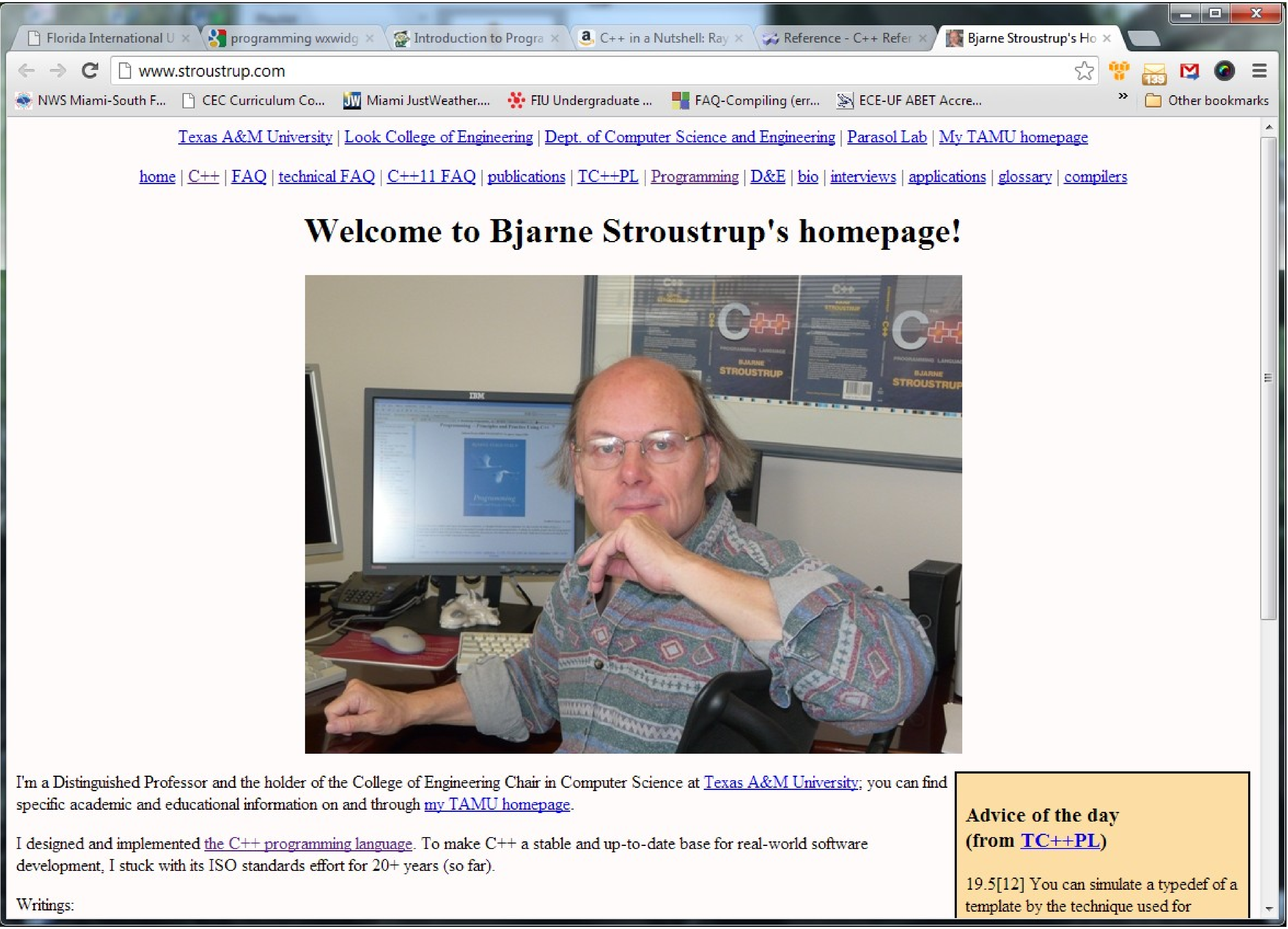
Introduction to C++ and Data Structures



Texas A&M University | Look College of Engineering | Dept. of Computer Science and Engineering | Parasol Lab | My TAMU homepage

home | C++ | FAQ | technical FAQ | C++11 FAQ | publications | TC++PL | Programming | D&E | bio | interviews | applications | glossary | compilers

Welcome to Bjarne Stroustrup's homepage!

