C# and .NET Framework

This book is about programming. It is intended to teach you to think as a programmer, to write code, to think in data structures and algorithms and to solve problems.

We use C# and Microsoft .NET Framework (the platform behind C#) only as means for writing programming code and we do not scrutinize the language's specifics. This same book can be found in versions for other languages like Java and C++, but the differences are not very significant.

Nevertheless, let's give a short account of C# (pronounced "see sharp").

C# is a modern programming language for development of software applications.

If the words "C#" and ".NET Framework" are unknown to you, you'll learn in details about them and their connection in the next chapter. Now let's explain briefly what C#, .NET, .NET Framework, CLR and the other technologies related to C# are.

The C# Programming Language

C# is a modern object-oriented, general-purpose programming language, created and developed by Microsoft together with the .NET platform. There is highly diverse software developed with C# and on the .NET platform: office applications, web applications, websites, desktop applications, mobile applications, games and many others.

C# is a high-level language that is similar to Java and C++ and, to some extent, languages like Delphi, VB.NET and C. All C# programs are objectoriented. They consist of a set of definitions in classes that contain methods and the methods contain the program logic – the instructions which the

computer executes. You will find out more details on what a class, a method and C# programs are in the next chapter.

Nowadays C# is one of the most popular programming languages. It is used by millions of developers worldwide. Because C# is developed by Microsoft as part of their modern platform for development and execution of applications, the .NET Framework, the language is widely spread among Microsoft-oriented companies, organizations and individual developers. For better or for worse, as of this book writing, the C# language and the .NET platform are maintained and managed entirely by Microsoft and are not open to third parties. Because of this, all other large software corporations like IBM, Oracle and SAP base their solutions on the Java platform and use Java as their primary language for developing their own software products. Unlike C# and the .NET Framework, the Java language and platform are open-source projects that an entire community of software companies, organizations and individual developers take part in. The standards, the specifications and all the new features in the world of Java are developed by workgroups formed out of the entire Java community, rather than a single company (as the case of C# and .NET Framework).

The C# language is distributed together with a special environment on which it is executed, called the Common Language Runtime (CLR). This environment is part of the platform .NET Framework, which includes CLR, a bundle of standard libraries providing basic functionality, compilers, debuggers and other development tools. Thanks to the framework CLR programs are portable and, once written they can function with little or no changes on various hardware platforms and operating systems. C# programs

are most commonly run on MS Windows, but the .NET Framework and CLR also support mobile phones and other portable devices based on Windows Mobile, Windows Phone and Windows 8. C# programs can still be run under Linux, FreeBSD, iOS, Android, MacOS X and other operating systems through the free .NET Framework implementation Mono, which, however, is not officially supported by Microsoft.

The Microsoft .NET Framework

The C# language is not distributed as a standalone product – it is a part of the Microsoft .NET Framework platform (pronounced "Microsoft dot net framework"). .NET Framework generally consists of an environment for the development and execution of programs, written in C# or some other language, compatible with .NET (like VB.NET, Managed C++, J# or F#). It consists of:

- the .NET programming languages (C#, VB.NET and others);
- an environment for the execution of managed code (CLR), which executes C# programs in a controlled manner;
- a set of development tools, such as the csc compiler, which turns C# programs into intermediate code (called MSIL) that the CLR can understand;
- a set of standard libraries, like ADO.NET, which allow access to databases (such as MS SQL Server or MySQL) and WCF which connects applications through standard communication frameworks and protocols like HTTP, REST, JSON, SOAP and TCP sockets.

The .NET Framework is part of every modern Windows distribution and is available in different versions. The latest version can be downloaded and installed from Microsoft’s website. As of this book’s publishing, the latest

version of the .NET Framework is 4.5. Windows Vista includes out-of-thebox .NET Framework 2.0, Windows 7 – .NET 3.5 and Windows 8 – .NET 4.5.

Why C#?

There are many reasons why we chose C# for our book. It is a modern programming language, widely spread, used by millions of programmers around the entire world. At the same time C# is a very simple and easy to learn (unlike C and C++). It is natural to start with a language that is suitable for beginners while still widely used in the industry by many large companies, making it one of the most popular programming languages nowadays.

C# or Java?

