# CSE 1320 - Intermediate Programming

## Arrays and Strings

Alex Dillhoff

University of Texas at Arlington

## Arrays in C

Arrays in C are defined with a **type** and **size**.

- **type** - defines the type of each value in the array.
- **size** - informs the compiler as to how much space is required.

When an array is defined, a contiguous block of memory is allocated on the stack large enough to hold the requested values.

## Arrays in C

Arrays are declared using the following syntax:

**Syntax**

```
type identifier[size];
```

**Example**

```
// Create a character array of size 10
char my_array[10];
```

When declared this way, they are called **static** arrays. This is because the size of the array cannot change.

## Arrays in C

Array values can be accessed by **indexing**.

```c
// Access the first element
int value = array[0];
```

## Arrays in C

Arrays can be assigned values during their definition. Bulk assignments of the same value can be done with a loop.

```c
// Create array of size 10
// with values of 1
int a[10];

for (int i = 0; i < 10; i++) {
    a[i] = 1;
}
```

## Arrays in C

Specific values can be assigned during initialization.

```c
char word[4] = {'w', 'o', 'r', 'd'};
```

## Arrays in C

The size of the array can be omitted if the values are assigned *explicitly*.

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

## Arrays in C

Left uninitialized, the values of an array are unspecified.

```
int a[5];

for (int i = 0; i < 10; i++) {
    printf("%d ", a[i]);
}
```

**Output** 654595952 21870 654595568 21870 1939332480

## Remembering Size

Arrays do NOT implicitly keep track of their size. **The programmer must do this manually!**

A common convention for using arrays with other functions is to include the number of elements of the array.

```
// Array processing func definition
int process_array(int a[], int len) {
    // do some processing
}
```

## Arrays in Function Declarations

The name of the array does not need to be included in function declarations.

```
// Array processing func declaration
int process_array(int[], int);
```

## Arrays as Arguments

To pass an array as an argument to a function, simply use the identifier itself.

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

process_array(a, 5);
```

Passing the name of the array itself refers to the address of the first value in that array.

## Array Examples

- Array Search
- Selection Sort

## Strings in C

In C, a string is a sequence of characters terminated by the null character `'\0'`.

For the string constant `"this is a string.\n "`, the address of the constant refers to the first character in the sequence.

## Strings in C

Static string variables can be created one of two ways:

1) Using a `char` array with predefined size:

```
    char s[10] = "Test";
```

2) Using a `char` array with implicitly defined size:

```
    char s[] = "This is a string.\n";
```

Each of the strings will be terminated with `'\0'`.

## Strings in C

As with any other array, the address of the variable refers to the first character in the string.

**Example: String Address**

## String I/O

Strings can be printed by passing the address of the array itself as input to `printf()` or by using the `%s` specifier.

**Example: Print Strings**

## String I/O

Strings can be read from standard input using `fgets`.

```
char s[128];

fgets(s, 128, stdin);

printf("%s", s);
```

The `string.h` library provides many useful string functions.

- `strlen()`
- `strcpy()`
- `strncpy()`
- `strcat()`
- `strcmp()`
- `strncmp()`
- `strchr()`
- `strstr()`
- `strtok()`