CPA: Programming Essentials in C++ (2.0) (aka C++ Essentials)

Scope and Sequence

Last updated: May 5, 2021

Scope and Sequence – Table of Contents

1. Target Audience
2. Prerequisites
3. Industry Certification
4. Curriculum Description
5. Curriculum Objectives
6. Course Structure
7. Minimum System Requirements
8. Statement of Achievement and PCAP certification discount

Target Audience

The **CPA: Programming Essentials in C++ (2.0)** curriculum (short: **C++ Essentials**) is designed for students with little or no prior knowledge of programming; students of K12 program, secondary school, college, university, vocational school, and simply anyone interested in learning the fundamentals of programming using the C++ language.

Prerequisites

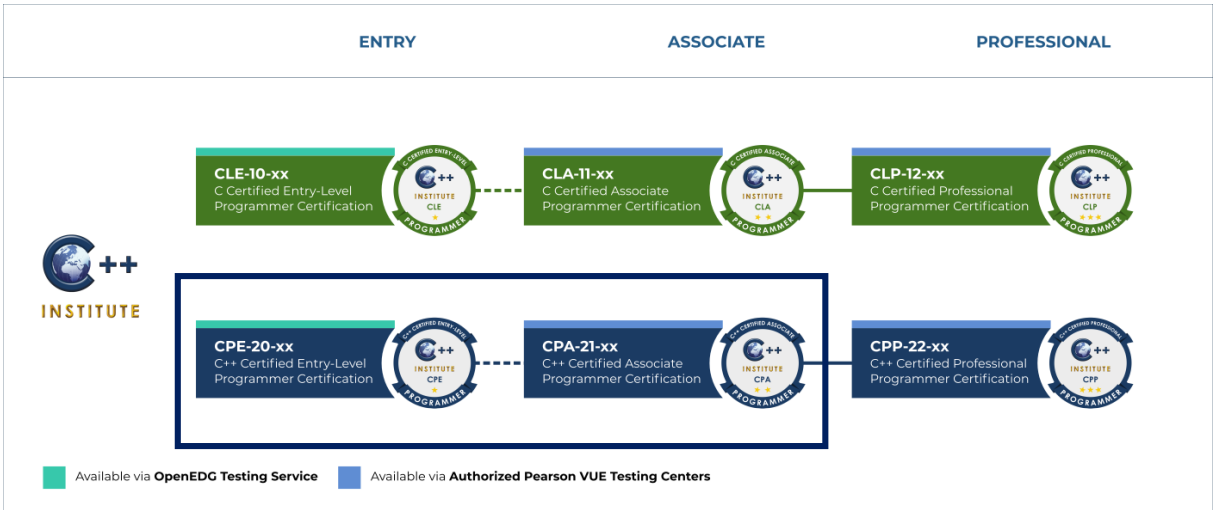
There are no prerequisites for this course. The only preliminary requirement is the ability to use a personal computer and basic knowledge in mathematics (e.g., basic arithmetic operations such as counting, addition, subtraction, multiplication, division, and exponentiation; order of operations, and numeral systems).

Industry Certification

The *CPA: Programming Essentials in C++ (2.0)* curriculum helps students prepare for the *CPE – C++ Certified Entry-Level Programmer* (C++ Essentials Part 1 – CPPE1: Modules 1-4) and *CPA – C++ Certified Associate Programmer* (C++ Essentials Part 2 – CPPE2: Modules 1-4) certification exams:

CPE – C++ Certified Entry-Level Programmer certification shows that you are familiar with universal computer programming concepts like compilation, variables, data types, typecasting, operators, conditional execution, loops, arrays, pointers, structures, and the runtime environment. CPE certification is an interim step to the *CPA – C++ Certified Associate Programmer* certification and the starting point to launch a career in software development, low-level and middle-level programming, C++ programming, and related technologies. Becoming CPE certified will help you stand out from other candidates and get your foot in the door.

CPA – C++ Certified Associate Programmer 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 – C++ Certified Associate Programmer* certification shows that you are familiar with the universal concepts of computer programming; the syntax and semantics of the C++ language as well as basic data types offered by the language; the principles of the object-oriented model and its implementation in the C++ language; the means useful in resolving typical implementation problems with the help of standard C++ language libraries.



Curriculum Description

This course covers all the most important universal aspects of computer programming, 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 focusing on the objective approach, and gradually extends to the more advanced issues the student will encounter when using the OOP approach.

The course is broken down into two parts (CPPE1 and CPPE2), each divided into four modules. 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.

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.

During the course, students will cover the following study material:

- 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, namespaces;
- Object-oriented approach and its vocabulary;
- Classes and objects, class hierarchy and inheritance;
- Defining overloaded operators, self-defined operators, exceptions;
- Fundamentals of STL.

