

# Algorithm Design In Programming Language Education

Tuğba Saray Çetinkaya<sup>1</sup>

Ali Çetinkaya<sup>2</sup>

## Abstract

The field of algorithm development on computer systems continues to grow in importance in today's world, highlighting the critical nature of algorithm design and implementation. With the increasing diversity of algorithm use cases, it is essential to emphasize the significance of proper algorithm design methods for each problem. As such, the design of new and effective algorithms is of paramount importance to the continued growth and innovation of computer systems. Algorithms play a crucial role in solving problems within established systems. Therefore, it is important to detail the appropriate algorithm methods for each problem. As the areas of algorithm use become more diverse, the programming languages used in these platforms also change.

This study aims to help learners understand the steps to take when designing algorithms and learning programming languages, regardless of the language used. Common algorithm designs are demonstrated on Python, C, C++, and C# programming languages. These designs cover shared concepts across all four languages and will aid in coding on these programming languages. The goal is to learn and apply algorithm design on multiple programming languages.

This study covers the basics of algorithms, programming, programming concepts, and the fundamentals of computer programming. It is essential to understand these topics in programming language education in order to correctly and error-free install algorithms, helping newcomers to the software industry take the correct steps.

- 
- 1 Lecturer, Istanbul Gelisim University, Istanbul Gelisim Vocational High School, Department of Computer Technologies, Information Security Technology Program, ORCID: 0000-0003-1639-553X
  - 2 Lecturer, Istanbul Gelisim University, Istanbul Gelisim Vocational High School, Department of Electronics and Automation, Electronic Technology Program, ORCID ID: 0000-0003-4535-3953

## 1. INTRODUCTION

Nowadays, when algorithms are combined with programming languages, great technological works are emerging. Examples of these works continue to differentiate and increase in military and police systems, agricultural applications, image processing applications, data engineering field, language processing works, and cyber security applications. With the differentiation of the areas of use of algorithms, the programming languages used on these platforms are also changing.

Python is an object-oriented and functional modern programming language. It is ideal for beginners due to its readability and ease of use. C# is a simple, modern, object-oriented, and type-safe programming language that combines the high productivity of application development languages with the raw power of C and C++. The Java programming language, on the other hand, shares many features that are common to most programming languages used today. Since it is designed with the structures of C and C++, where their languages are similar, the language is familiar to C and C++ programmers (Lerdorf, 2002; Hejlsberg, 2003; Arnold, 2005; Deitel, 2004; Kelly, 2016; Gavrilović, 2018; Pala, 2019; Chollet, 2021; Chen, 2023).

Nowadays, the use of computer systems in all sectors has increased. The analysis, design, development, application, and testing of the capabilities required by these systems are important for algorithms and software (Alaybeyoğlu, 2006; Özyurt, 2016; Akkaya, 2020; Shnaider, 2023). In the problems solved in systems established with algorithms on integrated systems, it is necessary to determine the algorithm method to be applied to the subject. In addition, determining the method to be used on this algorithm application is also of great importance in terms of integrity in the application. No matter how good the scope or evaluation of the mathematical models in the designed applications is, uncertainties may arise during the operation of the system. In this context, when any uncertain situation is encountered, linguistic variables and functions that best meet human thoughts should be created on the algorithm of the system.

In this chapter of this book, we will progress by learning and applying algorithm design with multiple programming languages and their concepts through C, C++, C#, and Python programming languages.

*What will we learn?*

- We will learn the basics of algorithms and algorithm logic,
- We will recognize the concept of flowchart,

- We will process the basics of programming with C and develop algorithms,
- We will process the basics of programming with C++ and develop algorithms,
- We will process the basics of programming with C# and develop algorithms,
- We will process the basics of programming with Python and develop algorithms,

In the following sections, we will detail our explanations and applications on what an algorithm is, what programming is, what the basic concepts of programming are, and what computer programming covers.

## 2. WHAT IS AN ALGORITHM?

All of the sequential logical steps required to solve a problem or solve a problem for a specific purpose are called “*algorithms*”. An algorithm constitutes the entire path to be followed within the scope of the solution of a problem. In short, an algorithm is a method of creating the desired output information based on the information we have. The most important task of an algorithm related to any problem is that it can create steps for solving the problem.

