C++ Week 7 Function Pointers and Templates

Function Pointers

Exercise 1

Within a new C++ Console Project create two functions

1. Named HelloMessage that returns void, has no parameters and displays the text **"Hello"** on the screen.
2. Named Goodbye that returns void, has no parameters and displays the text ***"Goodbye"*** on the screen.

Within main create a single pointer to the function type and invoke each of the functions through the pointer.

Exercise 2

Create a new function named Message that returns void but accepts a const string reference as an argument, that it displays on the screen. Using a pointer invoke the function.

Hint: include the string library <string>

Exercise 3

The function below sorts an array of integers in ascending order and print them to the screen. Add it to you program and from within main define an array containing 9 values { 3, 7, 9, 5, 6, 1, 8, 2, 4 } and pass them to the Sort function. Run the program to ensure the function is working correctly.

```
void Sort(int *numbers, int size) {
   for (int n= 0; n < size; n++) {
                int nBestIndex = n;
                for (int i = n + 1; i < size; i++){
                        if (numbers[i] < numbers[nBestIndex])
                                nBestIndex = i;
                }
                swap(numbers[n], numbers[nBestIndex]);
    }
    for (int i=0; i < size; i++)
                cout << numbers[i] << endl;
}
```

C++ Week 7 Function Pointers and Templates

Exercise 4

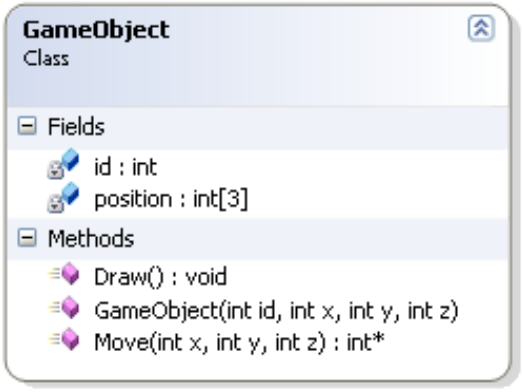
- Modify the sort function so that it takes a third argument, a pointer to a function that is responsible for determining in which order the array is sorted. You can think of the function as being a replacement for the < operator, in the if statement.
- Now add the function below and a second named Descending, which sorts in descending order.
- Add appropriate code to main that will invoke Sort twice, each time passing a pointer to a different function.

```
bool Ascending(int nX, int nY) {
        return nY > nX;
}
```

Member Function Pointers

Exercise 5

Define the GameObject class below.



Members

- id : unique identifier that is assigned during construction.
- position : represents the x,y,z location of the game object
- Constructor : Set the id and position to their respective values.
- Draw : displays the objects Class Name and id on the console
- Move- adds the parameter values to the respective data members and returns a pointer to the updated position.

C++ Week 7 Function Pointers and Templates

Exercise 6

From within main create a static instance of the GameObject and using a pointer to the Draw member function invoke it.

Exercise 7

From within main using a pointer to the Move member function, invoke it passing the arguments 1,2,3 and display the return value.

Function Template

Exercise 8

Define a new template function named DisplayArray that returns void and has two parameters, one being a pointer to an array and the second the size of the array. The function displays the contents of the array.
From within main define and initialise an array of 9 integer value and invoke DisplayArray to display their values.
Now define and initialise an array of 9 floats and pass this to the same DisplayArray function.

Exercise 9

Create a new template function SumArray that is passed a pointer to an array and the size of the array and returns the sum of all the elements. Within main invoke SumArray twice, once passing the array of integer values and once the float values. In each case print the sum on the screen.

Exercise 10

Modify the GameObject class so that it becomes a template class that can assign different types to the position data member. Test the new template class from within main by creating int and float versions and invoking their Move and Draw member functions using pointers.