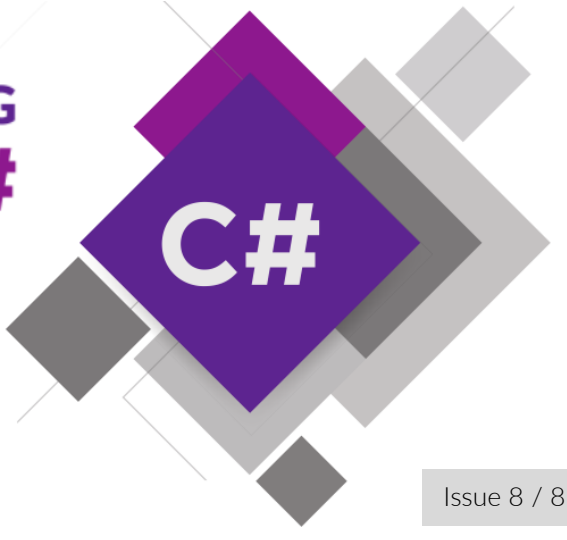
PROGRAMMING BASICS WITH C#



Issue 8 / 8

HOW TO BECOME A SOFTWARE ENGINEER?

Learn how to become a skillful software engineer and the 4 major groups of skills that every developer should have: coding skills, algorithmic thinking, computer science and software engineering concepts, programming languages and software technologies. Learn about the software academies, training courses, tutorials, books and community groups that could help you to become a software developer and start a job in the IT industry.

Dr. Svetlin Nakov

Software University – https://softuni.org

How to Become a Software Engineer?

In this mini book, we shall explain how a beginner could become a skillful software engineer and start a job in the IT industry.

Programming Is Learned by a Lot of Writing, Not Reading!

If someone thinks they will read a book, a tutorial or training and learn to program without writing code and solving problems, this is definitely a delusion. Programming needs **a lot of practice**, with code writing every day and solving hundreds, even thousands of problems, persistently for years.

You need **to solve a lot of problems**, to make mistakes, to fix, to search for solutions and information from the Internet, to try, to experiment, to find better solutions, to get used to the code, syntax, the programming language, the development environment, to search for errors and debugging the broken code, the algorithmic thinking, breaking the problems into smaller parts, gaining experience and raising your skills every day, because when you learn to write code, this is only the **first step to the profession of the "software engineer"**. You have a lot to learn, really!

How to Become a Software Developer?

Dear readers, probably many of you have the ambition to become programmers and develop software for a living, or work in the IT area. That's why we have prepared for you a **short guide on "How to become a programmer"**, so we can help you take the first steps towards this desired profession.

You can become a programmer (working in a software company) after **at least 1-2 years of intensive training and coding every day**, solving thousands of programming tasks, development of several more serious practical projects and gaining a lot of experience with code writing and software development. You can't become a programmer for a month or two! The profession of software engineer requires **a lot of knowledge**, covered with **a lot of practice**.

Video: Become a Software Engineer – 4 Essential Skills

Watch a video lesson about the 4 essential skills, that all software engineers should have: https://youtu.be/qO1ckspCqxs.

The 4 Essential Skills of the Software Developers

There are **4 main skill groups** where all programmers must have. Most of these skills are resistant in time and are not influenced by the development in specific technologies (that are changing constantly). These are the skills that **any good programmer** has and to which every beginner must strive:

- **coding** (20%)
- **algorithmic thinking** (30%)
- **computer science and software engineering concepts** (25%)
- **languages and software technologies** (25%)

Skill #1 – Coding (20%)

Writing code forms around 20% of the minimum knowledge and skills of a programmer, needed for starting a job in a software company. The skill of coding includes the following components:

- Working with variables, conditional statements, loops
- Using functions, methods, classes and objects
- Data manipulation: arrays, lists, hash tables, strings

The skill of coding **can be acquired in a few months** of hard learning and solving practical problems by writing code every day. This book covers only the first point of coding: **working with variables, conditional statements and loops**. The rest remains to be learned in follow-up trainings, courses and books.

