HEAP BASED STORAGE MANAGEMENT

Viva Voce on Heap based storage management

Q1. What is Heap ?

Ans. Heap is the segment where dynamic memory allocation usually takes place.

Q2. Where is heap memory located?

Ans. Heap is located in RAM.

Q3. What happens if heap memory is full ?

Ans. Heap Overflow.

Q4. What is stored in stack and heap?

Ans. Stack is used for static memory allocation and Heap for dynamic memory allocation.

Q5. Which operator is used to delete the memory used from the heap in C++?

Ans. Delete

Q6. What is difference between malloc() and calloc()?

Ans. The malloc() takes a single argument, while calloc() takes two. The malloc() function does not initialize the memory allocated, while calloc() function initializes the allocated memory to ZERO.

Q7. What is new operator in C++?

Ans. The new operator denotes a request for memory allocation on the heap. If sufficient memory is available, new operator initializes the memory and returns the address of the newly allocated and initialized memory to the pointer variable.

Q8. What is difference between static and dynamic memory allocation?

Ans. When memory is allocated at compile time, it is called static memory allocation. When memory is allocated at run time, it is called dynamic memory allocation.

Q9. Can we allocate memory for the objects dynamically in C++?
Ans. YES.
Q10. In C++, dynamic memory allocation is done using which operator?
Ans. New
Q11. Which operator is used to release the dynamically allocated memory in C?
Ans. Free.
Q12. During dynamic memory allocation in CPP, new operator returns what value if allocation is unsuccessful?
Ans. Null.
Q13.What is the return type of malloc() and calloc()?
Ans. Void*
Q14. What is realloc()?
Ans. The realloc() function deallocates the old object pointed to by ptr and returns a pointer to a new object that has the size specified by size.
Q15. Can I increase the size of statically allocated array?
Ans. No.
Q16. Malloc() returns a NULL if it fails to allocate the requested memory?
Ans. True.
Q17. When we dynamically allocate memory is there any way to free memory during run time?
Ans. Yes.
Q18. Specify the two library functions to dynamically allocate memory?
Ans. Malloc() and calloc()
Q19. What is memory leak?
Ans. A memory leak is a common and dangerous problem. It is a type of resource leak. In C language, a memory leak occurs when you allocate a block of memory using the memory management function and forget to release it.

Q20. What is dangling pointer?

Ans. Dangling pointers arise when the referencing object is deleted or deallocated.

MCQs on Heap based storage management

Q1. In Heap Memory is allocated in ….. order.

a. Contiguous block

b. Linear

c. Random

Q2. We use heap for ?

a. to allocate memory dynamically

b. to allocate memory statically

c. to reduce cost time

Q3. Which function is used to deallocate the memory in C++?

a. delete()

b. free()

c. new()

Q4. Which header files should be included to use malloc() or new()?

a. memory.h

b. stdlib.h

c. string.h

Q5. What is the size of Heap?

a. 10 MB

b. 500 MB

c. Size of the heap memory is limited by the size of the RAM and the SWAP Memory.

Q6. Can we implement Linked List without allocating memory dynamically?

a. Yes

b. No

Q7. calloc() returns a storage that is initialized to?

a. Zero

b. Null

c. Nothing

MCQs Answers

Q1. (c)

Q2. (a)

Q3. (b)

Q4. (b)

Q5. (c)

Q6. (b)

Q7. (a)

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: 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++