Programmazione++: Prima Lezione

Shahram Rahatlou

Sapienza Università di Roma

Corso di Programmazione++

http://www.roma1.infn.it/people/rahatlou/programmazione++/

Roma, 16 Marzo 2008

Introduzione

- Io sono un fisico e lavoro sulle particelle elementari
- Non sono un programmatore ne` un guru di C++!
- Allora perche` insegno questo corso?
  - Negli ultimi anni C++ ha sostituito Fortran come linguaggio di programmazione in Fisica delle Alte Energie
    - Grossi esperimenti hanno adottato C++ gia` nei primi anni 90
    - Molti strumenti di analisi dati e metodi numerichi sono stati riscritti in C++
    - Io utilizzo C++ dal 1998 quando ero laureando!
      - Simulazione di rivelatori
      - Estrazione segnali dei rivelatori
      - Analisi dati
      - Fit multidimensionale ed estrazione di parametri del Modello Standard di Fisica delle Particelle

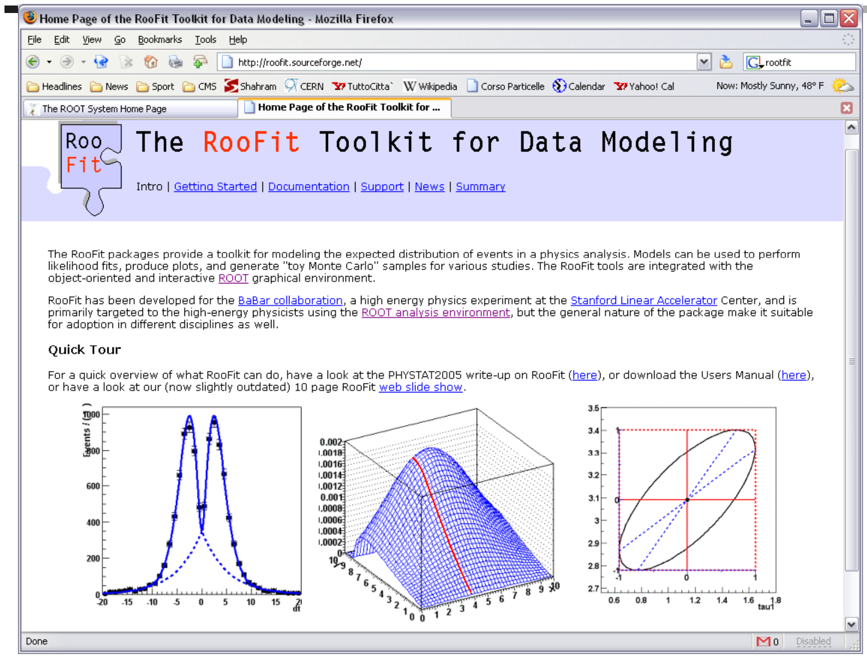
Cosa faremo in questo corso?

- Ipotesi di base
  - tutti voi avete gia` seguiti corsi di programmazione
    - Corsi tenuti da G. Organtini e L. Barone
  - Sapete perche` scriviamo programmi ed applicazioni nei vari campi della Fisica
  - Sapete cosa vuol dire compilare un programma

- Pero`... se siete del tutto nuovi alla programazione, questo corso vi spieghera` come fare
  - Conoscenze priori vi aiuteranno ma non sono necessarie

- Obiettivo del corso:
  - Capire l'importanza di programmazione ad oggetti e l'uso di C++ come una possibile implementazione
  - Essere in grado di scrivere semplici programmi in C++ usando classi appropriate per risolvere i vostri problemi di fisica!
  - Imparare ad usare librerie e tool esterni nei vostri programmi
    - Ad esempio root per analisi dei dati e librerie numeriche per integrazioni e simulazioni , librerie grafiche per creare GUI o anche giochi

