CPA: PROGRAMMING ESSENTIALS IN C++

OVERVIEW

DEVELOPED BY C++ INSTITUTE

CONTENTS

1. OVERVIEW
2. COURSE DESIGN
3. SCOPE AND SEQUENCE
4. HOW TO USE THE COURSE
5. CERTIFICATION
6. WHY LEARN PROGRAMMING
7. WHY LEARN C++
8. C/C++ EXAMPLES
9. KEY TAKEAWAYS

OVERVIEW

CPA: PROGRAMMING ESSENTIALS IN C++

- For beginners with little or no prior knowledge of programming.
- Designed to be a full semester course: 9 chapters, 16 quizzes and chapter assessments, 100+ lab exercises, pre-final and final tests.
- Accessed online with no special equipment or system requirements.
- Aligns to C++ Institute *CPA – C++ Certified Associate Programmer Certification.*
- Instructor-Led Training offered at no cost.
- Self-paced access offered at no cost (not available yet).

TARGET AUDIENCE

The *CPA: Programming Essentials in C++* curriculum is designed for students who want to learn the fundamentals of programming through the C++ language.

CURRICULUM DESCRIPTION

This course covers all the basics of programming in the C++ programming language as well as the fundamental concepts and techniques used in object-oriented programming. The course starts with some universal basics, without relying on object concepts, and gradually extends to the advanced issues the student will encounter when using the objective approach.

TARGET CERTIFICATION

The *CPA: Programming Essentials in C++* curriculum helps students prepare for the *CPA – C++ Certified Associate Programmer certification* exam. *C++ Certified Associate Programmer (CPA)* is a professional certification that measures the ability to accomplish coding tasks related to the basics of programming in the C++ language and the fundamental notions and techniques used in object-oriented programming.

CPA: PROGRAMMING ESSENTIALS IN C++
Small Market Trial (SMT) Survey Results

INSTRUCTOR FEEDBACK

89% of instructors say they are satisfied or very satisfied with **this course as a whole as well as with the labs and activities**.

82% of instructors say they are satisfied or very satisfied with **on-line curriculum materials**.

85% of instructors say they are satisfied or very satisfied with **assessments (including quizzes, chapter tests and the final test)**

85% of instructors say they are **likely or very likely to recommend this course**.

93% of instructors **plan to offer the course in the future.**

93% of instructors say **the course will help their students to learn skills that can be used in a current market or future job.**

85% of instructors say **the course will help their students to increase their value in the job market quite a bit or very much.**