The book (and the courses based on it) gives only the beginning of one long and serious training on the way to professional programming. If you don't learn perfectly the material from this book, you can't become a programmer. You are going to miss fundamentals and it will be harder in the future. For this reason, **give enough time to programming basics**: solve many problems and write code every day for months until you learn to **solve every problem from the book very easily**. Then go ahead.

We specifically note that **the programming language does not matter** for the ability to code. You can code or not. If you can code with C#, you will easily learn to code with Java, C++ or other languages. That's why **the coding skills** are being studied quite seriously at the beginners courses in SoftUni (curriculum), and each programming book for beginners starts with them, including this one.

Skill #2 – Algorithmic Thinking (30%)

The algorithmic (logical, engineering, mathematical, abstract) thinking forms around 30% of the minimum skills of a programmer needed to start the profession. **Algorithmic thinking** is the ability to break a particular problem into a logical sequence (algorithm), to find a solution for every separate step and then assemble the steps into a working solution. That is the most important skill of any programmer.

How to **build algorithmic thinking**?

- Algorithmic thinking is developed by solving **multiple programming (1000+) problems**, as diverse as possible. That is the recipe: solving thousands of practical problems, building algorithms and implementing the algorithms, along with debugging the issues that come up in the process.
- Sciences like physics, mathematics and identical ones helps a lot, but they are not mandatory! People with **engineering and technical skills** usually learn very easily to think logically, because they already **have problem solving skills**, even if it is not algorithmic.
- The ability of **solving programming problems** (for which algorithmic thinking is needed) is extremely important for a programmer. Many companies test particularly this skill during their job interviews.

The current book develops the **beginner level of algorithmic thinking**, but it's not enough to make you a good programmer. To become good at this profession you must add **logical thinking and problem solving skills**, beyond the range of this book. For example, working with **data structures** (arrays, lists, matrices, hash-tables, binary trees) and basic **algorithms** (searching, sorting, searching in tree structures, recursion, etc.).

Algorithmic thinking skill can be seriously developed at the beginner courses for software engineers in SoftUni (see curriculum) as well as in specialized algorithmic courses data structures and algorithms.

As you can guess the programming language doesn't matter for the development of algorithmic thinking. To think logically is a universal skill, and it's not related only to programming. Because of the well-developed logical thinking it's believed that programmers are smart people, and that a stupid person can't become a programmer.

Skill # 3 – Computer Science and Software Engineering Concepts (25%)

Fundamental knowledge and skills for programming, software development, software engineering and computer science comprise around 25% of the developer's minimum skills to start a job. Here are the more important of these skills and knowledge:

- basic mathematical concepts related to programming: coordinate systems, vectors and matrices, discrete and non-discrete mathematical functions, state automata and state machines, combinatorics and statistics concepts, algorithm complexity, mathematical modeling and others
- programming skills – code writing, working with data, using conditional statements and loops, working with arrays, lists and associative arrays, strings and text processing, working with streams and files, using programming interfaces (APIs), working with IDE, debugger, developer tools, etc.
- data structures and algorithms – lists, trees, hash-tables, graphs, search, sorting, recursion, binary search trees, etc.
- object-oriented programming (OOP) – working with classes, objects, inheritance, polymorphism, abstraction, interfaces, data encapsulation, exceptions management, design pattern
- functional programming (FP) – working with lambda functions, higher order functions, functions that return a function as a result, closure, etc.
- databases – relational and non-relational databases, database modeling (tables and links between them), SQL query language, object-relational mapping (ORM), transactions and transaction management
- network programming – network protocols, network communication, TCP/IP, concepts, tools and technologies from computer networks
- client-server interaction, peer to peer communication, back-end technologies, front-end technologies, MVC architectures
- technologies for server development (back-end) – Web server architecture, HTTP protocol, MVC architecture, REST architecture, web development frameworks, templating engines
- front-end technologies (client-side development) – HTML, CSS, JS, HTTP, DOM, AJAX, communication with back-end, calling REST API, front-end frameworks, basic design and UX (user experience) concepts
- mobile technologies – mobile apps, Android and iOS development, mobile user interface (UI), calling server logic
- embedded systems – microcontrollers, digital and analog input and output control, sensor access, peripheral management

- operating systems – working with operating systems (Linux, Windows, etc.), installation, configuration and basic system administration, process management, memory, file system, users, multitasking, virtualization and containers
- parallel and asynchronous programming – thread management, asynchronous tasks, promises, common resources, and access synchronization
- software engineering – source control systems, development management, task planning and management, software development methodologies, software requirements and prototypes, software design, software architectures, software documentation
- software testing – unit testing, test-driven development, QA engineering, error reporting and error tracking, automation testing, build processes and continuous integration

We need to clarify once again that the programming language does not matter for the assimilation of all these skills. They accumulate slowly, over many years of practice in the profession. Some knowledge is fundamental and can be learned theoretically, but for their full understanding and in-depth awareness, you need years of practice.