Esempi di Uso di C++ in Fisica



Home Page of the RooFit Toolkit for Data Modeling - Mozilla Firefox

http://roofit.sourceforge.net/

The RooFit Toolkit for Data Modeling

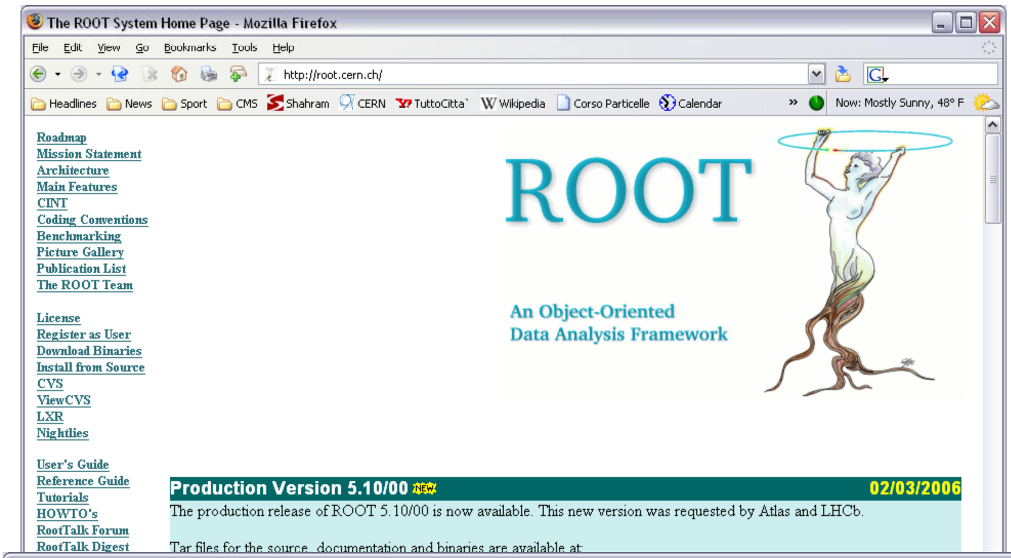
Intro | Getting Started | Documentation | Support | News | Summary

The RooFit packages provide a toolkit for modeling the expected distribution of events in a physics analysis. Models can be used to perform likelihood fits, produce plots, and generate "toy Monte Carlo" samples for various studies. The RooFit tools are integrated with the object-oriented and interactive ROOT graphical environment.

RooFit has been developed for the BaBar collaboration, a high energy physics experiment at the Stanford Linear Accelerator Center, and is primarily targeted to the high-energy physicists using the ROOT analysis environment, but the general nature of the package make it suitable for adoption in different disciplines as well.

Quick Tour

For a quick overview of what RooFit can do, have a look at the PHYSTAT2005 write-up on RooFit (here), or download the Users Manual (here), or have a look at our (now slightly outdated) 10 page RooFit web slide show.



The ROOT System Home Page - Mozilla Firefox

http://root.cern.ch/

Roadmap
Mission Statement
Architecture
Main Features
CINT
Coding Conventions
Benchmarking
Picture Gallery
Publication List
The ROOT Team

License
Register as User
Download Binaries
Install from Source
CVS
ViewCVS
LXR
Nightlies

User's Guide
Reference Guide
Tutorials
HOWTO's
RootTalk Forum
RootTalk Digest

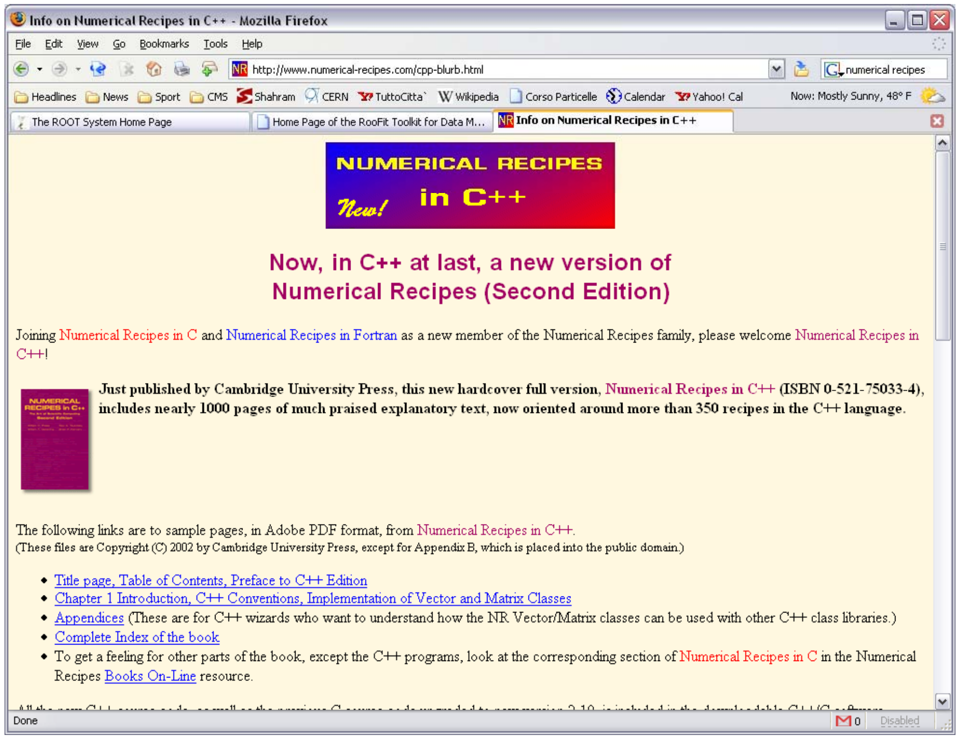
ROOT

An Object-Oriented
Data Analysis Framework

Production Version 5.10/00 NEW **02/03/2006**

The production release of ROOT 5.10/00 is now available. This new version was requested by Atlas and LHCb.

