**Вопросы для самоподготовки к экзамену**

**по дисциплине**

**«Основы алгоритмизации и программирования»**

*Специальность 09.02.07 «Информационные системы и программирование»*

1. **Определение алгоритма.**  
   это последовательность шагов или инструкций, которые выполняются для решения определенной задачи или достижения определенной цели. Он представляет собой формальное описание процесса, который может быть выполнен механически или автоматически. Алгоритмы используются в различных областях, включая математику, информатику, программирование и другие науки. Они позволяют систематизировать и упорядочить выполнение задач, обеспечивая предсказуемость и повторяемость результатов.
2. **Как классифицируются языки программирования?**

Языки программирования могут быть классифицированы по различным критериям.

1. Уровень абстракции: Языки программирования могут быть низкоуровневыми (ближе к машинному коду) или высокоуровневыми (более абстрактными и удобными для программистов).

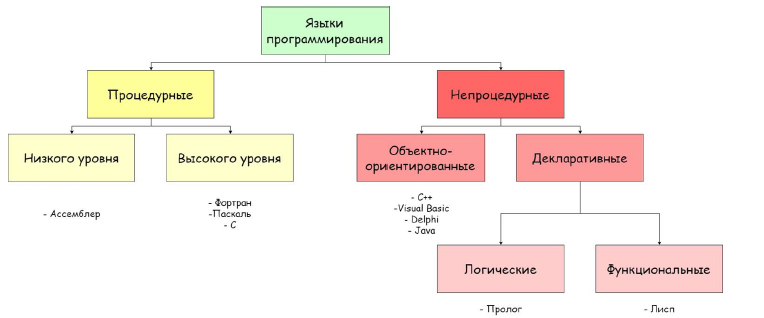
2. Типизация: Языки программирования могут быть статически типизированными (типы проверяются на этапе компиляции) или динамически типизированными (типы проверяются во время выполнения программы).

3. Парадигма программирования: Языки программирования могут быть процедурными, объектно-ориентированными, функциональными, логическими и другими, в зависимости от основных принципов организации кода.

4. Область применения: Языки программирования могут быть специализированными для определенных областей, таких как веб-разработка, научные вычисления, базы данных и т. д.

Это лишь некоторые из способов классификации языков программирования, и многие языки могут попадать в несколько категорий одновременно.

5. По модели исполнения (компилируемые, интерпретируемые)



1. Что характерно для программы, написанной с применением императивной парадигмы программирования?

Императивное программирование — это парадигма программирования (стиль написания исходного кода компьютерной программы), для которой характерно следующее:

* в исходном коде программы записываются инструкции (команды);
* инструкции должны выполняться последовательно;
* данные, получаемые при выполнении предыдущих инструкций, могут читаться из памяти последующими инструкциями;
* данные, полученные при выполнении инструкции, могут записываться в память.

Императивная программа похожа на приказы, выражаемые повелительным наклонением в естественных языках, то есть представляют собой последовательность команд, которые должен выполнить компьютер.

Примером императивного языка программирования является язык C.

1. Что характерно для программы, написанной с применением декларативной парадигмы программирования?

Декларативное программирование — это парадигма программирования, в которой задаётся спецификация решения задачи, то есть описывается, что представляет собой проблема и ожидаемый результат.

* Описание желаемого результата: Вместо описания последовательности шагов для достижения результата, программа фокусируется на описании желаемого результата или связей между данными.
* Отсутствие явного управления потоком выполнения: Программа не указывает точную последовательность операций. Вместо этого, она описывает связи и зависимости между данными, и исполнитель (например, интерпретатор или система управления базами данных) самостоятельно определяет порядок выполнения операций.
* Удобство для выражения сложных связей: Декларативные языки программирования обычно предоставляют более высокоуровневые конструкции для описания сложных связей и операций. Это может облегчать разработку и понимание программы.

Примеры декларативных парадигм программирования включают функциональное программирование, логическое программирование и SQL (Structured Query Language) для работы с базами данных.

1. Что характерно для программы, написанной с применением процедурной парадигмы программирования?

