Старые вопросы на экзамен по C++

Содержание

- 1 Классы, контейнеры
- 2 Исключения
- 3 Динамический полиморфизм
- 4 Компиляция
- 5 Области видимости
- 6 Шаблоны
- 7 Память
- 8 Ввод-вывод
- 9 STL

Классы, контейнеры

1. Что определяет класс? Чем обличается класс от объекта?

   Класс определяет тип объекта, а объект - это конкретный экземпляр класса.

2. Можно ли объявлять массив объектов? А массив классов?

   Смотря что считать массивом - в обычном смысле можно объявить массив объектов, с помощью метапрограммирования можно делать списки типов(классов)

3. Разрешается ли объявлять указатель на объект? А указатель на класс?

   Да. Нет.

4. Допускается ли передавать объекты в качестве параметров, и какими способами? А возвращать как результат?

   Допускается по значению и по ссылке. Возвращать также можно значение или ссылку.

5. Как называется использование объекта одного класса в качестве поля другого класса?

   Композиция

6. Является ли структура классом? Чем класс отличается от структуры?

   Является, у структуры все поля/методы по умолчанию public, у класса - private.

7. Объясните принцип инкапсуляции.

   Инкапсуляция - сокрытие методов и полей класса от пользователя, в целях сохранения внутренних инвариантов. При этом то, что помечено как public, называется интерфейсом класса.

8. Что такое композиция?

   Это создание нового класса путем объединения уже существующих в единую структуру.

9. Для чего используются ключевые слова public и private?

   см. инкапсуляция, также при public - наследовании все члены базового класса наследуются с теми же модификаторами, при private - все члены базового класса становятся private.

10. Можно ли использовать ключевые слова public и private в структуре?

    да

11. Существуют ли ограничения на использование public и private в классе? А в структуре?

    нет

12. Обязательно ли делать поля класса приватными?

    нет

13. Что такое метод? Как вызывается метод?

    Это функция, являющаяся членом класса. Если она не статическая, при ее вызове в нее неявно передается указатель на экземпляр, от которого она вызывается.

14. Может ли метод быть приватный?

    да

15. Как определить метод непосредственно внутри класса? А вне класса? Чем эти определения отличаются?

    Просто берем и определяем :) Вне класса - в описании класса пишем только объявление метода, вне класса реализацию.

16. Можно в методах присваивать параметрам значения по умолчанию?

    да

17. Что обозначается ключевым словом this?

    указатель на экземпляр класса, для которого вызван текущий метод. В статическом методе this недоступен.

18. Зачем нужны константные методы? Чем отличается определение константного метода от обычного?

    константный метод гарантирует, что мы не изменим состояние класса в процессе его работы. После сигнатуры надо написать const.

19. Может ли константный метод вызываться для объектов-переменных? А обычный метод — для объектов-констант?

    Да. Нет.

20. Объясните принцип полиморфизма.

    Полиморфизм - возможность работать с разными типами одинаковым образом. Статический - перегрузка функций, шаблоны. Динамический - виртуальные функции.

21. Сколько места в памяти занимает объект класса? Как это узнать?

    sizeof(a)

22. Каков размер «пустого» объекта?

    Здесь вроде понятно, что он должен быть ненулевой(у меня 1 байт, например), там добавляется фейковый член чтобы вообще можно было как-то хранить его в памяти.

23. Влияют ли методы на размер объекта?

    Если есть виртуальные методы, то размер увеличивается засчет указателся на таблицу виртуальных функций.

24. Одинаков ли размер класса и аналогичной структуры?

    да

25. Какие операции нельзя перегружать? Как вы думаете, почему?

    a ? b : c - в Страуструпе говорится что нет особых причин запрещать его перегрузку, так что непонятно.
    a::b - это вообще говоря не какая-то функция, а просто удобный способ избежать конфликта имен.
    a.b - непонятно, как тогда вызывать методы, и могут возникать конфликты ([1] (http://pastebin.com/fJm050jm))
    sizeof(a), typeid(a) - видимо, что-то низкоуровневое, тоже надо поподробнее узнать
    a.*b - вообще без понятия что он делает

26. Можно ли перегружать операции для встроенных типов данных?

    Нет

27. Можно ли при перегрузке изменить приоритет операции?

    Нет

28. Можно ли определить новую операцию?

    Нет

29. Перечислите особенности перегрузки операций как методов класса. Чем отличается перегрузка внешним образом от перегрузки как метода класса?

    Первый параметр оператора как метода класса - всегда неявно *this. Внешним образом нельзя обратиться к private полям класса. По этой причине, видимо, оператор>>/<< делается friend'ом - так как там он фактически перегружается внешним образом.

30. Какой результат должны возвращать операции с присваиванием?

    Ссылку на *this

31. Как различаются перегруженная префиксная и постфиксная операции инкремента и декремента?

    Префиксная(++a): T & operator++(); Постфиксная(a++): T operator++(int); - с фейковым параметром чтобы как-то отличать

32. Что означает выражение *this? В каких случаях оно используется?

    Разыменование указателя на текущий экземпляр. Например, чтобы вернуть объект :)

33. Какие операции не рекомендуется перегружать как методы класса? Почему?

    Нужно перегружать только те, которые меняют private члены(типа +=, -= и т.д.). А вот уже +, - и т.д. - могут воспользоваться уже перегруженными +=, -=. Не рекомендуется перегружать ||, &&, так как по умолчанию они оптимальны - т.е. если установлена истинность одного аргумента, второй даже не обрабатывается.