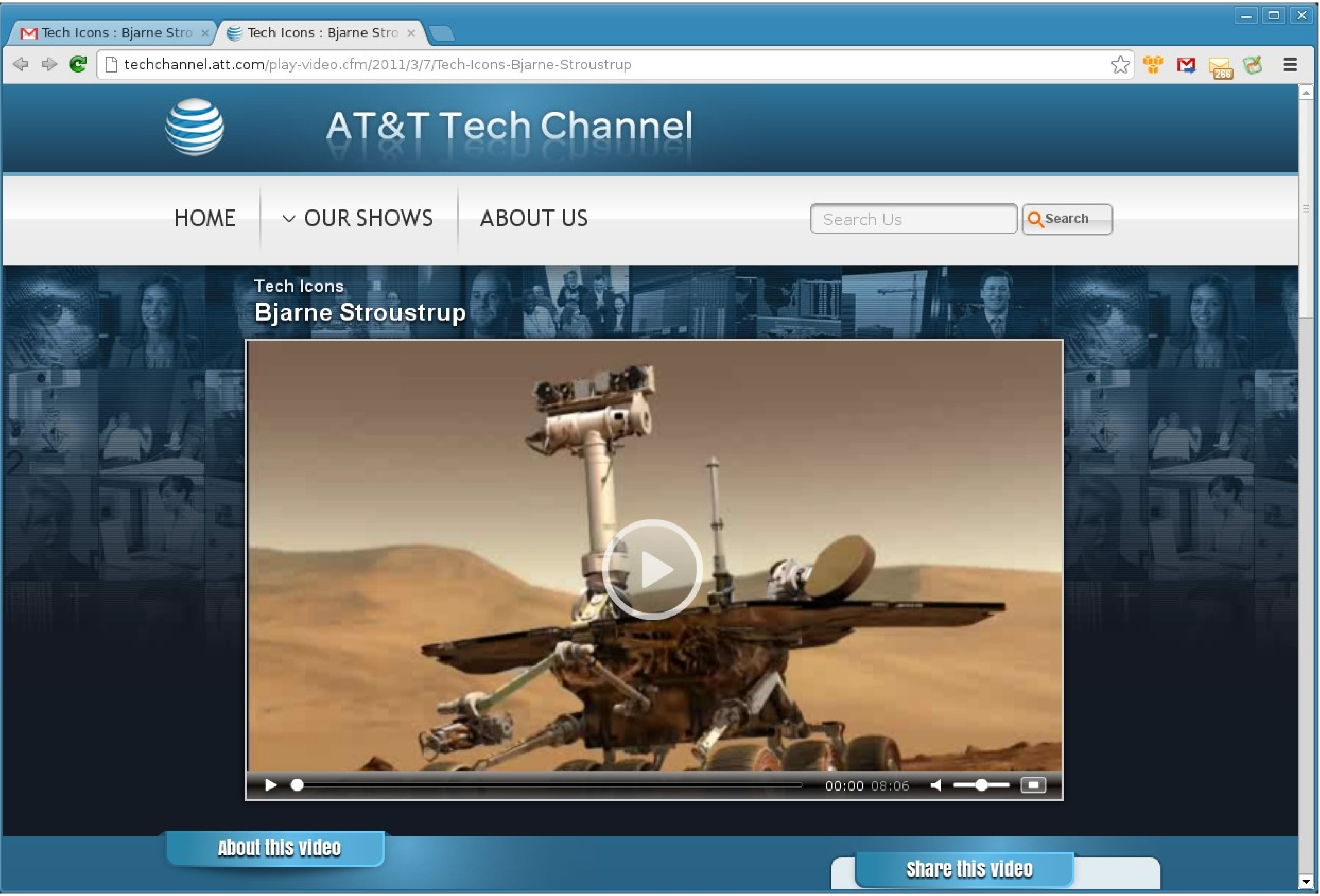
I'm a Distinguished Professor and the holder of the College of Engineering Chair in Computer Science at Texas A&M University; you can find specific academic and educational information on and through my TAMU homepage.

I designed and implemented the C++ programming language. To make C++ a stable and up-to-date base for real-world software development, I stuck with its ISO standards effort for 20+ years (so far).

Writings:

