SJIF Impact Factor (2024): 8.675| ISI I.F. Value: 1.241| Journal DOI: 10.36713/epra2016 ISSN: 2455-7838(Online)
EPRA International Journal of Research and Development (IJRD)
Volume: 9 | Issue: 2 | February 2024 - Peer Reviewed Journal

BASICS OF C# FOR BEGINNERS

Sabirbaev Askar Jalgasovich
Student, Department of Applied Mathematics and Intellectual Technology,
National University of Uzbekistan named after Mirzo Ulugbek

ABSTRACT
This article will take a look at the basic programming terminology and clarify first C# program. We will familiarize ourselves with programming – what it means and its connection to computers and programming languages. Briefly, the different stages of software development will be reviewed. The article will introduce the C# language, the .NET platform and the different Microsoft technologies used in software development as well as it will examine what tools are needed to program in C#.
KEY WORDS: *desktop application, product trials, deployment, HelloCSharp, static void, code intendation.*

INTRODUCTION

Nowadays computers have become irreplaceable. We use them to solve complex problems at the workplace, look for driving directions, have fun and communicate. They have countless applications in the business world, the entertainment industry, telecommunications and finance. It's not an overstatement to say that computers build the neural system of our contemporary society and it is difficult to imagine its existence without them. Despite the fact that computers are so wide-spread, few people know how they really work. In reality, it is not the computers, but the programs (the software), which run on them, that matter. It is the software that makes computers valuable to the end-user, allowing for many different types of services that change our lives.

The essence of programming is to control the work of the computer on all levels. This is done with the help of "orders" and "commands" from the programmer, also known as programming instructions. To "program" means to organize the work of the computer through sequences of instructions. These commands (instructions) are given in written form and are implicitly followed by the computer (respectively by the operating system, the CPU and the peripheral devices). A sequence of steps to achieve, complete some work or obtain some result is called an algorithm. This is how programming is related to algorithms. Programming involves describing what you want the computer to do by a sequence of steps, by algorithms.

In this article we will take a look at the C# programming language – a modern high level language. When a programmer uses C#, he gives commands in high level, like from the position of a general executive in a factory. The instructions given in the form of programs written in C# can access and control almost all computer resources directly or via the operating system. Before we learn how to write simple C# programs, let's take a good look at the different stages of software development, because programming, despite being the most important stage, is not the only one.

MAIN BODY

Writing software can be a very complex and time-consuming task, involving a whole team of software engineers and other specialists. As a result, many methods and practices, which make the life of programmers easier, have emerged. All they have in common is that the development of each software product goes through several different stages:

a) *Gathering the requirements for the product and creating a task.* In the beginning, only the idea for a certain product exists. It includes a list of requirements, which define actions by the user and the computer. In the general case, these actions make already existing activities easier – calculating salaries, calculating ballistic trajectories or searching for the shortest route on Google maps are some examples. In many cases the software implements a previously nonexistent functionality such as automation of a certain activity.

b) The requirements for the product are usually defined in the form of documentation, written in English or any other language. There is no programming done at this stage. The requirements are defined by experts, who are familiar with the problems in a certain field. They can also write them up in such a way that they are easy to understand by the programmers. In the general case, these experts are not programming specialists, and they are called business analysts.

b) *Planning and preparing the architecture and design.* After all the requirements have been gathered comes the planning stage. At this stage, a technical plan for the implementation of the project is created, describing the platforms, technologies and the initial architecture (design) of the program. This step includes a fair amount of creative work, which is done by software engineers with a lot of experience. They are sometimes called software architects. According to the requirements, the following parts are chosen:

SJIF Impact Factor (2024): 8.675| ISI I.F. Value: 1.241| Journal DOI: 10.36713/epra2016 ISSN: 2455-7838(Online)
EPRA International Journal of Research and Development (IJRD)
Volume: 9 | Issue: 2 | February 2024 - Peer Reviewed Journal

