```
 1 #include <iostream>
 2 #include <vector>
 3
 4 template<typename T> void p(T x) { std::cout << x; }
 5
 6 int main() {
 7     int my_data[] = {3,7,8,2,5};
 8     std::vector<int> values(&my_data[0], &my_data[sizeof(my_data)/sizeof(int)]);
 9     for(int i=0; i<values.size(); ++i)
10         p(values[i]);
11 }
```

What will this code print out? Please critizise the code. What will happen if we use std::list instead of std::vector? Please provide an alternative implementation of line 9-10.

§16.2.3, Library Organization and Containers

I get:

```
g++ -Wall scratch.cpp && ./a.out
scratch.cpp: In function 'int main()':
scratch.cpp:9: warning: comparison between signed and unsigned integer expressions
37825
```

On line 8, consider writing sizeof(my_data[0]) instead of sizeof(int)

How to rewrite this code to use short or long long instead of ints?

Replacing std::vector with std::list gives:

```
g++ -Wall scratch.cpp && ./a.out
scratch.cpp: In function 'int main()':
scratch.cpp:9: warning: comparison between signed and unsigned integer expressions
scratch.cpp:10: error: no match for 'operator[]' in 'values[i]'
```

Notice that list does not have an implementation of the [] operator. Why?

[p441] To avoid the problems of fat interfaces, operations that cannot be efficiently implemented for all containers are not included in the set of common operations. For example, subscripting is provided for std::vector but not for std::list.

How can we fix line 9-10? [p546, §18.5.1] Advice: Prefer algorithms to loops.

You might consider:

```
for_each(values.begin(), values.end(), p<int>);
```

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

struct print {
    std::string prefix_;
    print( std::string prefix ) : prefix_(prefix) {}
    void operator() (int i) {
        std::cout << prefix_ << i << std::endl;
    }
};

int main() {
    std::vector<int> v;
    v.push_back(1);
    v.push_back(2);
    v.push_back(3);
    std::for_each( v.begin(), v.end(), print("i=") );
}
```

what might happen if you try to compile, link and run this program?

§x.x, title

I get:

```
g++ -Wall scratch.cpp && ./a.out
i=1
i=2
i=3
```

```
 1 #include <iostream>
 2 #include <list>
 3 #include <algorithm>
 4
 5 template<typename T> void p(T x) { std::cout << x; }
 6
 7 template<class C> typename C::value_type sum(const C & c) {
 8     typename C::value_type s = 0;
 9     for (typename C::const_iterator i = c.begin(); i != c.end(); ++i )
10         s += *i;
11     return s;
12 }
13
14 int main() {
15     typedef long data_type;
16     data_type my_data[] = {3,7,8};
17     typedef std::list<data_type> container;
18     container values(&my_data[0], &my_data[sizeof(my_data)/sizeof(my_data[0])]);
19     for_each(values.begin(), values.end(), p<data_type>);
20     p(sum<container>(values));
21 }
```

What might happen if you try to compile, link and run this program? How many lines do we have to change if we want to work with a vector of ints instead? How can we make this code print "87318" instead? Why is the keyword typename on line 8 and 9 needed?

§16, Library Organization and Containers

I get:

```
g++ -Wall scratch.cpp && ./a.out
37818
```

To work with vector of ints instead you have to change 3 lines: 2, 15, and line 17.

If you want this code to print 87318, you might consider changing line 19:

```
for_each(values.rbegin(), values.rend(), p<data_type>);
```

Typename on line 8 and 9 is needed because C::value_type is a so called dependent-name. By default, the compiler assumes that you are refering to a non-type, but in this case it is a type and you specify that by adding typename infront of the dependent-name. (see note on p444)

See p857 for an example of what a dependent name is. Eg,

```
int y;
template<class T> void g(T & v) {
  T::x(y); // function call or variable declaration?
};
```

In this case, T::x(y) is treated as a function call, but if you add typename in front, it is treaded as a variable declaration.

```cpp
#include <iostream>
#include <queue>
#include <string>

template<typename T> void p(T x) { std::cout << x; }

class Message {
    std::string msg_;
    int priority_;
public:
    Message(std::string msg, int priority) : msg_(msg), priority_(priority) {}
    bool operator<(const Message & m) const { return priority_ < m.priority_; }
    std::string msg() const { return msg_; }
    int priority() const { return priority_; }
};

std::ostream & operator<<(std::ostream & ostm, const Message & m) {
    return ostm << m.msg() << '(' << m.priority() << ')';
}

int main() {
    std::priority_queue<Message> q;
    q.push(Message("Foo",4));
    q.push(Message("Bar",2));
    q.push(Message("Gaz",3));
    q.push(Message("Daz",5));
    q.push(Message("Boo",8));
    while ( !q.empty() ) {
        p(q.top());
        q.pop();
    };
}
```

what might happen if you try to compile, link and run this program?

§17.3.3, Priority Queue

I get:

```
g++ -Wall scratch.cpp && ./a.out
Boo(8)Daz(5)Foo(4)Gaz(3)Bar(2)
```

```
#include <iostream>
#include <string>
#include <map>

int main() {
    std::map<std::string,int> m;
    m["foo"] = 2;
    m["foo"] = 4;
    m["bar"] = 8;
    m["gaz"] = 5;
    m["gaz"]++;
    for( std::map<std::string,int>::iterator i = m.begin();
         i != m.end(); ++i) {
        std::cout << (*i).first << " " << (*i).second << std::endl;
    }

    std::pair<std::string,int> b("bar",1);
    std::pair<std::map<std::string,int>::iterator,bool> p = m.insert(b);
    std::cout << std::boolalpha << p.second << std::endl;
    std::cout << m["bar"] << std::endl;
}
```

what might happen if you try to compile, link and run this program?

§x.x, title

I get