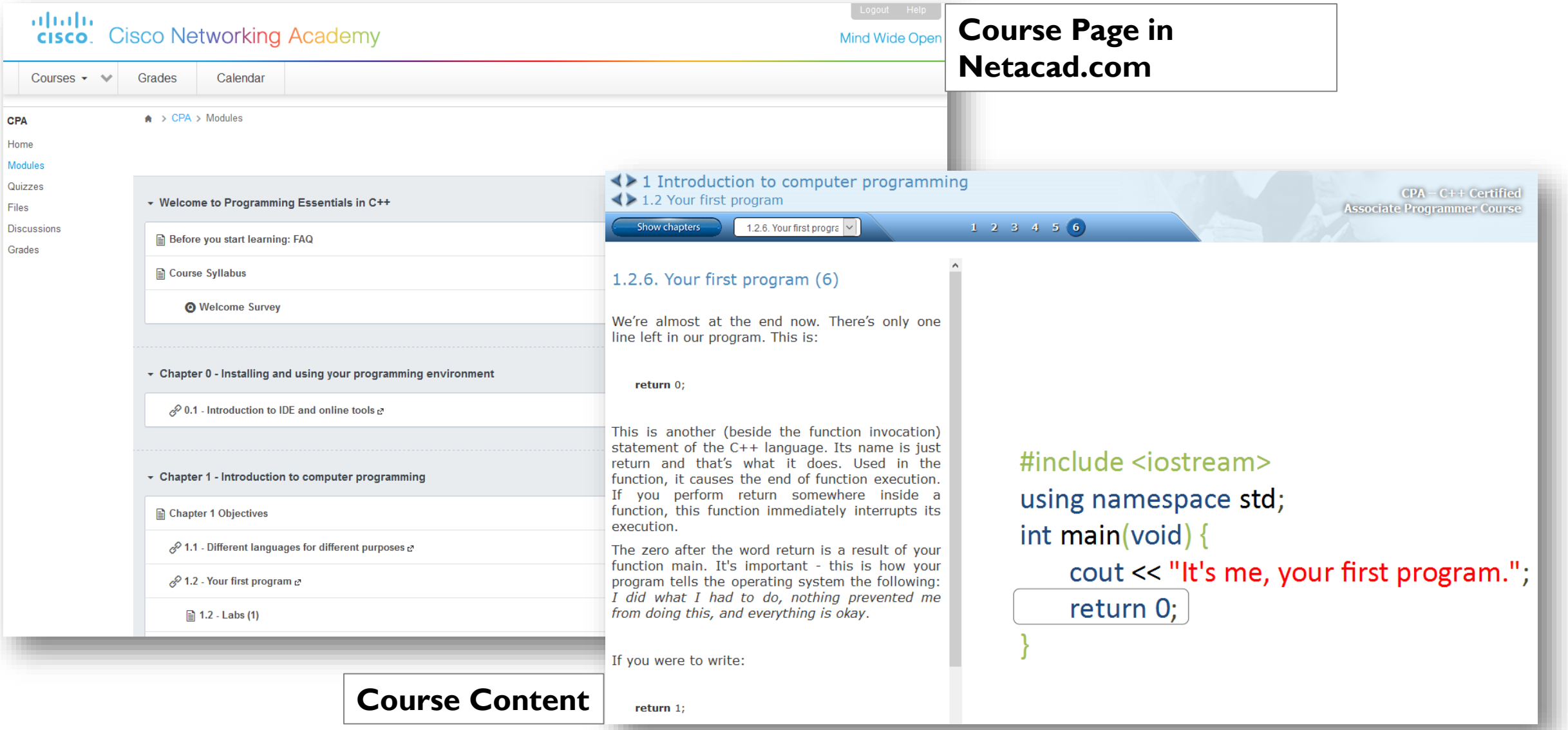
COURSE DESIGN

The course is broken down into 9 modules:

- Module 0: explains the process of installing and using the programming environment.
- Module 1: introduces common computer programming concepts, e.g. integers and variables.
- Module 2: discusses advanced flow control and data aggregates.
- Module 3: introduces the subject of pointers, functions and memory.
- Module 4: examines ways to access various kinds of data.
- Module 5: gives an introduction to the fundamentals of object-oriented programming.
- Module 6: discusses the subject of inheritance.
- Module 7: considers the subject of exceptions.
- Module 8: discusses the subject of operators and enumerated types.

Each student has access to hands-on practice materials, quizzes and assessments to learn how to utilize the skills and knowledge gained on the course and interact with some real-life programming tasks and situations.