Although this can be extensively discussed, it is commonly acknowledged that Java is the most serious competitor to C#. We will not make a comparison between Java and C#, because C# is undisputedly the better,

more powerful, richer and just better engineered. But, for the purposes of this book, we have to emphasize that any modern programming language will be sufficient to learn programming and algorithms. We chose C#, because it is easier to learn and is distributed with highly convenient, free integrated development environment (e.g. Visual C# Express Edition). Those who prefer Java can prefer to use the Java version of this book, which can be found here: www.introprogramming.info.

Why Not PHP?

With regards to programing languages popularity, besides C# and Java, another widely used language is PHP. It is suitable for developing small web sites and web applications, but it gives rise to serious difficulties when

implementing large and complicated software systems. In the software industry PHP is used first and foremost for small projects, because it can easily lead developers into writing code that is bad, disorganized and hard to maintain, making it inconvenient for more substantial projects. This subject is also debatable, but it is commonly accepted that, because of its antiquated concepts and origins it is built on and because of various evolutionary reasons, PHP is a language that tends towards low-quality programming, writing bad code and creating hard to maintain software. PHP is a procedural language in concept and although it supports the paradigms of modern object-oriented programming, most PHP programmers write procedurally. PHP is known as the language of "code monkeys" in the software engineering profession, because most PHP programmers write terrifyingly low-quality code. Because of the tendency to write low-quality, badly structured and badly organized programming code, the entire concept of the PHP language and platform is considered wrong and serious companies (like Microsoft, Google, SAP, Oracle and their partners) avoid it. Due to this reason, if you want to become a serious software engineer, start with C# or Java and avoid PHP (as much as possible).

Certainly, PHP has its uses in the world of programming (for example creating a blog with WordPress, a small web site with Joomla or Drupal, or a discussion board with PhpBB), but the entire PHP platform is not wellorganized and engineered for large systems like .NET and Java. When it comes to non-web-based applications and large industrial projects, PHP is not by a long shot among the available options. Lots and lots of experience is necessary to use PHP correctly and to develop high-quality professional

projects with it. PHP developers usually learn from tutorials, articles and lowquality books and pick up bad practices and habits, which then are hard to

eradicate. Therefore, do not learn PHP as your first development

language. Start with C# or Java.

Based on the large experience of the authors' collective we advise you to

begin programming with C# and ignore languages such as C, C++ and PHP

until the moment you have to use them.

Preface 21

Why Not C or C++?

Although this is also debatable, the C and C++ languages are considered

complex and requires deep understanding of hardware. They still have their

uses and are suitable for low-level programming (e.g. programming for

specialized hardware devices), but we do not advise you to use C / C++ when

you are beginner who wants to learn programming.

You can program in pure C, if you have to write an operating system, a

hardware device driver or if you want to program an embedded device,

because of the lack of alternatives and the need to control the hardware very

carefully. The C language is very low-level and in no way do we advise

you to begin programming with it. A programmer’s productivity under pure C

is many times lower compared to their productivity under modern generalpurpose programming languages like C# and Java. A variant of C is used

among Apple / iPhone developers, but not because it is a good language, but

because there is no decent alternative. Most Apple-oriented programmers do

not like Objective-C, but they have no choice in writing in something else. In

2014 Apple promoted their new language Swift, which is of higher level and

aims to replace Objective-C for the iOS platform.

C++ is good when you have to program applications that require very close work with the hardware or that have special performance requirements (like 3D games). For all other purposes (like Web applications development or business software) C++ is inadequate. We do not advise you to pursue it, if you are starting with programming just now. One reason it is still being studied in some schools and universities is hereditary, because these institutions are very conservative. For example, the International Olympiad in Informatics (IOI) continues to promote C++ as the only language permitted to use at programming contests, although C++ is rarely used in the industry. If you don't believe this, look through some job search site and count the percentage of job advertisements with C++.

The C++ language lost its popularity mainly because of the inability to quickly write quality software with it. In order to write high-quality software in C++, you have to be an incredibly smart and experienced programmer, whereas the same is not strictly required for C# and Java. Learning C++ takes much more time and very few programmers know it really well. The productivity of C++ programmers is many times lower than C#'s and that is why C++ is losing ground. Because of all these reasons, the C++ language is slowly fading away and therefore we do not advise you to learn it.