Tar files for the source, documentation and binaries are available at



Info on Numerical Recipes in C++ - Mozilla Firefox

http://www.numerical-recipes.com/cpp-blurb.html

NUMERICAL RECIPES
New! in C++

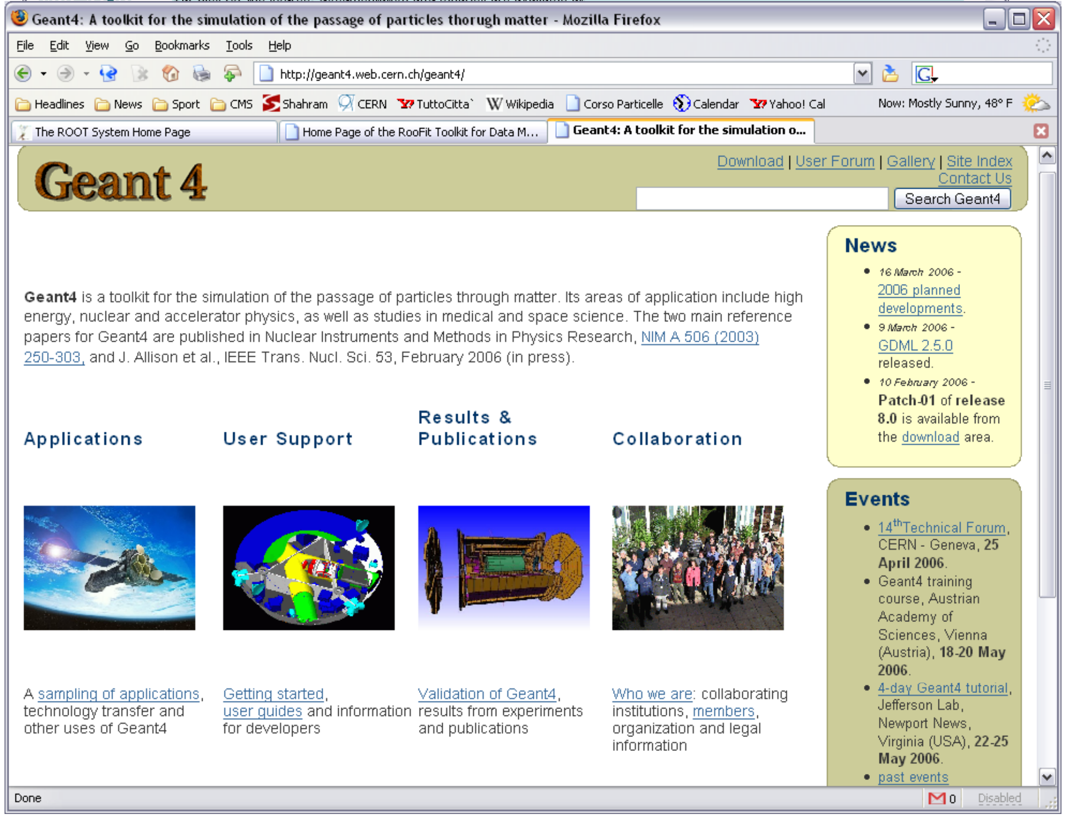
Now, in C++ at last, a new version of Numerical Recipes (Second Edition)

Joining Numerical Recipes in C and Numerical Recipes in Fortran as a new member of the Numerical Recipes family, please welcome Numerical Recipes in C++!

Just published by Cambridge University Press, this new hardcover full version, Numerical Recipes in C++ (ISBN 0-521-75033-4), includes nearly 1000 pages of much praised explanatory text, now oriented around more than 350 recipes in the C++ language.

The following links are to sample pages, in Adobe PDF format, from Numerical Recipes in C++.
(These files are Copyright (C) 2002 by Cambridge University Press, except for Appendix B, which is placed into the public domain.)

- Title page, Table of Contents, Preface to C++ Edition
- Chapter 1 Introduction, C++ Conventions, Implementation of Vector and Matrix Classes
- Appendices (These are for C++ wizards who want to understand how the NR Vector/Matrix classes can be used with other C++ class libraries.)
- Complete Index of the book
- To get a feeling for other parts of the book, except the C++ programs, look at the corresponding section of Numerical Recipes in C in the Numerical Recipes Books On-Line resource.



Geant4: A toolkit for the simulation of the passage of particles thorugh matter - Mozilla Firefox

http://geant4.web.cern.ch/geant4/

Geant 4

Download | User Forum | Gallery | Site Index
Contact Us

Geant4 is a toolkit for the simulation of the passage of particles through matter. Its areas of application include high energy, nuclear and accelerator physics, as well as studies in medical and space science. The two main reference papers for Geant4 are published in Nuclear Instruments and Methods in Physics Research, NIM A 506 (2003) 250-303, and J. Allison et al., IEEE Trans. Nucl. Sci. 53, February 2006 (in press).

