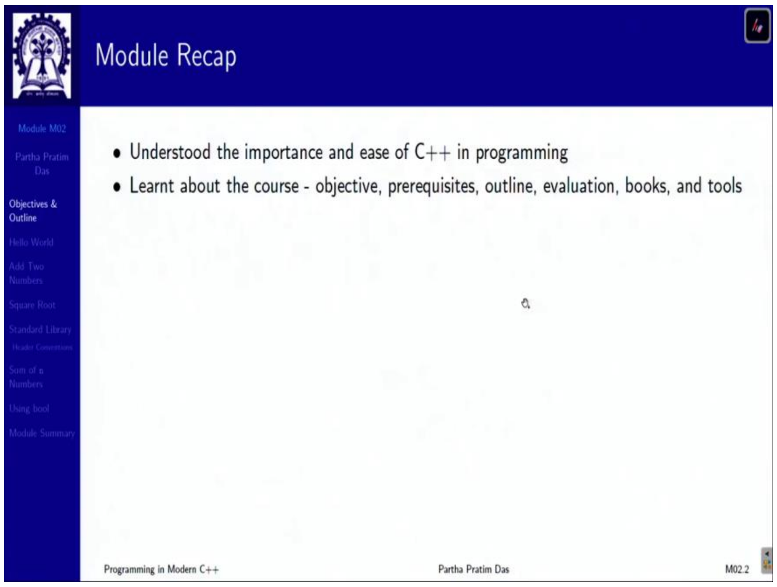
Programming in Modern C++
Professor. Partha Pratim Das
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur
Lecture No. 05
IO and Loop

Welcome to Programming in Modern C++, we are in week 1 and module 2.

(Refer Slide Time: 00:34)

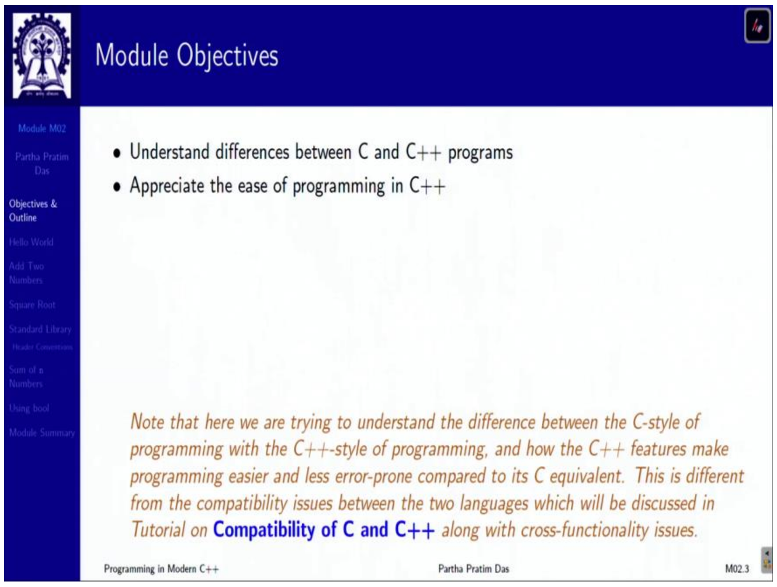


Module Recap

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

So, in the last module, we have understood the basic importance of C++ programming, particularly the modern aspects. And we learnt about the course overall.

(Refer Slide Time: 00:45)



Module Objectives

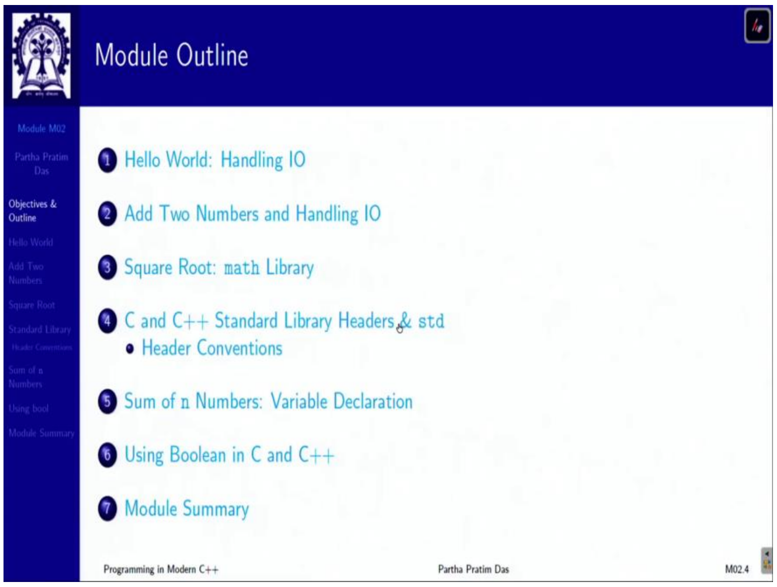
- Understand differences between C and C++ programs
- Appreciate the ease of programming in C++

Note that here we are trying to understand the difference between the C-style of programming with the C++-style of programming, and how the C++ features make programming easier and less error-prone compared to its C equivalent. This is different from the compatibility issues between the two languages which will be discussed in Tutorial on **Compatibility of C and C++** *along with cross-functionality issues.*

In this module, we would like to start by highlighting the differences between C and C++ programs, to make you understand the ease of programming in C++. So, when we discuss that please, please keep in mind that this is not related to compatibility of these two languages, that will be covered in a separate tutorial.

Here what we are going to do is, you write, we will take programs in C and write their appropriate C++ equivalent and try to show you how it becomes easier to write more efficient, safer, less error prone code in C++.

(Refer Slide Time: 01:30)



Module Outline

Module M02

Partha Pratim Das

Objectives & Outline

Hello World

Add Two Numbers

Square Root

Standard Library

Header Conventions

Sum of n Numbers

Using bool

Module Summary

1. Hello World: Handling IO
2. Add Two Numbers and Handling IO
3. Square Root: `math` Library
4. C and C++ Standard Library Headers & `std`
   - Header Conventions
5. Sum of n Numbers: Variable Declaration
6. Using Boolean in C and C++
7. Module Summary

Programming in Modern C++ Partha Pratim Das M02.4

So, there are several, 6 different examples, very common programs, which I am sure in C you must have written all of them or at least most of them. So, I will use them to provide a comparison with the corresponding C++ program.

(Refer Slide Time: 01:44)

Hello World: Handling IO

Module M02

Partha Pratim Das

Objectives & Outline

Hello World

Add Two Numbers

Square Root

Standard Library

Header Conventions

Sum of n Numbers

Using bool

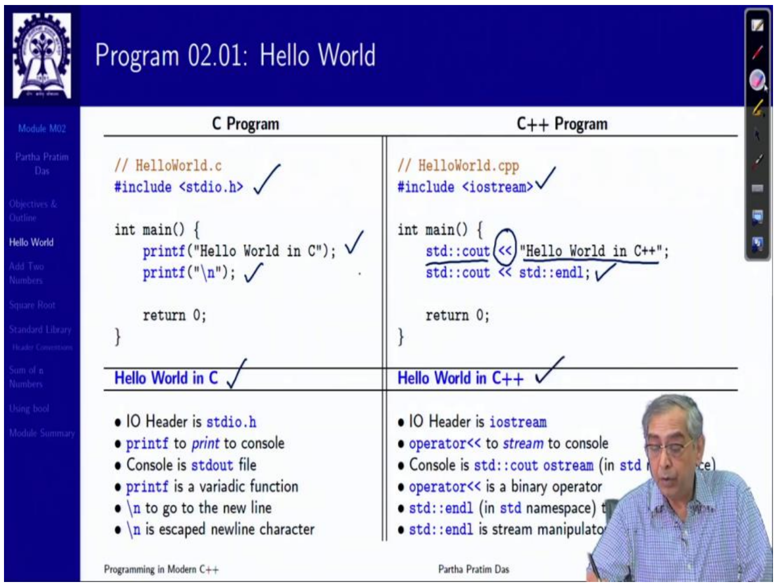
Module Summary

Hello World: Handling IO

Programming in Modern C++ Partha Pratim Das

So, the entire story as the world starts with the Hello World.

