Programming in Modern C++

Professor Partha Pratim Das
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur
Lecture No. 01
Course Outline

Welcome to Programming in Modern C++. This is our first meeting actually before the regular course module start. I am Professor Partha Pratim Das, from the Department of Computer Science and Engineering at IIT, Kharagpur. And a very passionate developer, designer, moderator, tutor of C++ language and its evolution.

(Refer Slide Time: 01:10)

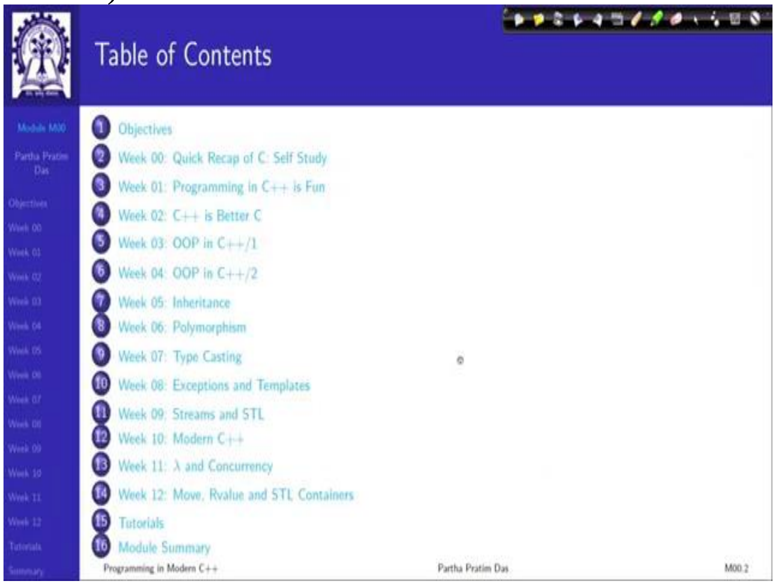


Table of Contents

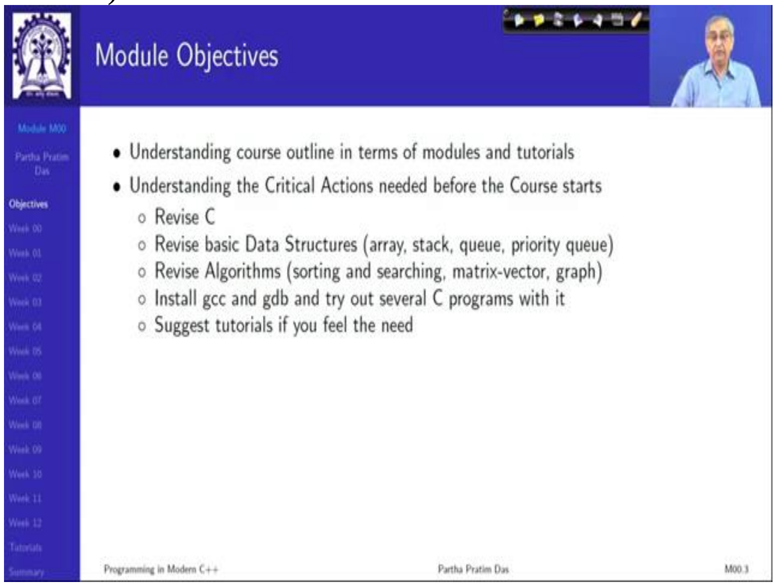
Module M00

Partha Pratim Das

Programming in Modern C++ Partha Pratim Das M00.2

So, welcome you to this course. This particular module, which we call modules 0 for weeks 0 is meant to give you some head start in terms of what the course will be about, and what it will contain. In module 1, when the course starts, we will give you very specifics in terms of the syllabus, in terms of the details of evaluation and all that we are not going to talk about all this. Here, I am just going to tell you primarily, whatever is listed, what is the overall plan.

(Refer Slide Time: 01:49)



Module Objectives

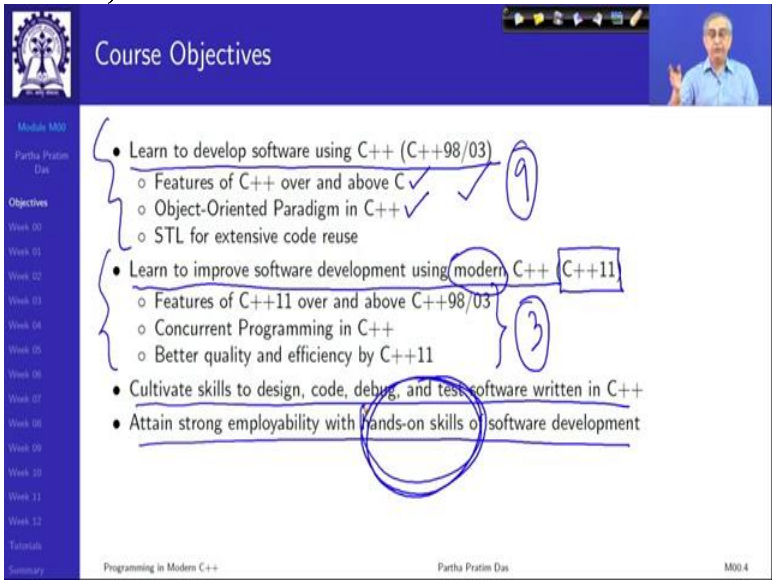
- Understanding course outline in terms of modules and tutorials
- Understanding the Critical Actions needed before the Course starts
  - Revise C
  - Revise basic Data Structures (array, stack, queue, priority queue)
  - Revise Algorithms (sorting and searching, matrix-vector, graph)
  - Install gcc and gdb and try out several C programs with it
  - Suggest tutorials if you feel the need

Programming in Modern C++ Partha Pratim Das M00.3

And the objective for this module is and you will see that this is the style that we follow in every module, every module will start with the objective of what we try to do there and summarize it at the end. So, the objective of this module is to understand the course outline in terms of the several modules and tutorials, that will be available to you through this course.

Further, and that is the key attention area is there are critical actions that you need to take before the course starts. So, before the course starts, how do you prepare, how do you make yourself ready is what we will outline here. There are four primary actions needed for you, revision of C, revision of basic data structure, revision of algorithms, installing the tools for building C and C++ programs. And last, but not the least, if you think that you would love to have some additional tutorial created for your better understanding, please let us know.

(Refer Slide Time: 03:04)



Course Objectives

- Learn to develop software using C++ (C++98/03)
  - Features of C++ over and above C ✓
  - Object-Oriented Paradigm in C++ ✓
  - STL for extensive code reuse
- Learn to improve software development using modern C++ (C++11)
  - Features of C++11 over and above C++98/03
  - Concurrent Programming in C++
  - Better quality and efficiency by C++11
- Cultivate skills to design, code, debug, and test software written in C++
- Attain strong employability with hands-on skills of software development

9

3

Programming in Modern C++ Partha Pratim Das M00.4

Now, what is the objective of this course? C++ is a strong language possibly as of now, it is the strongest programming language ever built. It is certainly with C, it is certainly the most efficient programming language. But it is not the easiest to learn. If you are looking for learning a easy language and quickly getting on to some programming, learn Python. If you want to, just focus on application program development, maybe you should start with Java.

But this C, C++ pair because C++ in a way, partly to a, to a great extent subsumes C gives you very specific skills to do good level of programming in complex systems in terms of building infrastructure software, and is one of the most valued programming skills in the market today. If you compare, the kind of pay packet of developers, skilled in different languages, C, C++ skill will be among the highest.

So, keeping that in mind, naturally, I just do not want to only impart your knowledge, because we are engineers. Knowledge is important for us. But what is even more important is the effective use of that knowledge to create wealth. To create wealth for others, create wealth for ourselves. Which means that we need to talk about practice all the time. And this course is focused to that direction.

So, what specifically you would learn is learn to develop software using C++, it has multiple versions what is been common till about a decade ago and still most people programming that is the C++ 98 or 03 standard. We will give you glimpses of what the standards are and when in the, in the actual module. So, we will even learn about what is the features of C++ on top of C.

