Определение алгоритма.

Алгоритм представляет собой конечную совокупность точно заданного набора инструкций, определяющего порядок действий исполнителя для решения некоторой задачи.

Как классифицируются языки программирования?

1) По парадигме (декларативные, императивные, структурированные и т.п.); 2) По системе типов (динамические, статические, сильно- и слаботипизированные, нетипизированные и т.п.); 3) По уровню абстракции (высокого, низкого уровня); 4) По модели исполнения (компилируемые, интерпретируемые).

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.

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.

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.

Чем отличаются языки низкого уровня от языков высокого уровня?

Язык низкого уровня - это язык программирования, предназначенный для определенного типа компьютера и отражающий его внутренний машинный код. Язык высокого уровня - это язык программирования, предназначенный для удовлетворения требований программиста, он не зависит от внутренних машинных кодов компьютера любого типа.

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. В объектно-ориентированных языках не описывают подробной последовательности действий для решения задачи, хотя они содержат элементы процедурного программирования. Программа пишется в терминах объектов, которые обладают свойствами и поведением. Объекты обмениваются сообщениями.

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

Основное различие состоит в переносимости программ, написанных на Java и C#, и объясняется тем, что программы компилируются не в машинный, а в промежуточный язык. В случае Java - это байт-код. В случае C# он называется CIL. В обоих случая промежуточный код выполняется специальной исполняющей системой. В случае С++ компилятор формирует машинный код, который затем непосредственно выполняется центральным процессором. Если необходимо будет запустить С++-программу в другой системе, необходимо будут перекомпилировать исходный текст программы в машинный код, предназначенный для выбранной вами среды.

Определение IDE. Известные IDE.

Интегрированная среда разработки (IDE) - объединяет этапы программирования, компиляции и компоновки в пределах единого пользовательского интерфейса, предоставляющего также средства отладки, облегчающие обнаружение ошибок и устранение проблем. Наиболее популярные IDE - Visual Studio, CLion, IntelliJ IDEA.

Определение стэйтмента. Виды стэйтиментов.

Стэйтмент — это тип инструкции в компьютерных программах. Он является наименьшей независимой единицей в языках программирования. Виды стэйтментов: Стэйтмент объявления, стэйтмент присваивания, стэйтмент вывода.

Структура программы на языке программирования С++.

#include "stdafx.h"// подключение всех необходимых препроцессорных директив #include <iostream> //заголовочный файл с классами, функциями и переменными для организации ввода-вывода using namespace std; // подключение стандартного пространства имён std int main() // начало главной функции с именем main (точка входа в программу) { //тело главной функции, в котором содержится программный код }

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 избегания конфликтов имен.

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) Текст, содержащийся между символами /\* и \*/, также является комментарием, даже если он занимает несколько строк.

Операторы консольного ввода-вывода данных.

Для записи простых тестовых данных на консоль в C++ используется оператор std::cout (произносится как standard see-out - «стандарт си-аут») и оператор std::cin (произносится как standard see-in - «стандарт си-ин») для чтения текста и чисел с консоли (как правило, с клавиатуры). В C# используются операторы Console.WriteLine() для записи и Console.ReadLine() для чтения.

Структура программы на языке программирования С#.

using System; // подключаемое пространство имен namespace HelloApp // объявление нового пространства имен { class Program // объявление нового класса { static void Main()// начало главной функции с именем main { //тело главной функции, в котором содержится программный код } } // конец объявления нового класса } // конец объявления нового пространства имен

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

Переменная - это средство, позволяющее программисту временно сохранить данные. Переменная представляет именованную область памяти, в которой хранится значение определенного типа. Переменные в C# и C++ объявляются по следующему шаблону: ТИП\_ПЕРЕМЕННОЙ ИМЯ\_ПЕРЕМЕННОЙ = ИСХОДНОЕ\_ЗНАЧЕНИЕ; ТИП\_ПЕРЕМЕННОЙ ИМЯ\_ПЕРЕМЕННОЙ(ИСХОДНОЕ\_ЗНАЧЕНИЕ); Атрибут «ТИП\_ПЕРЕМЕННОЙ» указывает компилятору характер данных, который может хранить переменная, и компилятор резервирует для этого необходимое пространство. Атрибут «ИМЯ\_ПЕРЕМЕННОЙ» является более осмысленной заменой для адреса области в памяти, где хранится значение переменной.

Что такое константа в языках программирования?

Константа - это средство, позволяющее программисту определить элемент, которому не позволено изменяться.

Что такое литерал? Как классифицируются литералы?

Литеральные константы (или просто «литералы») — это значения, которые вставляются непосредственно в код. Поскольку они являются константами, то их значения изменить нельзя. Литералы бывают логическими, целочисленными, вещественными, символьными и строчными. И отдельный литерал представляет ключевое слово null.

Какая операция применяется для изменения значения переменной?

Для инициализации переменной исходным значением применяется копирующая инициализация, выполняемая с помощью знака равенства =, или прямая инициализация с помощью круглых скобок ().

Перечислите правила именования переменных.

Имена же переменных задаются согласно следующим правилам: 1) Имя переменной не может начинаться с цифры. 2) В имени переменной не может быть пробелов и специальных символов. 3) Имя переменной не может совпадать с другими, ранее объявленными именами (функций, переменных, стандартных операторов и т. п.). 4) Именами переменных не могут быть также зарезервированные ключевые слова (return).

Что такое область видимости переменной? В чем отличие глобальной области видимости от локальной?

Область видимости - это четко определенная область у обычных переменных, в пределах которой они допустимы и применимы. Программа не будет откомпилирована если переменные находятся вне своих областей видимости. Глобальные переменные - это переменные, которые были созданы вне тела какого-то блока. Они имеют самую широкую область видимости, так как их можно всегда использовать во всей программе, а локальные переменные — это переменные, которые применимы в функции от места ее объявления до конца функции. Когда функция заканчивается, все ее локальные переменные ликвидируются, а занимаемая ей память освобождается.