```
g++ -Wall scratch.cpp && ./a.out
bar 8
foo 4
gaz 6
false
8
```

```cpp
#include <iostream>
#include <bitset>
#include <string>

int main() {
    std::bitset<8> b = 0x01;

    b << 1;
    std::cout << b << std::endl;

    b |= std::bitset<8>(std::string("11110000"));
    std::cout << b << std::endl;

    std::cout << b.count() << std::endl;
}
```

what might happen if you try to compile, link and run this program?

§x.x, title

I get:

```
g++ -Wall scratch.cpp && ./a.out
00000001
11110001
5
```

```cpp
#include <iostream>
#include <algorithm>
#include <vector>
#include <list>
#include <iterator>

int main() {
    std::vector<int> v1;
    v1.push_back(1);
    v1.push_back(4);
    v1.push_back(3);
    v1.push_back(4);
    v1.push_back(9);
    std::list<int> v2;
    std::copy( v1.begin(), v1.end(), front_inserter(v2));
    std::copy( v2.rbegin(), v2.rend(), std::ostream_iterator<int>(std::cout) );
}
```

what might happen if you try to compile, link and run this program?

§x.x, title

I get:

```
g++ -Wall scratch.cpp && ./a.out
14349
```

```
#include <iostream>
#include <algorithm>

template<typename T> void p(T x) { std::cout << x; }

class my_array {
public:
    typedef short value_type;
    typedef value_type * iterator;
    typedef value_type & reference;
    reference operator [] (ptrdiff_t i) { return v[i]; }
    iterator begin() { return v; }
    iterator end() { return v+size(); }
    size_t size() const { return sizeof(v) / sizeof(value_type); }
    my_array() : v() {}
private:
    value_type v[4];
};

int main() {
    my_array m;
    m[2] = 4;
    m[3] = 2;
    std::for_each( m.begin() + 2, m.end(), p<my_array::value_type> );
}
```

what might happen if you try to compile, link and run this program?

§x.x, title

I get:

```
g++ -Wall scratch.cpp && ./a.out
42
```

```cpp
#include <iostream>
#include <algorithm>
#include <vector>
#include <functional>

class Foo {
    int v;
public:
    Foo(int i) : v(i) {}
    void print() { std::cout << v; }
};

int main() {
    std::vector<Foo> v;
    v.push_back(Foo(1));
    v.push_back(Foo(2));
    v.push_back(Foo(3));
    std::for_each( v.begin(), v.end(), Foo::print );
}
```

This code does not compile. How to fix it so that it prints 123?

§x.x, title

I get:

```
g++ -Wall scratch.cpp && ./a.out
scratch.cpp: In function 'int main()':
scratch.cpp:18: error: invalid use of non-static member function 'void Foo::print()'
scratch.cpp:18: error: invalid use of non-static member function
/usr/include/c++/4.0.0/bits/stl_algo.h: In function '_Function std::for_each(_InputIterator, \
  _InputIterator, _Function) [with _InputIterator = __gnu_cxx::__normal_iterator<Foo*, \
  std::vector<Foo, std::allocator<Foo> > >, _Function = void (Foo::)()]':
scratch.cpp:18:   instantiated from here
/usr/include/c++/4.0.0/bits/stl_algo.h:158: error: cannot convert 'Foo' to 'Foo*' in \
  argument passing
/usr/include/c++/4.0.0/bits/stl_algo.h:159: error: invalid use of non-static member function
```

How to fix this? Eg, use an adapter:

```
std::for_each( v.begin(), v.end(), std::mem_fun_ref(&Foo::print) );
```

```cpp
#include <iostream>
#include <algorithm>
#include <vector>

bool equal_to_4_func(int i) {
    return i == 4;
}

struct equal_to_4_class {
    bool operator()(int i) { return i == 4; }
};

int main() {
    std::vector<int> v;
    v.push_back(1);
    v.push_back(4);
    v.push_back(3);
    v.push_back(4);
    v.push_back(9);
    std::cout << std::count_if( v.begin(), v.end(), equal_to_4_func );
    std::cout << std::count_if( v.begin(), v.end(), equal_to_4_class() );
}
```

what might happen if you try to compile, link and run this program?

§x.x, title

I get:

```
g++ -Wall scratch.cpp && ./a.out
22
```

```cpp
#include <iostream>
#include <algorithm>
#include <vector>
#include <functional>

template <int v> bool equal_to_func_templ(int i) {
    return i == v;
}

template <int v> struct equal_to_class_templ {
    bool operator()(int i) const { return i == v; }
};

struct my_equal_to : public std::binary_function<int, int, bool> {
    bool operator() (int a, int b) const { return a == b; }
};

int main() {
    std::vector<int> v;
    v.push_back(1);
    v.push_back(4);
    v.push_back(3);
    v.push_back(4);
    v.push_back(9);
    using namespace std;
    cout << count_if( v.begin(), v.end(), equal_to_func_templ<4> ) << endl;
    cout << count_if( v.begin(), v.end(), equal_to_class_templ<4>() ) << endl;
    cout << count_if( v.begin(), v.end(), bind2nd(equal_to<int>(),4) ) << endl;
    cout << count_if( v.begin(), v.end(), bind1st(my_equal_to(),4) ) << endl;
}
```

what might happen if you try to compile, link and run this program?

§x.x, title

I get:

```
g++ -Wall scratch.cpp && ./a.out
2222
```