תכנות מתקדם
מצגת 5

קלט ופלט

istringstream

```
struct PersonInfo {
     string name;
     vector<string> phones;
};
vector<PersonInfo> getData(istream &is) {
     string line, word;
     vector<PersonInfo> people;
     while (getline(is, line)) {
          istringstream record(line);
          PersonInfo info;
          record >> info.name;
          while (record >> word)
               info.phones.push_back(word);
          people.push_back(info);
     }
return people;
}
```

input file

```
morgan 2015552368 8625550123
drew 9735550130
lee 6095550132 2015550175 8005550000
```

ostringstream

```
ostream& process(ostream &os, vector<PersonInfo> people) {
for (const auto &entry : people) {
     ostringstream formatted, badNums;
     for (const auto &nums : entry.phones) {
          if (!valid(nums)) {
               badNums << " " << nums;
          } else
               formatted << " " << format(nums);
     }
     if (badNums.str().empty())
          os << entry.name << " " << formatted.str() << endl;
     else
          cerr << "input error: " << entry.name
           << " invalid number(s) " << badNums.str() << endl;
     }
return os;
}
```

istream_iterator

`istream_iterator<T> in(is);`	in reads values of type T from input stream is.
`istream_iterator<T> end;`	Off-the-end iterator for an istream_iterator that reads values of type T.

```
istream_iterator<int> in_iter(cin); // read ints from cin
istream_iterator<int> eof;          // end iterator
while (in_iter != eof)              // while there's input
// postfix increment returns the old value of the iterator
// we dereference that iterator to get the previous value
    vec.push_back(*in_iter++);
```

we can rewrite this program as:

```
istream_iterator<int> in_iter(cin), eof;
// construct vec from an iterator range
vector<int> vec(in_iter, eof);
```

Using istream_iterator with the Algorithms

```
// generate the sum of values read from the input
istream_iterator<int> in(cin), eof;
cout << accumulate(in, eof, 0) << endl;
```

ostream_iterator

`ostream_iterator<T> out(os);`	`out` writes values of type `T` to output stream `os`.
`ostream_iterator<T> out(os, d);`	`out` writes values of type `T` followed by `d` to output stream `os`. `d` points to a null-terminated character array.
`out = val`	Writes `val` to the `ostream` to which `out` is bound using the `<<` operator. `val` must have a type that is compatible with the type that `out` can write.
`*out, ++out,`	These operations exist but do nothing to `out`. Each operator returns `out`.

We can use an ostream_iterator to write a sequence of values.

We may provide a string to print following each element.

```
ostream_iterator<int> out_iter(cout, " ");
for (auto e : vec)
     *out_iter++ = e; // out_iter = e; // quivalent
```

Or:

```
copy(vec.begin(), vec.end(), out_iter);
```

ostream_iterator

```
// Copy the contents of the container to std::cout,
// separating elements with a single space

#include <iostream>
#include <vector>
#include <iterator>
int main() {
std::vector<int> vec { 10, 20, 30, 35, 40, 45, 50, 55 };
auto strm = std::ostream_iterator<int>(std::cout, " ");
std::copy(std::begin(vec), std::end(vec), strm);
std::cout << '\n';
}
```

ostream_iterator

```
// Copy the contents of the container to a text file
#include <fstream>
#include <vector>
#include <iterator>
int main() {
std::vector<int> vec { 10, 20, 30, 35, 40, 45, 50, 55 };
std::ofstream fout("output.txt");
if (fout.good()) {
     auto strm = std::ostream_iterator<int>(fout, " ");
     std::copy(std::begin(vec), std::end(vec), strm);
     fout << '\n';
}
}
```

ostream_iterator

```
int main() {
std::vector<int> seq { 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 };
auto output = ostream_iterator<int>(cout, " ");
std::copy(std::begin(seq), std::end(seq), output);
std::cout << '\n';
auto is_even = [](int n) { return n % 2 == 0; };
int even_count = count_if(begin(seq), end(seq), is_even);
// Make a copy of vec omitting all the odd numbers
std::vector<int> seq2(even_count);
std::copy_if(begin(seq), end(seq), begin(seq2), is_even);
copy(std::begin(seq2), std::end(seq2), output);
std::cout << '\n';
```