

# **Evolution of Programming Abstraction Mechanisms: C++ Data Abstraction**

**Douglas C. Schmidt**

[d.schmidt@vanderbilt.edu](mailto:d.schmidt@vanderbilt.edu)

[www.dre.vanderbilt.edu/~schmidt](http://www.dre.vanderbilt.edu/~schmidt)



**Professor of Computer Science**

**Institute for Software  
Integrated Systems**

**Vanderbilt University  
Nashville, Tennessee, USA**

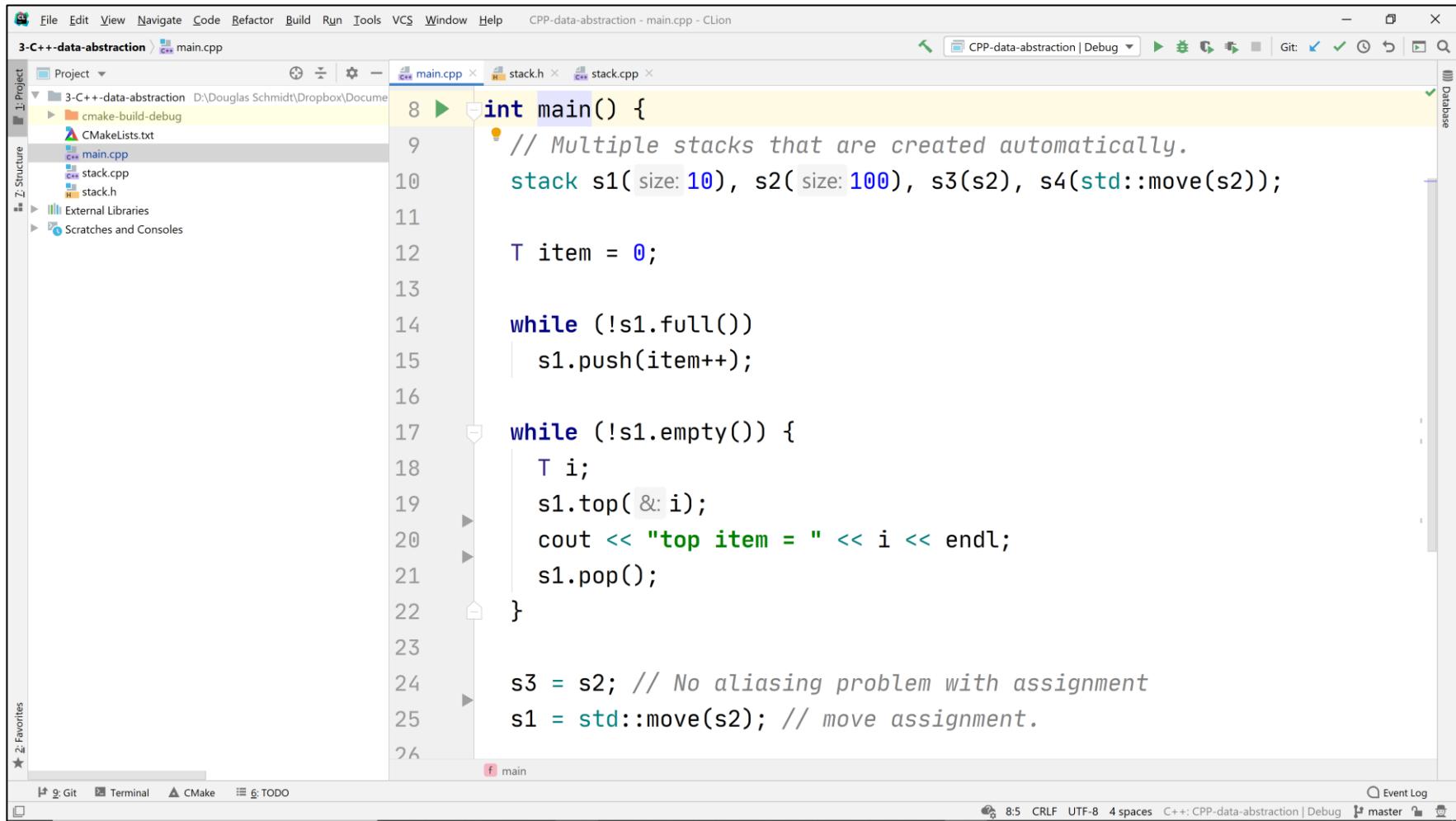


---

# C++ Data Abstraction Stack Implementations

# Data Abstraction Implementation in C++

- Use a C++ class to achieve encapsulation & create more than one stack



The screenshot shows the CLion IDE interface with the following details:

- File Menu:** File, Edit, View, Navigate, Code, Refactor, Build, Run, Tools, VCS, Window, Help.
- Project:** CPP-data-abstraction - main.cpp - CLion
- Toolbars:** Standard toolbar with icons for file operations.
- Code Editor:** Displays the `main.cpp` file content. The code implements multiple stacks using a class template named `stack`. It includes a constructor for creating multiple stacks of different sizes, and methods for pushing and popping items from each stack. The code also demonstrates move assignment and move construction.

```
int main() {
    // Multiple stacks that are created automatically.
    stack s1( size: 10 ), s2( size: 100 ), s3(s2), s4(std::move(s2));

    T item = 0;

    while ( !s1.full() )
        s1.push(item++);

    while ( !s1.empty() ) {
        T i;
        s1.top( & i );
        cout << "top item = " << i << endl;
        s1.pop();
    }

    s3 = s2; // No aliasing problem with assignment
    s1 = std::move(s2); // move assignment.
}
```

- Toolbars:** Git, Terminal, CMake, TODO.
- Status Bar:** 8:5 CRLF UTF-8 4 spaces C++:CPP-data-abstraction | Debug master

See [CPlusPlus/tree/master/overview/capabilities/3-C++-data-abstraction](https://CPlusPlus/tree/master/overview/capabilities/3-C++-data-abstraction)

# Pros of Data Abstraction in C++

- Information Hiding & data abstraction, e.g.,

```
stack s1 (200);  
s1.top_ = 10 // Error flagged by compiler!
```

- The ability to declare multiple stack objects

```
stack s1 (10), s2 (20), s3 (30);
```

- Automatic initialization & termination

```
{  
    stack s1 (1000); // constructor called automatically.  
    // ...  
    // Destructor called automatically  
}
```



# Cons of Data Abstraction in C++

- Error handling is obtrusive
  - Use exception handling to solve this (but be careful)!
- The example is limited to a single type of stack element (int in this case)
  - We can use C++ “parameterized types” to remove this limitation
- Function call overhead
  - We can use C++ inline functions to remove this overhead



---

# End of C++ Data Abstraction Stack Implementations