(Refer Slide Time: 01:53)



Program 02.01: Hello World

Module M02

Partha Pratim Das

Objectives & Outline

Hello World

Add Two Numbers

Square Root

Standard Library

Header Conventions

Sum of n Numbers

Using bool

Module Summary

C Program

```c
// HelloWorld.c
#include <stdio.h>

int main() {
    printf("Hello World in C");
    printf("\n");

    return 0;
}
```

Hello World in C

- IO Header is stdio.h
- printf to *print* to console
- Console is stdout file
- printf is a variadic function
- \n to go to the new line
- \n is escaped newline character

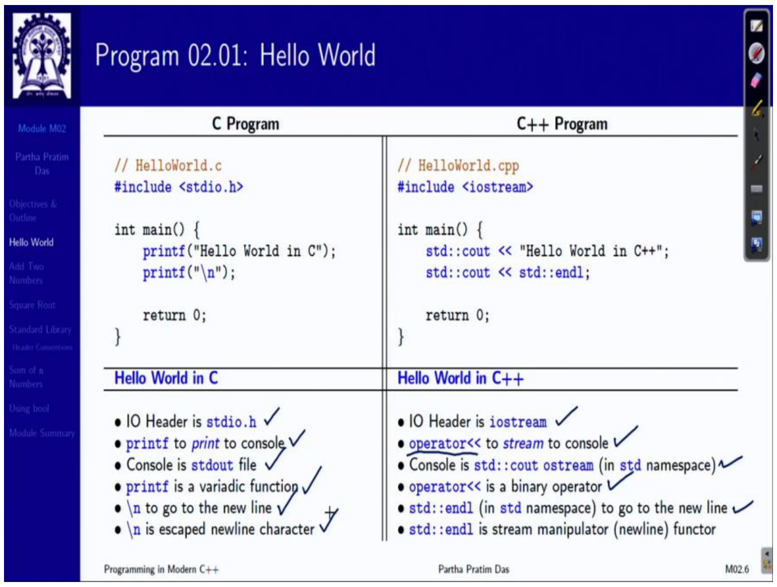
C++ Program

```cpp
// HelloWorld.cpp
#include <iostream>

int main() {
    std::cout << "Hello World in C++";
    std::cout << std::endl;

    return 0;
}
```

Hello World in C++

- IO Header is iostream
- operator<< to *stream* to console
- Console is std::cout ostream (in std ce)
- operator<< is a binary operator
- std::endl (in std namespace) t
- std::endl is stream manipulato

Programming in Modern C++ Partha Pratim Das



Program 02.01: Hello World

Module M02

Partha Pratim Das

Objectives & Outline

Hello World

Add Two Numbers

Square Root

Standard Library

Header Conventions

Sum of n Numbers

Using bool

Module Summary

C Program	C++ Program

`// HelloWorld.c`
`#include <stdio.h>`
 	
`int main() {`
`    printf("Hello World in C");`
`    printf("\n");`
 	
`    return 0;`
`}`
	`// HelloWorld.cpp`
	`#include <iostream>`
	
	`int main() {`
	`    std::cout << "Hello World in C++";`
	`    std::cout << std::endl;`
	
	`    return 0;`
	`}`

Hello World in C	Hello World in C++
• IO Header is stdio.h	• IO Header is iostream
• printf to *print* to console	• operator<< to *stream* to console
• Console is stdout file	• Console is std::cout ostream (in std namespace)
• printf is a variadic function	• operator<< is a binary operator
• \n to go to the new line	• std::endl (in std namespace) to go to the new line
• \n is escaped newline character	• std::endl is stream manipulator (newline) functor

Programming in Modern C++ Partha Pratim Das M02.6

Which is the basic handling of the IO. So, this is the HelloWorld program, which all of you know, so, you have to include stdio.h in C you know that and you do a printf with a new line, that is you say hello world in C. In C++ instead you use a different header called iostream. And note that in iostream there is no h, .h we will talk about this more.

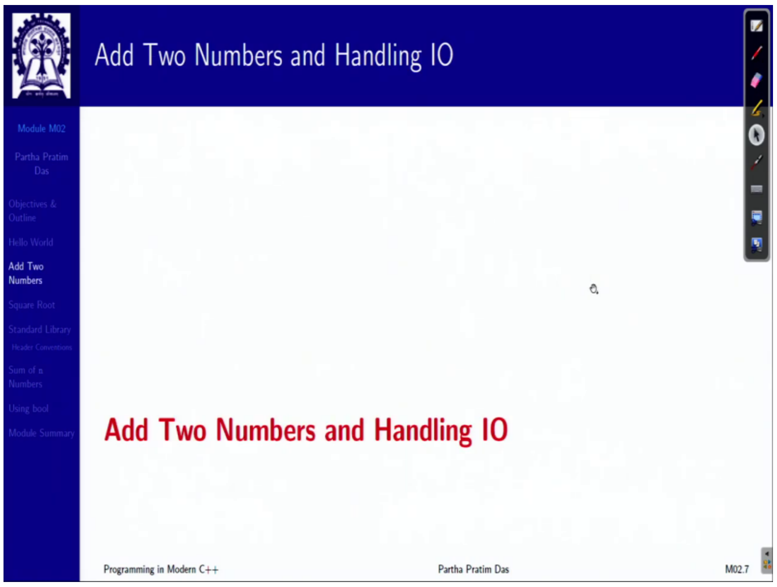
And you use the same string or a similar string to do the hello world, but you use an output streaming operator, this is called and stream to stdout, std::cout, which is actually the standard output. And then you put it a end line marker, which gives you exactly similar output as in here. So, having said that now, if you compare what you see is IO header is different. You are using printf to print to console, which is stdout.

And here use, you are using an operator, operator << called output streaming to stream to the console, the terms are also kind of changing, it is not only about syntax but in typical C you say that I print, in C++ you say I stream. The console is stdout file. Here it is std::cout, ostream O stands for output. And that is in the standard namespace we will talk about what it means.

printf is a variadic function in the sense that it takes any number of arguments after a format string, the format string is important, is mandatory at the beginning, and then you can have any number of them as you, as you know. But this operator is just binary. It just takes one, it takes the stream and what you want to print and does the printing, does a streaming and then returns the same stream, we will learn more.

\n is used for going to newline, here it is std::endl in the std namespace in line in the namespace, where \n is an escape new character, std::endl is a stream manipulator. So, we will see that besides putting a new line it can actually do more stuff. And it is a functor we will talk about this and I am just trying to show you the parallel between these two.