- The type of the application – for example console application, desktop application (GUI, Graphical User Interface application), client-server application, Web application, Rich Internet Application (RIA), mobile application, peer-to-peer application or other;
- The architecture of the software – for example single layer, double layer, triple layer, multi-layer or SOA architecture;
- The programming language most suitable for the implementation – for example C#, Java, PHP, Python, Ruby, JavaScript or C++, or a combination of different languages;
- The technologies that will be used: platform (Microsoft .NET, Java EE, LAMP or another), database server (Oracle, SQL Server, MySQL, NoSQL database or another), technologies for the user interface (Flash, JavaServer Faces, Eclipse RCP, ASP.NET, Windows Forms, Silverlight, WPF or another), technologies for data access (for example Hibernate, JPA or ADO.NET Entity Framework), reporting technologies (SQL Server Reporting Services, Jasper Reports or another) and many other combinations of technologies that will be used for the implementation of the various parts of the software system.
- The development frameworks that will simplify the development, e.g. ASP.NET MVC (for .NET), Knockout.js (for JavaScript), Rails (for Ruby), Django (for Python) and many others.
- The number and skills of the people who will be part of the development team (big and serious projects are done by large and experienced teams of developers);
- The development plan – separating the functionality in stages, resources and deadlines for each stage.
- Others (size of the team, locality of the team, methods of communication etc.).

Although there are many rules facilitating the correct analysis and planning, a fair amount of intuition and insight is required at this stage. This step predetermines the further advancement of the development process. There is no programming done at this stage, only preparation.

c) *Implementation (includes the writing of program code).* The stage, most closely connected with programming, is the implementation stage. At this phase, the program (application) is implemented (written) according to the given task, design and architecture. Programmers participate by writing the program (source) code. The other stages can either be short or completely skipped when creating a small project, but the implementation always presents; otherwise the process is not software development. This book is dedicated mainly to describing the skills used during implementation – creating a programmer's mindset and building the knowledge to use all the resources provided by the C# language and the .NET platform, in order to create software applications.

d) *Product trials (testing).* It is a very important stage of software development. Its purpose is to make sure that all the requirements are strictly followed and covered. This process can be implemented manually, but the preferred way to do it is by automated tests. These tests are small programs, which automate the trials as much as possible. There are parts of the functionality that are very hard to automate, which is why product trials include automated as well as manual procedures to ensure the quality of the code.

The testing (trials) process is implemented by quality assurance engineers (QAs). They work closely with the programmers to find and correct errors (bugs) in the software. At this stage, it is a priority to find defects in the code and almost no new code is written.

Many defects and errors are usually found during the testing stage and the program is sent back to the implantation stage. These two stages are very closely tied and it is common for a software product to switch between them many times before it covers all the requirements and is ready for the deployment and usage stages.

e) Deployment and operation. Deployment is the process which puts a given software product into exploitation. If the product is complex and serves many people, this process can be the slowest and most expensive one. For smaller programs this is a relatively quick and painless process. In the most common case, a special program, called installer, is developed. It ensures the quick and easy installation of the product. If the product is to be deployed at a large corporation with tens of thousands of copies, additional supporting software is developed just for the deployment. After the deployment is successfully completed, the product is ready for operation. The next step is to train employees to use it.

An example would be the deployment of a new version of Microsoft Windows in the state administration. This includes installation and configuration of the software as well as training employees how to use it.

The deployment is usually done by the team who has worked on the software or by trained deployment specialists. They can be system administrators, database administrators (DBA), system engineers, specialized consultants and others. At this stage, almost no new code is written but the existing code is tweaked and configured until it covers all the specific requirements for a successful deployment.

f) *Support.* During the exploitation process, it is inevitable that problems will appear. They may be caused by many factors – errors in the software, incorrect usage or faulty configuration, but most problems occur when the users change their requirements. As a

SJIF Impact Factor (2024): 8.675| ISI I.F. Value: 1.241| Journal DOI: 10.36713/epra2016 ISSN: 2455-7838(Online)
EPRA International Journal of Research and Development (IJRD)
Volume: 9 | Issue: 2 | February 2024 - Peer Reviewed Journal

result of these problems, the software loses its abilities to solve the business task it was created for. This requires additional involvement by the developers and the support experts. The support process usually continues throughout the whole life-cycle of the software product, regardless of how good it is.