So, we expect that you know C, and that is the reason it is critical that you revise C. Then C++ you have heard always that it is object-oriented language. So, what is the object-oriented paradigm which, on which C++ relies? And what is a huge, I should not say huge, very useful libraries that C++ has, which can make the effective use of them can make your programming easy, fast, reusable, reliable and so on.

So, learning the software development with C++ is a key interest area. Over the last two decades, C++ has gone through a lot of further development. And the major version of C++ that people have been talking about for the last 10 years or so, is C++ 11, which particularly is also being called as a modern C++.

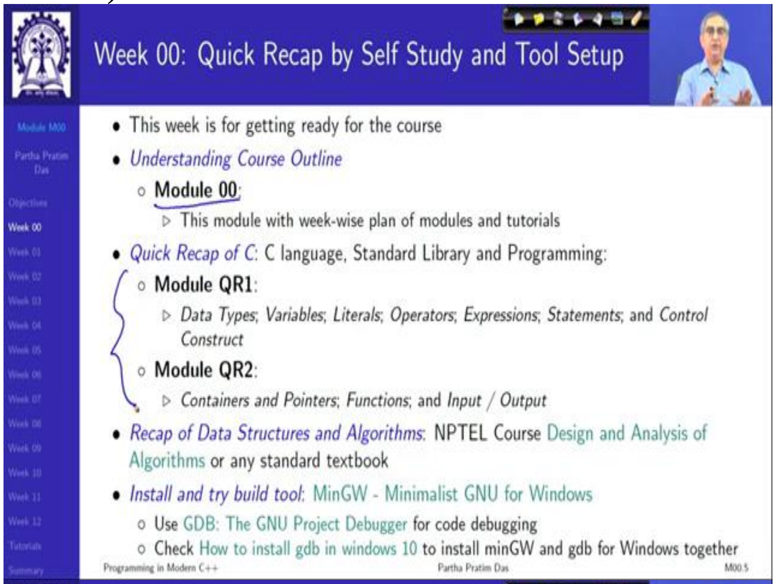
Which has features on top of C++ 98 03. It makes some of the very critical programming aspects directly supported in the language like concurrent programming, you know, parallel programming kind of concurrent programming. There are things like lambda expressions and so on, it provides much better quality and efficiency on top of C++ 03. So, this additionally, we'll also learn about.

So, if you would, you may have noticed that earlier till the semester of, autumn semester of 2021, NPTEL have been offering a course which again, I had created programming in C++, which was only this part, it was a 8 week course. Now, you are going to do a 12-week course, which is programming in modern C++, which we will have this on, which will spend nine weeks and which will have the last three weeks spent on the modern aspects of C++ to really equip you, make you a skilled developer in C++.

It is also important as I started saying that, just knowing the language will not benefit, that knowing the language will not give you a good job. You need to cultivate the skill of design of coding of debugging, testing the software in C++ and so on. Naturally, we cannot cover those in the course. So, I have been creating certain additional tutorials, which are not part of the syllabus, but would help you become an effective engineer of C++ development.

So, if you are inclined towards that, if you want to learn more, if you want to really become a good developer, follow those tutorials well, which are directly linked to several hands on that you should be doing. And this is the final objective, this is the ultimate objective that you must attain strong employability with hands on skills. You can pass the interview by memorizing lot of features of C++, maybe some code segments also. But to be an effective developer, who can contribute he will have to have hands on skills of software development and besides the regular C++ modules, on which based on the syllabus on which the tests will happen, the tutorials will go a long way in helping towards that.

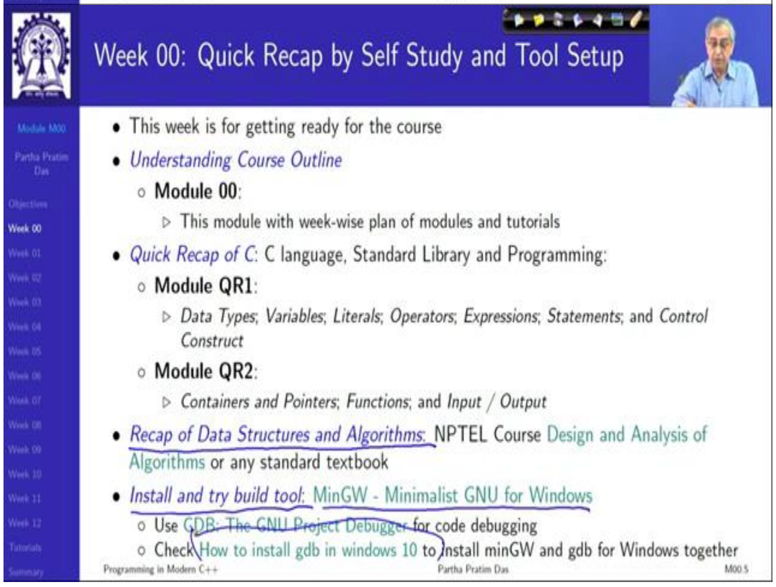
(Refer Slide Time: 10:08)



Week 00: Quick Recap by Self Study and Tool Setup

- This week is for getting ready for the course
- *Understanding Course Outline*
  - **Module 00**:
    - ▷ This module with week-wise plan of modules and tutorials
- *Quick Recap of C*: C language, Standard Library and Programming:
  - **Module QR1**:
    - ▷ *Data Types; Variables; Literals; Operators; Expressions; Statements;* and *Control Construct*
  - **Module QR2**:
    - ▷ *Containers and Pointers; Functions;* and *Input / Output*
- *Recap of Data Structures and Algorithms*: NPTEL Course Design and Analysis of Algorithms or any standard textbook
- *Install and try build tool*: MinGW - Minimalist GNU for Windows
  - Use GDB: The GNU Project Debugger for code debugging
  - Check How to install gdb in windows 10 to install minGW and gdb for Windows together

Programming in Modern C++ Partha Pratim Das M00.5



Week 00: Quick Recap by Self Study and Tool Setup

- This week is for getting ready for the course
- *Understanding Course Outline*
  - **Module 00**:
    - ▷ This module with week-wise plan of modules and tutorials
- *Quick Recap of C*: C language, Standard Library and Programming:
  - **Module QR1**:
    - ▷ *Data Types; Variables; Literals; Operators; Expressions; Statements;* and *Control Construct*
  - **Module QR2**:
    - ▷ *Containers and Pointers; Functions;* and *Input / Output*
- *Recap of Data Structures and Algorithms*: NPTEL Course Design and Analysis of Algorithms or any standard textbook
- *Install and try build tool*: MinGW - Minimalist GNU for Windows
  - Use GDB: The GNU Project Debugger for code debugging
  - Check How to install gdb in windows 10 to install minGW and gdb for Windows together

Programming in Modern C++ Partha Pratim Das M00.5

So, this is about the objective. So, the week zero plan I, as I said, which you will have to drive it by yourself and week zero is not one week, week 0 is the entire time from when you enroll to the time the course actually starts. So, it may be a couple of weeks. So, this is about getting ready. So, Module 0 is what we are discussing here, where you will get the entire plan of modules and tutorials and so on.

But what is most important are you will also get two additional modules, which we call a quick recap module. So, it needs a fair preparedness of C to be able to start understanding the course from day one. So, if your C is kind of got rusted, you have doubts and so on, these are just kind of summarization module which tell you all that you need to know about C. It is, these are not a course on C.

Where the courts on C will again be an eight weeks course, ten weeks course like that. But these are certainly summarizing, the key points about C, which you should go through and ask yourself that do you know. And if you know, can you write? If you can write, can you compile and run? If you can compile and run, can you debug? Can you fix, can you test? So, do that exercise by yourself and get ready.

So, expectation is that when you start attending the C++ modules or going through the C++ modules, you must have got this confidence over C because we will right away start assuming that you know C. Other thing that we need is a lot of these examples used or the problems discussed rely on basic data structures and algorithms. Now, certainly I am not talking about a whole range of data structures.