The student who completes **C++ Essential Part 1** (CPPE1):

- knows the basics of the C++ data types system and is able to choose a type adequate to their needs;
- thinks algorithmically and can analyze a problem using programmatic and conceptual apparatus;
- can design, develop, and improve simple C++ procedural programs;
- can choose a data type adequate to the problem being solved and use suitable flow control means;
- understands a programmer's work in the software development process and the role of fundamental development tools;
- knows how a program is compiled and executed in an actual computer environment;
- can create and develop their own programming portfolio.

The student who completes **C++ Essential Part 2** (CPPE2):

- is familiar with the fundamental concepts of OOP and their C++ implementation;
- thinks algorithmically and can analyze and model a problem using an objective conceptual apparatus;
- successfully develops and builds class hierarchies adequate to the problem being solved;
- designs, develops, and improves C++ programs using classes and objects, and constructs suitable inheritance paths;
- understands a programmer's work in the software development process and the role of fundamental development tools;

- knows the object life cycle and understands how classes and objects function in an actual computer environment;
- can create and develop their own programming portfolio.

Course Structure

The *CPA: Programming Essentials in C++* course is broken down into two parts (C++ Essentials Part 1 – CPPE1, and C++ Essentials Part 2 – CPPE2), each divided into four modules (and an additional Module 0 in part 1).

C++ Essentials Part 1 (CPPE1)

CPPE1: Module 0
Installing and using your programming environment

- What is an IDE?
- How to choose an IDE.
- Edube Interactive® (EI) – an online alternative to a standalone IDE.

CPPE1: Module 1
Introduction to Computer Programming

- Natural language vs. programming language, elements of a language.
- Machine language, compiler and compilation, linker and linking.
- "Hello world" – your first C++ program.
- Integer, floats, and their literals.
- Declaring variables – types, names, and initializations.
- C++ keywords – what are they for?
- Comments in C++.
- Arithmetic operators, priorities and bindings, building expressions, using shortcuts.
- The *char* type, fundamentals of ASCII encoding, character literals, escape characters.
- Conditional operators, the *bool* type and its literals.
- Taking control over the execution flow – the *if-else* instruction.
- Basic input/output facilities: *cin* and *cout* streams, manipulators.
- Basic data type conversions and the *static_cast* operator.

CPPE1: Module 2
Advanced flow control and data aggregates

- Extending *if* instruction capabilities with the *else* clause, nesting conditional instructions.
- More int and float data types: *long*, *short*, *unsigned*, *double*, and *long double*.
- Numerical anomalies – why floats can be so untrustworthy.
- Loops – why do we need them and how do they simplify a programmer's life?
- The *while, do* and *for* loops – similarities and differences.
- Some simple algorithms and their implementations.
- Controlling the execution flow inside loops – the *break* and *continue* instructions.
- Computer logic basics, bitwise and logical operators, dealing with a single bit's state.
- Branching the execution flow: the *switch-case-default* instruction as an alternative to the *if-else* cascade.
- Vectors – when do we need them and why can't we live without them?
- Declaring, initializing and using one-dimensional vectors, the indexing operator, basic vector methods.
- The *vector* template vs. obsolete array declarations.
- Multidimensional arrays – declaring, initializing, and indexing.
- Structures, their concept, role, and sphere of application.
- Declaring, initializing and using structures, building vectors of structures.

CPPE1: Module 3
Extending expressive power: pointers, functions, and memory

- C++ pointers: the concept, role, and purpose.
- Declaring, initializing, and assigning pointers, the *nullptr* symbol and its role.
- The reference, dereference and *sizeof* operators.
- Pointers vs. vectors ° similarities and differences.
- The arithmetic of pointers, comparing pointers.
- The function – what it is and how functions help the programmer to keep the code clean.
- Declaring, defining, and invoking functions – parameters vs. arguments and results, the *void* type and its purpose, side effects.
- Passing parameter conventions: by value, by reference, by pointer.
- Default parameters: declaring and using.
- Function prologue and epilogue, function inlining, inline functions, overloaded functions.
- Writing and testing functions of varying complexity.
- The ternary operator and its usage.
- The *Bubble sort* algorithm and its sample C++ implementation.
- Memory on demand – the *new* and *delete* keywords.

CPPE1: Module 4
Accessing different kind of data

- Arrays of pointers: their purpose, declaring, initializing, and using.
- Automatic vs. implicit data type conversions – typecasting, accuracy, precision, and promotion.
- C++ strings – concept and implementation, string literals.
- Declaring and using strings: required includes, initializations, string operators, reading strings from *cin*.
- Comparing strings using built-in C++ operators and with the help of strings member functions.
- Dealing with strings: extracting sub-strings, determining the string's length and actual size, comparing substrings, finding sub-strings inside strings, controlling the string's size, appending a string/character to a string, inserting a string into a string, etc.
- C++ namespaces: concept, role, and purpose.
- Introducing, using, expanding, and renaming user-defined namespaces.
- Anonymous vs. named namespaces.

