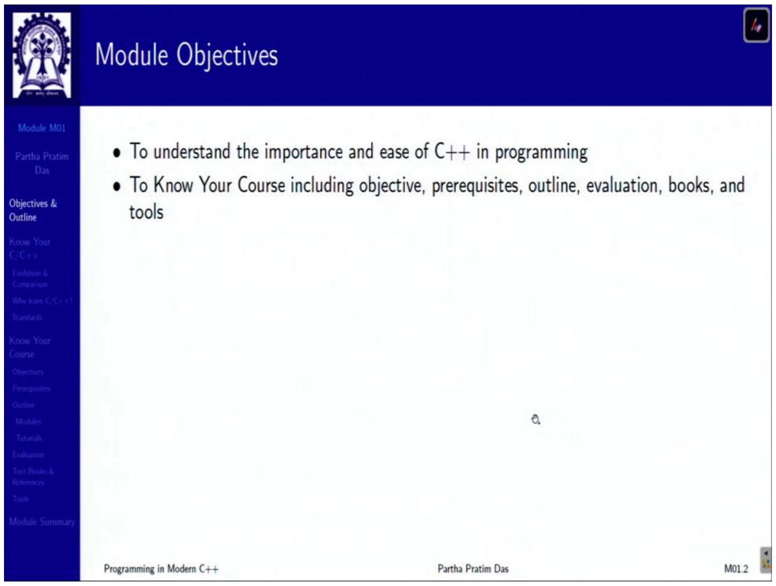
Programming in Modern C++
Professor. Partha Pratim Das
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur
Lecture No. 04
Course Overview

Welcome to Programming in Modern C++. This is the first week and we are going to talk on the module 01.

(Refer Slide Time: 00:51)



Module Objectives

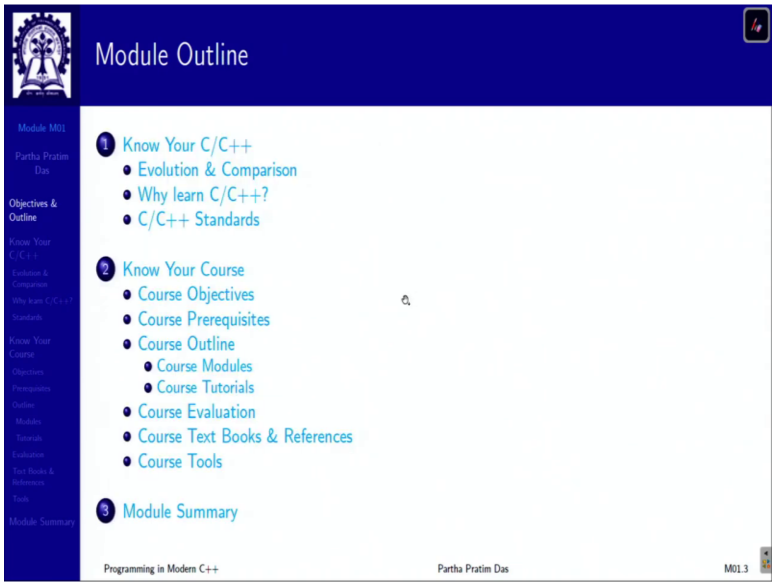
- To understand the importance and ease of C++ in programming
- To Know Your Course including objective, prerequisites, outline, evaluation, books, and tools

Module M01
Partha Pratim Das
Objectives & Outline
Know Your C/C++
Evolution & Comparison
Why learn C/C++?
Standards
Know Your Course
Objectives
Prerequisites
Outline
Modules
Tutorials
Evaluation
Text Books & References
Tools
Module Summary

Programming in Modern C++ Partha Pratim Das M01.2

So, this particular module will primarily discuss the importance and ease of C++ in programming in brief, just to introduce you to what has been happening in C++ over the couple of last decades. But primarily, this module will make you familiar with your course including the objective, prerequisites, outline, evaluation, textbooks and tools.

(Refer Slide Time: 01:25)



Module Outline

1. Know Your C/C++
   - Evolution & Comparison
   - Why learn C/C++?
   - C/C++ Standards
2. Know Your Course
   - Course Objectives
   - Course Prerequisites
   - Course Outline
     - Course Modules
     - Course Tutorials
   - Course Evaluation
   - Course Text Books & References
   - Course Tools
3. Module Summary

Module M01

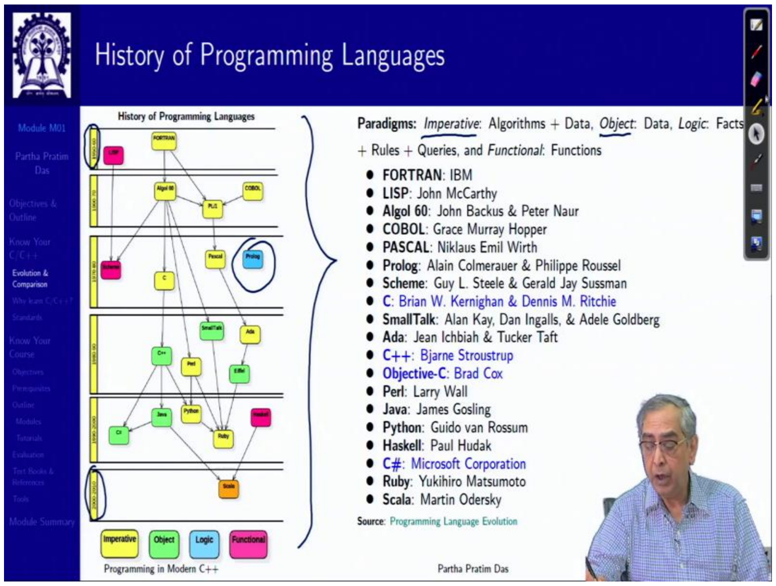
Partha Pratim Das

Objectives & Outline

Programming in Modern C++ Partha Pratim Das M01.3

So, this is the module outline, which will always be visible to you on the left column.

(Refer Slide Time: 01:40)



History of Programming Languages

Paradigms: *Imperative*: Algorithms + Data, *Object*: Data, *Logic*: Facts + Rules + Queries, and *Functional*: Functions

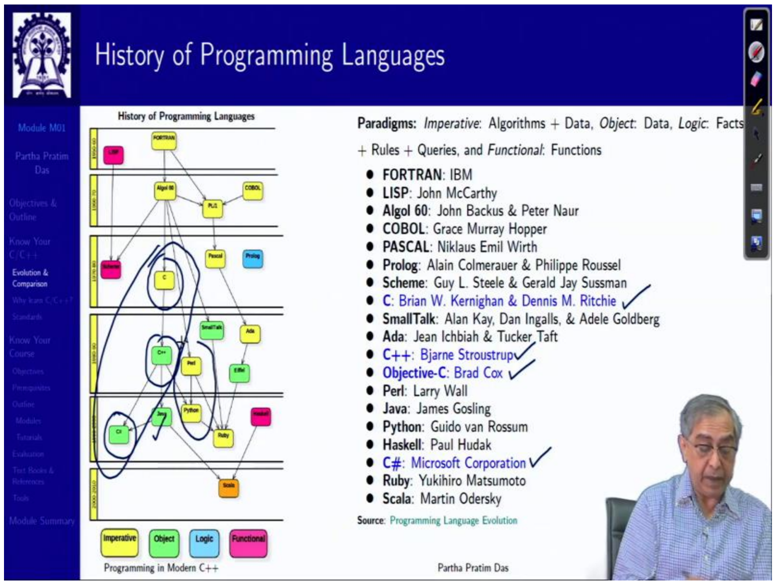
- **FORTRAN**: IBM
- **LISP**: John McCarthy
- **Algol 60**: John Backus & Peter Naur
- **COBOL**: Grace Murray Hopper
- **PASCAL**: Niklaus Emil Wirth
- **Prolog**: Alain Colmerauer & Philippe Roussel
- **Scheme**: Guy L. Steele & Gerald Jay Sussman
- **C**: Brian W. Kernighan & Dennis M. Ritchie
- **SmallTalk**: Alan Kay, Dan Ingalls, & Adele Goldberg
- **Ada**: Jean Ichbiah & Tucker Taft
- **C++**: Bjarne Stroustrup
- **Objective-C**: Brad Cox
- **Perl**: Larry Wall
- **Java**: James Gosling
- **Python**: Guido van Rossum
- **Haskell**: Paul Hudak
- **C#**: Microsoft Corporation
- **Ruby**: Yukihiro Matsumoto
- **Scala**: Martin Odersky

Source: Programming Language Evolution

Module M01

Partha Pratim Das

Programming in Modern C++ Partha Pratim Das



History of Programming Languages

Paradigms: *Imperative*: Algorithms + Data, *Object*: Data, *Logic*: Facts + Rules + Queries, and *Functional*: Functions

- **FORTRAN**: IBM
- **LISP**: John McCarthy
- **Algol 60**: John Backus & Peter Naur
- **COBOL**: Grace Murray Hopper
- **PASCAL**: Niklaus Emil Wirth
- **Prolog**: Alain Colmerauer & Philippe Roussel
- **Scheme**: Guy L. Steele & Gerald Jay Sussman
- C: Brian W. Kernighan & Dennis M. Ritchie ✓
- **SmallTalk**: Alan Kay, Dan Ingalls, & Adele Goldberg
- **Ada**: Jean Ichbiah & Tucker Taft
- C++: Bjarne Stroustrup ✓
- Objective-C: Brad Cox ✓
- **Perl**: Larry Wall
- **Java**: James Gosling
- **Python**: Guido van Rossum
- **Haskell**: Paul Hudak
- C#: Microsoft Corporation ✓
- **Ruby**: Yukihiro Matsumoto
- **Scala**: Martin Odersky

Source: Programming Language Evolution

Programming in Modern C++ Partha Pratim Das

So, to know your C, C++ I presume and that is a prerequisite, that you know your C language, you know, what the language features are, what is the standard library and what is the way to do programming in C. So, given that perspective, let me just quickly focus on this graph, which is kind of a timeline as you can see from your 1950s, when programming in terms of high-level language is primarily started till about the last decade.