Applications	User Support	Results & Publications	Collaboration
A sampling of applications, technology transfer and other uses of Geant4	Getting started, user guides and information for developers	Validation of Geant4, results from experiments and publications	Who we are: collaborating institutions, members, organization and legal information

News

- *16 March 2006* - 2006 planned developments.
- *9 March 2006* - GDML 2.5.0 released.
- *10 February 2006* - **Patch-01** of **release 8.0** is available from the download area.

Events

- 14th Technical Forum, CERN - Geneva, **25 April 2006**.
- Geant4 training course, Austrian Academy of Sciences, Vienna (Austria), **18-20 May 2006**.
- 4-day Geant4 tutorial, Jefferson Lab, Newport News, Virginia (USA), **22-25 May 2006**.
- past events

Organizzazione delle Lezioni

- Lunedi` : 2 ore di lezione
  - Soprattutto nuovi argomenti
  - Cercheremo di trattare un aspetto importante a settimana

- Martedi` : 2 ore lezione o 3 ore al laboratorio
  - Sessioni di laborarorio e lezioni saranno a settimane alterne
  - Rifiniture di argomenti della settimana
  - Applicazione ed implementazione di esempi degli argmenti trattati
  - Probabilmente tanti piccoli esempi anziche` un grosso progetto da finire durante tutta la durata del corso
    - Vantaggio: ripetendo certi passi di base ve li ricorderete meglio!

- Tutte le lezioni ed esempi disponibili online subito dopo la lezione
  http://www.roma1.infn.it/people/rahatlou/programmazione++/

Come Funzionera` il Corso?

- Non discutero` in dettaglio tutti i possibili operatori, comandi e sintassi di C++
  - Ci sono ottimi libri e siti web che illustrano con una varieta` di esempi tutti gli aspetti del linguaggio
  - Cercare di ripetere questo livello di dettaglio a lezione e` dispersivo ed inutile
- Le lezioni focalizzeranno su aspetti importanti che rendono C++ superiore a C e migliore di molti altri linguaggi in giro
- Forniro` esempi ben specifici per illustrarvi l'utilizzo di C++ e possibili problemi tecnici
- Vi consiglio di farvi un giro per i tantissimi siti web dedicati a C++ e provare i vari tutorial online gratis

Imparare un Linguaggio come una Lingua

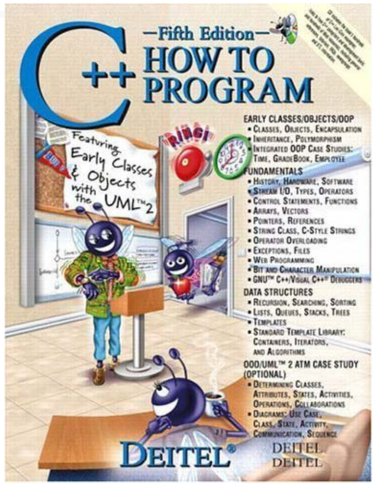
- Come con una lingua umana, un nuovo linguaggio si impara solo attraverso esempi e sbagliando la sintassi
  - Teoria e` inutile se poi il programma non compila oppure compila ma non gira!
- Scrivere programmi semplici per capire anche un solo aspetto di C++ e` fondamentale perche`
  - Imparerete a capire e risolvere gli errori di compilazione
    - Di vitale importanza quando vi troverete a lavorare con programmi di $O(10^6)$ righe di codice o codice non scritto da voi!
  - Cambiando in tutti i modi un programma banale saprete cosa aspettarvi da C++
  - Ripetendo alcuni passi base di programmazione in C++ farete sempre meno errori banali al passare del tempo e vi concentrete sugli aspetti piu` sofisticati e complessi del linguaggio

Lingua del Corso

- Le lezioni saranno in italiano ma le trasparenze in inglese

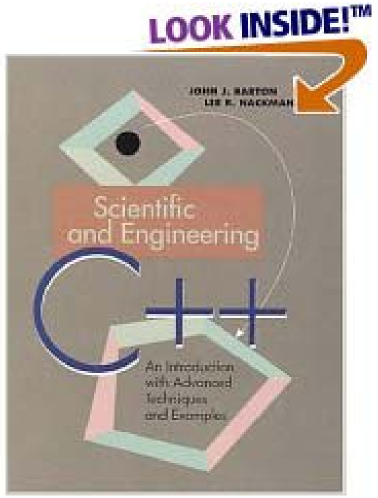
- Perche`?
  - Molti termini tecnici non sono nemmeno tradotti in italiano
    - Cercate ad esempio “template” o “design pattern”
  - Sara` piu` facile per voi cercare referenze e materiale aggiuntivo sugli argomenti trattati se imparate i termini giusti
  - Vi abituate ai testi in inglese che nei corsi che seguirete d’ora in avanti saranno sempre piu` comuni
  - Ed anche perche` cosi` io posso utilizare queste lezioni eventualmente anche altrove