В чем отличие l-value от r-value?

l-value — это значение, которое имеет свой собственный адрес в памяти. Поскольку все переменные имеют адреса, то они все являются l-values. r-value — это значение, которое не имеет постоянного адреса в памяти (единичные числа или выражения).

Что такое тип данных?

Тип данных определяет множество значений, набор операций, которые можно применять к таким значениям и, возможно, способ реализации хранения значений и выполнения операций.

Как классифицируются типы данных?

Типы данных классифицируются на: 1) Логический тип данных (bool); 2) Символьный тип данных (char); 3) Целочисленные типы данных (short int, unsigned short int, int, unsigned int, long int, unsigned long int); 4) типы данных с плавающей точкой (float, long float, double); 5) Дополнительные типы данных (string, object).

Дайте характеристику типу данных bool.

Язык C++ предоставляет тип, специально созданный для хранения логических значений true или false, оба из которых являются зарезервированными ключевыми словами C++. Типичное объявление инициализированной логической переменной имеет следующий вид: bool AlwaysOnTop = false;.

Дайте характеристику типу данных int.

Целочисленные типы данных используются для представления чисел. Их шесть штук: short int, unsigned short int, int, unsigned int, long int, unsigned long int. Все они имеют свой собственный размер занимаемой памяти и диапазон принимаемых значений. Приставки целочисленных типов данных: 1) short — приставка укорачивает тип данных, к которому применяется, путём уменьшения размера занимаемой памяти; 2) long — приставка удлиняет тип данных, к которому применяется, путём увеличения размера занимаемой памяти; 3) unsigned (без знака) — приставка увеличивает диапазон положительных значений в два раза, при этом диапазон отрицательных значений в таком типе данных храниться не может.

Дайте характеристику типу данных char.

Тип char используется для хранения одного символа. Типичное объявление: char Userlnput = 'Y'; // инициализированный символ 'Y' Числовое представление латинских символов, чисел, некоторых специальных клавиш и символов было стандартизировано в стандартный американский код обмена информацией (ASCII).

Дайте характеристику типу данных double/float.

Типы данных с плавающей точкой отлично подходят для работы с дробными числами. Переменная такого типа может хранить любые действительные дробные значения. Есть два типа данных с плавающей точкой: · float; · double. Если нужно использовать целое число с переменной типа данных с плавающей точкой, то тогда после этого числа нужно поставить разделительную точку и нуль.

Как определяется размер переменной?

Чтобы определить размер целого числа, вызывать оператор sizeof с параметром int (тип) или указать наименование переменной, для которой необходимо определить размер. cout << "Size of an int: " << sizeof(int);

Какой инструмент языков программирования позволяет не указывать явно тип данных переменной? Приведите пример.

В языке С++ есть возможность определять тип данных неявно, с использованием вместо него ключевого слова auto (например: auto Flag = true;)

Какие арифметические операции реализованы в языках программирования?

Арифметические операции производятся над числами. Значения, которые участвуют в операции, называются операндами. Арифметические операции бывают бинарными (производятся над двумя операндами, например: оператор сложения, вычитания, умножения, деления, получения остатка) и унарными (выполняются над одним операндом, например: префиксный и постфиксный инкремент, декремент).

В чем отличие постфиксного декремента от префиксного декремента?

Префиксный декремент - уменьшает значение переменной на единицу, и полученное значение используется как значение выражения -x, а постфиксный декремент тоже уменьшает значение переменной на единицу, но значением выражения x-- будет то, которое было до уменьшения на единицу.

Какие операторы сравнения и присваивания реализованы в языках программирования? Опишите их.

Существует пять операций присваивания, не считая основную операцию присваивания: =. 1) += операция присваивания-сложения; 2) -= операция присваивания-вычитания; 3) \*= операция присваивания-умножения; 4) /= операция присваивания-деления; 5) %= операция присваивания-остатка от деления. Существует 6 различных операторов сравнения: 1) Больше > ; 2) Меньше < ; 3) Больше или равно >= ; 4) Меньше или равно <= ; 5) Равно == ; 6) Не равно != .

Какие логические операторы реализованы в языках программирования? Опишите их.

Существует три логические операции: 1. Логическая операция И &&; 2. Логическая операция ИЛИ ||; 3. Логическая операция НЕ ! или логическое отрицание.

Какие методы используются для определения типа данных переменной?

Для определения типа переменной в C++ существует функция typeid(ИМЯ\_ПЕРЕМЕННОЙ).name() из библиотеки typeinfo. В C# есть похожая функция: ИМЯ\_ПЕРЕМЕННОЙ.GetType().

Какие существуют способы для неявного преобразования типов?

Есть 2 основных способа неявного преобразования типов: · числовое расширение; · числовая конверсия. Когда значение из одного типа данных конвертируется в другой тип данных побольше (по размеру и по диапазону значений), то это называется числовым расширением. Например, тип int может быть расширен в тип long, а тип float может быть расширен в тип double: long l(65); // расширяем значение типа int (65) в тип long Когда мы конвертируем значение из более крупного типа данных в аналогичный, но более мелкий тип данных, или конвертация происходит между разными типами данных, то это называется числовой конверсией. Например: double d = 4; // конвертируем 4 (тип int) в double

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.

Чем числовая конверсия отличается от числового расширения?

В отличие от расширений, которые всегда безопасны, конверсии могут привести к потере данных. Поэтому в любой программе, где выполняется неявная конверсия, компилятор будет выдавать предупреждение.

Как осуществляется явное преобразование типов в С++?

В языке C++ есть 5 видов операций явного преобразования типов: · конвертация C-style; · применение оператора static\_cast; · применение оператора const\_cast; · применение оператора dynamic\_cast; · применение оператора reinterpret\_cast. В программировании на языке Cи явное преобразование типов данных выполняется с помощью оператора (). Внутри круглых скобок мы пишем тип, в который нужно конвертировать. Этот способ конвертации типов называется конвертацией C-style.

Как осуществляется явное преобразование типов в С#?