(Refer Slide Time: 04:48)



Add Two Numbers and Handling IO

Module M02

Partha Pratim Das

Objectives & Outline

Hello World

Add Two Numbers

Square Root

Standard Library

Header Conventions

Sum of n Numbers

Using bool

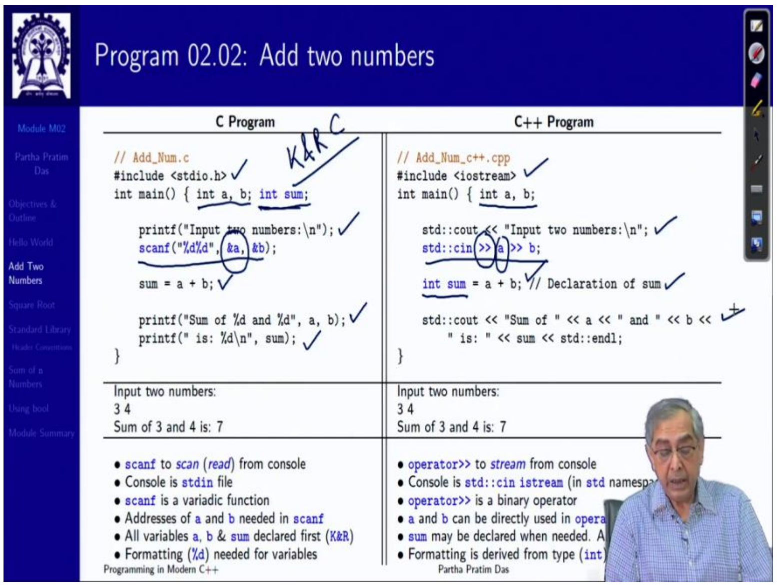
Module Summary

Add Two Numbers and Handling IO

Programming in Modern C++ Partha Pratim Das M02.7

So, let us go to the next example, which is adding two numbers and handling the IO again.

(Refer Slide Time: 04:56)



Program 02.02: Add two numbers

Module M02

Partha Pratim Das

Objectives & Outline

Hello World

Add Two Numbers

Square Root

Standard Library

Header Conventions

Sum of n Numbers

Using bool

Module Summary

C Program

K&R C

```
// Add_Num.c
#include <stdio.h> ✓
int main() { int a, b; int sum;

    printf("Input two numbers:\n"); ✓
    scanf("%d%d", &a, &b);

    sum = a + b; ✓

    printf("Sum of %d and %d", a, b); ✓
    printf(" is: %d\n", sum); ✓
}
```

```
Input two numbers:
3 4
Sum of 3 and 4 is: 7
```

- scanf to *scan* (*read*) from console
- Console is stdin file
- scanf is a variadic function
- Addresses of a and b needed in scanf
- All variables a, b & sum declared first (K&R)
- Formatting (%d) needed for variables

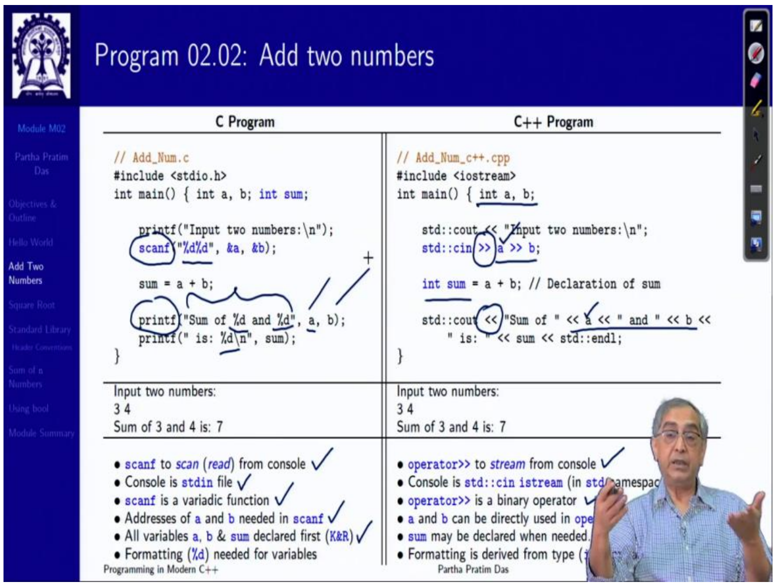
C++ Program

```
// Add_Num_c++.cpp
#include <iostream> ✓
int main() { int a, b;

    std::cout << "Input two numbers:\n"; ✓
    std::cin >> a >> b;

    int sum = a + b; ✓// Declaration of sum ✓

    std::cout << "Sum of " << a << " and " << b << ✓
        " is: " << sum << std::endl;
}
```

```
Input two numbers:
3 4
Sum of 3 and 4 is: 7
```

- operator>> to *stream* from console
- Console is std::cin istream (in std namesp
- operator>> is a binary operator
- a and b can be directly used in opera
- sum may be declared when needed. A
- Formatting is derived from type (int

Programming in Modern C++ Partha Pratim Das



Program 02.02: Add two numbers

Module M02

Partha Pratim Das

Objectives & Outline

Hello World

Add Two Numbers

Square Root

Standard Library

Header Conventions

Sum of n Numbers

Using bool

Module Summary

C Program	C++ Program

`// Add_Num.c`
`#include <stdio.h>`
`int main() { int a, b; int sum;`
 	
`    printf("Input two numbers:\n");`
`    scanf("%d%d", &a, &b);`
 	
`    sum = a + b;`
 	
`    printf("Sum of %d and %d", a, b);`
`    printf(" is: %d\n", sum);`
`}`
	`// Add_Num_c++.cpp`
	`#include <iostream>`
	`int main() { int a, b;`
	
	`    std::cout << "Input two numbers:\n";`
	`    std::cin >> a >> b;`
	
	`    int sum = a + b; // Declaration of sum`
	
	`    std::cout << "Sum of " << a << " and " << b <<`
	`        " is: " << sum << std::endl;`
	`}`

Input two numbers: 3 4 Sum of 3 and 4 is: 7	Input two numbers: 3 4 Sum of 3 and 4 is: 7
• scanf to *scan* (*read*) from console ✓ • Console is stdin file ✓ • scanf is a variadic function ✓ • Addresses of a and b needed in scanf ✓ • All variables a, b & sum declared first (K&R) ✓ • Formatting (%d) needed for variables	• operator>> to *stream* from console ✓ • Console is std::cin istream (in std namespac • operator>> is a binary operator • a and b can be directly used in ope • sum may be declared when needed. • Formatting is derived from type (

Programming in Modern C++ Partha Pratim Das

So, we want to add two numbers, read them from the input, add two numbers and write their sum. A simple program which you must have written n number of times in terms of C. So, let us see what is, what are the things that are same what are the things that are different. Naturally the input output library is different, we have already observed that. Now, here you print, you declare a, b, here also you declare a, b, these are what you want to read.

So, you put a message to the user asking to input the numbers, here you do the same. But it differs in in scanf as it differs in printf. In scanf you present, it is a variadic function again. So, you present a format and then you put all the different inputs that you want, input variables. And the key thing is while you put this, you do not put it as the name of the variable you put it as the address of the variable, you put an & you know that that is required to be able to read from the input.

Whereas, in C++ you use the input streaming operator which is coming in from the std::cin which is the standard in input console into the variable. And the variable is not written as with the address, it is just written without anything, it is just written as a simple variable. So, syntactically it makes it much easier to deal with it, then you make the sum of them, you do the sum of them here as well.

