**II. Керування пам’яттю**

1. Купа та стек.
2. Поняття купи. Призначення. Принципи роботи.
3. Стек. Принцип роботи. Реалізація апаратного стеку в оперативній памяті.
4. Стек викликів функцій. Стековий кадр.
5. Розміщення в стеку параметрів функцій, автоматичних змінних, адреси повернення.
6. Адреса повернення з функції.
7. Вказівники та посилання.
8. Операції над вказівниками. Адресна арифметика. Вказівник типу void\*.
9. Звязок між вказівниками та масивами. Символьні масиви. Масиви вказівників. Аргументи командного рядка.
10. Вказівники на функції.
11. Посилання. Опеарції над посиланнями.
12. Різниця між вказівниками та посиланнями.
13. Робота з памяттю.
14. Функції malloc/realloc/calloc/free.
15. Memory Leaks. ( Інстументи виявлення, Visual Leak Detector ).
16. Параметри та результат функції.
17. Передача параметрів в функції за значенням, через вказівник, через посилання.
18. Передача масиву у функцію.
19. Повернення результатів із функції. Оператор return, поверення результатів через параметри, void-функції.

**III. Основи ООП**

Вступ: Що таке обєкти ? Їх призначення. Порівнння ООП та функціонального підходів до програмування. З чого складається обєкт ( поля, різні для кожного обєкту, та методи – спільні для всіх обєктів ).

1. Області видимості елементів класу. Принцип інкапсуляції. Сеттери і гетери.
2. Inline методи. Розказати, що методи, оголошені inline повинні мати реалізацію в юніті компіляції, з якої використовуються і в результаті inline можна оголошувати або лиш публічні методи з реалізацією прямо в оголошенні класу, або приватні для внутрішнього використання.
3. Конструктори та деструктори.
4. Конструктори, конструктор за замовчуванням, конструктор копіювання. Порядок ініціалізації класу.
5. Список ініціалізації. Порядок виклику. Порівняння списку ініціалізації із присвоєнням в конструкторі.
6. Деструктор. Порядок деініціалізації класу.
7. Виклик конструкторів та деструктора при динамічному виділені памяті. Оператори new, new[], delete, delete[]. Порівняння із функціями стандартної бібліотеки С. Альтернатива nothrow, буферизований оператор new.
8. Переваги використання автоматичних змінних над динамічними ( технологія парних операцій – конструктор/деструктор ).
9. Використання операторів new та delete для полів-вказівників.
10. Виклик деструкторів та конструктора для глобальних обєктів.
11. Неявний виклик конструкторів. Зарезервоване слово explicit.
12. Перевизначення функцій та параметри по замовчуванню.
13. Перевизначення функцій та методів.
14. Параметри по замовчуванню.
15. Перевантаження операцій.
16. Оголошення операторної функції. Бінарні та унарні оператори. Перфіксна та постфіксна форма перевантаження. Стандартні прототипи стандартних операторів ( порівняння, присвоєння, інкременту/декркменту, арифметичних, квадратних дужок, круглих дужок ).
17. Перевантаження в зовнішній функції та дружні функції.
18. Перевантаження оператора присвоєння.
19. Перевантаження operator new та operator delete в класі та глобално.
20. Обмеження на перевантаження операцій.
21. Статичні поля і методи.
22. Поняття статичного поля. Оголошення та інстанціювання як глобальний обєкт.
23. Способи звернення до статичних полів.
24. Обмеження статичних функцій.
25. Застосування статичних членів класу
26. Оголошення класів, структур, енамів в середині класу ( публічне та приватне ).
27. Оголошення класів форвардами :
28. Уникнення зайвого включення h-файлів.
29. Уникнення кругових залежностей при використанні вказівників.
30. Константні поля та методи.
31. Констанні поля, статичні константи, enums. Ініціалізація.
32. Константні методи. Виклик методів константного обєкту.
33. Mutable, const\_cast<> та константні поля.
34. Шаблони. ( Пояснити як вони реалізуються. )
35. Шаблонні функції.
36. Перевантаження та спеціалізація шаблонних функцій.
37. Стандартні параметри в шаблонних функціях.
38. Шаблонні класи. Спеціалізація шаблонних класів. Методи шаблонного классу потрбіно включати в кожен юніт компіляції, що використовує шаблон.
39. Статичні члени в шаблонних класах. Статичні поля шаблону потрібно інстанціювати окремо для кожного набору параметрів, що зустрічається у програмі.
40. Обмеження на шаблони.
41. Розумні вказівники.
42. Поняття та види розумних вказівників.
43. Створення, присвоєння, знищення.
44. Перевантаження операторів розумного вказівника.
45. розумні вказівники, вимагаючи видимості конструктора і деструктора класу, збільшують залежності між заголовочними файлами і не дають використовувати в них оголошення класів форвардами. В результаті вони призводять до більшої зв’язаності модулів і збільшення часу компіляції програми при змінах