Тем не менее если преобразование нельзя выполнить без риска потери данных, компилятор требует выполнения явного преобразования, которое называется приведением. Приведение — это способ явно указать компилятору, что необходимо выполнить преобразование и что вам известно, что может произойти потеря данных или приведение может завершиться сбоем во время выполнения. Чтобы выполнить приведение, укажите тип, в который производится приведение, в круглых скобках перед преобразуемым значением или переменной. В следующей программе выполняется приведение типа double в int. Программа не будет компилироваться без приведения. class Test { static void Main() { double x = 1234.7; int a; // Cast double to int. a = (int)x; System.Console.WriteLine(a); } } // Output: 1234

Какие операторы используются для реализации ветвления процесса выполнения программы?

Чтобы позволить приложению реагировать по-другому, в зависимости от ситуации или введенных пользователем данных, необходимы условные операторы, выполняющие разные сегменты кода в разных ситуациях. Условное выполнение кода реализовано на базе конструкции оператора if...else, а также благодаря операторам множественного выбора и условному тернарному оператору.

57. Опишите синтаксис условного оператора. Приведите пример его использования.

Условное выполнение кода реализовано на базе конструкции if...else, синтаксис которой выглядит следующим образом: if (условное выражение) { //Сделать нечто, когда условное выражение возвращает true; } else { //Сделать нечто другое, когда условное выражение возвращает false; } Часть else конструкции if...else является необязательной.

Опишите синтаксис условного тернарного оператора. Приведите пример его использования.

Условный (тернарный) оператор (обозначается как ?:) является единственным тернарным оператором, который работает с 3-мя операндами. Он предоставляет сокращенный способ (альтернативу) ветвления if/else. Синтаксис условного тернарного оператора имеет следующий вид: (логическое условное выражение) ? выражение1 при true : выражение2 при false; Такой оператор применим при компактном сравнении двух чисел, как показано ниже. int Мах = (Numl > Num2) ? Numl : Num2; // Max содержит большее число из Numl и Num2

Опишите синтаксис оператора множественного выбора. Приведите пример его использования.

Конструкция switch-case имеет следующий синтаксис: switch (/\*переменная или выражение\*/) { case /\*константное выражение 1\*/: { /\*группа операторов 1\*/; break; } case /\*константное выражение 2\*/: { /\*группа операторов 2\*/; break; } //... default: { /\*группа операторов 3\*/; } }

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. Например: #include <iostream> #include <cmath>// для функции sqrt() using namespace std; int main() { double z; tryAgain: // это лейбл cout << "Enter a non-negative number: "; cin >> z; if (z < 0.0) goto tryAgain; // а это оператор goto cout << "The sqrt of " << z << " is " << sqrt(z) << endl; return 0; }

Какие существуют ограничения при использовании оператора goto?

Нельзя перепрыгнуть вперед через переменную, которая инициализирована в том же блоке, что и goto. Основная проблема с оператором goto заключается в том, что он позволяет программисту управлять выполнением кода так, что точка выполнения может прыгать по коду произвольно. А это, в свою очередь, создает то, что опытные программисты называют «спагетти-кодом».

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

Операторы for, while и do while используются для реализации многократного исполнения последовательности инструкций в коде программы. Они являются циклическими алгоритмами, которые в зависимости от исходных данных могут повторяться многократно.

В чем отличие циклов с предусловием от цикла с постусловием?

Цикл с предусловием (цикл while) - цикл, который выполняется, пока истинно некоторое условие, указанное перед его началом. Это условие проверяется до выполнения тела цикла, поэтому последнее может быть ни разу не выполнено. Цикл с постусловием (цикл do-while) - цикл, в котором условие проверяется после выполнения тела цикла. Отсюда следует, что тело данного цикла всегда выполняется хотя бы один раз.

Опишите синтаксис цикла с предусловием. Приведите пример его использования.

Цикл while объявляется с использованием ключевого слова while. В начале цикла обрабатывается условие. Если его значением является true (любое ненулевое значение), то тогда выполняется тело цикла. while (<условие >) { <тело цикла>; } Пример: программа выводит все числа от 0 до 9. #include <iostream> using namespace std; int main() { int count = 0; while (count < 10) { cout << count << " "; ++count; } return 0; }

Опишите синтаксис цикла с постусловием. Приведите пример его использования.

Синтаксис цикла с постусловием имеет следующий вид: do { < тело цикла>; } while (<условие >); Пример: Программа выводит число 1 #include <iostream> using namespace std; int main() int a = 1; do{ cout << a << " "; ++a; } while (a < 2); return 0; }

Опишите синтаксис цикла со счетчиком. Приведите пример его использования.

Цикл со счетчиком (цикл for) - это циклический алгоритм, в котором тело цикла выполняется заранее известное число раз. Синтаксис: for(<объявление переменных>;<условие>;<инкремент/декремент счетчика>) { <тело цикла>; } Пример: программа выводит числа с 0 до 9 #include <iostream> using namespace std; int main() { for(int count = 0; count < 10; count++) { cout << count << " "; } return 0; }

Как объявляется бесконечный цикл?

Бесконечный цикл - это цикл, написанный таким образом, что он никогда не прекратит своё выполнение, так как никогда не выполнится условие выхода из этого цикла. Например: for (;;){ <тело цикла>; }

Для чего используется оператор break и continue?

Когда оператор break выполняется в цикле, то досрочно прерывается исполнение оператора цикла, и управление передаётся следующему оператору после цикла. В теле оператора множественного выбора switch оператор break прерывал исполнение оператора switch. Оператор continue используется только в циклах. В операторах for, while, do while, оператор continue выполняет пропуск оставшейся части кода тела цикла и переходит к следующей итерации цикла.

Что такое указатель?

Указатель - это переменная, значением которого является адрес области в памяти.

Как объявляется указатель в коде программы?

Указатель объявляется, как и все остальные переменные: ТипУказателя \*ИмяПеременнойУказателя;

Какой оператор используется для записи или чтения данных, содержащихся в области памяти, адрес которой хранит указатель?

Для того, чтобы записать или прочитать данные, содержащиеся в области памяти, адрес которой хранит указатель, используется оператор обращения к значению (оператор разыменования) (\*).