Процедурное программирование — программирование на императивном языке, при котором последовательно выполняемые операторы можно собрать в подпрограммы, то есть более крупные целостные единицы кода, с помощью механизмов самого языка.

* Программа структурирована в отдельные процедуры или функции, которые выполняют определенные задачи.
* Используется последовательное выполнение команд, без возможности одновременного выполнения нескольких задач.
* Программа делится на модули для удобства управления кодом.
* Процедурная программа обычно содержит глобальные переменные, которые могут быть доступны из различных частей программы.
* Изменения в одной части программы могут потребовать изменений в других частях программы, так как они могут быть взаимозависимы.

1. Чем отличаются языки низкого уровня от языков высокого уровня?

Языки низкого уровня и языки высокого уровня отличаются прежде всего своим уровнем абстракции и близостью к машинному исполнению.

Языки низкого уровня, такие как ассемблер, ближе к машинному коду и предоставляют более прямой доступ к аппаратному обеспечению компьютера. Они обладают более высокой производительностью, но требуют более тщательной работы и более сложны в написании.

Языки высокого уровня, такие как Python, Java, C++, предоставляют более абстрактные инструменты и уровень абстракции, что делает их более удобными в использовании и понимании. Они позволяют программистам работать на более высоком уровне и избавляют от необходимости напрямую управлять аппаратным обеспечением.

1. Что характерно для программы, написанной с применением объектно-ориентированной парадигмы программирования?

Объектно-ориентированное программирование (ООП) — методология программирования, основанная на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определённого класса, а классы образуют иерархию наследования.

Для программы, написанной с применением ООП характерно:

* Программа состоит из объектов, которые являются экземплярами классов.
* Классы определяют структуру и поведение объектов.
* Программа использует наследование, полиморфизм и инкапсуляцию для организации и управления объектами.
* Объекты взаимодействуют друг с другом, передавая сообщения и вызывая методы.
* Программа обладает высокой степенью модульности и повторного использования кода.
* Объекты программы могут быть созданы, изменены и удалены во время выполнения программы.

1. В чем отличие языка программирования С++ от C# и Java?

1. Уровень сложности: C++ предоставляет более низкоуровневый доступ к памяти и более гибкий контроль над производительностью, чем C# и Java. Однако это также делает C++ более сложным и требующим больше опыта для работы.

2. Управление памятью: В C++ разработчик самостоятельно управляет памятью, что может привести к ошибкам в виде утечек памяти и некорректного освобождения. С другой стороны, C# и Java автоматически управляют памятью с помощью сборщика мусора, что упрощает процесс разработки и повышает надежность.

3. Платформозависимость: C++ компилируется в машинный код и обеспечивает возможность написания кроссплатформенных приложений, но требует отдельной компиляции для каждой платформы. C# и Java используют виртуальные машины (CLR и JVM соответственно), что обеспечивает им возможность работать на различных платформах без изменений в исходном коде.

4. Синтаксис и особенности языка: C++ имеет более сложный и гибкий синтаксис, позволяющий разработчику использовать указатели, множественное наследование, шаблоны и другие низкоуровневые концепции. C# и Java предоставляют более простые и удобные средства программирования, такие как сборки (assembly) в C# и дженерики (generics) в Java.

1. Определение IDE. Известные IDE.

IDE (Integrated development environment) — это программное обеспечение, предназначенное для создания и разработки программного обеспечения. Оно обычно включает в себя текстовый редактор, средства для отладки и компиляции/интерпретации, библиотеки подпрограмм.