Alcuni Testi da Consulatare



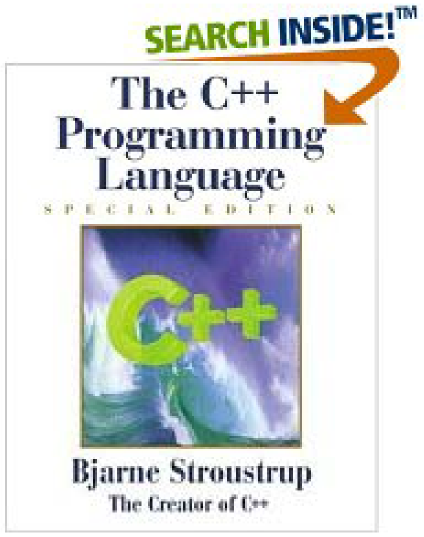
- Deitel & Deitel, **`C++ How To Program 5th Edition`**, editore Pearson - Prentice Hall
  - Buono se non conoscete C++ ed avete bisogno di un testo che vi segua passo per passo. Tantissimi esempi. Moltissimi consigli utili sia per principianti che per i piu` esperti
  - La versione italiana dovrebbe essere gia` disponibile nelle biblioteche Di recente e` uscita anche la 6ª edizione
  - Link su Amazon.com

- J. Barton & L. Nackman, `Scientific and Engineering C++`, editore Addison-Wesley
  - Semplice e conciso. Un po` minimalista
  - Tratta tutti gli argomenti importanti ma forse poco discorsivo per essere un libro di testo
  - Link su Amazon.com

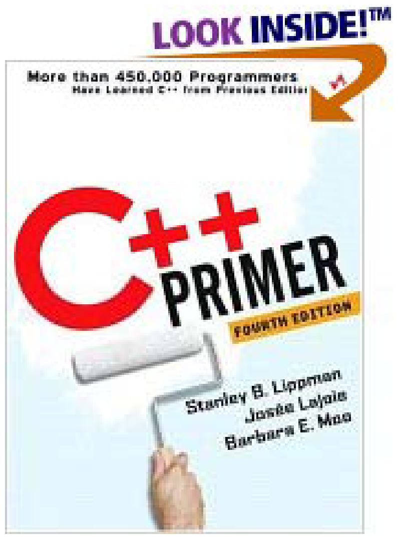


Altri Testi di Riferimento

- B. Stroustrup, C++ Programming Language
  - La bibbia del C++ direttamente dallo sviluppatore del C++
  - Molto piu` denso ma completo sotto tutti gli aspetti
  - Non un testo didattico ma ottimo punto di riferimento per capire meglio aspetti specifici del linguaggio
  - Un po` come il Landau per la meccanica!
  - Link su Amazon.com