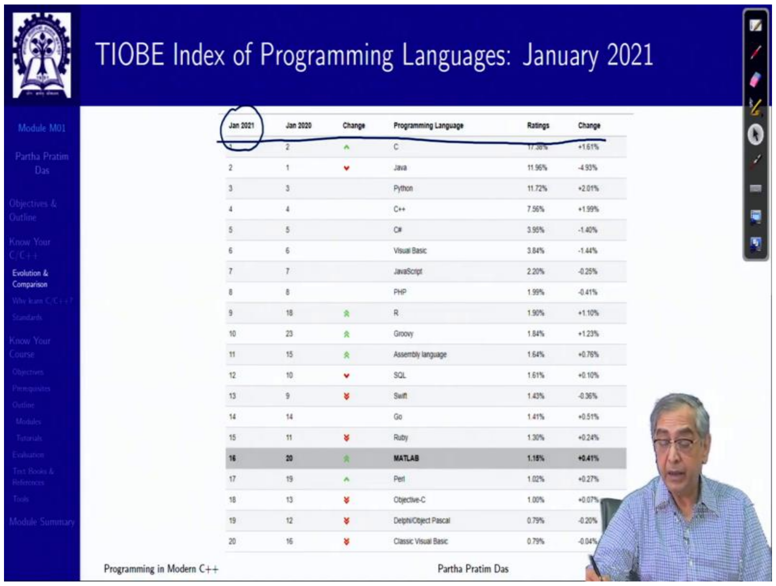
I mean, there are more advances happening. And languages have evolved with different paradigms. Paradigms mean that whether most of the languages shown here in yellow are imperative, which means that they deal with algorithm and data. So, C is an imperative language. Then you have languages which are object based, where you primarily focus on the data and put algorithms on top of that. So, those are what is shown in green here. So, C++, Java are examples of that.

Then you have languages which are facts, rules and queries based called the logic languages. And prologue in as you see prologue here in blue is kind of the leading component of that, leading language of that. And you have functional languages also which are very, very interesting in the sense that, they treat everything as a function and as such do not have an explicit knowledge of a memory.

So, these are different nuances, it is not important that you look into all of these at the same time as we are focusing on the C++, modern C++ primarily. As you can see, that we have, I have highlighted couple of languages here, which have happened say at different times C, C++, C sharp, C++. Then objective C is not shown in this chart, but it is a variant of C with, with certain classes and so on. So, this is kind of what we call is a C family of languages.

That is, they are basically, structurally similar to the C language, but has very different kinds of features. So, given this, we would like to primarily study on this part of the programming language hierarchy. As you can see, there are several other languages like Java, there is a Perl, Python branch and so on, which is, will not be particularly treated here, but they, we will keep on mentioning about some of those features from time to time.

(Refer Slide Time: 05:21)



TIOBE Index of Programming Languages: January 2021

Module M01

Partha Pratim Das

Objectives & Outline

Know Your C/C++

Evolution & Comparison

Why learn C/C++?

Standards

Know Your Course

Objectives

Prerequisites

Outline

Modules

Tutorials

Evaluation

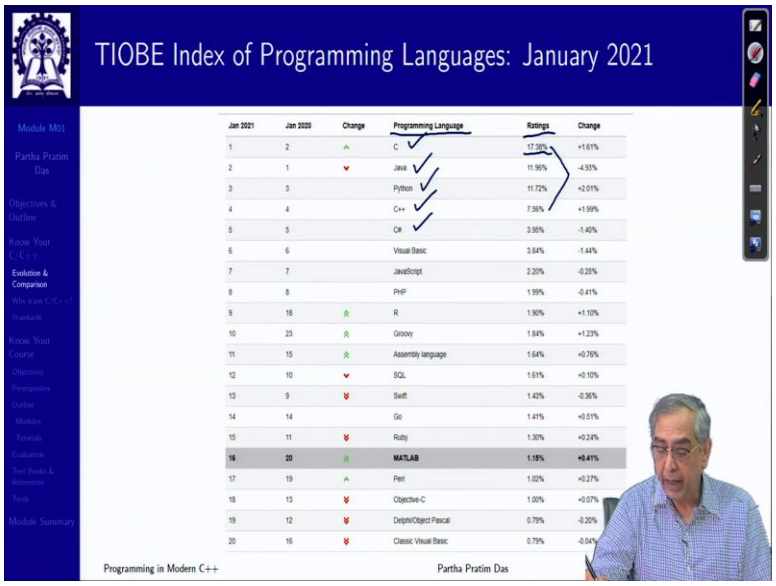
Text Books & References

Tools

Module Summary

Jan 2021	Jan 2020	Change	Programming Language	Ratings	Change
1	2	⌃	C	17.38%	+1.61%
2	1	⌄	Java	11.96%	-4.93%
3	3		Python	11.72%	+2.01%
4	4		C++	7.56%	+1.99%
5	5		C#	3.95%	-1.40%
6	6		Visual Basic	3.84%	-1.44%
7	7		JavaScript	2.20%	-0.25%
8	8		PHP	1.99%	-0.41%
9	18	⇈	R	1.90%	+1.10%
10	23	⇈	Groovy	1.84%	+1.23%
11	15	⇈	Assembly language	1.64%	+0.76%
12	10	⌄	SQL	1.61%	+0.10%
13	9	⇊	Swift	1.43%	-0.36%
14	14		Go	1.41%	+0.51%
15	11	⇊	Ruby	1.30%	+0.24%
16	**20**	⇈	**MATLAB**	**1.15%**	**+0.41%**
17	19	⌃	Perl	1.02%	+0.27%
18	13	⇊	Objective-C	1.00%	+0.07%
19	12	⇊	Delphi/Object Pascal	0.79%	-0.20%
20	16	⇊	Classic Visual Basic	0.79%	-0.04%

Programming in Modern C++ Partha Pratim Das



TIOBE Index of Programming Languages: January 2021

Module M01

Partha Pratim Das

Objectives & Outline

Know Your C/C++

Evolution & Comparison

Why learn C/C++?

Standards

Know Your Course

Objectives

Prerequisites

Outline

Modules

Tutorials

Evaluation

Text Books & References

Tools

Module Summary

Jan 2021	Jan 2020	Change	Programming Language	Ratings	Change
1	2	⌃	C ✓	17.38%	+1.61%
2	1	⌄	Java ✓	11.96%	-4.93%
3	3		Python ✓	11.72%	+2.01%
4	4		C++ ✓	7.56%	+1.99%
5	5		C# ✓	3.95%	-1.40%
6	6		Visual Basic	3.84%	-1.44%
7	7		JavaScript	2.20%	-0.25%
8	8		PHP	1.99%	-0.41%
9	18	⇈	R	1.90%	+1.10%
10	23	⇈	Groovy	1.84%	+1.23%
11	15	⇈	Assembly language	1.64%	+0.76%
12	10	⌄	SQL	1.61%	+0.10%
13	9	⇊	Swift	1.43%	-0.36%
14	14		Go	1.41%	+0.51%
15	11	⇊	Ruby	1.30%	+0.24%
16	**20**	⇈	**MATLAB**	**1.15%**	**+0.41%**
17	19	⌃	Perl	1.02%	+0.27%
18	13	⇊	Objective-C	1.00%	+0.07%
19	12	⇊	Delphi/Object Pascal	0.79%	-0.20%
20	16	⇊	Classic Visual Basic	0.79%	-0.04%

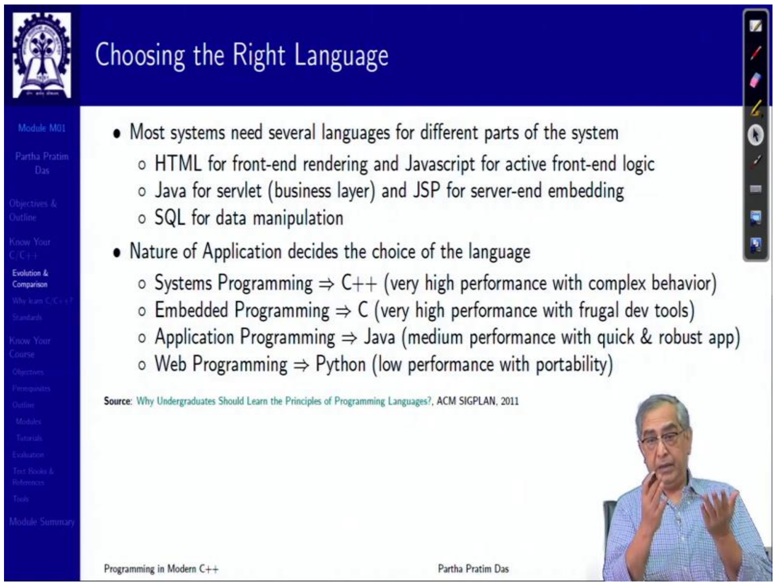
Programming in Modern C++ Partha Pratim Das

Now, many of you will have a question of why C++? I mean why do we learn C++? So, it is a, it is a good point to note that, if I, if we look into the kind of popularity of various languages over time, so, this is the data as of January 2021, then you can see that there are several languages which are ranked according to their popularity. And these are kind of the ratings. TIOBE is an organization which every month publishes this ranking information.