Performing the coding of the algorithm to be prepared for a problem in any programming language is the simple part of the job. When writing the expression form of an algorithm textually, it is written step by step as a textual pseudo code for the problem to be solved, each line that occurs is numbered, and should start with ‘start’ and end with ‘end’. There are five expectations that should be considered when working on algorithms in C, C++, C#, or Python programming languages. These expectations are effectiveness, finiteness, definiteness, input/output, and success/performance measurement. Our answer to the question of what to expect from algorithms should be within the scope of the following explanations:

1) *Effectiveness*: Each step of the designed algorithm should be expressed in an understandable, simple, and precise way. Endless loops should not be entered by creating unnecessary repetitions on the algorithm.

2) *Finiteness*: Each algorithm must have a starting point and an end point.

3) *Definiteness*: In order to reach the result information within the scope of the algorithm’s task, the input information must be compatible at each step and give the same result when it is newly executed.

4) *Input/Output*: It should have the result values that will be formed as a result of the operations to be performed on the algorithm.

5) *Success/Performance*: Algorithms that give different results in each application should be avoided. Even if the system seems to be working, the success rate may be low. At this point, the development of “high-performance algorithms” should be our goal.

### 3. WHAT IS PROGRAMMING?

In the field of education and training, big data analysis, cloud technologies, and wearable technologies have recently increased their impact (Turk, 2018). The tools that enable software production are programming languages, and algorithms constitute the most basic of these (Namlı, 2017). In programming teaching, algorithm design has a special place in defining the problem to be solved. In this context, how programmer candidates perceive the algorithm becomes important (Gökoğlu, 2017). An algorithm is a way where how to solve a problem or achieve a determined goal is explained. As can be understood from this statement, an algorithm is not a result, but a tool that leads to a result (Aytekin, 2018).

In the literature studies on programming, research has been conducted on preschool, kindergarten, primary school, secondary school, high school, university, and higher education (Yükselturk, 2016; Yalçinkaya, 2018; Karaman, 2019; Deniz, 2019; Kaban, 2021; Bayraktar, 2021). The importance of programming education in the development of cognitive skills of students has been emphasized (Akçay, 2016). Scientific studies conducted in Turkey on coding education in the preschool period have been systematically examined, and in this context, it has been stated that algorithms are an advantage in problem-solving in the coding process (Zurnacı, 2022). In a study, Özdener (2008) focused on how vocational high school and university students interpreted algorithms related to time efficiency in structured computer programming. In another study, it was aimed to determine the effect of secondary students’ attitudes towards computers and their self-directed learning skill levels on their success in programming language teaching (Alper, 2019). In the field of education, the inability to teach software languages is due to the complexity of the courses and the inability to comprehend the integrity of the subject (Ozoran, 2012). For programming students to learn a software language, adopting C and Python programming languages in “introduction to programming” courses is necessary (Balreira, 2023). Data structures can be used for the classification of algorithms on Java and C++ programming languages (Bui,

2019). An educational study developed using the artificial neural network from the sub-branches of artificial intelligence was carried out using the C#.NET programming language (Çevik, 2012).

*Program (software)*: It can be defined as a series of statements written using a computer programming language to solve a specific problem. The statements to be used in programming languages are control and loop expressions. “If/if, else/else” and “switch/case”, which have common usage in programming languages, are used as control expressions, while “while”, “do/while” and “for” are used as loop expressions.

Loop expressions used in programming languages are: while, do/while, for, foreach, range and len/in. The loop expressions used in the C and C++ programming languages are while, do/while, and for. The loop expressions used in the C# programming language are while, do-while, for, and foreach. The loop expressions used in the Python programming language are while, for, range, and len/in. The loop expressions used in the Java programming language are while, do-while, and for. The control expressions used in the C, C++, C# and Python programming languages are: if, if/else, “if/else if /... /else” and switch/case. However, there are no “switch/case” and “do/while” structures directly in the Python programming language. In Table 14, the “switch/case” structure is given by creating a function for the Python programming language. In Table 17, the “do/while” structure is given by creating a function for the Python programming language.