I am talking about simple ones, like array, stack, queue, priority queue maybe matrix vector a little bit of graph. If you know more, if you know you know different kinds of trees, different kind of all varieties of graph representation then skip list and randomized BST and so on, good. But the minimum is an array, stack, queue priority queue, matrix vector little bit of graph. If you got rusted, please revise from any source.

There is a course on this but certainly you cannot start attending that right away. But you can pick up the common books and do that. Similarly, you must be familiar with the basic algorithms, algorithms for doing simple things like adding a set of numbers, doing different types of sorting, at least few common sorting algorithms like quicksort, merge sort, bubble sort, selection sort this kind of.

And doing some searching like linear search, binary search and so on. These are, this is very minimal requirement, but it will be good to revise them before you get into the course because we will be continuously taking to illustrate how the language can, C++ language can effectively help you in programming over C and independently how we can model different scenarios.

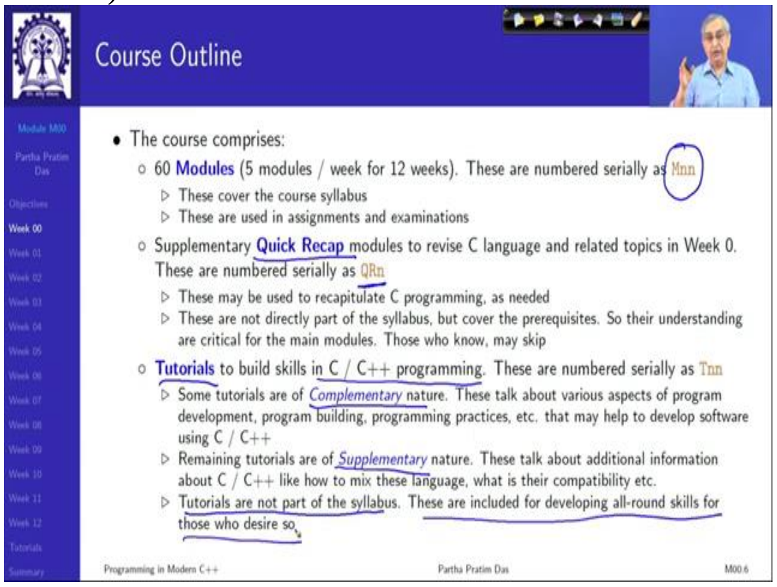
We will often refer to these data structures and these kinds of elementary algorithms. So please do a recapitulation of that, before you start doing the course modules. Last but not the least is you must have when you start doing the course, every module will have a number of examples discussed in terms of the code in terms of the expected output and so on. So, I will strongly advise, that the easiest way to get skilled as a developer would be to take each and every example and actually compile them, build them, run them and see the effect. Whether it is, some examples are meant for correctly working, some are meant to illustrate some errors or pitfalls and so on. You should be able to experience all of that, not merely by what I say in the slide, but by actually doing them.

So, you need a C and C++ compiler, because we have, we will be working on both of these, you will need a debugger. So, for this, you will if you are on Linux, then normally the Linux distribution has that, what we call is a GCC your G, GNU compiler collection. Which has C compiler as well as C++ compiler. If you are on windows, then you should use this particular compiler, which is called minimalist GNU for windows or MinGW.

Use MinGW, use the GDB from the GNU project and if you want assistance in terms of how to install them, I have given reference of a, given reference of a video here, which is from YouTube here that the presenter has very nicely described how to install GDB on the windows, so, get that installed. So, your actions, to summarize your actions before the course starts is to familiarize with the modules and tutorials overall.

Do not try to spend a lot of time, because they will gradually evolve. This is just a checklist for you, for the future also. Revise your C either independently or through this quick recap modules. Recap, revise your data structures, revise your algorithms both at the basic starting level and install and try out the build tool. If you have done this, you will be rightly positioned to start the course from the very first module. And he will be able to make most out of this course. So, please try to do that.

(Refer Slide Time: 17:24)



Course Outline

- The course comprises:
  - 60 **Modules** (5 modules / week for 12 weeks). These are numbered serially as Mnn
    - ▷ These cover the course syllabus
    - ▷ These are used in assignments and examinations
  - Supplementary **Quick Recap** modules to revise C language and related topics in Week 0. These are numbered serially as QRn
    - ▷ These may be used to recapitulate C programming, as needed
    - ▷ These are not directly part of the syllabus, but cover the prerequisites. So their understanding are critical for the main modules. Those who know, may skip
  - **Tutorials** to build skills in C / C++ programming. These are numbered serially as Tnn
    - ▷ Some tutorials are of *Complementary* nature. These talk about various aspects of program development, program building, programming practices, etc. that may help to develop software using C / C++
    - ▷ Remaining tutorials are of *Supplementary* nature. These talk about additional information about C / C++ like how to mix these language, what is their compatibility etc.
    - ▷ Tutorials are not part of the syllabus. These are included for developing all-round skills for those who desire so

Module M00 | Partha Pratim Das | Objectives | Week 00 | Week 01 | Week 02 | Week 03 | Week 04 | Week 05 | Week 06 | Week 07 | Week 08 | Week 09 | Week 10 | Week 11 | Week 12 | Tutorials | Summary

Programming in Modern C++ | Partha Pratim Das | M00.6

Now, in terms of the overall course outline, as you know, this is a 12-week course, every week you will get 5 modules, for 12 weeks. Each module would be about half an hour, some may be of 28 minutes, some may be 32 minutes, but on average of half an hour. And so, you will have total of 60 modules, the modules are numbered by M. So, like this is M00, your first module of week one will be M01, second will be M02 like that.

In this way you will have up to M60. So, they will cover the core syllabus and they are used for assignments and examinations. So, this is your primary source of information. You, I have already talked about supplementary quick recap module which are shown as keyword, which is primarily to help you recapitulation on C. Then there are a series of tutorials givens. The objective of the tutorials is twofold.

One is to help you practice some of the core things that we are doing in these modules. That is one part, but what I have been trying to focus very significantly is to cover those aspects of software development particularly in C++, that you must know but that is not a part of the course syllabus. For example, how to build a C++ program, well how to build a library? How to use automation in there in building a C++ project, which is slowly getting big?

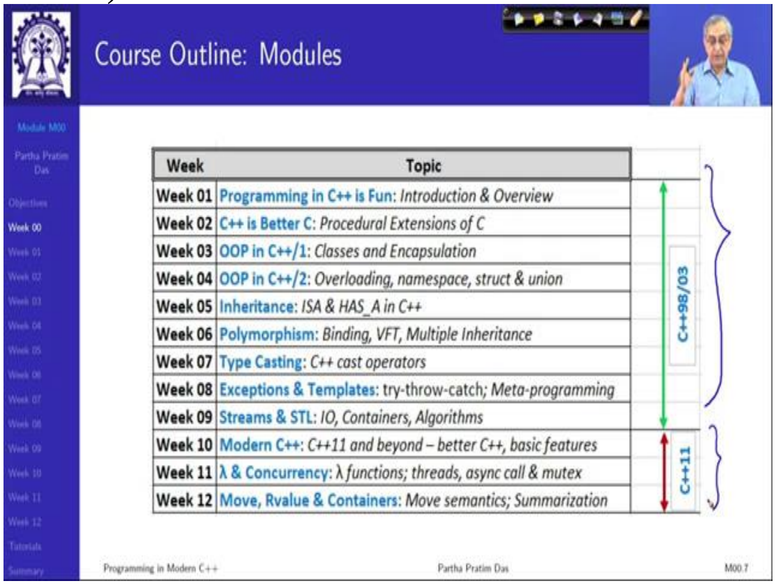
If we are talking about C and C++ both, are they compatible? Will a C program, can a C program be considered as a C++ program and be compiled with the C++ compiler? What are the different tools available for development for testing for archiving and so on so forth. So, there are various aspects, that are beyond the basic learning of the language, which are required for you to become good developers.