Некоторые известные IDE включают в себя:

- Visual Studio (Microsoft)

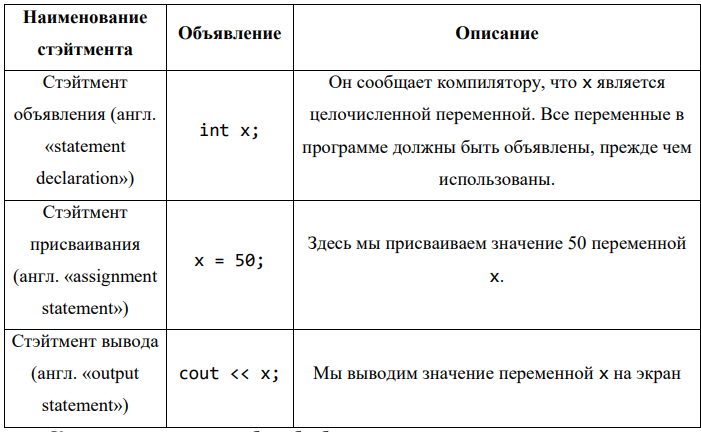
- Xcode (Apple)

- Android Studio (Google)

- PyCharm (JetBrains)

1. Определение стэйтмента. Виды стэйтиментов.

Стэйтмент (англ. «statement») — это тип инструкции в компьютерных программах. Он является наименьшей независимой единицей в языках программирования. Стэйтмент в программировании — это то же самое, что и «предложение» в русском языке. Мы пишем предложения, чтобы выразить какую-то идею. В языках программирования мы пишем стэйтменты, чтобы выполнить какое-то задание. Все стэйтменты в C++ и C# заканчиваются точкой с запятой.



1. Структура программы на языке программирования С++.

Подключение библиотек:

#include <iostream> (директива #include <имя файла> указывает препроцессору взять

содержимое файла (в данном случае – iostream) и включить его в строку, где расположена директива.)

Использование пространства имен (множество, в рамках которого определяются различные идентификаторы):

using namespace std;

Основная функция main (тело программы) – точка входа в программу:

int main() {

// код программы

return 0;

}

Дополнительные функции и классы могут быть определены после функции main (но для этого необходимо создать их прототипы в начале программы, иначе компилятор попросту не сможет запустить код, т.к. ему встретится неизвестная ему функция/класс).

1. Что представляет собой пространство имен?

Пространства имен – это множество, в рамках которого определяются различные идентификаторы (имена типов, функций, переменных и т. д.). Пространства имен используются для организации кода в виде логических групп и с целью избежания конфликтов имен, которые могут возникнуть, особенно в таких случаях, когда база кода включает несколько библиотек. Все идентификаторы в пределах пространства имен доступны друг другу без уточнения.

Чтобы избежать конфликта имен, разрешается также объявлять собственные пространства имен через ключевое слово namespace.

Для того, чтобы указать компилятору место поиска идентификатора в определенном пространстве имен необходимо использовать название необходимого пространства имен вместе с оператором разрешения области видимости (::) и требуемым идентификатором.

1. Комментарии в коде программы.

Комментарий – это способ размещения произвольного текста внутри кода программы. Комментарии игнорируются компилятором и обычно используются программистами для объяснений в коде.

Различают два вида комментариев:

* + - 1. Символ // означает, что следующая строка – комментарий.
      2. 2. Текст, содержащийся между символами /\* и \*/, также являются

комментариями, даже если он занимает несколько строк.

1. Операторы консольного ввода-вывода данных.

Мне лень писать, просто не забудь про $ и { } в C#

String.Format("Name = {0}, hours = {1:hh}", name, DateTime.Now);

Элементами формата являются {0}, индекс которых 0 соответствует объекту name, и {1:hh}, индекс которого 1 соответствует объекту DateTime.Now.

1. Структура программы на языке программирования С#.

1. Директивы using:

- Используются для добавления пространств имен, которые содержат необходимые классы и методы.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

2. Определение пространства имен:

- Пространство имен определяет область видимости классов, методов и других элементов программы.

namespace MyProgram

{

// Код программы

}

3. Определение класса:

- Обычно программа содержит хотя бы один класс, который может содержать метод Main, с которого начинается выполнение программы.

public class MyClass

{

static void Main()

{

// Основная логика программы

}

}

4. Определение методов:

- Методы содержат код исполняемых операций.

public class MyClass

{

static void Main()

{

// Основная логика программы

}

public void MyMethod()

{

// Тело метода

}

}

5. Другие элементы программы:

- Это могут быть поля, свойства, конструкторы, делегаты, события и другие элементы, которые различно организуют логику программы.