#### 4. WHAT IS COMPUTER PROGRAMMING?

The steps that we will follow respectively in writing a program (software) that we will develop to solve the problem by using an algorithm on a computer are as follows: understanding what the problem is, determining the requirements for the solution, determining the input-output and operations of the problem, writing the algorithm that solves the problem, writing the algorithm in a programming language, and testing the accuracy of the program. To understand what the problem is and solve the problem, answers to 3 questions are sought. The first of these questions is “what is necessary”, the second is “how to produce solutions”, and the third is “what are the current situations”. Detailed analyses should be carried out before operations are performed on the algorithms. These analyses include: determining the data to be used or input-output definitions, determining the equations and formulas to be used when developing the algorithm, developing an algorithm (pseudo codes and flowcharts), writing a program in a programming language by using the algorithm, demonstrating

accuracy and verification (determining whether the program meets the user's requirements), removing undetected errors, and preparing program documentation.

Relational and logical operators are used when running programs by utilizing comparison commands depending on the characteristics of the program. Programs (software) consist of many commands that run respectively. The commands used to control the program blocks which are requested to be executed or not to be executed based on certain conditions in programs are called "control commands". These commands are considered as operator precedence, mathematical operation operators, logical expression representation, logical operators, the equivalent of a mathematical expression in a programming language, representative examples and coding of logical operators, a sample table of logical operations, increment-decrement operators, and the expression for the use of increment-decrement operators in a programming language. The operators and operations used as standard in the C, C#, Python, and C++ programming languages are given between Table 1 and Table 9.

*Table 1. Operator Precedence (Order of Operations)*

Operator Precedence	The Relevant Operator or Operators
1 <sup>st</sup> order (first operation priority)	* / %
2 <sup>nd</sup> order	+ -
3 <sup>rd</sup> order	<< >>
4 <sup>th</sup> order	< > <= >=
5 <sup>th</sup> order	== !=
6 <sup>th</sup> order	&
7 <sup>th</sup> order	^
8 <sup>th</sup> order	
9 <sup>th</sup> order	&&
10 <sup>th</sup> order	
11 <sup>th</sup> order	?:
12 <sup>th</sup> order (last operation priority)	=

The related mathematical operation operators are shown, and their explanations are given in Table 2.

*Table 2. Mathematical Operation Operators*




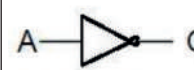
Math Operators	Explanation of the Operator
*	Operator for the multiplication process
/	Operator for the division process
+	Operator for the addition process
-	Operator for the subtraction process
%	Operator for the mod process

*Table 3. Logical Expressions*

Logical Expression	Explanation of the Logical Expression
=	the operator for the “assignment” expression
==	the operator for the “equivalence” expression
>	the operator for the “greater than” expression
<	the operator for the “less than” expression
>=	the operator for the “Greater than or equal to” expression
<=	the operator for the “less than or equal to” expression
&&	the operator for the logical “AND” expression
	the operator for the logical “OR” expression
!	the operator for the logical “NOT” expression
!=	the operator for the “not equal” expression
^	the operator for the “exclusive OR” expression

Table 3 shows the relevant logical expressions and their explanations. The use and representation of these expressions as operators are given in Table 4. On Table 4, the AND (&&) operator works with the result True (1) if both logical expressions are true, otherwise it works with the result False (0). The Or (||) operator works with the result True (1) if any of the logical expressions are true, otherwise, it works with False (0). The NOT (!) operator takes the “not”  $\rightarrow$  (that is, inverse) of the logical expression. If the expression is True (1), it works with the result False (0), if False (0), then it works with the result True (1). The exclusive OR (^) operator works with the result True (1) if any of the logical expressions is different from the other, otherwise (that is, if they are both the same), it works with the result False (0).

