Real Time Group

Real Time & Embedded Linux Solutions

C++ and OOD for RT-Embedded Systems

Course Duration: 90 AH
Type: Hands-on-Training

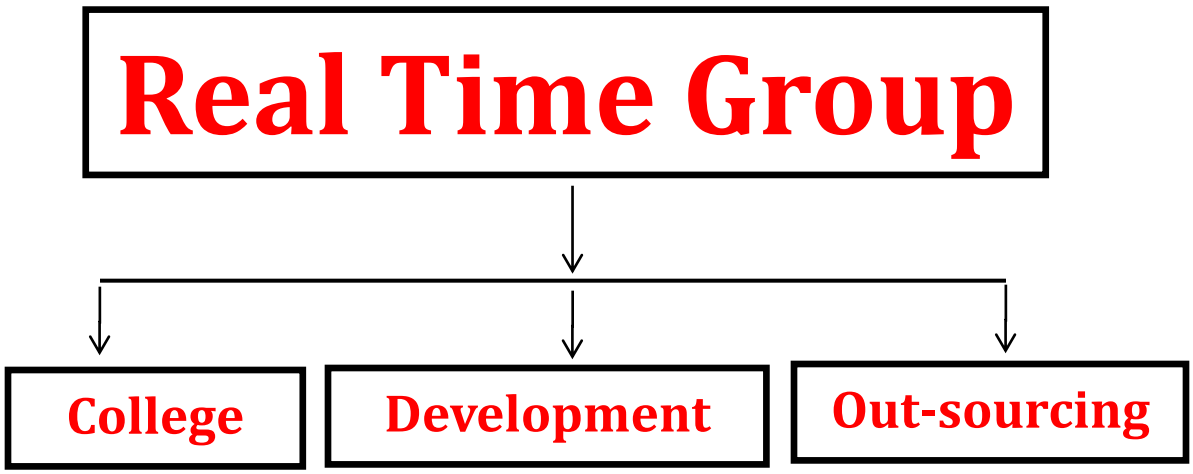
רח' רוזנסקי 14 ראשון לציון טל.077-7067057 / 050-3309318 פקס 077-5067058

www.rt-hr.co.il www.rt-ed.co.il www.rt-dev.com

Real Time Group is a multi-disciplinary dynamic and innovative Real-Time O.S. and Embedded Software Solutions Center, established in 2007.

Providing Bare-Metal and Embedded Linux solutions, professional services and consulting, end-to-end flexible system infrastructure, outsourcing, integration and training services for Hardware, Software and RT-OS \ Embedded Systems.

The company is divided into the following three Divisions:



rt-ed.co.il rt-dev.com rt-hr.co.il

Training Division:

Professional Training Services for Hardware, Software, RT-OS and Embedded systems industries.

We provide the knowledge and experience needed to enable professional engineers to Develop, Integrate and QA Hardware, Software and Networking Projects.

In order to ensure experience, all courses are practical – hands-on-training.
The latest Development, QA and Automation equipment which are adopted by the industry are used.

All students are supplied with Development-Boards for home-work and course projects.

Course Overview:

C++ one of the most used object-oriented and generic programming languages, providing facilities for high level applications, system programming, even used in embedded as well as low-level BSP \ Hardware manipulation.

Using C++ you can create applications that will run on a wide variety of hardware platforms such as personal computers running Windows, Linux, UNIX, and Mac OS X, as well as small form factor hardware such as IoT devices like the Raspberry PI and Arduino–based boards.

This course will emphasize on C++ and OOD from an RT-Embedded point of view, How to increase performance, efficiency and flexibility when using C++.

Who should attend:

- Software engineers who want to learn C++.
- C programmers who need to use C++ for Embedded systems.

Prerequisite:

- The course provides comprehensive hands-on-training to the C++ programming language. We recommend that students have experience \ knowledge in C programming language.
- The curriculum includes building C++ programs with real time constraints and understanding the inner workings of C++ and its effects on system behavior.

C++ and Object-Oriented Design for Embedded Systems (40 AH)

1. **Introduction to Object Oriented Design**
   a. History and evolution, from C to C++
   b. OOD principles
   c. Encapsulation, inheritance, polymorphism
   d. Why and when is C++ better than C.