34. Какие операции разрешается перегружать только как методы класса?

    =, [], (), ->. Вопрос - почему?

35. Дайте определение дружественной функции. Как объявляется дружественная функция? А как определяется?

    Дружественная для данного класса функция - которой предоставлен доступ к private и protected членам. Функция объявляется внутри класса, к которому надо предоставить доступ, определяется там, где нужно(уже без модификатора friend).

36. Дайте определение конструктора. Каково назначение конструктора? Перечислите отличия конструктора от метода.

    Конструктор - функция, которая вызывается при просто объявлении(конструктор по умолчанию) или инициализации объекта. Назначение - получать объект класса с какими-то начальными параметрами. Конструктор для данного можно вызвать только один раз.

37. Сколько конструкторов может быть в классе? Допускается ли перегрузка конструкторов? Какие виды конструкторов создаются по умолчанию?

    Сколько угодно. Допускается. По умолчанию создается пустой конструктор и конструктор копирования.

38. Может ли конструктор быть приватным? Какие последствия влечет за собой объявление конструктора приватным?

    Может. Последствия - объект не может быть сконструирован этим конструктором извне.

39. Приведите несколько случаев, когда конструктор вызывается неявно.

    bigint a;(вызван пустой конструктор)
    bigint a = "123214";(вызван конструктор от string)
    bigint a = 10;(вызван констрктор от long long)

40. Как проинициализировать динамическую переменную?

    T* t = new T();

41. Как объявить константу в классе? Можно ли объявить дробную константу?

    Просто взять и объявить. Да.

42. Каким образом разрешается инициализировать константные поля в классе?

    Инициализировать ее можно только через список инициализации. Можно еще static const, ее нужно инициализировать сразу при объявлении

43. В каком порядке инициализируются поля в классе? Совпадает ли этот порядок с порядком перечисления инициализаторов в списке инициализации конструктора?

    В порядке объявления. Нет.

44. Какие конструкции C++ разрешается использовать в списке инициализации качестве инициализирующих выражений?

    Где бы это найти?

45. Какой вид конструктора фактически является конструктором преобразования типов?

    Видимо, любой неявный констркутор от одного аргумента.

46. Для чего нужны функции преобразования? Как объявить такую функцию в классе?

    Что это??

47. Как запретить неявное преобразование типа, выполняемое конструктором инициализации?

    Для этого есть ключевое слово explicit

48. Какие проблемы могут возникнуть при определении функций преобразования?

    ????

49. Для чего служит ключевое слово explicit?

Чтобы не позволить вызвать конструктор от одного аргумента неявно.

50. Влияет ли наличие целочисленных констант-полей на размер класса?

    Они ничем не отличаются от обычных не-const полей.

51. Разрешается ли объявлять массив в качестве поля класса. Как присвоить элементам массива начальные значения?

    Разрешается, надо написать что-нибудь вроде int a[] = {1, 2, 3};

52. Сколько операндов имеет операция индексирования []? Какой вид результата должна возвращать эта операция?

    Операции индексирования нужен один аргумент - индекс. Возвращать T или T& в зависимости от того, можно ли менять объект.

53. Для чего нужны статические поля в классе? Как они определяются?

    Это какие-то параметры класса. Определяются с помощью ключевого слова static

54. Как объявить в классе и проинициализировать статический константный массив?

    struct S{static const int a[];}; const int S::a[] = {1, 2, 3};

55. Что такое выравнивание и от чего оно зависит? Влияет ли выравнивание на размер класса?

    Это способ расположения данных для ускорения доступа к ним. Да, размер может увеличиться.

56. Дайте определение контейнера.

    Это объект, который хранит "коллекцию" других объектов. Контейнер сам заботится о выделении памяти и других низкуровневых штуках, а нам предоставляет доступ к объектам, которые в нем находятся.

57. Какие виды встроенных контейнеров в C++ вы знаете?

    vector, deque, set, map, list

58. Какие виды доступа к элементам контейнера вам известны?

    Для чтения и(или) записи в одном/двух направлениях. Также есть произвольный доступ.

59. Чем отличается прямой доступ от ассоциативного?

    Прямой(direct access) - это вроеде через оператор[], а что такое ассоциативный??

60. Перечислите операции, которые обычно реализуются для последовательного доступа к элементам контейнера.

    Видимо, инкремент итератора.

61. Дайте определение итератора.

    Это объект - обертка над указателем на какой-либо элемент контейнера. В нем должны быть реализованы хотя бы операции инкремента и разыменования.

62. Можно ли реализовать последовательный доступ без итератора? В чем преимущества реализации последовательного доступа с помощью итератора?
63. Что играет роль итератора для массивов C++?

    Хм, указатели, видимо.

64. Что такое деструктор? Может ли деструктор иметь параметры?

Деструктор - то что будет вызвано после того, как объект выходит из своей области видимости для того, чтобы корректно освободить то, что выделяли в динамической памяти.

65. Почему для классов-контейнеров деструктор надо писать явным образом?

    Потому что там используется динамическая память.

66. Допускается ли перегрузка деструкторов?

    Нет

67. Что такое «глубокое копирование» и когда в нем возникает необходимость?

    Если мы копируем объект, в котором есть поле-указатель, надо скопировать объект, на который указывает тот указатель, а не просто указатель.

68. Какое копирование осуществляет стандартный конструктор копирования?

    Тупо копируются все поля.

69. Чем отличается копирование от присваивания?