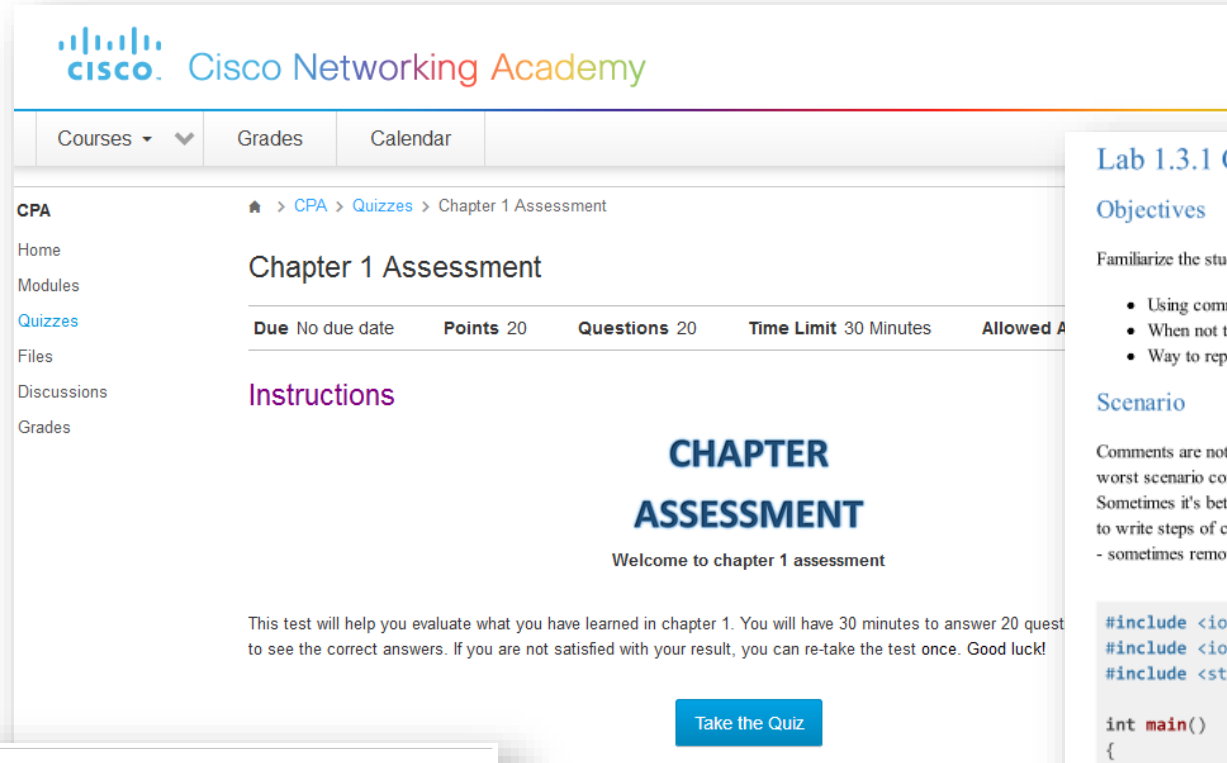
COURSEWARE

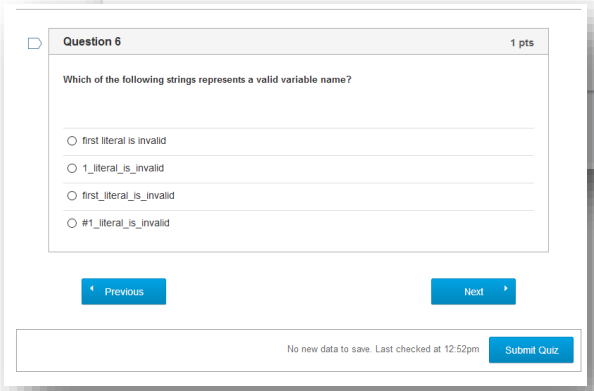


Course Page in Netacad.com

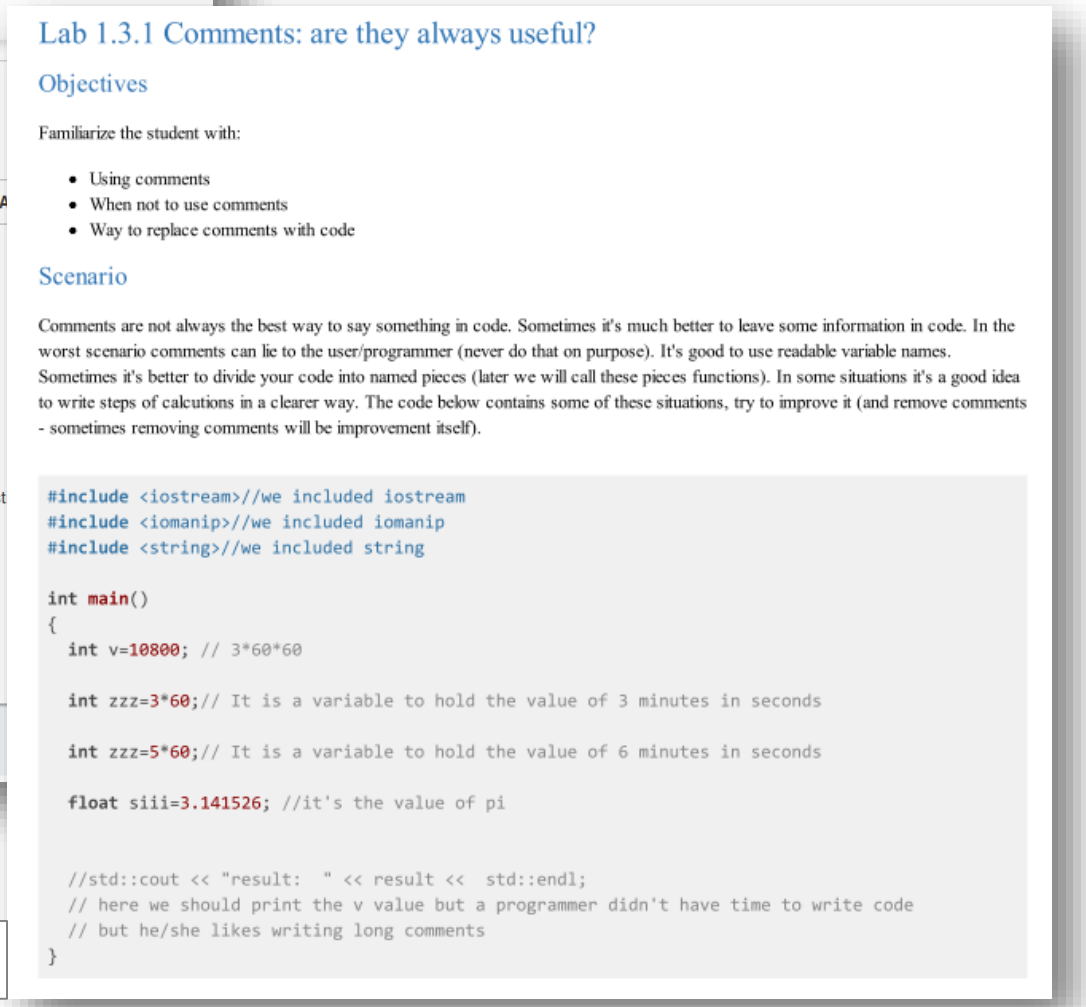
Course Content

Chapter Assessments and Quizzes





Lab exercises



Lab 1.3.1 Comments: are they always useful?

Objectives

Familiarize the student with:

- Using comments
- When not to use comments
- Way to replace comments with code

Scenario

Comments are not always the best way to say something in code. Sometimes it's much better to leave some information in code. In the worst scenario comments can lie to the user/programmer (never do that on purpose). It's good to use readable variable names. Sometimes it's better to divide your code into named pieces (later we will call these pieces functions). In some situations it's a good idea to write steps of calcutions in a clearer way. The code below contains some of these situations, try to improve it (and remove comments - sometimes removing comments will be improvement itself).

```
#include <iostream>//we included iostream
#include <iomanip>//we included iomanip
#include <string>//we included string

int main()
{
  int v=10800; // 3*60*60

  int zzz=3*60;// It is a variable to hold the value of 3 minutes in seconds

  int zzz=5*60;// It is a variable to hold the value of 6 minutes in seconds

  float siii=3.141526; //it's the value of pi


  //std::cout << "result:  " << result <<  std::endl;
  // here we should print the v value but a programmer didn't have time to write code
  // but he/she likes writing long comments
}
```