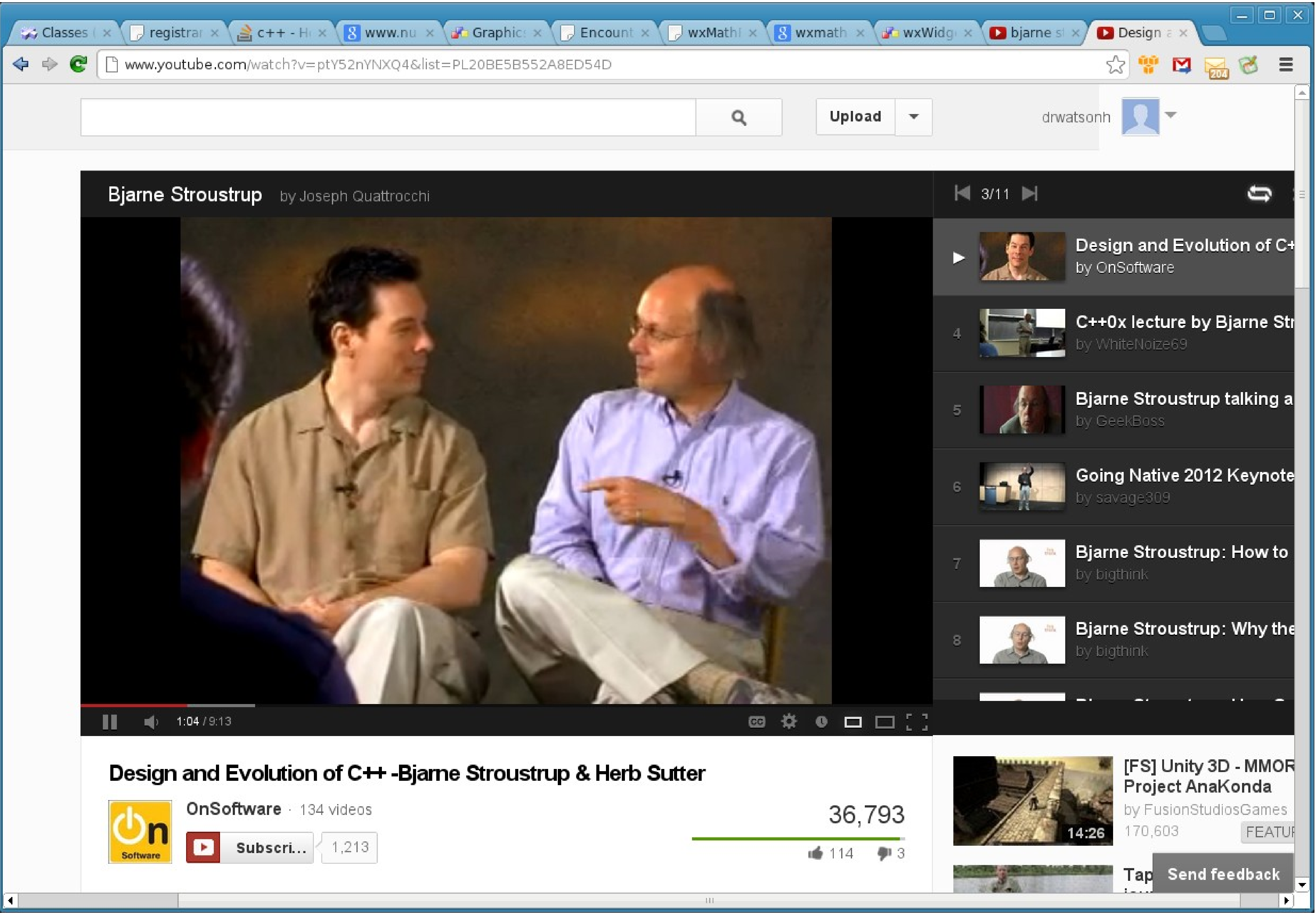
Advice of the day
(from TC++PL)

19.5[12] You can simulate a typedef of a template by the technique used for



Tech Icons : Bjarne Stro
Tech Icons : Bjarne Stro
techchannel.att.com/play-video.cfm/2011/3/7/Tech-Icons-Bjarne-Stroustrup
AT&T Tech Channel
HOME
OUR SHOWS
ABOUT US
Search Us
Search
Tech Icons
Bjarne Stroustrup
00:00 08:06
About this video
Share this video

Classes
registrar
c++ - H
www.nu
Graphic
Encount
wxMath
wxmath
wxWidg
bjarne s
C++ Inv
www.youtube.com/watch?v=fOJ2CntgRNI&list=PL20BE5B552A8ED54D
Upload
drwatsonh
Bjarne Stroustrup
by Joseph Quattrocchi
1/11
C++ Inventor Bjarn
by ieeeComputerSocie
2
Bjarne Stroustrup.
by milkushka
3
Design and Evoluti
by OnSoftware
4
C++0x lecture by B
by WhiteNoize69
5
Bjarne Stroustrup t
by GeekBoss
6
Going Native 2012
by savage309
04:23 / 15:24
C++ Inventor Bjarne Stroustrup
ieeeComputerSociety · 353 videos
Subscri...
1,211
8,908
45
1
One Trick to
by PimsleurApp
1,718,565 views
14:41
Send feedback



Classes | registrar | c++ - H | www.nu | Graphic | Encount | wxMath | wxmath | wxWidg | bjarne s | Design