So, you can see that C is ranked the highest with 17.38 percent of usage in the popular open domain, followed by Java, Python and C++, C sharp. So, if you look at this, you can see that these couple of four five languages on the top, are the key languages in which a majority of common programming is being done. And C++ happened to be at the fourth position according to this, it keeps on slightly changing, but the top few languages like C Java, Python and C++, they do not really change much.

So, given that C++ in a way subsumes the entirety of C with little variations, you can say that this and this together is what is going to be your skill about 25 percent, that is the highest percentage of programming language skills you are going to have once you master C++. Further, we will see that there are a lot of specific advantages of C++, there are a lot of specific items that C++ introduced in terms of programming and so on. But I just wanted to give you a glimpse about where does C++ stand in the overall rankings scenario.

(Refer Slide Time: 07:47)



Module M01

Partha Pratim Das

Objectives & Outline

Know Your C/C++

Evolution & Comparison

Why learn C/C++?

Standards

Know Your Course

Objectives

Prerequisites

Outline

Modules

Tutorials

Evaluation

Text Books & References

Tools

Module Summary

Choosing the Right Language

- Most systems need several languages for different parts of the system
  - HTML for front-end rendering and Javascript for active front-end logic
  - Java for servlet (business layer) and JSP for server-end embedding
  - SQL for data manipulation
- Nature of Application decides the choice of the language
  - Systems Programming ⇒ C++ (very high performance with complex behavior)
  - Embedded Programming ⇒ C (very high performance with frugal dev tools)
  - Application Programming ⇒ Java (medium performance with quick & robust app)
  - Web Programming ⇒ Python (low performance with portability)

Source: Why Undergraduates Should Learn the Principles of Programming Languages?, ACM SIGPLAN, 2011

Programming in Modern C++ Partha Pratim Das

So, going forward, you must always it, I often face a question as to if I know C++, can I do everything? If I know Java, can I do everything? Theoretically, yes, every programming language as it is, I mean or most of the programming languages I must say, not every are what is known as Turing complete, which means that any program can be written in them. But it is important to also note that why did people create so many programming languages?

So, many languages are created because some languages have certain advantages over other in terms of certain applications. So, it is always very important to choose the right language. And often you will find systems are not using only one language. For example, if you think about a very common, transactional application like net banking or even say, your Gmail system and so on, which will typically have multiple languages possibly, I mean certainly HTML in the front end with possibly having JavaScript for active front end logic.

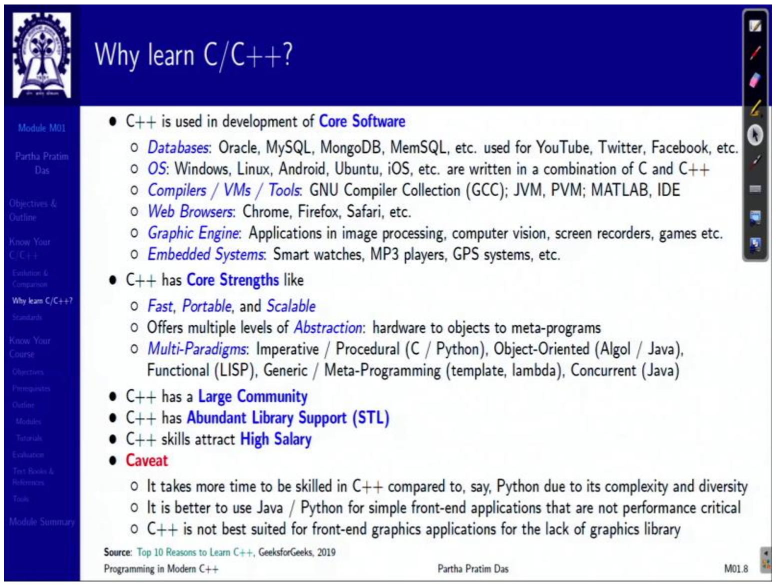
Then, in the business layer, you will possibly have Java and JavaScript for the server and embedding, you will have SQL for the backend database and so on. So, the overall system will involve several languages. So, when you learn a language like C++, you should also keep your mind open in terms of picking up, being aware about the related languages. The other point is as I was saying, that the nature of application decides the choice of language.

What do I want to do? If I want to do systems programming, which need high performance has a complex behavior, then C++ often is the best choice. If I want to do embedded programming, like in mobile phones and so on, then C probably would be a better choice, because it also has high performance. And the embedded systems or handled systems have very frugal development tools which may not be able to handle the whole of C++.

If you are doing primarily application programming, possibly you are going to work with Java with medium performance, but it is quick and robust. So, you are not looking at a very high performance there. Because in an application, you are interacting with a human being who herself is probably quite slow.

If you are primarily doing web programming, high portability, and so on, you are probably working with Python. So, it is low performance, but it has a very high portability, very easy to program in and so on, so forth.

(Refer Slide Time: 10:50)



Module M01

Partha Pratim Das

Objectives & Outline

Know Your C/C++

Evolution & Comparison

Why learn C/C++?

Standards

Know Your Course

Objectives

Prerequisites

Outline

Modules

Tutorials

Evaluation

Text Books & References

Tools

Module Summary

Why learn C/C++?

- C++ is used in development of **Core Software**
  - *Databases*: Oracle, MySQL, MongoDB, MemSQL, etc. used for YouTube, Twitter, Facebook, etc.
  - *OS*: Windows, Linux, Android, Ubuntu, iOS, etc. are written in a combination of C and C++
  - *Compilers / VMs / Tools*: GNU Compiler Collection (GCC); JVM, PVM; MATLAB, IDE
  - *Web Browsers*: Chrome, Firefox, Safari, etc.
  - *Graphic Engine*: Applications in image processing, computer vision, screen recorders, games etc.
  - *Embedded Systems*: Smart watches, MP3 players, GPS systems, etc.
- C++ has **Core Strengths** like
  - *Fast*, *Portable*, and *Scalable*
  - Offers multiple levels of *Abstraction*: hardware to objects to meta-programs
  - *Multi-Paradigms*: Imperative / Procedural (C / Python), Object-Oriented (Algol / Java), Functional (LISP), Generic / Meta-Programming (template, lambda), Concurrent (Java)
- C++ has a **Large Community**
- C++ has **Abundant Library Support (STL)**
- C++ skills attract **High Salary**
- **Caveat**
  - It takes more time to be skilled in C++ compared to, say, Python due to its complexity and diversity
  - It is better to use Java / Python for simple front-end applications that are not performance critical
  - C++ is not best suited for front-end graphics applications for the lack of graphics library

Source: Top 10 Reasons to Learn C++, GeeksforGeeks, 2019

Programming in Modern C++ Partha Pratim Das M01.8

So, choosing the right language is very important, which is also something that in terms of C++, we will have to keep in mind. So, focusing on C, C++, why learn them? So, there are several reasons. The first is C++ use, is used in the development of core software. I mean a lot of core infrastructure software like databases, talk about Oracle, MySQL, MongoDB, and so on, so forth.

Major part of these core software are written in C++. Operating systems like Windows, Linux, Android, Ubuntu they are written in a combination of C and C++. Compilers are written typically in C++, virtual machines on which Java runs, on which Python runs, MATLAB written in C++, web browsers are written in C++, graphics engines, embedded systems. So, C and C++ cover a whole lot of core software infrastructure.

So, if you are focusing on C plus learning C++, you are primarily focusing on these varied kinds of infrastructure software. Then C++ have some core strengths, like it is fast, it is one of the fastest, very close to C. It is portable, more and more modern features are being added, C++ is becoming more portable. It is highly scalable, it can be used in a very small system, small application, as well as it can be used in a moderate to very large application.

Then it offers multiple levels of abstraction. And we will talk more about this as we go forward. Because when you program, you program at multiple levels, so you have hardware, very close to hardware programming to objects, to meta programs as to where you just think about the algorithm and want something to happen in terms of the code and so on. Last, but not the least C++ is multi paradigm.

