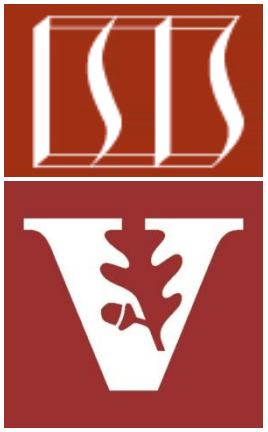
Evolution of Programming Abstraction Mechanisms: Object-Oriented Programming

Douglas C. Schmidt

d.schmidt@vanderbilt.edu

www.dre.vanderbilt.edu/~schmidt



Professor of Computer Science

Institute for Software Integrated Systems

Vanderbilt University
Nashville, Tennessee, USA

C++ Object-Oriented Stack Implementation

Object-Oriented Implementation in C++

- Problems with previous examples:
  - Changes to the implementation require recompilation & relinking of clients
  - Extensions require access to the source code


```
template<typename T>
class stack {
public:
  stack (size_t size);
  stack (const stack<T> &s);
  stack<T> &operator=(const
                      stack<T> &);
  ~stack (void);
  void push (const T &item);
  void pop (void);
  const T &top () const;
  T &top ();
  bool is_empty (void) const;
  bool is_full (void) const;
private:
  size_t top_, size_; T *stack_;
};
```

Change to a linked list implementation

Object-Oriented Implementation in C++

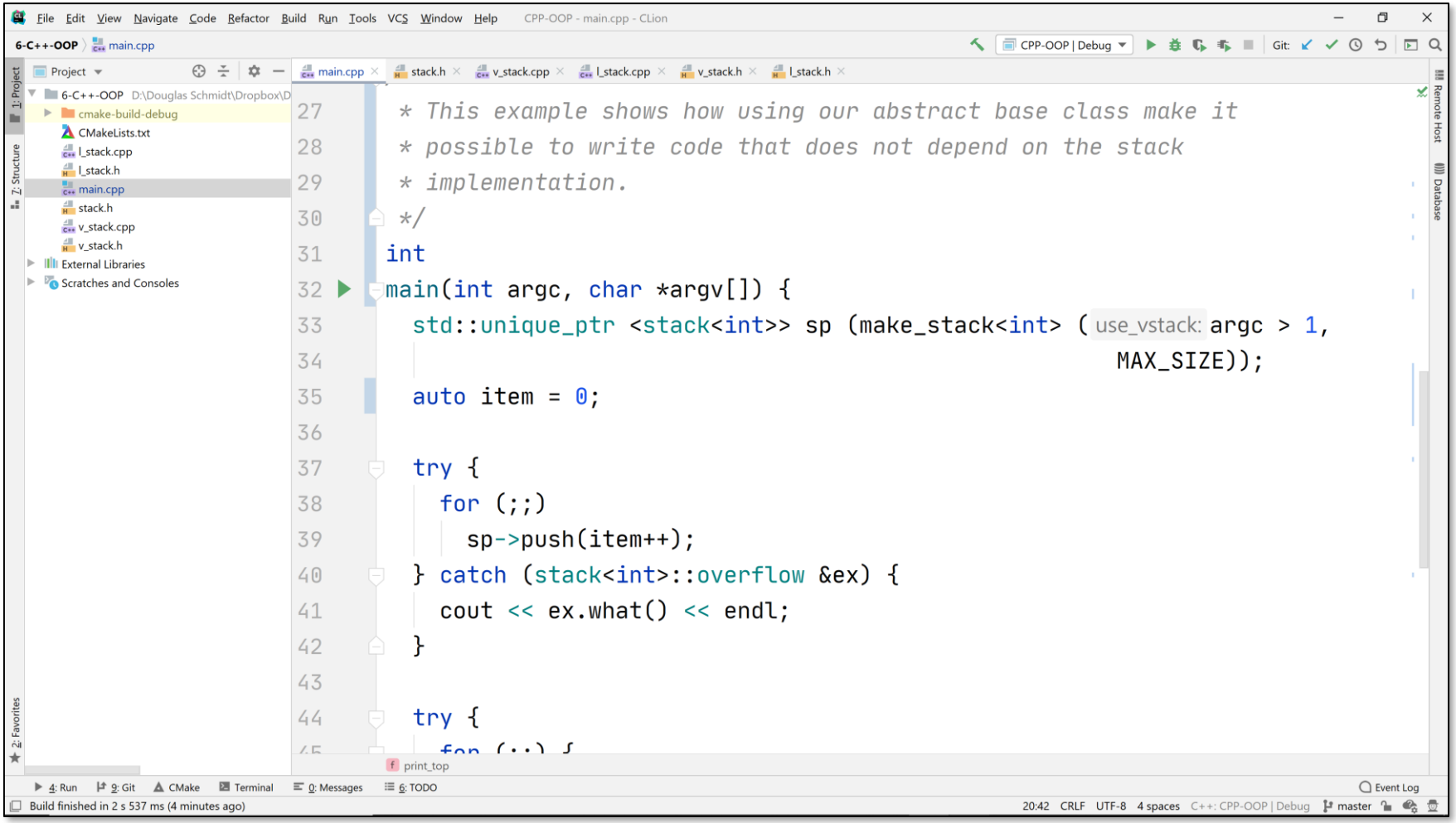
- Problems with previous examples:
  - Changes to the implementation require recompilation & relinking of clients
  - Extensions require access to the source code
- Solutions
  - Combine inheritance with dynamic binding to completely decouple interface from implementation & binding time
    - This requires the use of C++ abstract base classes

```
template<typename T>
class stack {
public:
  virtual ~stack (void);
  virtual void push (const T
                     &item) = 0;
  virtual void pop (void) = 0;
  virtual T &top (T &item) = 0;
  virtual const T &top (T &item)
                          const = 0;
  virtual bool is_empty (void)
                        const = 0;
  virtual bool is_full (void)
                        const = 0;
};
```

