Introduction to Programming Concepts

Duration: 7 hours

Prerequisites: Rudimentary knowledge of computer systems. Students should know how to basically operate a computer, browse the world wide web, send and receive email messages, and write and print memos.

Course Description: Courses that teach programming languages tend to focus on the particulars of each language (as they should), expending only minimal time discussing basic universal programming concepts. Students who are new to programming may find themselves engrossed in discussions of the details of matters such as control flow operators, executable images, third generation languages, and logical operators before they really understand the concepts. This situation can leave the student lost or frustrated or both.

This course is offered for the aspiring programmer so that they will never be left behind for want of having the foundational knowledge that is necessary to learn and advance in a programming career. Students will be exposed to fundamental concepts that are utilized by all programming languages in this course.

Students Will Learn

- Basic computer terminology and jargon
- Programming tasks: Analysis and requirements gathering
- Programming tasks: Design and planning
- Programming tasks: Coding
- Programming tasks: Testing
- Programming tasks: Deployment/Documentation
- Programming tasks: Maintenance
- The role of machine code
- The role of assembly language
- The utility of 1st through 5th Generation Programming
- The nature, pros and cons of compiled vs. interpreted languages
- The steps of compiling: parsing, optimization, object code, linking, executable images
- How to write and execute interpreted scripts
-
- Storage types: integers, floats, Booleans, characters, arrays, structures, objects, pointers
- Strong and weak typing
- Allocation
- Declaring and initializing variables and constants
- How binary storage affects data typing
- Scope
- Inheritance
- Operators
- Control flow structures
- Algorithms
- Strings and regular expressions
- I/O
- GUIs
- Events
- Frameworks
-

How to write and compile a rudimentary C program

- Programming styles: imperative, structured, procedural, object-oriented, declarative, and functional
- The utility of stand-alone, distributed, client-server and web-enabled programming
- Programming approaches: textual, integrated, visual
- Statements, commands, comments and reserved words
- Routines, subroutines, functions and libraries

Debugging

- Compile time vs. run time errors
- Database programming
- Technical communities
- Benchmarks
- Protocols
- Using the *command prompt* environment to write and compile scripts and programs

Overview

The Essence of Programming

- The language of computers
- Computer components
- Hardware and software
- Operating systems
- Programs

The Tasks of Programmers

- Analysis and requirements gathering
- Design and planning
- Coding
- Testing
- Deployment/Documentation
- Maintenance
- Your job

Types of Programming

- Computer languages
- Machine code
- Early programming
- The evolution of programming languages
- First Generation Languages
- Second Generation Languages
- Third Generation Languages
- Fourth Generation Languages
- Fifth Generation Languages
- Compilers
- The process of compiling
- Practical compiling
- Interpreted languages
- Pros and cons or compiled and interpreted languages
- Imperative programming
- Structured programming
- Procedural programming
- Object-oriented programming
- Declarative programming
- Functional programming
- Stand-alone programs
- Distributed programming
- Client-server programming
- Web-enabled programming

Statements and Storage

- The organization of programs
- Libraries
- Declaring and initializing variables and constants
- Data types: integers, floats, Booleans, characters, arrays, structures, objects, pointers
- Proper data typing
- Allocation
- Static and dynamic data typing
- Strong and weak data typing
- Scope
- Inheritance

- Textual programming
- IDEs
- Visual programming

Operators and Control Flow

- Assignment operators
- Arithmetic operators
- Relational operators
- Conditional operators
- Logical operators
- Bitwise operators
- Special operators
- *if*, *if-then*, *if-then-else* statements
- *case* or *switch-case* statements
- *for* or *foreach* statements
- *while* statements
- *do-while* statements

The Language of Programming

- Algorithms
- Strings
- Regular expressions
- Input/Output
- Graphical user interfaces
- Events
- Frameworks
- Debugging
- Compile time vs. run time errors
- Database programming
- Technical communities
- Benchmarks
- Protocols

The Real World

- Working with the *Command Prompt*
- Customizing your environment
- Help
- Online documentation
- Batch files and scripts
- Installing and using a compiler
- Man pages
- Writing a C program
- Debugging C programs

Related Bootcamps

Track	Duration	Price
Java Programmer	2-course track 3-course track 4-course track 5-course track	$2,400 $3,600 $4,800 $6,000
Advanced Java Developer	4-course track 5-course track 6-course track 7-course track 8-course track	$4,800 $6,000 $7,200 $8,400 $9,600
UNIX Software Developer	3-course track 4-course track	$3,600 $4,800
Linux System Admninistrator	2-course track	$2,400

C/C++ Programmer	2-course track 3-course track 4-course track	$2,400 $3,600 $4,800
Master SQL Server Developer	5-course track 6-course track	$6,000 $7,200
Python Programmer	1-course track	$1,495
Web Developer	4-course track 5-course track 6-course track 7-course track 8-course track	$4,800 $6,000 $7,200 $8,400 $9,600

Contact Us

Address: 1 Village Square, Suite 3 Chelmsford, MA 01824

Phone: 978.250.4983

Mon - Thur: 9 am - 5 pm EST
Fri: 9 am - 4 pm EST

E-mail: info@developer-bootcamp.com