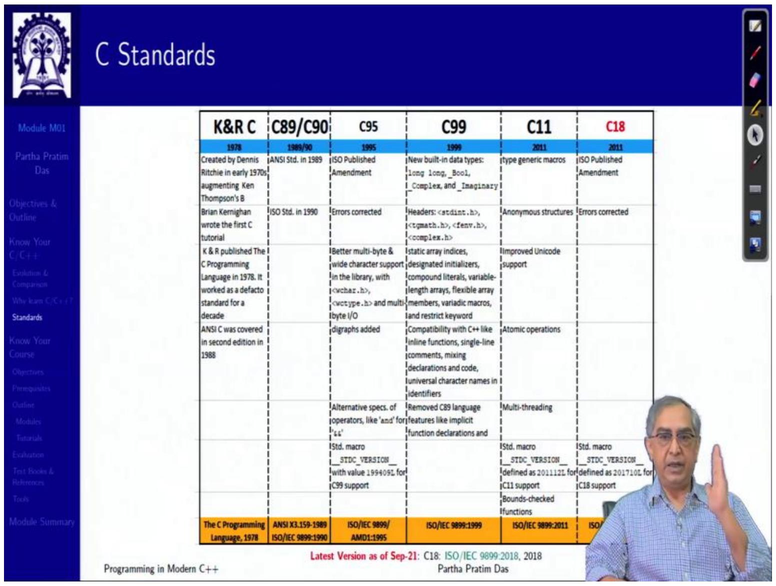
You saw in the graph chart of different evolutionary languages, that some languages are imperative like C. Some are object oriented, like Java, some are functional like Lisp, and so on. C++ has all of these together, it is it has got a very full support of imperative programming with a strong support of object oriented programming. And the modern C++ has introduced functional programming as well.

And it has metaprogramming at the same time, which means that it can write, you can write programs, which generate multiple programs by themselves. And then, again, modern C++ has introduced concurrency features. So, you can do concurrent programming as in Java. So, this multi paradigm support really helps you to go with one language and do a wide variety of jobs. In addition, C++ has a large community, it has abundant library support.

So, many things that you want to do, you can just use the library, and C++ scales attract high salary. I am sure you are learning all these to get a good, good placement, get a good job, and good pay package, for that C++ salaries are skilled salaries are usually higher than many others. At the same time, you must keep in mind that it takes more time to be skilled in C++ compared to say Python.

And it is better to use Java or Python if you are just doing front end applications. And also keep in mind that C++ is not best suited for graphics, front end graphics application. It is good for backend graphics engine where you do a lot of computation but it is not good for front end graphics, because C++ yet does not support a graphics library directly.

(Refer Slide Time: 14:52)



C Standards

Module M01

Partha Pratim Das

Objectives & Outline

Know Your C/C++

Standards

Know Your Course

Module Summary

K&R C	C89/C90	C95	C99	C11	C18
1978	1989/90	1995	1999	2011	2011
Created by Dennis Ritchie in early 1970s augmenting Ken Thompson's B	ANSI Std. in 1989	ISO Published Amendment	New built-in data types: `long long`, `_Bool`, `_Complex`, and `_Imaginary`	type generic macros	ISO Published Amendment
Brian Kernighan wrote the first C tutorial	ISO Std. in 1990	Errors corrected	Headers: `<stdint.h>`, `<tgmath.h>`, `<fenv.h>`, `<complex.h>`	Anonymous structures	Errors corrected
K & R published The C Programming Language in 1978. It worked as a defacto standard for a decade		Better multi-byte & wide character support in the library, with `<wchar.h>`, `<wctype.h>` and multi-byte I/O	static array indices, designated initializers, compound literals, variable-length arrays, flexible array members, variadic macros, and restrict keyword	Improved Unicode support	
ANSI C was covered in second edition in 1988		digraphs added	Compatibility with C++ like inline functions, single-line comments, mixing declarations and code, universal character names in identifiers	Atomic operations	
		Alternative specs. of operators, like '`and`' for '`&&`'	Removed C89 language features like implicit function declarations and	Multi-threading	
		Std. macro `__STDC_VERSION__` with value `199409L` for C99 support		Std. macro `__STDC_VERSION__` defined as `201112L` for C11 support	Std. macro `__STDC_VERSION__` defined as `201710L` for C18 support
				Bounds-checked functions	
The C Programming Language, 1978	ANSI X3.159-1989 ISO/IEC 9899:1990	ISO/IEC 9899/ AMD1:1995	ISO/IEC 9899:1999	ISO/IEC 9899:2011	ISO

Latest Version as of Sep-21: C18: ISO/IEC 9899:2018, 2018

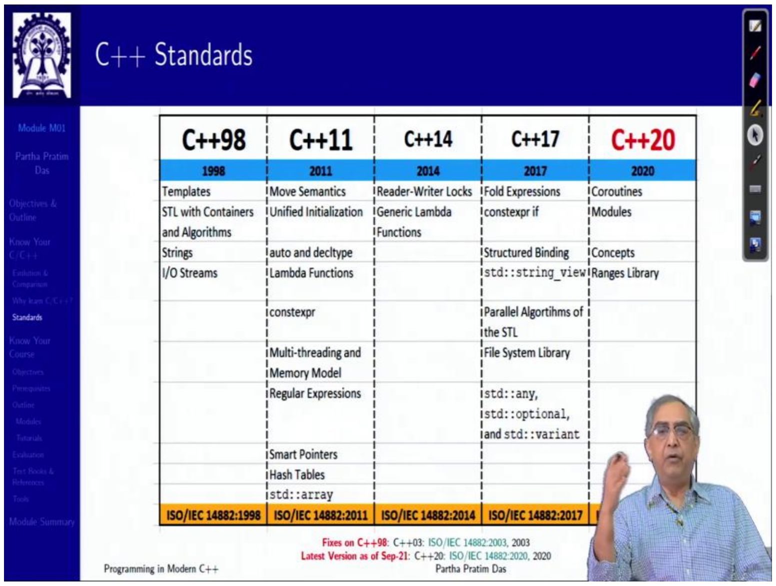
Programming in Modern C++ Partha Pratim Das

So, with these words, let me just present to you the evolution of standards in C and C++. So, this is how it started. So, as you know, Kernighan and Ritchie are known to be, it is primarily Ritchie who created, Dennis Ritchie who created C in early 70s. And for a long time, there has been no or no kind of standardization, but till they wrote the book, the C programming language, which all of you must have read.

And that kind of became known as K and RC, Kernighan and Ritchie C about from about 1978. And then, the ANSI standardization, the basic standardization process started, and in 89 90, the first ANSI standard in 89, the first answer standard was published on C and ISO standard International Standards Organization standard which standardizes every language came in 1990. So, that kind of is the base C language that we talk about.

But then there have been lot of further evolutions, we have C95, C99, and C11, the 2011 version. I have given here a few pointers of the kind of features that have got added to the language as we have moved. And C18 is the latest, though C18 does not significantly add anything on top of C11. It only kind of fixes some of the bugs in that earlier standard.

(Refer Slide Time: 16:36)



C++ Standards

Module M01

Partha Pratim Das

Objectives & Outline

Know Your C/C++

Evolution & Comparison

Why learn C/C++?

Standards

Know Your Course

Objectives

Prerequisites

Outline

Modules

Tutorials

Evaluation

Text Books & References

Tools

Module Summary

C++98	C++11	C++14	C++17	C++20
1998	2011	2014	2017	2020
Templates	Move Semantics	Reader-Writer Locks	Fold Expressions	Coroutines
STL with Containers and Algorithms	Unified Initialization	Generic Lambda Functions	constexpr if	Modules
Strings	auto and decltype		Structured Binding	Concepts
I/O Streams	Lambda Functions		`std::string_view`	Ranges Library
	constexpr		Parallel Algortihms of the STL	
	Multi-threading and Memory Model		File System Library	
	Regular Expressions		`std::any`, `std::optional`, and `std::variant`	
	Smart Pointers			
	Hash Tables			
	`std::array`			
ISO/IEC 14882:1998	ISO/IEC 14882:2011	ISO/IEC 14882:2014	ISO/IEC 14882:2017	[illegible]

Fixes on C++98: C++03: ISO/IEC 14882:2003, 2003

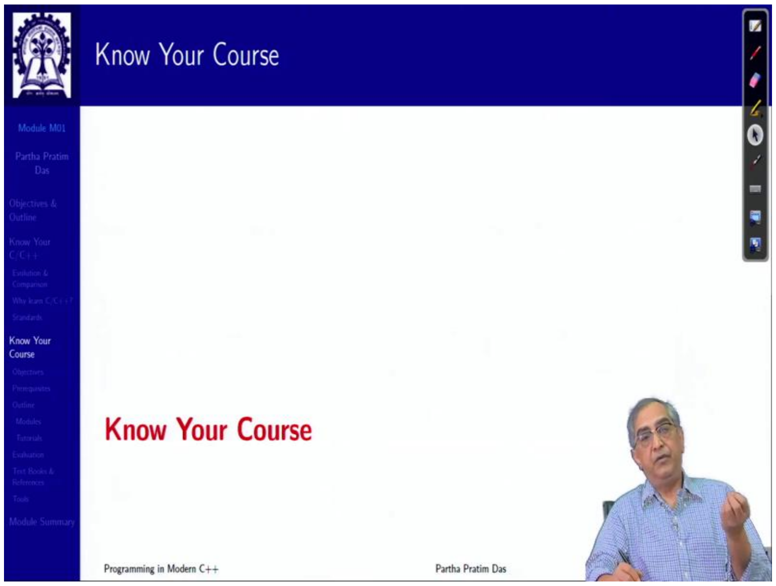
Latest Version as of Sep-21: C++20: ISO/IEC 14882:2020, 2020

Programming in Modern C++ Partha Pratim Das

In contrast, C++ start designed by Bjarne Stroustrup also evolved informally for quite some time till the first standard was created in 1998. And this is called C++ 98. You may have heard about C++ 03 also, which is a revision of this standard which was done in 2003. But that revision did not add any significant feature, it just fixed problems is in C++ 98. Then, the major change happened much later in 2011, when a whole lot of the modern features got started.

