FUNDAMENTALS OF OBJECT ORIENTED PROGRAMMING

Lecture 26: Exception Handling in C++

Thank you. Welcome to lecture 26, the new chapter, Exception Handling. So the agenda of this chapter is we are going to see types of exceptions. Then we will talk about the try-catch blocks. And then I will talk about the custom exceptions.

So exception handling in C++ or Java. Suppose when you are writing a program and then you are really worrying about the flow of the program. So, then what happen suddenly right for example, you are doing a divided by 0 right a by 0 suppose you are solving some numerical problems the denominator you have to always worry about the denominator. So, the denominator may tend to 0. So, in that case if you are not able to right notice this or if you are not able to locate this then there will be a runtime problem right.

So, whereas this problem can be solved with the help of exception handling, whether it is in C++ or Java, this is considered as one of the most powerful mechanism, all right, to handle the runtime error so that you will not worry about the flow of the application or a program. And in fact, the flow of the application will be maintained. So, in C++ or Java, exception, which is nothing but an even that disturbs the normal flow of the program. So, it is nothing but an object which will be thrown during the runtime. So, what are all the different kinds of errors?

So, when you are writing code, what are all the errors you can find? First, the syntax error. If you miss a semicolon or colon, or if you write your assignment operation incorrectly, then you will get a compilation error. So, this is a syntax error. Next, semantic or logical errors.

The program may run, but you may get unexpected output. Then, runtime errors, right? During compilation time, for example, if you write division by 0, right? A divided by 0, or A divided by B—division of two numbers. Assume you are writing a program for division of two numbers. You write: int a, b, c equals a divided by b. The program will run during compilation time, right?

But when it expects output—when the compiler expects output—and you provide, say, A and B, assume B is an integer and B is 0, which is also valid, right? So when you compute C equals A divided by B, and A and B are integers, there will be a problem, right? When B equals 0, there will be a problem. So, what will happen? There will be an abnormal termination of the program, right?

So, even this causes the system to crash, right? So, these kinds of errors, right? These kinds of runtime errors. So, even this causes the system to crash. So, how do you take care?

How do you take care of such a situation? So, there are several exceptions we are going to see. So, how do you handle such exceptions that we are going to study in C++ and Java. So, as I said, when there is an unexpected circumstance. So, one example I had given was a divided by b, where b is equal to 0 or b tends to 0.

So, that is what, right? So, dividing by 0. So, in that case, I want to handle such cases, right? So, what are all the keywords I can use in the exception? So, you have try, throw, catch.

So, the try-throw-catch mechanism. So, you are trying certain block. So, you are trying, let us say, I am going to write C is equal to A by B. So, if B is equal to 0, how I am going to handle? So, a try block may trigger, so contains a code, may trigger exceptions. So, C is equal to A by B when I write.

So, where if suppose b is equal to 0, so that may trigger exceptions, right. So, what you have to do? So, you should have, right. So, you should throw the exception and when you are throwing the exception, so it will be caught by the catch function, all right. So, this is what exactly we are going to see an example, all right.

So, in the try block, you will have throw mechanism and the throw mechanism will throw. For example, I am planning to create the nodes in data setters. right trying to create the nodes in data structures at one point of time I may not find the memory right so the exception will be thrown and the exception will be thrown and it will be caught by the catch function right so here you have so here the exception I do not have a memory so when there is no memory exception will be

thrown that is one case or when you are dividing by let us say 0 Exception will be thrown, right? So, throw.

So, an exception is thrown when a problem occurs. We will use the keyword called throw, right? So, throw divided by 0 occurs, right? It is like a print statement that you are writing. Throw with a double quote divided by 0 is occurring. If b is equal to 0, throw, right?

And this will be caught by This exception will be caught by the catch function or the catch keyword. So B, which is being thrown, and in the catch you are writing. So catch, you are writing. So, how am I handling this exception?

So, divided by 0 is not possible. So, you are writing. So, you can also directly handle if B is equal to 0: division is not possible. Else, A by B. That is the usual way when we are not talking about the exception. But whereas there may be several exceptions, like I talked about the no memory.

The exception will be thrown and it will be caught by the keyword called catch. And this will be handled at a specific point in the program with an exception handler. So this catch keyword. So that will handle at a specific point inside the program with an exception handler. So we use catch keyword.

right. So, which will be used to define these kind of handler. So, try throw catch in C++. So, which is playing a vital role to handle the exception. So, what are all the common runtime errors we get?