**Table 4.** Logical Operators

AND OPERATION (AND)	OR OPERATION (OR)	EXCLUSIVE OR (XOR)	NOT OPERATION (NOT)																																																			
																																																						
THE TABLE FOR “AND” GATE	THE TABLE FOR “OR” GATE	THE TABLE FOR “EXCLUSIVE OR” GATE	THE TABLE FOR “NOT” GATE																																																			
<table border="1"> <thead> <tr><th>A</th><th>B</th><th>Q</th></tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>1</td></tr> </tbody> </table>	A	B	Q	0	0	0	0	1	0	1	0	0	1	1	1	<table border="1"> <thead> <tr><th>A</th><th>B</th><th>Q</th></tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>1</td></tr> </tbody> </table>	A	B	Q	0	0	0	0	1	1	1	0	1	1	1	1	<table border="1"> <thead> <tr><th>A</th><th>B</th><th>Q</th></tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>0</td></tr> </tbody> </table>	A	B	Q	0	0	0	0	1	1	1	0	1	1	1	0	<table border="1"> <thead> <tr><th>A</th><th>Q</th></tr> </thead> <tbody> <tr><td>0</td><td>1</td></tr> <tr><td>1</td><td>0</td></tr> </tbody> </table>	A	Q	0	1	1	0
A	B	Q																																																				
0	0	0																																																				
0	1	0																																																				
1	0	0																																																				
1	1	1																																																				
A	B	Q																																																				
0	0	0																																																				
0	1	1																																																				
1	0	1																																																				
1	1	1																																																				
A	B	Q																																																				
0	0	0																																																				
0	1	1																																																				
1	0	1																																																				
1	1	0																																																				
A	Q																																																					
0	1																																																					
1	0																																																					

*Table 5. The Equivalences of Mathematical Expressions on the Programming Language*

Mathematical Expression	The Equivalent of the Operation in the C Programming Language
The operation that gives the sum of 5 and 2	variable = 5 + 2;
Multiplying the number 2 with the parameter a	variable = a * 2;
Obtaining the remaining number (mod) from the division of 7 with 3	mod = 7%3;
The operation that summit 5 and 4 first, multiplies the output value by 9, and divides the result of the operation by 2	A=((5+4)*9)/2;
$s = \frac{2x}{a+b}$	s = 2*x / (a+b);
$f = ax - (b + c)$	f=a*x-(b+c);
$f = \frac{1}{1+\frac{1}{n}}$	f=1 / (1+(1/n));



The equivalences of some mathematical expressions in the programming language are presented in Table 5. Representative examples and codings of logical operators are given in Table 6.

*Table 6. Representative Examples and Codings of Logical Operators*

Operation	Operator Sign	Representative Example	Representative Coding
Logical AND	&&	(A<B) AND (B<C)	(A<B) && (B<C)
Logical OR		(A<B) OR (B<C)	(A<B)    (B<C)
NOT	!	NOT(A<B)	!(A<B)
Exclusive OR (EXOR)	^	(A XOR B)	(A ^ B)

Table 7 shows the output information generated as a result of the logical operations applied based on the input operations on the algorithms.

*Table 7. Example Table of Logical Operations*

X	Y	!X	!Y	X && Y	X    Y	X ^ Y
0	0	1	1	0	0	0
0	1	1	0	0	1	1
1	0	0	1	0	1	1
1	1	0	0	1	1	0

In Tables 8 and 9, the increment and decrement operators are shown and their use in programming languages is given. In addition, different uses and explanations of increment and decrement operators on algorithms are given in Table 9.

*Table 8. Increment and Decrement Operators*

Increment Operators	Decrement Operators
++	--

**Table 9. The Use of Increment and Decrement Operators on a Programming Language**





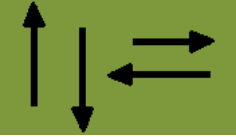


Example operation Statement	Explanation
++a	Increase “a” by one, and use the new value of “a” in the statement in which “a” is located.
a++	Use the value of “a” in the statement in which “a” is located, and then increase the value of “a” by one.
--a	Decrease “a” by one, and use the new value of “a” in the statement in which “a” is located.
a--	Use the value of “a” in the statement in which “a” is located, and then decrease the value of “a” by one.

## 5. THE USE OF FLOWCHARTS IN ALGORITHM DESIGN

An algorithm is a cluster of instructions arranged in a specific logical order that produces a solution for a specific problem when it is executed. When creating this cluster, we use two techniques: Pseudo Code and Flowchart. These two concepts are the techniques used when creating and defining algorithms. Pseudo Code is a language consisting of limited words and is similar to programming language. A flowchart, on the other hand, is a graphical demonstration of an algorithm with geometric shapes. The representation of the algorithm is revealed by the flow lines connecting these shapes.