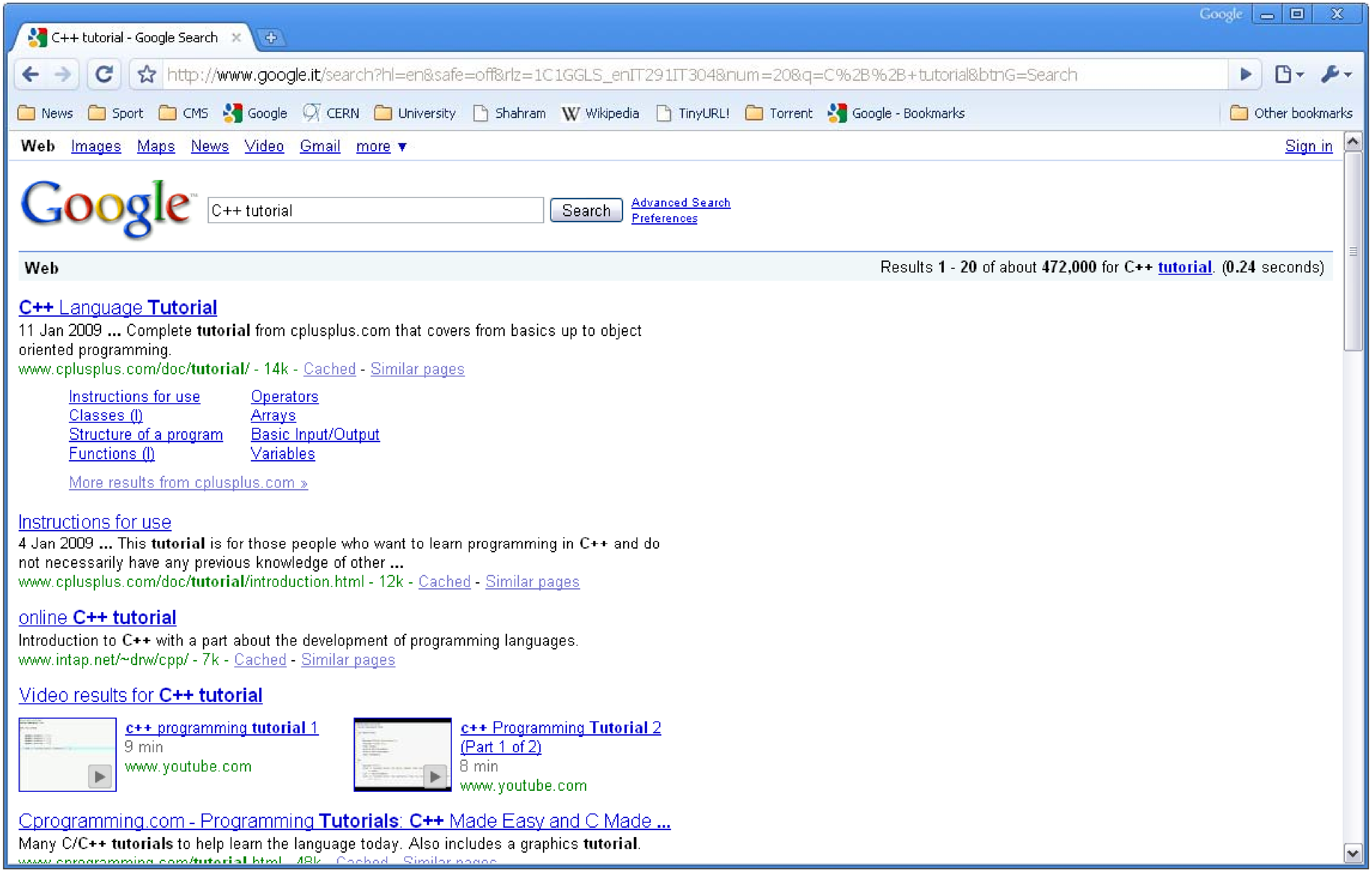
- Lippman, C++ Primer,
  - Un altro testo completo forse un po` meno pesante del precedente
  - Link su Amazon.com



Tematiche Trattate in questo Corso

- Elementi del linguaggio C++
  - aspetti comuni e differenze rispetto al linguaggio C
- Introduzione alla programmazione agli oggetti
- Classi ed oggetti
- Polimorfismo, ereditarieta`, ed incapsulamento
- Astrazione: classi virtuali ed interfacce
- Introduzione alla programmazione generica: templates
- Compilatori, makefile, ed organizzaione del codice
- Librerie e dipendenze tra oggetti
- Gestione degli errori: eccezioni

Moltissime Risorse in Rete



C++ tutorial - Google Search

http://www.google.it/search?hl=en&safe=off&rlz=1C1GGLS_enIT291IT304&num=20&q=C%2B%2B+tutorial&btnG=Search

News Sport CMS Google CERN University Shahram Wikipedia TinyURL! Torrent Google - Bookmarks Other bookmarks

Web Images Maps News Video Gmail more ▼ Sign in

C++ tutorial Search Advanced Search Preferences

Web Results **1** - **20** of about **472,000** for **C++ tutorial**. (**0.24** seconds)

C++ Language **Tutorial**
11 Jan 2009 ... Complete **tutorial** from cplusplus.com that covers from basics up to object oriented programming.
www.cplusplus.com/doc/**tutorial**/ - 14k - Cached - Similar pages

Instructions for use — Operators
Classes (I) — Arrays
Structure of a program — Basic Input/Output
Functions (I) — Variables

More results from cplusplus.com »

Instructions for use
4 Jan 2009 ... This **tutorial** is for those people who want to learn programming in C++ and do not necessarily have any previous knowledge of other ...
www.cplusplus.com/doc/**tutorial**/introduction.html - 12k - Cached - Similar pages

online **C++ tutorial**
Introduction to C++ with a part about the development of programming languages.
www.intap.net/~drw/cpp/ - 7k - Cached - Similar pages

Video results for **C++ tutorial**

c++ programming **tutorial** 1
9 min
www.youtube.com

c++ Programming **Tutorial** 2 (Part 1 of 2)
8 min
www.youtube.com

Cprogramming.com - Programming **Tutorials**: **C++** Made Easy and C Made ...
Many C/C++ **tutorials** to help learn the language today. Also includes a graphics **tutorial**.

Alcuni ottimi siti

- www.cplusplus.com
  - Tutorial, lezioni, manuali di riferimento, FAQ
- http://www.cppreference.com/
  - Un vero e proprio manuale online
- Librerie STL : http://www.sgi.com/tech/stl/
  - STL == Standard Template Library
  - Un vasto insieme di librerie per usare oggetti di uso comune come `vector`, `string`, `map` etc.
  - Non serve sapere tutto, e decisamente non a memoria!
  - Basta saper trovare le risorse giuste per utilizzare tutte le funzionalita` offerte da STL
- Moltissimi corsi di C++ nelle universita` americane sono disponibili in rete comprese lezioni e tanti tantissimi esempi!