SCOPE AND SEQUENCE

CURRICULUM OBJECTIVES

The aim of the course is to:

- familiarize the student with the universal concepts of computer programming,
- present the syntax, semantics and basic data types of the C++ language, discuss the principles of the object-oriented model and its implementation in the C++ language, and demonstrate the means to resolve typical implementation problems with the help of standard C++ language libraries,
- align the course to the C++ Institute *CPA – C++ Certified Associate Programmer certification*.

COURSE OBJECTIVES

During the course, students will study the following objectives:

- Introduction to compiling and software development,
- Basic scalar data types, operators, flow control, streamed input/output, conversions,
- Declaring, defining and invoking functions, function overloading,
- Data aggregates,

- Strings processing, exceptions handling, dealing with namespaces,
- Object-oriented approach and its vocabulary,
- Dealing with classes and objects, class hierarchy and inheritance,
- Defining overloaded operators, self-defined operators, exceptions,
- Fundamentals of STL.

COURSE OUTLINE

0 – Installing and using your programming environment	• introduction to compiling and software development.
1 – Introduction to computer programming	• machine and high-level programming languages, compilation process, • obtaining machine code: compilation process, • writing simple programs, • variables, • integers: values, literals, operators, • characters: values, literals, operators, • dealing with streams and basic input/output operations.

COURSE OUTLINE (cont.)

2 – Advanced flow control and data aggregates	• how to control the flow of the program, • floating point types: values, literals, operators, • more integer types: values and literals, • loops and controlling loop execution, • logic, bitwise and arithmetic operators, • structures.
3 – Extending expressive power: pointers, functions and memory	• pointers, pointers vs arrays, • functions, declaring and invoking functions, side effects, • different methods of passing parameters and their purpose, default parameters, • inline functions, overloaded functions, • sorting, memory on demand.

COURSE OUTLINE (cont.)

4 – Accessing different kinds of data	• converting values of different types, • strings: declarations, initializations, assignments, • the string as an example of an object: introducing methods and properties, • namespaces: using and declaring, • dealing with exceptions.
5 – Object programming essentials	• class, objects, class components, • constructors, • referring to objects, • static members, • classes and their friends.

COURSE OUTLINE (cont.)

6 – Inheritance	• base class, superclass, subclass, • inheritance: how it works, • types of inheritance, • inheriting different class components, • multiple inheritance, • polymorphism: notion and purpose, • virtual methods: declaration and usage, • inheriting virtual methods, • abstraction and abstract classes.

COURSE OUTLINE (cont.)

7 – Exceptions	• what is an exception, • catching and throwing exceptions, • different classes exceptions and hierarchies, • defining your own exceptions.
8 – Operators and enumerated types	• defining and overloading operators, • using operators with complex classes, • enumerated types.

HOW TO USE THE COURSE

ACADEMIC INSTITUTIONS

Academic institutions can use this course as follows:

- offer the course as a complete full semester course
- create interest and motivate new students to learn the fundamentals of computer programming
- motivate those students who already know another programming language to learn C++
- supplement an existing C++ course
- help students prepare for the *CPA – C++ Certified Associate Programmer certification*
- introduce Netacad.com to your peers and colleagues

There are **no formal requirements for instructors** to teach *CPA: Programming Essentials in C++*. However, the C++ Institute recommends that instructors earn a *CPA – C++ Certified Associate Programmer Certification* prior to teaching the class.

HOW TO SET UP A CLASS IN Netacad.com

1. Go to the Netacad.com home page at www.netacad.com and sign in.
2. Select the **TEACH** tab.
3. Select the **CREATE A COURSE** link.
4. Enter the course information (select an **Academy**, enter a **Course Name** and **Course ID**. Then Select the course **Partner: CPA: Programming Essentials in C++**, choose a language for the course, enter a **start** and **coclude date**, and select the **instructor**.
5. Click **Save** to set up your class.