Table 10 shows the flowchart elements used when creating algorithms. The ellipse shows the starting and ending places of a flowchart. The parallelogram is used to show the data entry points on a flowchart. The rhombus represents the decision-making processes on the flowchart. The rectangle shows the arithmetic operations in the flowchart. The arrows indicate the directions in which the process steps will go on a flowchart. The cylinder represents the database process on the flowchart. The document represents the information output process on the flowchart.

Table 10. Flowchart Symbols

Symbol	Symbol Name	Meaning
	Ellipse	It shows the starting and ending places of a flowcharts.
	Parallelogram	It is used to show the data entry points on a flowchart
	Rhombus	It represents the decision-making processes on a flowchart
	Rectangle	It shows the arithmetic operations in a flowchart.
	Arrows	It indicate the directions in which the process steps will go on a flowchart.
	Cylinder	It represents the database process on a flowchart.
	Document	It represents the information output process on a flowchart.

## 6. ALGORITHM OPERATIONS ON PROGRAMMING LANGUAGES

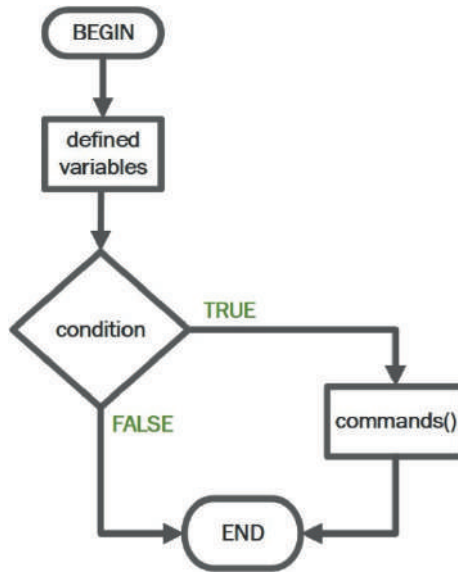
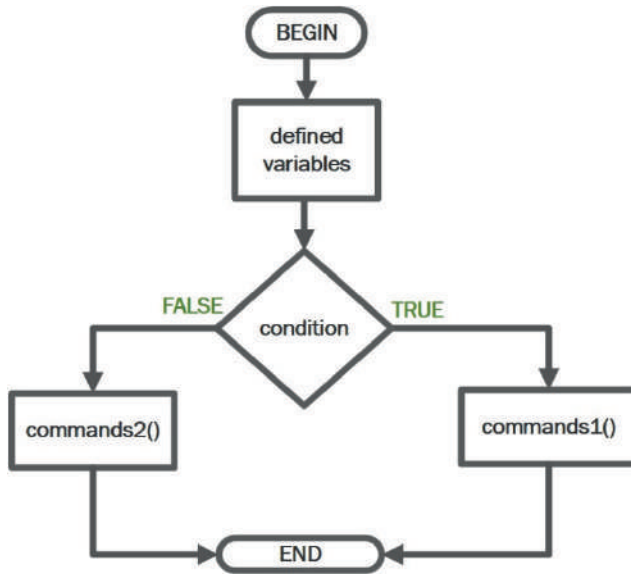


Figure 1. The Flowchart of the “if” Control Block in Programming

Table 11. Code Representation of the “if” Loop on Different Programming Languages

<pre> if (condition) {     commands(); }           </pre>	<pre> if condition:     commands()           </pre>
Code Representation of the “if” Control Block in the C Programming Language	Code Representation of the “if” Control Block in the Python Programming Language
<pre> if (condition) {     commands(); }           </pre>	<pre> if (condition) {     commands(); }           </pre>
Code Representation of the “if” Control Block in the C++ Programming Language	Code Representation of the “if” Control Block in the C# Programming Language

The “if” command, which is represented as a control block in Figure 1 and Table 11, is a conditional operation command. Depending on whether a certain condition is correct, it is ensured that a line of commands or a code block is executed. The comparison process is performed at the time of operation. The Boolean (true/false) value is returned depending on the control result of the expression that comes after the “if” command. This Boolean value is considered as “TRUE” if the operation is correct and as “FALSE” if the operation is false. If the result of the related condition is logically correct, the command or command block written after “if” will be executed. If the result of the condition is incorrect, the command or command block after the “if” will be skipped, and operations will not occur on commands inside the “if” structure. If the logical result of the condition in parentheses is TRUE, the function of the command(s) is executed. If the result of the condition in parentheses is FALSE, the next state on the command line is passed without any action. In the chart on the right side, the arrow goes to the end through the “no” loop. In Table 11, the use of the “if” control block is shown with codes on four separate programming languages (c, C++, c#, and Python).

