P.3 Express intent

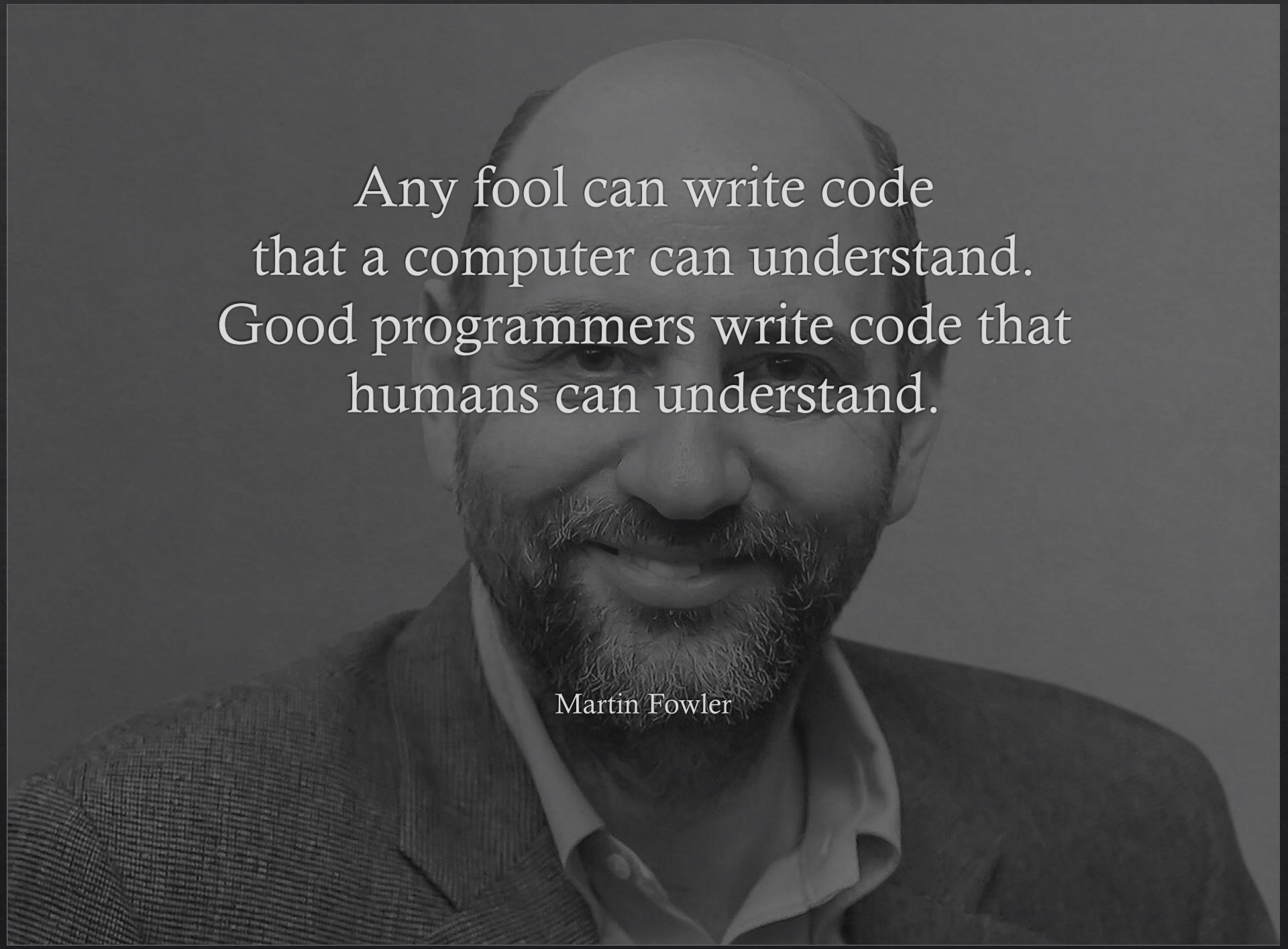
Ankur M. Satle

https://ankursatle.wordpress.com

https://www.linkedin.com/in/ankursatle

The ratio of time spent reading versus writing code is well over 10 to 1

- Robert C. Martin


“
Any fool can write code
that a computer can understand.
Good programmers write code that
humans can understand.
”
Martin Fowler

“ There are two ways of constructing a software design. One way is to make it so simple that there are obviously no deficiencies. And the other way is to make it so complicated that there are no obvious deficiencies. ”

C. A. R. Hoare,

The 1980 ACM Turing Award Lecture.

“

Easy to use correctly and difficult to use incorrectly

”

So desirable!