www.youtube.com/watch?v=ptY52nYNXQ4&list=PL20BE5B552A8ED54D

Upload

drwatsonh

Bjarne Stroustrup by Joseph Quattrocchi

3/11

Design and Evolution of C+
by OnSoftware

4 C++0x lecture by Bjarne Str
by WhiteNoize69

5 Bjarne Stroustrup talking a
by GeekBoss

6 Going Native 2012 Keynote
by savage309

7 Bjarne Stroustrup: How to
by bigthink

8 Bjarne Stroustrup: Why the
by bigthink

1:04 / 9:13

Design and Evolution of C++ -Bjarne Stroustrup & Herb Sutter

OnSoftware · 134 videos

Subscri... 1,213

36,793

114 3

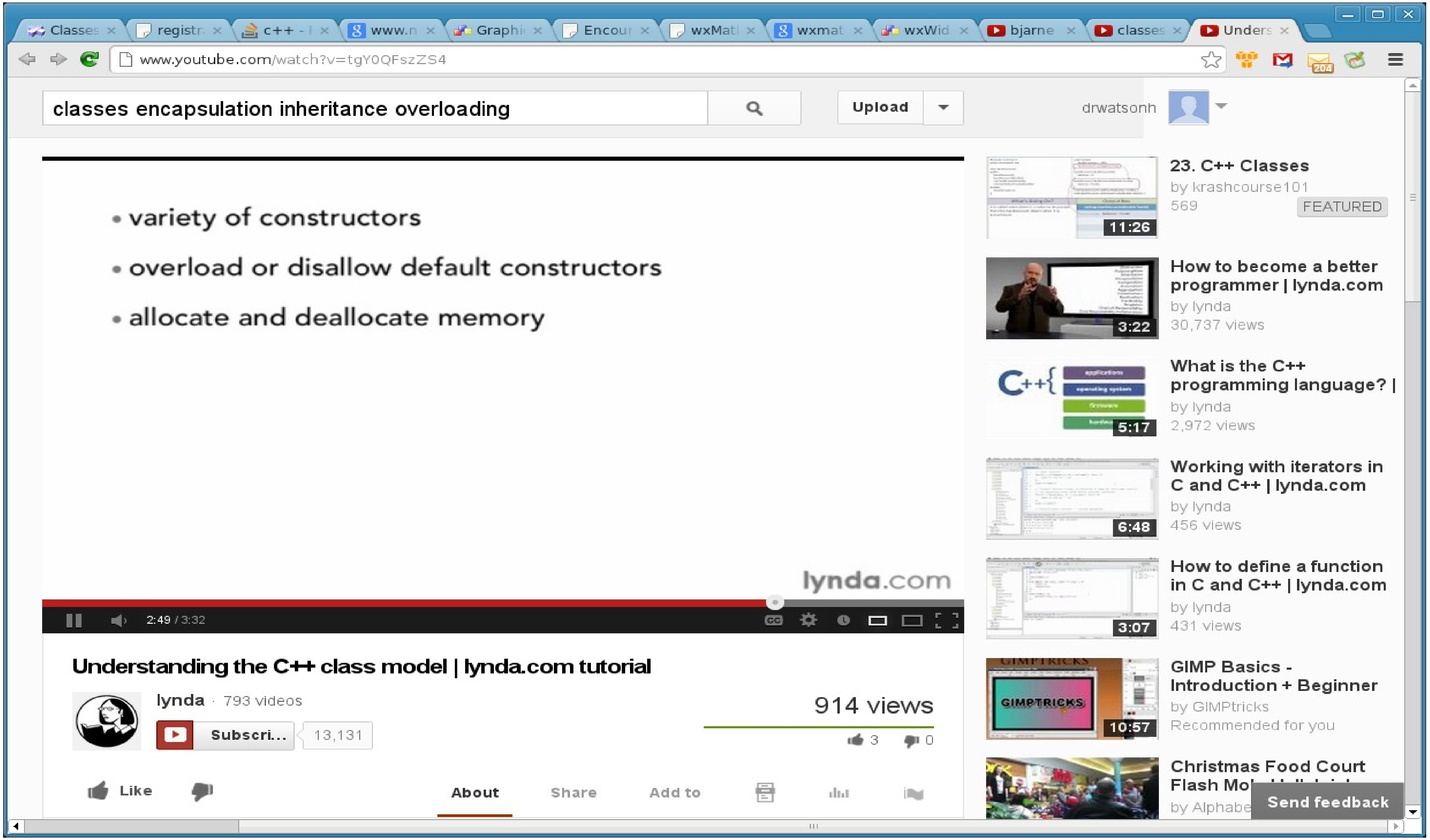
[FS] Unity 3D - MMOR
Project AnaKonda
by FusionStudiosGames
170,603
14:26
FEATU

Tap

Send feedback

What is the C++ programming language?

http://www.youtube.com/watch?v=tgY0QFszZS4



www.youtube.com/watch?v=tgY0QFszZS4
classes encapsulation inheritance overloading
Upload
drwatsonh
23. C++ Classes
by krashcourse101
569
FEATURED
How to become a better programmer | lynda.com
by lynda
30,737 views
What is the C++ programming language? |
by lynda
2,972 views
Working with iterators in C and C++ | lynda.com
by lynda
456 views
How to define a function in C and C++ | lynda.com
by lynda
431 views
GIMP Basics - Introduction + Beginner
by GIMPtricks
Recommended for you
Christmas Food Court Flash Mo
by Alphabe
Send feedback
2:49 / 3:32
lynda.com
lynda · 793 videos
Subscri...
13,131
914 views
3
0
Like
About
Share
Add to

- variety of constructors
- overload or disallow default constructors
- allocate and deallocate memory