В каких арифметических операциях могут принимать участие указатели?

Над указателями можно выполнять следующие арифметические операции: операции инкремента и декремента; операции сложения и вычитания. Кроме того, можно вычитать из одного указателя другой указатель. К указателю можно прибавлять целое число, и также можно вычитать из указателя целое число.

Что происходит с указателем при увеличении его значения с помощью операции инкремента?

В случае с указателем, увеличение на единицу будет означать увеличение адреса, который хранится в указателе, на размер типа указателя.

Как вычисляется «расстояние» между указателями?

«Расстояние» между указателями является результатом разности двух указателей.

Как определяется приоритет операцией при одновременном использовании операции разыменования и постфиксного инкремента? Приведите пример.

В выражении \*pa++ сначала к указателю присваивается единица. Затем так как инкремент постфиксный, с помощью операции разыменования возвращается значение, которое было до инкремента.

Перечислите случаи использования указателей в программировании.

Указатели используются для передачи по ссылке данных, что намного ускоряет процесс обработки этих данных, так как их не надо копировать, как при передаче по значению. Также указатели используются для организации динамического распределения памяти, например, при объявлении массива, не надо будет его ограничивать в размере.

Что такое динамическое распределение памяти?

Динамическое распределение памяти - это резервирование большего объёма памяти при необходимости и освобождение её, когда необходимости в ней больше нет.

В каких случаях возникает утечка памяти?

Утечка памяти возникает, если не освободить зарезервированную память после прекращения ее использования.

Перечислите комбинации использования ключевого слова const с указателями.

1) Данные, на которые указывает указатель, являются постоянными и не могут быть изменены, но сам адрес, содержащийся в указателе, вполне может быть изменен. 2) Содержащийся в указателе адрес является постоянным и не может быть изменен, однако данные, на которые он указывает, вполне могут быть изменены. 3) Третья комбинация. И содержащийся в указателе адрес, и значение, на которое он указывает, являются константами и не могут быть изменены.

Для чего используются указатели на указатели?

Указатели на указатели имеют несколько применений, но наиболее используемым является динамическое выделение массива указателей: int \*\*array = new int\*[20] // выделяем массив из 20 указателей типа int

Как происходит создание двумерного динамического массива?

Динамический двумерный массив — это динамический одномерный массив динамических одномерных массивов. Пример создания двумерного динамического массива 3 на 4: int \*\*array = new int\*[3] // выделяем массив из 15 указателей типа int — это наши строки for (int count = 0; count < 15; ++count){ array[count] = new int[4] // а это наши столбцы }

Что такое ссылочная переменная? В чем ее отличие от указателей?

Ссылка — это псевдоним переменной. При объявлении ссылки ее необходимо инициализировать переменной. Таким образом, ссылочная переменная — это только другой способ доступа к данным, хранимым в переменной. Отличие указателей от ссылочной переменной заключается только в том, что перед именем ставится символ звёздочки \*.

Определение функции.

Функция — это подпрограмма, которая может получать параметры на вход и возвращать выходное значение. Чтобы выполнить свою задачу, функция должна быть вызвана.

Опишите общую форму определения функции на языке С++ и С#.

Все функции С++ имеют общую форму, приведенную ниже: ТИП\_ВОЗВРАТА ИМЯ\_ФУНКЦИИ(СПИСОК\_ПАРАМЕТРОВ) { //тело функции } Все функции (методы) С# имеют общую форму, приведенную ниже: [модификаторы] ТИП\_ВОЗВРАТА ИМЯ\_МЕТОДА (СПИСОК\_ПАРАМЕТРОВ) { //тело метода }

Как происходит вызов функции?

Когда функция объявлена вместе со списком параметров, при ее вызове нужно указать имя вызываемой функции и передать аргументы, являющимися значениями, затребованными функцией в списке ее параметров.

Чем отличается передача параметров по значению от передачи параметров по адресу?

Когда аргумент передается по значению, то его значение копируется в параметр функции. Когда вы передаете указатель в функцию по адресу, то значение этого указателя (адрес, на который он указывает) копируется из аргумента в параметр функции.

В С++ функцию нельзя вызвать до объявления самой функции. Какими способами устраняется данная проблема?

При помощи прототипа функции. Прототип функции — это функция, в которой отсутствует блок кода (тело функции). В прототипе функции находятся: · Полное имя функции. · Тип возвращаемого значения функции.

С помощью какого оператора реализуется возврат функцией определенного значения?

Возврат по значению осуществляется при помощи оператора return.

89. Как создаются функции с параметрами по умолчанию? Как происходит вызов функций с параметрами по умолчанию?

Параметр по умолчанию — это необязательный параметр функции, который имеет определенное (по умолчанию) значение. Если пользователь не передает в функцию значение для параметра, то используется значение по умолчанию. Пример: double Area(double InputRadius, double Pi = 3.14)//pi - необязательный параметр { //тело функции } Int main(){ Area(Radius, Pi); // Pi определяется пользователем, либо отсутствует }

Как происходит перегрузка функций? Приведите пример перегрузки функций.

Функции с одинаковым именем и типом возвращаемого значения, но с разными параметрами или набором параметров называют перегруженными функциями. Пример: double Area(double Radius)//для круга { return Pi \* Radius \* Radius; } double Area(double Radius, double Height)//для цилиндра { return 2 \* Area(Radius) + 2 \* Pi \* Radius \* Height; } Int main(){ Area(Radius); Area(double Radius, double Height); }

Перечислите преимущества и недостатки передачи аргументов по значению.

Преимущества передачи аргументов по значению: · Аргументы, переданные по значению, могут быть переменными, литералами, выражениями, структурами, классами или перечислителями (т.е. почти всем, чем угодно). · Аргументы никогда не изменяются функцией, в которую передаются, что предотвращает возникновение побочных эффектов. Недостатком передачи аргументов по значению является то, что копирование структур и классов может привести к значительному снижению производительности (особенно, когда функция вызывается много раз).