2. **Basic Classes**
   a. Encapsulation of data and methods
   b. Declaring Classes
   c. Naming Conventions
   d. Defining an Object
   e. Private Versus Public
   f. Public Accessory Methods
   g. Accessing Class Members
   h. Implementing Class Methods
   i. Default arguments
   j. Possible Ambiguities
   k. Private Versus Public classes

3. **Creation and destruction of an Object**
   a. Life of an object
   b. Constructors
   c. Destruction of an Object
   d. Default Constructors \ Destructors
   e. Overloading Constructors
   f. Initializing Objects
   g. The copy assignment operator

4. **Interface Versus Implementation (*.hpp \ *.cpp)**
   a. What are Header files used for
   b. Implementing Header files in C++
   c. Where do libraries fit in ?
   d. What Is a Rreference?

5. **Const members**
   e. Member Functions vs Global Functions
   f. Implementing Class Methods
   g. Const Member Functions
   h. Const Data Members
   a. Const declarations
   b. Constants ver defines
   c. Enumerations

6. **Reference variables**
   a. By val Vs. by reference
   b. References Vs. Pointers
   c. What Can Be Referenced
   d. References usage and Syntax

7. **Dynamic allocation in C++**
   a. The Stack and the Free Store
   b. Creating Objects on the Free Store
   c. The new Operator
   d. new vs malloc
   e. New & Constructors
   f. The delete Operator
   g. Creating & Deleting Arrays on the Free Store
   h. How Avoid \ detect Memory Leaks
   i. const Pointers

8. **Copy Constructors:**
   a. Passing by Reference for Efficiency
   b. Implicit Member Function
   c. The Copy Constructors
   d. The Default Copy Constructors

9. **Working with Streams**
   a. Overview of Streams
   b. Standard I/O Objects & Redirection
   c. Input – Cin
   d. Cin Member Functions
   e. Cin.get() overloaded & Usage
   f. Output: Using cout
   g. cout.put()
   h. cout.write()

10. **Static Variables and Functions**
    a. Static data members
    b. Static member functions
    c. Friend classes
    d. Friend Functions & Classes
    e. The this Pointer

11. **Operator Overloading**
    a. Overloading operator++.
    b. Operator Overloading Unary Operators
    c. Operator Overloading: Binary Operators
    d. The Assignment Operator

12. **Working with Files:**
    a. file open () & close ()
    b. Condition State
    c. R/W from Text Files
    d. Buffers synchronization

13. **Inheritance and Derivation**
    a. Scope and initialization
    b. The Order of construction in Memory
    c. Member Initialization lists
    d. Private Versus Protected
    e. Passing Arguments to Base Constructors
    f. Namespaces

14. **Overriding Functions**
    a. Overloading Versus Overriding
    b. Virtual Methods
    c. Virtual Destructors
    d. Virtual Copy Constructors

15. **Composition and Container Classes**
    a. collections of objects
    b. Template classes
    c. Template functions
    d. vector
    e. Iterators

16. **Polymorphism**
    a. Virtual Inheritance
    b. Virtual destructors
    c. How are virtual function implemented?
    d. Pure virtual functions
    e. Percolating Upward \ Casting Down

17. **Multiple Inheritance**
    a. Inheriting from Shared Base Class
    b. Multiple Inheritance – Introduction.
    c. Constructors in Multiple Inheritance
    d. Ambiguity Resolution in. Multiple Inheritance
    e. virtual from Shared Base Class
    f. virtual Inheritance.
    g. Problems that may occur

18. **Embedded and Real Time C++ Considerations (Based on Time constraints)**
    a. Comparing C and C++ Performance analysis
    b. Implementing Inheritance in C
    c. The Embedded C++ Language Standard
    d. C++ Programing Size Considerations
    e. Pit falls caused by:
        i. Exceptions,
        ii. RTTI, with Templates,
        iii. Multiple Inheritance,
        iv. Operator Overloading
    f. Placing objects at a specific address
    g. Interrupts and interrupt vectors in C++
    h. Combining C and C++ code

19. **Abstract Data Types and Pure Virtual Functions**
    a. How are virtual function implemented?
    b. Abstract Data Types
    c. Pure virtual functions & ADTs
    d. Pure virtual destructors

20. **Dynamic Casting (Based on Time constraints)**
    a. Casting between different classes, Can it be done ?
    b. Run Time Type Identification

21. **Errors and Exceptions**
    a. The Idea Behind Exceptions
    b. Exceptions building blocks
    c. Throw, Catch, Tr Specific Catches
    d. Specific Catches
    e. Standard exceptions
    f. Resource Acquisition Is Initialization or RAII

22. **Advanced topics (Based on Time constraints)**
    a. Smart pointers
    b. Interfacing with C
    c. Inline function mechanism

23. **Basic Design Patterns (Based on Time constraints)**
    a. What Are Design Patterns
    b. UML
    c. Singleton
    d. Abstract Factory
    e. Abstract Method