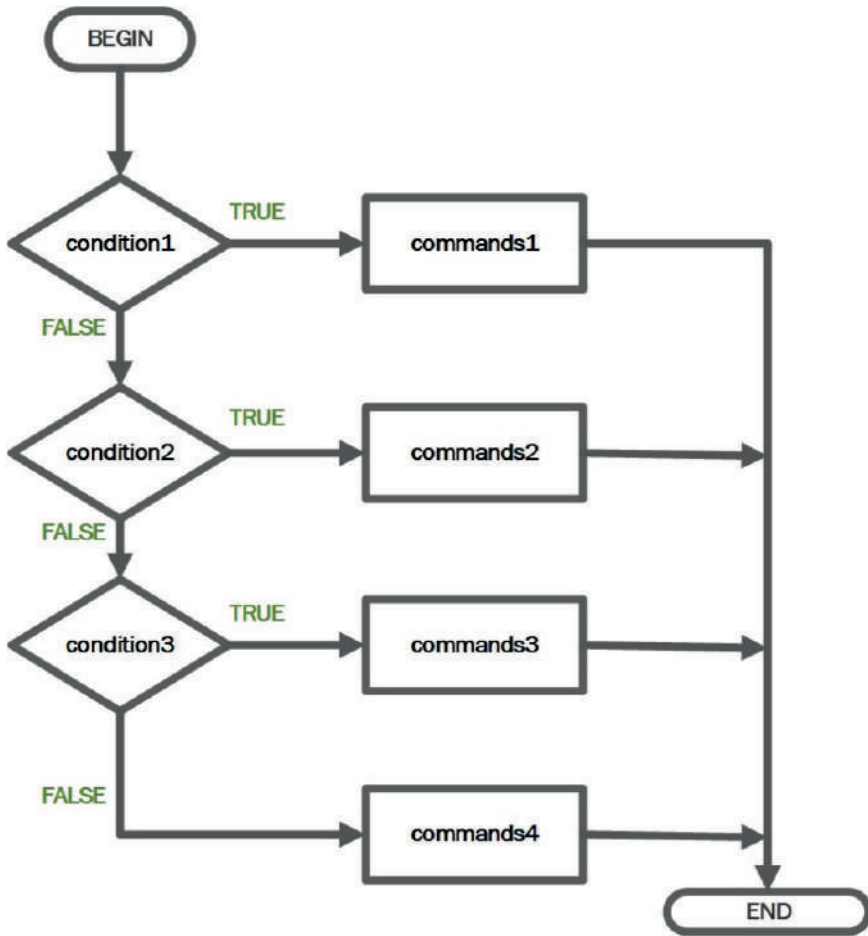


*Figure 2. Chart of the “if.. else” Control Block in Programming*

Table 12. The use of the “if...else” Loop with Codes on Different Programming Languages

<pre> if (condition) {     commands1(); } else {     commands2(); } </pre>	<pre> if condition:     commands1() else:     commands2() </pre>
Code Representation of the “if...else” Control Block in the C Programming Language	Code Representation of the “if...else” Control Block in the Python Programming Language
<pre> if (condition) {     commands1(); } else {     commands2(); } </pre>	<pre> if (condition) {     commands1(); } else {     commads2(); } </pre>
Code Representation of the “if...else” Control Block in the C++ Programming Language	Code Representation of the “if...else” Control Block in the C# Programming Language

If the logical result of the condition given in the parenthesis of the “if...else” control block shown in Figure 2 and Table 12 is TRUE, commands1 is executed, if the result is FALSE, commands2 is executed and the next command is passed. This use of “if...else” is a comparison command that allows one command block or another command block to operate depending on whether a certain condition is true. In Table 12, code representations of the “if...else” control block in four separate programming languages (C, C++, C#, and Python) are presented with examples.