Brief History of C

- C was developed in 1967 mainly as a language for writing operating systems and compilers
  - Think about the gcc compiler and Linux today
  - You can compile the gcc compiler yourself
  - You can get the latest linux kernel (core of the linux operating system) from www.kernels.org and compile it yourself

- C was the evolution of two previous languages: B and BCPL
  - Both used to develop early versions of UNIX at Bell Labs

- C became very popular and was ported to variety of different hardware platforms
  - C was standardized in 1990 by International Organization for Standardization (ISO) and American National Standards Institute (ANSI)
    - ANSI/ISO 9899: 1990

Object Oriented Programming and Birth of C++

- By 1970's the difficulties of maintaining very large software projects for companies and businesses had lead to structured programming
  - From Wikipedia:
    Structured programming can be seen as a subset or subdiscipline of procedural programming, one of the major programming paradigms. It is most famous for removing or reducing reliance on the GOTO statement (also known as "go to")
- By late 70's a new programming paradigm was becoming trendy: object orientation
- In early 1980's Bjarne Stroustrup developed C++ using features from C but adding capabilities for object orientation

What is ‘Object Oriented Programming` anyway?

- Objects are software units modeled after entities in real life
  - Objects are entities with attributes: length, density, elasticity, thermal coefficient, color
  - Objects have a behavior and provide functionalities
    - A door can be opened
    - A car can be driven
    - A harmonic oscillator oscillates
    - A nucleus can decay
    - A planet moves in an orbit
- Object orientation means writing your program in terms of well defined units (called objects) which have attributes and offer functionalities
  - Program consists in interaction between objects using methods offered by each of them

Object-Oriented According to Wikipedia

- In computer science, **object-oriented programming**, **OOP** for short, is a computer programming paradigm.

- The idea behind object-oriented programming is that a computer program may be seen as comprising a collection of individual units, or *objects*, that act on each other, as opposed to a traditional view in which a program may be seen as a collection of functions or procedures, or simply as a list of instructions to the computer. 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 or actor with a distinct role or responsibility.

- Object-oriented programming is claimed to promote greater flexibility and maintainability in programming, and is widely popular in large-scale software engineering. Furthermore, proponents of OOP claim that OOP is easier to learn for those new to computer programming than previous approaches, and that the OOP approach is often simpler to develop and to maintain, lending itself to more direct analysis, coding, and understanding of complex situations and procedures than other programming methods. Critics dispute this, at least for some domains (industries).

C++ is not C !

- Don’t be fooled by the name!

- C++ was developed to overcome limitations of C and improve upon it
  - C++ looks like C but feels very differently
  - C++ shares many basic functionalities but improves upon many of them
    - For example input/output significantly better in C++ than in C

- C excellent language for structural programming
  - Focused around actions on data structures
  - Provides methods which act on data and create data

- C++ focused on inter-action between objects
  - Objects are ‘smart’ data structures: data with behavior!

What You Need to compile your C++ Program?

- On Linux/unix machines you should have the g++ compiler installed by default
- On Windows you can use cygwin which looks like UNIX and has also gcc/g++
- Or you can also use the free Visual C++

Simple

- Harness latest features of the Windows Platform SDK starting with the streamlined installation
- Create native Win32 applications
- Develop rich 2D and 3D games with The Game Creators GDK

- I prefer the bare gcc since you need to also understand the compiler and its options

Structure of a C++ Program

```
// your first C++ application!
#include <iostream> // required to perform C++ stream I/O

// function main begins program execution
int main() {
   return 0; // indicate that program ended successfully

} // end function main
```

Precompiler/Preprocessor Directives

What is the preprocessor? What does it do?

```
// your first C++ application!
#include <iostream> // required to perform C++ stream I/O

// function main begins program execution
int main() {
   return 0; // indicate that program ended successfully

} // end function main
```

`iostream` will be included before compiling this code!

What does the Preprocessor do?

- Replace user directives with requested source code
  - Foo.h is included in ExamplePreprocessor.cpp

```
// Foo.h
class Foo {
public:
 Foo() {};
 Foo(int a) { x_ = a; };

private:
 int x_;
};
```

```
// ExamplePreprocessor.cpp
#include "Foo.h"


int main() {

  return 0;
}
```

Pre-compile only

```
$ g++ -E ExamplePreprocessor.cpp > prep.cc
$ cat prep.cc
# 1 "ExamplePreprocessor.cpp"
# 1 "<built-in>"
# 1 "<command line>"
# 1 "ExamplePreprocessor.cpp"

# 1 "Foo.h" 1
class Foo {
public:
 Foo() {};
 Foo(int a) { x_ = a; };

private:
 int x_;
};
# 3 "ExamplePreprocessor.cpp" 2

int main() {

  return 0;
}
```