1. Определение переменной. Как объявляются переменные в С++ и С#?

В С++ /C# переменные объявляются следующим образом:

Тип\_данных имя\_переменной;

Например:

int age;

1. Что такое константа в языках программирования?

Неизменная переменная, префикс const. константа является именованным значением, которое нельзя изменить

1. Что такое литерал? Как классифицируются литералы?

Литерал - это конкретное значение, используемое в программировании. Литералы классифицируются в зависимости от типа данных, к которому они относятся. Например, литералы могут быть целыми числами, числами с плавающей точкой, строками, символами и т. д.

1. Какая операция применяется для изменения значения переменной?

Для изменения значения переменной применяется операция присваивания. Например, для изменения значения переменной x на 5, мы можем использовать операцию x = 5.

1. Перечислите правила именования переменных.
2. Что такое область видимости переменной? В чем отличие глобальной области видимости от локальной?
3. В чем отличие l-value от r-value?

L-value - это термин, который обозначает объект, который может быть ссылкой. Он может быть присвоен другой переменной или изменен. В основном, это переменные, которые имеют адрес в памяти.

R-value - это термин, который обозначает объект, который не может быть ссылкой. Он обычно не может быть присвоен другой переменной или изменен. Это значения, которые не имеют адреса в памяти.

Таким образом, основное отличие между l-value и r-value заключается в том, что l-value может быть ссылкой, в то время как r-value не может.

1. Что такое тип данных?
2. Как классифицируются типы данных?
3. Дайте характеристику типу данных bool.
4. Дайте характеристику типу данных int.
5. Дайте характеристику типу данных char.
6. Дайте характеристику типу данных double/float.
7. Как определяется размер переменной?

Размер переменной определяется в зависимости от типа данных, которые она содержит. Например, для числовых типов данных (целочисленных, с плавающей запятой) размер определяется в битах или байтах в зависимости от используемого языка программирования. Для строковых переменных размер определяется в зависимости от количества символов в строке и используемой кодировки. Некоторые языки программирования имеют ограничения на размер переменных, а в некоторых случаях размер может динамически изменяться в процессе выполнения программы.

1. Какой инструмент языков программирования позволяет не указывать явно тип данных переменной? Приведите пример.
2. Какие арифметические операции реализованы в языках программирования?
3. В чем отличие постфиксного декремента от префиксного декремента?

int a = 5;

int b;

b = a--; // постфиксный декремент, b будет равно 5, а a будет уменьшено до 4 после этой строки

b = --a; // префиксный декремент, a будет уменьшено до 3, и затем b будет равно 3

1. Какие операторы сравнения и присваивания реализованы в языках программирования? Опишите их.
2. Какие логические операторы реализованы в языках программирования? Опишите их.
3. Какие методы используются для определения типа данных переменной?

Метод typeof() - возвращает тип данных переменной в виде строки (например, "number", "string", "object", "boolean", "undefined", "function").

Метод Object.prototype.toString.call() - возвращает тип данных переменной в виде строки с помощью вызова метода toString объекта, передаваемого в качестве контекста.

1. Какие существуют способы для неявного преобразования типов?

Неявное преобразование типов (или «автоматическое преобразование типов») выполняется всякий раз, когда требуется один фундаментальный тип данных, но предоставляется другой, и пользователь не указывает компилятору, как выполнить конвертацию (не использует явное преобразование типов через операторы явного преобразования).

Есть 2 основных способа неявного преобразования типов:

* числовое расширение;
* числовая конверсия.

1. Какие существуют варианты числового расширения?

В языке C++ есть два варианта расширений:

* Интегральное расширение (или «целочисленное расширение»). Включает в себя преобразование целочисленных типов, меньших, чем int (bool, char, unsigned char, signed char, unsigned short, signed short) в int (если это возможно) или unsigned int.
* Расширение типа с плавающей точкой. Конвертация из типа float в тип double.

1. Чем числовая конверсия отличается от числового расширения?