See isocpp.org/wiki/faq/abcs

Object-Oriented Implementation in C++

- C++ object-oriented programming enables runtime binding of method calls



See CPlusPlus/tree/master/overview/capabilities/6-C++-OOP

Pros of C++ Object-Oriented Implementation

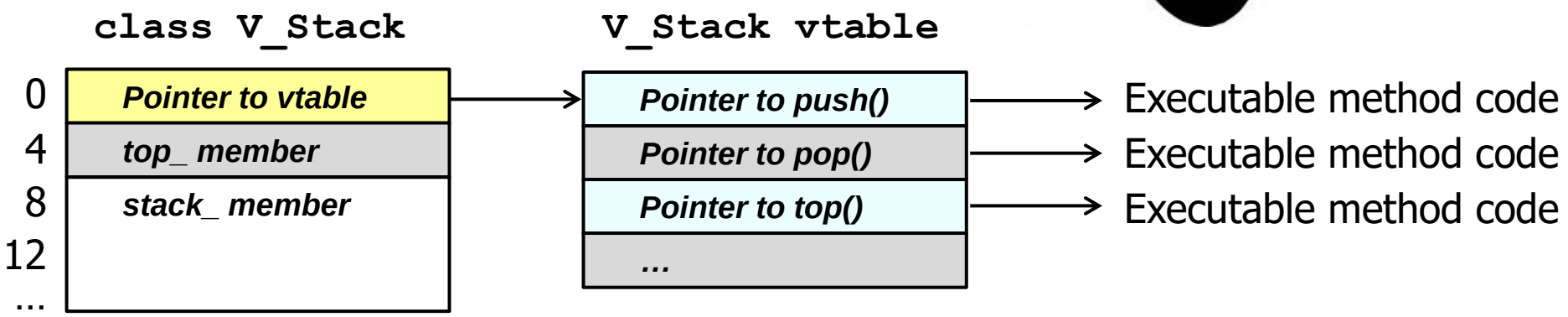
- **Pros**
  - Can reuse code without knowing the specifics of future subclasses!
  - Can also write code that doesn't expose implementation details (including the size of an object) at all to clients

Cons of C++ Object-Oriented Implementation

- **Cons**
  - Some cool C++ optimizations don't work with virtual methods
    - Member templates with variadic arguments, such as emplace()

Cons of C++ Object-Oriented Implementation

- **Cons**
  - Some cool C++ optimizations don't work with virtual methods
  - Each class with virtual methods has a so-called "vtable" may (slightly) increase method call overhead



en.wikipedia.org/wiki/Dynamic_dispatch has more info

C++ End of Object-Oriented Stack Implementation

Summary

- A major contribution of C++ is its support for abstract data types (ADTs) & generic programming
  - e.g., classes, parameterized types, & exception handling

```
template<typename T>
class stack {
public:
  stack (size_t size);
  stack (const stack<T> &s);
  stack<T> &operator=(const
                      stack<T> &);
  ~stack (void);
  void push (const T &item);
  void pop (void);
  void top (T &item) const;
  bool is_empty (void) const;
  bool is_full (void) const;
private:
  size_t top_, size_; T *stack_;
};
```

Summary

- A major contribution of C++ is its support for abstract data types (ADTs) & generic programming
- For some types of programs, C++'s OO features are essential to build highly flexible & extensible software
  - e.g., inheritance, dynamic binding, & RTTI

```
template<typename T>
class stack {
public:
  virtual ~stack (void);
  virtual void push (const T
                     &item) = 0;
  virtual void pop (void) = 0;
  virtual void top (T &item)
                     const = 0;
  virtual bool is_empty (void)
                     const = 0;
  virtual bool is_full (void)
                     const = 0;
};
```

Summary

- A major contribution of C++ is its support for abstract data types (ADTs) & generic programming
- For some types of programs, C++'s OO features are essential to build highly flexible & extensible software
- For other types of programs, C++'s ADT & generic programming support is more important than using its OO features
  - Modern C++ emphasizes generic programming more than OOP