“The purpose of a GUI
is to make the functionality of the application
apparent to the user
and enable good use of it”

Express Intent!

“

अपने विवेक का इस्तेमाल करें

Please exercise your judgement

”

C++ Core Guidelines

August 19, 2021

Editors:

- Bjarne Stroustrup
- Herb Sutter

This is a living document under continuous improvement. Had it been an c
(code) project, this would have been release 0.8. Copying, use, modificatior
of derivative works from this project is licensed under an MIT-style license
to this project requires agreeing to a Contributor License. See the accompa
LICENSE file for details. We make this project available to "friendly users" t
modify, and derive from, hoping for constructive input.

Comments and suggestions for improvements are most welcome. We plan
extend this document as our understanding improves and the language an
available libraries improve. When commenting, please note the introductio
outlines our aims and general approach. The list of contributors is here.

Problems:

- The sets of rules have not been completely checked for completeness,
  or enforceability.
- Triple question marks (???) mark known missing information
- Update reference sections; many pre-C++11 sources are too old.
- For a more-or-less up-to-date to-do list see: To-do: Unclassified proto-

You can read an explanation of the scope and structure of this Guide or jus
straight in:

P.3: Express intent

Reason Unless the intent of some code is stated (e.g., in names or comments), it is impossible to tell whether the code does what it is supposed to do.

Example

```
gsl::index i = 0;
while (i < v.size()) {
    // ... do something with v[i] ...
}
```

The intent of "just" looping over the elements of `v` is not expressed here. The implementation detail of an index is exposed (so that it might be misused), and `i` outlives the scope of the loop, which might or might not be intended. The reader cannot know from just this section of code.

Better:

```
for (const auto& x : v) { /* do something with the value of x */ }
```

Now, there is no explicit mention of the iteration mechanism, and the loop operates on a reference to `const` elements so that accidental modification cannot happen. If modification is desired, say so:

```
for (auto& x : v) { /* modify x */ }
```

For more details about for-statements, see ES.71. Sometimes better still, use a named algorithm. This example uses the `for_each` from the Ranges TS because it directly expresses the intent:

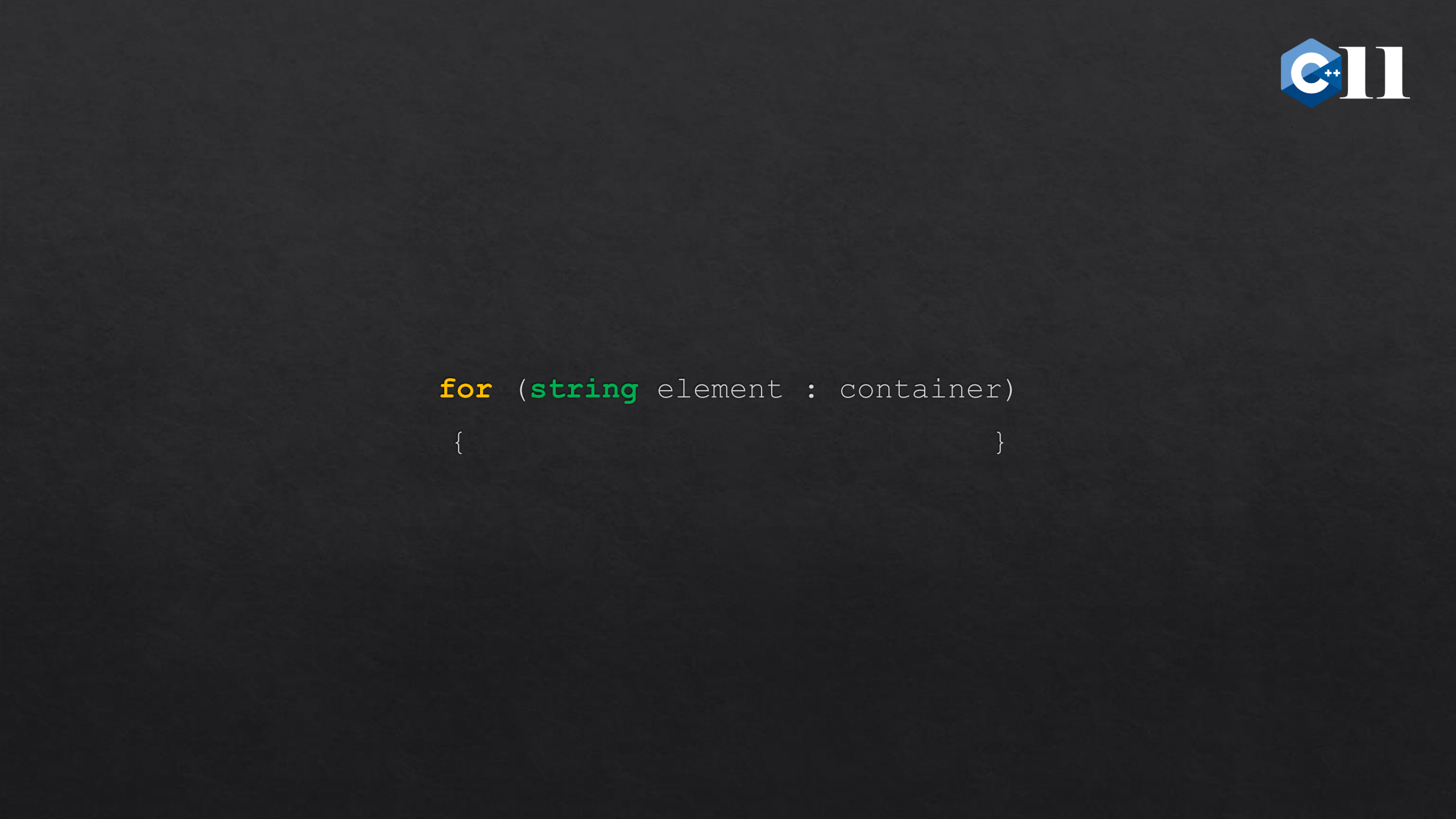
```
for_each(v, [](int x) { /* do something with the value of x */ });
for_each(par, v, [](int x) { /* do something with the value of x */ });
```