So, it is a good point to note here that in Swayam NPTEL earlier, we used to have a course on programming in C++, that course, that eight weeks course focused on C++98, C++03 up to that level. But in this present course, which is programming in C++, programming in modern C++, we are going to focus not only on 98, 03, but we are going to take you primarily through C++ 11 and further. Then we after C++11 we had some minor, additions in C++17, 14, C++17 and C++20, which is a current version.

(Refer Slide Time: 18:08)



Know Your Course

Module M01

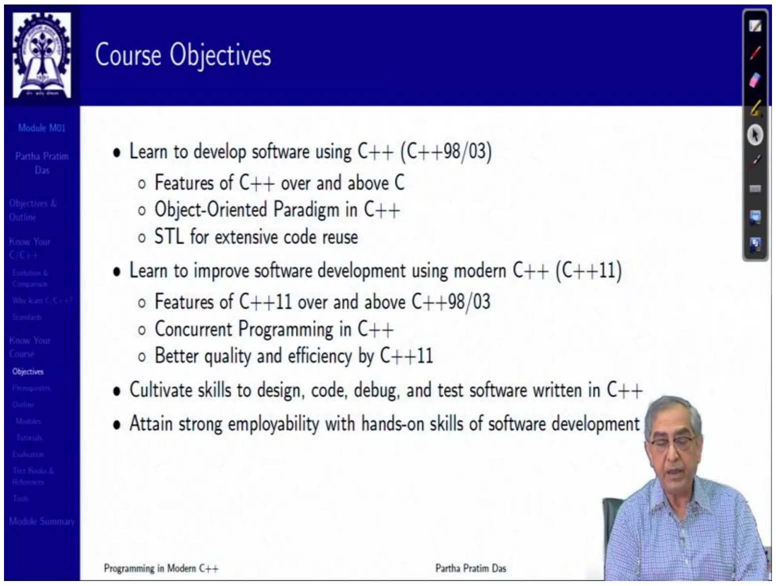
Partha Pratim Das

Know Your Course

Programming in Modern C++ Partha Pratim Das

So, remember that we will, we will primarily here you deal with in terms of these different C and C++ versions and I will, when I talk about different features, I will tell you exactly which particular language standard will that feature be available or effective.

(Refer Slide Time: 18:27)



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

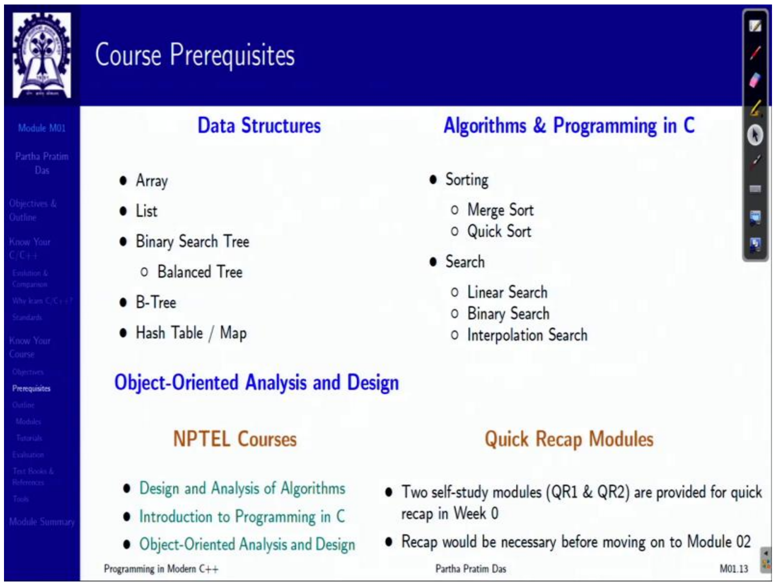
Programming in Modern C++ Partha Pratim Das

Now, going over to give you a glimpse of the course, this is our course objective. Our course objective is certainly to learn to develop a software using C++, by which I mean C++98/03. So, features of C++ over C, object oriented paradigm. STL, the Standard Template Library extensive use is a core objective to learn. Further to go into the modern C++, we want to learn as to how software development is being improved with this modern C++, which is C++11.

So, what are the C++11 features over and above C++98? Primarily the concurrent programming in C++ functional programming and so on, better quality and efficiency and so on. And it is just not learning the language, the objective is to cultivate skills to design code, debug and test software it means C++, that is what has to be your focus.

Just because you know the language well, you will not be getting good offers from companies, you have to have the skills which you have to develop through practice problem solving and so on so that you can attain a strong employability and you need the hands-on skill and strong employability is a core objective of this course.

(Refer Slide Time: 19:42)



Course Prerequisites

Module M01

Partha Pratim Das

Objectives & Outline

Know Your C/C++

Evolution & Comparison

Why learn C/C++?

Standards

Know Your Course

Objectives

Prerequisites

Outline

Modules

Tutorials

Evaluation

Text Books & References

Tools

Module Summary

Data Structures

- Array
- List
- Binary Search Tree
  - Balanced Tree
- B-Tree
- Hash Table / Map

Algorithms & Programming in C

- Sorting
  - Merge Sort
  - Quick Sort
- Search
  - Linear Search
  - Binary Search
  - Interpolation Search

Object-Oriented Analysis and Design

NPTEL Courses

- Design and Analysis of Algorithms
- Introduction to Programming in C
- Object-Oriented Analysis and Design

Quick Recap Modules

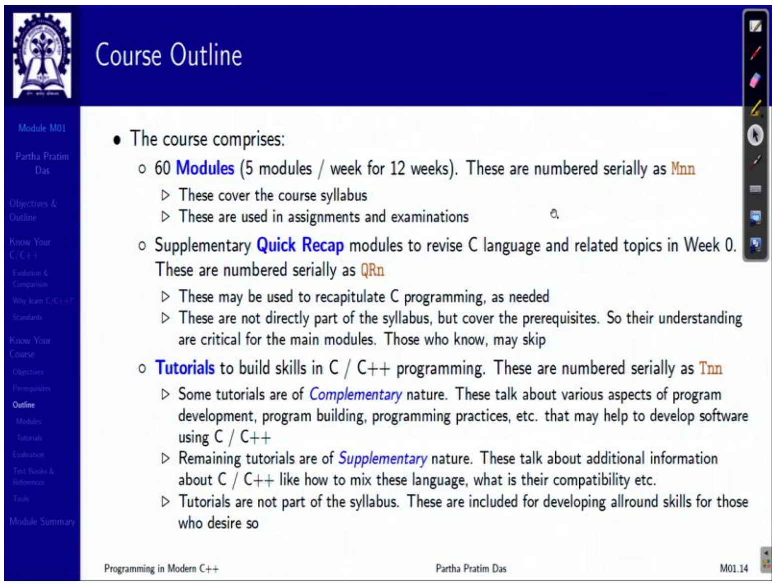
- Two self-study modules (QR1 & QR2) are provided for quick recap in Week 0
- Recap would be necessary before moving on to Module 02

Programming in Modern C++ Partha Pratim Das M01.13

The prerequisites as I have, you already know are certainly data structures or basic data structures a list is given here. Algorithms and programming in C, which you must know. And it will be good to have some idea about object oriented analysis and design, but it is not mandatory, we will introduce that in the process.

So, here I have mentioned some NPTEL courses, which you can go through to learn about to, recap these prerequisites if you are not familiar already. And we will also as a part of week 0 provide some specific modules for self-study, particularly on the various aspects of C.

(Refer Slide Time: 20:28)



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 allround skills for those who desire so

Programming in Modern C++ Partha Pratim Das M01.14

Now, this is our course outline, the detailed outline of weeks and modules are already given in week 0. So, we will have as a standard, we will have 60 modules, 5 modules per week for 12 weeks and they will be numbered by M followed by the serial number, which will cover the course syllabus, the assignments and examinations will be based on this. We have supplementary quick recap modules as I mentioned in week 0, to recap, C if you are already not on top of it. So, it is up to you to use those modules and enhance your C skills.