Comments in C++

- Comments preceded by //
  - Can start anywhere in the program either at the beginning or right after a statement in the middle of the line

```
// your first C++ application!
#include <iostream> // required to perform C++ stream I/O

// function main begins program execution
int main() {
   return 0; // indicate that program ended successfully

} // end function main
```

Compiling a C++ application

- We will be using the free compiler gcc throughout the examples in this course

```
$ g++ -o Welcome     Welcome.cpp
```

Name of the binary output

C++ file to compile and link

```
$ ls -l
-r--r--r--  1 rahatlou None   1379 Apr 18 22:55 Welcome.cpp
-rwxr-xr-x  1 rahatlou None 476600 Apr 18 22:57 Welcome
```

Some basic aspects of C++

- All statements must end with a semi-colon ;
  - Carriage returns are not meaningful and ignored by the compiler
- Comments are preceded by //
  - Comments can be an entire line or in the middle of the line after a statement
- Any C++ application must have a **`main`** method
- **`main`** must return an **`int`**
  - Return value can be used by user/client/environment
  - E.g. to understand if there was an error condition

What about changing a different type of **`main`**?

```
// VoidMain.cpp
#include <iostream>
using namespace std;

void main(){

  // no return type

} // end function main
```

```
$ g++ -o VoidMain VoidMain.cpp
VoidMain.cpp:6: error: `main' must return `int'
```

- Compiler requires **`main`** to return an **`int`** value!
- Users must simply must satisfy this requirement
  - If you need a different type there is probably a mistake in your design!

Output with **`iostream`**

- **`iostream`** provides output capabilities to your program

```
// SimpleIO.cpp
#include <iostream>
using namespace std;

int main() { // main begins here

   // print message to STDOUT
   cout << “Moving baby steps in C++!” << endl;

   return 0;

} // end of main
```

End of line
start a new line!

```
$ g++ -o SimpleIO SimpleIO.cpp
$ ./SimpleIO
Moving baby steps in C++!
```

Input with `iostream`

- **`iostream`** provides also input capabilities to your program

```
// SimpleInput.cpp
#include <iostream>
using namespace std;

int main() { // main begins here

   int nIterations;

   cout << "How many iterations? ";
   cin >> nIterations;

   // print message to STDOUT
   cout << "Number of requested iterations: " << nIterations << endl;

   return 0;
} // end of main
```

Put content of cin into variable nIterations

```
$ g++ -o SimpleInput SimpleInput.cpp
$ ./SimpleInput
How many iterations? 7
Number of requested iterations: 7
```

Typical Compilation Errors So Far

```
// BadCode1.cpp
#include <iostream>
using namespace std;

int main() { // main begins here

   int nIterations;

   cout << "How many
           iterations? "; // cannot break in the middle of the string!

   cin >> nIteration; // wrong name! the s at the end missing

   // print message to STDOUT
   cout << "Number of requested iterations: " << nIterations << endl;

   return 0 // ; is missing!

} // end of main
```

```
$ g++ -o BadCode1 BadCode1.cpp
BadCode1.cpp: In function `int main()':
BadCode1.cpp:9: error: missing terminating " character
BadCode1.cpp:10: error: `iterations' undeclared (first use this function)
BadCode1.cpp:10: error: (Each undeclared identifier is reported only once for each function
 it appears in.)
BadCode1.cpp:10: error: missing terminating " character
BadCode1.cpp:12: error: `nIteration' undeclared (first use this function)
BadCode1.cpp:12: error: expected `:' before ';' token
BadCode1.cpp:12: error: expected primary-expression before ';' token
BadCode1.cpp:19: error: expected `;' before '}' token
```

Declaration and Definition of Variables

```
// SimpleVars.cpp
#include <iostream>
using namespace std;

int main() {

   int samples; // declaration only

   int events = 0; // declaration and assignment

   samples = 123; // assignment

   cout << "How many samples? " ;
   cin >> samples; // assigment via I/O

    cout << "samples: " << samples
         << "\t"  // insert a tab in the printout
         <<  "events: " << events
         << endl;

   return 0;
} // end of main
```

```
$ g++ -o SimpleVars SimpleVars.cpp
$ ./SimpleVars
How many samples? 3
samples: 3      events: 0
```

Loops and iterations in C++

```
int main() { // main begins here

   int nIterations;
   cout << "How many iterations? ";
   cin >> nIterations;

   int step;
   cout << "step of iteration? " ;
   cin >> step;

   for(int index=0; index < nIterations; index+=step) {
     cout << "index: " << index << endl;
   }
   return 0;
} // end of main
```

Maximum

Starting value

step

```
$ g++ -o SimpleLoop SimpleLoop.cpp
$ ./SimpleLoop
How many iterations? 7
step of iteration? 3
index: 0
index: 3
index: 6
```