**Заяц Александр Владимирович:**

1. Опишите устройство памяти в **С#**.

**Ответ:** Память в c# разделяется на стек и кучу. В стеке хранятся данные с фиксированным временем жизни, такие как локальные переменные, аргументы методов, управляющие структуры. Куча используется для хранения объектов, для которых не известен срок жизни на этапе компиляции.

1. Какие есть типы данных в **C#** и как они хранятся в памяти?

**Ответ:** В с# есть два типа данных: значимый и ссылочный. Значимые типы хранятся в стеке, ссылочные типы хранятся в куче, при этом для ссылочных типов в стеке хранится ссылка на адрес участка памяти в куче. Значимые типы подразделяются на bool, byte, char, decimal, double, float, int, long, sbyte, short, uint, ulong, ushort, struct. Ссылочные типы подразделяются на типы классов, интерфейсов, массивов, делегатов.

1. Что такое явное/неявное преобразование типов? Приведите пример.

**Ответ:** Явное преобразование требует применить операцию приведения типов. Неявное не требует особых синтаксических конструкций и выполняется компилятором.

Пример неявного преобразования:

int x = 1;

float y = x;

Пример явного преобразования:

float x = 1.25;

int y = (int)x;

1. Что такое операция упаковки/распаковки? Приведите пример этих операций.

**Ответ:** Операция упаковки представляет собой преобразование значимых типов в ссылочные. Операция распаковки представляет собой преобразование ссылочных типов в значимые.

Пример:

int i = 123;

object o = i; // операция упаковки

int j = (int)o; // операция распаковки

1. Чем отличается логический оператор **&** от **&&**? Приведите пример.

**Ответ:** Логический оператор &, или && возвращают true если все операнды равны true. При использовании логического оператора & происходит проверка обоих частей выражения на true или false, при использовании && проверка второго операнда не будет осуществляться если первый уже равен false. Если нам не важно будет ли выполняться вторая часть проверки, то следует использовать &&

**Пример:**

static void Main(string[] args)

{

static int method1()

{

Console.WriteLine("some text 1");

return 3;

}

static int method2()

{

Console.WriteLine("some text 2");

return 2;

}

int a = 2;

bool x = (a == 1) & (method1() == 4);

bool y = (a == 1) && (method2() == 4);

}

В консоли будет выведено только some text 1

1. Чем отличается цикл **while** от **do-while**?

**Ответ:** Цикл while является оператором с предусловием, то есть код внутри цикла выполняется только если условие истинно. Цикл do-while является оператором с постусловием, то есть сначала будет выполняться код внутри цикла, а затем проверка истинности условия, следовательно код внутри цикла будет выполнен как минимум один раз.

1. Объявить двумерный массив одновременно с его инициализацией?

**Ответ:**

int [,] array = new int [2,2]

{

{1, 2},

{3, 4}

};

1. В чем отличие класса от объекта? Приведите пример класса и объекта.

**Ответ:** Класс является шаблоном для создания объектов. Объекты являются экземплярами классов

static void Main(string[] args)

{

TestClass testObject = new TestClass(); // пример создания объекта

}

public class TestClass //пример класса

{

public int TestField1;

public string TestField2;

public void TestMethod()

{

Console.WriteLine("some text");

}

}

1. Что такое деструктор?

**Ответ:** метод, вызываемый перед уничтожением объекта, чтобы освободить ресурсы.

1. Приведите пример автоматического свойства и во что оно разворичивается?

**Ответ:**

Пример:

public class Person

{

public string Name {get; set;}

}

Разворачивается в:

public class Person

{

public string \_name;

public string Name

{

get

{

return \_name;

}

set

{

\_name=value;

}

}

}

1. Поддерживает ли **C#** множественное наследование? Приведите пример.

**Ответ:** не поддерживает

1. Какое преимущество использования класса **StringBuilder** перед **String**?

**Ответ:** при работе с String, каждая операция изменения строки создает новый объект, при работе с StringBuilder строка изменяется без создания нового объекта, соответственно экономится память

1. Какие существуют модификаторы доступа в **С#**? Опишите их назначение.

**Ответ:** private – элемент с данным модификатором доступен только в том типе, в котором определен.

protected – элемент с данным модификатором доступен в том типе, в котором определен, и в наследниках этого типа.

internal – элемент с данным модификатором доступен без ограничений в той сборке, где он определен

protected internal – элемент с данным модификатором доступен в той сборке где определен, а вне сборки – в наследниках того типа, в котором он определен.

public – элемент с данным модификатором доступен без ограничений, и в сборках где определен, и в других сборках, к которым подключается сборка с данным элементом.