Числовая конверсия отличается от числового расширения следующим образом:

Числовая конверсия - это процесс преобразования числа из одной системы счисления в другую, например, из десятичной системы счисления в двоичную или шестнадцатеричную.

Числовое расширение - это процесс добавления к числу новых цифр, например, добавление дробной части к целому числу или расширение числа за счет добавления нулей в конце.

Таким образом, числовая конверсия связана с изменением системы счисления, в то время как числовое расширение связано с изменением самого числа.

1. Как осуществляется явное преобразование типов в С++?

В языке C++ есть 5 видов операций явного преобразования типов:

* конвертация C-style; float x = (float)i1 / i2;
* применение оператора static\_cast; float x = static\_cast<float>(i1) / i2
* применение оператора const\_cast;
* применение оператора dynamic\_cast;
* применение оператора reinterpret\_cast

1. Как осуществляется явное преобразование типов в С#?

<тип\_преобразования> <переменная\_или\_выражение> = (тип\_преобразования) <значение\_другого\_типа>;

int a = 10;

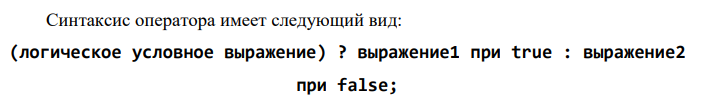
double b = (double) a;

1. Какие операторы используются для реализации ветвления процесса выполнения программы?

Для реализации ветвления процесса выполнения программы используются операторы ветвления, такие как if-else, switch-case, а также операторы цикла, такие как while, for, do-while.

1. Опишите синтаксис условного оператора. Приведите пример его использования.
2. Опишите синтаксис условного тернарного оператора. Приведите пример его использования.

Условный тернарный оператор — это удобное упрощение ветвления if/else



1. Опишите синтаксис оператора множественного выбора. Приведите пример его использования.

switch (выражение) {

case значение1:

// действия, если выражение равно значению1

break;

case значение2:

// действия, если выражение равно значению2

break;

// и так далее

default:

// действия по умолчанию, если ни одно из значений не совпало

break;

}

1. Опишите синтаксис оператора управления потоком выполнения программы. Приведите пример его использования.

Оператор goto — это оператор управления потоком выполнения программ,

который заставляет центральный процессор выполнить переход из одного участка кода

в другой (осуществить прыжок). Другой участок кода идентифицируется с помощью

лейбла.

1. Какие существуют ограничения при использовании оператора goto?

1. Оператор goto может усложнить понимание кода, особенно если он используется часто и в сложных структурах.

2. Использование оператора goto может привести к созданию запутанных и трудно поддерживаемых программ.

3. Оператор goto может нарушить принцип структурного программирования и привести к созданию "спагетти-кода".

4. При использовании оператора goto возможен переход к неожиданным участкам кода, что усложняет отслеживание логики программы.

5. В некоторых языках программирования (например, Python) оператор goto отсутствует вообще, поэтому использование его может привести к непереносимости кода.

6. В многопоточных приложениях использование оператора goto может привести к непредсказуемым последствиям из-за возможных гонок данных и других проблем, связанных с параллельным выполнением кода.

1. Какие операторы используются для реализации многократного исполнения последовательности инструкций в коде программы?

Для реализации многократного исполнения последовательности инструкций в коде программы часто используются циклы. Наиболее распространённые операторы циклов в программировании это:

1. for - цикл со счётчиком, который выполняет указанное количество итераций.

2. while - цикл с предусловием, который выполняется, пока верно заданное условие.

3. do-while - цикл с постусловием, который принудительно выполняет хотя бы одну итерацию, а затем проверяет условие для продолжения выполнения.