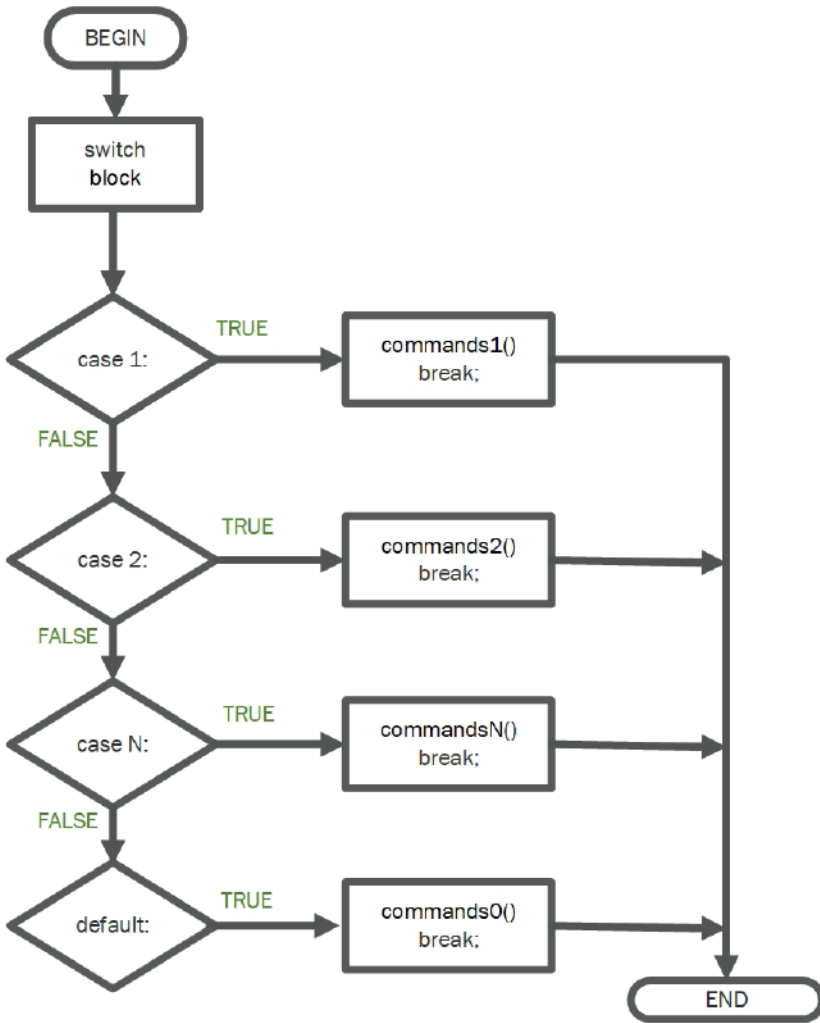
*Figure 3. In programming, “if/else if/.../else” Chart of the Control Block*

In Figure 3, how to use the “if/else if/.../else” control block is shown in general. In Table 13, the use of the “if/else if/.../else” control block in four separate programming languages (C, C++, C#, and Python) is shown with examples. If the logical result of the condition given in parentheses of the “if/else if/.../else” control block is TRUE, commands1 is executed, and if the result is FALSE, commands2 is executed, and the next command is passed.

**Table 13. The use of the “if/else if / .... /else” Control Block in Different Programming Languages**

<pre> if (statement1) {     case1(); } else if (statement2) {     case2(); } else if (statement3) {     case3(); } else {     case4(); } </pre>	<pre> if statement1:     case1()  elif statement2:     case2()  elif statement3:     case3()  else:     case4() </pre>
Code Representation of the “if/else if / .... /else” Control Block in the C Programming Language	Code Representation of the “if/else if / .... /else” Control Block in the Python Programming Language
<pre> if (statement1) {     case1(); } else if (statement2) {     case2(); } else if (statement3) {     case3(); } else {     case4(); } </pre>	<pre> if (statement1) {     case1(); } else if (statement2) {     case2(); } else if (statement3) {     case3(); } else {     case4(); } </pre>
Code Representation of the “if/else if / .... /else” Control Block in the C++ Programming Language	Code Representation of the “if/else if / .... /else” Control Block in the C# Programming Language





