Objects & Structures

Objects are a way to think about grouping together similar items in order to organize their structure and inter-relations. In programming, objects are a way to group variables together (as in arrays). *Arrays* allow us to group together objects so long as they are the same type.

```
int grades[150];
```

But what if we want more flexibility? We need to move away from the *primitive* variable types and begin to group different variables together in order to make a sort of container.

Structs are custom objects (structures) that allow us to mix and match data types. Traditionally, structures contain only data and no member functions i.e. a clump of related variables. The following shows an example.

```
struct book{
  int pages;
  unsigned in pub_date;
  string title; // a string inside the struct
  int num_authors;
  string* authors; // a pointer to a string
};

// declare a book struct
book text_book;
```

Inside of *book* we have defined a number of useful variables such as a string for the title, an int for the number of pages, etc... How do we access these?

```
book bookshelf[10];
for (int i = 0; i < 10; i++){
  bookshelf[i].num_pages = 100;
  bookshelf[i].title = ``Place holder''
  bookshelf[i].authors = new string[2]; // dynamically allocate array
}
```

Pointers

- Pointers == memory addresses.
- Variable declaration creates a variable on the stack

  ```
  int a = 5;
  ```

- Pointer declaration

  ```
  int* b = &a;
  ```

  This creates a pointer variable of type **int** which points to the address of **a** (using the address operator &)

- Dereferencing a pointer:

  ```
  cout << *b << endl;
  ```

  This will take the pointer b and grab the variable held at that memory location.

Array

- An *array* is a collection variables of one data type and its memory is stored contiguously
- *static arrays* are created on the stack and are of a fixed size

  ```
  int stack_array[10];
  ```

- *Dynamic arrays* are created on the heap and their size may change during runtime.

  ```
  int *heap_array = new int[10];
  ```

- Arrays can be of one ore more dimensions

  ```
  int stack_array_2d[5][7];
  ```