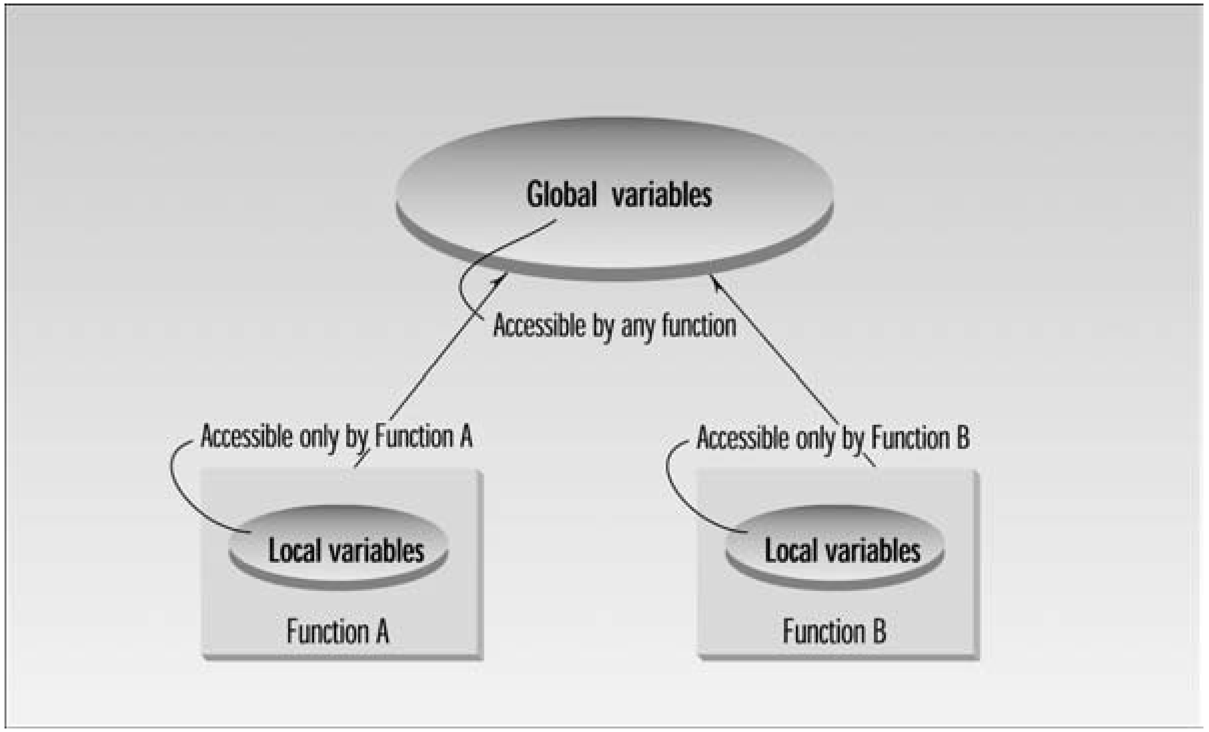
Understanding the C++ class model | lynda.com tutorial

C++ Classes - Part 1 of 2 - Definition and Instantiation
http://www.youtube.com/watch?v=QKCfQtSX8Rg