CERTIFICATION

C++ Certified Associate Programmer (CPA) is a professional certification that measures a test candidate's ability to accomplish coding tasks related to the basics of programming in the C++ language and the fundamental notions and techniques used in object-oriented programming.

- Professional certification
- Associate level
- Delivered through the network of Pearson VUE Test Centers
- Digital transcript, badge, and paper certification
- Complete the *CPA: Programming Essentials in C++* course and get a **51% discount** for the certification exam!

C++ INSTITUTE CERTIFICATION – SURVEY RESULTS

83% of respondents said that obtaining a C++ Institute certification had directly translated into receiving some career benefit.

62% of respondents said that obtaining a C++ Institute certification had a positive impact on professional image and reputation.

49% of respondents said that earning a C++ Institute certification had helped them to do their job more confidently.

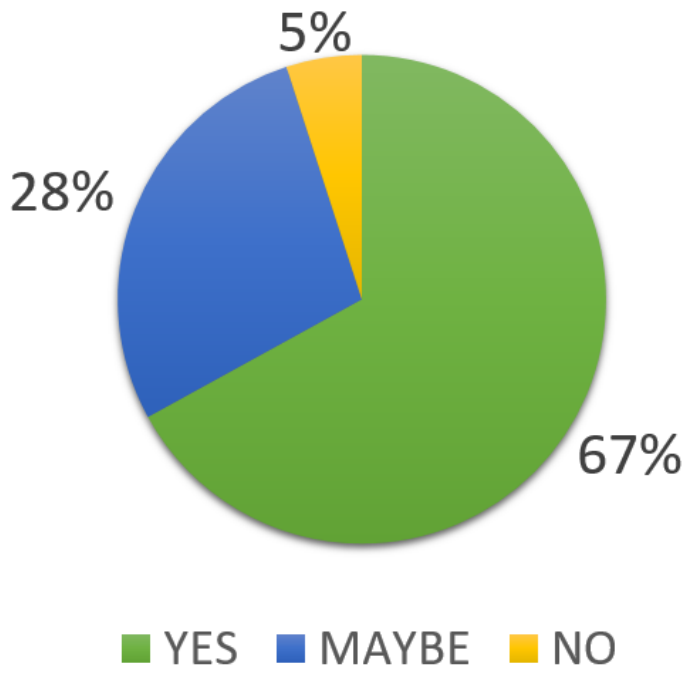
41% of respondents claimed that *learning new things* was the biggest benefit from obtaining a C++ Institute certification.

75% of respondents said they had experienced the first benefit of obtaining a C++ Institute certification within 3 months.

Source: *The 2016 Value of C++ Institute Certification Report* based on a survey conducted online from June 22 to June 24, 2016, among C/C++ certified individuals. The survey was conducted by Fundacja IT and C++ Institute.

C++ INSTITUTE CERTIFICATION – SURVEY RESULTS

Would you recommend C++ Institute certification to your friends or colleagues when discussing a career or advancement in IT/programming?



78% of respondents said that using C++ Institute study materials had **increased confidence to take the certification exam** and had **played a role in passing the certification exam**.

73% of respondents said they wanted to take another C++ Institute exam in the future.

Source: *The 2016 Value of C++ Institute Certification Report* based on a survey conducted online from June 22 to June 24, 2016, among C/C++ certified individuals. The survey was conducted by Fundacja IT and C++ Institute.

WHY LEARN PROGRAMMING

FOR SEVERAL REASONS.

- To become a **creator**: a highly **creative** and **powerful** one. Go as far as your imagination lets you.
- Strong programming skills are a **hot commodity** on the job market!
- Boost your **earning potential**!
- Programming is the language of the **future**.
- Learning to program means learning to **think in abstract** and more **precise ways**.
- It will help you **do better in other areas**!
- It will be **fun**!