    Копирование - инициализация объектом того же типа. Присваивание - замена уже существующего объекта.

70. Объясните, почему в операции присваивания требуется проверка присваивания самому себе?

    Потому что если мы работаем с ссылками, можно попортить объект при очистке памяти:
    struct A{int* a; A(){a = new int(10);} A & operator=(const A &b){delete a; a = new int(*b.a); return *this}
    В этом коде если присваиваем самому себе, неявно мы изменяем b после вызова delete a, и теряем значение *a.

71. Можно ли в качестве операции индексирования использовать операцию вызова функции ()? В чем ее преимущества перед операцией []?

    Можно впринципе. Плюс, видимо, в том что можно передавать сколько хочешь аргументов.

72. Почему необходимо писать два определения операции индексирования? Чем они отличаются?

    Первое определение возвращает ссылку на элемент, которых храним в контейнере, для того чтобы можно было его изменять.
    T & operator[](size_t index);
    Второе определение возвращает новый объект, для случая, когда мы работаем с контейнером, который нельзя менять.
    T operator[](size_t index);

73. Дайте определение вложенного класса.

    Класс внутри класса(and we need to go deeper :))

74. Можно ли класс-итератор реализовать как внешний класс? А как вложенный? В чем отличия этих методов реализации?
75. Может ли объемлющий класс иметь неограниченный доступ к элементам вложенного класса? А вложенный класс — к элементам объемлющего?

    Я пока не очень понимаю, как иметь допуск не к static элементам. Надо получше про это узнать.

76. Ограничена ли глубина вложенности классов?

    ?? Где про это можно узнать?

77. Можно ли определить вложенный класс внешним образом? Зачем это может понадобиться?

Да. Зачем - непонятно.

78. Каким образом вложенный класс может использовать методы объемлющего класса? А объемлющий — методы вложенного?

    Я пока не очень понимаю, как иметь допуск не к static элементам. Надо получше про это узнать.

79. Что такое «запредельный» элемент, какую роль он играет в контейнерах?

    В стандартных контейнерах это end(), используется для обозначения конца контейнера, возвращения в случае неудачного поиска, и т.д.

80. Объясните, по каким причинам трудно написать универсальный контейнер, элементы которого могут иметь произвольный тип.

Исключения

1. Назовите ключевые слова C++, которые используются для обработки исключений.

   try, catch, throw

2. Исключение — это:
   1. событие;
   2. ситуация;
   3. объект;
   4. ошибка в программе;
   5. прерывание;
3. Каким образом исключение генерируется?
4. Каковы функции контролируемого блока?
5. Что обозначается ключевым словом catch?
   1. контролируемый блок;
   2. блок обработки исключения;
   3. секция-ловушка;
   4. генератор исключения;
   5. обработчик прерывания;
6. Какого типа может быть исключение?

   вроде можно кинуть что угодно

7. Сколько параметров разрешается писать в заголовке секции-ловушки?
8. Какими способами разрешается передавать исключение в блок обработки?

   по значению и по указателю?

9. Объясните, каким образом преодолеть ограничение на передачу единственного параметра в блок обработки.
10. Почему нельзя выполнять преобразования типов исключений при передаче в секцию-ловушку?
11. Напишите конструкцию, которая позволяет перехватить любое исключение.
12. Могут ли контролируемые блоки быть вложенными?
13. Зачем нужен «контролируемый блок-функция» и чем он отличается от обычного контролируемого блока?
14. Перечислите возможные способы выхода из блока обработки.
15. Каким образом исключение «передать дальше»?
16. Сколько секций-ловушек должно быть задано в контролируемом блоке?
17. Что такое «спецификация исключений»?
18. Что происходит, если функция нарушает спецификацию исключений?
19. Учитывается ли спецификация исключений при перегрузке функций?
20. Что такое «иерархия исключений»?
21. Существуют ли стандартные исключения? Назовите два-три типа стандартных исключений.

    runtime_error, bad_alloc

22. Поясните «взаимоотношение» исключений и деструкторов.

23. Объясните, зачем может понадобиться подмена стандартных функций завершения.
24. Какие виды нестандартных исключений вы знаете?
25. В чем отличие механизма структурной обработки исключений Windows от стандартного механизма?

Динамический полиморфизм

1. Какие две роли выполняет наследование?
2. Какие виды наследования возможны в C++?

   Простое и множественное?

3. Чем отличается модификатор доступа protected от модификаторов private и public?

   ну очевидно

4. Чем открытое наследование отличается от закрытого и защищенного?

   тоже очевидно -

   какие все тут, блин, умные, всем всё очевидно.

   а вот и не очевидно, читайте книги, читайте страуструпа

5. Какие функции не наследуются?

   Чисто виртуальные

6. Сформулируйте правила написания конструкторов в производном классе.

   ???

7. Каков порядок вызова конструкторов? А деструкторов?

   Конструкторы - сначала базовый, потом производный. Деструкторы наоборот.

8. Можно ли в производном классе объявлять новые поля? А методы?

   да. да

9. Если имя нового поля совпадает с именем унаследованного, то каким образом разрешить конфликт имен?

   конфликта не будет

10. Что происходит, если имя метода-наследника совпадает с именем базового метода?

    Метод перегружается, вроде как

11. Сформулируйте принцип подстановки.

    Если это принцип подстановки Лисков, то: свойство, верное для базового класса, должно выполняться и для наследника

12. Когда выполняется понижающее приведение типов?

    Это приведение ссылки на базовый класс к производному.

13. Объясните, что такое «срезка» или «расщепление».