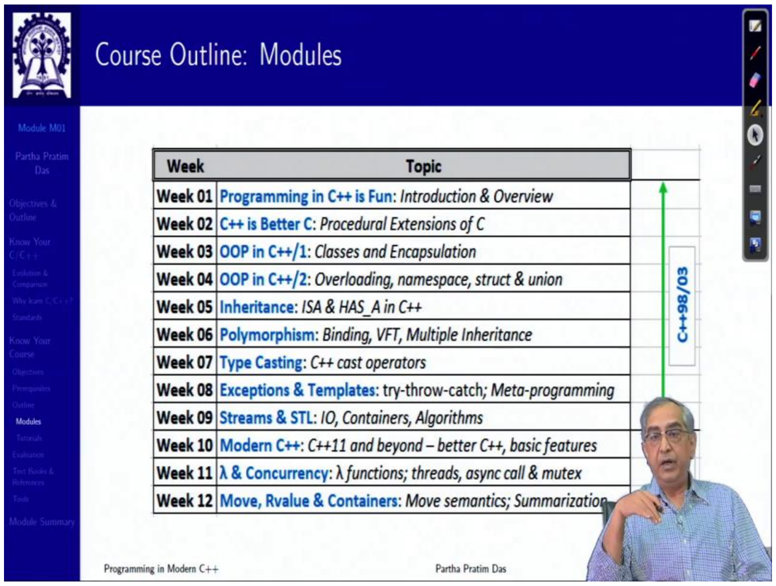
In addition, we will have a number of tutorials to build in C, C++ programming, they are numbered with T. And some tutorials are complimentary in nature, complimentary in the sense that they do not really talk about the language, but it talks about the program development aspects. When we run the earlier course, often students used to ask tell us how to build a program, tell us how to, how to organize the source of a program and so on so forth, the what are the good practices and so on.

So, these are the, of complementary in nature. These are not part of the syllabus, but they are primarily to help you gain your skills. The remaining tutorials are of supplementary in nature, which mean that they talk about the language, but not the core part, which is included in the syllabus. Things like how do you write mixed language programs in C and C++? How compatible they are?

If you write a program in C, we say that C++ has C, but that is at a very high philosophical level. It is not guaranteed, that if you have written this program in C, it will run exactly, it will compile in C++, first of all, it may not. And even if it compiles, it does, there is not a guarantee that it will execute and show you the same behavior.

So, there are compatibility issues, which you will have to know and be aware of if you want to become a good C++ programmer. But all of the tutorials are just for your development for your help. They are not part of the course syllabus. So, we will not have assignments or questions, examination questions on them.

(Refer Slide Time: 22:45)



Course Outline: Modules

Module M01

Partha Pratim Das

Objectives & Outline

Know Your C/C++

Evolution & Comparison

Why learn C/C++?

Standards

Know Your Course

Objectives

Prerequisites

Outline

Modules

Tutorials

Evaluation

Text Books & References

Tools

Module Summary

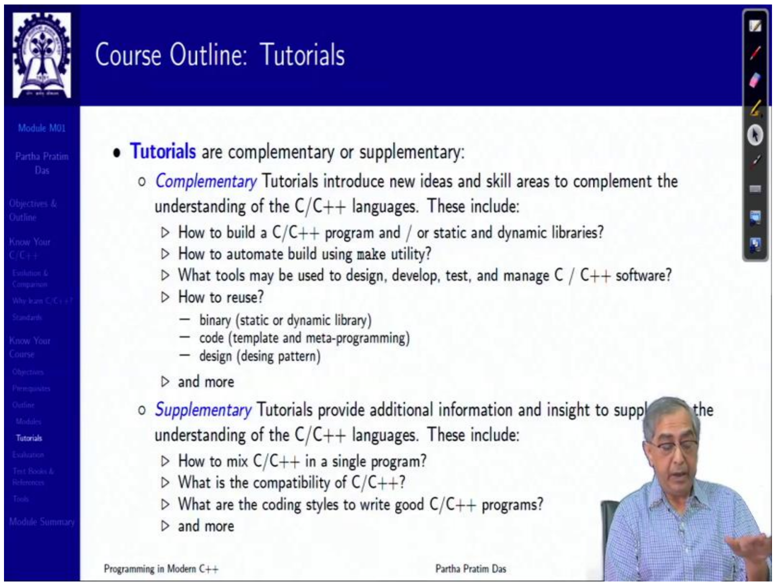
Week	Topic
Week 01	Programming in C++ is Fun: *Introduction & Overview*
Week 02	C++ is Better C: *Procedural Extensions of C*
Week 03	OOP in C++/1: *Classes and Encapsulation*
Week 04	OOP in C++/2: *Overloading, namespace, struct & union*
Week 05	Inheritance: *ISA & HAS_A in C++*
Week 06	Polymorphism: *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

Programming in Modern C++ Partha Pratim Das

This is the overall module outline. So, of different weeks, and you can see that the first 9 weeks are focused on C++98, 03, which is kind of what the earlier course used to do. And the remaining 3 weeks focus on the really modern part, which is C++11, the evolutions of some of the very important features and some of the efficiency aspects like move semantics, rvalue, move constructor, rvalue semantics, and so on, so forth. So, and in again, module 0 in week 0, gives you the details of the different modules in every week.

(Refer Slide Time: 23:29)



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 suppl 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

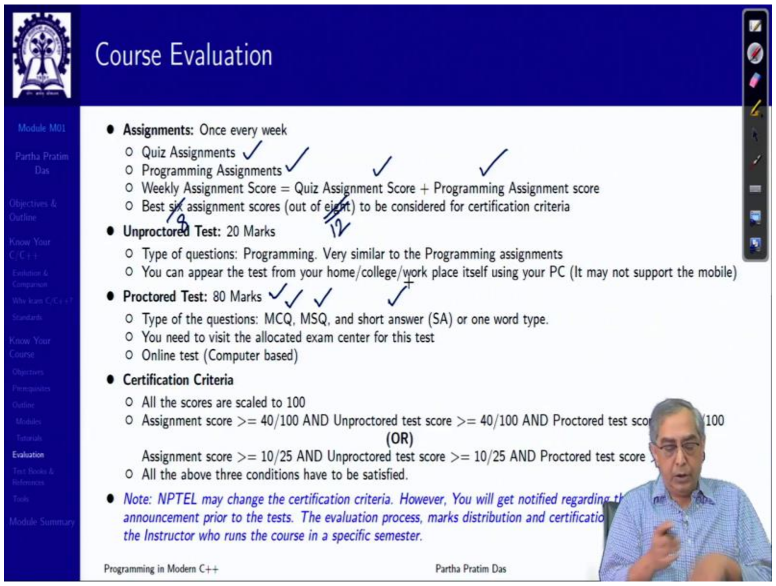
In terms of tutorial, as I said, that are complementary ones and the supplementary ones, the complementary ones, as you can see, will include things like how to build C program, how to build static and dynamic libraries, how to make use of the make utility, which is a very great utility to build your programs easily. Then about different tools that you may be using how to reuse programs at different levels at a binary level, at a code level, at a design level, and so on.

So, these are the complimentary tutorials we will supply you with at different points. And you may go through them, practice them, they are more practice oriented, you may go through and practice them so that you can get really skilled better in terms of program development. And the supplementary tutorials, as I mentioned, we will talk about different extensional features of these languages.

Primarily, how to make C and C++ in a single program, which is often an issue that comes up because you are not implementing a project from scratch most of the time. There is something already existing possibly that is in some version of C and you are writing in some version of C++. Now how do you make these programs and how to make them work together? Then what is the compatibility? If you take a C program and compile in C++, what would you expect?

What are the, what are the coding styles that are good for C, what are the coding styles that are good for C++ and so on the industry practices. So, we are, tutorials are primarily focused on practice that you must have, beyond doing these modules and doing the assignments. See, if you have to get skilled, it is not enough to just go through the language and go through the quiz and examination that will give you the score, the certificate, but your real value of employability will come from practice and these tutorials are focused towards doing that.

(Refer Slide Time: 25:35)



Course Evaluation

Module M01

Partha Pratim Das

Objectives & Outline

Know Your C/C++

Evolution & Comparison

Why learn C/C++?

Standards

Know Your Course

Objectives

Prerequisites

Outline

Modules

Tutorials

Evaluation

Text Books & References

Tools

Module Summary

- **Assignments:** Once every week ✓
  - Quiz Assignments ✓
  - Programming Assignments ✓
  - Weekly Assignment Score = Quiz Assignment Score + Programming Assignment score
  - Best ~~six~~ 8 assignment scores (out of ~~eight~~ 12) to be considered for certification criteria
- **Unproctored Test:** 20 Marks
  - Type of questions: Programming. Very similar to the Programming assignments
  - You can appear the test from your home/college/work place itself using your PC (It may not support the mobile)
- **Proctored Test:** 80 Marks ✓ ✓ ✓
  - Type of the questions: MCQ, MSQ, and short answer (SA) or one word type.
  - You need to visit the allocated exam center for this test
  - Online test (Computer based)
- **Certification Criteria**
  - All the scores are scaled to 100
  - Assignment score >= 40/100 AND Unproctored test score >= 40/100 AND Proctored test sco 100

    (OR)

    Assignment score >= 10/25 AND Unproctored test score >= 10/25 AND Proctored test score
  - All the above three conditions have to be satisfied.
- *Note: NPTEL may change the certification criteria. However, You will get notified regarding th announcement prior to the tests. The evaluation process, marks distribution and certificatio the Instructor who runs the course in a specific semester.*

Programming in Modern C++ Partha Pratim Das

The evaluation is more or less similar to other NPTEL courses, except that this has some programming components. So, if you note, we have a quiz, every week one quiz will be there. And then we have in the quiz we have programming assignments also. So, weekly assignment score will be the sum of the quiz assignment. And as well as the programming assignment and the best 6 typically would be considered out of the 08 this is, this is a typo out of 12, probably it will not be best six, probably it will be best eight.