Перечислите преимущества и недостатки передачи аргументов по адресу:

1) Передача по адресу позволяет функции изменить значение аргумента. 2) Поскольку копирования аргументов не происходит, то скорость передачи по адресу достаточно высокая, даже если передавать большие структуры или классы. 3) Мы можем вернуть сразу несколько значений из функции, используя параметры вывода. Минусы передачи по адресу: 1) Все указатели нужно проверять, не являются ли они нулевыми. Попытка разыменовать нулевой указатель приведет к сбою в программе. 2) Поскольку разыменование указателя выполняется медленнее, чем доступ к значению напрямую, то доступ к аргументам, переданным по адресу, выполняется также медленнее, чем доступ к аргументам, переданным по значению.

Когда следует и когда не следует использовать возврат по значению в функции?

Когда использовать возврат по значению: · при возврате переменных, которые были объявлены внутри функции; · при возврате аргументов функции, которые были переданы в функцию по значению. Когда не использовать возврат по значению: · при возврате стандартных массивов или указателей; · при возврате больших структур или классов.

Когда следует и когда не следует использовать возврат по адресу в функции?

Когда использовать возврат по адресу: · при возврате динамически выделенной памяти; · при возврате аргументов функции, которые были переданы по адресу. Когда не использовать возврат по адресу: · при возврате переменных, которые были объявлены внутри функции; · при возврате большой структуры или класса, который был передан по ссылке.

Когда следует и когда не следует использовать возврат по ссылке в функции?

Когда использовать возврат по ссылке: · при возврате ссылки-параметра; · при возврате элемента массива, который был передан в функцию; · при возврате большой структуры или класса, который не уничтожается в конце функции. Когда не использовать возврат по ссылке: · при возврате переменных, которые были объявлены внутри функции; · при возврате стандартного массива или значения указателя.

Что такое указатель на функцию? Как объявляется указатель на функцию?

Указатель на функцию хранит адрес функции. По сути указатель на функцию содержит адрес первого байта в памяти, по которому располагается выполняемый код функции. Объявление указателя на функцию: тип (\*имя\_указателя) (параметры)

Как происходит передача указателя на функцию в другую функцию и возврат функцией указателя на другую функцию? Приведите пример.

Указатель на функцию может передаваться в другую функцию в качестве параметра. Например: int operation(int(\*op)(int, int), int a, int b). Функция может возвращать указатель на другую функцию, если имеется ограниченное количество вариантов - выполняемых функций, и надо выбрать одну из них. Но при этом набор вариантов и выбор из них определяется в промежуточной функции. Например: void(\*message(int hour))().

Что такое рекурсия? Что такое базовый случай? Приведите пример рекурсивной функции.

Рекурсия — это перебор. Функция называется рекурсивной, когда может фактически вызывать сама себя. Базовый случай - это условие остановки, без которой функция будет бесконечной. Пример: Функция вычисления факториала int factorial(int N) { if (N == 1) return 1; return N \* factorial(N - 1); }

Что представляет собой делегат? Укажите синтаксис объявления делегата.

Делегаты - это указатели на методы и с помощью делегатов программист может вызвать данные методы. Для объявления делегата используется ключевое слово delegate, после которого идет возвращаемый тип, название и параметры. Например: delegate void Message();

Как происходит добавление методов в делегат и удаление методов из делегата?

Все методы в делегате попадают в список вызова или invocation list. При вызове делегата все методы из этого списка последовательно вызываются. Для добавления метода в делегат применяется операция +=. Для удаления метода из делегата применяется операция -=. Пример: MyCalculation calc = null; // создаем объект делегата calc += CalculateSum; // добавляем метод calc -= CalculateSum; // удаляем метод calc(); //так вызываются методы

В чем заключается принцип объектно-ориентированного программирования?

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

Перечислите основные принципы объектно-ориентированного программирования.

Для моделирования объектов необходим принцип абстрагирования, т.е. выделение наиболее существенных для решаемой задачи черт исходного объекта и отбрасывание второстепенных деталей. В ООП существует три основных принципа построения классов: 1. Инкапсуляция — это свойство, позволяющее объединить в классе и данные, и методы, работающие с ними и скрыть детали реализации от пользователя. 2. Наследование — это свойство, позволяющее создать новый класс-потомок на основе уже существующего, при этом все характеристики класса родителя присваиваются классу-потомку. 3. Полиморфизм — свойство классов, позволяющее использовать объекты классов с одинаковым интерфейсом без информации о типе и внутренней структуре объекта.

Что такое класс?

Класс представляет собой формализованный способ описания однотипных объектов, т.е. объектов с одинаковым набором свойств и методов.

Можете ли вы вызвать метод базового класса, не создавая экземпляр?

Мы можем вызвать метод класса без экземпляра, объявив его статическим. Модификатор static используется для объявления статического члена, принадлежащего собственно типу, а не конкретному объекту. class Program { static void Main(string[] args) {} }

В чем разница между классом и объектом?

Класс - это некий шаблон, который предоставляет абстрактный вид любого объекта, а объект или экземпляр класса - это уже конкретная сущность, которая сделана по шаблону класса.

Что такое модификатор доступа? Перечислите модификаторы доступа, указав их функциональность.

Модификаторы доступа — это ключевые слова, которые задают объявленный уровень доступности члена или типа. Модификаторы доступа позволяют задать допустимую область видимости для членов класса. В C# применяются следующие модификаторы доступа: · public: публичный, общедоступный класс или член класса. Такой член класса доступен из любого места в коде, а также из других программ и сборок. · private: закрытый класс или член класса. Представляет полную противоположность модификатору public. Такой закрытый класс или член класса доступен только из кода в том же классе или контексте. · protected: такой член класса доступен из любого места в текущем классе или в производных классах. При этом производные классы могут располагаться в других сборках. · internal: класс и члены класса с подобным модификатором доступны из любого места кода в той же сборке, однако он недоступен для других программ и сборок. · protected internal: совмещает функционал двух модификаторов. Классы и члены класса с таким модификатором доступны из текущей сборки и

В чем преимущество использования объектно-ориентированной парадигмы программирования?

