ВЪВЕДЕНИЕ В ДИСЦИПЛИНАТА
„Приложно програмиране“

Дисциплината „Приложно програмиране“ има за цел студентите да придобият знания и умения за създаване на софтуерни приложения чрез средствата на езика от високо ниво C#. Дисциплината запознава студентите с архитектурата на .NET платформата, принципите на обектно ориентираното програмиране и типовете данни използвани в .NET. Изучават се теми свързани с масиви, регулярни изрази, входно/изходната система в .NET и управление на паметта в .NET. Разглежда се изграждането на графичен потребителски интерфейс с използване на Windows Forms и изграждане на web приложения с ASP.NET. Изучаваните програмни техники се онагледяват с демонстрационни програми.

За провеждане на лабораторните занятия се използва интегрираната развойна среда (IDE – Integrated Development Environment) Microsoft Visual Studio.

ВЪВЕДЕНИЕ В ДИСЦИПЛИНАТА „Приложно програмиране“

Изпитът е писмен под формата на тест, съдържащ въпроси от теоретичния и от практическия материал. Част от въпросите са с възможен избор на отговора, а другата част изискват формулиране на кратък отговор от студента, като всеки от въпросите дава определен брой точки в зависимост от неговата сложност. Времето за решаване на теста е един астрономически час. Оценките от теста се формират съгласно Таблица 1, където MAX е максималният брой точки от теста.

Таблица 1. Оценяване на изпитния тест

Оценка	Среден(3)	Добър(4)	Мн. добър(5)	Отличен(6)
Необходими точки	40% от MAX	60% от MAX	80% от MAX	90% от MAX

Всеки студент получава индивидуална задача за самостоятелна работа до 5-та учебна седмица (включително). Решената задача трябва да бъде защитена в деня на изпита. Окончателната оценка се формира на база резултатите от изпитния тест (70%) и разработената задача (30%) !

ВЪВЕДЕНИЕ В ДИСЦИПЛИНАТА
„Приложно програмиране“

ПРЕПОРЪЧИТЕЛНА ЛИТЕРАТУРА

Основна

1. Светлин Наков и колектив, Основи на програмирането със C#, Фабер, гр. Велико Търново, София, 2017, ISBN: 978-619-00-0635-0
2. Светлин Наков, Веселин Колев и колектив, Принципи на програмирането със C#", Фабер, Велико Търново, 2018, ISBN: 978-619-00-0778-4
3. Светлин Наков и колектив, Програмиране за .NET Framework – Том 1, Faber, 2005-2006, ISBN: 954-775-505-6
4. Светлин Наков и колектив, Програмиране за .NET Framework – Том 2, Faber, 2005-2006, ISBN: 954-775-672-9; 978-954-775-672-4
5. Денис Колисниченко, Въведение в .Net - практическо програмиране на C#, Асеневци, 2016, ISBN: 9789548898867
6. Microsoft technical documentation, https://docs.microsoft.com/en-us/
7. C# documentation, https://docs.microsoft.com/en-us/dotnet/csharp/

ВЪВЕДЕНИЕ В ДИСЦИПЛИНАТА
„Приложно програмиране“

ПРЕПОРЪЧИТЕЛНА ЛИТЕРАТУРА

Допълнителна

1. Христо Крушков, Практическо ръководство по програмиране на C#, Коала Прес, 2017, ISBN 9786197134445
2. Andrew Stellman and Jennifer Greene, Head First C#, O’Reilly Media, 2010, ISBN: 978-1-449-38034-2
3. Chris Sells, Windows Forms Programming in C#, AddisonWesley Professional, 2003, ISBN-13: 978-0321116208, ISBN-10: 0321116208

ЛЕКЦИЯ 1

СЪДЪРЖАНИЕ

I. ЗАПОЗНАВАНЕ С АРХИТЕКТУРАТА НА ПЛАТФОРМАТА .NET И .NET FRAMEWORK

I.1. Какво представляват платформата .NET ?

Платформата .NET представлява обобщена визия на компанията Microsoft, че информацията трябва да бъде максимално достъпна за хората. .NET платформата е съвкупност от технологии, които свързват хората с информацията – навсякъде, по всяко време, от всяко устройство. .NET платформата осигурява стандартизирана инфраструктура за разработка, използване, хостинг и интеграция на .NET приложения и XML уеб услуги, базирана на .NET сървърите на Microsoft, средствата за разработка (.NET Framework и Visual Studio .NET).

.NET Framework е само част от .NET платформата – тази част, която е насочена към разработчиците на софтуер. Тя осигурява среда за разработка и контролирано изпълнение на .NET приложения и предоставя програмен модел и библиотеки от класове за разработка, независимо от езиците за програмиране.

I. ЗАПОЗНАВАНЕ С АРХИТЕКТУРАТА НА ПЛАТФОРМАТА .NET И .NET FRAMEWORK

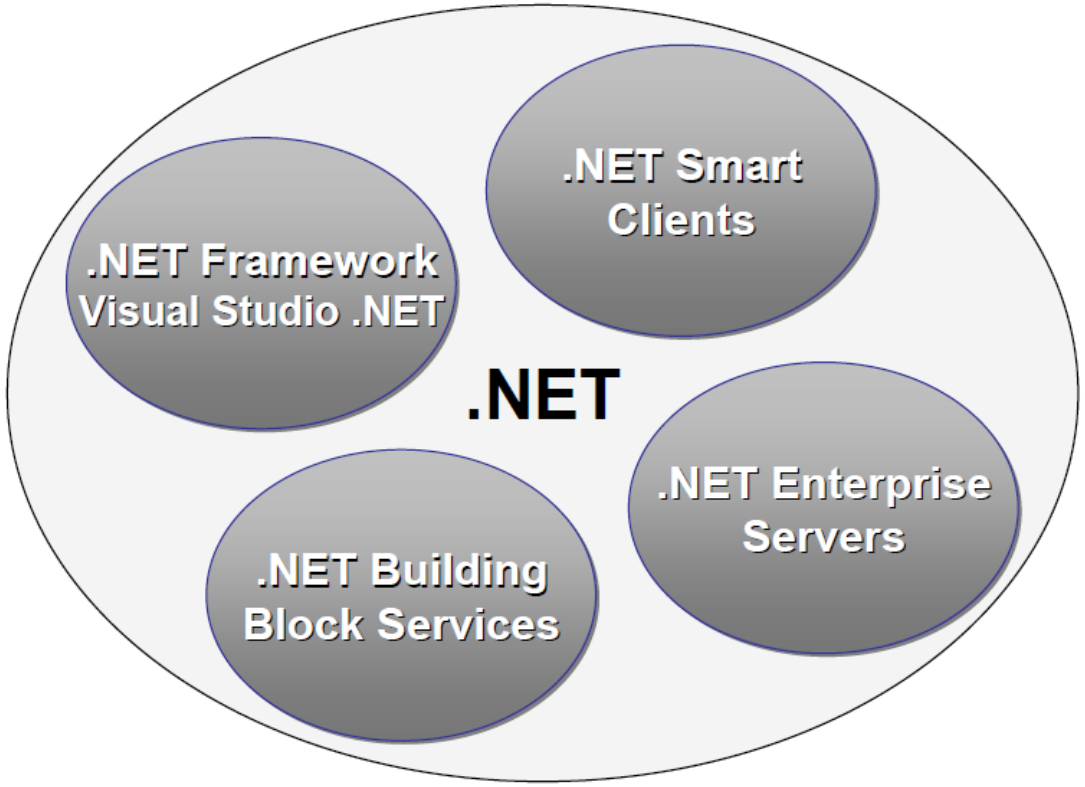
1

I.2. Архитектура на .NET платформата

Платформата .NET обединява в себе си четири технологични и идеологически компонента:

1) Инфраструктурата от сървъри .NET Enterprise Servers;
2) Средствата за разработка .NET Framework и Visual Studio .NET;
3) Глобалните услуги .NET Building Block Services;
4) Идеологията .NET Smart Clients.

Нека разгледаме тези компоненти по отделно!



I. ЗАПОЗНАВАНЕ С АРХИТЕКТУРАТА НА ПЛАТФОРМАТА .NET И .NET FRAMEWORK

.NET Enterprise Servers

.NET Enterprise Servers предоставят сървърната инфраструктура на .NET платформата и същевременно среда за изпълнение, управление и интеграция на XML уеб услуги.

Ключовите характеристики на .NET Enterprise сървърите са:

- Силна поддръжка на XML – всички .NET сървъри използват широко XML стандарта за представяне и обмяна на информация;
- Висока надеждност – ключова характеристика, изключително важна за бизнеса;
- Добра скалируемост – възможност за поемане на огромно натоварване при необходимост;
- Управление и контрол на бизнес процесите в приложенията и услугите (business process orchestration);
- Повишена сигурност – сигурността е основна архитектурна концепция при .NET сървърите.
- Лесно управление – леснота за администриране, настройка, наблюдение и управление на работата на сървърите.

I. ЗАПОЗНАВАНЕ С АРХИТЕКТУРАТА НА ПЛАТФОРМАТА .NET И .NET FRAMEWORK

.NET Framework и Visual Studio .NET

.NET Framework е софтуерна платформа за разработка и изпълнение на .NET приложения. Тя предоставя програмен модел и стандартна библиотека с класове за разработка на приложения и унифицирана среда за изпълнение на управляван код. Поддържа различни езици за програмиране и позволява тяхната съвместна работа.

.NET Framework съществува в два варианта:

- .NET Framework – пълна версия.
- .NET Compact Framework – съкратена версия за изпълнение върху мобилни устройства. Създадена е специално за устройства с ограни-чени хардуерни ресурси.

Visual Studio .NET представлява цялостна интегрирана среда за разработка на .NET приложения. Позволява създаване на различни видове приложения, писане на програмен код, изпълнение и дебъгване на приложения, изграждане на потребителски интерфейс и др. VS.NET предоставя единна среда за всички технологии и за всички програмни езици, поддържани стандартно от .NET Framework (C#, VB.NET, C++ и J#).