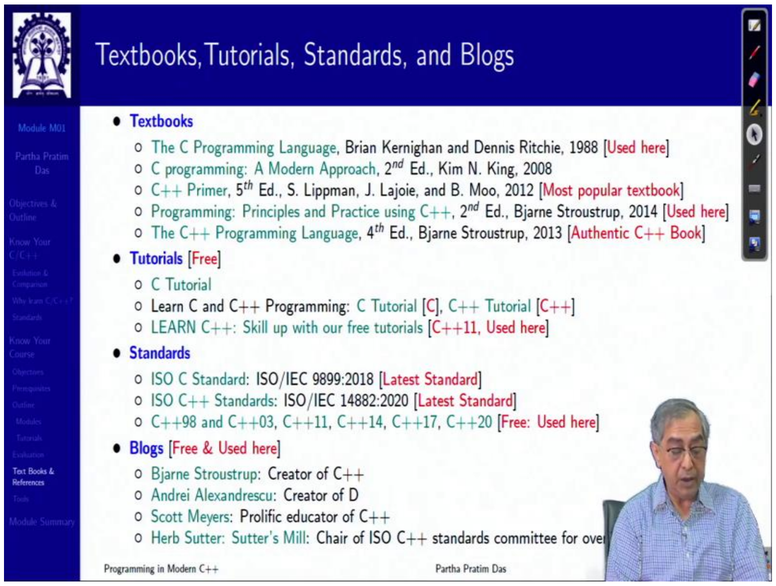
Best 0 out of 12 will be considered for the certification criteria. Now, there will also be an unproctored test, unproctored test is of the programming kind of the programming exercises that we will have. And this is something which you take from your home or workplace or college using a PC, you may not be able to use a mobile because it uses, it will not have all the components to do programming there.

And there will be about 20 marks on that. Then there is a proctored test which is the main for the certification. Which will have multiple choice multiple select and short answer type of questions as you have already seen in the assignments. And this is proctored. So, you will have to go to an allotted examination center and take this test. And the test will be online it is not on paper. But it has to be taken in that center itself.

So, with all these your overall scoring will get decided for your certification. There are certain, certain criteria are given, like the all scores are scaled to 100 and assignment score greater than 40 out of 100 or unproctored such and such and so on is the certification criteria. But keep in mind that this is, these are I mean, your overall structure of evaluation as well as certification criteria are not frozen forever.

Every time the course is done, the NPTEL will announce a certification criterion, you must follow that very carefully. And the instructor who is running the course that time will decide on what will be the structure of this evaluation. So, this is the overall evaluation information.

(Refer Slide Time: 27:59)



Textbooks, Tutorials, Standards, and Blogs

- Textbooks
  - The C Programming Language, Brian Kernighan and Dennis Ritchie, 1988 [Used here]
  - C programming: A Modern Approach, 2nd Ed., Kim N. King, 2008
  - C++ Primer, 5th Ed., S. Lippman, J. Lajoie, and B. Moo, 2012 [Most popular textbook]
  - Programming: Principles and Practice using C++, 2nd Ed., Bjarne Stroustrup, 2014 [Used here]
  - The C++ Programming Language, 4th Ed., Bjarne Stroustrup, 2013 [Authentic C++ Book]
- Tutorials [Free]
  - C Tutorial
  - Learn C and C++ Programming: C Tutorial [C], C++ Tutorial [C++]
  - LEARN C++: Skill up with our free tutorials [C++11, Used here]
- Standards
  - ISO C Standard: ISO/IEC 9899:2018 [Latest Standard]
  - ISO C++ Standards: ISO/IEC 14882:2020 [Latest Standard]
  - C++98 and C++03, C++11, C++14, C++17, C++20 [Free: Used here]
- Blogs [Free & Used here]
  - Bjarne Stroustrup: Creator of C++
  - Andrei Alexandrescu: Creator of D
  - Scott Meyers: Prolific educator of C++
  - Herb Sutter: Sutter's Mill: Chair of ISO C++ standards committee for ove

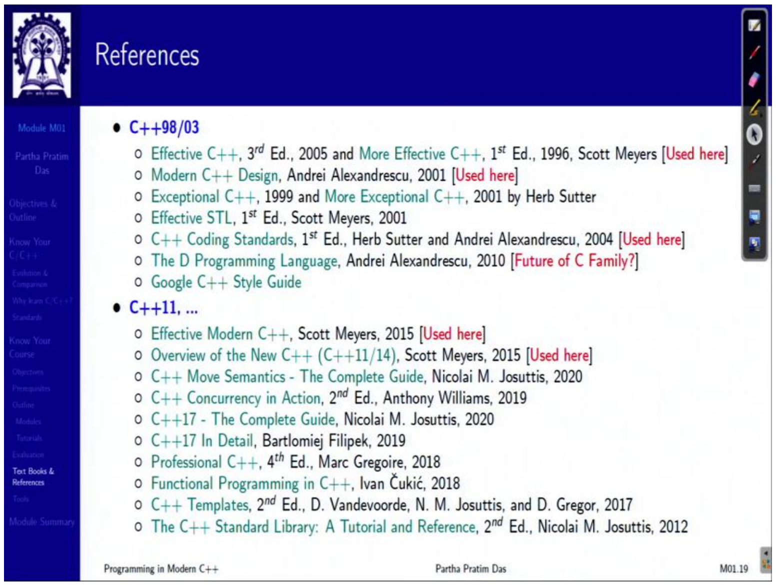
Programming in Modern C++ Partha Pratim Das

Now, coming to textbooks and references, there are many. So, what I have done is in terms of textbooks and tutorials and standards and blogs, even blogs are very important. But you must know which blogs to look at, not all blogs are, give you the right information, some blogs may give you incomplete information, some may give you even wrong information.

So, in terms of textbooks, I have mentioned several textbooks and you can follow any of them. But I have also mentioned as to what is the particular book that we would be following here like for C programming, we will follow the Kernighan and Ritchie’s book. Whereas Lipman's book probably is the most popular and for C++, we will use Stroustrup's book, the latest standards, the blogs.

I heavily rely on these blogs, because these are the people who are leading the C++ development starting from Stroustrup to the creator of D language, which is also in the family now. And Scott Meyer, Herb Sutter these are people who, who actually drive the language and their blogs are very important to follow.

(Refer Slide Time: 29:12)



Module M01

Partha Pratim Das

Objectives & Outline

Know Your C/C++

Evolution & Comparison

Why learn C/C++?

Standards

Know Your Course

Objectives

Prerequisites

Outline

Modules

Tutorials

Evaluation

Text Books & References

Tools

Module Summary

References

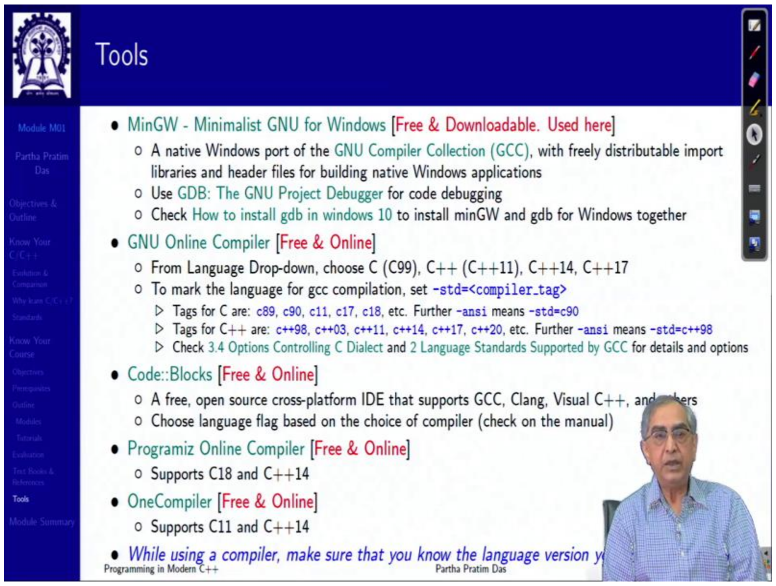
- C++98/03
  - Effective C++, 3rd Ed., 2005 and More Effective C++, 1st Ed., 1996, Scott Meyers [Used here]
  - Modern C++ Design, Andrei Alexandrescu, 2001 [Used here]
  - Exceptional C++, 1999 and More Exceptional C++, 2001 by Herb Sutter
  - Effective STL, 1st Ed., Scott Meyers, 2001
  - C++ Coding Standards, 1st Ed., Herb Sutter and Andrei Alexandrescu, 2004 [Used here]
  - The D Programming Language, Andrei Alexandrescu, 2010 [Future of C Family?]
  - Google C++ Style Guide
- C++11, ...
  - Effective Modern C++, Scott Meyers, 2015 [Used here]
  - Overview of the New C++ (C++11/14), Scott Meyers, 2015 [Used here]
  - C++ Move Semantics - The Complete Guide, Nicolai M. Josuttis, 2020
  - C++ Concurrency in Action, 2nd Ed., Anthony Williams, 2019
  - C++17 - The Complete Guide, Nicolai M. Josuttis, 2020
  - C++17 In Detail, Bartlomiej Filipek, 2019
  - Professional C++, 4th Ed., Marc Gregoire, 2018
  - Functional Programming in C++, Ivan Čukić, 2018
  - C++ Templates, 2nd Ed., D. Vandevoorde, N. M. Josuttis, and D. Gregor, 2017
  - The C++ Standard Library: A Tutorial and Reference, 2nd Ed., Nicolai M. Josuttis, 2012

Programming in Modern C++ Partha Pratim Das M01.19

Several reference books, I have mentioned here in C++98, 03 and with special marking of what is used in the modules here, and C++ also in C++11. So, obviously, you cannot you will not be able to study all of them, but these are just representative so that, if you have to otherwise, even most of what you will need will be covered in the module itself. I mean, you do not specifically need to go to the book.