The support is carried out by the development team and by specially trained support experts. Depending on the changes made, many different people may be involved in the process – business analysts, architects, programmers, QA engineers, administrators and others.

For example, if we take a look at a software program that calculates salaries, it will need to be updated every time the tax legislation, which concerns the serviced accounting process, is changed. The support team's intervention will be needed if, for example, the hardware of the end user is changed because the software will have to be installed and configured again.

Implementation, testing, deployment and support are mostly accomplished using programming.
The documentation stage is not a separate stage but accompanies all the other stages. Documentation is an important part of software development and aims to pass knowledge between the different participants in the development and support of a software product. Information is passed along between different stages as well as within a single stage. The development documentation is usually created by the developers (architects, programmers, QA engineers and others) and represents a combination of documents.

OUR FIRST C# PROGRAM

Before we continue with an in depth description of the C# language and the .NET platform, let's take a look at a simple example, illustrating how **a program written in C#** looks like:

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

The only thing this program does is to *print the message "Hello, C#!"* on the default output. It is still early to execute it, which is why we will only take a look at its structure. Later we will describe in full how to compile and run a given program from the command prompt as well as from a development environment.

HOW DOES OUR FIRST C# PROGRAM WORK?

Our first program consists of three logical parts:
- Definition of a class. On the first line of our program we define a class called *HelloCSharp*. The simplest definition of a class consists of the keyword *class*, followed by its name. In our case the name of the class is *HelloCSharp*. The content of the class is located in a block of program lines, surrounded by curly brackets: {}.
- Definition of a method Main ( ). On the third line we define a method with the name Main(), which is the starting point for our program. Every program written in C# starts from a Main() method with the following title (signature):
- Contents of the method Main ( ).

```
static void Main (string [ ] args)
```

The method must be declared as shown above, it must be static and void, it must have a name Main and as a list of parameters it must have only one parameter of type array of string. In our example the parameter is called args but that is not mandatory. This parameter is not used in most cases so it can be omitted (it is optional). In that case the entry point of the program can be simplified and will look like this:

```
static void Main ( )
```

If any of the aforementioned requirements is not met, the program will compile but it will not start because the starting point is not defined correctly.

CONTENTS OF THE MAIN () METHOD

The content of every method is found after its signature, surrounded by opening and closing curly brackets. On the next line of our sample program we use the system object System. Console and its method Write Line () to print a message on the default output (the console), in this case "Hello, C#!". In the Main () method we can write a random sequence of expressions and they will be executed in the order we assigned to them.

SJIF Impact Factor (2024): 8.675| ISI I.F. Value: 1.241| Journal DOI: 10.36713/epra2016 ISSN: 2455-7838(Online)
EPRA International Journal of Research and Development (IJRD)
Volume: 9 | Issue: 2 | February 2024 - Peer Reviewed Journal

MAIN FORMATTING RULES

If we want our code to be correctly formatted, we must follow several important rules regarding indentation:
- Methods are indented inside the definition of the class (move to the right by one or more [Tab] characters);
- Method contents are indented inside the definition of the method;
- The opening curly bracket { must be on its own line and placed exactly under the method or class it refers to;
- The closing curly bracket } must be on its own line, placed exactly vertically under the respective opening bracket (with the same indentation);
- All class names must start with a capital letter;
- Variable names must begin with a lower-case letter;
- Method names must start with a capital letter;

Code indentation follows a very simple rule: when some piece of code is logically inside another piece of code, it is indented (moved) on the right with a single [Tab]. For example, if a method is defined inside a class, it is indented (moved to the right). In the same way if a method body is inside a method, it is indented. To simplify this, we can assume that when we have the character "{", all the code after it until its closing "}" should be indented on the right.

