



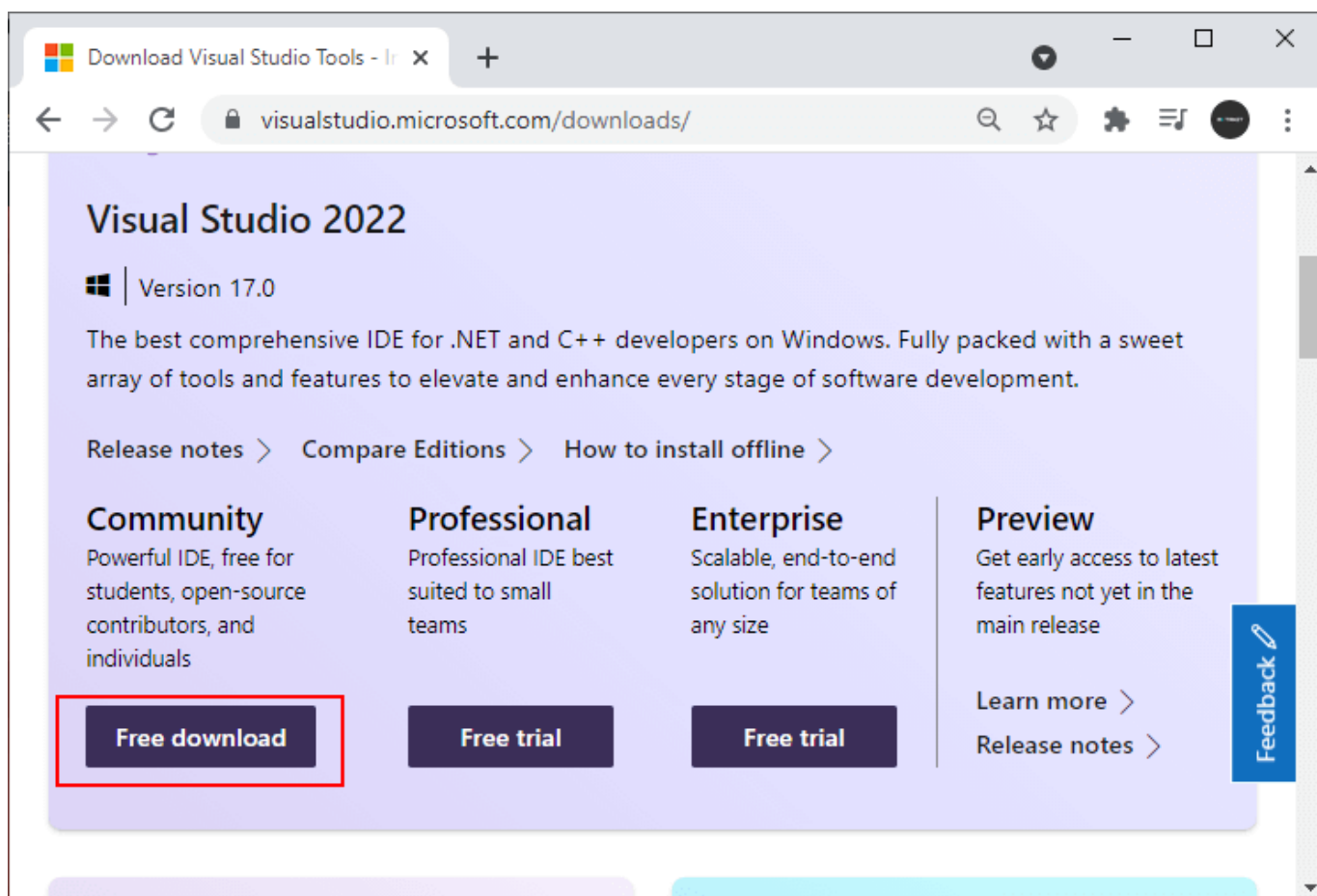
Начало работы. Visual Studio

Последнее обновление: 15.11.2021



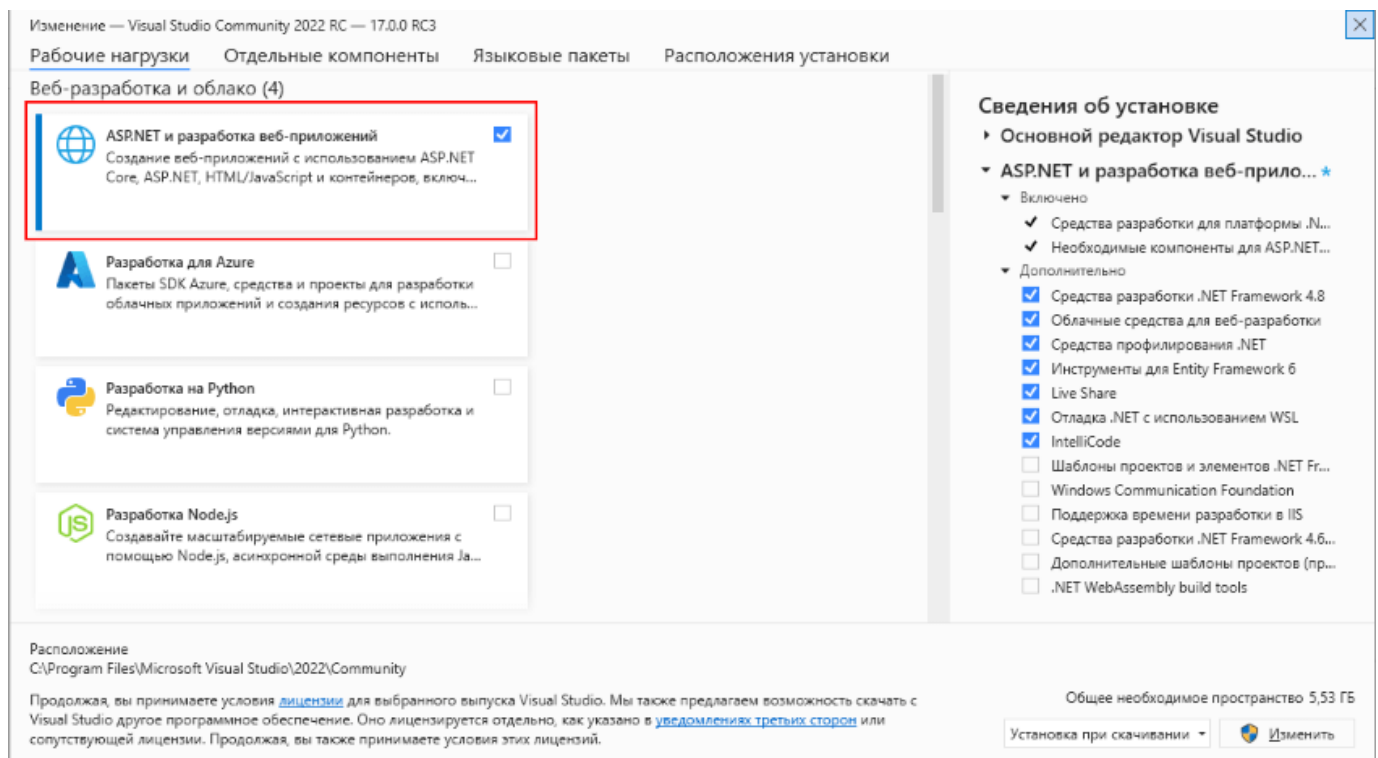
Создадим первое приложение на языке Visual Basic .NET. Что потребуется для работы с VB.NET? В первую очередь нам нужен текстовый редактор, чтобы набирать код программы, и компилятор, который позволит скомпилировать написанный нами код в программу. Ну и конечно нам нужен фреймворк .NET, который требуется для выполнения программы. Для облегчения написания программного кода обычно используют специальные среды разработки, которые предоставляют многообразие различных возможностей по созданию программ.

Для создания приложений на VB.NET будем использовать бесплатную и полнофункциональную среду разработки - Visual Studio Community 2022, которую можно загрузить по следующему адресу: [Microsoft Visual Studio 2022](https://visualstudio.microsoft.com/downloads/)



