Functors and for_each

BugSquashing Seminar
14.05.2014

Based on: http://www.bogotobogo.com/cplusplus/functors.php

What is a functor?

- Any object that can be used with ()
  - Normal functions
  - Pointer to functions
  - Class objects for which the () operator is overloaded

Pointer to functions

```
#include <iostream>

double add(double left, double right) {
    return left + right;
}

double multiply(double left, double right) {
    return left * right;
}

double binary_op(double left, double right, double (*f)(double,
double)) {
    return (*f)(left, right);
}

int main( ) {
    double a = 5.0;
    double b = 10.0;

    std::cout << "Add: " << binary_op(a, b, add) << std::endl;
    std::cout << "Multiply: " << binary_op(a, b, multiply) <<
std::endl;

    return 0;
}
```

```
Add: 15
Multiply: 50
```

Pointer to functions

- Legacy feature of C
- Drawbacks:
  - Low efficiency
  - Unflexible
  - No Templates

  → Should not be used in C++

Another functor

```
#include <iostream>

struct absValue
{
    float operator()(float f) {
        return f > 0 ? f : -f;
    }
};

int main( )
{
    using namespace std;

    float f = -123.45;
    absValue aObj;
    float abs_f = aObj(f);
    cout << "f = " << f << " abs_f = " << abs_f << endl;
    return 0;
}
```

```
f = -123.45 abs_f = 123.45
```

Function class objects

```
#include <iostream>
using namespace std;

class Line {
    double a;    // slope
    double b;    // y-intercept

public:
    Line(double slope = 1, double yintercept = 1):
        a(slope),b(yintercept){}
    double operator()(double x){
        return a*x + b;
    }
};

int main () {
    Line fa;                    // y = 1*x + 1
    Line fb(5.0,10.0);          // y = 5*x + 10

    double y1 = fa(20.0);       // y1 = 20 + 1
    double y2 = fb(3.0);        // y2 = 5*3 + 10

    cout << "y1 = " << y1 << " y2 = " << y2 << endl;
    return 0;
}
```

```
y1 = 21 y2 = 25
```

Stl::for_each

```
template<class Iterator, class Function>
Function for_each(Iterator first, Iterator last, Function f)
{
    while (first != last)
    {
        f(*first);
        ++first;
    }
    return f;
}
```

for_each example

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;

class Print {
public:
    void operator()(int elem) const {
        cout << elem << " ";
    }
};

int main () {
    vector<int> vect;
    for (int i=1; i<10; ++i) {
        vect.push_back(i);
    }

    Print print_it;
    for_each (vect.begin(), vect.end(), print_it);
    cout << endl;
    return 0;
}
```

```
1 2 3 4 5 6 7 8 9
```

for_each example

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;

template <int val>
void setValue(int& elem)
{
    elem = val;
}

template <typename T>
class PrintElements
{
public:
    void operator()(T& elm) const {cout << elm << ' ';}
};

int main()
{
    int size = 5;
    vector<int> v(size);
    PrintElements<int> print_it;
    for_each(v.begin(), v.end(), print_it);
    for_each(v.begin(), v.end(), setValue<10>);
    for_each(v.begin(), v.end(), print_it);
    return 0;
}
```

```
0 0 0 0 0 10 10 10 10 10
```

Functors and for_each

BugSquashing Seminar
14.05.2014

Based on: http://www.bogotobogo.com/cplusplus/functors.php

What is a functor?

- Any object that can be used with ()
  - Normal functions
  - Pointer to functions
  - Class objects for which the () operator is overloaded

Pointer to functions

```
#include <iostream>

double add(double left, double right) {
    return left + right;
}

double multiply(double left, double right) {
    return left * right;
}

double binary_op(double left, double right, double (*f)(double,
double)) {
    return (*f)(left, right);
}

int main( ) {
    double a = 5.0;
    double b = 10.0;

    std::cout << "Add: " << binary_op(a, b, add) << std::endl;
    std::cout << "Multiply: " << binary_op(a, b, multiply) <<
std::endl;

    return 0;
}
```

```
Add: 15
Multiply: 50
```

Pointer to functions

- Legacy feature of C
- Drawbacks:
  - Low efficiency
  - Unflexible
  - No Templates

  → Should not be used in C++

Another functor

```
#include <iostream>

struct absValue
{
    float operator()(float f) {
        return f > 0 ? f : -f;
    }
};

int main( )
{
    using namespace std;

    float f = -123.45;
    absValue aObj;
    float abs_f = aObj(f);
    cout << "f = " << f << " abs_f = " << abs_f << endl;
    return 0;
}
```

```
f = -123.45 abs_f = 123.45
```

Function class objects

```
#include <iostream>
using namespace std;

class Line {
     double a;    // slope
     double b;    // y-intercept

public:
     Line(double slope = 1, double yintercept = 1):
          a(slope),b(yintercept){}
     double operator()(double x){
          return a*x + b;
     }
};

int main () {
     Line fa;                  // y = 1*x + 1
     Line fb(5.0,10.0);        // y = 5*x + 10

     double y1 = fa(20.0);     // y1 = 20 + 1
     double y2 = fb(3.0);      // y2 = 5*3 + 10

     cout << "y1 = " << y1 << " y2 = " << y2 << endl;
     return 0;
}
```

```
y1 = 21 y2 = 25
```

Stl::for_each

```
template<class Iterator, class Function>
Function for_each(Iterator first, Iterator last, Function f)
{
    while (first != last)
    {
        f(*first);
        ++first;
    }
    return f;
}
```

for_each example

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;

class Print {
public:
    void operator()(int elem) const {
        cout << elem << " ";
    }
};

int main () {
    vector<int> vect;
    for (int i=1; i<10; ++i) {
        vect.push_back(i);
    }

    Print print_it;
    for_each (vect.begin(), vect.end(), print_it);
    cout << endl;
    return 0;
}
```

```
1 2 3 4 5 6 7 8 9
```

for_each example

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;

template <int val>
void setValue(int& elem)
{
    elem = val;
}

template <typename T>
class PrintElements
{
public:
    void operator()(T& elm) const {cout << elm << ' ';}
};

int main()
{
    int size = 5;
    vector<int> v(size);
    PrintElements<int> print_it;
    for_each(v.begin(), v.end(), print_it);
    for_each(v.begin(), v.end(), setValue<10>);
    for_each(v.begin(), v.end(), print_it);
    return 0;
}
```

```
0 0 0 0 0 10 10 10 10 10
```