Advantages of C#

C# is an object-oriented programming language. Such are all modern programming languages used for serious software systems (like Java and C++). The advantages of object-oriented programming are brought up in many passages throughout the book, but, for the moment, you can think of object-oriented languages as languages that allow working with objects from the real world (for example student, school, textbook, book and others).

Objects have properties (e.g. name, color, etc.) and can perform actions (e.g. move, speak, etc.).

By starting to program with C# and the .NET Framework platform, you are on a very perspective track. If you open a website with job offers for programmers, you'll see for yourself that the demand for C# and .NET specialists is huge and is close to the demand for Java programmers. At the same time, the demand for PHP, C++ and other technology specialists is far lower than the demand for C# and Java engineers.

For the good programmer, the language they use is of no significant meaning, because they know how to program. Whatever language and technology they might need, they will master it quickly. Our goal is not to teach you C#, but rather teach you programming! After you master the fundamentals of programming and learn to think algorithmically, when you acquaint with other programming languages, you will see for yourself how much in common they have with C# and how easy it will be to learn them. Programming is built upon principles that change very slowly over the years and this book teaches you these very principles.

Examples Are Given in C# 5 and Visual Studio 2012

All examples in this book are with regard to version 5.0 of the C# language and the .NET Framework 4.5 platform, which is the latest as of this book's publishing. All examples on using the Visual Studio integrated development environment are with regard to version 2012 of the product, which were also the latest at the time of writing this book.

The Microsoft Visual Studio 2012 integrated development environment (IDE) has a free version, suitable for beginner C# programmers, called

Microsoft Visual Studio Express 2012 for Windows Desktop. The difference between the free and the full version of Visual Studio (which is a commercial software product) lies in the availability of some functionalities, which we will not need in this book.

Although we use C# 5 and Visual Studio 2012, most examples in this book will work flawlessly under .NET Framework 2.0 / 3.5 / 4.0 and C# 2.0 / 3.5 / 4.0 and can be compiled under Visual Studio 2005 / 2008 / 2010.

It is of no great significance which version of C# and Visual Studio you'll use while you learn programming. What matters is that you learn the principles of programming and algorithmic thinking! The C# language, the .NET Framework platform and the Visual Studio integrated development environment are just tools and you can exchange them for others at any time. If you read this book and VS2012 is not currently the latest, be sure almost all of this book's content will still be the same due to backward compatibility.

How To Read This Book?

Reading this book has to be accompanied with lots and lots of practice. You won't learn programming, if you don't practice! It would be like trying to learn

how to swim from a book without actually trying it. There is no other way! The more you work on the problems after every chapter, the more you will learn from the book.

Everything you read here, you would have to try for yourself on a computer. Otherwise you won't learn anything. For example, once you read about Visual Studio and how to write your first simple program, you must by all means download and install Microsoft Visual Studio (or Visual C# Express) and try to write a program. Otherwise you won't learn! In theory, everything seems

easy, but programming means practice. Remember this and try to solve the problems from this book. They are carefully selected – they are neither too hard to discourage you, nor too easy, so you'll be motivated to perceive solving them as a challenge. If you encounter difficulties, look for help at the discussion group for the "C# Programming Fundamentals" training course at Telerik Software Academy: http://forums.academy.telerik.com (the forum is intended for Bulgarian developers but the people "living" in it speak English and will answer your questions regarding this book, don't worry). Thousands students solve the exercises from this book every year so you will find many solutions to each problem from the book. We will also publish official solutions + tests for every exercise in the book at its web site.

Reading this book without practicing is meaningless! You must spend much more time on writing programs than reading the text itself. It is just like learning to drive: no one can learn driving by reading books. To learn driving, you need to drive many times in different situations, roads, cars, etc. To learn programming, you need to program!

Everybody has studied math in school and knows that learning how to solve math problems requires lots of practice. No matter how much they watch and listen to their teachers, without actually sitting down and solving problems, they won't learn. The same goes for programming. You need lots of practice. You need to write a lot, to solve problems, to experiment, to endeavor in and to struggle with problems, to make mistakes and correct them, to try and fail, to try anew and experience the moments when things finally work out. You need lots and lots of practice. This is the only way you will make progress.

So people say that to become a developer you might need to write at least 50,000 – 100,000 lines of code, but the correct number can vary a lot. Some people are fast learners or just have problem-solving experience. Others may need more practice, but in all cases practicing programming is very important! You need to solve problems and to write code to become a developer. There is no other way!

