Экзамен АиП C Sharp

C#

1. C#. Платформа .NET. Процесс выполнения программы.

.NET - это кроссплатформенный фреймворк от компании Microsoft, ранее известная как .NET Framework, когда она ещё была ориентированно только на ОС Windows. Она поддерживает множество языков, главный из которых **C#**, разработанный специально для неё.
Для выполнения кода в .NET служит **CLR** (Common Language Runtime, общеязыковая исполняющая среда). Она выполняет **CIL** (Common Intermediate Language, байт-код) - специальный промежуточный язык, который представляет собой "высокоуровневый ассемблер" виртуальной машины .NET. В него компилируются программы, написанные на .NET совместимых языках. При выполнение программы происходит преобразование CIL кода в машинный код, так называемая **Just-In-Time (JIT)** компиляция. Производительность повышается за счёт того, что во время выполнения компилируется лишь часть кода, к которой происходит обращение.

2. C#. Структура программы. Пространство имен. Сборка.

Структура программы

Базовая структура программы на языке C# включает в себя главный класс и статическую функцию `Main`, принимающую в качестве параметра массив аргументов командной строки:

```
using System;
namespace program
{
    class Program
    {
        public static void Main(string[] argc)
        {

        }
    }
}
```

В новых версия появилась поддержка операторов верхнего уровня, что позволяет писать код программы не в функции `Main`, а сразу в файле, главный класс и функция `Main` тогда создаются автоматически.
Также каждый проект включает файл проекта с расширением **.csproj**, который содержит описание свойств проекта.

Пространства имён

Пространства имён (в примере выше - `program`) используются для объединения кода в логические блоки и предотвращения конфликтов имён. Пространство имён объявляется с помощью специального ключевого слова и последующего блока кода: `namespace <имя> { ... }`. Обращаться к членам пространства имён можно с помощью оператора точки: в примере выше полное имя функции `Main` будет `program.Program.Main`. Чтобы не обращаться к объектам по полному имени, можно подключить пространство имён с помощью директивы `using`. В примере выше так подключается пространство

имён `System`. Пространства имён допускают вложенность. Можно объявить пространство имён для всего файла, добавив в начале: `namespace <имя>;`, оно применится к нему целиком.

Сборка

В результате компиляции приложения создаётся файл `exe` или `dll` (в зависимости от выбранных настроек), который называется сборкой приложения. Сборка является базовой структурной единицей в .NET, на уровне которой проходит контроль версий, развертывание и конфигурация приложения. Каждая сборка включает в себя манифест, метаданные типов, код приложения и ресурсы. Атрибуты сборки, такие как версия, название, автор и прочая информация о продукте, содержатся в специальном файле `AssemblyInfo.cs`.

3. C#. Типы данных. Различия типов-значений и типов-ссылок.

Базовые типы в языке C#:

- **bool** - хранит значение `true` или `false`, представлен системным типом `System.Boolean`
- **byte** - хранит целое число от 0 до 255, системный тип - `System.Byte`
- **sbyte** - знаковый байт, целое число от -128 до 127, системный тип - `System.SByte`
- **short** - знаковое целое число от -32768 до 32767, занимает 2 байта, системный тип - `System.Int16`
- **ushort** - беззнаковое целое число от 0 до 65535, 2 байта, системный тип - `System.UInt16`
- **int** - знаковое целое число от -2147483648 до 2147483647, занимает 4 байта, системный тип - `System.Int32`, тип по умолчанию для численных литералов
- **uint** - беззнаковое целое число от 0 до 4294967295, занимает 4 байта, системный тип - `System.UInt32`
- **long** - знаковое целое число от –9223372036854775808 до 9223372036854775807, 8 байт, системный тип - `System.Int64`
- **ulong** - беззнаковое целое число от 0 до 18446744073709551615, 8 байт, системный тип - `System.UInt64`
- **float** - число с плавающей точкой от $-3.4 \cdot 10^{38}$ до $3.4 \cdot 10^{38}$, 4 байта, системный тип - `System.Single`
- **double** - число с плавающей точкой от $\pm 5.0 \cdot 10^{324}$ до $\pm 1.7 \cdot 10^{308}$, 8 байт, системный тип - `System.Double`
- **decimal** - десятичное дробное число, без десятичной запятой имеет значение от $\pm 1.0 \cdot 10^{-28}$ до $\pm 7.9228 \cdot 10^{28}$, 16 байт, системный тип `System.Decimal`
- **char** - одиночный символ в кодировке Unicode, 2 байта, системный тип - `System.Char`, тип по умолчанию для символьных литералов
- **string** - набор символов Unicode, системный тип - `System.String`, тип по умолчанию для строковых литералов
- **object** - может хранить значения любого типа данных, занимает 4 байта на 32-разрядной системе и 8 на 64-разрядной, системный тип - `System.Object`, являющийся базовым для всех других типов и классов .NET.

В C# возможна неявная типизация с использованием ключевого слова `var` (аналог `auto` в C++):

```
// можно писать так:
var hello = "hello world";
var a = 1;
var b = 12.345;
var c = 'A';
```

```
// но так нельзя!
var n = null;
```

Все типы данных подразделяются на типы-значения и типы-ссылки в зависимости от того, как для них происходит организация памяти. Память подразделяется на стек и кучу.