Going Native 2012 Keynote Stroustrup
http://www.youtube.com/watch?v=OB-bdWKwXsU

Object-Oriented Programming in C++, Fourth Edition

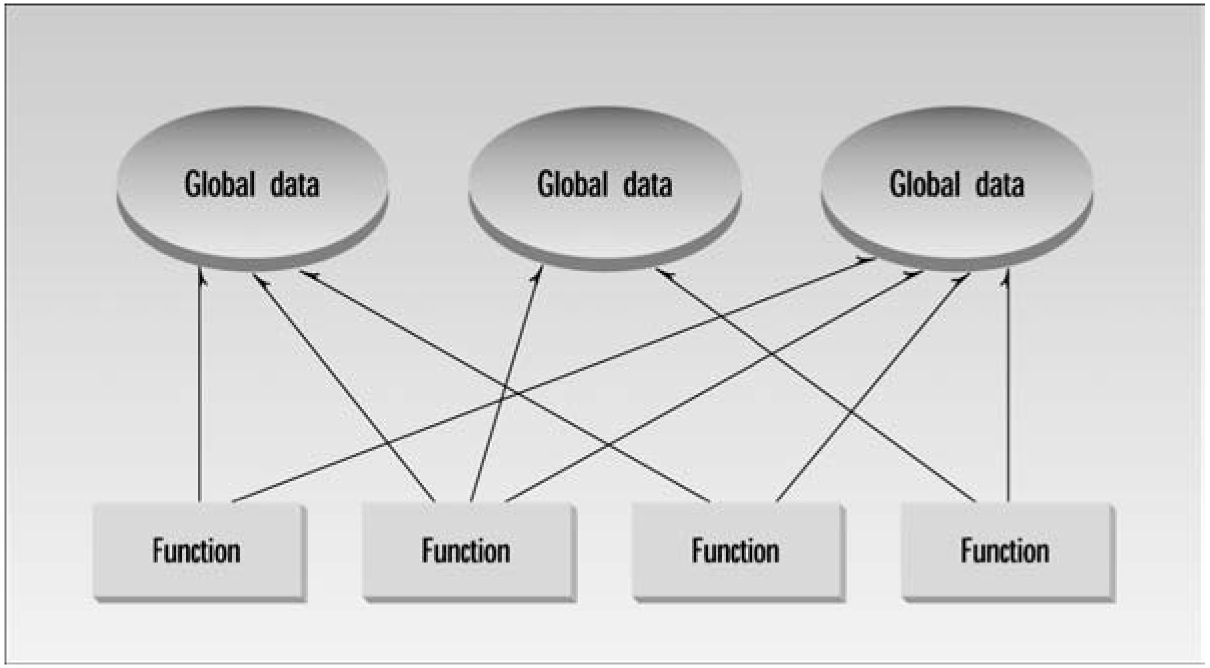
Robert Lafore


Global variables
Accessible by any function
Accessible only by Function A
Accessible only by Function B
Local variables
Local variables
Function A
Function B

Unrestricted Access - Bad

In a procedural program, one written in C for example, there are two kinds of data.
Local data is hidden inside a function, and is used exclusively by the function.

However, when two or more functions must access the same data—and this is true of the most important data in a program—then the data must be made global, as our collection of inventory items is. **Global data** can be accessed by any function in the program



In a large program, there are many functions and many global data items. The problem with the procedural paradigm is that this leads to an even larger number of potential connections between functions and data, as shown in Figure 1.2

This **large number of connections causes problems** in several ways. First, it makes a program's structure difficult to conceptualize. Second, it makes the program difficult to modify. A change made in a global data item may necessitate rewriting all the functions that access that item

Real-World Modeling

The second—and more important—problem with the **procedural paradigm is that its arrangement of separate data and functions does a poor job of modeling things in the real world**. In the physical world we deal with objects such as people and cars. Such objects aren't like data and they aren't like functions. Complex real-world objects have both attributes and behavior.

Attributes (Properties)

Examples of attributes (sometimes called characteristics) are, for people, eye color and job title; and, for cars, horsepower and number of doors. As it turns out, attributes in the real world are equivalent to data in a program: they have a certain specific values, such as blue (for eye color) or four (for the number of doors).

Behavior (Methods)

Behavior is something a real-world object does in response to some stimulus. If you ask your boss for a raise, she will generally say yes or no. If you apply the brakes in a car, it will generally stop. Saying something and stopping are examples of behavior.

Behavior is like a function: you call a function to do something (display the inventory, for example) and it does it.So neither data nor functions, by themselves, model real-world objects effectively.