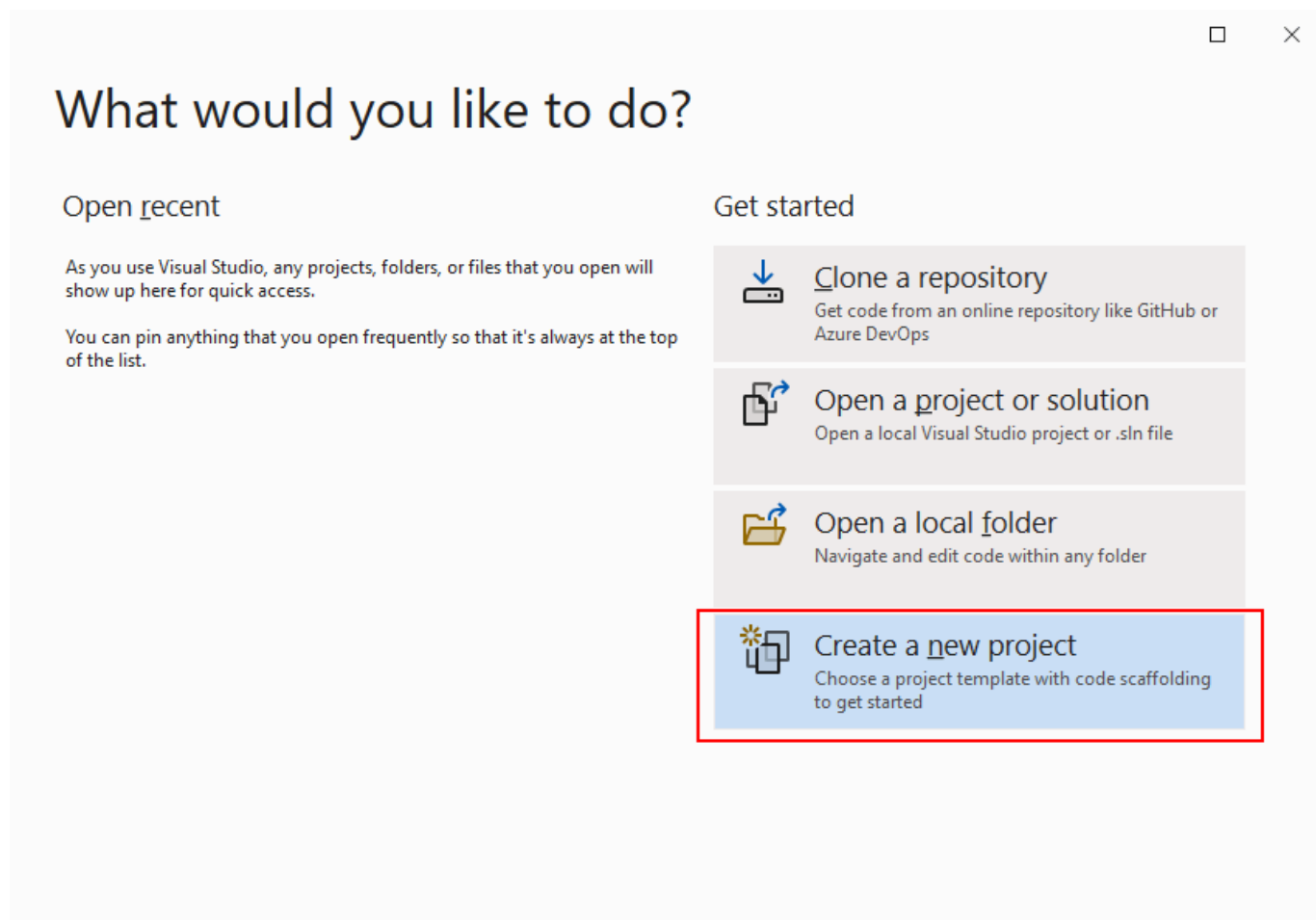
После загрузки запустим программу установщика. В открывшемся окне нам будет предложено выбрать те компоненты, которые мы хотим установить вместе Visual Studio. Стоит отметить, что Visual Studio – очень функциональная среда разработки и позволяет разрабатывать приложения с помощью множества языков и платформ. В нашем случае нам будет интересно прежде всего VB.NET и .NET.

Чтобы добавить в Visual Studio поддержку проектов для VB.NET и .NET 6, в программе установки среди рабочих нагрузок можно выбрать только пункт **ASP.NET и разработка веб-приложений**. Можно выбрать и больше опций или вообще все опции, однако стоит учитывать свободный размер на жестком диске – чем больше опций будет выбрано, соответственно тем больше места на диске будет занято.

