form.
 - Called *call-by-reference* (or by *address* or by *location*).
- Normally, arguments are passed to a function by value.
 - The data items are copied to the function.
 - Changes are not reflected in the calling program.

Example: passing arguments by value

```
#include <stdio.h>
main()
Ł
   int a, b;
   a = 5; b = 20;
   swap (a, b);
   printf ("\n a=%d, b=%d", a, b);
void swap (int x, int y)
   int t;
   t = x;
   \mathbf{x} = \mathbf{y};
   \mathbf{v} = \mathbf{t};
```



Example: passing arguments by reference

```
#include <stdio.h>
main()
Ł
   int a, b;
   a = 5; b = 20;
   swap (&a, &b);
  printf ("\n a=%d, b=%d", a, b);
}
void swap (int *x, int *y)
   int t;
  t = *x;
   *x = *y;
   *y = t;
```

```
<u>Output</u>
a=20, b=5
```

scanf Revisited

int x, y;
printf ("%d %d %d", x, y, x+y);

What about scanf ?

scanf ("%d %d %d", x, y, x+y) ; NO
scanf ("%d %d", &x, &y) ; YES

Example: Sort 3 integers

Three-step algorithm:

- 1. Read in three integers x, y and z
- 2. Put smallest in x
 - Swap x, y if necessary; then swap x, z if necessary.
- 3. Put second smallest in y
 - Swap y, z if necessary.

Contd.

```
#include <stdio.h>
main()
{
    int x, y, z;
    ......
    scanf ("%d %d %d", &x, &y, &z);
    if (x > y) swap(&x,&y);
    if (x > z) swap(&x,&z);
    if (y > z) swap(&y,&z);
    ......
}
```

sort3 as a function

```
#include <stdio.h>
main()
{
    int x, y, z;
    .....
    scanf ("%d %d %d", &x, &y, &z);
    sort3 (&x, &y, &z);
    .....
}
void sort3 (int *xp, int *yp, int *zp)
Ł
    if (*xp > *yp) swap (xp, yp);
    if (*xp > *zp) swap (xp, zp);
    if (*yp > *zp) swap (yp, zp);
```

Spring Semester 2011 Programming and Data Structure

Contd.

- Why no '&' in swap call?
 - Because xp, yp and zp are already pointers that point to the variables that we want to swap.

Pointers and Arrays

- When an array is declared,
 - The compiler allocates a base address and sufficient amount of storage to contain all the elements of the array in contiguous memory locations.
 - The base address is the location of the first element (*index 0*) of the array.
 - The compiler also defines the array name as a constant pointer to the first element.

Example

Consider the declaration:

int $x[5] = \{1, 2, 3, 4, 5\};$

 Suppose that the base address of x is 2500, and each integer requires 4 bytes.

Element	Value	Address
x[0]	1	2500
x[1]	2	2504
x[2]	3	2508
x[3]	4	2512
x[4]	5	2516

Contd.

Both x and &x [0] have the value 2500.

p = x; and p = &x[0]; are equivalent.

- We can access successive values of x by using p++ or p-- to move from one element to another.
- Relationship between p and x:

p = &x[0] = 2500	
p+1 = &x[1] = 2504	*(p+i) gives the value of x[i]
p+2 = &x[2] = 2508	
p+3 = &x[3] = 2512	
p+4 = &x[4] = 2516	

Programming and Data Structure

Example: function to find average

```
float avg (array, size)
#include <stdio.h>
main()
                                   int array[], size;
                                   {
  int x[100], k, n;
                                     int *p, i , sum = 0;
  scanf ("%d", &n);
                                     p = array;
  for (k=0; k<n; k++)</pre>
                                     for (i=0; i<size; i++)</pre>
     scanf ("%d", &x[k]);
                                         sum = sum + *(p+i);
  printf ("\nAverage is %f",
                                     return ((float) sum / size);
                 avq (x, n);
```