Fundamental knowledge and skills for programming, software development, software engineering, and computer science are taught during the Software Engineering Program, as well as a number of elective courses. Working with a variety of software libraries, APIs, frameworks and software technologies and their interaction gradually builds this knowledge and skills, so do not expect that you will understand them from a single course, book or project.

To start working as a programmer, only basic knowledge in the areas listed above is enough and the improvement happens at the workplace according to the technology and development tools used in the company and the team.

Skill #4 – Programming Languages and Software Technologies (25%)

Programming languages and software development technologies form around 25% of a programmer's skills. They have the largest learning content, but they change very fast over time. If we look at the job offers in the software industry, they usually mention words like the ones below, but in the job offers they secretly mention the three main skills: coding, algorithmic thinking and knowing the fundamentals of computer science and software engineering.

For those clearly technological skills the programming language does matter.

- Note: only for these 25% of the profession the programming language does matter!
- For the rest 75% of the skills the programming language doesn't matter, and these skills are resistant in time and transportable between different languages and technologies.

Here are some commonly used software development stacks which software companies are looking for (as of Jan 2019):

- C# + OOP + FP + classes from .NET + SQL Server databases + Entity Framework (EF) + ASP.NET MVC + HTTP + HTML + CSS + JS + DOM + jQuery
- JavaScript (JS) + OOP + FP + databases + MongoDB or MySQL + HTTP + web programming + HTML + CSS + JS + DOM + jQuery + Node.js + Express + Angular or React
- Python + OOP + FP + databases + MongoDB or MySQL + HTTP + web development + HTML + CSS + JS + DOM + jQuery + Django or Flask
- Java + Java API classes + OOP + FP + databases + MySQL + HTTP + web programming + HTML + CSS + JS + DOM + jQuery + JSP / Servlets + Spring MVC or Java EE / JSF

- **PHP** + OOP + databases + MySQL + HTTP + web development + HTML + CSS + JS + DOM + jQuery + Laravel or Symfony or other MVC framework for PHP
- **C++** + OOP + STL + Boost + native development + databases + HTTP + other languages
- **Swift** + MacOS + iOS + Cocoa + Cocoa Touch + XCode + HTTP + REST + other languages

If the words above look scary and absolutely incomprehensible, then you are at the very beginning of your career and you need **many years of learning** until you reach the profession of a "software engineer". Do not worry, every programmer goes through one or several technology stacks and needs to study **a set of interconnected technologies**, but the bottom line is **the ability to write programming logic (coding)**, and the skill of **algorithmic thinking** (to solve programming problems). It's not possible without them!

The Programming Language Doesn't Matter!

As it already became clear, **the difference between programming languages** and more specifically between the skills of developers in different languages and technologies forms around **10-20% of the skills**.

- All programmers have around **80-90% of the same skills** that do not depend on the programming language! These are the skills to program and to design and develop software, that are very similar in different programming languages and development technologies.
- The more languages and technologies you know, the faster you will learn new ones, and the less you will feel the difference between them.

Indeed, **the programming language almost does not matter**, you just have to learn to program, and this starts with **coding** (this book) goes on in the more complex **programming concepts** (like data structures, algorithms, OOP and FP) and includes the use of **fundamental knowledge and skills for software development, software engineering and computer science**.

Only when you start working with a specific technology into a software project you will need **a specific programming language**, knowledge about specific programming libraries (APIs), frameworks and software technologies (front-end UI technologies, back-end technologies, ORM technologies, etc.). Keep calm, you will learn them, all programmers are learning them, but first you need to learn the foundation: **to program and do it well**.