    Видимо, это копирование полей производного объекта в базовый тип.

14. Объясните, зачем нужны виртуальные функции.

Чтобы перегрузить их поведение при наследовании.

15. Что такое связывание?

    Связывание - сопоставление вызова функции с телом функции, которую вызовут.

16. Чем «раннее» связывание отличается от «позднего»?

    Раннее связывание - компилятор/компоновщик решают какая функция будет вызвана на этапе компиляции. Позднее - нужная функция выбирается на в рантайме.

17. Какие два вида полиморфизма реализованы в C++?

    Статический и динамический

18. Дайте определение полиморфного класса.

    Класс, в котором есть виртуальные методы.

19. Может ли виртуальная функция быть дружественной функцией класса?

    да

20. Наследуются ли виртуальные функции?

    наследуются по определению :)

21. Каковы особенности вызова виртуальных функций в конструкторах и деструкторах?

    Виртуальные функции, вызванные из конструкторов и деструкторов не будут виртуальными.

22. Можно ли сделать виртуальной перегруженную операцию, например, сложение?

    Можно

23. Может ли конструктор быть виртуальным? А деструктор?

    Нет. Да.

24. Как виртуальные функции влияют на размер класса?

    К размеру класса добавляется размер виртуальной таблицы функций.

25. Как объявляется «чистая» виртуальная функция?

    В конце сигнатуры пишут =0;

26. Дайте определение абстрактного класса.

    Абстрактный класс - класс, содержащий чисто абстрактные функции. У чисто абстрактного класса не может быть экзмепляра, но можно хранить на него указатель.

27. Наследуются ли чистые виртуальные функции?

    Да

28. Можно ли объявить деструктор чисто виртуальным?

    да

29. Чем отличается чистый виртуальный деструктор от чистой виртуальной функции? //чистый деструктор o_O
30. Зачем требуется определение чистого виртуального деструктора?

    чтобы можно было удалять объекты по указателю на базовый класс.

31. Наследуется ли определение чистой виртуальной функции?

    Нет. У чистой виртуальной функции нет смысла в определении.

32. Приведите классификацию целей наследования.

    Специализация базового класса. Реализация интерфейса базового класса(классов). Ограничение базового класса(адаптеры).

33. Объясните разницу наследования интерфейса от наследования реализации.

    Видимо, наследование интерфейса - public - наследование, реализации - protected?

Компиляция

1. Назовите причины, требующие разделения программ на части.

   во-первых, просто декомпозиция, для меньшей путаницы
   во-вторых, для того чтобы уменьшить время компиляции

2. Дайте определение термина «единица трансляции»?

   Данные, которые поступают компилятору для создания объектного файла, уже после того как прошли препроцессинг.

3. Чем отличается файл с исходным текстом от единицы трансляции?

   файл с исходным кодом не всегда является единицей трансляции.

4. Существуют ли в C++ конструкции, позволяющие идентифицировать отдельный модуль?

   Что такое модуль??

5. Какие способы сборки программы вы можете назвать?
6. Что такое «объектный модуль»? Программа, которая «собирает» объектные модули в программу, называется _____ ?

   ???. Программа - линкер или компоновщик.

7. В чем заключается отличие аргумента «файл» от <файл> в директиве #include?

   <file> - ищет в папках, которые указаны в свойствах какой-то переменной Include(в вижаке задается свойствами проекта), "file" - в текущей директории.

8. Что такое ODR?

   One Definition Rule - принцип, по которому, переменная/класс/метод может быть опередлена только один раз в одной из единиц трансляции. При этом объявлять можно сколько угодно.

9. Объясните, что такое «страж» включения и зачем он нужен.

   Это или #pragma once или #ifndef H_NAME #define H_NAME //код хедера #endif. Нужен чтобы соблюдать ODR.

10. Является ли интерфейс класса его определением?
11. Сколько определений класса может быть в единице трансляции?

    одно

12. Сколько определений класса может быть в многофайловой программе?

    несколько

13. Чем отличаются стандартные заголовки <string>, <string.h> и <cstring>?

    <string> - строки C++, библиотеки STL. <string.h> - Строки C. <cstring> - то же, что <string.h>, но все обернуто в пространство имен std.

14. Объясните суть идиомы Pimpl.
15. Что такое делегирование и как его можно использовать для повышения степени инкапсуляции?
16. Каким образом глобальную переменную, определенную в одной единице трансляции, сделать доступной в другой единице трансляции? А константу?
17. Можно ли использовать слово extern при объявлении функций?
18. Как локализовать объявление функции в файле?
19. Чем отличается «внешнее» связывание от «внутреннего» связывания?
20. Что такое «спецификации компоновки»?
21. Какие объекты обладают внутренним связыванием по умолчанию?

Области видимости

1. Какие области видимости имен вы знаете?
2. Для чего используются пространства имен?

   Для какой-то логической группировки имен, и чтобы в область видимости не лезло то, что не нужно.

3. Чем отличаются именованные и неименованные пространства имен?

   К неименованному пространству имен нельзя будет обратиться откуда-то, кроме текущей единицы трансляции.

4. Могут ли пространства имен быть вложенными?

   Да

5. Для чего применяются алиасы пространства имен?

   Для удобства, чтобы каждый раз не писать длинный путь к функции или еще чему-нибудь.

6. Как сделать члены пространства имен доступными в нескольких (в пределе — во всех) файлах программного проекта?

   Вынести объявления всего что в них лежит в хедер.

