

Weeks 5-6

- Pointers and Arrays
 - Basic pointer type
 - Pointers and Arrays
 - Address arithmetic
 - Pointer Arrays
- User-defined data types
 - Structures
 - Unions
 - Pointers and arrays of user-defined data types
- Next two weeks
 - Memory allocation and dynamic structures
 - Complex structures with pointers, structures, etc
 - Pointers functions

Basic Pointer

- Variable, memory storage and addresses

The diagram illustrates memory storage for different data types: a single cell for 'char', two cells for 'short', and four cells for 'int'. It also shows two pointer variables, 'ip1' and 'ip2', each represented by a single cell. 'ip1' has an arrow pointing to a memory cell, and 'ip2' has an arrow pointing to the first cell of an array of eight cells.

Basic Pointer -- continued

- Pointers and references

```
int i, *ip1, *ip2, a[8];
char c, *cp;

ip1 = &i;
ip2 = &a[0];
i = 8;
ip1 = ip2;

cp = &c;
```

Parameter Passing with Pointers

- Still Pass-by-value with pointers
- However the memory content can change
- Implications:
 - Be aware of the argument type when passing parameters
 - Be aware of the content is changing
 - When you attempt to change pointer argument values, ask yourself twice, typically a no-no.

Parameter Passing -- Example

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

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

Pointers and Arrays

- Essentially, pointers and array identifier are memory addresses.
- Array name, in particular, is the address of the first element

The diagram shows a pointer variable 'ip' and an array 'a'. 'ip' is shown with three memory addresses: 'ip', 'ip+1', and 'ip+2'. 'a' is shown as an array of eight cells, with the first cell labeled 'a[0]' and the last cell labeled 'a[7]'. Below the array, the first two cells are labeled 'ip[0]' and 'ip[1]', and the last two cells are labeled 'ip[7]'.

Address Arithmetic

- Pointers can have arithmetic operations!

```
int *ip1, *ip2, a[8];
```

```
ip1 = &a[7];  
ip2 = &a[0];
```

Then:

```
ip2 - ip1 + 1?  
ip1 - ip2 ?  
ip1 == NULL?  
ip1 > ip2 ?
```

Pointer Arrays

- Comparisons with multi-dimensional arrays
 - A typical usage, command line arguments

```
int *ip[3], a[3][4], i=0;
```

```
ip[0] = &a[0][0];  
ip[1] = &a[1][0];  
ip[2] = &a[2][0];
```

Sample Usage -- command-line arguments

```
int main (int argc, char * argv[])  
{  
    int i=0;  
    while (i++< argc) {  
        printf("%s\n", argv[i]);  
    }  
}
```