I. ЗАПОЗНАВАНЕ С АРХИТЕКТУРАТА НА ПЛАТФОРМАТА .NET И .NET FRAMEWORK

.NET Building Block Services

.NET Building Block Services са съвкупност от XML уеб услуги, насочени към крайния потребител. Основната им задача е да осигуряват персонализиран достъп до данните на даден потребител по всяко време и от всякакво устройство. За целта се използват отворени стандарти и протоколи за комуникация. .NET Building Block Services са създадени с цел да позволяват лесна интеграция с други услуги и приложения и да позволяват връзка между тях. Ето няколко области, в които има изградени такива Building Block услуги:

- автентикация – на базата на .NET Passport
- доставка на съобщения
- съхранение на лични потребителски данни – документи, контакти, електронна поща, календар, любими сайтове и други
- съхранение на настройки на приложения, които потребителят използва.

I. ЗАПОЗНАВАНЕ С АРХИТЕКТУРАТА НА ПЛАТФОРМАТА .NET И .NET FRAMEWORK

.NET Smart Clients

Smart clients представлява архитектурна концепция, която позволява изграждането на клиентски приложения, които:

- предоставят гъвкав потребителски интерфейс (за разлика от уеб приложенията и WAP приложенията)

- използват XML уеб услуги (чрез които си осигуряват връзка с останалия свят и обменят данни със сървърите, които съхраняват и обработват техните данни)

- могат да работят в online и offline режим (като синхронизират данните си когато са online)

- имат възможност да се самообновяват (и това може да става автоматично, с минимални усилия от страна на потребителя).

.NET smart клиентите работят както върху обикновени настолни компютри, така и върху различни преносими устройства: мобилни телефони, hand held устройства, вградени системи и т. н.

I. ЗАПОЗНАВАНЕ С АРХИТЕКТУРАТА НА ПЛАТФОРМАТА .NET И .NET FRAMEWORK

I.3. Какво представляват .NET Framework ?

.NET Framework е среда за разработка и изпълнение на приложения за .NET платформата. Тя предоставя програмен модел, библиотеки от типове и единна инфраструктура за разработка на приложения и поддържа различни езици за програмиране.

Приложенията, базирани на .NET Framework, се компилират до междинен код (на езика IL) и се изпълняват контролирано от средата за изпълнение на .NET Framework. Компилираният .NET код се нарича още управляван код и може да работи без да се прекомпилира върху различни платформи, за които има имплементация за .NET Framework (Windows, Linux и др.).

I. ЗАПОЗНАВАНЕ С АРХИТЕКТУРАТА НА ПЛАТФОРМАТА .NET И .NET FRAMEWORK

I.3.1. Компоненти на .NET Framework:

Можем да разделим .NET Framework на два основни компонента:

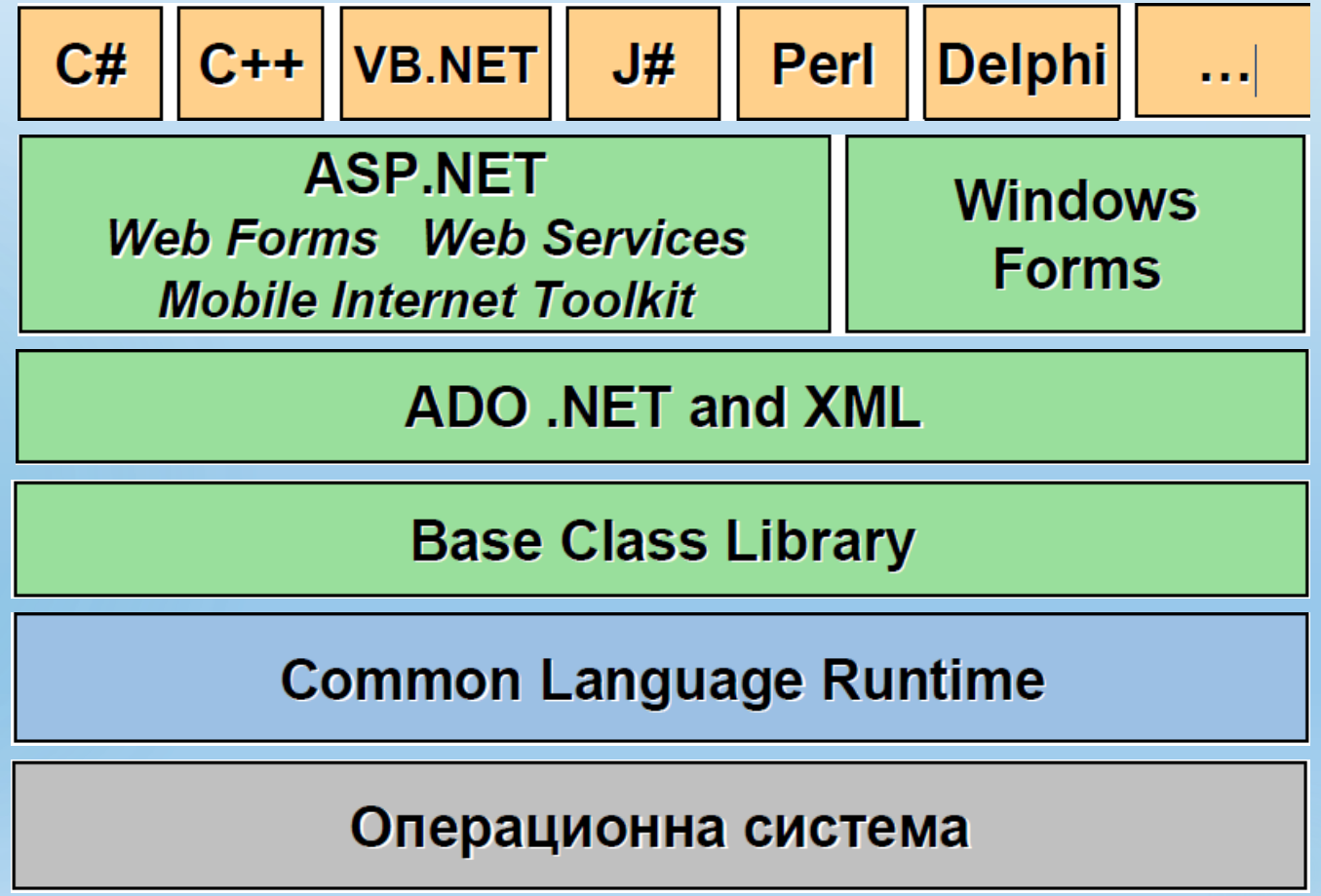
- Common Language Runtime (CLR) – средата, в която се изпълнява управляваният код на .NET приложенията. Представлява виртуална машина, която контролирано изпълнява .NET кода и осигурява различни услуги, като управление на сигурността, управление на паметта и др.

- Framework Class Library (FCL) – представлява основната библиотека от типове, които се използват при изграждането на .NET приложения. Съдържа основната функционалност за разработка, необходима за повечето приложения, като вход/изход, връзка с бази данни, работа с XML, изграждане на уеб приложения, използване на уеб услуги, изграждане на графичен потребителски интерфейс и др. Стандартните класове и типове от FCL можем да използваме навсякъде, където има инсталиран .NET Framework.

I. ЗАПОЗНАВАНЕ С АРХИТЕКТУРАТА НА ПЛАТФОРМАТА .NET И .NET FRAMEWORK

I.3.2. Архитектура на .NET Framework

Архитектурата на .NET Framework често пъти се разглежда на нива.



I. ЗАПОЗНАВАНЕ С АРХИТЕКТУРАТА НА ПЛАТФОРМАТА .NET И .NET FRAMEWORK

Операционна система

Операционната система управлява ресурсите, процесите и потребителите на машината. Средата, която изпълнява .NET приложенията (CLR), е обикновен процес в операционната система и се управлява от нея, както останалите процеси. Най-често операционната система, която изпълнява CLR е Microsoft Windows, но .NET Framework има имплементации и за други операционни системи.

Common Language Runtime (CLR)

Общата среда за изпълнение Common Language Runtime (CLR) управлява процеса на изпълнение на .NET код. Тя се грижи за заделяне и освобождаване на паметта, управлява конкурентността, грижи се за сигурността на приложенията и изпълнява други важни задачи, свързани с изпълнението на кода.

I. ЗАПОЗНАВАНЕ С АРХИТЕКТУРАТА НА ПЛАТФОРМАТА .NET И .NET FRAMEWORK

Base Class Library

Base Class Library (BCL) е част от стандартната библиотека на .NET Framework – Framework Class Library (FCL). BCL представлява богата обектно-ориентирана библиотека с основни класове, които осигуряват базова системна функционалност. BCL осигурява вход-изход, работа с колекции, символни низове, мрежови ресурси, сигурност, отдалечено извикване, многонишковост и др. Технологиите ADO.NET, XML, ASP.NET и Windows Forms не са част от BCL, тъй като те са по-скоро допълнителни библиотеки, отколкото базова системна функционалност.

ADO.NET и XML

Слоят на ADO.NET и XML предоставя удобен начин за работа с релационни и други бази от данни и средства за обработка на XML.

I. ЗАПОЗНАВАНЕ С АРХИТЕКТУРАТА НА ПЛАТФОРМАТА .NET И .NET FRAMEWORK

ASP.NET и Windows Forms

ASP.NET и Windows Forms изграждат слоя за интерфейс към крайния потребител на приложенията и ни предоставят богата функционалност за създаване на уеб и Windows базиран потребителски интерфейс, както и уеб услуги. ASP.NET позволява по лесен начин да бъдат изграждани гъвкави динамични уеб сайтове и уеб приложения и уеб услуги. Windows Forms позволява изграждане на прозоречно-базиран графичен потребителски интерфейс с богати възможности. ASP.NET и Windows Forms използват компонентно-базирана архитектура и благодарение на нея позволяват изграждане на потребителския интерфейс визуално, чрез сглобяване на компоненти в специално разработени за това редактори, предоставени от средите за разработка.

I. ЗАПОЗНАВАНЕ С АРХИТЕКТУРАТА НА ПЛАТФОРМАТА .NET И .NET FRAMEWORK

Езици за програмиране

.NET Framework позволява на разработчика да използва различни езици за програмиране, както и да интегрира в едно приложение компоненти, разработвани на различни езици. Възможно е дори клас, написан на един език, да бъде наследен и разширен от клас, написан на друг език. Microsoft .NET Framework поддържа стандартно езиците C#, VB.NET, Managed C++ и J#, но трети доставчици предлагат допълнително .NET версия на още много други езици, като Pascal, Perl, Python, Fortran, Cobol и други.