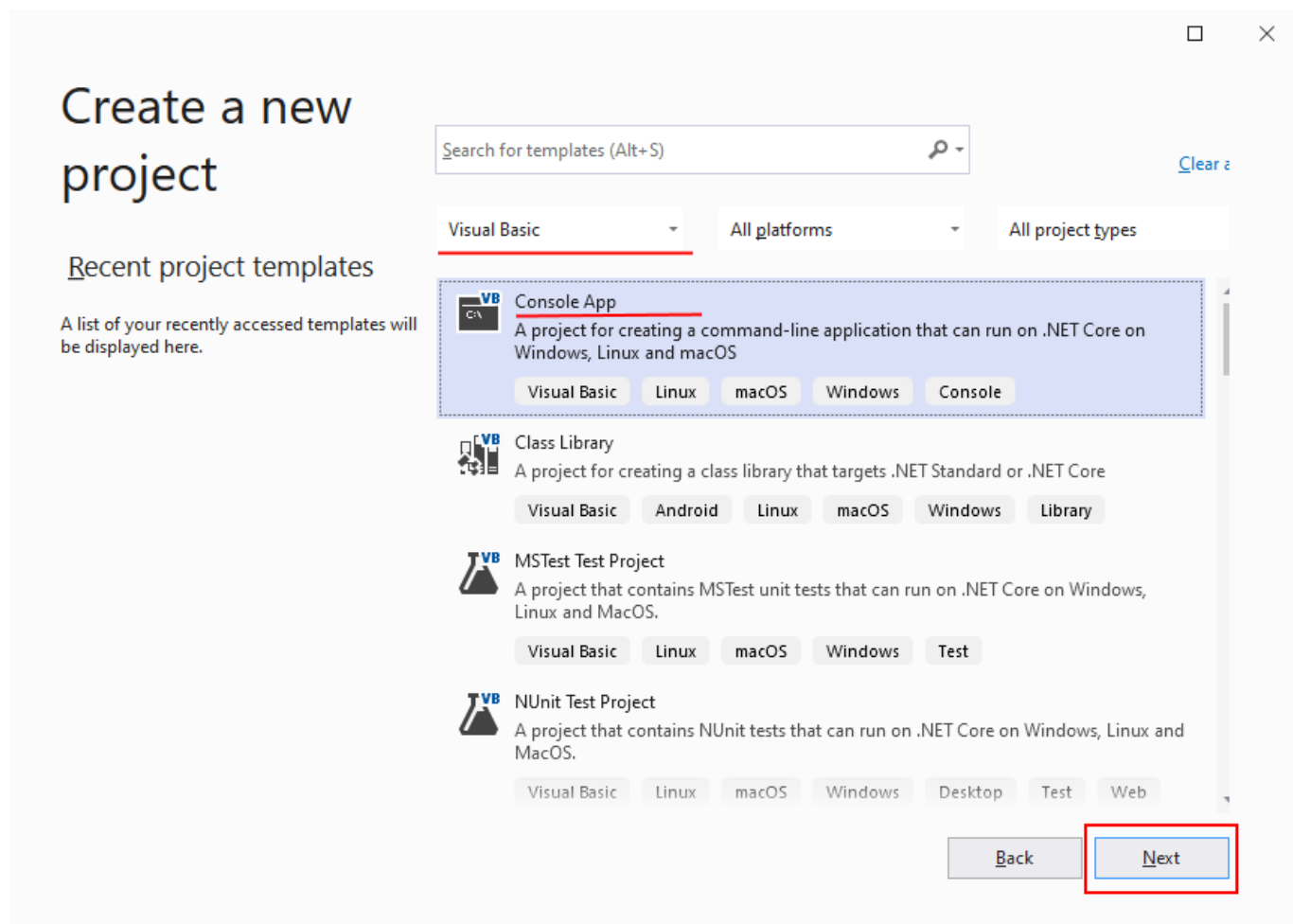


И при инсталляции Visual Studio на ваш компьютер будут установлены все необходимые инструменты для разработки программ, в том числе фреймворк .NET 6.

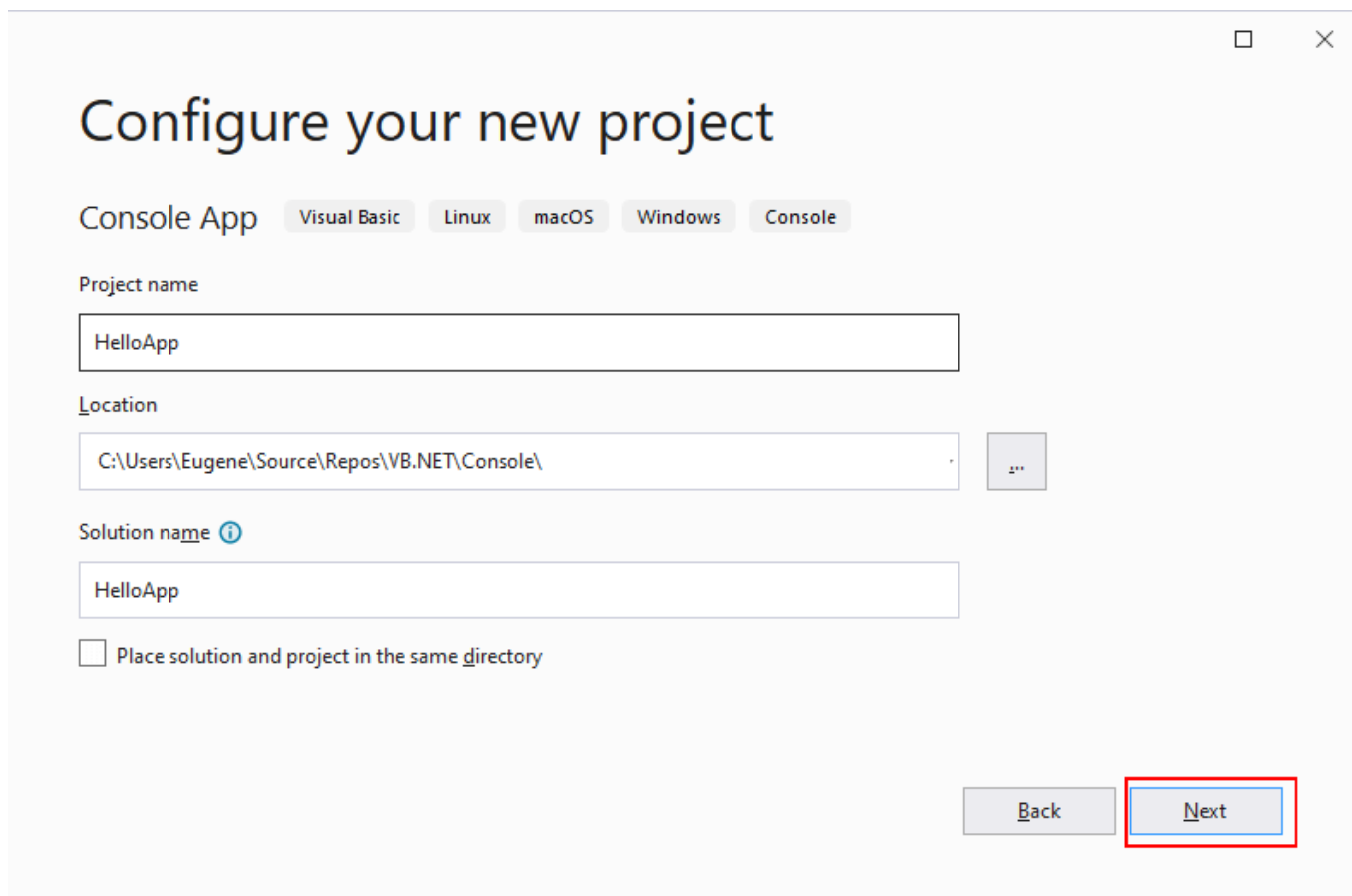
После завершения установки создадим первую программу. Она будет простенькой. Вначале откроем Visual Studio. На стартовом экране выберем **Create a new project** (Создать новый проект)



