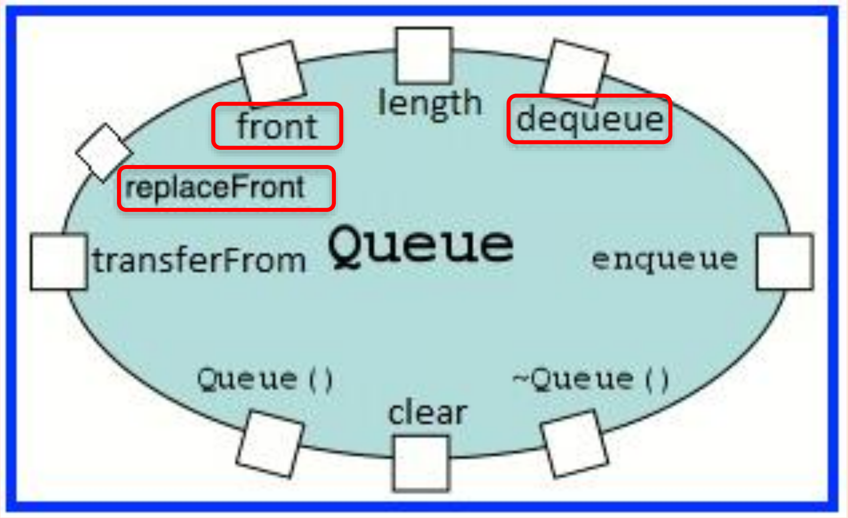
Review of Checking Component

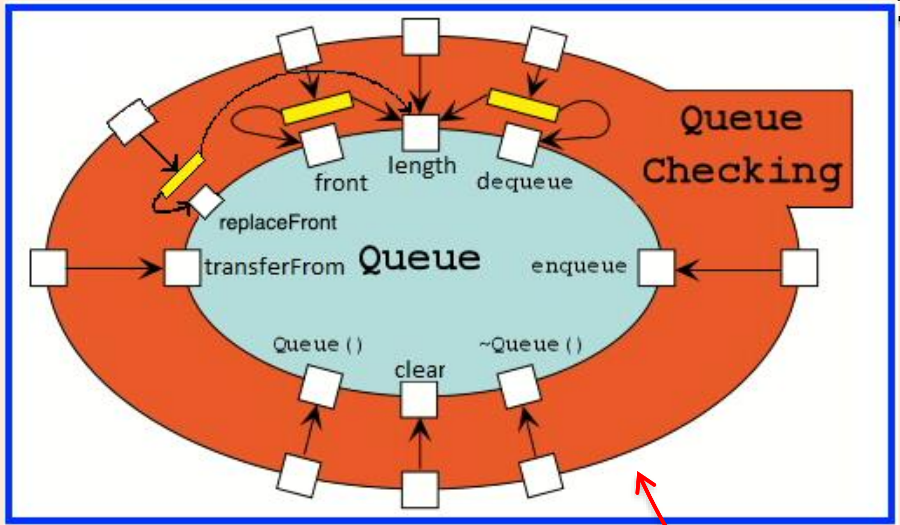


A Design-by-Contract Component

With Design-by-Contract (DbC):

- Each component operation is written so that it does not check its own precondition (i.e., requires clause)
- That is, each operation assumes that its precondition holds at the time of the call
- For example:
  - Queue's *dequeue*, *front*, and *replaceFront* each have a non-trivial requires clause
  - Each require that the controlling queue variable be non-empty
  - Each is implemented following the DbC principle of assuming that the controlling queue is non-empty at the time of the call

Testing & Debugging and Design-by-Contract

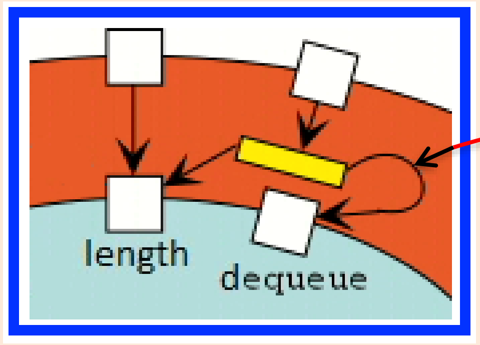


- During testing & debugging, we want to detect as early as possible when a client program calls an operation but fails to satisfied the operation's contract, i.e., its precondition

- Instead of having checking code embedded in our components, we instead we can leverage DbC and have it *wrapped* around our components

- The yellow filled rectangles 🟨 in the diagram represent code that checks the precondition of the respective operation by calling the *length* operation

Testing & Debugging and Design-by-Contract



- The yellow filled rectangles 🟨 in the diagram represent code that checks the precondition of the respective operation by calling the *length* operation

- If the requires clause has been met by the calling operation, then the call passes on through to the internal *unchecked* component

Testing & Debugging – Using C++'s Conditional Compilation

When building a VisualStudio project in Debug configuration

- `_Debug` preprocessor symbol will be defined automatically by VisualStudio
- So, this code will be compiled
- And the unchecked Queue will be wrapped by QueueChecking

```
#prgama once
// Filename QueueOfText.h
#include "wrapper.h"

#ifdef _DEBUG
// Compiling in Debug configuration
#include "Queue/Queue1.hpp"
#include "Queue/QueueChecking.hpp"
typedef QueueChecking1 <Text, Queue1<Text>> TextQueue;

#else

// When _DEBUG is not defined, that means:
//   Compiling in Release Configuration = Not Debug configuration
#include "Queue/Queue1.hpp"
typedef Queue1<Text> TextQueue;

#endif
```

Testing & Debugging – Using C++'s Conditional Compilation

When building a VisualStudio project in Release configuration

- _Debug preprocessor symbol **will not** be defined automatically by VisualStudio
- So, this code will be compiled
- The client program will be using the unchecked version of Queue

```
#prgama once
// Filename QueueOfText.h
#include "wrapper.h"

#ifdef _DEBUG
// Compiling in Debug configuration
#include "Queue/Queue1.hpp"
#include "Queue/QueueChecking.hpp"
typedef QueueChecking1 <Text, Queue1<Text>> TextQueue;

#else
// When _DEBUG is not defined, that means:
//   Compiling in Release Configuration = Not Debug configuration
#include "Queue/Queue1.hpp"
typedef Queue1<Text> TextQueue;

#endif
```