I. ЗАПОЗНАВАНЕ С АРХИТЕКТУРАТА НА ПЛАТФОРМАТА .NET И .NET FRAMEWORK

I.3.3. Подробности за Common Language Runtime

Common Language Runtime (CLR) е сърцето на .NET Framework. Той представлява среда за контролирано изпълнение на управляван код. На практика CLR е тази част от .NET Framework, която изпълнява компилираните .NET програми в специална изолирана среда.

В своята същност CLR представлява виртуална машина, която изпълнява инструкции, на езика IL (Intermediate Language), езикът до който се компилират всички .NET езици. CLR е нещо като виртуален компютър, който обаче не изпълнява асемблерен код за процесор Pentium, AMD или някакъв друг, а IL код.

Има голямо сходство между .NET CLR и Java Virtual Machine. По предназначение те служат за едно също нещо – да изпълняват код за някакъв виртуален процесор. В .NET това е IL кода, а при Java платформата – т. нар. Java bytecode. Основната разлика между IL и bytecode е, че IL е език от по-високо ниво, а това позволява да бъде компилиран много по-ефективно от Java bytecode.

I. ЗАПОЗНАВАНЕ С АРХИТЕКТУРАТА НА ПЛАТФОРМАТА .NET И .NET FRAMEWORK

Задачи и отговорности на CLR

Отговорностите на CLR включват:

- Изпълнение на IL кода. Реално IL инструкциите, преди да бъдат изпълнени за първи път, се компилират до инструкции за текущия процесор и след това се изпълняват от системния процесор. Този процес на междинно компилиране до машиннозависим (native) код се нарича JIT компилация (Just-In-Time compilation).

- Управление на паметта и ресурсите на приложенията. CLR включва в себе си система за заделяне на памет и система за почистване на неизползваната памет и ресурси (т. нар. garbage collector). Управлението на паметта при .NET приложенията се извършва в голяма степен автоматизирано и в повечето случаи програмистът не трябва да се грижи за освобождаване на заделената памет.

I. ЗАПОЗНАВАНЕ С АРХИТЕКТУРАТА НА ПЛАТФОРМАТА .NET И .NET FRAMEWORK

Задачи и отговорности на CLR

Отговорностите на CLR включват:

- Осигуряване безопасността на типовете. .NET Framework е среда за контролирано изпълнение на програмен код (managed execution environment). Тя не позволява директен достъп до паметта, не позволява директна работа с указатели, не позволява преобразуване от един тип към друг, който не е съвместим с него, не позволява излизане от границите на масив, както и всякакви други опасни операции. По тази причина .NET се нарича управлявана среда – защото тя управлява изпълнението на кода и по този начин предпазва програмите от много досадни проблеми, които възникват при неуправляваните среди.

I. ЗАПОЗНАВАНЕ С АРХИТЕКТУРАТА НА ПЛАТФОРМАТА .NET И .NET FRAMEWORK

Задачи и отговорности на CLR

Отговорностите на CLR включват:

- Управление на сигурността. NET Framework има добре изградена концепция за сигурност на различни нива. От една страна .NET приложенията могат да се изпълняват с различни права. Правата могат да се задават от администраторите чрез т. нар. политики за сигурност. CLR следи дали кодът, който се изпълнява, спазва зададената политика за сигурност и не позволява тя да бъде нарушена. Тази техника се нарича "code access security".

I. ЗАПОЗНАВАНЕ С АРХИТЕКТУРАТА НА ПЛАТФОРМАТА .NET И .NET FRAMEWORK

Задачи и отговорности на CLR

Отговорностите на CLR включват:

- Управление на изключенията. NET Framework е изцяло обектно-ориентирана среда за разработка и изпълнение на програмен код. В нея механизмът на изключенията е залегнал като основно средство за управление на грешки и непредвидени ситуации. Една от задачите на CLR е да се грижи за изключенията, които възникват по време на изпълнение на кода. При настъпване на изключение CLR има грижата да намери съответния обработчик и да му предостави управлението.

- Управление на конкурентността. CLR контролира паралелното изпълнението на нишки (threads) като за целта си взаимодейства с операционната система.

I. ЗАПОЗНАВАНЕ С АРХИТЕКТУРАТА НА ПЛАТФОРМАТА .NET И .NET FRAMEWORK

Задачи и отговорности на CLR

Отговорностите на CLR включват:

- Взаимодействие с неуправляван код. CLR осигурява връзка между управляван (.NET) код и неуправляван (Win32) код. За целта той изпълнява доста сложни задачи, свързани с конвертиране на данни, синхронизация, прехвърляне на извиквания, взаимодействие с компонентния модел на Windows (COM) и много други.

- Подпомагане процесите на дебъгване (debugging) и оптимизиране (profiling) на управлявания код. CLR осигурява инфра-структура и средства за реализацията на дебъгване и оптимизиране на кода от външни специализирани програми.

I. ЗАПОЗНАВАНЕ С АРХИТЕКТУРАТА НА ПЛАТФОРМАТА .NET И .NET FRAMEWORK

Управляван код (managed code) - е кодът, който се изпълнява от CLR. Той представлява поредица от IL инструкции, които се получават при компилацията на .NET езиците. По време на изпълнение управляваният код се компилира допълнително до машиннозависим код за текущата платформа и след това се изпълнява директно от процесора.

Ралика между управляван и неуправляван код - Управляваният код е машиннонезависим, т. е. може да работи на различни хардуерни архитектури, процесори и операционни системи, стига за тях да има имплементация на CLR. Неуправляваният код е машиннозависим, компилиран за определена хардуерна архитектура и определен процесор. Например програмите, написани на езика C, се компилират до неуправляван код за определена архитектура.

I. ЗАПОЗНАВАНЕ С АРХИТЕКТУРАТА НА ПЛАТФОРМАТА .NET И .NET FRAMEWORK

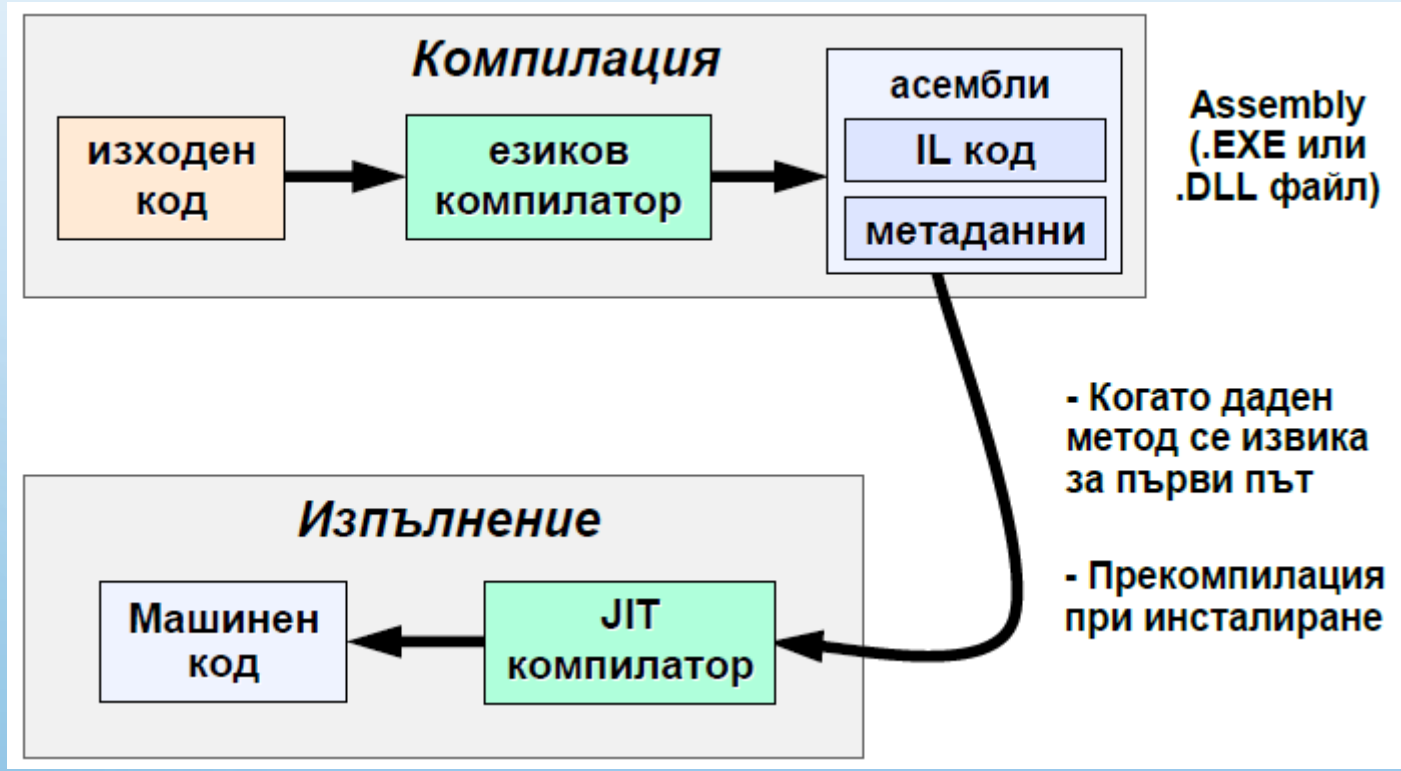
I.4. Компилация и изпълнение

В тази точка се разгледа процеса на компилиране и изпълнение на .NET приложенията. Дадени са пояснения как се извършва компилирането на програми от високо ниво, как се получава IL код, как този код се записва в специален файлов формат (асемблi) и как след това компилираните асемблита се изпълняват от CLR като се компилират междувременно до машинен код от JIT компилатора.

.NET езиците (C#, VB.NET и т. н.) се компилират до междинния език (IL), а след това полученият код се изпълнява от CLR.

I. ЗАПОЗНАВАНЕ С АРХИТЕКТУРАТА НА ПЛАТФОРМАТА .NET И .NET FRAMEWORK

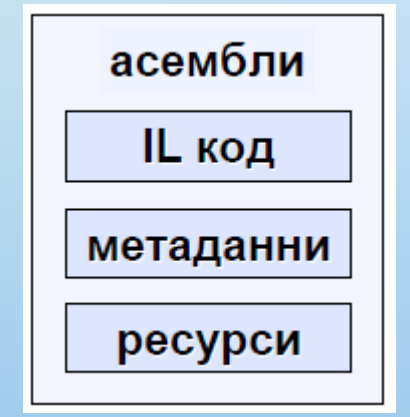
1



Изходният код на .NET програмите може да е написан на предпочитания от нас .NET език, например C#, VB.NET, Managed C++ или друг. За да го компилираме до IL управляван код, използваме компилатора за съответния език. В резултат получаваме асембли.