На следующем окне в качестве типа проекта выберем **Console App**, то есть мы будем создавать консольное приложение на языке VB.NET



Далее на следующем этапе нам будет предложено указать имя проекта и каталог, где будет располагаться проект.



Configure your new project

Console App Visual Basic Linux macOS Windows Console

Project name

HelloApp

Location

C:\Users\Eugene\Source\Repos\VB.NET\Console\

Solution name ⓘ

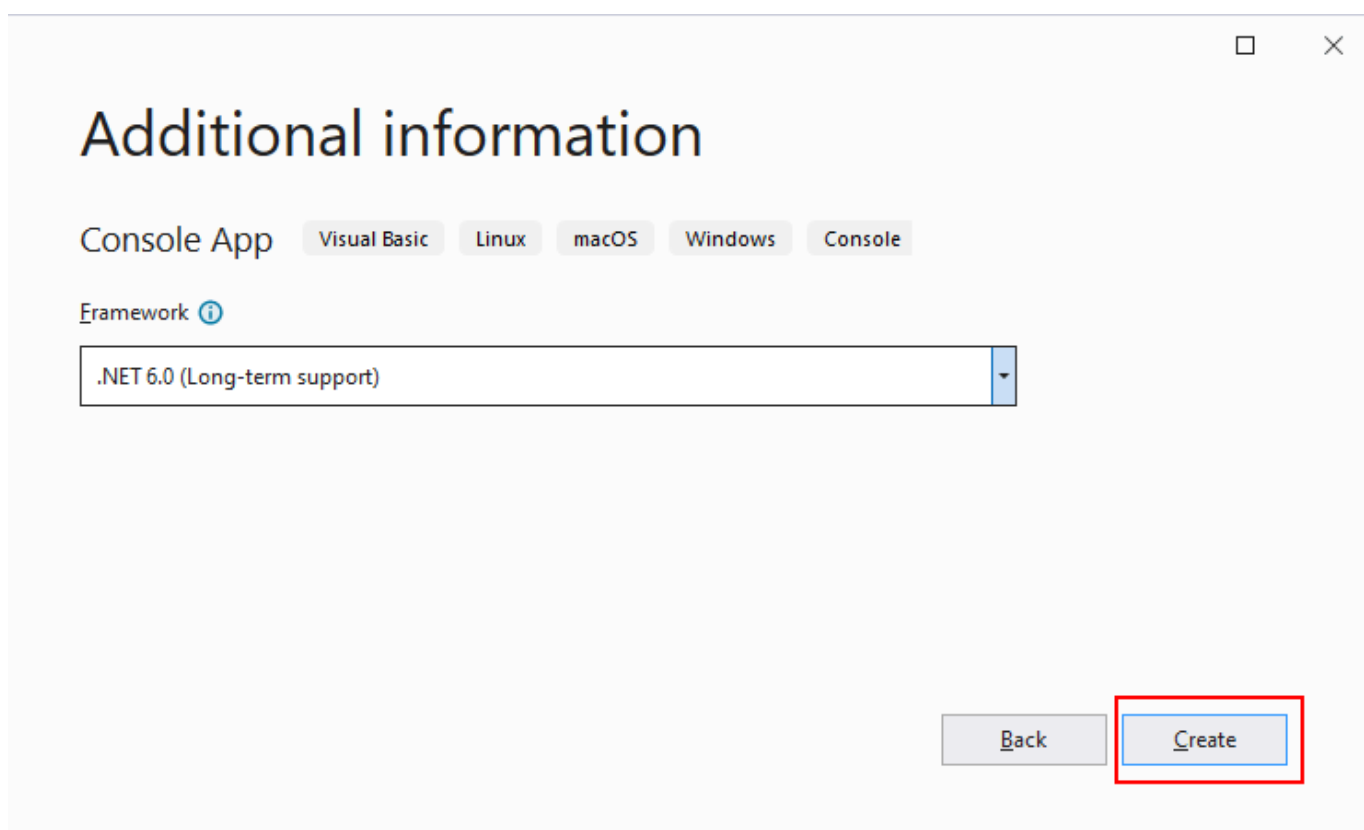
HelloApp

☐ Place solution and project in the same directory

Back Next

В поле **Project Name** дадим проекту какое-либо название. В моем случае это **HelloApp**.