This book uses the C# language, but it is not required and can be replaced with Java, JavaScript, Python, PHP, C++, Ruby, Swift, Go, Kotlin, or any other language. To be a **software developer**, you need to learn **coding** (20%), learn **algorithmic thinking**, and **solve problems** (30%), to have **fundamental knowledge of programming and computer science** (25%) and to master a **specific programming language and the technologies around it** (25%). Be patient, for a year or two all this can be mastered on a good starting level, if you are serious.

About the Software University (SoftUni)

The Software University (SoftUni) is **the largest training center for software engineers in the South-Eastern Europe**. Tens of thousands of students pass through the university every year. SoftUni was founded in 2014, as a continuation of the hard work of Dr. Svetlin Nakov in training **skillful software engineering professionals** by a practical, modern and high-quality education, that combines fundamental knowledge with modern software technologies and a lot of practice.

Video: SoftUni

Watch a video lesson about SoftUni and SoftUni Judge here: https://youtu.be/TDIDXnFCzoo.

The Software University

The **Software University** (SoftUni) provides **quality education**, **profession**, **job** and **diploma** for programmers, software engineers and IT professionals. SoftUni builds an extremely successful and strong **connection between education and industry** by collaboration with hundreds of software companies, provides job and internship of its students, creates quality professionals for the software industry and directly responds to the needs of employers via the training process.

Free Programming Courses at SoftUni

SoftUni organizes **free programming lessons for beginners**: online and physically in few locations. The purpose is to give a chance to **everyone who is interested** in programming and technologies to **try programming** and check if they are interested and if they would get seriously involved in software development. You can sign up for the free course in **programming basics** using the SoftUni application page: https://softuni.org.

The free courses at SoftUni have the purpose to introduce you to **basic programming constructions** in the software development world, that you can use at any programming language. These include working with **data**, **variables** and **expressions**, using **conditional statements**, constructing **loops**, defining and calling **methods** and other approaches for building programming logic. The trainings are **highly practically oriented** which means that **the emphasis is strongly on exercises**, and you get the opportunity to apply your knowledge during the learning process.

This **programming book** accompanies the free programming lessons for beginners in SoftUni and serves as an additional teaching aid to help the learning process.

Study to Become a Software Engineer

If you are a beginner or you have some programming experience and skills, but still you don't feel like a professional and you don't have a developer job, generally you have the **following possibilities** to **become a software engineering professional**:

- To study for a **software engineer at SoftUni** and make programming your profession;
- To continue developing as a programmer **in your own way**, for example through self-training or via online lessons;
- To **stay at coder level**, without going more seriously into programming.

Study Software Engineering in SoftUni

The first, and respectively recommended option to master fully and on high level the profession of a "software engineer", is to start your training via the **end-to-end SoftUni program for software engineers**: https://softuni.org. The SoftUni curriculum is carefully developed by **Dr. Svetlin Nakov and his team**, in order to provide you consequently and with gradually increasing complexity all the skills that a software engineer must have, in order **to start a career as a software developer** in an IT company.

Training Duration in SoftUni

The training in SoftUni has a duration of **2-3 years** (depending on the profession and the selected specializations) and during that period it is normal to reach a good starting level (junior developer), but this is **only if you study seriously** and write code intensely every day. Upon having good grades, a typical student **starts a job around the middle of the training (after around 1.5 years)**. Thanks to the well-developed partners network, **the career center of SoftUni offers work** in a software or IT company to all SoftUni students who have very good or excellent grades. **Starting a job** in the major in case of having good grades at SoftUni, combined with willingness to work and reasonable expectations towards the employers, is almost guaranteed.

It Takes at Least a Year of Intense Code Writing to Become a Programmer

Keep in mind that **to become a programmer takes a lot of efforts**, writing tens of thousands of lines of code, and solving hundreds, even thousands of practical problems, and this takes years! If someone offers you "**an easier program**" and promises you to become a programmer and start working within 3-4 months, then either they are **lying** to you, or they will give you such a low level, that **companies won't even take you as a trainee**, even if you pay to the company that is wasting its time with you. There are exceptions, of course, for example if you are not starting from scratch, or if you have extremely well-developed engineering thinking, or if you apply for a very low position (for example technical support), but in general, **you cannot become a programmer if you haven't spent at least 1 year of intense learning and code writing**!

