Purpose of Exception Handling:

- Exception handling in C++ is used to manage and recover from unexpected errors or exceptional situations that might occur during program execution.

Try-Catch Blocks:

- The try block is used to enclose the code that might cause an exception.
- If an exception occurs within the try block, the program jumps to the nearest matching catch block.

Throwing Exceptions:

- To indicate an error, you can “throw” an exception using the throw keyword.
- Exceptions can be of any type, including standard library exceptions or user-defined types derived from std::exception.

Catching Exceptions:

- The catch block is used to handle exceptions thrown within the corresponding try block.
- You can catch exceptions by their types, including base classes, to handle related exceptions in a single block.
- Usually, a reference (often const) to the exception type refers to the caught exception.

Standard Exceptions:

- C++ provides a set of standard exception classes, such as std::runtime_error,

std::logic_error, and std::invalid_argument, which are derived from std::exception.

Custom Exception Classes:

- You can create your own exception classes by deriving them from std::exception or other existing exception classes.
- Custom exception classes should typically provide a custom error message using the what() function.

Multiple Catch Blocks:

- You can have multiple catch blocks to handle different types of exceptions.
- Catch blocks are evaluated sequentially, and the first matching block is executed.

Order of Catch Blocks:

- Place more specific catch blocks before more general ones. Specific exceptions should be caught before their base classes.

Unhandled Exceptions:

- The program will terminate and display an error message if no catch block within the current scope is able to catch an exception.

Rethrowing Exceptions:

- You can use the throw statement inside a catch block to rethrow the caught exception.
- This allows an exception to be caught at one level of the call stack and then handled or rethrown at a higher level.

Resource Management:

- Even during exceptions, use exception handling to release memory or close files.

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

50. Garbage Collection
51. Issues in Language Translation
52. PPL Previous years solved papers
53. Type Checking | PPL | Prof. Jayesh Umre
54. PPL RGPV May 2018 solved paper discussion| Prof. Jayesh Umre
55. PPL Viva Voce
56. PPL RGPV June 2017 Solved paper | Prof. Jayesh Umre
57. Concurrency
58. Basic elements of Prolog
59. Introduction and overview of Logic programming
60. Application of Logic programming
61. PPL: Influences on Language Design
62. Language Evaluation Criteria PPL
63. PPL: Sequence Control & Expression
64. PPL: Programming Environments
65. PPL: Virtual Machine
66. PPL: Programming Paradigm
67. PPL: Pointer & Reference Type
68. try-catch block in C++