The only difference I show that the sum variable needed to be declared before and this is what you read in Kernighan and Ritchie's book, that all variables must be declared before the first executable statement of the function. So, it was declared before but here what I have done is I have declared it in the middle, this is what C++ allows from the very beginning. In fact, C also allows it from C89.

So, but I have shown it a comparison. So, this this basically, you can know that this basically is written for K and RC, where it was not allowed to put the variable declaration in the middle of a function. But I just wanted to show you that, now this is also used allowed in C but C++ will often have that. Rest of it is printf and the std::cout which we have already seen in the earlier example.

So, if I go back to the comparison proper, then these are the things that you must note that you are using scanf in C to read from the console, you are using input streaming operator to read from the input stream, the console in the C++. Console is stdin here the console is std::cin which is a istream, I stands for input, it is a the std namespace. scanf is a variadic function like printf whereas operator input streaming is a binary operator.

This comparison is similar to the print. This is very critical, in C you need to take addresses of variables that you want to read. In C++, you do not need to do that, you direct directly use a variable. This reference I have already mentioned, all variables need to be declared first. This is a K and RC. Whereas sum can be declared whenever it was actually needed. And the same thing is allowed in from C89 as well.

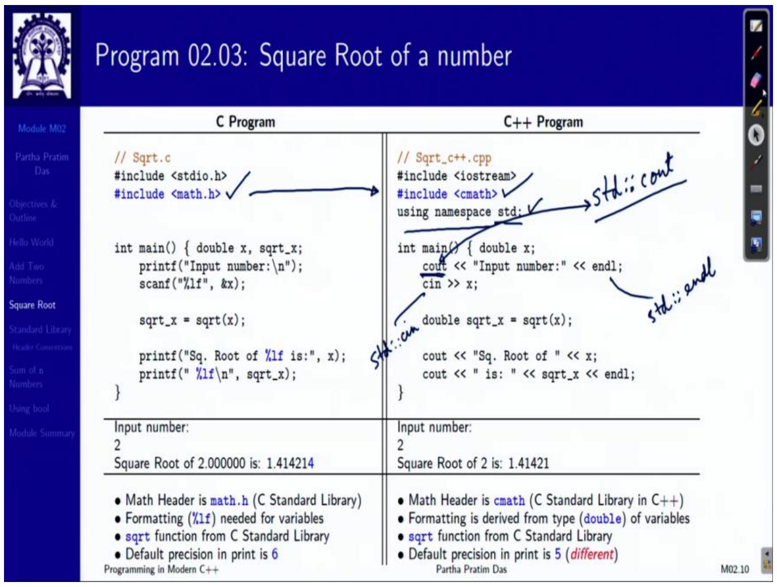
Then you need formatting %d for variables, whereas no specific formatting is required. In terms of, for example, here, you are using formatting of %d to say that you read a you read b or you scan a, you scan b as an integer. Similarly here in printf, you are doing this, you say that you print a as an integer. Here, you do not need to say any of that, just write this. So, that makes life a lot simpler.

So, it is kind of the, kind of way you want the input or the output you just keep on writing that with the repeated streaming operators, and this will serve your purpose. What happens in C++ is since you have already declared a and b as int or sum as int, while you try to stream it to the output or when you are trying to stream it from the input, the compiler knows that it is an integer. So, it knows what is the corresponding formatting that needs to be specified.

This is, this is possible because you have these operators. And it was not possible because these were functions which are variadic, which had no way of knowing what your parameters and what the types of those parameters are, because it is variadic, there is no there is no signature, which says this is their, this is the type of the parameters it is getting. So, it knows that it will get a constant string in the beginning which is a format. After that, it can get you can put as many variables as possible of any type whatsoever.

So, printf, scanf cannot have this smartness of knowing the or deciding the format based on the type of the variable which C++ streaming operators make heavy use of which makes programming far more easier in this language.

(Refer Slide Time: 11:24)



Program 02.03: Square Root of a number

Module M02
Partha Pratim Das
Objectives & Outline
Hello World
Add Two Numbers
Square Root
Standard Library
Header Conventions
Sum of n Numbers
Using bool
Module Summary

C Program

```
// Sqrt.c
#include <stdio.h>
#include <math.h>

int main() { double x, sqrt_x;
    printf("Input number:\n");
    scanf("%lf", &x);

    sqrt_x = sqrt(x);

    printf("Sq. Root of %lf is:", x);
    printf(" %lf\n", sqrt_x);
}
```

```
Input number:
2
Square Root of 2.000000 is: 1.414214
```

- Math Header is math.h (C Standard Library)
- Formatting (%lf) needed for variables
- sqrt function from C Standard Library
- Default precision in print is 6

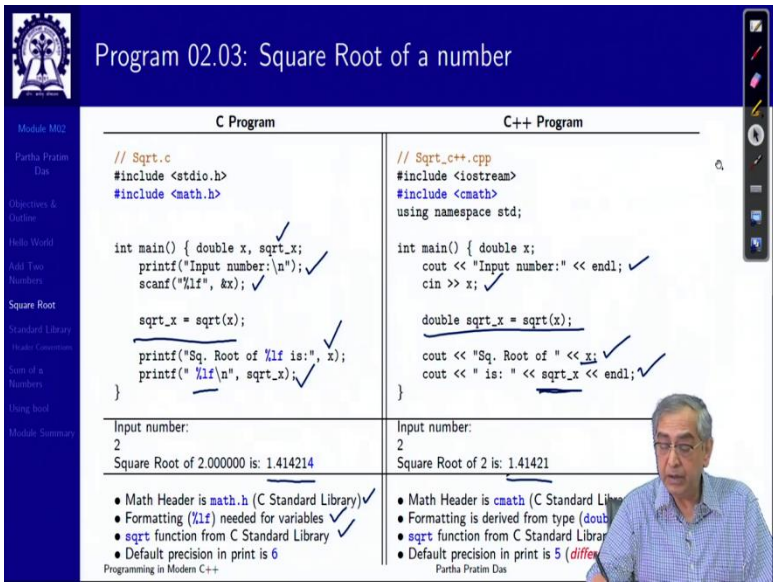
C++ Program

```
// Sqrt_c++.cpp
#include <iostream>
#include <cmath>
using namespace std;

int main() { double x;
    cout << "Input number:" << endl;
    cin >> x;

    double sqrt_x = sqrt(x);

    cout << "Sq. Root of " << x;
    cout << " is: " << sqrt_x << endl;
}
```

```
Input number:
2
Square Root of 2 is: 1.41421
```

- Math Header is cmath (C Standard Library in C++)
- Formatting is derived from type (double) of variables
- sqrt function from C Standard Library
- Default precision in print is 5 (*different*)

Programming in Modern C++ Partha Pratim Das M02.10



Program 02.03: Square Root of a number

C Program

```
// Sqrt.c
#include <stdio.h>
#include <math.h>

int main() { double x, sqrt_x;
    printf("Input number:\n");
    scanf("%lf", &x);

    sqrt_x = sqrt(x);

    printf("Sq. Root of %lf is:", x);
    printf(" %lf\n", sqrt_x);
}
```

```
Input number:
2
Square Root of 2.000000 is: 1.414214
```

- Math Header is math.h (C Standard Library)
- Formatting (%lf) needed for variables
- sqrt function from C Standard Library
- Default precision in print is 6

C++ Program

```
// Sqrt_c++.cpp
#include <iostream>
#include <cmath>
using namespace std;

int main() { double x;
    cout << "Input number:" << endl;
    cin >> x;

    double sqrt_x = sqrt(x);

    cout << "Sq. Root of " << x;
    cout << " is: " << sqrt_x << endl;
}
```

```
Input number:
2
Square Root of 2 is: 1.41421
```

- Math Header is cmath (C Standard Li
- Formatting is derived from type (doub
- sqrt function from C Standard Librar
- Default precision in print is 5 (diffe

Programming in Modern C++ Partha Pratim Das

So, let us move to the next example, which is finding the square root which is again a very simple these are all very simple programs. So, besides the input output, for finding square root, you need to include math.h, mathematical library, in the C standard library. Equivalently in C++, you include cmath.