The Entrance Exam in SoftUni

In order **to enroll at SoftUni** you need to attend an **entrance exam** in "Programming Basics" on the material from this book. If you easily solve the problems in this book, then you are ready for the exam. Also, pay attention to the chapters on **preparation for the practical exam in programming**. They will give you a good idea of the level of difficulty of the exam and the types of tasks that you need to learn solving.

If the tasks from the book and the preparation examples are hard for you, then you **need more preparation**. Enroll for the free course in "Programming Basics" or go through the book carefully one more time, without skipping solving **the problems in any of the studied topics**! You must learn how **to solve them with ease**, without helping yourselves with the guidelines and the sample solutions.

The SoftUni Curriculum for Software Engineers

What follows after the entrance exam is a **serious curriculum** in the SoftUni program for training software engineers. It is formed as a sequence of **modules in a number of courses** in programming and software technologies, fully directed towards gaining fundamental knowledge in software development and acquiring **practical skills for working** as a programmer with the most contemporary software technologies. Students are given a choice between **a number of professions** and specializations focused on C#, Java, JavaScript, Python, PHP and other languages and technologies. Each profession is trained in a number of modules with 4 months duration, and each module consists of 2 or 3 courses. The classes are divided into **theoretical preparation** (30%) and **practical exercises, projects and trainings** (70%), and each course ends with a practical exam or practical academic project.

How Many Hours per Day Does the Training Take?

The training for software engineers at SoftUni is a **very serious occupation** and you need to spend on it **at least 4-5 hours every day**, preferably your entire attention and time. Combining **working and training** is not always successful, but if you work something easy and you have a lot of spare time, it is a good option. SoftUni is an appropriate option for **school students**, **university students** and **people who work**, but it is best if you assign your entire time to your training and mastering the profession. It will not work if you spend 2 or 4 hours a week on it!

The forms of training at SoftUni are **on-site** (the better choice) and **online** (if you don't have another option). In both forms of training, in order to learn the program in the curriculum (that is required by software companies for starting a job), you need **a lot of learning**. You just need to **find the time for it**! Reason #1 for having hard time on the road to the profession in SoftUni is not spending enough time for the training: as a minimum you need to spend at least 20-30 hours a week.

SoftUni for People who Work and Study

We recommend to everyone who gets **excellent score at the SoftUni entrance exam** and are really passionate about making programming their profession, to leave the rest of their commitments aside and **spend their entire time** on learning the profession of a "software engineer" and start making a living through it.

- For people who **work** this means quitting their job (and getting a loan or decreasing their expenses, in order to spend with a lower income a period of 1-2 years until they start working in the new profession).
- For people who **study** in a traditional university, this means to move significantly their focus towards programming and the practical courses in SoftUni, by decreasing the time spend in the traditional university.
- For **unemployed** people this is an excellent chance to assign their entire time, power and energy on acquiring a new, perspective, well paid and highly sought profession, that will give them good life quality and a long-term prosperity.
- For **students** in secondary schools and high-schools this means **giving a priority** to what is more important in their development: studying practical programming in SoftUni, that will give them a profession and a job, or giving their full attention to the traditional education system or combining smartly both undertakings. Unfortunately, often priorities are determined by parents, and we don't have a solution for these cases.

We recommend to all who **cannot get an excellent score at the SoftUni entrance exam** to spend more time on better learning, understanding, and most of all, practicing the material studied in the present book. If you cannot easily solve the problems in this book, you will not be able to cope with programming and software development in the future.

Do not skip the programming basics! Do not under any circumstances make bold decisions and quit your job or the traditional university, making great plans for your future profession of a software engineer, if you don't have an excellent grade at the SoftUni entrance exam! It measures if programming is suitable for you, to what extend you like it and if you are motivated to study it seriously, and work this for years every day with joy and pleasure.

Study Software Engineering in Your Own Way

Another possibility to develop after this book is to **continue studying programming outside of SoftUni**. You can enroll or subscribe to **video trainings** that go into more details in programming with C# or other languages and development platforms. You can **read books** on programming and software technologies, follow **online tutorials** and other online resources – there are plenty of free materials on the Internet. However, keep in mind that the most important thing towards the profession of a programmer is **to do practical projects**!