So, one I discuss number divided by 0. One example, No memory we had seen. So, one example, first example is dividing by 0, right. So, this is whether in C++ or Java, divide by 0 or any programming language, divide by 0, right or even in mathematics.

So, you have, this is the reason you have limit continuity, right, etc. Whether the limit exists or not or tending to infinity, right. So, you might have studied in your basic mathematics, right. And then, if there is an out of bound of an array, Assume that you have dynamically allocated 100 elements in array, particularly in Java, right?

So, there is out of bound error which is common. So, this is during runtime, right? And trying to store some incompatible data elements, right? So, when I am trying

to store some incompatible data elements, the runtime error occur. And in the array, if I try to use a negative index, right?

So, a of minus 1. Right? Runtime error. In the case of Java. Right?

In the case of Java, if I use a of minus 1, I will get the runtime error. Or when you are converting, let us say, string which contains x, y, z or a, b, c to integer values. Right? The string data into certain specific data value. Runtime occurs.

Right? So these are all the common runtime errors. Similarly, you may have when you are handling the file. So you may have this problem when you are handling the file. Let us say opening a file in read mode.

That does not exist or no read permission. The file does not exist. But whereas you are trying to read the file, runtime error will occur. Or the user or administrator did not give the permission to read the file. Anyway, the file handling we are going to see soon.

So before that, these are all the Common runtime errors, and similarly, opening a file in write or update mode, right? So, which has only the read-only permission, right? Suppose a file has been given only read-only permission, and you are trying to update that particular file or write something on it. Correct.

So these are all the cases where the runtime error occurs. Right. So these are all the different cases where the runtime error occurs. So now we will see in C++. So we will see one program with the help of try, throw, and catch.

Right. So this is the mechanism we had talked about. So how am I going to handle the divide-by-zero error? So, let us consider a function named division, whose return type is double, and it has two integer arguments. If b is equal to 0, if you look carefully at the syntax.

I am just writing through. It is like a cout, right? It is like cout statement divided by 0, division by 0 condition, right? So, then you have return a by b, all right? So, return a by b. So, suppose b is not 0, it will come to this particular part and it will compute a divided by b, correct?

So, now go to the main program. In a main program, I have Int L is equal to the number 1729, right? And int M is equal to 0. So, let us have double N is equal to 0 because I am going to do the division, right?

And the return type is also double, correct? In the division function, the return type is double. So, in this case, right? Assume that I am calling the function by passing L and M. L is 1729 and M is 0 I am passing, right? So, this function will be called.

So, which function will be called? I am trying this block. Line numbers 16 to 18 are a try block. So, I am trying. So, let us see what is happening, right? So, try block. So, division is calling the division function with passing L and M. So, this is nothing but L, and this is nothing but M. So, you are passing the value.

So, A will take 1729, and B will take 0, right? So, if B is equal to 0, it is throwing the exception, right? It is throwing the exception, division by 0 condition, throwing this, B is 0, correct? So, B is taking the value 0, throw the exception. So, that will be caught over here, catch, all right?

So, catch, so it has a character array which has a message, right? The character array, which is a dynamic array, which is a message, which has a message. So, C error, like C out, this is called C error, right? So, like C out, so this is called C error. So, C error, you are printing the message.

So what message you have thrown? I have thrown the message division by zero condition. So this message I have thrown. right, the compiler is throwing, right. So, if b is equal to 0, so this will be thrown and that will be caught over here and you are trying to print out the message that has been thrown, right.

So, then you have carefully handled this case. So, that means your compiler will not go to this line at all, right. So, this is thrown. So, you are coming out, that means you are avoiding line number 18. And it was coming to return 0, program is over.

So, that means you have line number 20. So, it will throw that C error, right. So, division by 0 condition will be thrown. So, when I run the code, so your input is empty, output is also empty. So, there is no output.

I am not getting any n value. So, here you can see the error, standard error because you are seeing an error. You are trying to print the message 'division by 0 condition'. So, this message is being printed, right? So, if you look at line number 16, which is a try block. The try block is like a usual code, right?

So, try. It will work like a function, right? So, without any argument notation, right? So, it works like a function, or you can say it is a keyword, right? Try and that is going into a block.

Inside the block, you can see the usual C++ code, and on line number 17, you have n equal to the division of l and m. So, that means you are passing 1729 and 0. So, a will take the value 1729, and b will take the value 0. So, if b is equal to 0, it is throwing the exception. So, what exception is it throwing? The 'division by 0' condition is being thrown, and then it will be caught by the function on line number 19, catch, right?