Основное преимущество ООП заключается в том, что и данные, и операции (код), используемые для манипулирования ими, инкапсулируются в одном объекте. Достоинством объектных программ является их четкая структуризация, а также специфичное для компонента поведение, разделение задач, повторное использование кода.

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?

Инкапсуляция позволяет рассматривать объект в виде самостоятельной программной единицы, отвечающей за хранение и обработку своих данных и предоставляющей посторонним пользователям четко определенный набор услуг.

Как называется элемент данных объекта? (Свойство объекта)

Каждый элемент данных часто называют свойством объекта (хотя есть и различия в трактовке этого термина) Набор данных-свойств определяется при описании объекта и при выполнении программы изменяться не может, могут изменяться лишь значения свойств. Текущие значения свойств определяют текущее состояние объекта. Их рекомендуется делать недоступными для прямого использования за пределами объекта (закрытость данных)

Какие типы могут иметь свойства класса? (Простейшие, структурные, объектные)

Свойства могут иметь разные типы: простейшие (целочисленные, символьные, логические), структурные (строки, массивы, списки) и даже объектные. Методы могут иметь ограничения по доступности: некоторые методы можно сделать недоступными (закрытыми) за пределами объекта, но всегда должен быть определен набор открытых методов, образующих внешний интерфейс объекта.

Как называется логически связанный блок кода, связанный с обработкой данных внутри класса?

Метод класса - именованный блок выполняемого кода, который может быть вызван на выполнение из разных частей программы. При вызове метода он выполняет свой код, а затем возвращает управление тому коду, который его вызвал. Методы описывают поведение объектов класса.

Дайте определение метода-конструктора.

Конструктор отвечает за динамическое создание нового объекта при выполнении программы, которое включает в себя: -выделение памяти, необходимой для хранения значений свойств создаваемого объекта (совместно с операционной системой и средой поддержки выполнения программ) -занесение в выделенную область начальных значений свойств создаваемого объекта

Дайте определение метода-деструктора.

Деструктор отвечает за уничтожение объекта, т.е. освобождение памяти, выделенной объекту. В отличие от конструкторов, механизм деструкторов реализован не во всех языках: деструкторы есть в языке С++, в языке Delphi Pascal, предусмотрены (но практически не используются) в языке С# и отсутствуют в языке Java. В последних двух языках вместо деструкторов реализован специальный механизм «сборки мусора» (garbage collect).

Какие методы организуют контролируемый доступ к закрытым элементам данных объекта?

Это методы доступа к закрытым свойствам объекта. Введение таких методов позволяет организовать контролируемый доступ к внутренним данным объекта. В общем случае для закрытого свойства можно ввести два метода доступа: -метод для чтения хранящегося в свойстве значения (часто такие методы называют get-методами) -метод для изменения значения свойства (set-метод)

Перечислите этапы жизненного цикла объекта.

Каждый объект проходит определенный жизненный цикл: 1. объект создается методом-конструктором 2. объект используется другими объектами, предоставляя им свои открытые методы и неявно - закрытые данные 3. объект уничтожается (явно деструктором или неявно механизмом сборки мусора)

Что включает описание класса?

Описание класса включает в себя: -заголовок класса, включающий специальную директиву class и имя класса; практически всегда в заголовке класса задается дополнительная информация о классе; -тело класса, содержащее перечень свойств (полей данных), заголовки методов и их программную реализацию (не всегда).

Перечислите общие правила для описания свойств класса.

Общие правила описания свойств: -каждое свойство объявляется как обычная переменная, т.е. для нее обязательно задается имя и тип (простейший, структурный или объектный) -имена всех свойств в классе должны быть различными -свойства рекомендуется объявлять закрытыми с помощью специальной директивы private (такие свойства доступны для прямого использования только внутри методов данного класса); открытые свойства (если необходимо нарушить принцип инкапсуляции) объявляются с помощью директивы public.

В каких случаях допускается перегрузка конструкторов класса?

В классе допустимо создавать несколько конструкторов, если это необходимо. Имена, согласно пункту 2 нашего списка, будут одинаковыми. Компилятор будет их различать по передаваемым параметрам (как при перегрузке функций). Если мы не передаем в конструктор параметры, он считается конструктором по умолчанию;

В виде какой функции оформляется Set-метод?

Методы доступа для изменения значений закрытых свойств (Set методы) рекомендуется именовать с префиксом Set (например, SetColor, SetSize) и оформлять как процедуру (void-функцию) с одним входным параметром.

В виде какой функции оформляется Get-метод?

Методы доступа для чтения значений закрытых свойств (Get методы) рекомендуется именовать с префиксом Get (например, GetColor, GetSize) и оформлять как функцию без параметров, возвращающую тип соответствующего свойства.

Укажите синтаксические правила для определения класса на языке С#. Приведите пример.

По умолчанию проект консольного приложения уже содержит один класс Program, с которого и начинается выполнение программы. По сути класс представляет новый тип, который определяется пользователем. Класс определяется с помощью ключевого слова сlass. class имя\_класса { уровень\_доступа тип переменная1; уровень\_доступа тип переменная2; уровень\_доступа тип\_возвращаемого\_значени метод1(параметры) { //тело метода } уровень\_доступа тип\_возвращаемого\_значени метод2(параметры) { //тело метода } }

Как создается экземпляр класса?

Точно так же, как определение переменной фундаментального типа данных (например, int x) приводит к выделению памяти для этой переменной, так же и создание объекта класса (например, Person person) приводит к выделению памяти для этого объекта. Класс можно представлять себе, как нечто вроде типа данных более высокого уровня. class Person { public: string name; int age; }; int main() { Person person; person.name = "Tom"; person.age = 22; return 0; }

Как происходит обращение к свойствам и методам экземпляра класса?

К членам (переменным и функциям) класса доступ осуществляется через оператор выбора членов (.). Например person.name = "Tom"; и person.move(); На объекты классов, как и на объекты других типов, можно определять указатели. Затем через указатель можно обращаться к членам класса - переменным и методам. Однако если при обращении через обычную переменную используется символ точка (.), то для обращения к членам класса через указатель применяется стрелка (->):

Какой уровень доступа к элементу данных класса определяет модификатор public?