You cannot become a programmer without a lot of code writing and intense practicing. Allocate **sufficient time** to it. You cannot become a programmer for a month or two. On the Internet you will find a wide variety of **free resources**, such as books, manuals, video lessons, online and on-site courses on programming and software development. However, you need to invest **at least a year or two** to acquire a foundation level, needed for starting a job.

After you gain some experience, find a way to start **an internship in a company** (which will be almost impossible unless you'd spent at least a year of intense code writing before that) or come up with **your own practical project**, on which you need to spend a few months, even a year, in order to learn based on the trial-and-error principle.

Keep in mind that there are many ways to become a programmer, but they all have something in common: **intense code writing and years of practice**!

Recommended Resources for Developers

A **huge amount of resources** is available on the Web for developers: online trainings, courses, tutorials, books, interactive training sites, etc.

We cannot mention all of them, because they change over the time and because the list might be huge. What we recommend is to **join the developer communities** in your region, because this will help you a lot when you study.

Online Communities for Beginners in Programming

Regardless of the path you have chosen, if you are seriously involved in programming, we recommend subscribing to specialized **online forums, discussion groups and communities**, from which you can get assistance by your colleagues and track the novelties in the software industry.

If you will study programming seriously, **surround yourselves with people who are involved in programming** seriously. Join **communities of software developers**, attend software conferences, go to events for programmers, find friends with whom you can talk about programming and discuss problems and bugs, find environment that can help you. In Sofia and other cities there are free events for programmers, a few times a week. In smaller localities you have the Internet and an access to the entire online community.

Here are some recommended **resources** that will be useful for your development as a programmer:

- https://softuni.org – official **website of SoftUni**. In it you will find free (and not only) courses, seminars, video tutorials and trainings in programming, software technologies and digital competences.
- https://fb.com/softuni.org – official **Facebook page of SoftUni**. By it you will learn regarding new courses, seminars and events related to programming and software development.

- https://introprogramming.info – official website of the **books "Programming Basics"** with **C#** and **Java** by Dr. Svetlin Nakov and his team. The books examine in-depth programming basics, basic data structures and algorithms, object-oriented programming, and other basic skills, and are an excellent continuation for reading after the current book. However, **besides reading, you need to do intense code writing**, do not forget that!
- https://stackoverflow.com – **Stack Overflow** is one of the **widest** discussion forums for programmers worldwide, where you will find assistance for any possible question in the world of programming. If you are fluent in English, look up at StackOverflow and ask your questions there.
- https://udemy.com – **Udemy** is one of the biggest **marketplaces** for technical trainings, offered free or at affordable prices.
- https://meetup.com/find/tech – look for **tech meetups** around your city and involve in communities that you like. Most of the tech meetups are free and newbies are welcome.

Good Luck to All!

We **wish you endless success in the profession and in your life**! We will be really happy if we have helped you get **passionate about programming** and we have inspired you to go bravely towards becoming a "software engineer", which will bring you a good job that you will work with pleasure, give you a quality life and prosperity, as well as amazing perspectives for development and possibilities for making impressive projects with inspiration and passion.

What's Next?

The "**Programming Basics**" series consists of publications for beginners in programming that covers the following topics:

1. First Steps in Programming – commands, programs, C#, Visual Studio,
2. Simple Calculations – variables, calculations, console input / output
3. Simple Conditions – conditional statements, the "if-else" construction
4. More Complex Conditions – nested if-else, logical "and", "or", "not"
5. Repetitions (Loops) – simple for-loops (repeat from 1 to n)
6. Nested Loops – nested loops and problem solving, drawing 2D figures
7. More Complex Loops – loops with a step, infinite loops with breaks
8. **How to Become a Software Engineer?**

You reached the last publication. Congratulations!

Sign-Up to Study in SoftUni

The easiest way to become a software engineer is to go through the "**Software University**" training program at SoftUni. Signup now for the **free Programming Basics training course**:

https://softuni.org/apply

Join the SoftUni community and study programming for free in our **interactive training platform**. Get **live support** and mentoring from our trainers. Become a software developer now!