Tutorials will not be used for your assignments or examination. So, if you do not immediately have time to do them all, it is your call. But if you want to become a good developer of C C++, and want to attract attention of a good pay package, I will say that do the tutorials very thoroughly, repeatedly and practice whatever we are trying to say in those. Tutorials are of true nature one way is saying the complimentary type.

Complimentary type of those which actually does not talk about make reference to the language so much, but talks about other things like what are the programming practices? How do you build programs, various aspects of program development and so on. Supplementary ones are extensions of your language knowledge. For example, how to mix in a project, if you have some files, which are in C, some files in C++.

How do you deal with that mix situation? Can a C function call a C++ function? Can a C++ function call a C function and so on so forth. What is the compatibility between these languages and so on, which are talking about the language to a good extent. But talking beyond the scope of the language learning in the course, but very much in the scope of becoming a good C++ developer. So, I have clearly said tutorials are not part of the syllabus. They are included for developing all-around skills for you, if you desire to do so.

(Refer Slide Time: 21:44)



Course Outline: Modules

Week	Topic
Week 01	Programming in C++ is Fun: *Introduction & Overview*
Week 02	C++ is Better C: *Procedural Extensions of C*
Week 03	OOP in C++/1: *Classes and Encapsulation*
Week 04	OOP in C++/2: *Overloading, namespace, struct & union*
Week 05	Inheritance: *ISA & HAS_A in C++*
Week 06	Polymorphism: *Binding, VFT, Multiple Inheritance*
Week 07	Type Casting: *C++ cast operators*
Week 08	Exceptions & Templates: try-throw-catch; *Meta-programming*
Week 09	Streams & STL: *IO, Containers, Algorithms*
Week 10	Modern C++: *C++11 and beyond – better C++, basic features*
Week 11	λ & Concurrency: *λ functions; threads, async call & mutex*
Week 12	Move, Rvalue & Containers: *Move semantics; Summarization*

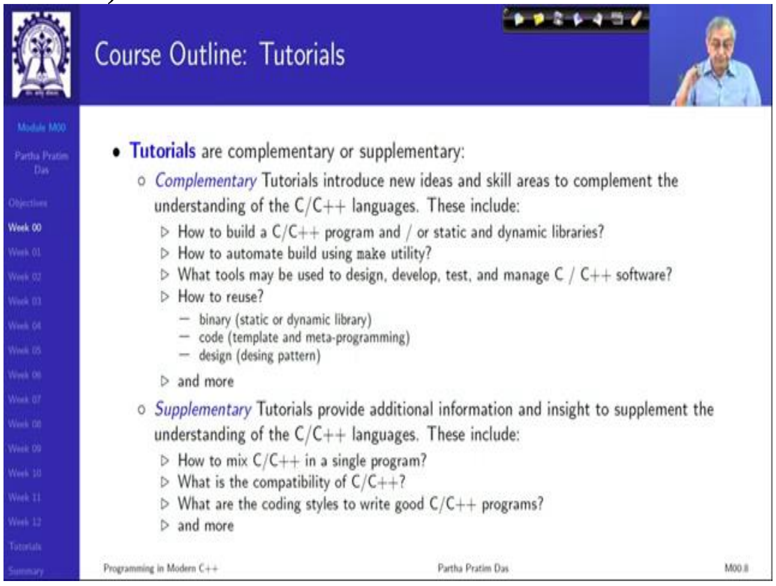
C++98/03

C++11

Programming in Modern C++ Partha Pratim Das M00.7

These are the week wise topics, normally in every, I mean the way I have organized in every week, I have a primary theme that we are trying to learn. And how do they cover, so here are those. And as you can see that as first 9 weeks, we will primarily focus on the commonly known C++, which is C++ 98, or 03. And the last three weeks will focus on the modern aspects of C++, which is modern C++ and C++ 11. So, these are the basic topical details.

(Refer Slide Time: 22:28)



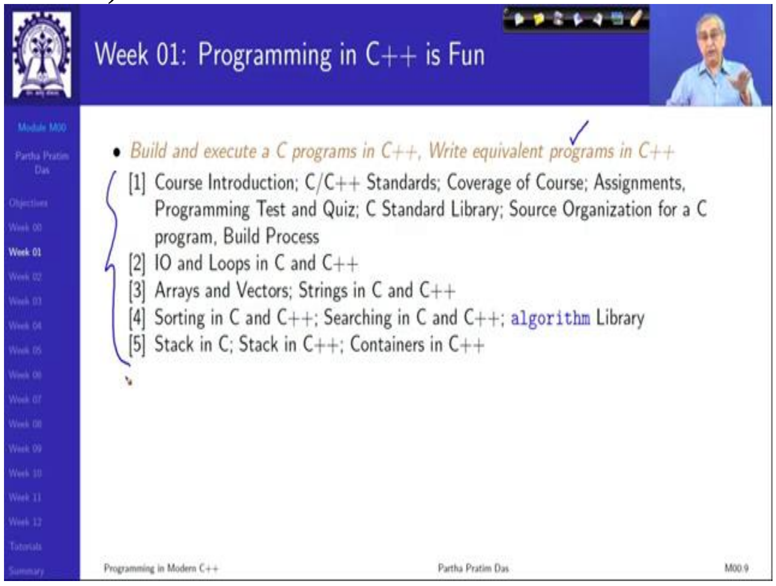
Course Outline: Tutorials

- **Tutorials** are complementary or supplementary:
  - *Complementary* Tutorials introduce new ideas and skill areas to complement the understanding of the C/C++ languages. These include:
    - ▷ How to build a C/C++ program and / or static and dynamic libraries?
    - ▷ How to automate build using make utility?
    - ▷ What tools may be used to design, develop, test, and manage C / C++ software?
    - ▷ How to reuse?
      - binary (static or dynamic library)
      - code (template and meta-programming)
      - design (desing pattern)
    - ▷ and more
  - *Supplementary* Tutorials provide additional information and insight to supplement the understanding of the C/C++ languages. These include:
    - ▷ How to mix C/C++ in a single program?
    - ▷ What is the compatibility of C/C++?
    - ▷ What are the coding styles to write good C/C++ programs?
    - ▷ and more

Programming in Modern C++ Partha Pratim Das M00.8

In terms of tutorial, I have already said that we will have complementary nature tutorials, and we will have supplementary in nature tutorials. Later on, in this module itself, I have some I have a tentative list of the tutorials that we are going to provide to you.

(Refer Slide Time: 22:48)



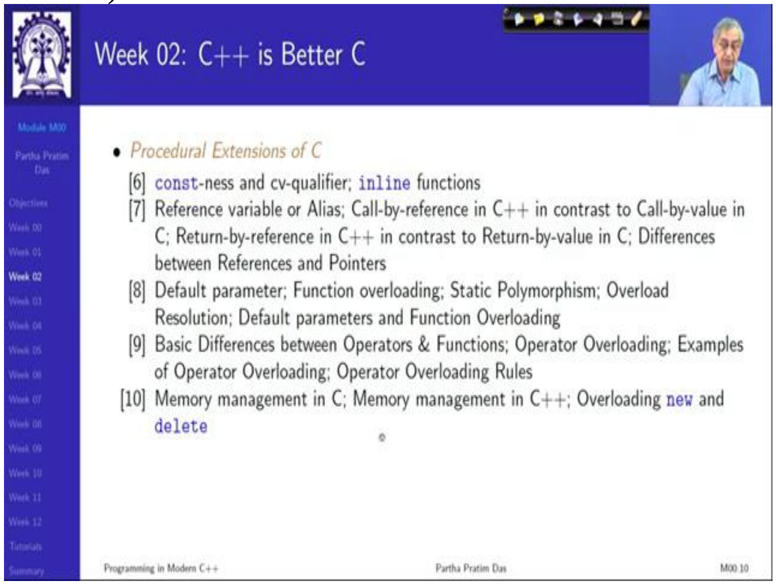
Week 01: Programming in C++ is Fun

- *Build and execute a C programs in C++, Write equivalent programs in C++*
  - [1] Course Introduction; C/C++ Standards; Coverage of Course; Assignments, Programming Test and Quiz; C Standard Library; Source Organization for a C program, Build Process
  - [2] IO and Loops in C and C++
  - [3] Arrays and Vectors; Strings in C and C++
  - [4] Sorting in C and C++; Searching in C and C++; `algorithm` Library
  - [5] Stack in C; Stack in C++; Containers in C++

Programming in Modern C++ Partha Pratim Das M00.9

Now, from this point onward, for every week, you will have, you will find a slide which talks about the theme of that week, which is on top written in brown. And these are the different module numbers. So, it is M01 to M05 is what you get in the first week. And what is the topic of that particular module is described here. It is no point reading, keep on reading this out. So, I will just, you know flip through and you can make references and you can make your own notes based on this plan of how you want to schedule, your study and so on. So, this is about building and executing C programs in C++ writing equivalent programs.

