**Types**

**Source File->**C++, C#, Eiffel, Python, Visual Basic, Jscript, Fortran  
**IL - Езиково-специфичен компилатор->**C++, C# **Изпълним файл ->**IL Code & Metadata  
**Runtime->**Изпълнима Среда  
**CLR Type System**

**Value types(Стойностни типове)**

**CIL FCL description CLS   
 support**

bool System.Boolean true false Y

char System.Char Unicode character Y

int8 System.SByte signed 8 bits N

int16 System.Int16 signed 16 bits Y

int32 System.Int32 signed 32 bits Y

int64 System.Int64 signed 64 bits Y

unsigned int8 System.Byte unsigned 8 bits Y

unsigned int16 System.UInt16 unsigned 16 bits N

unsigned int32 System.UInt32 unsigned 32 bits N

unsigned int64 System.UInt64 unsigned 64 bits N

float32 System. Single IEEE 32 bits FP Y

Float64 System.Double IEEE 64 bits FP Y

native int System.IntPtr signed native int Y

native unsigned int System.UIntPtr unsigned native int N

**Sealed Value Type**

value type не се наследява   
\* не допуска virtual методи   
\* тип може да се маркира като sealed (C#, Sealed в CLR) – не може да бъде базов;   
\* тип може да се маркира като abstract ( C#, Abstract в CLR) – не допуска инстанция, но може да се ползва за базов. Ако наследникът не е абстрактен, от него могат да се правят инстанции;   
\*засега не е допустимо Abstract Sealed. Затова, ако ще дефинирате собствен тип само със static членове , маркирате го sealed. Дефинирате му и private, безпараметърен конструктор (не може да се вика отвън за инстанция, а и пречи на компилатора да създаде свой по подразбиране) и така постигате абстракцията. Такива са напр. обекти Math, Console.  
**Boxed types** Стойностните типове са олекотени, в стека а не в хипа, не се управляват от GC, не се сочат от указатели. Това е достатъчно в много случаи, но не винаги...  
 Например, искаме да създадем масив ( стандартен тип ArrayList pp;) от точки- дефинирани като struct Point{…}. После да викаме pp.Add(my\_point); Или : Point new\_point = (Point) pp[0];  
Всеки стойносен тип може да се конвертира в референтен (операцията е boxing). Възможност за всеки value type, вкл. user-defined. Автоматично се генерира обвиващ обект (boxing) след копиране на стойността на value типа в boxed обект който се е алокирал в heap. Операция unboxing – връща value type в нормално състояние. Изработва указател към данната (до момента съдържана в boxed обект), копира данните на boxed обекта във value type (в стека). boxed обектите са в heap.  
Всеки value type, може да се разглежда като обект, при необходимост и да взаимодейства с всеки друг обект (всички са наследници на Object). Общото наследство наподобява IUnknown в компонентното програмиране. Всеки boxed type поддържа интерфейси ,чийто методи могат да се ползват.  
**Reference Types(референтни типове)**Комбинира информация за местоположение и за съдържание. Местоположението е “Type Safe”, т.е. само assignment-compatible типове могат да се пазят там. Garbage collector мести тези обекти свободно. Затова достъпът до тях е през силно типизирани референции, а не директно.  
Object Types: всички класове го наследяват, винаги са в GC heap, всички наследяват Object:  
System.Object  
Основни методи на типа (с възможност за припокриване) са:  
***Equals*** (реализацията проверява или identity или equality); Finalize (вика се преди gc да освободи паметта); ***GetType*** (важно е че методът е невиртуален – т.е. не е възможно някой да се представи за друг); ***ToString*** ( връща информация за типа, ако не се припокрие).  
**Тип String** Той е sealed, всеки метод, модифициращ низ, създава нов низ (immutable). Основни методи: конструктори ( няма default) за различни аргументи – char, масив char, указатели към char ***Compare(); Concatenate(); Format(); IndexOf(); Insert(); Length(), ….*Interface Types(интерфейсни типове)** Въведени в COM и CORBA. Целта е не само функционална наследяемост(inheritance), но и възможност за поделяне на общ външен договор ( интерфейс) между несвързани обекти. Интерфейсният тип е частична спецификация на тип. Поддържа се наследяемост на интерфейсни типове. Интерфейсният тип съдържа: **методи** (статични и на инстанция ) **полета** (статични) но не и в C# property events. Методите на интерфейс са ***public, abstract, virtual***. Framework поддържа множество интерфейсни типове. CLR не поддържа множествено наследяване на типове, но поддържа такова за интерфейси.  
**Pointer type  
*unmanaged function pointer***   
***managed pointers*** (CLS съвместими и GC управляеми)   
***unmanaged pointer referring values*** (CLS несъвместими).Не всички езици ги поддържат синтаксисът е езиково специфичен.

***User- Defined Object Type***