7. Объясните разницу между статической и динамической инициализацией.

   Видимо, статическая - на этапе компиляции, динамичестая - в рантайме.

8. В чем состоит проблема инициализации глобальных статических переменных?

   Скорее всего в том, что непонятно, в каком порядке они инициализируются.

9. Какие элементы класса можно объявлять статическими?

   Поля и методы.

10. Можно ли объявить в классе статическую константу? А константный статический массив?

    Да. хм, надо попробовать.

11. А какие статические поля можно инициализировать непосредственно в классе?
12. Как определяются статические поля? В какой момент работы программы выполняется инициализация статических полей?
13. Сколько места в классе занимают статические поля ?

    нисколько

14. Чем отличается статический метод от обычного?

    Статический метод - метод класса, у него нет доступа к this.

15. Какие методы класса не могут быть статическими?

    Видимо, конструктор и деструктор?

16. Какие применения статических полей вы можете привести? А каким образом применяются статические методы?

    Тут можно привести нашу длинку, например, и то, как мы храним базу. Например, можно сделать класс со статическими методами Math(как в джаве), и реализовать там всякие математические операции, очень даже удобно.

17. Приведите структуру и принцип действия паттерна Singleton.

Шаблоны

1. Для чего предназначены шаблоны?
2. Какие виды шаблонов в C++ вы знаете?
3. Объясните термин «инстанцирование шаблона».
4. В чем разница между определением и объявлением шаблона?
5. Объясните назначение ключевого слова typename.
6. Какие виды параметров разрешается задавать в шаблоне класса? А в шаблоне функции?
7. Можно ли параметрам шаблона присваивать значения по умолчанию?
8. Может ли параметром шаблона быть другой шаблон? Каковы особенности объявления параметра-шаблона?
9. Что такое специализация шаблона? Объясните разницу между полной и частичной специализацией.
10. Разрешается ли специализировать шаблон функции?
11. Может ли класс-шаблон быть вложенным в другой класс-шаблон? А в обычный класс?
12. Можно ли объявить в классе шаблонный метод? А шаблонный конструктор?
13. Можно ли перегружать функцию-шаблон?
14. Какие параметры функции-шаблона выводятся автоматически?
15. Может ли шаблон класса быть наследником обычного класса? А обычный класс от шаблона?
16. Объясните, что такое класс свойств (класс трактовок).
17. Каким образом можно использовать возможность наследования обычного класса от шаблона?
18. Может ли шаблонный конструктор быть конструктором по умолчанию?
19. Для чего применяются директивы явного инстанцирования?
20. Объясните, в чем состоят проблемы, возникающие при разделении шаблонного класса на интерфейс и реализацию?
21. Что такое «модель явного инстанцирования» и как она работает?
22. Может ли шаблонный класс иметь «друзей»?
23. Какие проблемы возникают при объявлении дружественной функции для класса-шаблона?
24. Разрешается ли определять в классе-шаблоне статические поля? А статические методы?
25. Что такое «инициализация нулем»?

Память

1. Что является единицей памяти в C++? Какие требования к размеру единицы памяти прописаны в стандарте C++?

    Байт. Чтобы в нее вмещался любой символ, причем не требуется чтобы в байте было 8 бит.

2. В каких единицах выдает результат операция sizeof? Какие типы данных имеют размер 1?

    В байтах. char(signed/unsigned), bool

3. Какие три вида памяти входят в модель памяти C++?
4. Сколько видов динамической памяти обеспечивает C++?
5. Какие функции для работы с динамической памятью достались C++ по наследству от C? В какую библиотеку они включены?

malloc, calloc, realloc, free. <cstdlib>

6. Какие функции выделяют память, и с помощью каких функций память освобождается?

   malloc, calloc, realloc, new([]) выделяют, free, delete([]) освобождают(realloc, видимо, тоже может)

7. Какое важное отличие имеет функция calloc() от функции malloc()?

   calloc не только

8. Какие действия выполняют функции выделения памяти, если память не может быть выделена?

   Бросит исключение.

9. Зависит ли объем выделенной памяти от типа указателя? Влияет ли выравнивание на объем выделяемой динамической памяти?

   Сколько скажешь, столько и выделит.

10. Можно ли с помощью функции realloc() уменьшить объем выделенной памяти?

    Да

11. Что произойдет, если функции free() передать в качестве аргумента нулевой указатель?

    Ничего

12. В чем главное отличие объектно-ориентированного механизма new/delete от механизма malloc()/free()?

    new - бросает исклчение, вызывает конструкторы, по умолчанию. malloc - выделяет просто сырую память, возвращает null, если не получилось.

13. Сколько существует форм new/delete? В чем их отличие?

    new, delete, new[], delete[]. Первые два - для объектов, вторые - для массивов объектов.

14. Какие типы являются POD-типами? Чем отличается работа механизма new/delete с POD-объектами и nonPOD-объектами?

    Plain Old Data Structure in C++ is an aggregate class that contains only PODS as members, has no user-defined destructor, no user-defined copy assignment operator, and no nonstatic members of pointer-to-member type

15. Какие функции выполняет обработчик new?

???

1. Можно ли реализовать собственный обработчик new и «прицепить» его к механизму new/delete?

???

1. В чем главное отличие объединения от других видов классов C++?

   Размер объединения всегда равен размеру максимального его члена. Соответственно, в любой момент он корректно хранит только одно из своих полей.

2. Может ли объединение участвовать в иерархии наследования?
3. Разрешается ли определять для объединения конструкторы и деструктор? А виртуальные функции?