So, that is the message you are passing, right? It is being passed, and that message you are printing over here, right? So, then it is coming out of this catch block. Coming to line number 23, return 0. So, if you run the code, the standard error, I am getting division by 0 condition, right.

So, whatever I have thrown, this is happening. So, in the same code, suppose m is not 0, right. So, let us say m is some 10, right, which is not a problem, division l comma m, right. This function will be called. And if you look, M is not 0, return A by B. What is A by B? 1729 by 10, right.

So, 172.9 will be returned. It is a double, right. And then it will be printed. It will not go to the catch block at all, right. See, out N it will be printed. It will avoid the catch block. It is not going at all because no exception is being thrown, and then return 0. So that means here it will be N will be double, right. So, the double, whatever 1729 by 10 will be returned, and c out n that will be printed.

It will not go to the catch block at all if m is not equal to 0. I hope it is very clear how this try-throw-catch is working. So, you may think of several problems. So, array out of bounds, right? So, several runtime errors.

So, whenever you feel there are runtime errors, you have to think about this. You are converting, let us say, a string to double or a string to integer, right? So, it is always better to use these try-throw-catch statements, right, or the keywords. The reason is your program's flow will be maintained. That is the main idea.

So, now, what are all the C++ standard exceptions, right? So, in detail, we will also see. So, now I will talk about the overview. Right. So here, std exception is the superclass.

Right. std::exception. Right. So, which is a superclass. And then you can have several exceptions.

Right. So, under this, you have bad_alloc. Right. Bad_cast. Bad_typeid.

Right. Bad_typeid. Bad_exception. Right. Logic_error.

And then you have a runtime error. So, under logic error, you have domain error. Invalid argument. Length error. Alright.

So, like array length I talked about. Alright. Out of range. Alright. So, these are all under your logic error.

Alright. So, under your runtime error, the overflow error, the range error, and then there is an underflow error. So, these are all the different types of exceptions. So, you might have come across at least one, right? We will talk about, maybe we will take some examples and see what all the different kinds of these errors are, right?

So, we will see in details. So, here you go the C++ standard exceptions right. So, as I already said standard double colon exception. So, which is nothing but C++. The exception, this particular exception is a parent class of all the standard C++ exceptions.

All right. So this particular exception is a parent class of all the standard C++ exceptions. So under this, so you may have the bad underscore allocation. All right. So in the new operator, assume that you are trying to allocate a memory.

All right. So suppose you are not able to do this, that exception is called bad underscore allocation. Similarly, you are trying to have a dynamic casting, right? So, try to have a dynamic casting. So, for example, I am converting a string into, let us say, integer, right?

So, string to integer. So, I will have the exception. The exception is called bad_underscore_cast, right? And bad_underscore_exception. So, an unexpected exception is happening, right?

So, an unexpected exception is happening. So, in that case, the exception is called bad_underscore_exception and bad_underscore_type_id. So, this can be

thrown by the type_id and then bad_underscore_logic, right? So, sorry, standard double colon logic_underscore_error. So, theoretically, you can find it.

So, suppose you are reading the code. Theoretically, you are finding out, right? So, that error is called logic_underscore. That exception is called logic_underscore_error. And domain_underscore_error, so this is an exception thrown when a mathematically invalid domain is used.

Invalid arguments. So, when you have the invalid arguments, alright, so invalid underscore argument due to invalid argument. Instead of int, when you are passing the function, instead of int, you have a character, alright. So, that is one case. So, the exception, name of the exception is invalid underscore argument.

So, length error. right. So, suppose you have created a too big string, right, too big standard double colon string. So, when you are creating this, so there will be a problem. So, that exception is called length underscore error and out of range, all right.

So, out of range, So, when you are dealing vector or bit set or operator, right? So, you may have the problem. Suppose you are using the fixed set of vector, size of the vector and if it is going beyond that, right? So, you call it as out of range.

Runtime underscore error. So, theoretically, you cannot find it, right? So, even that is what I said when you are writing the code A by B, right? So, theoretically, there is no problem. C is equal to A by B when I write.

So, theoretically, you do not find any problem, but the moment you give B equal to 0. So, you will get a runtime error. So, std double colon runtime underscore error, this exception, all right. So, theoretically, I cannot detect anything, but the exception may occur when you are giving the input—that is one case, right. If the mathematical overflow occurs, right.

So, that is called overflow underscore error exception range error. So, when you try to store the, let us say, integer, right? There is a range, and you are trying to store more. So, range underscore error, right? And similarly, the underflow error. So, these are all the different kinds of exceptions.