So, we will discuss this more, but this is just to show you that anything that you have in C standard library, you can use it, use that library function in C++ by including the library as C followed by the library name. C is, C meaning that this is a C library and not having the .h. So, math.h becomes cmath, stdio.h becomes cstdio and so on. Then what you are doing in C++ is you are saying that anything that I do in the library will be in the standard namespace.

We will talk about what namespace is. But just think of now that it is a way to define the symbols with std::. So, earlier we were writing std::cout saying that std, the cout symbol which is the output stream ostream is defined in the std namespace. Now, we have said, we have said that we are using std namespace. So, we are, I am just writing cout.

So, by default the compiler looks in the std namespace and checks is there a cout? It finds one and uses that. So, this actually means this. But it is a lot of typing of the code for example, I am just writing endl but it actually means std::endl. I am just writing cin it actually means std::cin. So, that is the advantage of having the using part of it. So, so, then what we have?

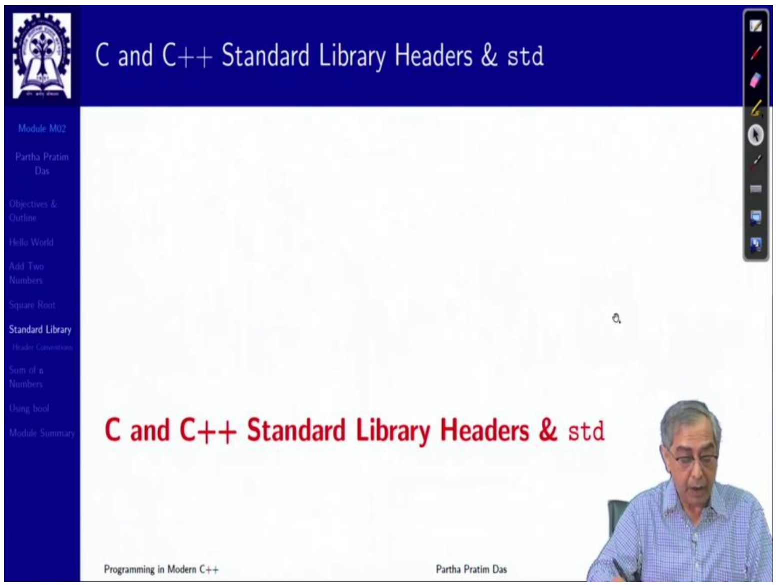
We have the print I am sorry, we have the print message, cout this we have seen, scan to read the input and then you do square root for this. Again, here I have defined it in the K and R style. I can define it, declared sqrt_x right here as well. So, sqrt_x has the value of the square root of x. Then I print x, I print sqrt x similarly I do in. So, it is, it is rest of it is all very similar. And since it is a double I am using a formatting which is for double.

Here again, as I said that, for this or for this I do not need to use any formatting, because it knows, the compiler knows from the type itself. So, here are the main differences. Here math library is math.h, here it is cmath, c standard library in C++ %lf for formatting, here it is not required. Square root is a c standard library function which C++ can you, so, you can whole of the C standard library can be used in this way.

Of course, you see that there is a difference in the precision of printing here you can see that in C you are printing upto 4 up to 6 decimal points that is precision is 6, where here you are printing five, this is for the default setting of the compiler that is doing C as well as that is doing C++. Later on we will show how to control this, you can decide as to, in C you know how to do that the size dot precision kind of qualifier with the format string.

In C++ also we will show how to control the precision as to what you want, but I just wanted to highlight that by default in GCC, C follows %lf follows one precision and cout for a double follow the different precision. So, this is just a, gives you an idea of how to use the C standard library in C++ as well.

(Refer Slide Time: 16:34)



C and C++ Standard Library Headers & std

Module M02

Partha Pratim Das

Objectives & Outline

Hello World

Add Two Numbers

Square Root

Standard Library

Header Conventions

Sum of n Numbers

Using bool

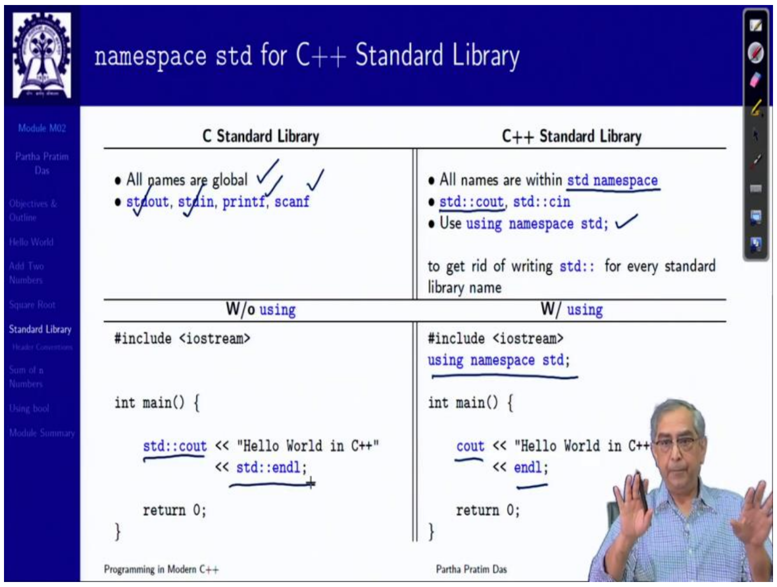
Module Summary

C and C++ Standard Library Headers & std

Programming in Modern C++ Partha Pratim Das

Now having said that, so, let us take a quick look as to how do, how does a C standard library work in C++.