   Да. Нет. Нет.

4. В чем похожи и чем отличаются объединение и размещающий new?

5. Объясните, почему при использовании размещающего new нужно явным образом вызывать деструктор?
6. Зачем нужны интеллектуальные указатели?

   Чтобы не запутаться в new/delete. Умные указатели сами следят за объектами, которыми владеют.

7. Что такое «стратегия владения»? Сколько стратегий владения вы знаете?

   RAII. Garbage Collector тоже вроде ок?

8. Какой интеллектуальный указатель реализован в стандартной библиотеке STL, и какую стратегию владения он реализует?

   auto_ptr, RAII

9. Объясните, в чем преимущества и недостатки интеллектуальных указателей со счетчиком ссылок.

   преимущества - указатель владеет объектом. Безопасность в плане исключений. Недостатки - циклические ссылки и возможная путаница с обычными указателями.

10. Разрешается ли перегружать new и delete и какими способами?
11. Опишите схему функции, перегружающей глобальную функцию new.
12. Отличается ли реализация перегруженной функции new[]() для массивов от реализации «обычной» функции new()?
13. Как вы думаете, почему функции new/delete, перегружаемые для класса, являются статическими?
14. Зачем при перегрузке new/delete для класса нужно проверять размер запрашиваемой памяти?
15. Объясните, чем определяется «динамичность» контейнеров?
16. Что такое «стратегия распределения памяти», и какие стратегии выделения памяти вы знаете?
17. Рассмотрите следующую стратегию распределения памяти: память выделяется для нескольких элементов блоками фиксированной длины, но блоки связываются в список. Для какого вида контейнера можно использовать такую стратегию?
18. Какие операции можно перегрузить для доступа к элементам двумерного массива?
19. В чем заключаются сложности использования операции индексирования [] для доступа к элементам двумерного массива?
20. Каковы способы реализации операций с контейнерами?
21. Какую конструкцию можно назвать «обобщенный алгоритм»?
22. Каким образом объявить указатель на метод?
23. Объясните разницу между указателем на функцию и указателем на метод.
24. Каким образом получить адрес метода?
25. Можно ли указателю на функцию присваивать адрес метода?
26. Какие операции определены в C++ для косвенного вызова метода через указатель?
27. Что такое «функтор»? Приведите пример функционального класса.

   Это класс с перегруженным оператором(). Например, можно сделать абстрактный интерфейс унарной функции, у которой будет виртуальный оператор().

28. Какими способами функтор вызывается?
29. Можно ли использовать наследование при разработке функторов?
30. Разрешается ли операцию вызова функции () определять как виртуальный метод? А как статический?
31. В чем преимущества функторов перед указателями на функции?
32. Объясните, зачем нужны адаптеры функторов? Какие виды адаптеров вы знаете?
33. Как используются классы свойств при разработке функторов?
34. Объясните, что такое «композиция» и приведите примеры?
35. Объясните, чем отличается множественное наследование от простого?
36. Приведите структуру и принцип действия паттерна Adapter.
37. Сформулируйте основную проблему множественного наследования.

   Читай ромбическое наследование(diamond problem)

38. Выполняется ли принцип подстановки при открытом множественном наследовании?
39. Что такое виртуальное наследование? Каковы его преимущества и недостатки по сравнению с обычным наследованием?
40. Может ли виртуальное наследование быть одиночным?

41. Влияет ли виртуальное наследование на размер класса?
42. Объясните, каким образом с помощью виртуального наследования можно вообще запретить наследование.
43. Какие средства C++ составляют RTTI?

    dynamic cast и typeid

44. Объясните разницу между повышающим, понижающим и перекрестным приведением.
45. Какими свойствами должен обладать класс, чтобы с ним работал механизм RTTI?
46. В чем приведение указателей отличается от приведения ссылок?
47. Какие исключения связаны с механизмом RTTI?

Ввод-вывод

1. Что такое «поток» — дайте определение.

   Поток - абстрактный объект, который представляет собой определенное устройство для выполнения операций ввода/вывода. Обычно представляет собой источник/получатель символов неопределенной длины. Чаще всего ассоциируются с физическими источниками/получателями символов, такие как файлы, клавиатура, консоль, etc

2. Как классифицируются потоки, реализованые в библиотеках ввода/вывода C++?

   (???) Вопрос под вопросом.

3. Что такое буферизация и зачем она нужна?

   Буферизация нужна для более эффективной передачи данных: поток с буферизацией собирает символы в streambuf, который хранит символы в массиве, пока не будет вынужден из-за переполнения записать их по назначению. Небуферизованный ввод-вывод сразу осуществляет передачу символов, не заботясь о том, чтобы сделать передачу эффективной.

4. Какие библиотеки ввода/вывода реализованы в C++ и чем они отличаются?

   Основная библиотека для ввода/вывода в C++ - iostream. Ее организация:

   - <ios>, <istream>, <ostream>, <streambuf> and <iosfwd> aren't usually included directly in most C++ programs. They describe the base classes of the hierarchy and are automatically included by other header files of the library that contain derived classes.
   - <iostream> declares the objects used to communicate through the standard input and output (including cin and cout).
   - <fstream> defines the file stream classes (like the template basic_ifstream or the class ofstream) as well as the internal buffer objects used with these (basic_filebuf). These classes are used to manipulate files using streams.
   - <sstream>: The classes defined in this file are used to manipulate string objects as if they were streams.
   - <iomanip> declares some standard manipulators with parameters to be used with extraction and insertion operators to modify internal flags and formatting options.

