CLP: Advanced Programming in C

Scope and Sequence

Table of Contents

Target Audience

The *CLP: Advanced Programming in C* curriculum is designed for students who already have a good knowledge of the C language, students of secondary school, university, vocational schools or simply anyone interested in deepening programming skills. The requirement is good knowledge of C language programming as well as ability to effectively use MS Windows and Linux/Unix OS environments.

Prerequisites

There are no formal prerequisites for this course. However, it is recommended that the student complete the *CLA: Programming Essentials in C* course prior to signing up for the *CLP: Advanced Programming in C* course.

Target Certification

The *CLP: Advanced Programming in C* curriculum helps students prepare for the CLP – C Certified Professional Programmer certification exam.

C Certified Professional Programmer (CLP) is a professional certification that measures the student's ability to accomplish coding tasks related to the syntax and semantics of the C language, as well as advanced data types offered by the language, advanced libraries, the universal concepts of computer programming and developer tools, ability to identify code bugs and bottlenecks, programming of advanced data structures, solving non-trivial problems with the use of data structures and algorithms, designing and writing programs using standard language infrastructure regardless of the hardware or software platform.

Curriculum Description

The course is broken down into eight modules:

- Module 1: Evolution of C (e.g., new C11 keywords, trigraphs and digraphs)
- Module 2: Handling variable number of parameters (<stdarg.h>)
- Module 3: Memory and strings (<string.h> et al.)
- Module 4: Internationalization I18N.
- Module 5: Processes and threads.
- Module 6: Floats and ints once again (<math.h>, <fenv.h>, <inttypes.h> et al.)
- Module 7: Network sockets – absolute basics.
- Module 8: Miscellaneous (eg. portability issues and undefined behaviours, const variables vs. volatile variables)

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

Taking the course gives the student opportunity to dive into advanced computer programming connected not only with just C language but also with some aspect of system and network programming. The main goal of the course is to guide the student from the state of intermediate/good

C language knowledge to a level of conscious and advanced programmer able to cope with problems appearing in different OS platforms and execution environments.

The aim of the course is to:

- familiarize the student with advanced programming techniques,
- present the coding and design tasks related to advanced topics of the C programming language,
- discuss the advanced libraries, programming of advanced data structures, solving non-trivial problems with the use of data structures and algorithms,
- teach the student to design and write programs using standard language infrastructure regardless of the hardware or software platform in the C programming,
- align the course to the C++ Institute CLP – C Certified Professional Programmer certification.

The key features and benefits of the course are:

- Develop deep knowledge of C language evolution and changing capabilities
- Develop basic skill of system and network programming
- Introducing advanced techniques not covered by CLA course
- Improve ability to effectively use C language as a low-level programming Swiss army knife

The course objectives are:

- Familiarize the student with evolution of the C programming language including presentation of main development milestones (ANSI C, C89, C95, C99, C11) as well as some obsolete but still valid elements like digraphs/trigraphs and older ways of declaring functions.
- Demonstrate new keywords introduced by C11.
- Familiarize the student with techniques used by compilers and runtime environments to handle argument passing and assigning arguments to function's parameters including issues connected to stack usage, stack frame role, different calling conventions and their presence in the C language as well as recursion and returning function's result.
- Introduction to the facilities offered by stdarg.h mechanisms and detailed description of function and macros needed to write and use variadic functions.
- Making the student acquainted with notions of API and ABI as well as POSIX and WINAPI conventions.
- Common set of functions for handling low level I/O operations is presented and important differences between POSIX and WINAPI are pointed and discussed.
- Presentation of means used to creating, writing, reading, re-positioning, removing, linking and unlinking files.
- Familiarize the student with the C language's facilities designed for handling memory blocks as well as strings. Presenting the way used by qsort() and bsearch() functions to compare elements of virtually any type.
- Showing the means used to copy, fill, zero and compare memory blocks and strings.
- Introduction to internationalization issues (I18N) including UNICODE, UCS and UTF-8 notions as well as "wide" versions of routine string functions.
- Familiarize the student with fundamental concepts of multi-processed and multi-threaded programming.