I. ЗАПОЗНАВАНЕ С АРХИТЕКТУРАТА НА ПЛАТФОРМАТА .NET И .NET FRAMEWORK

Асемблита – това са най-малката самостоятелна градивна единица в .NET Framework. Те представляват наследници на файлови формати .exe и .dll файлове и съдържат IL изпълним код, метаданни и ресурси.



Метаданните - описват съдържанието на всяко асембли, за което се отнасят. Метаданните съдържат имената на класовете и типовете в асемблито, информация за членовете на класовете (методи, полета, свойства и други). Едно асембли може да бъде изпълним файл (.exe файл) или динамична библиотека (.dll файл). Изпълнимите файлове съдържат допълнителна информация, която подпомага началното им стартиране (например входна точка на изпълнение).

Ресурси - иконки, картинки, низове и др., необходими на приложението. Ресурсите могат да се пакетират във файла на асемблито, заедно с изпълнимия код и могат да се извличат от него по време на изпълнение. Тази секция не е задължителна

II. ПОМОЩНИ ИНСТРУМЕНТИ ЗА .NET РАЗРАБОТЧИЦИ

Съществуват редица инструменти, използвани в разработката на .NET приложения. С тяхна помощ можем значително да улесним изпълнението на някои често срещани програмистки задачи. В настоящата точка са изброени инструменти, които ни помагат да разработваме по-качествени решения по-бързо, като могат значително да ни улеснят в писането на код и в поддръжката му. Всички средства, които са споменати, са отлично допълнение към интегрираните среди за .NET разработка.

Списък с най-често използваните инструменти:

- Изследване на .NET асемблита с **.NET Reflector**
- Анализ на .NET асемблита с **FxCop**
- Генериране на код с **CodeSmith**
- Писане на unit тестове с **NUnit**
- Генериране на лог съобщения с **log4net**
- Работа с релационни бази от данни с **NHibernate**
- Автоматизиране на build процеса с **NAnt**

III. ИНТЕГРИРАНА СРЕДА ЗА РАЗРАБОТКА VISUAL STUDIO .NET.

1. Инсталация на Visual Studio Community

Среда за разработка – Integrated Development Environment (IDE) е редактор за програми, в който пишем програмния код и можем да го компилираме и изпълняваме, да виждаме грешките, да ги поправяме и да стартираме програмата отново.

• За програмиране на C# използваме средата Visual Studio за операционната система Windows или за Linux, или Mac OS X.

• Ако програмираме на Java, подходящи са средите IntelliJ IDEA, Eclipse или NetBeans.

• Ако ще пишем на Python, можем да използваме средата PyCharm.

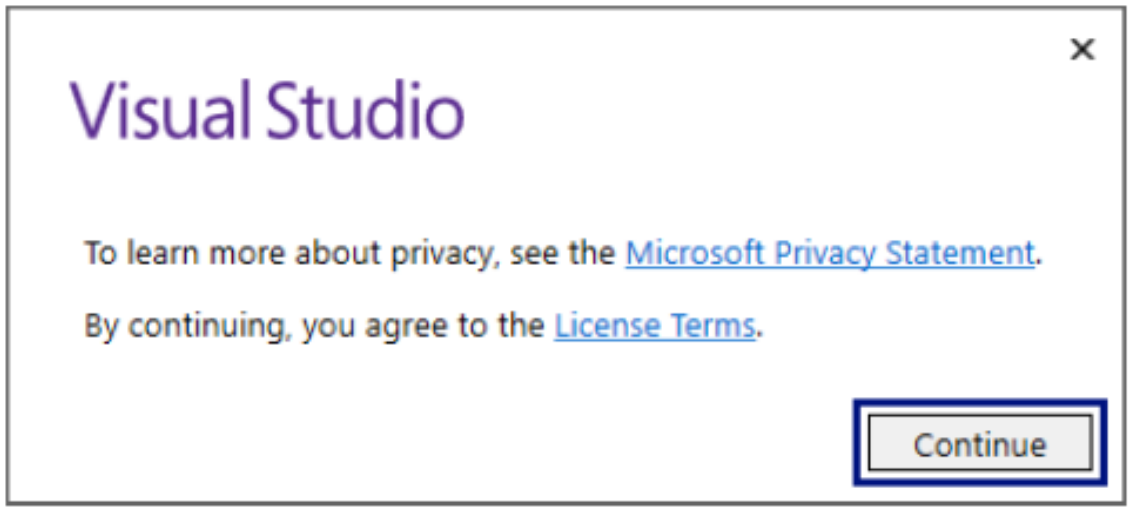
Инсталация на Visual Studio Community

Започваме с инсталацията на интегрираната среда Microsoft Visual Studio Community (версия 2022, актуална към Септември 2022 г.). Community версията на Visual Studio (VS) се разпространява безплатно от Microsoft и може да бъде изтеглена от: https://www.visualstudio.com/vs/community

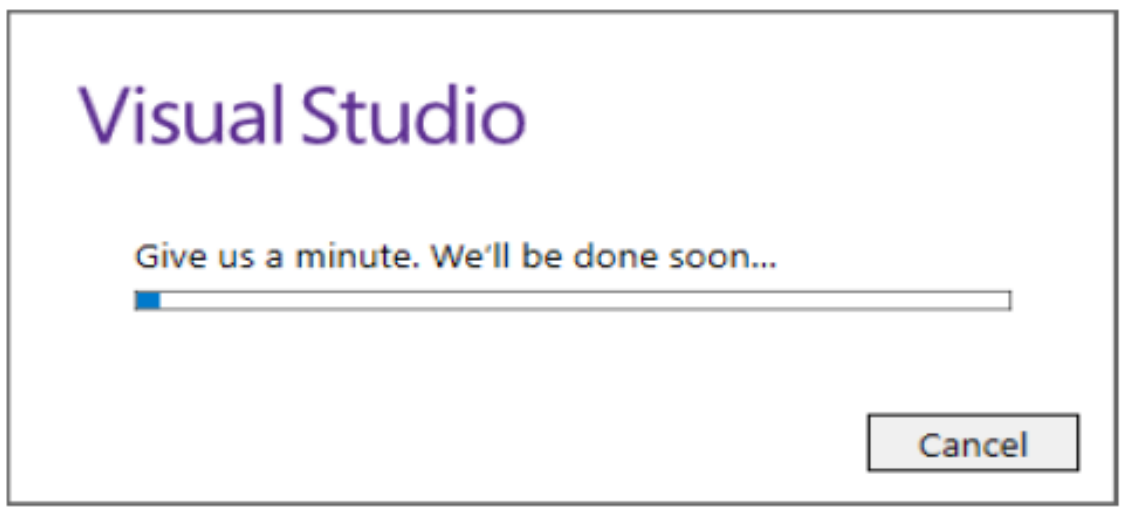
III. ИНТЕГРИРАНА СРЕДА ЗА РАЗРАБОТКА VISUAL STUDIO .NET.

Инсталацията е типичната за Windows с [Next], [Next] и [Finish], но е важно да включим компонентите за "desktop development" и "ASP.NET". Не е необходимо да проме-няме останалите настройки за инсталация.

След като свалим и стартираме инста-лационния файл, се появява този екран:



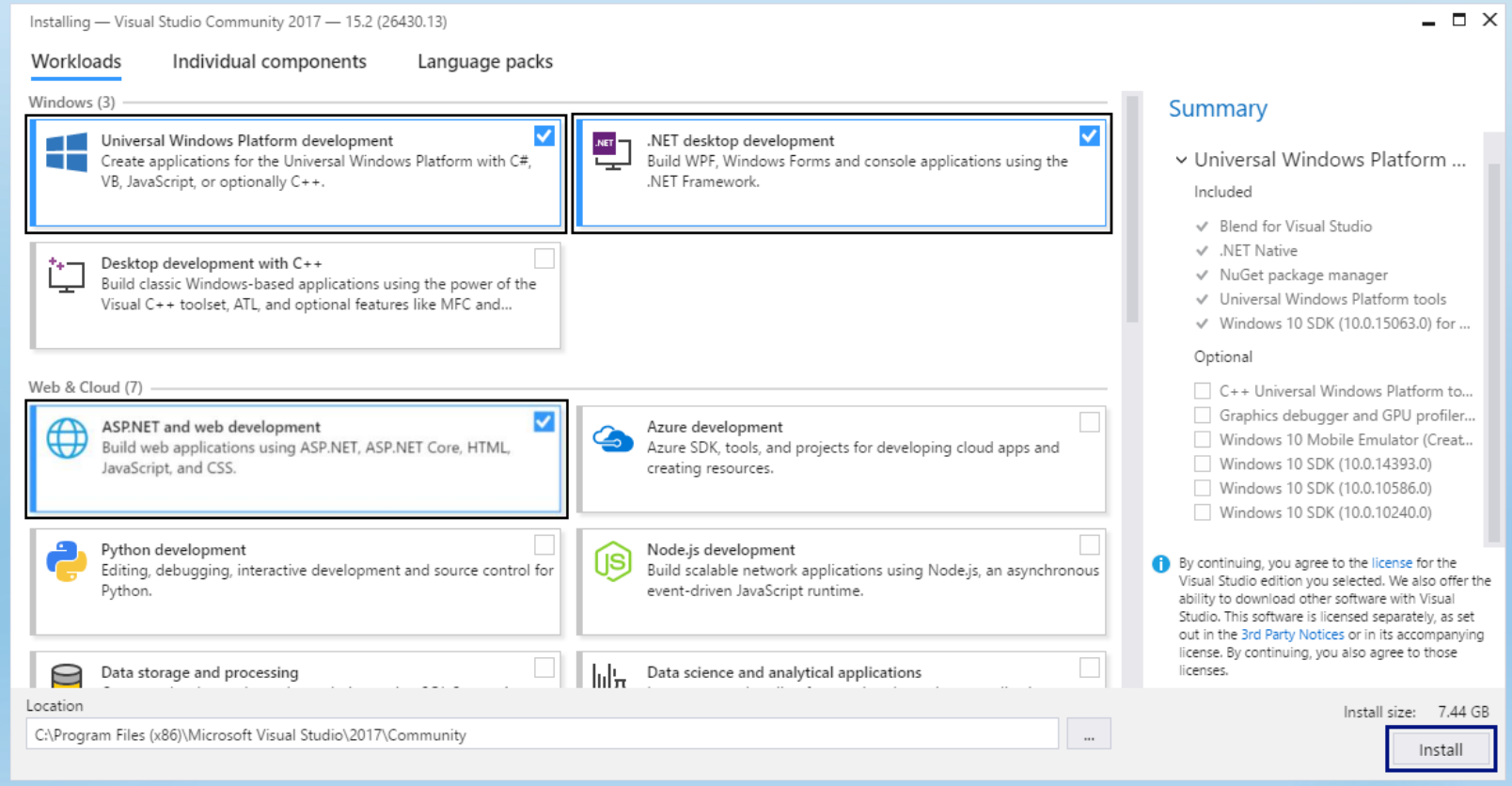
Натискаме бутона [Continue] и ще се появи прозорецът по-долу:



III. ИНТЕГРИРАНА СРЕДА ЗА РАЗРАБОТКА VISUAL STUDIO .NET.

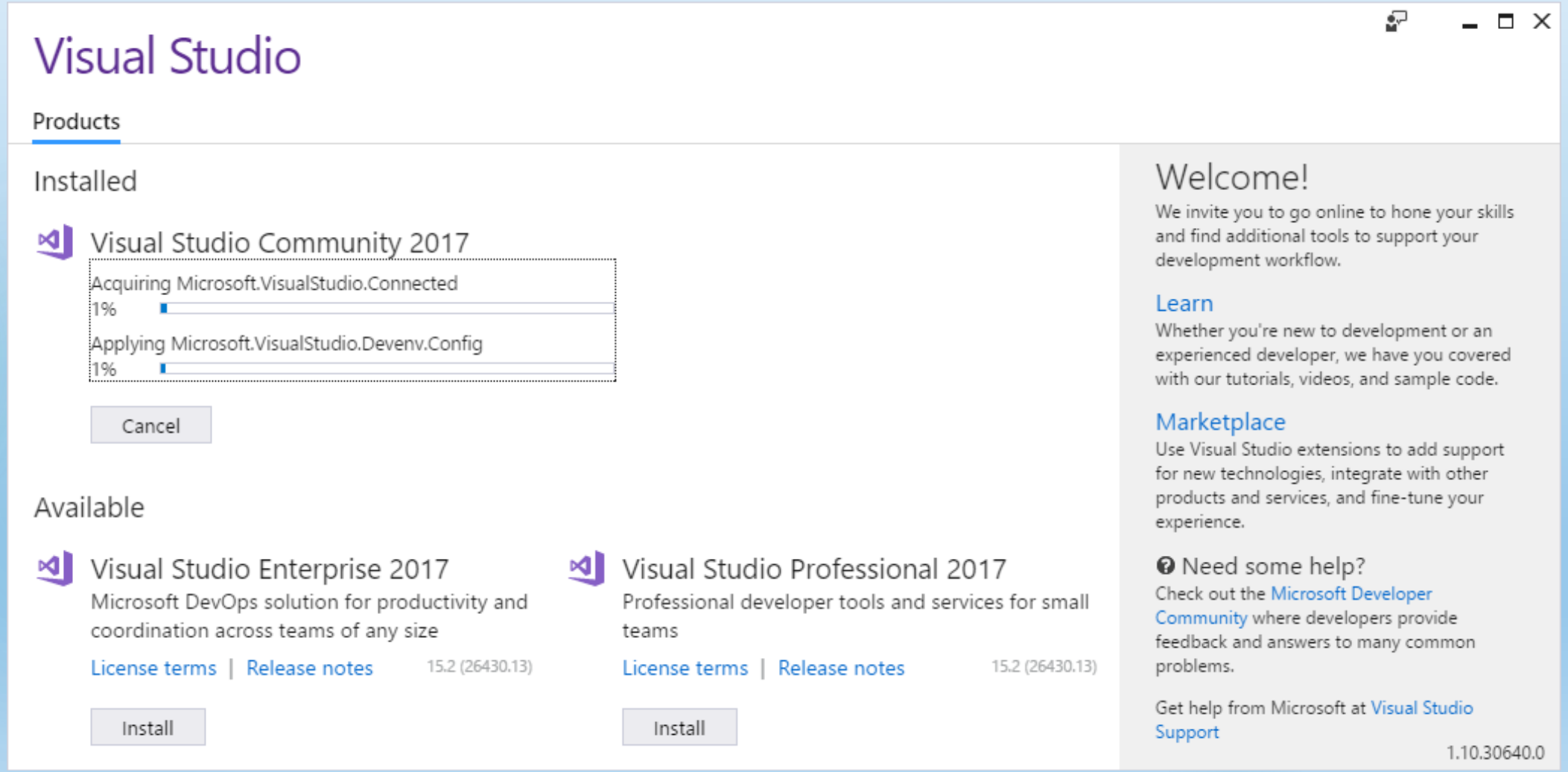
1

Зарежда се прозорец с инсталационния панел на Visual Studio. Слагаме отметка на [.NET desktop development], [ASP.NET and web development] и на [Universal Windows Platform development], след което натискаме бутона [Install]. Общо взето това е всичко.



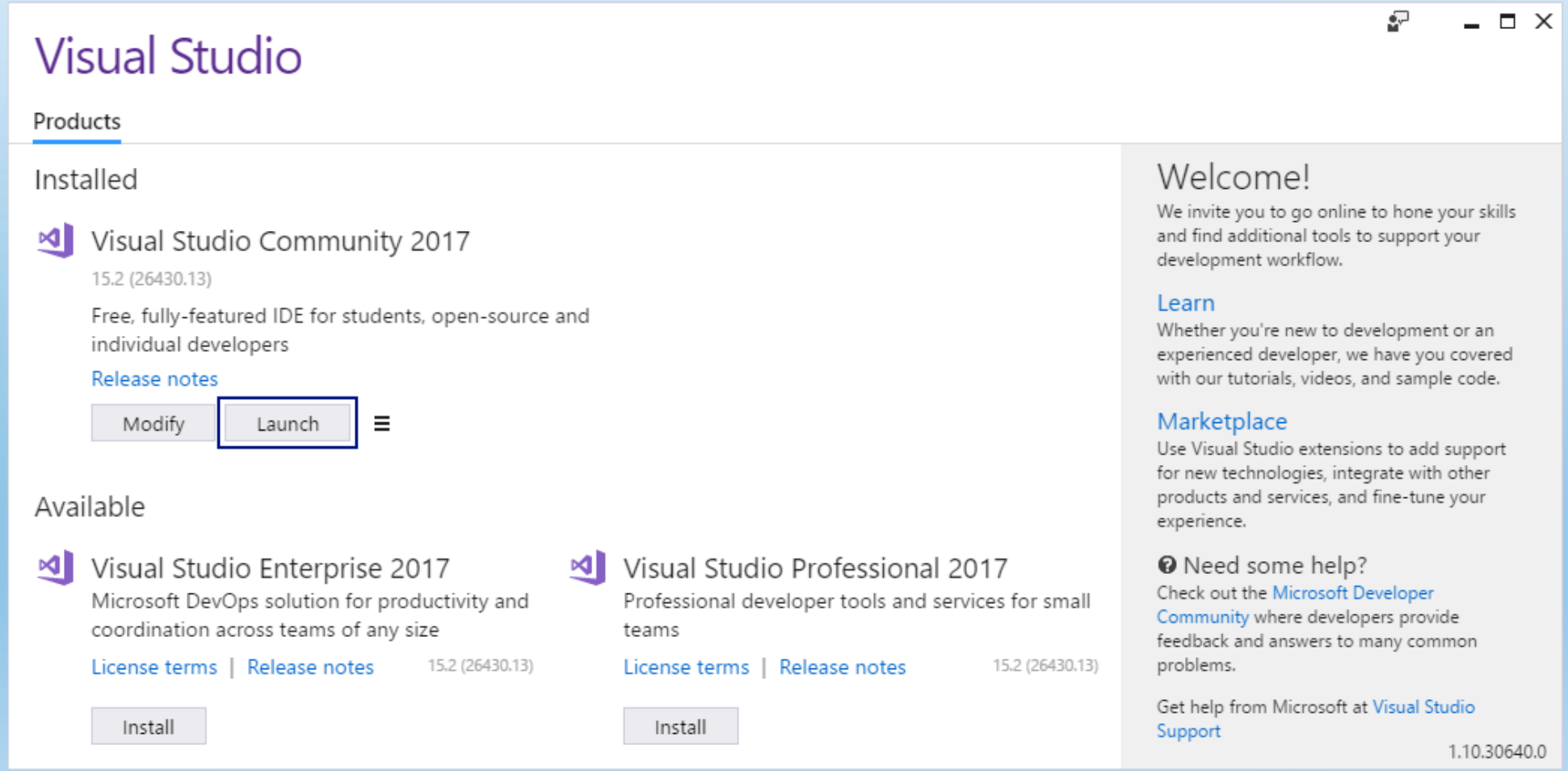
III. ИНТЕГРИРАНА СРЕДА ЗА РАЗРАБОТКА VISUAL STUDIO .NET.