The last variant makes it clear that we are not interested in the order in which the

```
int i = 0;
 while (i < container.size())
{                                  }
```

```
for (int i = 0; i < container.size(); ++i)
    {                                    }
```

```
int i = 0;
for (; i < container.size(); ++i)
 {                                  }
```


for (string element : container)
{ }


```
for (string& element : container)
  {                                }
```

```
for (const string& element : container)
    {                                  }
```


```
for (auto& element : container)
{                                  }
```

```
for (const auto& element : container)
 {                                    }
```

```
for (const auto& element : container)
{   if (element.is_empty()) break;  }
```


```
for (const auto& element : container)
{   if (!element.is_empty()) continue; }
```

```
for (auto it = begin(c);
  it != end(c) && !it->empty();
 ++it)
   {                          }
```


```
auto it = begin(c);
for (; it != end(c) && !it->empty(); ++it)
{                       ...                 }
if (it != end(c))
   use(*it);
```

```
auto it = std::find_if(begin(c), end(c),
  [](const auto& elem) { return elem.empty(); });
if (it != end(c))
   use(*it);
```

```
auto is_empty = [](const auto& elem) { return elem.empty(); });
auto it = std::find_if(begin(c), end(c), is_empty);
if (it != end(c))
    use(*it);
```

```
auto it = std::find_if(begin(c), end(c), std::mem_fn(&Class::empty));
if (it != end(c))
    use(*it);
```

```
auto it = std::for_each(begin(c), end(c), [](auto& elem) {
            if (!elem.empty()) {
               std::cout << to_string(use(elem));
            }
   });
```

```
std::vector<T> inter;
auto is_not_empty = [](const auto& elem) { return !elem.empty(); });
auto it = std::copy_if(begin(c), end(c), back_inserter(inter),
                                         is_not_empty);
for (auto& elem : inter) {
   std::cout << to_string(use(elem));
}
```


```
using namespace std::views;
auto is_not_empty = [](const auto& elem) { return !elem.empty(); });
auto pipeline = c | filter(is_not_empty) | transform(use) |
      transform(to_string);
for (auto str : pipeline) {
   std::cout << str;
}
```

```
int max_transaction;
```


```
int max_transaction = 50’000;
```

```
int max_transactions = 4096;
```

```
int max_transactions = 4096;
Transaction all_trans[max_transactions];
```

```
const int max_transactions = 4096;
Transaction all_trans[max_transactions];
```

```
const int max_transactions = config.no_trans_max;
Transaction all_trans[max_transactions];
```

```
Transaction all_trans[config.no_trans_max];
```

```
std::vector<Transaction> all_trans(config.no_trans_max);
```

```
std::vector<Transaction> all_trans;
                    ...
all_trans.reserve(config.no_trans_max);
```

```
std::vector<Transaction> all_trans =
    make_trans_store(config.no_trans_max);
```


```
const int max_transactions = 4096;
std::array<Transaction, max_transactions> all_trans;
```


```
void revert_transactions(std::span<Transaction> trans);
```


```
void revert_transactions(std::span<const Transaction> trans);
```

```
void revert_transactions(std::vector<Transaction>&& trans);
```

समाप्त

धन्यवाद