1. В чем отличие циклов с предусловием от цикла с постусловием?
2. Опишите синтаксис цикла с предусловием. Приведите пример его использования.
3. Опишите синтаксис цикла с постусловием. Приведите пример его использования.
4. Опишите синтаксис цикла со счетчиком. Приведите пример его использования.
5. Как объявляется бесконечный цикл?

while True:

# код бесконечного цикла

1. Для чего используется оператор break и continue?
2. Что такое массив?
3. Как происходит создание и инициализация массива на языке программирования С++?
4. Как происходит создание и инициализация массива на языке программирования С#?
5. Как происходит доступ к данным, хранимым в массиве?
6. Что такое указатель?
7. Как объявляется указатель в коде программы?
8. Какой оператор используется для записи или чтения данных, содержащихся в области памяти, адрес которой хранит указатель?
9. В каких арифметических операциях могут принимать участие указатели?
10. Что происходит с указателем при увеличении его значения с помощью операции инкремента?
11. Как вычисляется «расстояние» между указателями?
12. Как определяется приоритет операцией при одновременном использовании операции разыменования и постфиксного инкремента? Приведите пример.

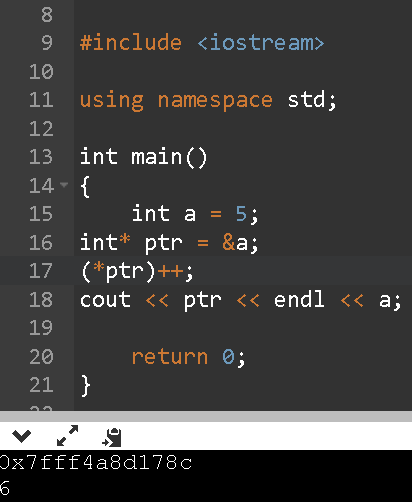
При одновременном использовании операции разыменования (\*) и постфиксного инкремента (++) в выражении, приоритет имеет операция постфиксного инкремента (++).

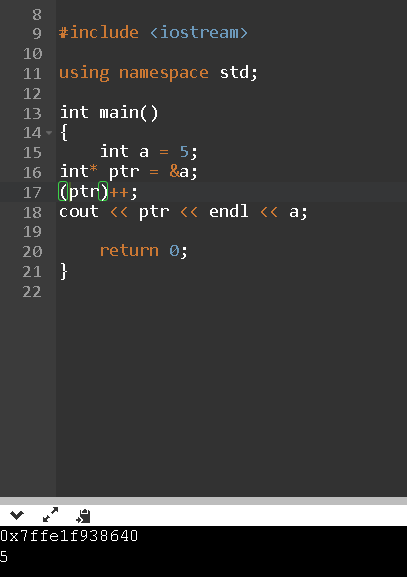
int a = 5;

int\* ptr = &a;

(\*ptr)++;

В этом примере сначала мы разыменовываем указатель ptr, чтобы получить доступ к значению переменной a, затем применяем постфиксный инкремент, увеличивая значение переменной a на 1.