(Refer Slide Time: 23:36)



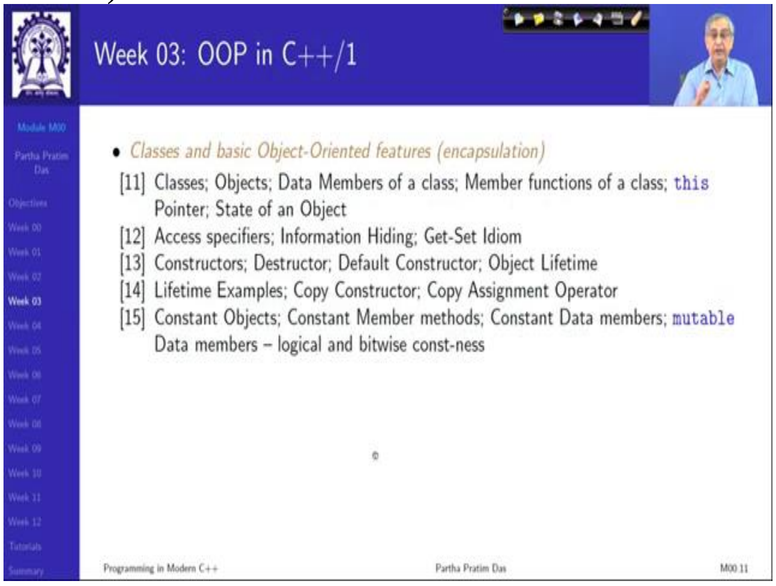
Week 02: C++ is Better C

- *Procedural Extensions of C*
  - [6] `const`-ness and cv-qualifier; `inline` functions
  - [7] Reference variable or Alias; Call-by-reference in C++ in contrast to Call-by-value in C; Return-by-reference in C++ in contrast to Return-by-value in C; Differences between References and Pointers
  - [8] Default parameter; Function overloading; Static Polymorphism; Overload Resolution; Default parameters and Function Overloading
  - [9] Basic Differences between Operators & Functions; Operator Overloading; Examples of Operator Overloading; Operator Overloading Rules
  - [10] Memory management in C; Memory management in C++; Overloading `new` and `delete`

Programming in Modern C++ Partha Pratim Das M00.10

Week two will primarily talk about the procedural extensions of C into C++, which are not object oriented but very important for the language.

(Refer Slide Time: 23:48)



Week 03: OOP in C++/1

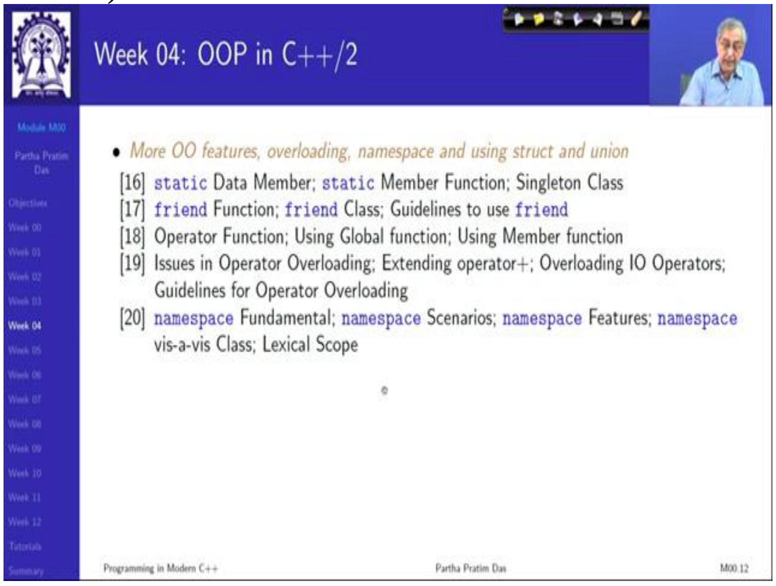
- *Classes and basic Object-Oriented features (encapsulation)*

  [11] Classes; Objects; Data Members of a class; Member functions of a class; `this` Pointer; State of an Object
  [12] Access specifiers; Information Hiding; Get-Set Idiom
  [13] Constructors; Destructor; Default Constructor; Object Lifetime
  [14] Lifetime Examples; Copy Constructor; Copy Assignment Operator
  [15] Constant Objects; Constant Member methods; Constant Data members; `mutable` Data members – logical and bitwise const-ness

Programming in Modern C++ Partha Pratim Das M00.11

Week three, we will start on the object-oriented programming and we will continue into week four. So, we say OOP in C++, part one. Which talks about the basic features encapsulations and so on.

(Refer Slide Time: 24:00)



Week 04: OOP in C++/2

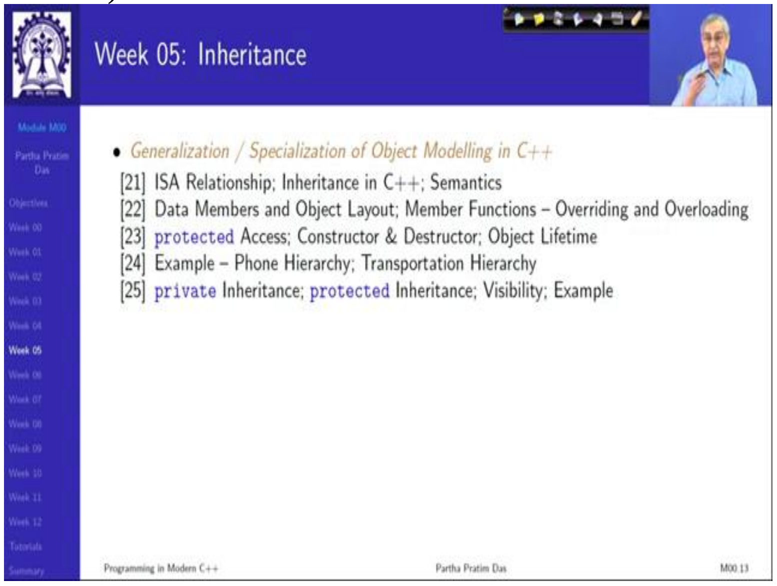
- *More OO features, overloading, namespace and using struct and union*

  [16] `static` Data Member; `static` Member Function; Singleton Class
  [17] `friend` Function; `friend` Class; Guidelines to use `friend`
  [18] Operator Function; Using Global function; Using Member function
  [19] Issues in Operator Overloading; Extending operator+; Overloading IO Operators; Guidelines for Operator Overloading
  [20] `namespace` Fundamental; `namespace` Scenarios; `namespace` Features; `namespace` vis-a-vis Class; Lexical Scope

Programming in Modern C++ Partha Pratim Das M00.12

OOP in other features, overloading, namespace, structure, union, this will be done in the second part, in week four.

