Arrays

Arrays in C (1)

- Group of memory locations referenced using the ***same name*** and with the ***same data type***
- Each memory location is assigned a ***position number*** or ***index***
- Declaration:

```
<data type> <array name>[<length>];
```

- Example:

```
int array[50]; //declares an integer
               //array 50 elements
```

Arrays in C (2)

- To access an array element, use array name and index
- Example:

```
array[10] = 90;
array[44] = 34;
```

- Range of the index = 0 .. (length – 1)
- To access the ith element, use index i – 1
- Example:
  - To access the 6th element, use `array[5]`

Initializing Arrays

- Arrays can be initialized upon declaration
- Example:

```
int array[5] = {1, 2, 3, 4, 5};
```

- If there are fewer initializers than elements, the rest are initialized to 0
- Can also be initialized using a loop
- Example:

```
for(i = 0; i <= 4; i++) {
    array[ i ] = 0;
}
```

Loops and Arrays

- Usually, a for-loop is used with arrays
- Recall: A for-loop is best for counter-controlled arrays
- Arrays have a fixed length that is usually known beforehand

The #define directive

- Used to declare ***symbolic constants***
- Syntax:

```
#define <SYMBOL> <constant>
```

- Example:

```
#define SIZE 10
main() {
    int array[SIZE];
}
```

- At compile time, C will replace the SIZE symbol in the array declaration to the constant 10 in the #define directive

Character Arrays and Strings (1)

- String declarations are of the form

```
    char name[30];
```

- Strings are actually represented as character arrays by C
- Strings can be initialized by string literals
- Example:

```
    char name[] = “Kei”
```

Character Arrays and Strings (2)

- Note that strings always allot an extra char element for the ***null character*** `('\0')`
- The null character denotes the end of the string
- If there is no null character, C may go beyond the bounds of the string in search of one
- Therefore, the declaration

```
    char name[30];
```

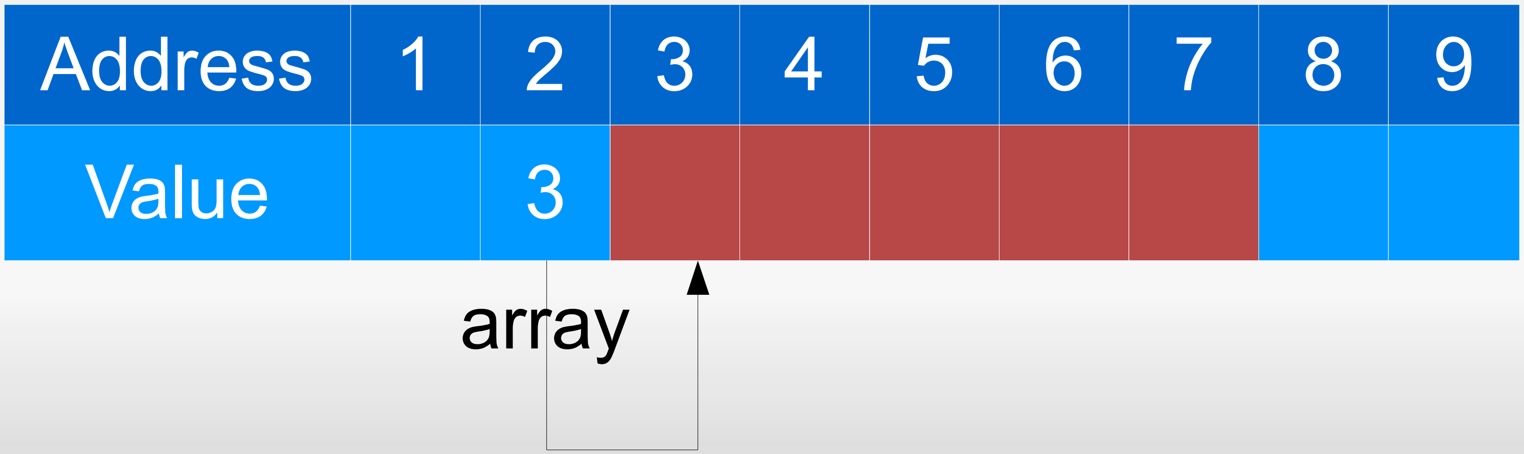
can only store 29 characters plus the null character

Memory Allocation of Arrays (1)

- When arrays are declared, they are allocated space in memory, just like normal variables
- However, for arrays, C needs to find a patch of memory with consecutive free memory cells equal to the length of the array
  - If array has length 10, then C looks for 10 consecutive free memory cells
- So to which memory cell is the array name assigned?

Memory Allocation of Arrays (2)

- Aside from the consecutive memory cells used for the array's elements, a pointer variable is allocated
- The value of the pointer variable is the address of the first memory cell for the arrays elements
- Example: `int array[5];`



Address	1	2	3	4	5	6	7	8	9
Value		3	🟥	🟥	🟥	🟥	🟥		