5. Перечислите стандартные потоки и объясните их назначение.

   Стандартный поток ввода: std::cin, вывода - std::cout, error output stream - std::cerr

6. Зачем нужен процесс форматирования и когда он выполняется?

   (???) Зачем - понятно, но когда?

7. Что такое «форматная строка», и в каких функциях она используется?

1. Объясните назначение элементов спецификатора формата.
2. Сколько спецификаторов формата может быть в форматной строке?
3. Какой из элементов спецификатора формата не является умалчиваемым?
4. Перечислите несколько известных вам обозначений типов в спецификаторе формата, и укажите их назначение.

5. Сколько модификаторов типа вы знаете, и какую роль модификатор типа играет в спецификаторе формата?
6. С помощью какого флага можно выровнять выводимое значение влево? А каким образом вывести ведущие нули?
7. Какое действие оказывают на выводимую строку ширина, точнойть и флаги в спецификаторе формата?
8. Для чего в спецификаторе формата может использоваться символ звездочка («*»)? Чем отличается действие этого символа при воде и при выводе?
9. Каковы особенности ввода строк?
10. Каким образом ограничить набор вводимых символов при вводе?
11. Что является главной проблемой при использовании форматного ввода/вывода из библиотеки <cstdio>?
12. Объясните, для чего нужны строковые потоки. Почему строковые потоки —всегда форматируемые?
13. С помощью каких функций выполняется работа со строковыми потоками?
14. Можно ли использовать тип string (и каким образом) со строковыми потоками?
15. Объясните, в чем заключается различие между текстовым и двоичным файлом.
16. Объясните, что означает «открыть» файл и «закрыть» файл?
17. Каким образом внешний файл связывается с потоком?
18. Можно ли один и тот же поток связать с разными файлами? А один и тот же файл с разными потоками?
19. Перечислите режимы открытия файла. Чем отличается режим “r” от режима “a”?
20. Какую роль в режиме открытия играет знак плюс («+»)?
21. В каких случаях необходимо следить за ситуацией «конец файла»? Каким способом это делается?
22. Можно ли текстовый файл открыть как двоичный? А двоичный — как текстовый?
23. Какие функции ввода/вывода используются для обмена с текстовыми файлами?
24. Перечислите функции ввода/вывода для работы с двоичными файлами.
25. Какие функции реализованы в библиотеке <cstdio> для обеспечения прямого доступа к записям двоичного файла? Можно ли их использовать для работы с текстовыми файлами?
26. Объясните назначение функции fseek().
27. Чем отличается функция ftell() от функции fgetpos()?
28. Объясните, что означает «перенаправление» потока? Какие потоки можно перенаправлять и куда?
29. Каким образом перенаправление ввода можно использовать для ввода строк с пробелами?
30. В чем преимущества объектно-ориентированной библитеки по сравнению с процедурной?
31. В каких состояних может находиться поток? Каким образом отслживается состояние «конец потока»?
32. Какие объектно-ориентированные потоки связаны со стандартными потоками?
33. Чем отличаются объектно-ориентированные строковые потоки от процедурных строковых потоков?
34. Каким образом строковые потоки можно использовать для ограничения ширины поля ввода? А можно ли с той же целью использовать строковые потоки <cstdio>?
35. Сравните средства форматирования объектно-ориентированной и процедурной библиотеки.
36. Каким образом ввести строку типа string с пробелами?
37. Каково назначение флагов форматированя? Какие средства реализованы в библиотеке для работы с флагами форматирования?
38. Что такое «манипулятор»? В чем преимущества манипуляторов перед флагами форматирования?
39. Как связываются файлы с потоками в объектно-ориентированной библиотеке?
40. Можно ли файлы, записанные функциями библиотеки <cstdio>, прочитать объектно-ориентированными средствами? А наоборот?
41. Перечислите режимы открытия объектно-ориентированных файловых потоков. каким образом комбинируются режимы открытия файлоавых потоков?
42. Обязательно ли закрывать файл, связанный с объектно-ориентированным файловым потоком? А открывать?
43. Каким образом открыть файловый поток для чтения и записи одновременно?
44. Как открыть файловый поток для дозаписи?
45. Можно ли вывести значение переменной в двоичном виде и как это сделать?
46. Разрешается ли наследовать от классов библиотеки ввода/вывода?
47. Каким образом можно еренаправить объектно-ориентированный поток?
48. Как используется буфер потока для копирования потока?
49. Какими операциями выполняется форматированный ввод/вывод в файловые потоки? А неформатированный?
50. Реализованы ли в объектно-ориентированной библиотеке средства прямого доступа к файловым потокам? Сравните их с аналогичными средствами библиотеки <cstdio>.
51. С какими объектно-ориентированными потоками разрешается, и с какими не разрешается использовать средства прямого доступа?

52. Покажите, каким образом можно выполнить перегрузку операций ввода/вывода для нового типа данных.
53. Как выполняется обработка ошибок ввода/вывода в объектно-ориентированной библиотеке?
54. Какое стандартное исключение генерируется при ошибках ввода/вывода? Обязательно ли оно генерируется?
55. Чем стандартные широкие потоки отличаются от узких?
56. Что такое — «локаль», и каково ее назначение?
57. Как установить русский шрифт при выводе в консольное окно?
58. Чем отличается ли ввод/вывод широких файловых потоков от узких?