(Refer Slide Time: 24:12)



Week 05: Inheritance

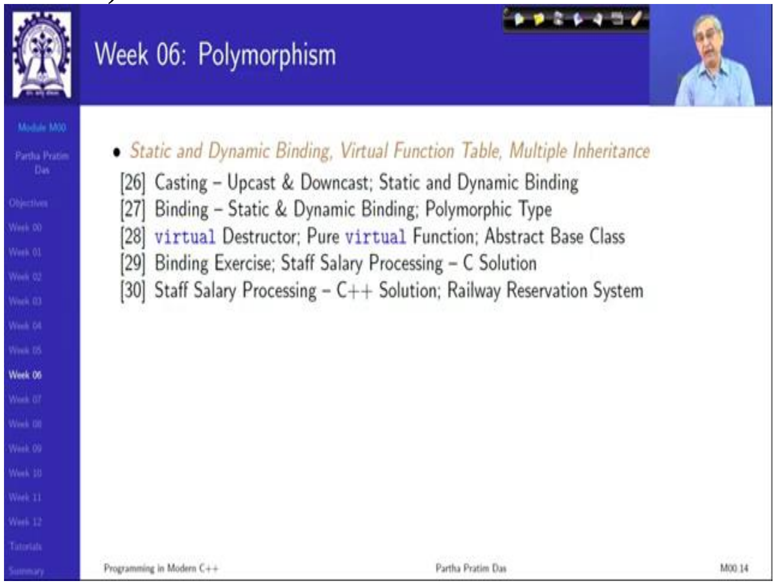
- *Generalization / Specialization of Object Modelling in C++*

[21] ISA Relationship; Inheritance in C++; Semantics
[22] Data Members and Object Layout; Member Functions – Overriding and Overloading
[23] `protected` Access; Constructor & Destructor; Object Lifetime
[24] Example – Phone Hierarchy; Transportation Hierarchy
[25] `private` Inheritance; `protected` Inheritance; Visibility; Example

Programming in Modern C++ Partha Pratim Das M00.13

Week five, we will talk about another very, very strong feature of object orientation that is inheritance. What is generalization, specialization concept?

(Refer Slide Time: 24:22)



Week 06: Polymorphism

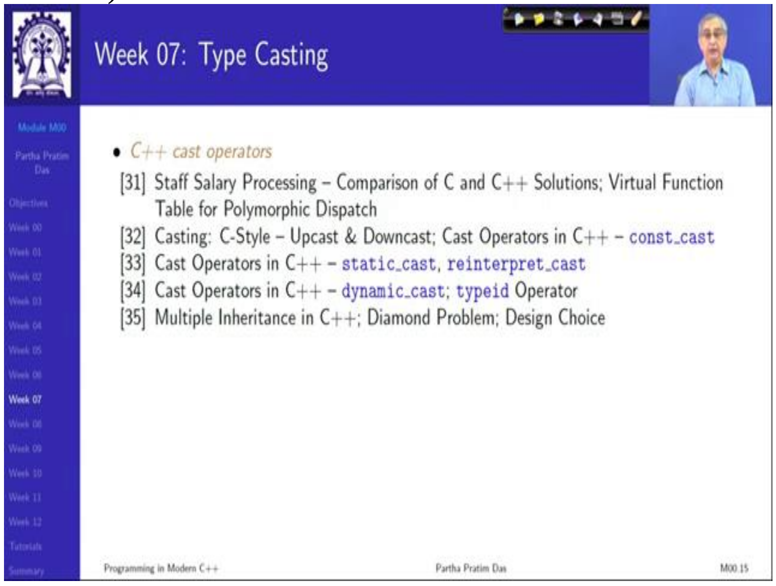
- *Static and Dynamic Binding, Virtual Function Table, Multiple Inheritance*

[26] Casting – Upcast & Downcast; Static and Dynamic Binding
[27] Binding – Static & Dynamic Binding; Polymorphic Type
[28] `virtual` Destructor; Pure `virtual` Function; Abstract Base Class
[29] Binding Exercise; Staff Salary Processing – C Solution
[30] Staff Salary Processing – C++ Solution; Railway Reservation System

Programming in Modern C++ Partha Pratim Das M00.14

It will extend to polymorphism, where you can various ways you can write the same function which behaves in multiple different ways, based on the context of the object.

(Refer Slide Time: 24:36)



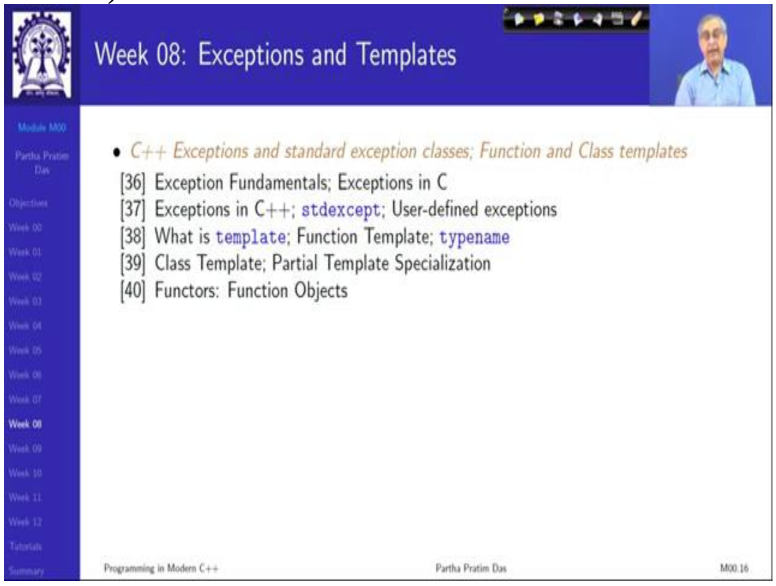
Week 07: Type Casting

- *C++ cast operators*
  - [31] Staff Salary Processing – Comparison of C and C++ Solutions; Virtual Function Table for Polymorphic Dispatch
  - [32] Casting: C-Style – Upcast & Downcast; Cast Operators in C++ – `const_cast`
  - [33] Cast Operators in C++ – `static_cast`, `reinterpret_cast`
  - [34] Cast Operators in C++ – `dynamic_cast`; `typeid` Operator
  - [35] Multiple Inheritance in C++; Diamond Problem; Design Choice

Programming in Modern C++ Partha Pratim Das M00.15

We will have information on casting, how do you take object of certain type and use it as an object of a different type which is a very key area of understanding, it is important to understand, it is as well as for to for effective use and as well as for avoidance of common errors.

(Refer Slide Time: 24:57)



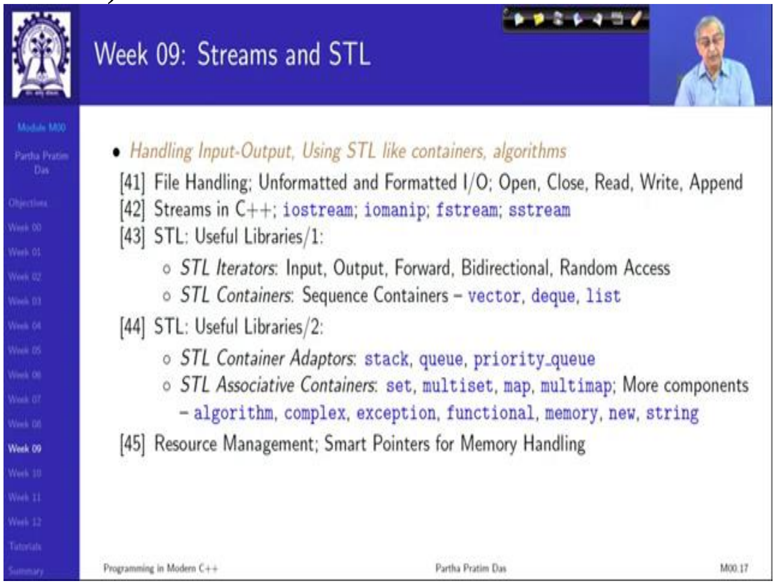
Week 08: Exceptions and Templates

- *C++ Exceptions and standard exception classes; Function and Class templates*
  - [36] Exception Fundamentals; Exceptions in C
  - [37] Exceptions in C++; `stdexcept`; User-defined exceptions
  - [38] What is `template`; Function Template; `typename`
  - [39] Class Template; Partial Template Specialization
  - [40] Functors: Function Objects

Programming in Modern C++ Partha Pratim Das M00.16

Week 8, we will talk about exceptions. That is one theme, which is what happens if your program gets into some difficulty at the runtime. So, that C++ has a very strong feature to talk about that. And the other theme, so, it has two themes actually, the other theme is templates, which is called Meta Programming in C++. In simple terms, templates are kind of, that kind of programming in C++, where you write a function and then you have certain parameters of that, which we see our type parameters. When the, when someone uses that function, then based on the actual type of use, the compiler will generate multiple functions. So, it is a code generator within your program. So, it is a very, very strong feature.

