NAMED CONSTANTS

A named constant is a variable that is bound to a value only once.

Viva Voce on Named Constants

Q1. What in const?

Ans. Constant is something that doesn't change. In C and C++ we use the keyword const to make program elements constant. const keyword can be used with:

1. Variables
2. Pointers
3. Function arguments and return types
4. Class Data members
5. Class Member functions
6. Objects

Q2. Can we use pointer with const?

Ans. Yes using const keyword.

Q3. If a function has a non-const parameter, can be passed a const argument while making a call ?

Ans. No. But, a function which has a const type parameter, can be passed a const type argument as well as a non-const argument.

Q4. How to define class object const.

Ans. When an object is declared or created using the const keyword, its data members can never be changed, during the object's lifetime.

```
const class_name object;
```

Q5. What happen if we make function with const?

Ans. A const member function never modifies data members in an object.

Q6. What is mutable keyword?

Ans. Mutable keyword is used to create Mutable data members, and a Mutable data members of a const objects can be modified.

Q7. Can constructor used with const ?

Ans. Yes

Q8. What is the use of const keyword after method declaration as in void fun() const ?

Ans. The const keyword after the method declaration essentially makes the this pointer a constant so you can not change the member data (if they aren't mutable) .

Q9. What is int const*?

Ans. int const* is pointer to constant integer.

Effectively, this implies that the pointer is pointing to a value that shouldn't be changed.

Q10. What is int *const?

Ans. int *const is a constant pointer to integer

Q11. Const Qualifier in C ?

Ans. The qualifier const can be applied to the declaration of any variable to specify that its value will not be changed (Which depends upon where const variables are stored, we may change value of const variable by using pointer).

Q12. Can const object can only call const functions.?

Ans. Yes

Q13. Const keyword can be put after the variable name or before variable name?

Ans. Both are ok. But most programmers prefer to put const keyword before the variable name.

Q14. Why copy constructor argument should be const in C++?

Ans. One reason is to prevents objects from accidentally modified.

Q15. Explain Const member functions in C++.

Ans. Const member functions in C++ uses const keyword in function's declaration. This not allowed them to modify the object on which they are called. It is recommended practice to make as many functions const as possible so that accidental changes to objects are avoided.

Q16. Can Const member functions call non const object.

Ans. No.

Q17. What is static const?

Ans. static const : “static const” is basically a combination of static(a storage specifier) and const(a type qualifier).

MCQs on Named Constant

Q1. The constants are also called as

A. const

B. preprocessor

C. literals

MCQs Answer

Q1. (c)

Related posts:

1. Sequence Control & Expression | PPL
2. Parse Tree | PPL | Prof. Jayesh Umre
3. Basic elements of Prolog
4. Loops | PPL | Prof. Jayesh Umre
5. Subprograms Parameter passing methods | PPL | Prof. Jayesh Umre
6. Programming Paradigms | PPL | Prof. Jayesh Umre
7. Subprograms Introduction | PPL | Prof. Jayesh Umre
8. Phases of Compiler | PPL | Prof. Jayesh Umre
9. Parse Tree | PPL
10. Influences on Language design | PPL | Prof. Jayesh Umre

11. Fundamentals of Subprograms | PPL | Prof. Jayesh Umre
12. Programming Paradigm
13. Influences on Language Design
14. Language Evaluation Criteria
15. OOP in C++ | PPL
16. OOP in C# | PPL
17. OOP in Java | PPL
18. PPL: Abstraction & Encapsulation
19. PPL: Semaphores
20. PPL: Introduction to 4GL
21. PPL: Variable Initialization
22. PPL: Conditional Statements
23. PPL: Array
24. PPL: Strong Typing
25. PPL: Coroutines
26. PPL: Exception Handler in C++
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++