На следующем окне Visual Studio предложит нам выбрать версию .NET, которая будет использоваться для проекта. По умолчанию здесь выбрана последняя на данный момент версия - .NET 6.0. Оставим и нажмем на кнопку Create (Создать) для создания проекта.



Additional information

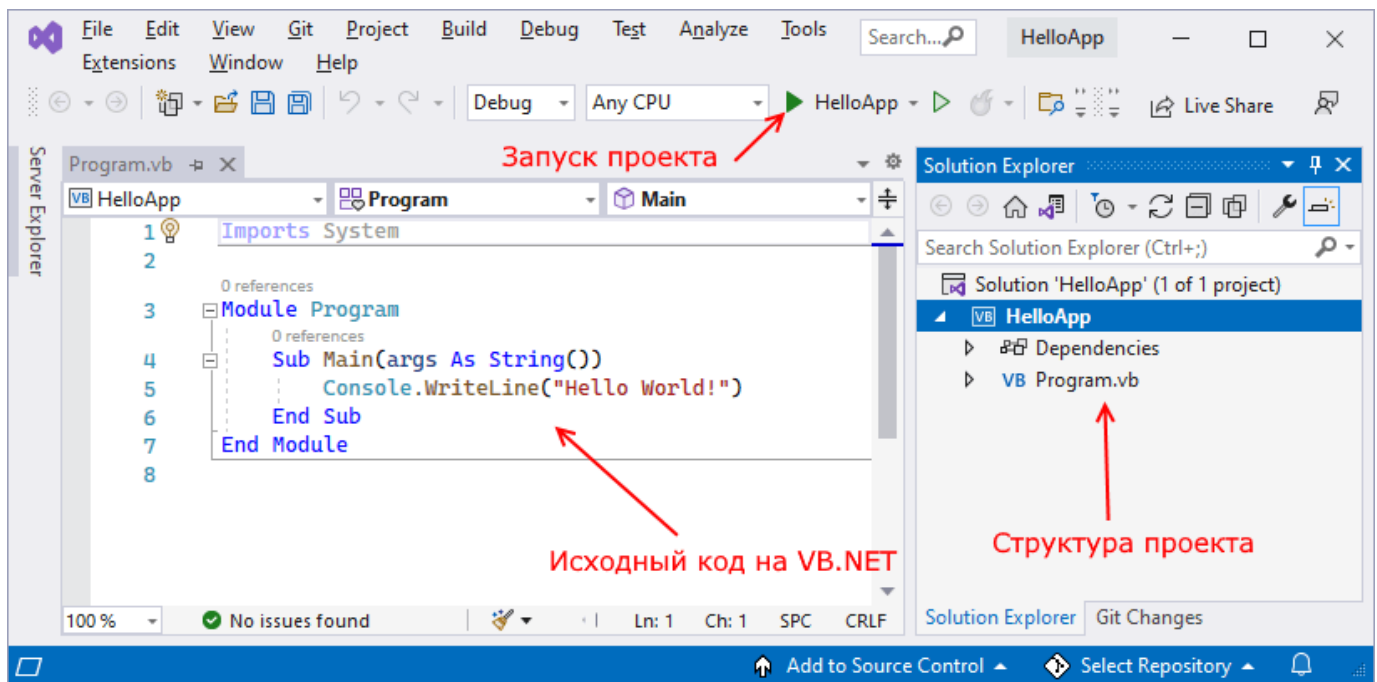
Console App Visual Basic Linux macOS Windows Console

Framework ⓘ

.NET 6.0 (Long-term support)

Back Create

После этого Visual Studio создаст и откроет нам проект:



В большом поле в левой части Visual Studio располагается текстовый редактор с подсветкой, в котором открыт автоматически сгенерированный код файла **Program VB.NET**.

Справа находится окно **Solution Explorer**, в котором можно увидеть структуру нашего проекта. В данном случае у нас сгенерированная по умолчанию структура.

