Jacek Matulewski, e-mail: *jacek@fizyka.umk.pl*
Strona wykładu: *http://www.fizyka.umk.pl/~jacek/dydaktyka/programowanie/*

.NET Framework, .NET Core i .NET Standard

W 2002 roku Microsoft opublikował pierwszą wersję **platformy .NET** (ang. *.NET Framework*) i dedykowanego jej języka C#, które były odpowiedzią na języka Java i jego wirtualną maszynę rozwijane wówczas przez firmę Sun. To była rewolucja w programowaniu aplikacji – w obu platformach celem kompilacji nie był konkretny typ komputera z jego specyficzną architekturą, ale wirtualne środowisko uruchomieniowe. W przypadku platformy .NET środowisko to nazywa się CLR (ang. *Common Language Runtime*). Takie podejście pozwoliło na większą przenaszalność wytworzonego oprogramowania, a tym samym znacząco zmniejszało koszty jego wytwarzania. Równocześnie utrzymano wysoką wydajność dzięki technologii podwójnej kompilacji (o tym poniżej).

Platforma .NET była rozwijana aż do wersji o numerze 4.8, która została opublikowana w kwietniu 2019. W tym momencie jej młodsza siostra, platforma **.NET Core**, która jest rozwijana dopiero od 2016 roku, ale w 2019 miała już wersję 3.0, uzyskała wystarczającą dojrzałość, aby przejął pałeczkę sztandarowej platformy Windows. Platforma .NET Core jest wolna i otwarta – rozwijana jest przez społeczność działającą pod kierunkiem Microsoft, a jej kod źródłowy jest publicznie dostępny. Od wersji 3.0 .NET Core pozwala na tworzenie tzw. aplikacji desktopowych tj. aplikacji z graficznym interfejsem użytkownika (okienkowych), korzystając zarówno z kontrolek WPF, jak i Windows Forms. Platforma .NET Core ma ponadto tę ogromną przewagę nad starszą .NET Framework, że pozwala na tworzenie oprogramowania nie tylko dla systemu Windows, ale również dla Linux i macOS. Mówiąc o wieloplatformowości, należy pamiętać także o platformie UWP. Jest częścią wielkiego, ale niestety niezbyt udanego projektu „Windows na każdym ekranie". Platforma UWP jest bowiem dostępna zarówno dla systemu Windows 10 (korzystają z niej aplikacje pobierane ze sklepu on-line *Microsoft Store*), jak i na konsole Xbox oraz systemy mobilne Windows 10 Mobile. Problem w tym, że te ostatnie się nie przyjęły. Nie należy także zapominać o rozwijanej od 2004 roku platformie Mono, której celem było umożliwienie przenoszenia aplikacji .NET na systemy inne niż Windows. Platforma ta straciła na znaczeniu po uruchomieniu projektu .NET Core, ale nadal jest ważnym elementem Xamarin. Xamarin to jeszcze jedna platforma związana z Microsoft i językiem C#, która pozwala na przygotowywanie w języku C# aplikacji mobilnych dla systemów Android i iOS (także dla Windows 10 Mobile). Jej ważnym atutem jest możliwość przygotowania wspólnego „rdzenia" aplikacji na te platformy, co bardzo ułatwia i zmniejsza koszty utrzymywania aplikacji mobilnych.

Taka mnogość platform może być przytłaczająca, a dodajmy do tego jeszcze Unity, czyli system tworzenia gier w języku C#, który także jest wspierany przez Microsoft i również korzysta z platformy .NET. Dobrą stroną jest to, że oprogramowanie dla wszystkich tych platform można przygotowywać w jednym języku. Przydatna byłaby jednak także możliwość łatwego przenoszenia kodu źródłowego między nimi, a nawet gotowych skompilowanych bibliotek. Stąd pomysł utworzenia **.NET Standard**. To nie jest kolejna platforma. To specyfikacja platform z rodziny .NET, włączając w to Mono i UWP. Nie można jednak tworzyć aplikacji zgodnych z .NET Standard, można natomiast tworzyć takie biblioteki. Najnowsza wersja tej specyfikacji to 2.1. Jest z nią zgodna .NET Core 3.0, ale nie będzie już zgodnej z nią platformy .NET Framework. Natomiast w przypadku UWP i Unity takie wersje dopiero są zapowiadane.