STL

1. Перечислите все последовательные контейнеры стандартной библиотеки. Чем они отличаются друг от друга?
2. Перечислите адаптеры последовательных контейнеров и дайте их подробную характеристику.
3. Почему для адаптеров-очередей нельзя использовать вектор в качестве базового?
4. Чем простая очередь queue отличается от приоритетной очереди priority_queue?
5. Каким требованиям должны удовлетворять элементы контейнера?
6. Могут ли быть указатели элементами контейнера? А итераторы?
7. Почему нельзя использовать в качестве элементов контейнера стандартный интеллектуальный указатель auto_ptr?
8. Зачем в контейнере list реализованы собственные методы сортировки поиска и слияния? Можно ли пользоваться соответствующими стандартными алгоритмами при обработке списка?
9. Перечислите типовые виды конструкторов, с помощью которых можно создавать последовательный контейнер.
10. Можно ли инициализировать контейнер элементами встроенного массива? А элементами другого контейнера? Какими способами это можно сделать?
11. Почему конструктор инициализации, параметрами которого являются итераторы, сделан шаблонным во всех контейнерах?
12. Какие методы реализованы в контейнере-векторе для доступа к элементам?
13. Отличается ли функция at() доступа по индексу от перегруженной операции индексирования и чем?
14. Перечислите методы контейнера deque, относящиеся к определению размеров контейнера.
15. Чем метод size() отличается от метода capacity()? А в чем отличие этих методов от метода max_size()?
16. Перечислите методы контейнера list, предназначенные для вставки удаления и замены элементов. Отличаются ли эти методы от соответствующих методов вектора и дека?
17. Каким образом выполняются операции сравнения контейнеров?
18. Разрешается ли изменять элемент ассоциативного контейнера, доступный в данный момент по итератору?
19. Какие контейнеры называются ассоциативными и почему?
20. Чем контейнер map отличается от контейнера multimap?
21. Объясните, почему в ассоциативных контейнерах нельзя изменять элемент, доступный в данный момент по итератору.
22. По каким причинам в контейнере-множестве не реализованы типовые операции объединения, пересечения, разности и другие?
23. Как используется структура-пара в ассоциативных контейнерах?
24. Объясните, что такое «критерий сортировки», и каким требованиям он должен удовлетворять? Какой критерий сортировки принят по умолчанию?
25. Какими преимуществами обладает функция make_pair() по сравнению с конструктором pair()?
26. Почему в контейнерах-отображениях операция индексирования перегружена, а в контейнерах-множествах — нет?
27. Какие гарантии безопасности обеспечивают контейнеры стандартной библиотеки?
28. Что такое «транзакционная гарантия безопасности» и чем она отличается от базовой?
29. На какие 4 класса по надежности можно разделить все операции с контейнерами?
30. Что такое «распределитель памяти» и зачем он нужен?
31. Чем отличается битовый вектор bitset от битового вектора vector<bool>?
32. Дайте определение итератора.
33. Что такое «начальный» итератор и «конечный» итератор? Какие методы, связанные с итераторами, обязательно включает каждый контейнер?
34. Чем константный итератор отличается от неконстантного?
35. Объясните, что такое «недействительный» итератор. В каких случаях итераторы становятся недействительными?

36. Какие категории итераторов вы знаете? Какие операции обязательно реализуются для всех категорий итераторов?
37. К какому виду итераторов можно отнести встроенный указатель и почему?
38. Какие вспомогательные функции для итераторов вы знаете? В каких случаях оправдано их применение?
39. Какие адаптеры итераторов реализованы в библиотеке?
40. Объясните, почему итераторы реализованы как вложенные классы в контейнерах.
41. Чем отличаются итераторы вставки от обычных итераторов?
42. Каким образом используются потоковые итераторы?
43. Какие стандартные функторы реализованы в библиотеке STL? Каково их основное назначение?
44. Для чего нужны адаптеры функторов bind1st() и bind2nd()?
45. Как применяются адаптеры-отрицатели?
46. Почему алгоритмы remove() не удаляют элементы из контейнеров? Как реально удалить элементы из контейнера?
47. Чем отличается стабильная сортировка от обычной?
48. Какую функцию выполняет алгоритмы unique()?
49. Могут ли стандартные алгоритмы работать со строками?
50. Нужно ли сортировать ассоциативные контейнеры?
51. Можно ли алгоритмы для работы с множествами применять для последовательных контейнеров? При каких условиях?
52. Какие алгоритмы предназначены для заполнения контейнера значениями? С какими контейнерами они могут работать?
53. Каким образом заполнить с помощью алгоритма generate() последовательный контейнер, не имеющий ни одного элемента?
54. Перечислите алгоритмы, предназначенные для операций с каждым элементом контейнера.
55. Можно ли с помощью алгоритма for_each() изменить элементы контейнера?

PS Обращаю Ваше внимание на то, что это - базовый список. В него не включены вопросы, которые обсуждаются на лекциях, поэтому не стоит считать, что они не будут заданы Вам на экзамене.

С уважением, Ковалев Антон

Это вопросы из книги: Лаптев В.В. C++. Объектно-ориентированное программирование. - Спб.: Питер, 2008. - 464 с. Первоначальный список вопросов был опубликован на сайте РСДН (27.04.06): http://rsdn.ru/?Forum/?fuid=18459

Источник — «http://neerc.ifmo.ru/wiki/index.php?title=Старые_вопросы_на_экзамен_по_C%2B%2B&oldid=27180»

- Эта страница последний раз была отредактирована 26 июля 2012 в 16:34.