So, now can we have the user-defined exceptions, all right? So, can we have these? These are all the system exceptions, standard exceptions—C++ standard exceptions—that we have seen, all right? So, can we have the user-defined exceptions? All right. So, assume that I have a class, myException.

All right. Which is publicly inheriting from the exception class available in the system. All right. I am inheriting the properties. All right.

So, this is the name of the class. Let us say, myException. So, we are including one more header file called exception. #include . So, now here you have the pointer function or function pointer, what?

All right. So, it is a built-in. All right. Which is a character function. Right return type, right? The function is what, right? And what it is doing is returning what constant, and you have throw, right.

So, it is returning 'attempted to divide by 0,' right? It is returning 'attempt to divide by 0' const, we know, and there is a throw, try, throw, catch, throw, throw function here. So, return 'attempt to divide by 0.' So, this is in the class myException under the public function, what? It is a built-in one under exception. So, in the exception, you can see, and I am just modifying over here, and it will be overridden.

It will be overridden. Now, in the main, I have a try block. In the main, I have a try block: int l, m. So, we use int l, m. So, you have, you are entering two numbers: int l, comma, m, and then you are entering two numbers. So, taking l and m, correct? If m is equal to 0, you are trying to create an object n, right? You are trying to create an object n. So, I do not have any right here, my exception. I do not have any constructor here.

So, I am trying to create an object n, right. And then, right, if it is 0, if m is equal to 0, throw n, right. So, throw n, correct. So, throw n in the sense, suppose n is, m is 0, throw n, right.

So, you have what function over here, right. So, it will be caught. If m is equal to 0, you are creating an object n under the class myException. And this object will be thrown if M is equal to 0, right? This object will be thrown.

This is what we studied. It is nothing but, I mean, when the exception occurs, it will throw the object, right? So, this object will be caught over here, right? You are throwing this and this object will be caught over here and you have a statement, right? This is under exception.

So, this is a base class, right? The reference you are passing, and E is invoking what? Right? E is invoking what? When E is invoking what?

Here you go, right? So, this particular function will be called, right? And this will return 'attempted to divide by 0', right? So, this will return 'attempted to divide by 0'. So, this will be returned, okay?

So, maybe before going this, We will test the case. Suppose I am giving the input 25 and 0, right. So, that means L is equal to 25 and M is equal to 0, right. So, this is what the CN, L, and M I am doing.

So, M is 0, right? So, we will create an object N. And this will be thrown, all right? So, throw N. Throw N will be caught over here by the catch function, right? And then you have an object E, the reference object E under exception. And here you go, the object E which is invoking what, right? Object E which is invoking what? So, when it is invoking what, right? So, here you go. So, this will be attempted to divide by 0. So, this will be returned. So, when I run the code, this is your input, and I will get the output: attempted to divide by 0.

So, this is the way you can handle it. So, I have used try, throw, and catch. So, these keywords I have used—we have carefully used them, and then we could handle it. Another case: suppose it is not 0. So, assume that my another input is 25 and 5.

So, 25 and 5 when I pass L and M. So, 25 and 5 when I pass, M is not 0. It will come to the else part. So, L by M. What is L by M? 25 by 5. I am expecting 5.

Right. So, it will print, 'You have done 50 percent of OOP,' and then you are also printing, 'All the best.' Alright. So, when I run the code, I will get the output. Alright. So, like this, L by M is 25 by 5, which is 5. 'You have done 50 percent of OOP now. All the best.' Alright. So, this is the output you will get.

So, in this lecture, we talked about exception handling. Particularly, when we talk about the try, catch, and throw blocks in C++. Alright. So, the try block is written

as if you are writing the usual program. And then, if there is any exception thrown, one of the cases we have seen is division by zero.

You can also try, alright. For example, giving an array out of bounds. Alright. When dynamically allocating memory, assume it is going out of bounds. Right.

Or assume that you have created dynamically an array of 4 elements. Assume that in the input you are giving 5 elements. Right. So the array goes out of bounds. Runtime error.

Compile time there won't be any error because the compiler is expecting 4 elements. Right. Or you are converting a string to an integer. Right. So these are all some of the cases I talked about.

What are all the cases? The runtime error occurs. So, try one of the runtime errors we have seen: divided by 0, and you have seen with the help of try-throw-catch how we can handle the situation. So, we have seen several exceptions, right? Standard C++ exceptions, and we had seen in the last program the user-defined exceptions as well, all right. So, with this, I am concluding this lecture.

Thank you.