```cpp
// $Id: accumulate.cpp,v 1.4 2012-06-05 20:48:14-07 - - $

//
// Accumulate.
// Takes an object and an action and accumulates a sequence,
// producing a final result.
//

#include <iostream>
#include <list>
#include <vector>

using namespace std;

//
// algorithm for_each_do
//
template <typename itor_t, typename action_t>
void for_each_do (const itor_t &begin, const itor_t &end,
                  action_t &action) {
   for (itor_t itor = begin; itor != end; ++itor) action (*itor);
}

//
// struct accumulator
//
template <typename value_t, typename binary_fn>
struct accumulator {
   value_t value;
   binary_fn *fn;
   accumulator (const value_t _value, const binary_fn _fn):
                value(_value), fn(_fn) {}
   void operator() (const value_t &opnd) {
      value = fn (value, opnd);
   }
};

//
// algorithm accumulate
//
template <typename itor_t, typename value_t, typename binary_fn>
value_t accumulate (const itor_t &begin, const itor_t &end,
                    const value_t &initial, const binary_fn &binfn) {
   accumulator<value_t, binary_fn> accum (initial, binfn);
   for_each_do (begin, end, accum);
   return accum.value;
}

//
// algorithm copyit
//
template <typename itor_t, typename dest_t>
void copyit (const itor_t &begin, const itor_t &end,
             const dest_t &dest) {
   dest_t desti = dest;
   for (itor_t itor = begin; itor != end; ++itor) *desti++ = *itor;
}

```

```

//
// Rest is local user code for testing.
//
template <typename value_t>
value_t add (const value_t &left, const value_t &right) {
   return left + right;
}

template <typename value_t>
value_t multiply (const value_t &left, const value_t &right) {
   return left * right;
}

double array[] = {3.1, 4.1, 5.9, 2.6, 5.3};
double *a_end = array + sizeof array / sizeof *array;

void array_test() {
   cout << "array_test:" << endl;
   cout << accumulate (&*array, a_end, 0.0, add<double>) << endl;
   cout << accumulate (&*array, a_end, 1.0, multiply<double>) << endl;
}

void vector_test() {
   vector<double> vec (a_end - array); // reserve enough space
   copyit (&*array, a_end, vec.begin());
   cout << "vector_test:" << endl;
   cout << accumulate (vec.begin(), vec.end(), 0.0, add<double>)
        << endl;
   cout << accumulate (vec.begin(), vec.end(), 1.0, multiply<double>)
        << endl;
}

int main() {
   array_test();
   cout << endl;
   vector_test();
   cout << endl;
}

//TEST// ./accumulate >accumulate.out 2>&1
//TEST// mkpspdf accumulate.ps accumulate.cpp* accumulate.out

```

```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ mkc: starting accumulate.cpp
accumulate.cpp: $Id: accumulate.cpp,v 1.4 2012-06-05 20:48:14-07 - - $
g++ -g -O0 -Wall -Wextra accumulate.cpp -o accumulate -lm
rm -f accumulate.o
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ mkc: finished accumulate.cpp
```

```
array_test:
21
1033.35

vector_test:
21
1033.35

```