Tutorial 7

• Arrays.

• Pointer arithmetic.

• Abstract Data Types (ADT).
Arrays

They can be used to store a **fixed number** of elements of the **same** type.

Example of array syntax:

```c
int my_array[3] = { 1, 2, 3 };
int x = my_array[0]; // x = 1
```
Array Initialization

There are several ways to define an array:

```c
int a[3]; // array is not initialized, but it's defined
int b[3] = { 1, 2, 3 }; // array is initialized
int d[3] = {0}; // array of length 3, filled with zeros
int e[8] = { 7, 4, 1 }; // {7, 4, 1, 0, 0, 0, 0, 0}
Array Exercise

// reverse_array(arr, len) reverses the contents of arr
// requires: arr is an array with length (at least) len
// effects: modifies arr
void reverse_array(int arr[], int len);
Pointer Arithmetic

Certain arithmetic operations can be performed on pointers. An integer can be **added or subtracted** to a pointer, and pointers of the same type can be **subtracted** from one another.

```c
int a[10];
int *p = a;   // a is a pointer to first element
int *q = &a[9]; // address of 10th element
q = a + 9; // equivalent
a[2] = q - p; // set the value of 3rd element as 9
q = p + 1; // now q == &a[1]
```

Addition of pointers is not allowed.
Array Exercise 2

Write Reverse again, now using pointer arithmetic
(hint, this code will be essentially identical to reverse)

// reverse_array(arr, len) reverses the contents of arr
// requires: arr is an array with length (at least) len
// effects: modifies arr
void reverse_array(int *arr, int len);

The syntax a[i] is shorthand for the equivalent expression *(a+i).
ADT Exercise

Write the following program using Stack as an ADT

// Read in integers from input and print
// them in reverse order WITHOUT using recursion