OCS785	ADVANCED C++ FOR EMBEDDED PROGRAMMING	L	T	P	C
		3	0	0	3

Seventh Semester

UNIT I Operator Overloading and Object Relationships

Introduction to operator overloading - Overloading the arithmetic operators using friend functions - Overloading operators using normal functions - Overloading the 1/0 operators - Overloading operators using member functions - Overloading unary operators +, -, and ! - Overloading the comparison operators - Overloading the increment and decrement operators - Overloading the subscript operator - Overloading the parenthesis operator - Overloading typecasts - The copy constructor - Copy initialization - Converting constructors, explicit, and delete - Overloading the assignment operator - Shallow vs. deep copying - Overloading operators and function templates - Object relationships - Composition - Aggregation - Association - Dependencies - Container classes.

UNIT II Inheritance and Virtual Functions

Introduction to inheritance - Basic inheritance in C++ - Order of construction of derived classes - Constructors and initialization of derived classes - Inheritance and access specifiers - Adding new functionality to a derived class - Calling inherited functions and overriding behavior - Hiding inherited functionality - Multiple inheritance - Pointers and references to the base class of derived objects - Virtual functions and polymorphism - The override and final specifiers, and covariant return types - Virtual destructors, virtual assignment, and overriding virtualization - Early binding and late binding - The virtual table - Pure virtual functions, abstract base classes, and interface classes - Virtual base classes - Object slicing - Dynamic casting

UNIT Ill Templates and Standard Template Libraries

Template classes - Template non-type parameters - Function template specialization - Class template specialization - Partial template specialization - Partial template specialization for pointers - The Standard Library - STL containers overview - STL iterators overview - STL algorithms overview

UNIT IV Multi-threading and Inter Process Communication

Thread creation - Thread operations - Thread Synchronization Mechanism- Basic Inter Process communications - Advanced Inter Process communications

UNIT V Embedded System development

Linux Basics - Make files - Coding in C++ for controlling Rasberry Pi components - Control the GPIO Pins

Total : 45 Periods

Links to learn:

10 Things Every Linux Beginner Should Know I Codementor

Getting Started with C++ on Raspberry Pi (Guide & examples) - RaspberryTips

Makefile Tutorial By Example

ADVANCED C++ FOR EMBEDDED PROGRAMMING INTEGRATED LAB

<table>
<tr><th>S. No.</th><th>Problem Statement / Exercises</th></tr>
<tr><td>1</td><td>Write a program in CPP
a. to create multiple threads and compile the program with lpthread library.
b. to pass any data type in thread callback.</td></tr>
<tr><td>2</td><td>Write a CPP program to simulate decimal number representation with LEDs interfaced with Eight GPIO pins on Raspberry Pi. Example: -
2(10) →0010(2)→ Blink LED connected to Raspberry Pi (GPIOs)
<table>
<tr><th colspan="4">GPIO PINS</th></tr>
<tr><td>4 (GPIO-3)</td><td>3 (GPIO-2)</td><td>2 (GPIO-1)</td><td>1 (GPIO-0)</td></tr>
<tr><td>LED-OFF</td><td>LED-OFF</td><td>LED-ON</td><td>LED-OFF</td></tr>
</table></td></tr>
<tr><td>3</td><td>Write a CPP program to demonstrate mutex lock.</td></tr>
<tr><td>4</td><td>Write a program to create spin locks in user space.</td></tr>
<tr><td>5</td><td>Write a CPP program to create a shared memory and synchronize the threads access.</td></tr>
<tr><td>6</td><td>Write a CPP program to schedule thread execution in specific order.</td></tr>
<tr><td>7</td><td>Write a CPP program to
a. simulate buffer overflow
b. identify the buffer overflow</td></tr>
<tr><td>8</td><td>Write a program to Bit Set/Reset/Complement/Test/Shifting using LEDs interfaced with GIPO pins on Raspberry Pi.</td></tr>
<tr><td>9</td><td>Write CPP program using extern modifier (This needs at least two programs); Use simple Makefile to compile them.</td></tr>
<tr><td>10</td><td>Write a CPP program to handle its own debugging print levels as below
<table>
<tr><th>Debug Information</th><th>Level</th></tr>
<tr><td>INFO</td><td>0</td></tr>
<tr><td>ALERT</td><td>1</td></tr>
<tr><td>WARNING</td><td>2</td></tr>
<tr><td>ERROR</td><td>3</td></tr>
</table>
Example: If we pass input to the program as “0”, it should print INFO print statements alone.
If we pass input to the program as “3”, it should print ERROR statements from the program.</td></tr>
</table>