Tabela 2.1. Zgodność standardu .NET Standard z poszczególnymi wersjami platform[1]

.NET Standard	**1.0**	**1.1**	**1.2**	**1.3**	**1.4**	**1.5**	**1.6**	**2.0**	**2.1**
.NET Core	1.0	1.0	1.0	1.0	1.0	1.0	1.0	2.0	3.0
.NET Framework	4.5	4.5	4.5.1	4.6	4.6.1	4.6.1	4.6.1	4.6.1	nie ma i nie będzie
Mono	4.6	4.6	4.6	4.6	4.6	4.6	4.6	5.4	6.4
Xamarin.iOS	10.0	10.0	10.0	10.0	10.0	10.0	10.0	10.14	12.16
Xamarin.Mac	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.8	5.16
Xamarin.Android	7.0	7.0	7.0	7.0	7.0	7.0	7.0	8.0	10.0
UWP (wersja Windows)	10.0	10.0	10.0	10.0	10.0	10.0.16299	10.0.16299	10.0.16299	jeszcze nie przygotowano
Unity	2018.1	2018.1	2018.1	2018.1	2018.1	2018.1	2018.1	2018.1	jeszcze nie przygotowano

Warto powiedzieć też kilka słów o języku C# i jego historii. Został zaprojektowany i jest rozwijany przez grupę programistów Microsoftu pod kierunkiem Andersa Hejlsberga. Jest to osoba, która wcześniej była odpowiedzialna za kompilator Turbo Pascala, środowisko Delphi i bibliotekę VCL w firmie Borland, a od 1996 roku, już w Microsofcie, za projekt J++, a potem C# i platformę .NET. Język C#, jeżeli brać pod uwagę jego słowa kluczowe i podstawową składnię, bez wątpienia należy do tej samej rodziny co języki C++ i Java. Dynamiczny rozwój tego języka spowodował jednak, że pojawiło się w nim ostatnio wiele nowych konstrukcji wykraczających poza ramy ustalone przez standard języka C++. Najlepszym przykładem są zapytania LINQ i używane w nich operatory lub wsparcie dla programowania asynchronicznego. Inne elementy języka były natomiast często wprowadzone w C# wcześniej, niż w C++, czego dobrym przykładem są wyrażenia Lambda. Rozpoczynając naukę języka C#, można więc bazować na znajomości innych języków, ale warto przygotować się również na nowości. Najnowsza wersja języka C# 8.0 implementowana jest w platformie .NET Core 3.0 (specyfikacja .NET Standard 2.1). Ostatnia wersja platformy .NET Framework 4.8 jest natomiast zgodna z C# 7.0.

Ważnym składnikiem technologii .NET Core jest środowisko uruchomieniowe **CoreCLR**, odpowiednik CLR w .NET Framework. Separuje ono aplikacje od warstwy systemu operacyjnego, co ma kilka istotnych korzyści. Przede wszystkim pozwala chronić stabilność systemu. Ponadto pozwala na wprowadzenie nowych mechanizmów, których nie oferuje sam system operacyjny. Najlepszym przykładem jest zarządzanie pamięcią. Korzyścią, którą chciałbym jednak podkreślić w szczególny sposób, jest przenaszalność. Platforma .NET Core jest dostępna nie tylko dla systemu Windows, ale również Linux i macOS.

Kod źródłowy C# aplikacji lub biblioteki nie jest kompilowany bezpośrednio do kodu maszynowego rozumianego przez procesor. Kompilacja jest dwustopniowa. W pierwszej fazie zostaje on skompilowany do kodu pośredniego, wspólnego dla wszystkich środowisk uruchomieniowych .NET Core, bez względu na system operacyjny i procesor. Ten kod jest czytelny dla środowiska uruchomieniowego CoreCLR. Druga faza to kompilacja kodu pośredniego przez CoreCLR za pomocą kompilatorów *Just-In-Time* (skrót JIT). Nie odbywa się ona bezpośrednio po pierwszej kompilacji ani nawet na tym samym komputerze. Platforma .NET Core przeprowadza ją dopiero w momencie uruchamiania programu.

Nieco zamieszania może powodować fakt, że wynikiem pierwszej kompilacji kodu źródłowego C# są pliki *.exe* (w .NET Core 3.0 i nowszych) lub *.dll* (do .NET Core 2.2). Mimo tego samego rozszerzenia i nagłówka rozpoczynającego się od „MZ" struktura tych plików jest zupełnie inna niż tradycyjnych plików wykonywalnych i bibliotek zawierających kod uruchamiany w platformie Win32 lub Win64. Zawierają one bowiem kod pośredni. Z punktu widzenia użytkownika, jeżeli tylko platforma .NET jest zainstalowana w systemie, nie ma to jednak większego znaczenia — po prostu uruchamia on program, klikając dwukrotnie plik *.exe* (w .NET Core 3.0 i nowszych). Jednak z punktu widzenia systemu operacyjnego różnica jest ogromna.

[1] Źródło: *https://docs.microsoft.com/pl-pl/dotnet/standard/net-standard*