Pointers & Functions (Introduction)

CS2023 Winter 2004

Outcomes: Introduction to Pointers

- “C for Java Programmers”, Chapter 8
- Other textbooks on C on reserve
- After the conclusion of this section you should be able to
  - Write functions that modify more than one variable by using pointers as parameters

Pointers as Arguments

In order to modify a parameter `x`:

- Pass `&x` to the function
- Declare corresponding formal parameter `p` to be a pointer
- `p` will have the value `&x`, hence `*p` is an indirect reference to `x`
- Therefore can both read and modify `x`

Pointers as Arguments

Trace the execution of:

```
void swap(int *x, int *y) {
    int temp;

    temp = *x;
    *x = *y;
    *y = temp;
}
/* call:  int i = 2, j = 3; swap(&i, &j); */
```

Swap example

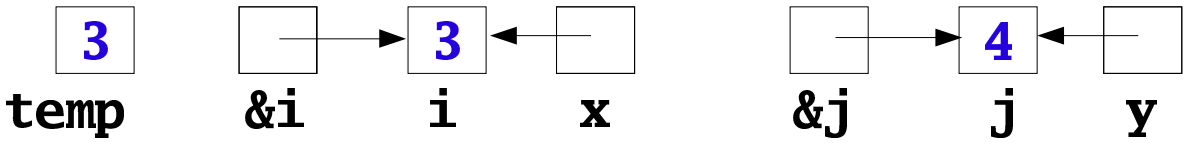
- Initial values: **`i = 3, j =4`**
- Before call to **`swap()`**

Swap example

- Immediately after call

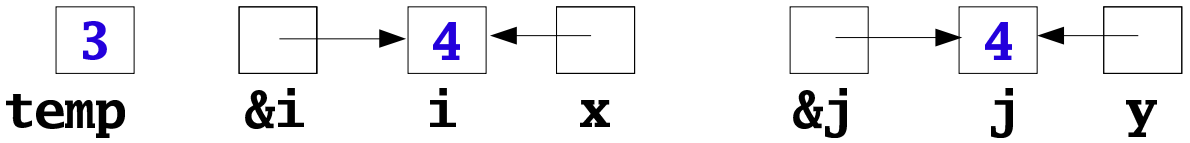
Swap example

- After **`temp = *x`**



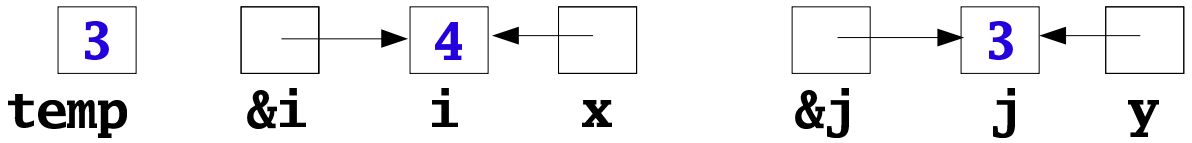
Swap example

- After `*x = *y`



Swap example

- After `*y = temp`



Swap example

- After return from **`swap()`**

Modifying an Argument to a Function

1. Declare the formal parameter **`FP`** as a pointer, for example **`int *FP`**
2. In the body of the procedure, dereference **`FP`**, that is use **`*FP`**
3. In the call
   - if the actual parameter **`AP`**, is a *variable*, use the address of **`AP`**;

     for example **`f(&AP)`**

   - if actual parameter **`AP`** is a *pointer*, use **`AP`** without the address operator;

     for example **`f(AP)`**

Example

Find the largest and smallest elements in an array:

```
void maxMin(int a[], int n, int *max, int *min)
{
  int i;

  *max = *min = a[0];
  for (i = 1; i < n; i++) {
    if (a[i] > *max)
      *max = a[i];
    else if (a[i] < *min)
      *min = a[i];
  }
}
```

Calling Program

```
#define N 100

void maxMin(int a[], int n, int *max, int *min);

int main() {
  int b[N], i, big, small;

  for (i = 0; i < N; i++)
    scanf("%d", &b[i]);

  maxMin(b, N, &big, &small);

  printf("Largest: %d\n Smallest: %d\n",
          big, small);
  return 0;
}
```

Pointers as Return Values

```
/* Given pointers to two integers, return
 * pointer to whichever integer is larger
*/
int *max(int *a, int *b)
{
  if (*a > *b)
    return a;
  else
    return b;
}
/* int *p, x, y; p = max(&x, &y); */
```

Caution!

- Never return a pointer to an *automatic* local variable

```
int *f()
{
  int i;
  ...
  return &i;
}
```

- The variable i doesn't exist once f returns, so the pointer to it won't be valid