LOCAL REFERENCING ENVIRONMENTS

Local Variables

- Vars that are defined inside subprograms are called local vars.
- Local vars can be either static or stack dynamic.
- By default an variable in stack dynamic.

Advantages of using stack dynamic variable:

a. Support for recursion.
b. Storage for locals is shared among some subprograms.

Disadvantages of using stack dynamic variable:
a. Allocation/deallocation time required.
b. Indirect addressing “only determined during execution.”
c. Subprograms cannot be history sensitive “can’t retain data values of local vars between calls.”

Advantages of using static variable:
a. Static local vars can be accessed faster because there is no indirection.
b. No run-time overhead for allocation and deallocation.
c. Allow subprograms to be history sensitive.

Disadvantages of using static variable:
a. Inability to support recursion.
b. Their storage can’t be shared with the local vars of other inactive subprograms.

Example for stack dynamic variable:

```
int addition(int a, int b) {
int sum = 0;
sum = a+b;
return sum;
}
```

Example for static variable:

```
//static variable uses static keyword
int addition(int a, int b) {
static int sum = 0;
sum = a+b;
return sum;
}
```

Nested Subprograms

- Residing of a subprogram under another subprogram is known as nested subprogram.
- When a subprogram is placed within another subprogram,it gets hide from the rest of the program.
- Algol 68, Pascal, Ada, JavaScript, Python, Ruby allows nesting of subprogram.
- C, do not allow subprogram nesting.

Video lecture on Local Referencing Environments

References:

1. Sebesta,”Concept of programming Language”, Pearson Edu
2. Louden, “Programming Languages: Principles & Practices” , Cengage Learning

3. Tucker, “Programming Languages: Principles and paradigms “, Tata McGraw –Hill.
4. E Horowitz, “Programming Languages”, 2nd Edition, Addison Wesley

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. 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++