C++ Essentials Part 2 (CPPE2)

CPPE2: Module 1
The essentials of Object-Oriented Programming (OOP)

- Basic concepts of OOP: identifying objects in real life, classes, subclasses and superclasses, inheritance, class objects, object attributes and activities.
- Procedural vs. object approach: how the objective approach neutralizes some of procedural threats?
- Implementing a sample integer stack using procedural and object approaches in order to demonstrate the main OOP techniques and tricks.
- Anatomy of the class: class components, behaviors, and traits.

CPPE2: Module 2
Inheritance

- Class hierarchies: classes, subclasses, and superclasses, class diagrams, overriding.
- How inheritance works and how to express interdependencies of classes in C++ code.

- Polymorphism – concept and implementations in C++.
- Class constructors and destructors – defining and using, default as well as implicit and explicit copying constructors.
- Access specifiers vs. inheritance.
- Virtual and non-virtual methods.
- Passing objects as function arguments.
- Static class components and how to access them.
- Objective dynamic typecasting – *static_cast* vs. *dynamic_cast* operators.
- The *const* keyword and how it affects class behavior – const variables, structures, pointers, member function parameters and results, class fields.
- The *friend* keyword – friend classes and functions in C++.

CPPE2: Module 3
Exceptions

- Introduction to exceptions: concept and rationale, required includes.
- The *try-catch* instruction, its variants and behavior, handling different classes of exceptions.
- The *throw* instruction, creating exception objects, the *noexcept* clause.
- The explicit specifier and how it affects class behavior.
- C++ exception categories and hierarchies, distributing exception handling, branching and ordering catches, the *what()* function.
- Defining, throwing, catching, and using user-defined exceptions.

CPPE2: Module 4
Operators and enumerated types

- Overloading operators in C++ – concept and limitations.
- Introducing user-defined operators into C++ code: syntax conventions and operator behavior.
- Possible techniques for operator overloading as well as prohibited practices.
- Defining a sample set of operators for the LIFO class.
- An overview of overloadable C++ unary and binary operators.
- Implementing a sample fraction class with a set of dedicated operators.
- C++ enumerated types: the idea and its purpose.
- Declaring user-defined enumerated types and how to deal with them.

Minimum System Requirements

Using the courseware does not require the installation of any software applications – it is possible to use a dedicated, interactive on-line programming environment (Edube Interactive) that allows C++ code to be run in an Internet browser. Edube Interactive is a tool integrated with the course, which can be used as a browser-based C++ sandbox that allows you to test the code discussed throughout the course, as well as a compiler that enables you to launch, perform, and test lab exercises.

The course content modules, labs, quizzes and assessments can be accessed online through any Internet browser. For the best learning experience, we recommend using the most recent versions of Mozilla Firefox, Microsoft Edge, Google Chrome, Safari, or Opera.

Recommended equipment and technical requirements to use Edube Interactive:

- a desktop computer or a laptop (recommended: a desktop computer with a mouse/pointing device and keyboard)
- minimum RAM: 1 GB or more;
- minimum processor: 1.0 GHz or more;
- the most recent version of Mozilla Firefox, Microsoft Edge, Google Chrome, Safari, Opera (preferred: Google Chrome)

- JavaScript enabled in your browser (mandatory requirement)
- a fast and stable Internet connection (recommended Internet download speed: 1.0 Mbps or higher; recommended Internet upload speed: 0.5 Mbps or higher)
- a color monitor, minimum screen resolution: 640 by 480 pixels (recommended: 1024 by 768 pixels)
- Windows 7/8/10 OS, MacOS X 10.0x or newer, Linux OS;
- whitelist the domains "*.edube.org" and "*.openedg.org"
- full access through ports 80 (http), 443 (https), and http redirects permitted.

Statement of Achievement and PCAP certification discount

A Statement of Achievement will be issued to participants who successfully complete the *CPA: Programming Essentials in C++ (2.0)* course. The Statement of Achievement will acknowledge that the individual has completed the course and is now ready to attempt the qualification *CPA – C++ Certified Associate Programmer*, taken through Pearson VUE computer-based testing/OnVUE Online Proctoring from Pearson VUE, at a 50% discount.

To receive the Statement of Achievement, instructors must mark the student as having successfully passed the course.

For additional information about the *C++ Institute CPA – C++ Certified Associate Programmer* certification, please visit www.cppinstitute.org/certification.