(Refer Slide Time: 16:45)



namespace std for C++ Standard Library

C Standard Library	C++ Standard Library
• All names are global • stdout, stdin, printf, scanf	• All names are within std namespace • std::cout, std::cin • Use using namespace std; to get rid of writing std:: for every standard library name
W/o using	W/ using

```
#include <iostream>


int main() {

    std::cout << "Hello World in C++"
              << std::endl;

    return 0;
}
```

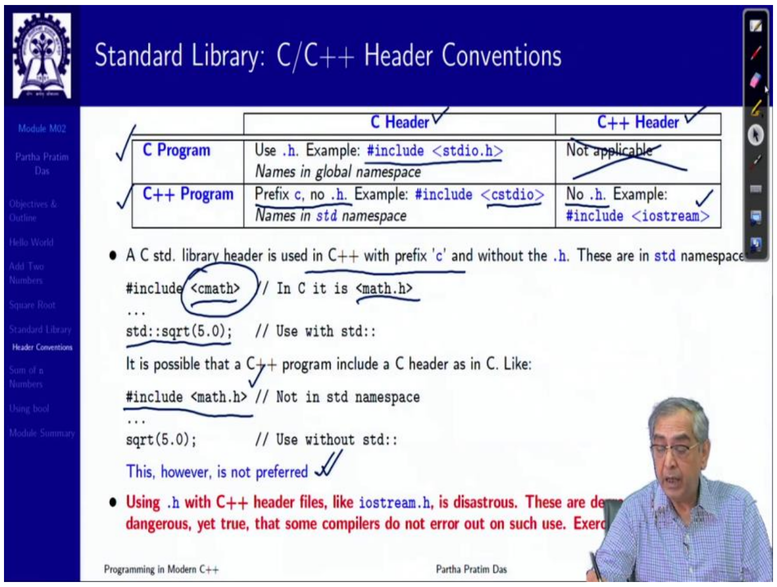
```
#include <iostream>
using namespace std;

int main() {

    cout << "Hello World in C++"
         << endl;

    return 0;
}
```

Programming in Modern C++ Partha Pratim Das

And what is the role that the std namespace play there? So, what I have is C standard library where all names are global as you know, in C everything all functions are global. So, it has global a whole collection of global functions and some global symbols like stdout, stdin these are of course global functions. Now, C stand, C++ standard library all names are with, within std namespace, standard namespace.

So, your cout is std:: we have seen this and the way to do this is doing using, so that you do not have to. So, this is a, this is a comparison of code where you are using, using the namespace std. So, you write less here you are not using, so you write more, it is a syntactic difference nothing else. So, this is the basic of the libraries.

(Refer Slide Time: 17:52)



Standard Library: C/C++ Header Conventions

	C Header	C++ Header
C Program	Use .h. Example: #include <stdio.h> *Names in global namespace*	~~Not applicable~~
C++ Program	Prefix c, no .h. Example: #include <cstdio> *Names in std namespace*	No .h. Example: #include <iostream>

- A C std. library header is used in C++ with prefix 'c' and without the .h. These are in std namespace

```
#include <cmath>  // In C it is <math.h>
...
std::sqrt(5.0);   // Use with std::
```

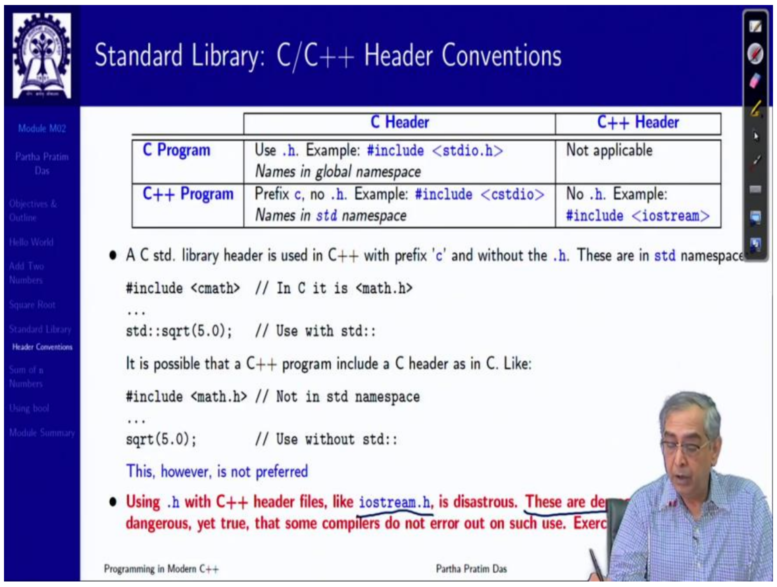
It is possible that a C++ program include a C header as in C. Like:

```
#include <math.h> // Not in std namespace
...
sqrt(5.0);        // Use without std::
```

This, however, is not preferred

- Using .h with C++ header files, like iostream.h, is disastrous. These are de
dangerous, yet true, that some compilers do not error out on such use. Exerc

Programming in Modern C++ Partha Pratim Das



Standard Library: C/C++ Header Conventions

	C Header	C++ Header
C Program	Use .h. Example: #include <stdio.h> *Names in global namespace*	Not applicable
C++ Program	Prefix c, no .h. Example: #include <cstdio> *Names in std namespace*	No .h. Example: #include <iostream>

- A C std. library header is used in C++ with prefix 'c' and without the .h. These are in std namespace

```
#include <cmath>  // In C it is <math.h>
...
std::sqrt(5.0);   // Use with std::
```

It is possible that a C++ program include a C header as in C. Like:

```
#include <math.h> // Not in std namespace
...
sqrt(5.0);        // Use without std::
```

This, however, is not preferred

- Using .h with C++ header files, like iostream.h, is disastrous. These are de
dangerous, yet true, that some compilers do not error out on such use. Exerc

Programming in Modern C++ Partha Pratim Das

Now, how to include the headers? That is the most important thing. So, there are kind of four possible I mean, kind of three possibilities. That you are writing a C program or you are writing a C++ program is the first choice, you want to use a C header, C library standard library header or C++ standard library header. So, how do you write the header in every case? Now, writing a C program, using C standard library header is known to you.

You do an include of the library .h - stdio.h, stdlib.h, math.h and so on so forth, you have been using it. So, you are already familiar. This is not possible that you are you are writing a C program and using a C++ header that is not possible. So, that is ruled out. Now in C++, if you want to use a header, then you use no .h. You will just write iostream, you will just write vector, you will just write stack, you will just write algorithm.

So, these are the different components of the C++ standard library. And you do the same #include but no h, .h. If we are using a C standard library component, then the change that you have to make is put a C before the library name and not use a h again, .h again. And the names will be in std namespace. So, it becomes cstdio. We have seen this. So, when you have C standard library header used in C++, so we wrote cmath.

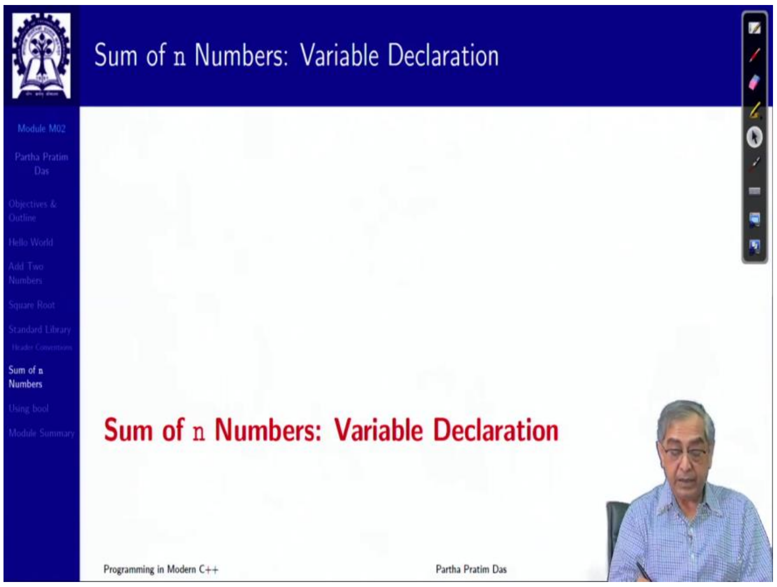
And in C, it was math.h. And it is in C++ it is actually std::sqrt, not just sqrt why? because cmath is a C++ library now. And it is in the namespace, is put in the namespace of std. So, every symbol in math.h which has become now cmath are available not in the global namespace, but in the namespace of std only.