WHY LEARN C++

FOR SEVERAL REASONS.

- It is **omnipresent**, people use numerous C++ powered devices on a daily basis, whether they realize it or not.
- There have been millions (well, actually **billions**) of lines of code written in C++, which means almost unlimited opportunities for **code reuse** and learning from well-crafted examples.
- It is **simple**, **readable**, and **flexible.**
- It has been the **backbone** of a number of other languages (e.g. Java derives much of its syntax from C/C++).
- It is **versatile**, **portable**, and **fast.**
- There is a **large** and very **active C/C++ community.**
- It will give you a **solid foundation** and allow you to learn other programming languages much easier and much faster.
- It will be **fun**!

C/C++ EXAMPLES

DID YOU KNOW...?

Do you remember **Doom III**, **StarCraft**, **Master of Orion III**, or **Warcraft III**? You have probably played (or at least heard of) **Diablo I** or **Diablo II**?

If you like computer games, then you must have heard of **Electronic Arts**. All of these games have to do with C/C++ programming.

The truth is that a large majority of **computer games and game engines** have been developed in C/C++. Electronic Arts' video game engine and (probably) all **Microsoft games** are no exception.

Not surprisingly, most of the **operating systems** are written in the C/C++ languages. These not only include Windows and Linux (the Linux kernel is almost entirely written in C), but also Google Chrome OS, RIM Blackberry OS 4.x, Symbian OS, Apple Mac OS X, iPAD OS, Apple iPhone iPod Touch, and Cisco IOS (which is mainly comprised of compiled C and C++ code).

Think of **Internet Browsers** like Microsoft Internet Explorer, Google Chrome, Mozilla Firefox, Safari, Netscape Navigator and Opera. Yes, they all, too, were developed in C/C++.

And what about the major **websites**? Google? Facebook? Twitter? YouTube? Amazon? PayPal?
Yes. They were all written, to a greater or lesser extent, in C/C++. Other examples?

- **Microsoft Office products** (Word, Excel, Access, PowerPoint, etc.)
- **e-mail clients** (Microsoft Outlook, Mozilla Thunderbird, IBM Lotus)
- **Multimedia players** (Winamp, Windows Media Player, VLC media player, and Apple iPod software)
- **Database systems** (Oracle database, MySQL, IBM DB2, Microsoft SQL Server, IBM Informix, SAP DB/MaxDB, and MongoDB)
- **Graphical User Interface** (Microsoft Windows UI, Apple MacOS UI - Aqua, and KDE)
- **compilers and virtual machines** for programming languages (such as Microsoft Visual C++ Compiler, Microsoft Visual Basic Compiler, Microsoft Visual C# Compiler, Microsoft .NET CLR, or Java Virtual Machine – JVM)
- and **thousands of other examples** including: Sun Microsystem's compilers, Solaris OS, Google File System, Google Earth and Picasa, Adobe's Photoshop, Illustrator, Acrobat Reader, InDesign, Intel's chip design and manufacturing software, IBM's OS/400 and K42, Microsoft's DirectX, Exchange Server, and Visual Studio, CERN data analysis applications, Bloomberg, Autodesk's applications, e.g. Autodesk Maya, 12D, Vodaphone infrastructure, and FlightGear...

Source: http://www.stroustrup.com/applications.html

KEY TAKEAWAYS

- *CPA: Programming Essentials in C++* is developed by the C++ Institute
- The course introduces your students to computer programming using the C++ language
- The course aligns to the *C++ Institute CPA – C++ Certified Associate Programmer* certification
- The C++ Institute provides all content
- The course is available in Netacad.com
- Students who successfully complete the course and pass the final test will receive a 51% discount for the *CPA – C++ Certified Associate Programmer* certification exam at Pearson VUE