Започва инсталацията на Visual Studio и се появява екран като този по-долу:



III. ИНТЕГРИРАНА СРЕДА ЗА РАЗРАБОТКА VISUAL STUDIO .NET.

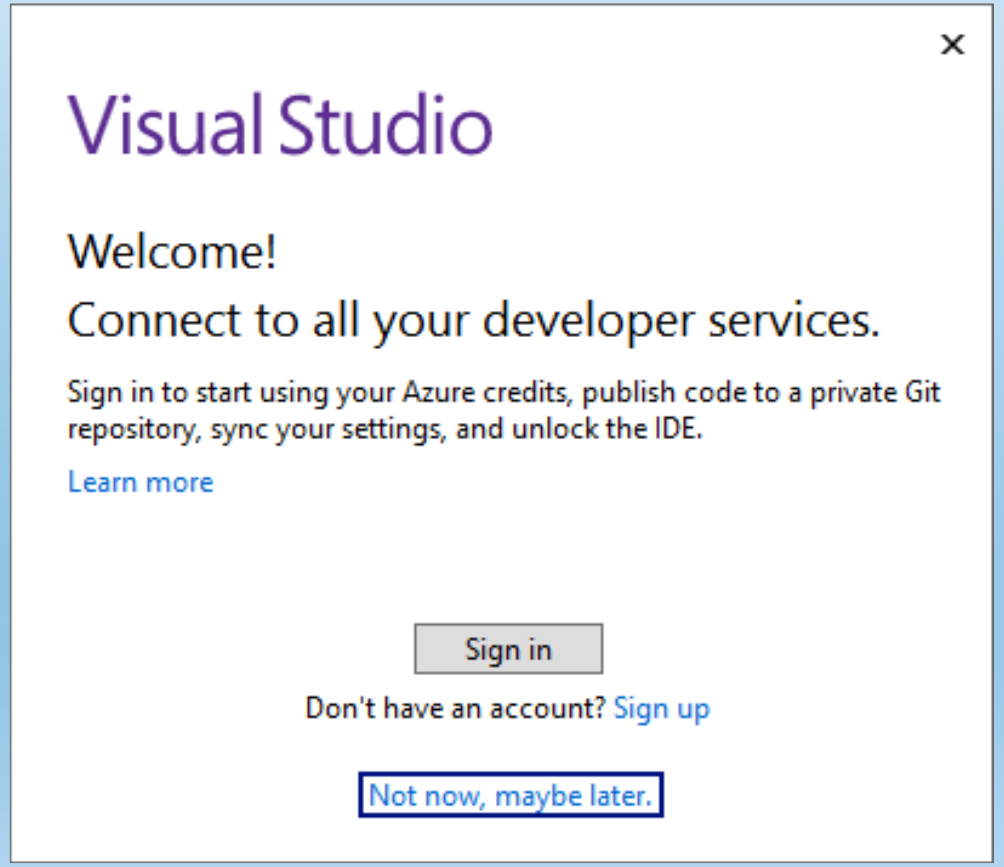
След като Visual Studio се инсталира, ще се появи информативен екран и трябва да натиснем бутона [**Launch**], за да го стартираме.

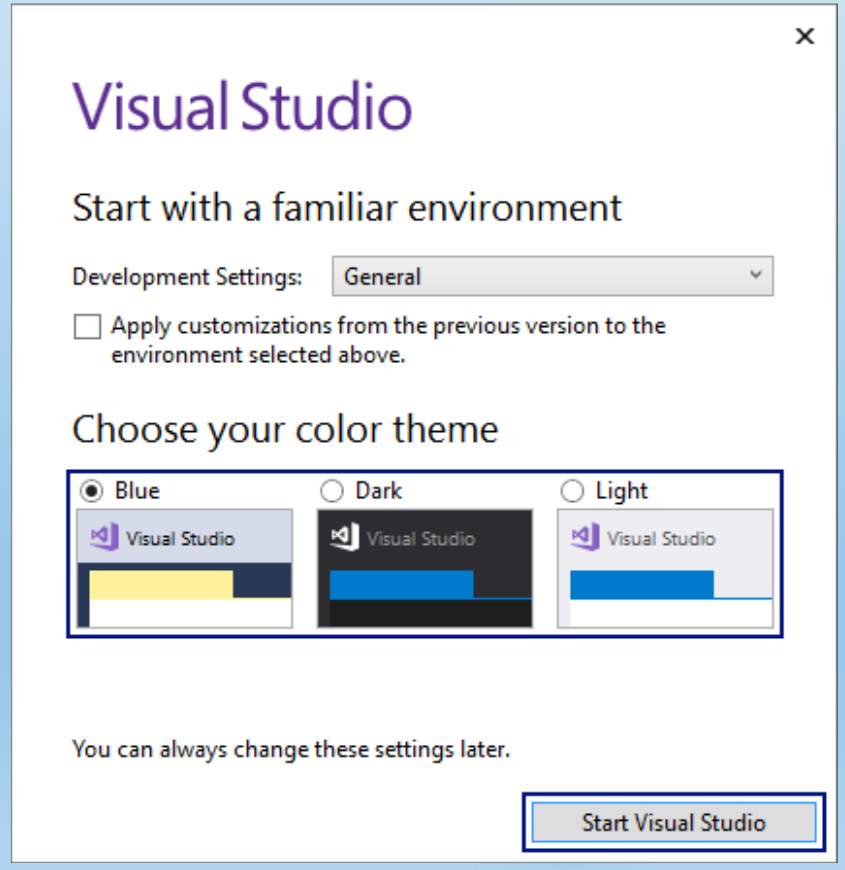


III. ИНТЕГРИРАНА СРЕДА ЗА РАЗРАБОТКА VISUAL STUDIO .NET.

1

След старта на VS излиза екран като този по-долу. От него можем да изберем дали да влезем с Microsoft профила си във Visual Studio. За момента избираме да работим без да сме се логнали с Microsoft акаунта си, затова избираме опцията [**Not now, maybe later**.].