- Объекты типов-значений размещаются в стеке, по адресу непосредственно хранится само значение переменной или параметра функции. К таким типам относятся все целочисленные типы, типы чисел с плавающей запятой, типы `decimal`, `bool`, `char`, перечисления `enum` и структуры `struct`
- Объекты ссылочных типов хранятся в куче, в стеке на них хранится лишь ссылка. Ссылочными типами являются `object`, `string`, все классы, интерфейсы и делегаты.

Процесс копирования разных типов различается: типы-значения копируются по значению, а для типов-ссылок копируется лишь ссылка на объект, то есть 2 ссылки начинают указывать на одну и ту же область в памяти. Это также относится к более сложным ситуациям, когда структура содержит поле с типом класса и при копировании этой структуры, получится, что все поля с типами-значениями скопировались по значению, а это поле класса по ссылке, то есть 2 разных объекта будут иметь поля с ссылкой на один и тот же объект. Таким образом, ссылки работают как указатели и дают те же возможности.
Стоит учитывать, что при передаче объекта класса в функцию через параметры, передаётся копия ссылки на исходный объект, то есть функция получает к нему доступ и может изменить его поля. При этом функция не может изменить сам объект, так как передана лишь копия ссылки. Чтобы это сделать, нужно использовать ключевое слово `ref`, тогда станет возможно, например, создание нового другого объекта и сохранение его по исходной ссылке.

4. C#. Литералы. Примеры. Переменные. Примеры. Область действия переменной.

Литералы

Литералы (константы) - неизменяемые значения. Их можно передавать переменным в качестве значения. Виды литералов в C#:

- логические литералы: `true` и `false`
- целочисленные литералы:
  - в десятичной форме: `1`, `-2`, `123`
  - в двоичной форме: `0b11`, `0b1011`, `0b100001`
  - в шестнадцатеричной форме: `0x0A`, `0xFF`, `0xA1`
- вещественные литералы:
  - в десятичной форме: `12.34`, `-0.28`
  - в экспоненциальной форме: `3.2e3` ($3.2 \cdot 10^3 = 3200$), `1.2E-1`
- символьные литералы: `'A'`, `'\n'`, `'\x75'`, `'\u0420'`
- строковые литералы: `"hello world"`
- `null` - ссылка, которая не указывает ни на какой объект

Переменные

Для хранения данных в программе применяются переменные. Переменная представляет именованную область памяти, в которой хранится значение определенного типа. Переменная имеет тип, имя и значение. Тип определяет, какого рода информацию может хранить переменная. Чтобы объявить переменную, нужно указать её тип и имя, например, переменная целого типа с именем `a`: `int a;`,

строка `str`: `string str;`
Имя переменной может содержать любые цифры, буквы и символ подчёркивания, при этом первый символ должен быть буквой или подчёркиванием. В нём не должно быть пробелов и знаков пунктуации и оно не должно являться ключевым словом.
Чтобы инициализировать переменную, ей нужно присвоить значение:

```
int a; // объявляем переменную целого типа
a = 5; // присваиваем значение
char c = 'a'; // можно объявить переменную и сразу же инициализировать её
string name = "Max";
```

Для переменных существует модификатор `const`, который запрещает изменение значения после инициализации (она при этом обязательна при объявлении переменной):

```
// константы принято называть большими буквами
const int SIZE = 10; // объявление константы с значением 10
SIZE = 5; // ошибка!!! константы нельзя изменить
```

Области действия (видимости) переменных

Каждая переменная существует только в определённом контексте:

- Контекст класса - переменные, объявленные на уровне класса, доступны во всех его методах, их называют глобальными или полями класса.
- Контекст метода - переменные, объявленные на уровне метода, являются локальными и доступны только в рамках этого метода.
- Контекст блока кода - переменные, объявленные на уровне блока кода, также являются локальными и доступны только в этом блоке кода.

```
class Obj
{
        int n = 10; // переменная уровня класса
        // в этом месте досутпна только переменная n
        public void Print()
        {
                char c = 'A'; // локальная переменная метода
                // в этом месте доступны переменные n и c
                { // блок кода
                        string str = "hello"; // локальная переменная блока кода
                        // здесь доступны все 3 переменные: n, c и str
                }
        }
        public void Save()
        {
                // здесь доступно только поле класса n
        }
}
```

5. C#. Выражения. Преобразование типов при выполнении операций. Примеры.

6. C#. Ввод-вывод консольного приложения. Примеры.

7. C#. Исключения. Примеры.

8. C#. Объявление классов и их компонентов. Примеры.

9. C#. Спецификации доступа классов, структур и их компонентов.

10. C#. Конструкторы классов. Примеры.

11. C#. Поля: константные, объекта, класса, только для чтения. Примеры.

12. C#. Методы: конструкторы объектов, статические, деструкторы. Примеры.

13. C#. Методы: объектов, классов. Примеры.

14. C#. Параметры методов. Передача параметров по значению и по ссылке. Выходные параметры. Примеры.

15. C#. Одномерные массивы с элементами типов-значений и ссылочных типов. Примеры объявления.

16. C#. Оператор foreach. Примеры применения для массивов разных типов.

17. C#. Массивы прямоугольные и ступенчатые. Различие. Примеры

18. C#. Строка String. Примеры создания и использования.

19. C#. Регулярные выражения. Примеры.

20. C#. Структуры. Примеры.

21. C#. Наследование. Пример.

22. C#. Полиморфное наследование. Абстрактные классы. Пример.

23. C#. Композиция и агрегация. Пример.

24. C#. Интерфейсы. Пример.

25. C#. Свойства. Пример.