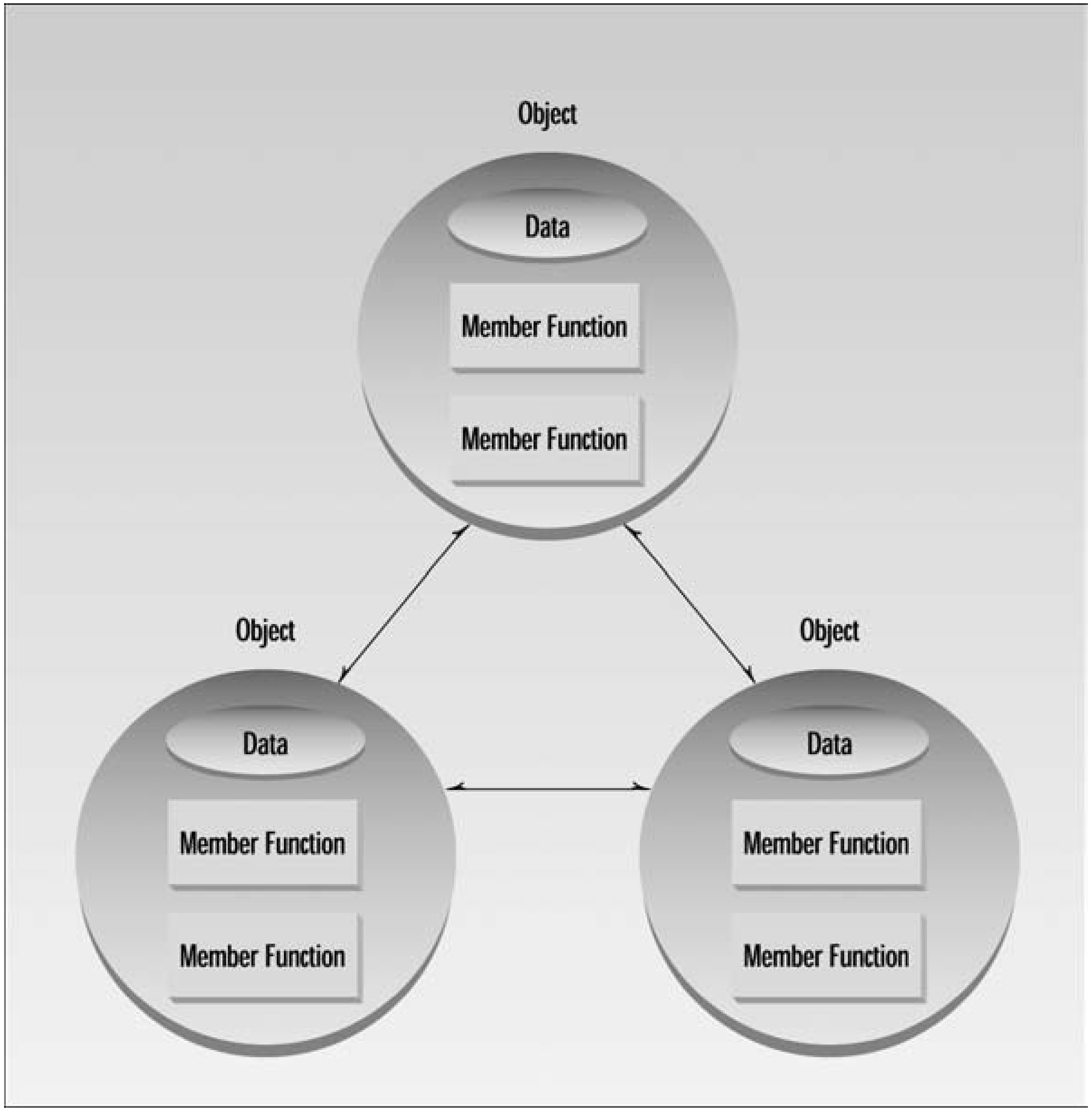
The Object-Oriented Approach

The fundamental idea behind object-oriented languages is to combine into a single unit both data and the functions that operate on that data. Such a unit is called an
Object

An object's functions (methods), called **member functions** in C++, typically provide the only way to access its data. If you want to read a data item in an object, you call a member function in the object. It will access the data and return the value to you. You can't access the data directly.

The data is hidden (properties), so it is safe from accidental alteration. Data and its functions are said to be encapsulated into a single entity. **Data encapsulation and data hiding** are key terms in the description of object-oriented languages.

If you want to modify the data in an object, you know exactly what functions interact with it: the member functions in the object. **No other functions can access the data**. This simplifies writing, debugging, and maintaining the program.


Object
Data
Member Function
Member Function
Object
Data
Member Function
Member Function
Object
Data
Member Function
Member Function

OOP: An Approach to Organization

Object-oriented programming is not primarily concerned with the details of program operation. Instead, it deals with the overall organization of the program.
Most individual program statements in C++ are similar to statements in procedural languages, and many are identical to statements in C. Indeed, an entire member function in a C++ program may be very similar to a procedural function in C. It is only when you look at the larger context that you can determine whether a statement or a function is part of a procedural C program or an object-oriented C++ program