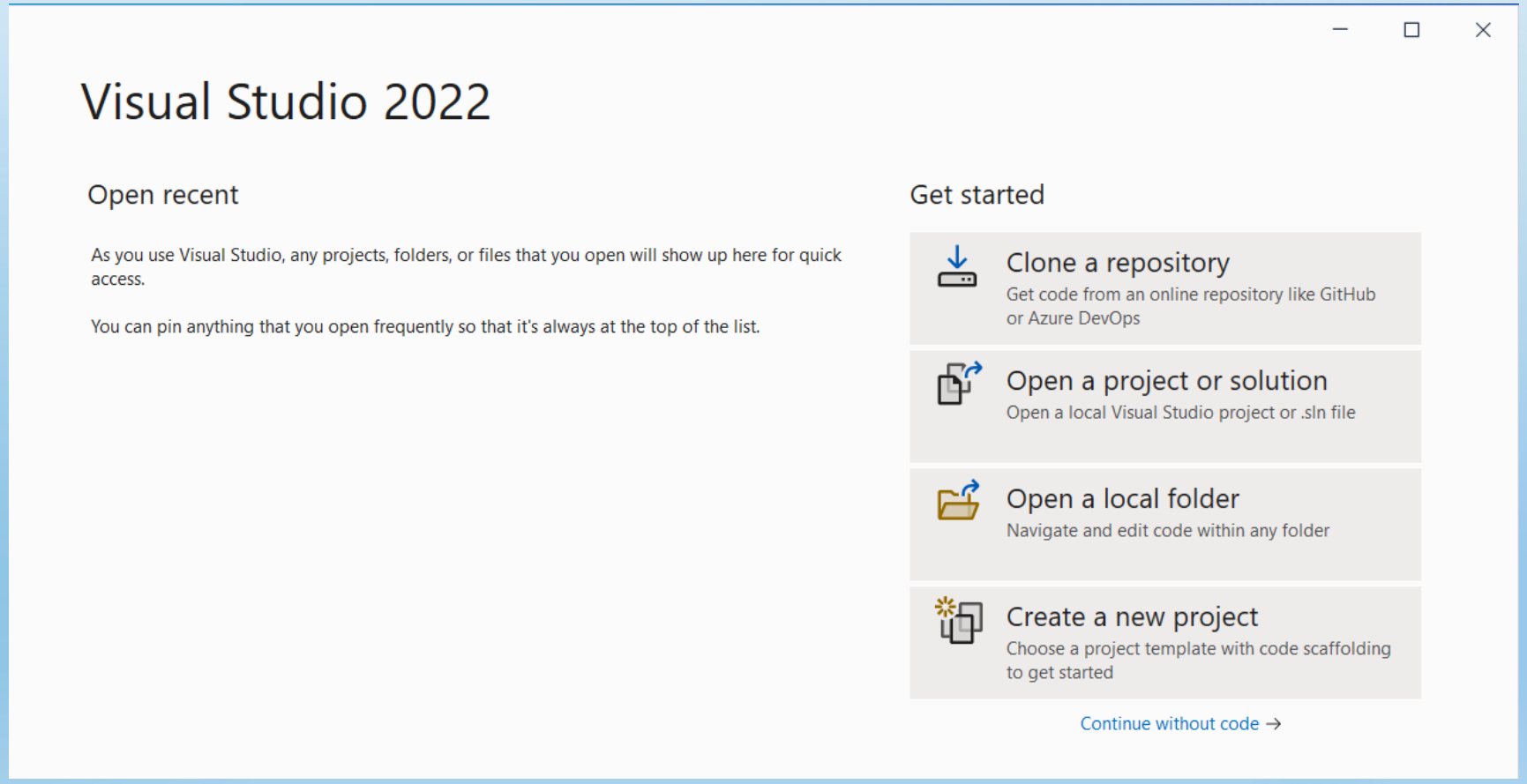




III. ИНТЕГРИРАНА СРЕДА ЗА РАЗРАБОТКА VISUAL STUDIO .NET.

1

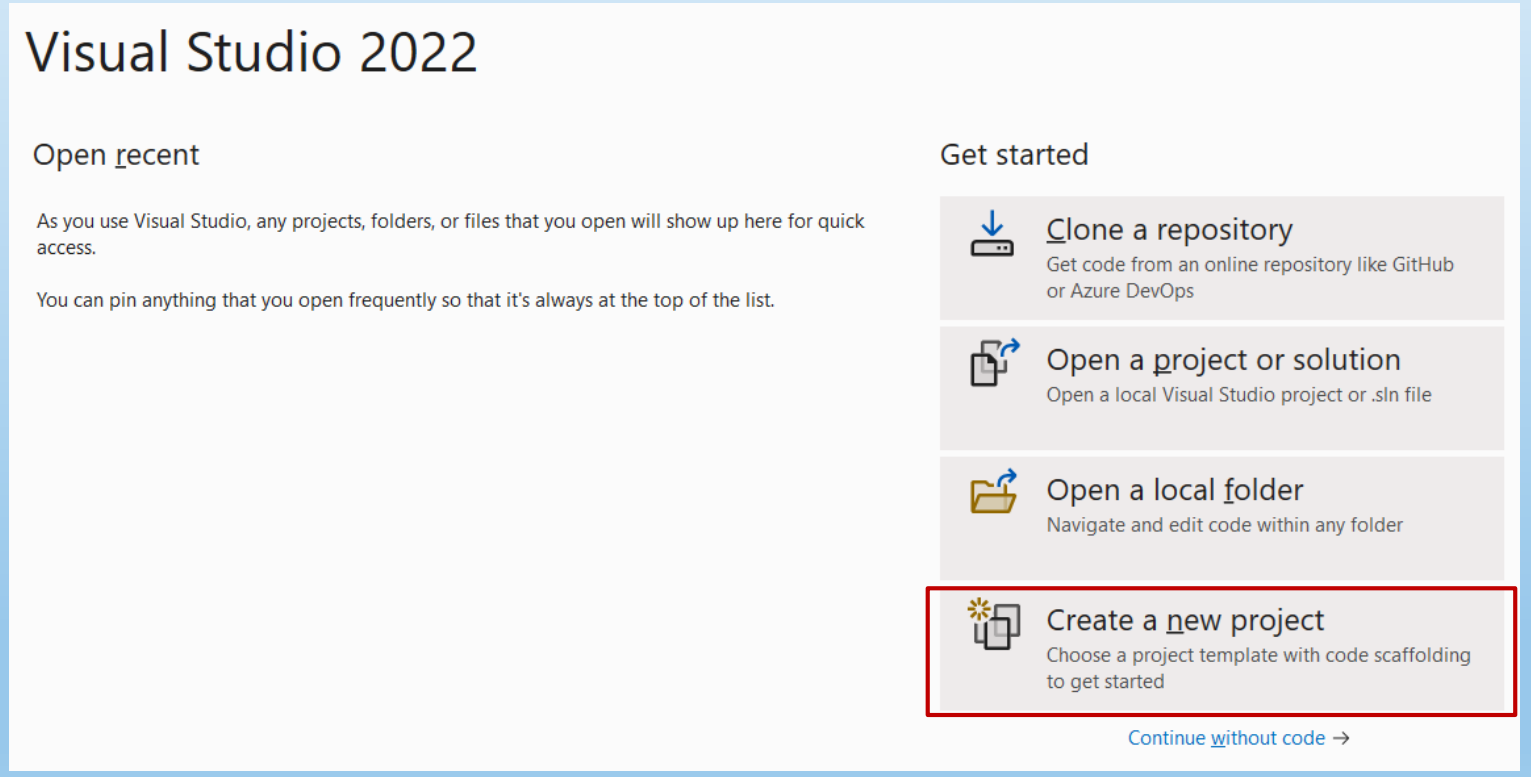
Натискаме бутона [Start Visual Studio] и се зарежда в началния изглед на Visual Studio Community:



III. ИНТЕГРИРАНА СРЕДА ЗА РАЗРАБОТКА VISUAL STUDIO .NET.

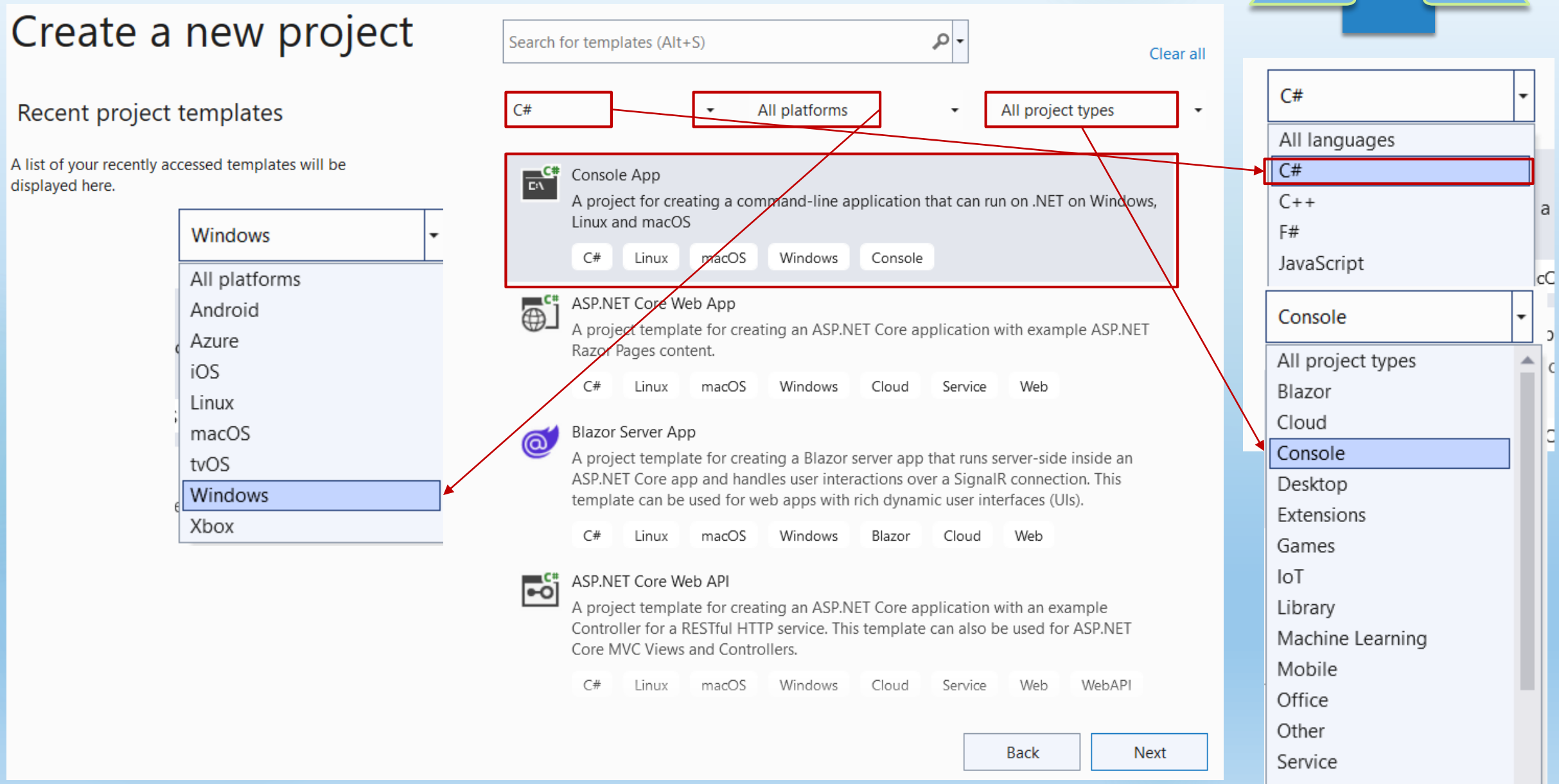
2. Създаване на нов проект с Visual Studio

След успешна инсталация и стартиране на Visual Studio Community 2022 се появява екрана показан по-долу, от който можем да създадем нов проект или да заредим вече съществуващ такъв. За да създадем нов проект е необходимо да се избере опцията “Create new project”.



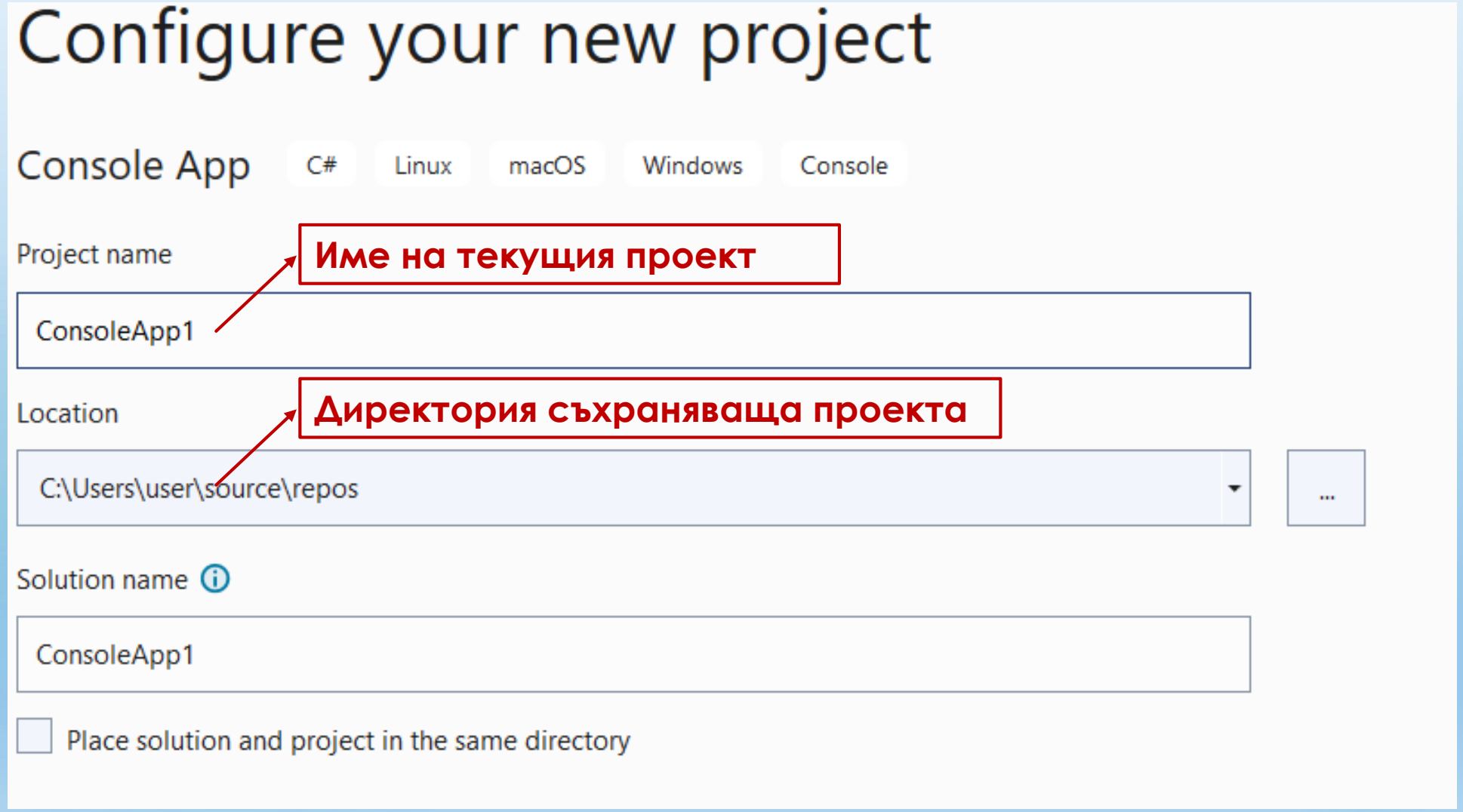
III. ИНТЕГРИРАНА СРЕДА ЗА РАЗРАБОТКА VISUAL STUDIO .NET.