So, you have to use it with that. Now, if you happen to use a C standard library component in C++, not with the C++ style, but just as math.h, you I mean, you are perfectly okay to do that. It is not preferred, it is not preferred, but it is not wrong. So, you can do that, but if you do that, then the name is still, the name from math.h is still in the global namespace. So, you will not use std::sqrt you will just use it as sqrt.

But what I will strongly suggest is do not do this. Finally, what if you include a C header as with a .h, iostream.h. Now, this is, which is something which is disastrous. Because the good part would be that in many of systems, this will give you an error saying that iostream.h does not exist, then you got saved. But in some old systems particularly, a iostream.h file may also exist. And if it does, then it is some old wrong stuff.

This C++ standard library with the .h extension used to be there much earlier, these are now deprecated and completely out of use. So, it may be very dangerous, if you happen to use them and if your system or your compiler is old enough so that it still has some copies of iostream.h or say vector.h and so on. So, always remember that if you are using, when you are using C++ standard library, never use .h.

(Refer Slide Time: 22:57)



Sum of n Numbers: Variable Declaration

Module M02

Partha Pratim Das

Objectives & Outline

Hello World

Add Two Numbers

Square Root

Standard Library

Header Conventions

Sum of n Numbers

Using bool

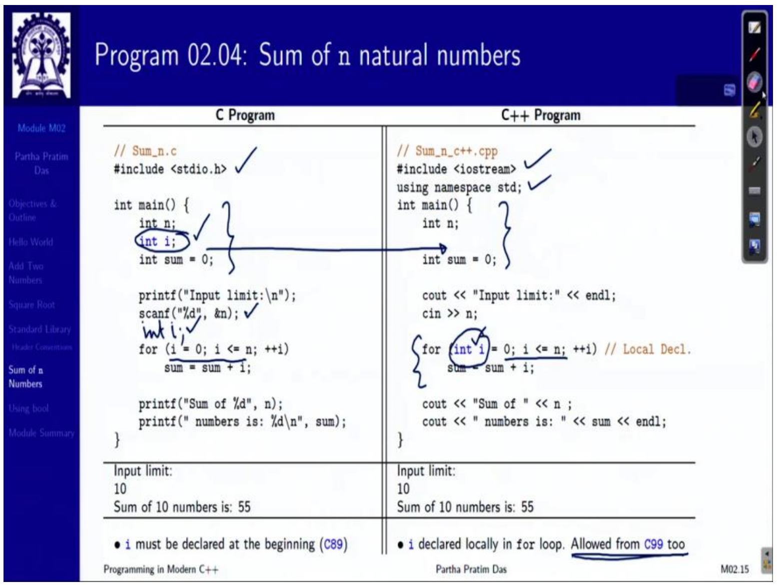
Module Summary

Sum of n Numbers: Variable Declaration

Programming in Modern C++ Partha Pratim Das

So, I move on to the last example, or no sorry, there are one more.

(Refer Slide Time: 13:06)



Program 02.04: Sum of n natural numbers

C Program

```
// Sum_n.c
#include <stdio.h>

int main() {
    int n;
    int i;
    int sum = 0;

    printf("Input limit:\n");
    scanf("%d", &n);
    int i;
    for (i = 0; i <= n; ++i)
        sum = sum + i;

    printf("Sum of %d", n);
    printf(" numbers is: %d\n", sum);
}
```

```
Input limit:
10
Sum of 10 numbers is: 55
```

- i must be declared at the beginning (C89)

C++ Program

```
// Sum_n_c++.cpp
#include <iostream>
using namespace std;
int main() {
    int n;

    int sum = 0;

    cout << "Input limit:" << endl;
    cin >> n;

    for (int i = 0; i <= n; ++i) // Local Decl.
        sum = sum + i;

    cout << "Sum of " << n ;
    cout << " numbers is: " << sum << endl;
}
```

```
Input limit:
10
Sum of 10 numbers is: 55
```

- i declared locally in for loop. Allowed from C99 too

Programming in Modern C++ Partha Pratim Das M02.15

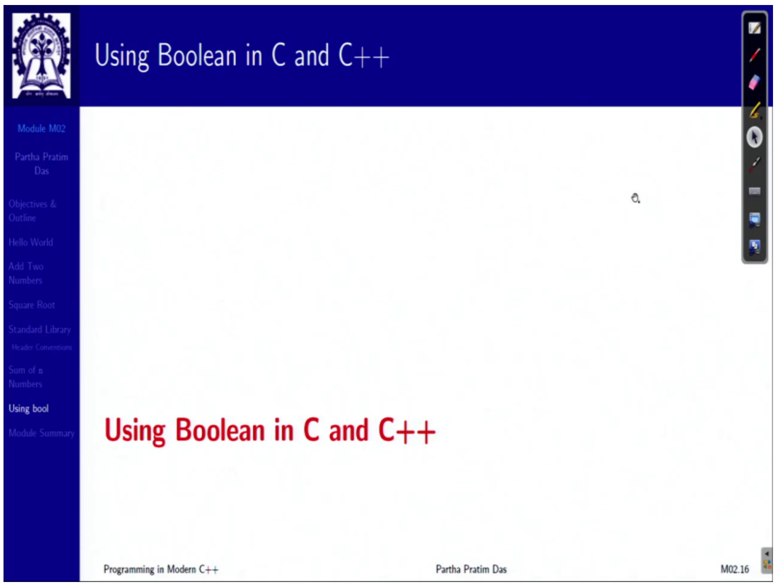
So, the next example which is doing the addition of n numbers, which is something very very common that you have done often I have the input libraries, namespace declarations here. I read the number of numbers. And then, I say that I will basically do an addition of integers from the, from 0 up to that number. So, that is what I am doing here, that is what I am doing here. And then I will print, which is this it is very straightforward simple program.

The feature that I want to highlight is if you are writing it in C, you can write you will have to declare int i here, or if you are doing K and R, or you can declare it somewhere here int i, if you are using C89 or later. But in C++, you have to use, you may use not have to, you can use it as well here. But you may use it right inside the loop. If you use it right inside the loop, it is just a local declaration here, which is often very useful.

Because wherever you need locally, you can just within that control construct, it can declare and use it and outside of that it is not defined. So, it can be reused, that same name can be reused. You may note that from C99 onwards, this has also been included, this feature has also been included in the C language.

So, if you are on K and RC, which is very unlikely you will have to write it here. If you are on C89, not on C99 onwards then you have to use it here. If you are C99 onwards, then you can use it in any one of the places. So, this is some of the basic nuances that you have.