1. Как можно передавать параметры методу в **С#**? Модификаторы параметров.

**Ответ:** Передачу параметров методу можно осуществлять по ссылке и по значению. При передаче параметров по ссылке используется модификатор ref. Выходные параметры объявляются с модификатором out. Входные параметры объявляются с модификатором in. При передаче параметров-списков используется модификатор params.

1. Что такое **var**? Как оно работает?

**Ответ:** Ключевое слово var предназначено для объявления переменных без явного указания их типа данных. Компилятор автоматически определяет тип переменной на основе значения, задаваемой ей при инициализации.

1. Отличаются ли **Equals** и operator **==** ? Если да, в чем отличие?

**Ответ:** Для значимых типов Equals и оператор == проверяют равенство двух значений. Для ссылочных типов Equals проверяет равенство значений, а оператор == проверяет равенство ссылок.

1. Перечислите основыне концепции **ООП**. Коротко опишите, что они означают.

**Ответ:** Инкапсуляция – скрытие деталей реализации объекта и предоставление доступа через публичный интерфейс.

Наследование – создание нового класса-наследника на основе существующего для повторного использования кода.

Полиморфизм- способность объекта одного типа обрабатывать данные по-разному в зависимости от контекста.

Абстракция- выделение общих характеристик объектов и создание на их основе абстракций.

1. Отличие **abstract** метода от **virtual** метода.

**Ответ:** Абстрактный метод не имеет реализации в базовом классе и должен быть реализован в классах-наследниках. Виртуальный метод может иметь стандартную реализацию в базовом классе, и может быть переопределен в классах-наследниках.

1. Назовите методы класса System.Object и их назначение.

**Ответ:** Equals(Object) – определяет равен ли текущий объект другому объекту.

GetHashCode() – служит в качестве хэш-функции по умолчанию.

GetType() – возвращает тип объекта в виде объекта Type.

MemberwiseClone() – создает поверхностную копию текущего объекта.

ReferenceEquals(Object, Object) – определяет, ссылаются ли указанные объекты на один и тот же объект в памяти.

ToString() – возвращает строковое представление текущего объекта.

1. Что такое абстрактный класс? Коротко опишите.

**Ответ:** В абстрактном классе хотя бы один из элементов, таких как методы, свойства, индексаторы, события, объявлен абстрактным, используя ключевое слово abstract. Нельзя создавать объекты абстрактного класса. Абстрактные методы должны быть переопределены в производных классах. Абстрактные члены классов не должны иметь модификатора private. Переопределение осуществляется при помощи ключевого слова override.

1. Как запретить наследование от класса?

**Ответ:** Объявить класс с использованием ключевого слова sealed.

1. Дан следующий цикл:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | int j = 2;  for (int i = 1; i < 100; i = i + 2)  {      j = j - 1;      while(j < 15)      {          j = j + 5;      }  } |

Сколько раз в этом цикле будет выполняться строка j = j – 1?

**Ответ:** 50

Задача 1:

Задан следующий трехмерный массив:

|  |  |
| --- | --- |
| 1  2  3  4  5 | int[,,] mas = { { { 1, 2 },{ 3, 4 } },                  { { 4, 5 }, { 6, 7 } },                  { { 7, 8 }, { 9, 10 } },                  { { 10, 11 }, { 12, 13 } }                }; |

С помощью циклов переберите все элементы этого массива и выведите их на консоль в следующем виде:

{{{1 , 2} , {3 , 4}} , {{4 , 5} , {6 , 7}} , {{7 , 8}, {9 , 10}} , {{10 , 11} , {12 , 13}}}

Задача 2:

Разработайте класс MyString для работы со строками символов. Определите для класса несколько конструкторов и public функцию реверса строки.

Задача 3:

Разработать метод расширения для типа DataTime возвращающий в результате своего выполнения строку в виде «День ‘DD’ месяца ‘MonthName’ года ‘YYYY+ 5508’ от сотворения мира.»

Задача 4: Разработайте класс MyStack, как стек целых чисел фиксированного размера, определите функции pop и push. При разработке нельзя использовать готовые классы , представляющие Stack, ArrayList, List<T> и аналогичные. В качестве внутреннего хранилища использовать только массивы.

Задача 5:

Разработайте базовый класс содержащий 2 поля и метод выводящий в консоль значения этих полей. Разработайте 2 производных класса каждый из которых должен содержать как минимум одно дополнительное поле и переопределять метод базового класса. Напишите код в методе Main() таким образом, чтобы изменяя только одну строку вы выбирали реализацию любого из производных классов.