(Refer Slide Time: 25:54)



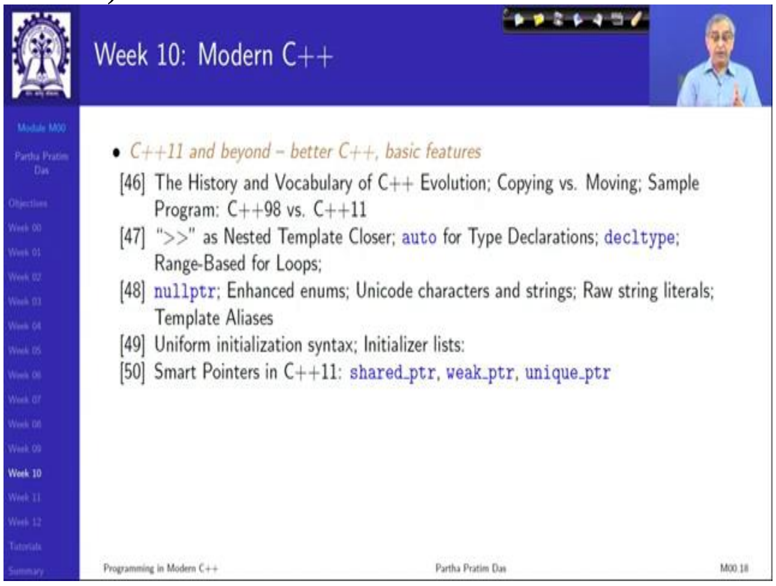
Week 09: Streams and STL

- *Handling Input-Output, Using STL like containers, algorithms*
  - [41] File Handling; Unformatted and Formatted I/O; Open, Close, Read, Write, Append
  - [42] Streams in C++; `iostream`; `iomanip`; `fstream`; `sstream`
  - [43] STL: Useful Libraries/1:
    - *STL Iterators*: Input, Output, Forward, Bidirectional, Random Access
    - *STL Containers*: Sequence Containers – `vector`, `deque`, `list`
  - [44] STL: Useful Libraries/2:
    - *STL Container Adaptors*: `stack`, `queue`, `priority_queue`
    - *STL Associative Containers*: `set`, `multiset`, `map`, `multimap`; More components – `algorithm`, `complex`, `exception`, `functional`, `memory`, `new`, `string`
  - [45] Resource Management; Smart Pointers for Memory Handling

Programming in Modern C++ Partha Pratim Das M00.17

And we will close on C++ 03 which discussions on input output structures like streams, vary, a variety of them, and the brief overview of the standard library.

(Refer Slide Time: 26:10)



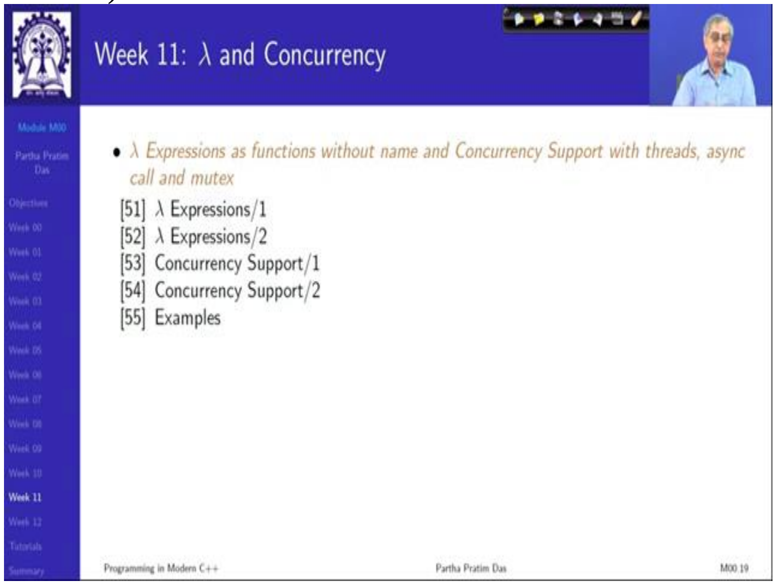
Week 10: Modern C++

- *C++11 and beyond – better C++, basic features*
  - [46] The History and Vocabulary of C++ Evolution; Copying vs. Moving; Sample Program: C++98 vs. C++11
  - [47] ">>" as Nested Template Closer; `auto` for Type Declarations; `decltype`; Range-Based for Loops;
  - [48] `nullptr`; Enhanced enums; Unicode characters and strings; Raw string literals; Template Aliases
  - [49] Uniform initialization syntax; Initializer lists;
  - [50] Smart Pointers in C++11: `shared_ptr`, `weak_ptr`, `unique_ptr`

Programming in Modern C++ Partha Pratim Das M00.18

From week 10, we will move into modern C++ introducing you. Again, in a similar manner, which part of C++ is just better C++? It so sounds a number of features are again provided just to enhance the language, not really primarily giving you something new.

(Refer Slide Time: 26:34)


Week 11: λ and Concurrency

Module M00

Partha Pratim Das

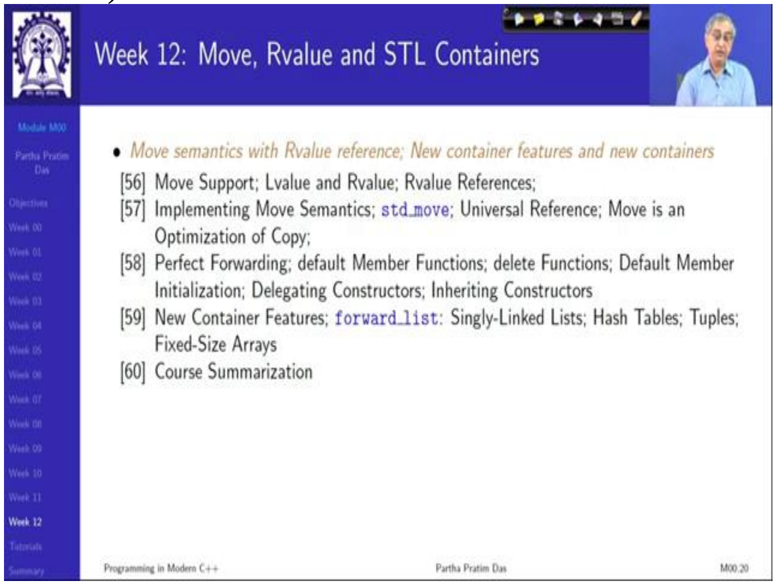
- λ Expressions as functions without name and Concurrency Support with threads, async call and mutex

[51] λ Expressions/1
[52] λ Expressions/2
[53] Concurrency Support/1
[54] Concurrency Support/2
[55] Examples

Programming in Modern C++ Partha Pratim Das M00.19

Week 11 would be very, very interesting and somewhat difficult, where we will talk about again, two key things one is lambda expressions. Lambda expressions are nothing but they are functions, but they are anonymous function. That is, they do not have a name. These are functions which do not have a name. Now, what does that, all that mean? So, you will see. And then support for concurrency for writing concurrent programs. And we will try to do some specific examples work out in this.

