In C++, a try-catch block is used to handle exceptions. It helps you manage errors that may occur within a specific section of code.

Syntax

C++

```cpp
try {
    // Code that might throw an exception
}
catch (ExceptionType1& ex1) {
    // Handle ExceptionType1
}
catch (ExceptionType2& ex2) {
    // Handle ExceptionType2
}
// ... More catch blocks for other exception types
catch (...) {
    // Handle any other exceptions not caught above
}
```

Explanation of the above code:

1. The code inside the try block is executed.
2. If an exception is thrown during the execution of the try block, the program immediately jumps to the corresponding catch block that matches the type of the thrown exception.
3. The first matching catch block is executed, and any remaining catch blocks are skipped.
4. If none of the catch blocks match the thrown exception, the program jumps to the

catch (...) block, which is used for catching any unhandled exceptions.

Example

C++

```
#include <iostream>
#include <stdexcept>

using namespace std;

int main() {
    try {
        int numerator = 10;
        int denominator = 0;
        if (denominator == 0) {
            throw runtime_error("Division by zero");
        }
        int result = numerator / denominator;
        cout << "Result: " << result << endl;
    }
    catch (const runtime_error& e) {
        cerr << "Runtime error caught: " << e.what() << endl;
    }
    catch (const exception& e) {
        cerr << "An exception occurred: " << e.what() << endl;
    }

    return 0;
}
```

Explanation of the above code:

1. The program attempts to perform a division operation with the values 10 (numerator) and 0 (denominator).
2. It employs a try block to enclose the code that might generate exceptions.
3. Inside the try block:
    - It checks if the denominator is zero using if (denominator == 0).
    - If the denominator is indeed zero, it throws a runtime_error exception with the message “Division by zero”.
4. If the denominator is not zero, it calculates the result of the division (numerator / denominator) and displays it using cout.
5. In case a runtime error occurs (division by zero), the program catches the runtime_error exception using the first catch block.
    - It uses cerr to display the error message “Runtime error caught: ” followed by the exception’s error message retrieved with e.what().
6. If any other type of exception occurs, the second catch block catches and handles it.
    - It uses cerr to display a more general error message: “An exception occurred: ” followed by the exception’s error message.
7. Finally, the program returns 0 to indicate successful execution.

Points to remember

- Catch exceptions by reference.
- Prioritize more specific exception types.
- Avoid catching exceptions too broadly.
- Nest try-catch blocks for different levels of handling.

Related posts:

1. Sequence Control & Expression | PPL
2. PPL:Named Constants
3. Parse Tree | PPL | Prof. Jayesh Umre
4. Basic elements of Prolog
5. Loops | PPL | Prof. Jayesh Umre
6. Subprograms Parameter passing methods | PPL | Prof. Jayesh Umre
7. Programming Paradigms | PPL | Prof. Jayesh Umre
8. Subprograms Introduction | PPL | Prof. Jayesh Umre
9. Phases of Compiler | PPL | Prof. Jayesh Umre
10. Parse Tree | PPL
11. Influences on Language design | PPL | Prof. Jayesh Umre
12. Fundamentals of Subprograms | PPL | Prof. Jayesh Umre
13. Programming Paradigm
14. Influences on Language Design
15. Language Evaluation Criteria
16. OOP in C++ | PPL
17. OOP in C# | PPL
18. OOP in Java | PPL
19. PPL: Abstraction & Encapsulation
20. PPL: Semaphores
21. PPL: Introduction to 4GL
22. PPL: Variable Initialization
23. PPL: Conditional Statements
24. PPL: Array
25. PPL: Strong Typing

26. PPL: Coroutines
27. PPL: Exception Handler in C++
28. PPL: OOP in PHP
29. PPL: Character Data Type
30. PPL: Exceptions
31. PPL: Heap based storage management
32. PPL: Primitive Data Type
33. PPL: Data types
34. Programming Environments | PPL
35. Virtual Machine | PPL
36. PPL: Local referencing environments
37. Generic Subprograms
38. Local referencing environments | PPL | Prof. Jayesh Umre
39. Generic Subprograms | PPL | Prof. Jayesh Umre
40. PPL: Java Threads
41. PPL: Loops
42. PPL: Exception Handling
43. PPL: C# Threads
44. Pointer & Reference Type | PPL
45. Scope and lifetime of variable
46. Design issues for functions
47. Parameter passing methods
48. Fundamentals of sub-programs
49. Subprograms
50. Design issues of subprogram
51. Garbage Collection
52. Issues in Language Translation

53. PPL Previous years solved papers
54. Type Checking | PPL | Prof. Jayesh Umre
55. PPL RGPV May 2018 solved paper discussion| Prof. Jayesh Umre
56. PPL Viva Voce
57. PPL RGPV June 2017 Solved paper | Prof. Jayesh Umre
58. Concurrency
59. Basic elements of Prolog
60. Introduction and overview of Logic programming
61. Application of Logic programming
62. PPL: Influences on Language Design
63. Language Evaluation Criteria PPL
64. PPL: Sequence Control & Expression
65. PPL: Programming Environments
66. PPL: Virtual Machine
67. PPL: Programming Paradigm
68. PPL: Pointer & Reference Type
69. Programming examples
70. Program to prints even numbers from 1 to 20
71. Program to calculate sum of all numbers from 1 to 100.
72. Program to get factorial of a number
73. Program to get Fibonacci sequence
74. Program to checks if number is prime
75. Program to get multiplication table
76. Program to find largest element in an array
77. Program to prints reverse of a string
78. Program to calculates sum of all elements in a list
79. Program determines integer is positive, negative, or zero

80. Program to find largest among three numbers using conditional statements.
81. Program determines it is a leap year or not
82. Program to determines even or odd
83. Program to calculate student exam grade
84. Program determines character is a vowel or consonant
85. Program to determines product is positive or negative
86. Program to determine divisible by both 5 and 7
87. Program to determines equilateral, isosceles, or scalene triangle
88. Programme to check if number is inside range
89. Function to calculate the factorial
90. Write a function to detect palindromes in strings
91. Write a function to find the greatest common divisor of two numbers
92. Program to calculate the area of different geometric shapes