Organization

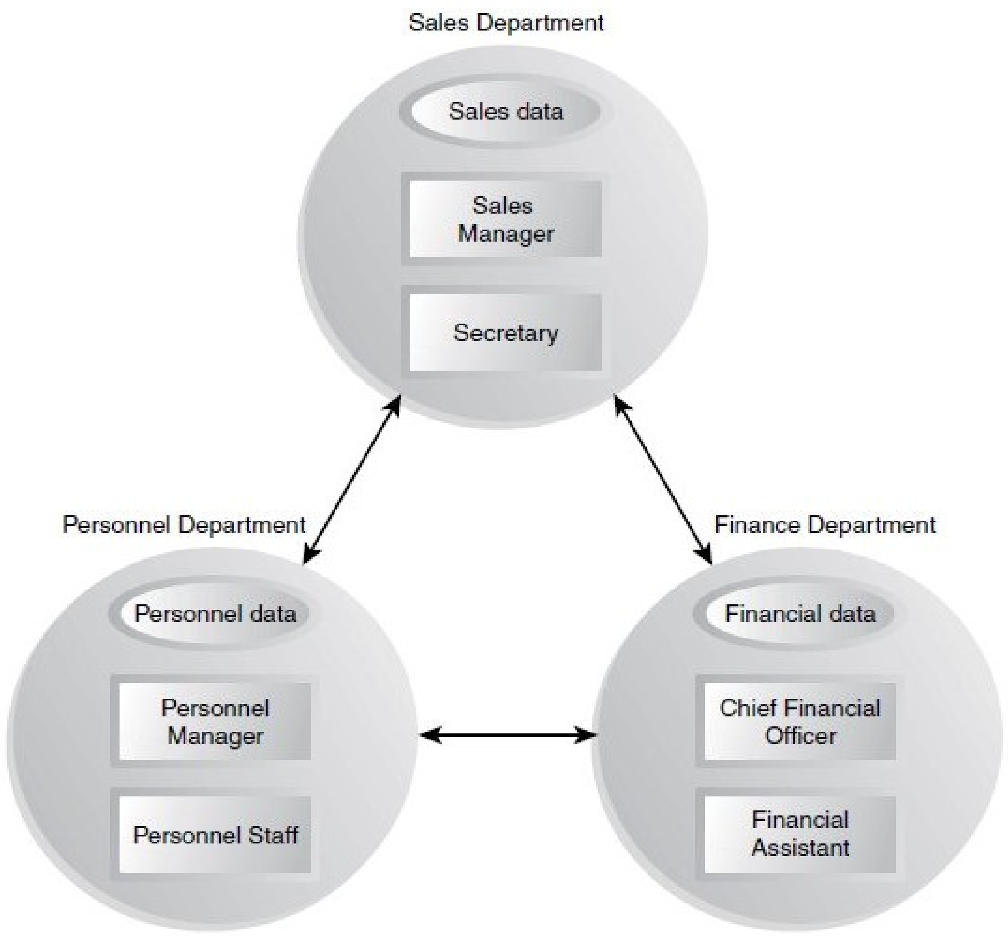


FIGURE 1.4
The corporate paradigm.

Objects

When you approach a programming problem in an object-oriented language, you no longer ask how the problem will be divided into functions, but how it will be divided into objects.

Thinking in terms of objects, rather than functions, has a surprisingly helpful effect on how easily programs can be designed. This results from the close match between objects in the programming sense and objects in the real world.

- Physical objects
  - Automobiles in a traffic-flow simulation
  - Electrical components in a circuit-design program
  - Countries in an economics model
  - Aircraft in an air traffic control system
- Elements of the computer-user environment
  - Windows
  - Menus
  - Graphics objects (lines, rectangles, circles)
  - The mouse, keyboard, disk drives, printer
- Data-storage constructs
  - Customized arrays
  - Stacks
  - Linked lists
  - Binary trees
- Human entities
  - Employees
  - Students
  - Customers
  - Salespeople
- Collections of data
  - An inventory
  - A personnel file
  - A dictionary
  - A table of the latitudes and longitudes of world cities
- User-defined data types
  - Time
  - Angles
  - Complex numbers
  - Points on the plane

Object-oriented programming

Object-oriented programming is a **programming paradigm** that uses abstraction to create models based on the real world. It uses several techniques from previously established paradigms, including modularity, polymorphism, and encapsulation. Today, many popular programming languages (such as Java, JavaScript, C#, C++, Python, PHP, Ruby and Objective-C) support object-oriented programming (OOP).

Object-oriented programming may be seen as the **design of software using a collection of cooperating objects, as opposed to a traditional view in which a program may be seen as a collection of functions, or simply as a list of instructions to the computer**. In OOP, each object is capable of receiving messages, processing data, and sending messages to other objects. **Each object can be viewed as an independent little machine with a distinct role or responsibility.**

Object-oriented programming is intended to promote greater flexibility and maintainability in programming, and is widely popular in large-scale software engineering. By virtue of its strong emphasis on modularity, **object oriented code is intended to be simpler to develop and easier to understand later on,** lending itself to more direct analysis, coding, and understanding of complex situations and procedures than less modular programming methods.[2]

5 Object-orientated programming

Object-orientated languages support to a greater or lesser extent the following concepts

- Encapsulation (including in an object everything it needs, hiding elements that other objects needn't know about). This keeps data and related routines together and unclutters the large-scale organisation of the program. Each object has a 'public' set of routines that can be called, and these routines are all that other objects need to know.
- Inheritance (creating new types of objects from existing ones). Rather than having many seemingly unrelated objects, objects can be organised hierarchically, inheriting behaviour. Again, this simplifies the large-scale organisation.
- Polymorphism (different objects responding to the same message in different ways). Rather than having a different routine to do the same thing to each of many different types of objects, a single routine does the job. An example of this is how the + operator can be overloaded in C++ so that it can be used with new classes.

Using an Object-orientated language often means that

- program entities can more closely model real-world entities. As Stroustrup wrote, "For small to medium projects there often is no distinction made between analysis and design: These two phases have been merged into one. Similarly, in small projects there often is no distinction made between design and programming."
- complexity is more localised
- code re-use is easier

Terminology

Class

Defines the characteristics of the Object.

Object

An Instance of a Class.

Property

An Object characteristic, such as color.

Method

An Object capability, such as walk.

Constructor

A method called at the moment of instantiation.

Inheritance

A Class can inherit characteristics from another Class.

Encapsulation

A Class defines only the characteristics of the Object, a method defines only how the method executes.

Abstraction

The conjunction of complex inheritance, methods, properties of an Object must be able to simulate a reality model.

Polymorphism

Different Classes might define the same method or property

https://developer.mozilla.org/en-US/docs/JavaScript/Introduction_to_Object-Oriented_JavaScript