(Refer Slide Time: 25:25)



Using Boolean in C and C++

Module M02

Partha Pratim Das

Objectives & Outline

Hello World

Add Two Numbers

Square Root

Standard Library

Header Conventions

Sum of n Numbers

Using bool

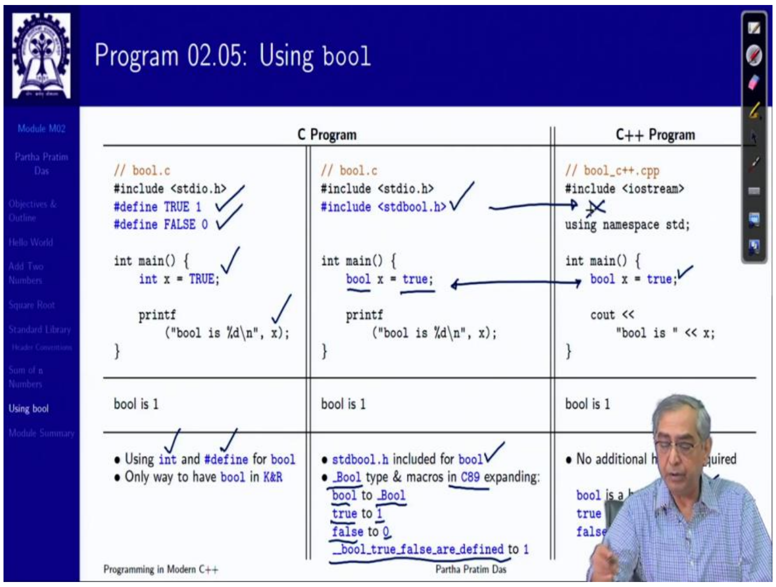
Module Summary

Using Boolean in C and C++

Programming in Modern C++ Partha Pratim Das M02.16

Now, let us, let me move to the last example, which is regarding Boolean.

(Refer Slide Time: 25:31)



Program 02.05: Using bool

Module M02

Partha Pratim Das

Objectives & Outline

Hello World

Add Two Numbers

Square Root

Standard Library

Header Conventions

Sum of n Numbers

Using bool

Module Summary

C Program

```
// bool.c
#include <stdio.h>
#define TRUE 1
#define FALSE 0

int main() {
    int x = TRUE;

    printf
        ("bool is %d\n", x);
}
```

```
bool is 1
```

- Using int and #define for bool
- Only way to have bool in K&R

```
// bool.c
#include <stdio.h>
#include <stdbool.h>

int main() {
    bool x = true;

    printf
        ("bool is %d\n", x);
}
```

```
bool is 1
```

- stdbool.h included for bool
- _Bool type & macros in C89 expanding:
  bool to _Bool
  true to 1
  false to 0
  __bool_true_false_are_defined to 1

C++ Program

```
// bool_c++.cpp
#include <iostream>
using namespace std;

int main() {
    bool x = true;

    cout <<
        "bool is " << x;
}
```

```
bool is 1
```

- No additional h quired

  bool is a
  true
  false

Programming in Modern C++

Partha Pratim Das

Now, Boolean as we all know is an important feature of the language. We always need to make choices true or false and we need a boolean value. So, in, in traditional C say K and RC, what will be Boolean. So, typically you will, you can have Boolean as a manifest constant true define as one and in other false defined as 0, you can put say x is assigned true. And then if you print, it will print it as 1.

So, using int and #define, you can create a bool kind of variable in K and RC which is the only possible way. So, otherwise, you are, I mean basically you are maintaining it as an integer only, but you are just, interpreting it as a boolean. If it is 0, you say it is false if it is not zero, you say it is true. Now, when you move to C89 things improved. So, C89 provided a new header known as stdbool.h.

This header is provided specifically to support boolean value as bool, you want to call it bool as if it is a different thing. So, it includes bool and it actually is, the actually the defined type is _bool, but there is a macro which defines bool to _bool. There are macros which define true as 1 false as 0, mind you these are predefined in stdbool.h and these are all in lowercase.

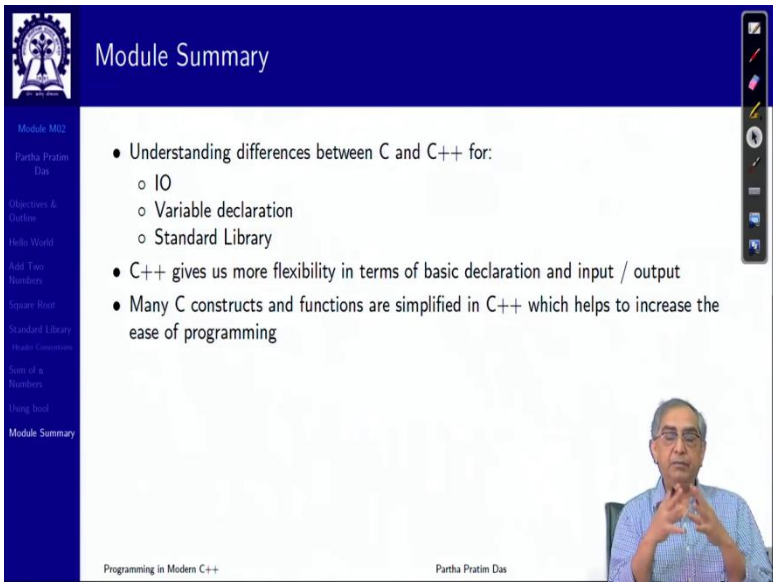
So, with that you can write this, mind you this is defined in that stdbool.h header. So, if it is not included, this code may give an error, but some other C standard library components also in turn include stdbool.h. So, it is possible that if you are including stdlib.h you may not need to include this, but I have specifically shown it here, so that you can have these macros available. And this, this there is also a macro to check that whether you have done it, which gives you a sense of a true boolean type.

In C++, it goes further, it is not macros which are defining it. It is been supported as a built in type, like you have integer, like you have double, you have a bool type. And like 1, 0, 2, 3, 4, 17 all of these are literals of int type. Bool type has two literals, true and false. So, they are not mappings to 1 or 0 they are just literals by themselves and C++. So, code wise it looks same.

And certainly you do not need to include anything special, because it is no more a feature of the library, it is a feature of the language. So, in the leftmost in K and RC, there was no support in the language nor in the library. So, you are creating and interpreting your bool in your one way. In the second column, you are in C99 onwards, C89 onwards, where you have a library support to give you a feeling of a Boolean, bool.

And rightmost column in C++, you now have bool as a built in type with true and false mean actual literal. So, it is now a language support, no more a library support. So that is, that is kind of the development that has happened. And that gives us a very easy way of using bool in C++.

(Refer Slide Time: 29:58)



Module Summary

Module M02

Partha Pratim Das

Objectives & Outline

Hello World

Add Two Numbers

Square Root

Standard Library

Header Conventions

Sum of n Numbers

Using bool

Module Summary

- Understanding differences between C and C++ for:
  - IO
  - Variable declaration
  - Standard Library
- C++ gives us more flexibility in terms of basic declaration and input / output
- Many C constructs and functions are simplified in C++ which helps to increase the ease of programming

Programming in Modern C++ Partha Pratim Das

So, to summarize, we have, I have tried to highlight a few differences between C and C++ which are fundamental nature particularly in terms of IO, input and output, variable declaration and the use of standard library, which are the commonest things you start doing in starting to program.

And we have seen that C++ gives us better flexibility in terms of declarations and input, output and makes it simpler to deal with. And many C constructs and functions get simplified in C++, which helped to increase the readability as well as the ease of programming.

So, with these words, I will conclude on this module 2. And I hope these examples and variants of them, you will not only study in the module, but you will start practicing them on the system, that you have installed the GCC both in C as well as in C++ versions and actually check for yourself what you are getting and then you make modifications to understand that how elementary C++ programs can be created corresponding to the C versions. Thank you all very much. See you again in next module.