Модификатор public является спецификатором доступа (access specifier), то есть определяет параметры доступа к членам класса - переменным и функциям. В частности, он делает их доступными из любой части программы. По сути спецификатор public определяет общедоступный интерфейс класса.

Какой уровень доступа к элементу данных класса определяет модификатор private?

Закрытые члены классы, объявленные с помощью модификатора private, могут быть прочитаны и изменены только классом, к которому принадлежат. Уровень доступа может быть изменен при помощи соответствующих ключевых слов.

Какой уровень доступа к элементу данных класса определяет модификатор protected?

Модификатор protected открывает доступ классам, производным от данного. То есть, производные классы получают свободный доступ к таким свойствам или методам. Все другие классы такого доступа не имеют.

Что определяет интерфейс класса?

Открытые члены классов составляют интерфейс класса. Поскольку доступ к открытым членам класса может осуществляться извне класса, то открытый интерфейс и определяет, как программы, использующие класс, будут взаимодействовать с этим же классом. Интерфейс — то, что доступно при использовании класса извне. Через него происходит взаимодействие с пользователем.

Какой модификатор доступа имеет член класса, если он явно не указан?

Если модификатор доступа явно необъявлен, то класс по умолчанию устанавливает всем своим членам спецификатор доступа private. Структура же по умолчанию устанавливает всем своим членам спецификатор доступа public.

Чем поле отличается от свойства в языке С#?

Свойства предоставляют поля. Поля должны (почти всегда) оставаться закрытыми для класса и доступны через get и set свойства. Свойства обеспечивают уровень абстракции, позволяющий изменять поля, не влияя при этом на внешний способ доступа к ним объектов, использующих ваш класс. Важным отличием является то, что интерфейсы могут иметь свойства, но не поля.

Укажите стандартный синтаксис для определения свойства и автосвойства в языке С#?

Стандартное описание свойства имеет следующий синтаксис: [модификатор\_доступа] возвращаемый\_тип произвольное\_название { // код свойства } Если у нас с десяток и более полей, то определять каждое поле и писать для него однотипное свойство было бы утомительно. Поэтому были добавлены автоматические свойства. Они имеют сокращенное объявление. Автосвойствам можно присвоить значения по умолчанию (инициализация автосвойств): class Person { public string Name { get; set; } = "Tom"; public int Age { get; set; } = 23; }

При каких условиях можно установить модификатор доступа для блока set или get?

-Модификатор для блока set или get можно установить, если свойство имеет оба блока (и set, и get); -Только один блок set или get может иметь модификатор доступа, но не оба сразу; -Модификатор доступа блока set или get должен быть более ограничивающим, чем модификатор доступа свойства. Например, если свойство имеет модификатор public, то блок set/get может иметь только модификаторы internal, protected, private.

Какой может иметь модификатор доступа для блока set или get, если свойство имеет модификатор public?

Если свойство имеет модификатор public, то блок set/get может иметь только модификаторы internal, protected, private.

Для чего используются конструкторы класса?

Конструкторы обычно используются для инициализации переменных-членов класса значениями, которые предоставлены по умолчанию/пользователем, или для выполнения любых шагов настройки, необходимых для используемого класса (например, открыть определенный файл или базу данных).

Укажите правила описания методов-конструкторов

По сути конструктор представляет функцию, которая может принимать параметры, и которая должна называться по имени класса. В отличие от обычных методов, конструкторы имеют определенные правила по поводу их имен: -конструкторы всегда должны иметь то же имя, что и класс (учитываются верхний и нижний регистры); -конструкторы не имеют типа возврата (даже void). Person(string n, int a) { name = n; age = a; }

Что такое конструктор по умолчанию?

Конструктор по умолчанию - конструктор, который не имеет параметров (или имеет параметры, все из которых имеют значения по умолчанию). Он вызывается, если пользователем не указаны значения для инициализации.

В каких ситуациях происходит удаление объекта с использованием деструктора?

Чаще всего деструктор используют тогда, когда в конструкторе, при создании объекта класса, динамически был выделен участок памяти и необходимо эту память очистить, если эти значения уже не нужны для дальнейшей работы программы.

Укажите правила описания методов-деструкторов.

1) конструктор и деструктор, мы всегда объявляем в разделе public; 2) при объявлении конструктора, тип данных возвращаемого значения не указывается; 3) у деструктора также нет типа данных для возвращаемого значения, к тому же деструктору нельзя передавать никаких параметров; 4) имя класса и конструктора должно быть идентично; 5) имя деструктора идентично имени конструктора, но с приставкой ~ ; 6) в классе допустимо создавать несколько конструкторов, если это необходимо. Имена, согласно пункту 2 нашего списка, будут одинаковыми. Компилятор будет их различать по передаваемым параметрам (как при перегрузке функций). Если мы не передаем в конструктор параметры, он считается конструктором по умолчанию; 7) обратите внимание на то, что в классе может быть объявлен только один деструктор.

В чем отличие значимых типов и ссылочных типов в С#?

Значимые типы хранят значение, а ссылочные - ссылку на значение. Главное отличие - это где храниться значение переменной. Для ссылочных типов - это куча, для значимых - это стек. При этом, ссылочная переменная (сама ссылка) храниться на стеке. Значимыми могут быть только экземпляры структур, в противоположность им ссылочные - это экземпляры классов. Все встроенные типы (int, double и т.д.) - это структуры. Ссылочные - object, string, class, interface, delegate.

Как работает копирование значение при присвоении данных объекту значимого типа?

При присвоении данных объекту значимого типа он получает копию данных. Поэтому при изменении копии данных исходные данные не меняются.

Как работает копирование значение при присвоении данных объекту ссылочного типа?

При присвоении данных объекту ссылочного типа он получает не копию объекта, а ссылку на этот объект в хипе. Потому при изменении объекта ссылочного типа меняется и исходное значение.

Для чего используется ключевое слово ref?

При передаче параметра по ссылке (с помощью ключевого слова ref) в метод в качестве аргумента передается сама ссылка на объект в памяти. Поэтому можно изменить как поля и свойства объекта, так и сам объект.