FILE NAMES CORRESPOND TO CLASS NAMES

Every C# program consists of one or several class definitions. It is accepted that each class is defined in a separate file with a name corresponding to the class name and a .cs extension. When these requirements are not met, the program will still work but navigating the code will be difficult. In our example, the class is named HelloCSharp, and as a result we must save its source code in a file called HelloCSharp.cs.

THE C# LANGUAGE AND THE .NET PLATFORM

The first version of C# was developed by Microsoft between 1999 and 2002 and was officially released to the public in 2002 as a part of the .NET platform. The .NET platform aims to make software development for Windows easier by providing a new quality approach to programming, based on the concepts of the "virtual machine" and "managed code". At that time the Java language and platform reaped an enormous success in all fields of software development; C# and .NET were Microsoft's natural response to the Java technology.

C# is a modern, general-purpose, object-oriented, high-level programming language. Its syntax is similar to that of C and C++ but many features of those languages are not supported in C# in order to simplify the language, which makes programming easier.
The C# programs consist of *one or several files* with a .cs extension, which contain definitions of classes and other types. These files are compiled by the C# compiler (csc) to executable code and as a result assemblies are created, which are files with the same name but with a different extension (.exe or .dll). For example, if we compile HelloCSharp.cs, we will get a file with the name HelloCSharp.exe (some additional files will be created as well, but we will not discuss them at the moment).

We can run the compiled code like any other program on our computer (by double clicking it). If we try to execute the compiled C# code (for example HelloCSharp.exe) on a computer that does not have the .NET Framework, we will receive an error message.
C# uses the following *keywords* to build its programming constructs:

Abstract	As	base	bool	Break	Byte
Case	Catch	char	checked	class	const
Continue	Decimal	default	delegate	Do	double
else	Enum	event	explicit	Extern	false
Finally	Fixed	float	for	Foreach	goto
If	Implicit	in	int	Interface	internal
Is	Lock	long	namespace	New	null
Object	operator	out	override	Params	private
Protected	Public	readonly	ref	Return	sbyte
Sealed	Short	sizeof	stackalloc	Static	string
Struct	Switch	this	throw	True	try
Typeof	Uint	ulong	unchecked	Unsafe	ushort
Using	Virtual	void	volatile	While	

(the list is taken from MSDN in March 2013 and may not be complete)
Since the creation of the first version of the C# language, not all keywords are in use. Some of them were added in later versions. The main program elements in C# (which are defined and used with the help of keywords) are classes, methods, operators, expressions, conditional statements, loops, data types, exceptions and few others. In the next few chapters of this book, we will review in details all these programming constructs along with the use of the most of the keywords from the table above.

SJIF Impact Factor (2024): 8.675| ISI I.F. Value: 1.241| Journal DOI: 10.36713/epra2016 ISSN: 2455-7838(Online)
EPRA International Journal of Research and Development (IJRD)
Volume: 9 | Issue: 2 | February 2024 - Peer Reviewed Journal

One of the biggest advantages of the .NET Framework is the *built-in automatic memory management.* It protects the programmers from the complex task of manually allocating memory for objects and then waiting for a suitable moment to release it. This significantly increases the developer productivity and the quality of the programs written in C#. In the .NET Framework, there is a special component of the CLR that looks after memory management. It is called a "*garbage collector*" (automated memory cleaning system). The garbage collector has the following main tasks: to check when the allocated memory for variables is no longer in use, to release it and make it available for allocation of new objects.

It is important to note that it is not exactly clear at what moment the memory gets cleaned of unused objects (local variables for example). According to the C# language specifications, it happens at some moment after a given variable gets out of scope but it is not specified, whether this happens instantly, after some time or when the available memory becomes insufficient for the normal program operation.

USED LITERATURE

1. *S. Nakov., Co. (2013) Fundamentals of Computer Programming with C#*
2. *R. Vystavel. (2017) C# Programming for Absolute Beginners*
3. *A. Harris. (2002) Microsoft C# Programming for the Absolute Beginner*
4. *B.Albahari, P.Drayton, B.Merrill. (2001) C# Essentials.*