В начале идет узел **Dependencies**. Этот узел содержит сборки dll, которые добавлены в проект по умолчанию. Эти сборки как раз содержат классы библиотеки .NET, которые будет использовать проект VB.NET.

Далее идет непосредственно сам файл кода программы - **Program.vb**. Как раз этот файл и открыт в текстовом редакторе кода.

Рассмотрим код файла **Program.vb**:

```

1 Imports System
2
3 Module Program                                'начало объявления модуля
4     Sub Main(args As String())                'начало объявления метода
5         Console.WriteLine("Hello World!")
6     End Sub                                    'конец объявления метода
7 End Module                                    'конец объявления модуля

```

В начале файла идет директива **Imports**, после которой идет название подключаемого пространства имен. **Пространства имен** представляют собой организацию функциональности в общие блоки. Например, на первой строке

```

1 Imports System

```

подключается пространство имен **System**, которое содержит фундаментальные и базовые классы платформы .NET.

Команда **Imports** позволяет импортировать функциональность из других, подключенных библиотек. В частности, здесь подключается функциональность из пространства имен **System**.

Хотя VB.NET – это объектно-ориентированный язык программирования, от классического Visual Basicа он унаследовал возможность использования модулей. И код программы, генерируемый по умолчанию, представляет модуль.

Объявление модуля начинается со слова **Module**, после которого идет название модуля – в данном случае модуль называется Program:

```
1 Module Program
```

Завершаться объявление модуля должно выражением **End Module**. А весь непосредственный код программы должен располагаться между началом объявления модуля и его завершением.

В модуле по умолчанию определен метод **Main** – это главный метод:

```
1 Sub Main(args As String())  
2     Console.WriteLine("Hello World!")  
3 End Sub
```

Программа может содержать множество методов, но среди них будет один главный – это метод **Main**, через который будут вызываться все остальные методы. И метод с данным названием обязательно должен быть в программе.

Метод **Main** начинается выражением **Sub Main**, где **Sub** указывается, что дальше у нас идет метод, который не возвращает никакого значения. Позже мы подробнее разберем, что все это значит. И затем идет название – то есть **Main**.

Далее в скобках у нас идут параметры метода – **args As String()** – это массив **args**, который хранит значения типа **String**, то есть строки. В данном случае они нам пока не нужны, но в реальной программе это те параметры, которые передаются при запуске программы из консоли.

Метод заканчивается выражением **End Sub**.

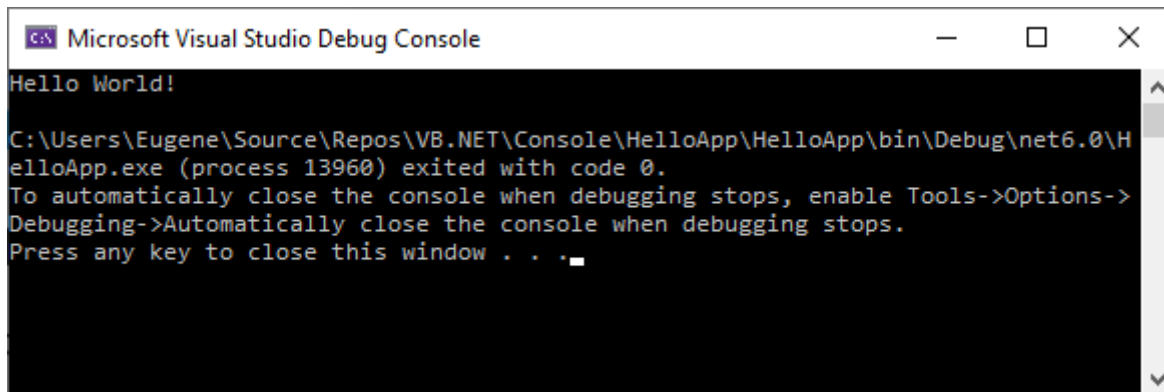
Внутри метода располагаются действия, которые этот метод выполняет:

```
1 Console.WriteLine("Hello World!")
```

По умолчанию здесь выполняется одно действие – вызов встроенного метода **Console.WriteLine()**, который выводит строку на консоль. То есть при выполнении

данной программы на консоль будет выводиться "Hello World!"

Теперь мы можем запустить на выполнение с помощью клавиши F5 или с панели инструментов, нажав на зеленую стрелку. И если вы все сделали правильно, то при запуске приложения на консоль будет выведена строка "Hello World!".



Теперь изменим весь этот код на следующий:

```
1 Module Program
2     Sub Main()
3         Console.Write("Введите имя: ")
4         Dim name = Console.ReadLine()
5         Console.WriteLine($"Привет, {name}")
6     End Sub
7 End Module
```

Здесь в методе Main вначале идет вызов метода `Console.Write("Введите имя: ")`, он, как и метод `Console.WriteLine()` выводит на консоль некоторую строку.