Do Not Skip the Exercises!

At the end of each chapter there is a considerable list of exercises. Do not skip them! Without exercises, you will not learn a thing. After you read a

chapter, you should sit in front of the computer and play with the examples you have seen in the book. Then you should set about solving all problems. If you cannot solve them all, you should at least try. If you don't have all the time necessary, you must at least attempt solving the first few problems from each chapter. Do not carry on without solving problems after every chapter, it would just be meaningless! The problems are small feasible situations where you apply the stuff you have read. In practice, once you have become programmers, you would solve similar problems every day, but on a larger and more complex scale.

You must at all cost strive to solve the exercise problems after every chapter from the book! Otherwise you risk not learning anything and simply wasting your time.

How Much Time Will We Need for This Book?

Mastering the fundamentals of programming is a crucial task and takes a lot of time. Even if you're incredibly good at it, there is no way that you will learn programming on a good level for a week or two. To learn any human

skill, you need to read, see or be shown how it is done and then try doing it yourselves and practice a lot. The same goes for programming – you must either read, see or listen how it is done, then try doing it yourself. Then you would succeed or you would not and you would try again, until you finally realize you have learned it. Learning is done step by step, consecutively, in series, with a lot of effort and consistency.

If you want to read, understand, learn and acquire thoroughly and in-depth the subject matter in this book, you have to invest at least 2 months for daylong activity or at least 4-5 months, if you read and exercise a little every day. This is the minimum amount of time it would take you to be able to grasp in depth the fundamentals of programming.

The necessity of such an amount of lessons is confirmed by the free trainings at Telerik Software Academy (http://academy.telerik.com), which follow this very book. The hundreds of students, who have participated in trainings based on the lectures from this book, usually learn all subjects from this book within 3-4 months of full-time work. Thousands of students every year solve all exercise problems from this book and successfully sit on programming exams covering the book's content. Statistics shows that anyone without prior exposure to programming, who has spent less than the equivalent of 3-4 months daylong activity on this book and the corresponding courses at Telerik Academy, fails the exams.

The main subject matter in the book is presented in more than 1100 pages, which will take you a month (daylong) just to read them carefully and test the sample programs. Of course, you have to spend enough time on the exercises (few more months); without them you would hardly learn programming.

Exercises: Complex or Easy?

The exercises in the book consist of about 350 problems with varying difficulty. For some of them you will need a few minutes, for others several hours (if you can solve them at all without help). This means you would need a month or two of daylong exercising or several months, if you do it little by little.

The exercises at each chapter are ordered in increasing level of difficulty. The first few exercises are easy, similar to the examples in the chapter. The last few exercises are usually complex. You might need to use external resources (like information from Wikipedia) to solve them. Intentionally, the last few exercises in each chapter require skills outside of the chapter. We want to push you to perform a search in your favorite search engine. You need to learn searching on the Internet! This is an essential skill for any programmer. You need to learn how to learn. Programming is about learning every day. Technologies constantly change and you can't know everything. To be a programmer means to learn new APIs, frameworks, technologies and tools every day. This cannot be avoided, just prepare yourself. You will find many problems in the exercises, which require searching on the Internet. Sometimes you will need the skills from the next chapter, sometimes some well-known algorithm, sometimes something else, but in all cases searching on the Internet is an essential skill you need to acquire.

Solving the exercises in the book takes a few months, really. If you don't have that much time at your disposal, ask yourselves if you really want to pursue programming. This is a very serious initiative in which you must invest a really great deal of efforts. If you really want to learn programming on a good level, schedule enough time and follow the book or the video lectures

based on it.

Why Are Data Structures and Algorithms Emphasized?

This book teaches you, in addition to the basic knowledge in programming, proper algorithmic thinking and using basic data structures in programming. Data structures and algorithms are a programmer's most important fundamental skills! If you have a good grasp of them, you will not have any trouble becoming proficient in any software technology, development tool, framework or API. That is what the most serious software companies rely on when hiring employees. Proof of this are job interviews at large companies like Google and Microsoft that rely exclusively on algorithmic thinking and knowledge of all basic data structures and algorithms.

The information below comes from Svetlin Nakov, the leading author of this book, who passed software engineering interviews at Microsoft and Google in 2007-2008 and shares his own experience