1. Перечислите случаи использования указателей в программировании.
2. Что такое динамическое распределение памяти?
3. В каких случаях возникает утечка памяти?
4. Перечислите комбинации использования ключевого слова const с указателями.
5. Для чего используются указатели на указатели?
6. Как происходит создание двумерного динамического массива?
7. Что такое ссылочная переменная? В чем ее отличие от указателей?
8. Определение функции.
9. Опишите общую форму определения функции на языке С++ и С#.
10. Как происходит вызов функции?
11. Чем отличается передача параметров по значению от передачи параметров по адресу?
12. В С++ функцию нельзя вызвать до объявления самой функции. Какими способами устраняется данная проблема?
13. С помощью какого оператора реализуется возврат функцией определенного значения?
14. Как создаются функции с параметрами по умолчанию? Как происходит вызов функций с параметрами по умолчанию?
15. Как происходит перегрузка функций? Приведите пример перегрузки функций.
16. Перечислите преимущества и недостатки передачи аргументов по значению.
17. Перечислите преимущества и недостатки передачи аргументов по адресу.
18. Когда следует и когда не следует использовать возврат по значению в функции?
19. Когда следует и когда не следует использовать возврат по адресу в функции?
20. Когда следует и когда не следует использовать возврат по ссылке в функции?
21. Что такое указатель на функцию? Как объявляется указатель на функцию?
22. Как происходит передача указателя на функцию в другую функцию и возврат функцией указателя на другую функцию? Приведите пример.
23. Что такое рекурсия? Что такое базовый случай? Приведите пример рекурсивной функции.
24. Что представляет собой делегат? Укажите синтаксис объявления делегата.
25. Как происходит добавление методов в делегат и удаление методов из делегата?
26. В чем заключается принцип объектно-ориентированного программирования?
27. Перечислите основные принципы объектно-ориентированного программирования.
28. Что такое класс?
29. Можете ли вы вызвать метод базового класса, не создавая экземпляр?
30. В чем разница между классом и объектом?
31. Что такое модификатор доступа? Перечислите модификаторы доступа, указав их функциональность.
32. В чем преимущество использования объектно-ориентированной парадигмы программирования?
33. Какой принцип ООП позволяет рассматривать объект в виде самостоятельной программной единицы, полностью отвечающей за хранение и обработку своих данных и предоставляющей пользователям четко определенный набор функций?
34. Как называется элемент данных объекта?
35. Какие типы могут иметь свойства класса?
36. Как называется логически связанный блок кода, связанный с обработкой данных внутри класса?
37. Дайте определение метода-конструктора.
38. Дайте определение метода-деструктора.
39. Какие методы организуют контролируемый доступ к закрытым элементам данных объекта?
40. Перечислите этапы жизненного цикла объекта.
41. Что включает описание класса?
42. Перечислите общие правила для описания свойств класса.
43. Перечислите общие правила для описания методов класса.
44. В каких случаях допускается перегрузка конструкторов класса?
45. В виде какой функции оформляется Set-метод?
46. В виде какой функции оформляется Get-метод?
47. Укажите синтаксические правила для определения класса на языке С++. Приведите пример.
48. Укажите синтаксические правила для определения класса на языке С#. Приведите пример.
49. Как создается экземпляр класса?
50. Как происходит обращение к свойствам и методам экземпляра класса?
51. Какой уровень доступа к элементу данных класса определяет модификатор public?
52. Какой уровень доступа к элементу данных класса определяет модификатор private?
53. Какой уровень доступа к элементу данных класса определяет модификатор protected?
54. Что определяет интерфейс класса?
55. Какой модификатор доступа имеет член класса, если он явно не указан?
56. Чем поле отличается от свойства в языке С#?
57. Укажите стандартный синтаксис для определения свойства и автосвойства в языке С#?
58. При каких условиях можно установить модификатор доступа для блока set или get?
59. Какой может иметь модификатор доступа для блока set или get, если свойство имеет модификатор public?
60. Для чего используются конструкторы класса?
61. Укажите правила описания методов-конструкторов.
62. Что такое конструктор по умолчанию?
63. В каких ситуациях происходит удаление объекта с использованием деструктора?
64. Укажите правила описания методов-деструкторов.
65. В чем отличие значимых типов и ссылочных типов в С#?
66. Как работает копирование значение при присвоении данных объекту значимого типа?
67. Как работает копирование значение при присвоении данных объекту ссылочного типа?
68. Для чего используется ключевое слово ref?
69. В чем заключается назначение ключевого слова this?
70. Какой параметр неявно добавляется в метод класса на этапе компиляции?
71. Что представляют собой статические члены класса? Как они определяются в коде программы?
72. Какие существуют ограничения при определении статических функций (методов)?
73. Как можно осуществить обращение к статическим членам класса?
74. Что представляет собой статический конструктор и статический класс?
75. Какие операции используются для обращения к открытым свойствам и методам классов в языке программирования С++?
76. Чем отличается функциональность ключевого слова this в С++ и С#?
77. Какие переменные и методы следует определять как статические?
78. Какие операторы разрешено перегружать в унарном и бинарном представлении?
79. Какая операции не имеет фиксированного числа операндов при перегрузке?
80. Опишите общую форму метода для перегрузки бинарной операции в языке программирования С#?
81. Перечислите ссылочные типы в языке программирования С#?
82. Перечислите значимые типы языке программирования С#?
83. В чем заключается принцип абстрагирования, используемый при построении классов?