Затем с помощью оператора **Dim** определяется переменная `name`

```
1 Dim name = Console.ReadLine()
```

Этой переменной присваивается результат другого встроенного метода - **Console.ReadLine**, который позволяет считать с консоли введенную строку. То есть мы введем в консоли строку (точнее имя), и эта строка окажется в переменной `name`.

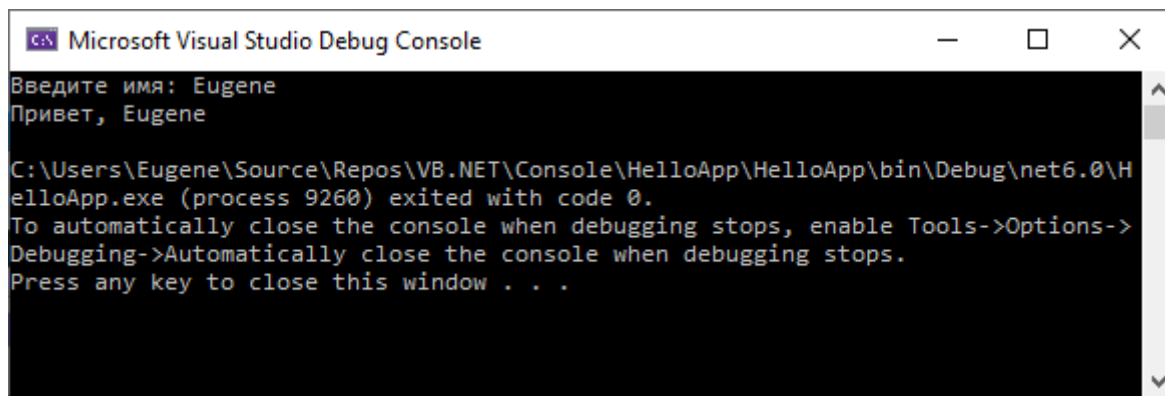
Затем введенное имя выводится на консоль:

```
1 Console.WriteLine($"Привет, {name}")
```

Чтобы ввести значение переменной `name` внутрь выводимой на консоль строки, применяются фигурные скобки `{}`. То есть при выводе строки на консоль выражение `{name}` будет заменяться на значение переменной `name` - введенное имя.

Однако чтобы можно было вводить таким образом значения переменных внутрь строки, перед строкой указывается знак доллара `$`.

Запустим проект на выполнение с помощью клавиши F5 и после приглашения к вводу введем какое-нибудь имя:



```
Microsoft Visual Studio Debug Console

Введите имя: Eugene
Привет, Eugene

C:\Users\Eugene\Source\Repos\VB.NET\Console\HelloApp\HelloApp\bin\Debug\net6.0\HelloApp.exe (process 9260) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

Итак, мы создали первое приложение. Вы его можете найти на жестком диске в папке проекта в каталоге **bin\Debug\net6.0**. Оно будет называться по имени проекта и иметь расширение exe. И затем этот файл можно будет запускать без Visual Studio, а также переносить его на другие компьютеры, где установлен .NET 6.

[Назад](#) [Содержание](#) [Вперед](#)



ТАКЖЕ НА METANIT.COM

<p>5 месяцев назад • 1 комме...</p> <p>Встроенные компоненты ввода</p>	<p>10 дней назад • 1 коммент...</p> <p>Клиент на Xamarin Forms для SignalR</p>	<p>3 мес</p> <p>Асс МА Уст</p>
---	---	---

5 Комментариев

G

Присоединиться к обсуждению...

войти с помощью

или через DISQUS 

Имя



1

Поделиться

A

Аноним Анонимус

год назад

p>В начале файла идет директива Imports

p> Тут явно лишняя

0 0 Ответить • Поделиться ›



Metanit

Модератор

➔ Аноним Анонимус

год назад

поправил

0 0 Ответить • Поделиться ›

S

Stark

2 года назад

Автор, обновите пожалуйста VB.NET под VB.NET 16.9 - 16.10 .NET 5 или .NET 6

0 0 Ответить • Поделиться ›



Metanit

Модератор

➔ Stark

2 года назад

в ближайшее время постараюсь обновить и добавить какие-нибудь доп руки

0 0 Ответить • Поделиться ›

S

Stark

➔ Metanit

2 года назад

Спасибо!

0 0 Ответить • Поделиться ›

Подписаться

О защите персональных данных

Не продавайте мои данные

YooMoney:

410011174743222

Qiwi:qiwi.com/n/METANIT

Перевод на карту

Номер карты:

4048415020898850

[Вконтакте](#) | [Телеграм](#) | [Twitter](#) | [Помощь сайту](#)

Контакты для связи: metanit22@mail.ru

Copyright © metanit.com, 2023. Все права защищены.