- Discussing differences between processes and threads as well as demonstration of techniques offered by Unix/Linux/POSIX, MS Windows OS environments and built-in means introduced by C11 standard.
- Making the student acquainted with hardware formats used by floating-point values including IEEE-754 standard.
- Presentation of some important numerical anomalies including the ULP concept and its importance in assessing evaluation accuracy.
- Discussing the floating point exceptions and rounding modes.
- Short introduction into GMP multi-precision library.
- Making the student acquainted with the notion of network sockets and basics of building and using network connections using TCP and UDP protocols.
- Server-client and peer-to-peer communication models are discussed and illustrated by code samples.
- Differences between traditional BDS/POSIX sockets and WinSock implementation are shown and discussed.
- Familiarize the student with some details of specific C language issues as const and volatile variables, local (goto) and non-local (longjmp) jumps, static array indices, designated initializers, compound literals, variable-length arrays, flexible array members and restrict keyword.
- Introducing the notion of sequence point.
- Introducing mixed-programming technique using asm keyword as well as describing portability issues and undefined behaviors appearing in different C language implementations.

Course Outline

Learning Module	CLP – C Certified Professional Programmer **Certification Objectives Covered**
1 – Evolution of C – from past to eternity	• milestones: ANSI C, C89, C95, C99, C11 • obsolete (but still valid) language elements • how function declaration changed over time? • trigraphs and digraphs • new C11 keywords: ○ __Noreturn ○ _Alignof and _Alignas ○ _Bool ○ _Exit ○ _Complex ○ _Pragma, _ _func_ _ ○ _Generic
2 – Handling variable number of parameters (<stdarg.h>)	• calling conventions, parameters passing, stack usage, stack frame, returning a value, recursion • va_start() • va_arg() • va_end() • va_copy() • vsprintf(), vprintf(), vfprintf() • vscanf(), vsscanf(), vfscanf() • _ _VA_ARGS_ _

3 – Low level IO (<unistd.h>)	• POSIX, API, ABI, WINAPI, etc. • access() • open() • errno • close() • read() • write() • lseek() • dprintf() • stat() • symlink(), link() • readlink() • unlink() • fcntl() and ioctl()
4 – Memory and strings (<string.h> et al.)	• manipulating memory blocks • string manipulation: strchr(), strrchr(), strstr(), strtok() • qsort(), bsearch() • aligned_alloc(), calloc(), malloc(), and realloc() • bcopy() • memcpy() • memccpy() • memmove() • bzero() • memset() • memcmp() • Internationalization I18N ○ UCS, UTF-8 - how to deal with multilingual environment? ○ universal character names ○ wide characters support in different C dialects (e.g., <wchar.h>, <wctype.h>) ○ strcoll() and wcscoll()
5 – Processes and threads	• definitions, implementations and history • thread safety • system(), getenv(), setenv() • processes in Unix way: ○ fork() ○ exit() ○ execxx() ○ wait() and waitpid() • processes in MS Windows way: ○ CreateProcess() ○ WaitForSingleObject() • POSIX threads • MS Windows threads • C11 threads (<thread.h>)
6 – Floats and ints once again (<math.h>, <fenv.h>, <inttypes.h> et al.)	• IEEE-754: a different universe • NaN, infinity, zero • floats and doubles – should we trust them? • numerical anomalies vs precision • ULP • what is pragma? • FENV_ACCESS pragma • floating-point exceptions • rounding • multi-precision libraries (GMP, MPFR, MPIR)

7 – Network sockets – absolute basics	• what is a socket? what is a network socket? • TCP/IP protocol stack, UDP • connection and connectionless transmissions • servers and clients • big and little endians and why you should be aware of them • socket addressing: IP4, IP6, service numbers • getaddrinfo() • socket() • connect() • bind() • listen() • accept() • send() and recv() • simple example of client-server communication • simple example of peer-to-peer communication
8 – Miscellaneous	• const variables vs. volatile variables • goto – why and why not, advantages, disadvantages and limitations • long (non-local) jumps: setjmp() and longjmp() • static array indices, designated initializers, compound literals, variable-length arrays, flexible array members, restrict keyword • sequence points: why ++/-- may sometimes make you crazy? • the asm keyword • portability issues and undefined behaviours

Minimum System Requirements

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, or Google Chrome.

Industry certification

The course curriculum helps students prepare for the C++ Institute CLP – C Certified Professional Programmer certification.

A Statement of Achievement will be issued to participants who successfully complete the *CLP: Advanced Programming in C* course. The Statement of Achievement will acknowledge that the individual has completed the course and is now ready to attempt the qualification *CLP – C Certified Professional Programmer* certification, taken through Pearson VUE computer-based testing, at a 51% 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 *CLP – C Certified Professional Programmer* certification, please visit www.cppinstitute.org/certification.