1 Loops

Computers are great for doing the same thing over and over. C++ provides four looping mechanisms: the while loop, do/while loop, for loop, and *ranged-based* for loop.

1.1 The while loop

Perform a block of code repeatedly as long as an expression evaluates to true.

```
int i = 0;
while (i < 5) {
    std::cout << "This is silly" << std::endl;
    ++i;
}
```

Keyword **break** can be used within a loop to immediately get out of the loop and continue with the rest of the program. The keyword **continue** can be used to return to the top of the loop and reevaluate the while expression.

1.2 The do/while loop

Similar to while loop, except that the code to be executed comes first, and the conditional check for whether or not to continue happens at the end. So the code will always be executed at least once and maybe additional times based on the condition.

```
int i = 100;
do {
   std::cout << "This is silly." << std::endl;
   ++i;
} while (i<5);
```

1.3 The for loop

The for loop is the same as while loop but the syntax is often more convenient because it looks at a loop in terms of a starting expression, an ending condition, and a statement to execute at the end of every iteration.

```
for (int i = 0; i < 5; ++i) {
   std::cout << "This is silly." << std::endl;
}
```

1.4 Range-Based for loop

This type of loop works for C-style arrays, initializer lists, and any type that has a begin() and end() method returning iterators, such as std::array and all other STL containers. The following first defines an arary of four integers. Then the ranged-based for loop iterates over a **copy** of every element in this array and prints each value.

```
std::array<int,4> arr = {1,2,3,4};
for (int i: arr) {
   std::cout << i << std::endl;
}
```