STL Functors (Function Objects)

STL Functors (Function Objects)

- STL functors (*function objects*) declare & define **`operator()`**

```
template <typename T>
struct is_odd {
   bool operator()(const T& t) const {
     return (t % 2) == 1;
  }
};
```

See en.wikipedia.org/wiki/Function_object#In_C_and_C++

STL Functors (Function Objects)

- STL provides helper base class templates **`unary_function`** & **`binary_function`** to facilitate user-defined functors

```
template <typename _Arg,
          typename _Result>
struct unary_function {
  typedef _Arg argument_type;
  typedef _Result result_type;
};

template <typename _Arg1,
          typename _Arg2,
          typename _Result>
struct binary_function {
  typedef _Arg1 first_argument_type;
  typedef _Arg2 second_argument_type;
  typedef _Result result_type;
};
```

STL Functors (Function Objects)

- STL provides a # of popular functor class templates

- **Arithmetic**: `plus`, `minus`, `divides`, `times`, `modulus`, `negate`
- **Comparison**: `equal_to`, `not_equal_to`, `greater`, `less`, `greater_equal`, `less_equal`
- **Logical**: `logical_and`, `logical_or`, `logical_not`

STL Functors (Function Objects)

- Many STL generic algorithms can take STL-provided or user-defined functor arguments to extend algorithm behavior

```
vector<int> aVect {3, 2, 5, 4};
vector<int> bVect;
auto first = aVect.begin();
auto last = aVect.end();

transform(first, last,
          first,
          back_inserter(bVect),
          multiplies<>());

transform(first, last,
          bVect.begin(),
          bind2nd(divides<int>(), 3));

sort(first, last, greater<>());
```

STL Functor Examples

```
#include <vector>
#include <algorithm>
#include <functional>
#include <string>

int main (int argc, char *argv[]) {
  std::vector <std::string> projects;

  for (int i = 0; i < argc; ++i)
    projects.push_back(std::string (argv [i]));

  std::sort (projects.begin (), projects.end (),
             std::greater<std::string> ());

  return 0;
}
```

See github.com/douglascraigschmidt/CPlusPlus/tree/master/STL/S-08