But I mean you will know that these are coming from this book. So, if you want more details, you can go there and get further clarified. And these are references that you can use if you really want to make an advanced carrier in modern C++.

(Refer Slide Time: 30:01)



Tools

- MinGW - Minimalist GNU for Windows [Free & Downloadable. Used here]
  - A native Windows port of the GNU Compiler Collection (GCC), with freely distributable import libraries and header files for building native Windows applications
  - 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
- GNU Online Compiler [Free & Online]
  - From Language Drop-down, choose C (C99), C++ (C++11), C++14, C++17
  - To mark the language for gcc compilation, set `-std=<compiler_tag>`
    - Tags for C are: `c89`, `c90`, `c11`, `c17`, `c18`, etc. Further `-ansi` means `-std=c90`
    - Tags for C++ are: `c++98`, `c++03`, `c++11`, `c++14`, `c++17`, `c++20`, etc. Further `-ansi` means `-std=c++98`
    - Check 3.4 Options Controlling C Dialect and 2 Language Standards Supported by GCC for details and options
- Code::Blocks [Free & Online]
  - A free, open source cross-platform IDE that supports GCC, Clang, Visual C++, and others
  - Choose language flag based on the choice of compiler (check on the manual)
- Programiz Online Compiler [Free & Online]
  - Supports C18 and C++14
- OneCompiler [Free & Online]
  - Supports C11 and C++14
- *While using a compiler, make sure that you know the language version y*

Programming in Modern C++ Partha Pratim Das

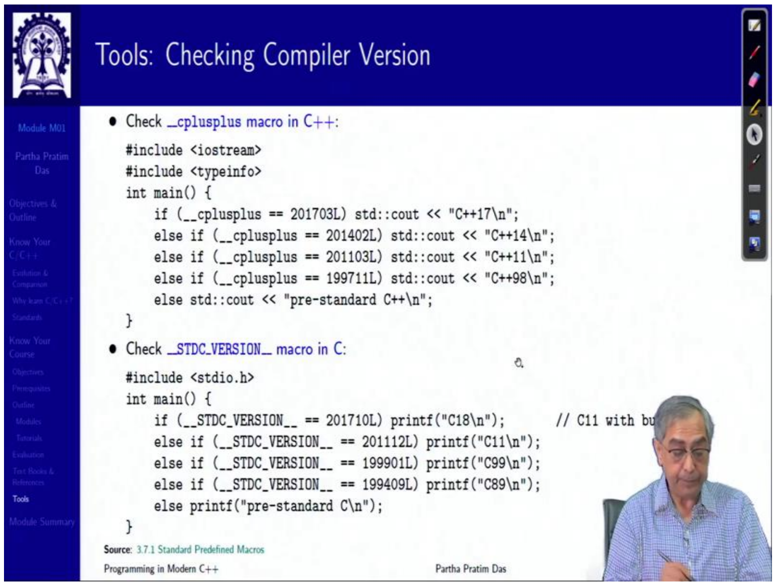
Finally, you need tools. So, we will be on GNU tools. So, we will use GCC for primarily, for our examples and the results that you will see the outputs that you will see or the behavior that we will see in different slides will be primarily from the GNU compiler. So, if you have Linux, you will have the GCC and that with the debugger GDB.

If you are on Windows, because I believe many of you would be on Windows, so, it will be minimalistic GNU for Windows, minGW as it is called, it is free downloadable. I have given the links here as to where to download it from and how to install video and so on. So, please use that and install it in your system. Whether you are, if you are using Linux, you will have that, if you are using Windows, install it so that you have the built debug tools available with you.

For quick checkup, of course, you can use multiple different the following ones are genuine line compiler, code blocks, programiz, one compiler these are all online free services, they are like software as a service model. So, you can just put your code there and check for different versions, different because, there are many of them support variety of different versions of the language there.

So, you can use for those checking, but it will be most critical to install Linux or minGW on Windows to be ready to run your code and get hands on.

(Refer Slide Time: 31:38)



Tools: Checking Compiler Version

Module M01

Partha Pratim Das

Objectives & Outline

Know Your C/C++

Evolution & Comparison

Why learn C/C++?

Standards

Know Your Course

Objectives

Prerequisites

Outline

Modules

Tutorials

Evaluation

Text Books & References

Tools

Module Summary

- Check __cplusplus macro in C++:

```
#include <iostream>
#include <typeinfo>
int main() {
    if (__cplusplus == 201703L) std::cout << "C++17\n";
    else if (__cplusplus == 201402L) std::cout << "C++14\n";
    else if (__cplusplus == 201103L) std::cout << "C++11\n";
    else if (__cplusplus == 199711L) std::cout << "C++98\n";
    else std::cout << "pre-standard C++\n";
}
```

- Check __STDC_VERSION__ macro in C:

```
#include <stdio.h>
int main() {
    if (__STDC_VERSION__ == 201710L) printf("C18\n");      // C11 with bu
    else if (__STDC_VERSION__ == 201112L) printf("C11\n");
    else if (__STDC_VERSION__ == 199901L) printf("C99\n");
    else if (__STDC_VERSION__ == 199409L) printf("C89\n");
    else printf("pre-standard C\n");
}
```

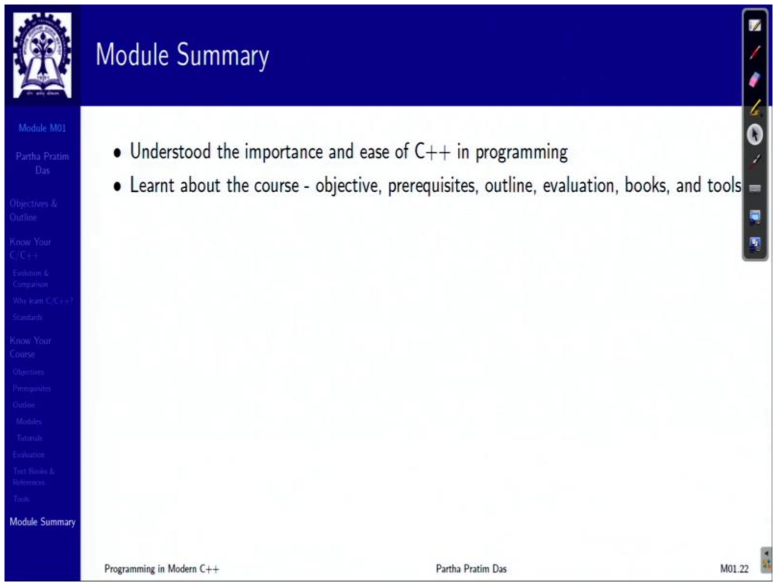
Source: 3.7.1 Standard Predefined Macros

Programming in Modern C++ Partha Pratim Das

Now when you are running programs, you will always need to know which particular version of C or C++ we are compiling for. So, here I have given the code snippets which you can use and know exactly which version is being used, if, if you have not explicitly specified and want to know well, I am using a compiler.

So, what version is it compiling for? So, you can use this code snippet and this will tell you the magic numbers there which every language standard embeds. And with that you will be able to know what particular version you are using.

(Refer Slide Time: 32:13)



Module Summary

- Understood the importance and ease of C++ in programming
- Learnt about the course - objective, prerequisites, outline, evaluation, books, and tools

Module M01

Partha Pratim Das

Objectives & Outline

Know Your C/C++

Evolution & Comparison

Why learn C/C++?

Standards

Know Your Course

Objectives

Prerequisites

Outline

Modules

Tutorials

Evaluation

Text Books & References

Tools

Module Summary

Programming in Modern C++ Partha Pratim Das M01.22

So, that is all for this module. So, we have tried to give you a basic idea about the importance and ease of programming in C++ with a, know your language as well as know your course outline. Look forward to lots of interactions with you and look forward to an exciting, exciting time with the remaining 59 modules where we really deal with modern C++. Thank you very much. See you in the next module.