(Refer Slide Time: 27:06)


Week 12: Move, Rvalue and STL Containers

Module M00

Partha Pratim Das

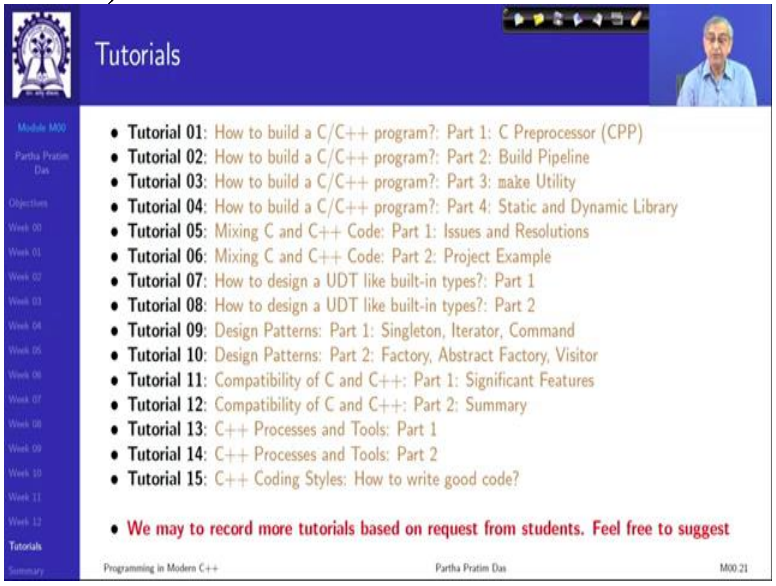
- Move semantics with Rvalue reference; New container features and new containers

[56] Move Support; Lvalue and Rvalue; Rvalue References;
[57] Implementing Move Semantics; std::move; Universal Reference; Move is an Optimization of Copy;
[58] Perfect Forwarding; default Member Functions; delete Functions; Default Member Initialization; Delegating Constructors; Inheriting Constructors
[59] New Container Features; forward_list: Singly-Linked Lists; Hash Tables; Tuples; Fixed-Size Arrays
[60] Course Summarization

Programming in Modern C++ Partha Pratim Das M00.20

In the concluding week, we will talk about some of the other foundational concepts in C++ 11, the move semantics, the R value, you have heard about our value L value. But what is R value semantics? What is R-value reference? And what are the extensions to the C++ standard library coming to C++ 11. And then we will summarize on the course. So, this is this is a overall plan. And as you as you keep doing, it will keep on unfolding.

(Refer Slide Time: 27:43)



Tutorials

- **Tutorial 01**: How to build a C/C++ program?: Part 1: C Preprocessor (CPP)
- **Tutorial 02**: How to build a C/C++ program?: Part 2: Build Pipeline
- **Tutorial 03**: How to build a C/C++ program?: Part 3: make Utility
- **Tutorial 04**: How to build a C/C++ program?: Part 4: Static and Dynamic Library
- **Tutorial 05**: Mixing C and C++ Code: Part 1: Issues and Resolutions
- **Tutorial 06**: Mixing C and C++ Code: Part 2: Project Example
- **Tutorial 07**: How to design a UDT like built-in types?: Part 1
- **Tutorial 08**: How to design a UDT like built-in types?: Part 2
- **Tutorial 09**: Design Patterns: Part 1: Singleton, Iterator, Command
- **Tutorial 10**: Design Patterns: Part 2: Factory, Abstract Factory, Visitor
- **Tutorial 11**: Compatibility of C and C++: Part 1: Significant Features
- **Tutorial 12**: Compatibility of C and C++: Part 2: Summary
- **Tutorial 13**: C++ Processes and Tools: Part 1
- **Tutorial 14**: C++ Processes and Tools: Part 2
- **Tutorial 15**: C++ Coding Styles: How to write good code?

- **We may to record more tutorials based on request from students. Feel free to suggest**

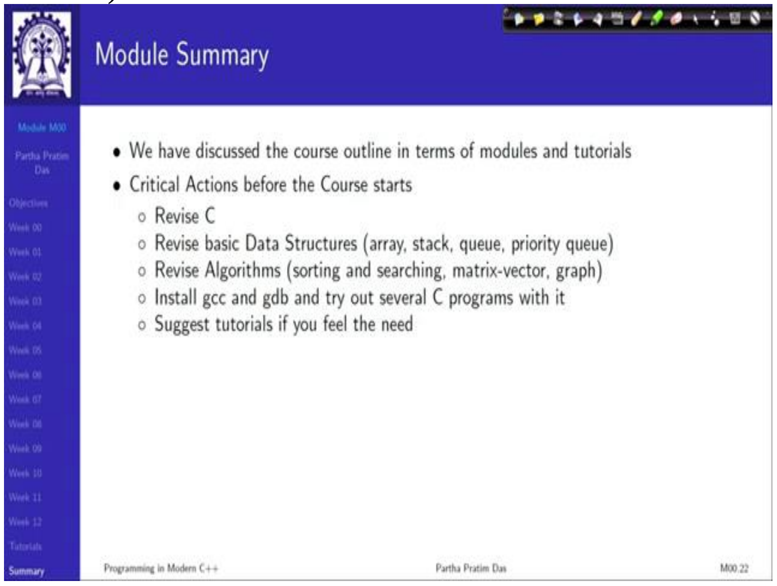
Programming in Modern C++ Partha Pratim Das M00.21

In terms of the tutorials, there are some about 15 tutorials are currently planned. First four of them is about building C, C++ program, next 02, 05 and 06 is for mixing C and C++ code. The next two 07 and 08 are basically will be directly useful to the module because we work out a number of examples to show you, how you can make a user defined type which behaves similar to any built-in type like int or double.

Like how do you build a type for complex or how do you build a type for, say, a fraction? You want a fraction to type to be there and behave exactly like int? How do you do that? Then, in 09 and 10, we will talk about the concepts of design pattern, which will be foundational in terms of your design understanding, if you want to specialize into that. 11 and 12, we will talk about compatibility between C and C++.

We will talk about several standard software development processes for C++ and tools in 13, and 14, and in 15, we plan to cover C++ coding style, that is, what is a good way to write code in C++. This is not an exhaustive list like unlike the modules which are frozen. The tutorials are not an exhaustive list. These are what currently we plan to do. We may record appropriately more tutorials if there are requests from you all. So, please feel free to suggest if you think that well in this area or in this topic, it would be good to have a tutorial, which will complement and or supplement your knowledge to become a good developer.

(Refer Slide Time: 29:30)



Module Summary

- We have discussed the course outline in terms of modules and tutorials
- Critical Actions before the Course starts
  - Revise C
  - Revise basic Data Structures (array, stack, queue, priority queue)
  - Revise Algorithms (sorting and searching, matrix-vector, graph)
  - Install gcc and gdb and try out several C programs with it
  - Suggest tutorials if you feel the need

Module M00
Partha Pratim Das
Objectives
Week 00
Week 01
Week 02
Week 03
Week 04
Week 05
Week 06
Week 07
Week 08
Week 09
Week 10
Week 11
Week 12
Tutorials
Summary
Programming in Modern C++
Partha Pratim Das
M00.22

So, with this I conclude on this module, we have discussed on the course outline in terms of the different modules and tutorials that you can expect for the course. And just a reminder for the critical actions. Please take these actions before you start getting into the course revise your C, revise your basic data structure, revise the basic algorithms and install GCC and GDB and try out several C programs with it.

Even if you have an access to some other C or C++ compiler on your system, you must use install these GCC and GDB. Because throughout the course, we will take this as a standard and all examples, all comparisons will be done based on that. So, if you come and say that what can my Turbo C++ is giving this I am not going to, I am not will be, I will not be able to listen and respond to that.

Because I, I do not have that compiler. There are hundreds of compilers. We have the most popular is GCC and it is free. So, all of us can use it. And we will use that as a standard tool. In fact, much of the industry also uses this as a standard tool. So, we will keep to that. And if you think suggest more tutorials, I will be happy to record it for you.

So, welcome again, to programming in modern C++ course. I am really happy that you have decided to skill yourself up in a very critical area of computer science and I hope you will have excellent couple of months, very successful couple of months with this course. Thank you.