1

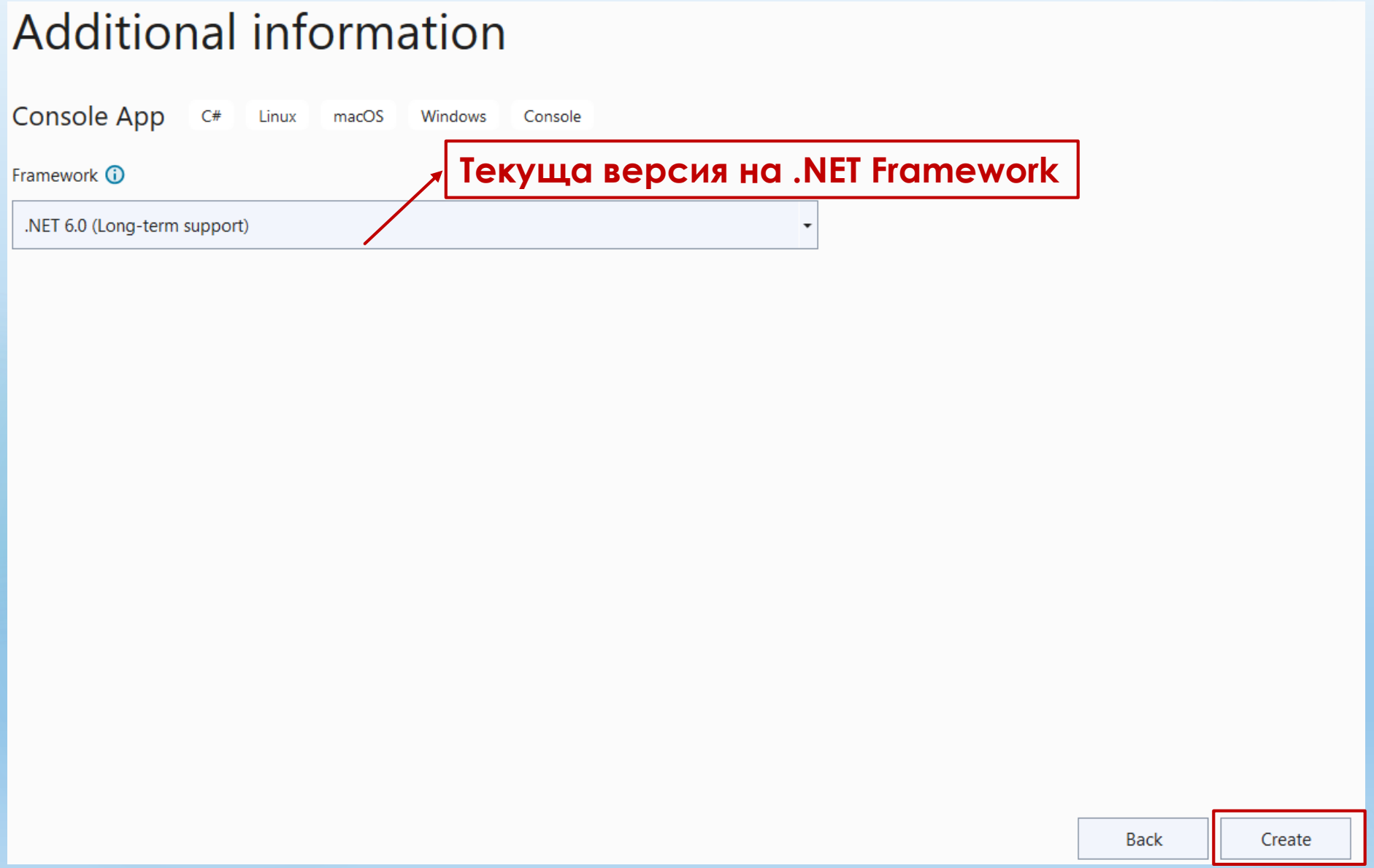


III. ИНТЕГРИРАНА СРЕДА ЗА РАЗРАБОТКА VISUAL STUDIO .NET.

1

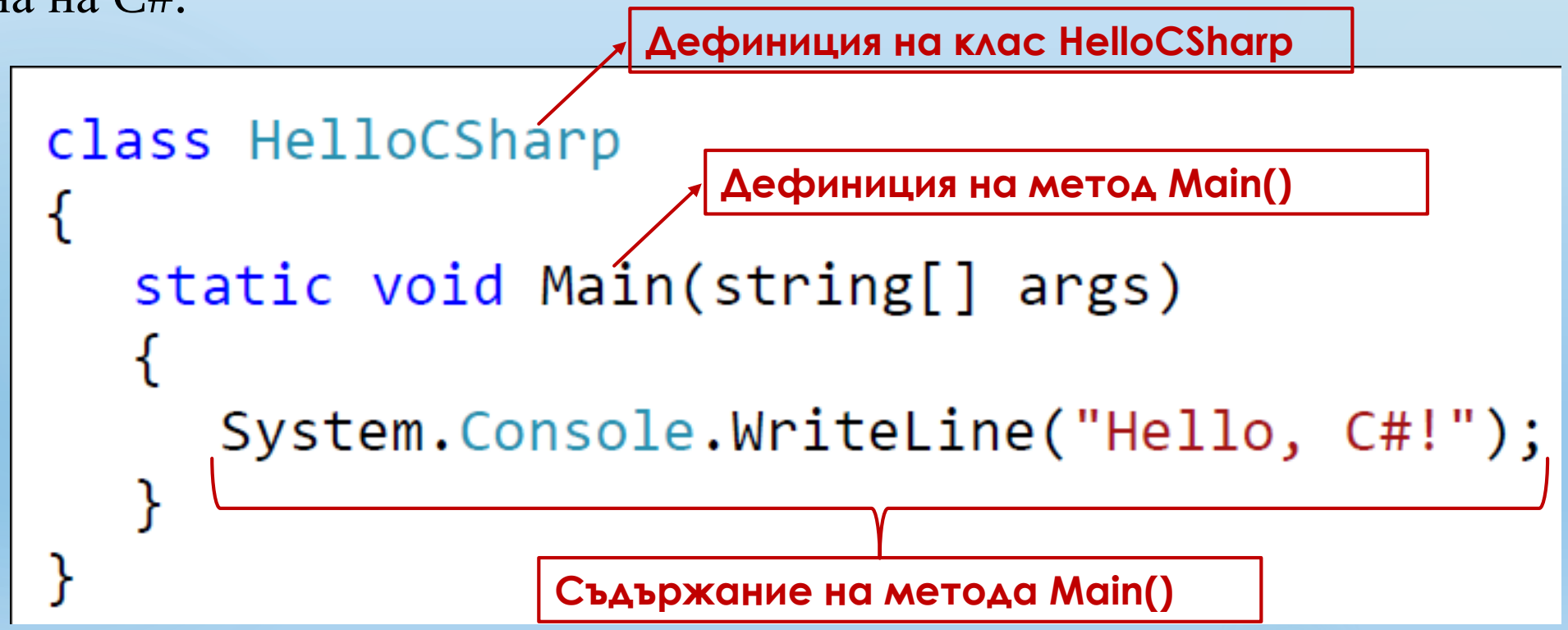


III. ИНТЕГРИРАНА СРЕДА ЗА РАЗРАБОТКА VISUAL STUDIO .NET.



3. Първа C# програма. Компилиране и изпълнение

Преди да преминем към подробно описание на езика C# и на .NET платформата, нека да се запознаем с прост пример на това какво представлява една програма, написана на C#:



III. ИНТЕГРИРАНА СРЕДА ЗА РАЗРАБОТКА VISUAL STUDIO .NET.

C# различава главни от малки букви!

Внимавайте, докато пишете! Изписването на един и същ текст с главни, малки букви или смесено в C# означава различни неща. Да напишем `Class` е различно от `class` и да напишем `System.Console` е различно от `SYSTEM.CONSOLE`.

Това правило важи за всички конструкции в кода – ключови думи, имена на променливи, имена на класове и т.н.

Програмният код трябва да е правилно форматиран!

Форматирането представлява добавяне на символи, несъществени за компилатора, като интервали, табулации и нови редове, които структури-рат логически програмата и улесняват четенето й.

III. ИНТЕГРИРАНА СРЕДА ЗА РАЗРАБОТКА VISUAL STUDIO .NET.

Недобре форматиран код:

```
class HelloCSharp
{
static void Main()
{
System.Console.WriteLine("Hello, C#!");
}
}
```

```
                                        class
    HelloCSharp
{
    static void                         Main()
    {                           System          .
Console.WriteLine("Hello, C#!")     ;}          }
```

```
class HelloCSharp{static void Main(){System.Console.WriteLine(
"Hello, C#!");}}
```

Горните примери **ще се компилират и изпълнят** по абсолютно същия начин като форматирания, **но са далеч по-нечетливи, трудни за разбиране и осмисляне** и съответно неудобни за промяна.

Не допускайте програмите ви да съдържат неформатиран код! Това силно намалява четимостта и довежда до трудно модифициране на кода.

III. ИНТЕГРИРАНА СРЕДА ЗА РАЗРАБОТКА VISUAL STUDIO .NET.

Основни правила на форматирането:

- Методите се отместват по-навътре от дефиницията на класа;

- Съдържанието на методите се отмества по-навътре от дефиницията на метода;

- Отварящата фигурна скоба { трябва да е сама на ред и да е разположена точно под метода или класа, към който се отнася;

- Затварящата фигурна скоба } трябва да е сама на ред и да е поставена вертикално точно под съответната й отваряща скоба (със същото отместване като нея);

- Имената на класовете трябва да започват с главна буква;

- Имената на променливите трябва да започват с малка буква;

- Имената на методите трябва да започват с главна буква;