Pointers and Strings
Chapters 10, 12
Pointers and Arrays (10.3)

- An array of ints can be declared as
  - `int numbers[] = {1, 2, 3, 4, 5};`
- `numbers` is also a pointer to the first element in the array
- Therefore, it can be dereferenced to access the elements of the array
  - `*numbers = 2;`
  - What are the contents of the above array now?
Pointers and Arrays (10.3)

• The name of the array is a pointer to the *first* element in the array

• What about the other elements in the array?
  ○ You can add 1 to the array name to access the second element
  ○ You can add 2 to the array name to access the second element….and so on

• When adding a number to the array name, you are actually adding that number times the size of the element in the array
int numbers[] = {1, 2, 3, 4, 5};
*(numbers + 1) = 1;
*(numbers + 2) = 1;
*(numbers + 3) = 1;
*(numbers + 4) = 1;

• What are the contents of the array now?
• What would happen if we did the following:
  ○ *(numbers + 5) = 1;
Pointers and Arrays (10.3)

• Rewrite the following so that it uses pointer notation instead of subscript notation

```cpp
for(int x = 0; x < 100; x++)
{
    cout << array[x] << endl;
}
```
Strings

- What is a string in C++?
- How have we declared string variables? We have used two ways.
C-Strings (12.1)

- In C++, strings are arrays of characters that end in the null character \0

- A C-string can be declared as:
  
  - `char pet[] = "cat";`
  - `char *pPet = "cat";`
Strings and Pointers

- When declaring an array, the name of the array is also a constant pointer to the first element in the array

```cpp
int array[] = {2, 4, 6, 8};
int *pArray;

pArray = array;
pArray = &array[0];
cout << array[2]
    << *(pArray + 2);
pArray ++;
array ++; // ERROR
```
Strings

• Assuming that the string pet has been declared as:
  
  \[
  \text{char pet[]} = \text{“cat”};
  \]

• Write a function that will output the contents of the string. The function should accept the array and its size

• Write a function that will output the contents of the string. The function should accept a pointer to char
Strings and Pointers

- Write a function `strLength` that accepts a string (as a pointer) and returns the length of the string
int strLength (const char *pStr)
{
    int index;
    for (index = 0; *(pStr + index) != '\0'; index ++);
    return index;
}

• What is the purpose of const in the function header?
• Is the ; at the end of the for loop a mistake?
• What would happen if the ; was eliminated?
int strLength2 (char *pStr) {
    char *pTemp = pStr;
    while (*pTemp) {
        pTemp ++;
    }
    return pTemp - pStr;
}
What is happening?

```c
int sumInts (int *pArray, int size)
{
    int sum = 0;
    int index;

    for (index = 0; index < size; index ++)
    {
        sum += *pArray ++;
    }

    return sum;
}
```

• `int array[] = {10, 20, 30, 40, 50};` creates an array as follows:

<table>
<thead>
<tr>
<th>Address</th>
<th>Value</th>
<th>Element</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>2004</td>
<td>20</td>
<td>1</td>
</tr>
<tr>
<td>2008</td>
<td>30</td>
<td>2</td>
</tr>
<tr>
<td>2012</td>
<td>40</td>
<td>3</td>
</tr>
<tr>
<td>2016</td>
<td>50</td>
<td>4</td>
</tr>
</tbody>
</table>
Constant Pointers

- So far we have seen:
  - Nonconstant pointers to nonconstant data
  - Nonconstant pointers to constant data
- What about constant pointers?
- We said that array names are constant pointers to the first element in the array. What does that mean?
Constant Pointers

```c
int * const pNum, num, num2;
num = 9;
num2 = num + 8;
pNum = &num;
*pNum *= 2;
pNum = &num2; // ERROR
```

- pNum has been declared as a constant pointer
- It cannot point to any other memory location
Arrays of Pointers

• What do you make of the following declaration?

```cpp
```

• What gets output in each of the following cases?

```cpp
cout << cardSuits[1] << endl;
cout << *cardSuits[1] << endl;
```