В чем заключается назначение ключевого слова this?

Ключевое слово this представляет указатель на текущий объект данного класса. Соответственно через this мы можем обращаться внутри класса к любым его членам - к свойствам и методам. В пределах метода класса, когда вызывается другой метод, компилятор неявно передает ему в вызове указатель this как невидимый параметр.

Какой параметр неявно добавляется в метод класса на этапе компиляции?

При компиляции обычного метода, компилятор неявно добавляет к нему параметр \*this. Указатель \*this — это скрытый константный указатель, содержащий адрес объекта, который вызывает метод класса.

Что представляют собой статические члены класса? Как они определяются в коде программы?

Статические члены класса - переменные и методы, которые относятся непосредственно к классу. Статические переменные и методы относят в целом ко всем объектам классу. Для их определения используется ключевое слово static. Также стоит отметить, что так как статические члены относятся в целом ко всему классу, то для обращения к статическим членам используется имя класса, после которого идет оператор разрешения области видимости (::).

Какие существуют ограничения при определении статических функций (методов)?

В статических функциях(методах) нельзя использовать указатель this, так как this указывает на текущий объект, а статические функции относятся в целом ко всему классу.

Как можно осуществить обращение к статическим членам класса?

Так как статические члены относятся в целом ко всему классу, то для обращения к статическим членам используется имя класса, после которого идет оператор разрешения области видимости (::). Также можно обращаться к публичным членам класса через переменные данного класса.

Что представляет собой статический конструктор и статический класс?

Кроме обычных конструкторов у класса также могут быть статические конструкторы. Статические конструкторы обычно используются для инициализации статических данных, либо же выполняют действия, которые требуется выполнить только один раз

В C# можно объявлять статические классы. Они объявляются с модификатором static и могут содержать только статические поля, свойства и методы.

Какие операции используются для обращения к открытым свойствам и методам классов в языке программирования С++?

Через указатели можно обращаться к членам класса - переменным и методам. Однако, если при обращении через обычную переменную используется символ точка (.), то для обращения к членам класса через указатель применяется стрелка (->).

Чем отличается функциональность ключевого слова this в С++ и С#?

В C++ ключевое слово this представляет указатель на текущий объект данного класса. Соответственно через this можно обращаться внутри класса к любым его членам - к свойствам и методам.

В C# ключевое слово this представляет ссылку на текущий экземпляр класса. Использование ключевого слова this отличается одной особенностью - для обращения к членам класса после ключевого слова будет использоваться оператор выбора членов (.).

Какие переменные и методы следует определять, как статические?

Которые относятся непосредственно которые к классу. Для определения статических переменных и методов используется ключевое слово static.

Какие операторы разрешено перегружать в унарном и бинарном представлении?

Операции «+», «-», «\*», «&» допускается перегружать и как унарные, и как бинарные.

Какая операции не имеет фиксированного числа операндов при перегрузке?

Единственная операция, которая не имеет фиксированного количества операндов - это операция вызова функции ().

Опишите общую форму метода для перегрузки бинарной операции в языке программирования С#?

// Общая форма перегрузки бинарного оператора.

public static возвращаемый\_тип operator@(тип\_параметра1 операнд1, тип\_параметра2 операнд2)

{

/

}

Перечислите ссылочные типы в языке программирования С#?

Ссылочные типы:

1) Тип object

2) Тип string

3) Классы (class)

4) Интерфейсы (interface)

5) Делегаты (delegate)

Перечислите значимые типы языке программирования С++?

Значимые типы:

1. Целочисленные типы (byte, sbyte, short, ushort, int, uint, long, ulong)

2. Типы с плавающей запятой (float, double)

3. Тип decimal

4. Тип bool

5. Тип char

В чем заключается принцип абстрагирования, используемый при построении классов?

Наследование позволяет производному классу наследовать свойства и методы другого класса, что путём абстрагирования общих атрибутов и поведений даёт возможность создавать абсолютно новые классы.

Что такое массив?

Массив в программировании - это контейнер для хранения элементов одинакового типа, размещенных в последовательной памяти. Массивы используются для хранения структурированных данных.

Как происходит создание и инициализация массива на языке программирования С++?

// Способ 1: Объявление и инициализация массива одновременно

int my\_array[] = {1, 2, 3, 4, 5};

// Способ 2: Объявление массива и инициализация позже

int my\_array[5];

my\_array[0] = 1;

my\_array[1] = 2;

my\_array[2] = 3;

my\_array[3] = 4;

my\_array[4] = 5;

Как происходит создание и инициализация массива на языке программирования С#?

// Способ 1: Объявление и инициализация массива одновременно

int[] my\_array = { 1, 2, 3, 4, 5 };

// Способ 2: Объявление массива и инициализация позже

int[] my\_array = new int[5];

my\_array[0] = 1;

my\_array[1] = 2;

my\_array[2] = 3;

my\_array[3] = 4;

my\_array[4] = 5;

Как происходит доступ к данным, хранимым в массиве?

Доступ к данным, хранимым в массиве, осуществляется с помощью индексов. Индексы в массиве начинаются с нуля, и каждый элемент массива имеет свой уникальный индекс.

Как создаются функции с параметрами по умолчанию? Как происходит вызов функций с параметрами по умолчанию?

Функции с параметрами по умолчанию создаются путем указания значения по умолчанию для параметра в определении функции. Значение по умолчанию должно быть указано после имени параметра и знака равенства.

Вызов функций с параметрами по умолчанию осуществляется путем указания только тех параметров, для которых необходимо задать значение, отличное от значения по умолчанию. Необязательные параметры можно опустить, и для них будут использованы значения по умолчанию.

Перечислите общие правила для описания методов класса.

Методы класса должны быть объявлены внутри определения класса.

Методы класса могут иметь параметры и возвращать значения.

Методы класса могут быть определены как статические (статические методы) или нестатические (экземплярные методы).

Экземплярные методы имеют доступ к атрибутам и методам как класса, так и экземпляра объекта.