*Figure 4. Chart of “switch/case” Control Block in Programming*

The “switch/case” control block is shown in Figure 4 and Table 14 in general. How to use the “switch/case” control block is shown in Figure 4. In Table 14, on the other hand, the use of the “switch/case” control block in four separate programming languages (C, C++, C#, and Python) is shown with examples. In the “switch/case” structure, if the “variable” value does not equal any value, the commands in the “default” section will be executed. When the value of the variable specified for the “switch” structure matches one of the “case” statements, the matching loop block is executed regardless of the equality status of the “case” statements. In the “case” statements, it is ensured that the cycle is completed by placing the “break;” command at the

end of the software blocks. The statements next to the “case” statement must be constant. There are no variables in these statements. A “case” statement can be any integer, character, or string constant but cannot be a decimal value. A “switch” block cannot contain more than one case statement with the same constant value.

In the structure of the “switch/case” control block shown in Figure 4 and Table 14, if the “variable” value does not equal any value, the commands0() in the “default” section will be executed. When the value of the variable specified for the “switch” structure matches one of the “case” statements, the related commands() block will be executed.

*Table 14. Code Representation of the “switch/case” Loop on Different Programming Languages*

<pre>switch(variable) { case value1: commands1(); break; case value2: commands2(); break; .... case valueN: commandsN(); break; default: commands0; break; }</pre>	<pre>def function(int variable):     new(variable)={         value1: commands1(); break;         value2: commands2(); break;         ....         valueN: commandsN(); break;     }     return new.get(variable,         “switch+case”)</pre>
<p>Code Representation of the Control Block “switch/case” in the C Programming Language</p>	<p>Code Representation of the Control Block “switch/case” in the Python Programming Language</p>
<pre>switch(variable) { case value1: commands1(); break; case value2: commands2(); break; .... case valueN: commandsN(); break; default: commands0; break; }</pre>	<pre>switch(variable) { case value1: commands1(); break; case value2: commands2(); break; .... case valueN: commandsN(); break; default: commands0; break; }</pre>
<p>Code Representation of the Control Block “switch/case” in the C++ Programming Language</p>	<p>Code Representation of the Control Block “switch/case” in the C# Programming Language</p>

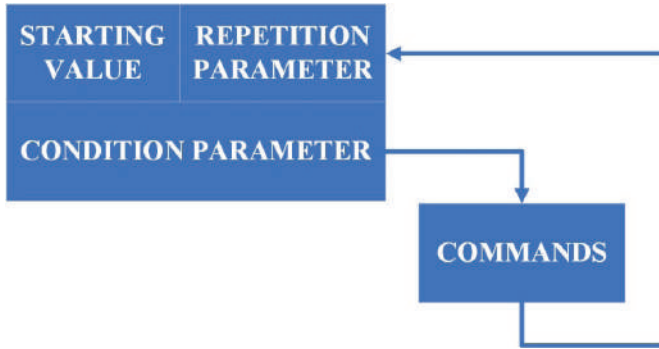


Figure 5. The diagram of the “for” Loop in Programming

Table 15. The Use of the “for” Loop on Different Programming Languages

<pre>for(start; condition; repetition) {     commands(); }</pre>	<pre>array = [1,2,3,4,5,6,7] for array_item_number (i) in array:     print(i)</pre>
Code Representation of the “for” Loop Block in the C Programming Language	Code Representation of the “for” Loop Block in the Python Programming Language
<pre>for(start; condition; repetition) {     commands(); }</pre>	<pre>for(start; condition; repetition) {     commands(); }</pre>
Code Representation of the “for” Loop Block in the C++ Programming Language	Code Representation of the “for” Loop Block in the C# Programming Language

Figure 5 shows how to use the “for” loop block. In Table 15, on the other hand, the use of the “for” control block in four different programming languages (C, C++, C#, and Python) is presented with examples.

In the “for” loop, when the loop is first entered, the first operation of the loop is performed with the starting value; then the condition parameter is checked, and if the condition statement is “TRUE”, the commands()(function or commands block) in the loop are executed. Once the operation is completed, the condition is checked again. With the repetition parameter,

the number of times the “for” loop will repeat or the number of times the loop will be executed is indicated.

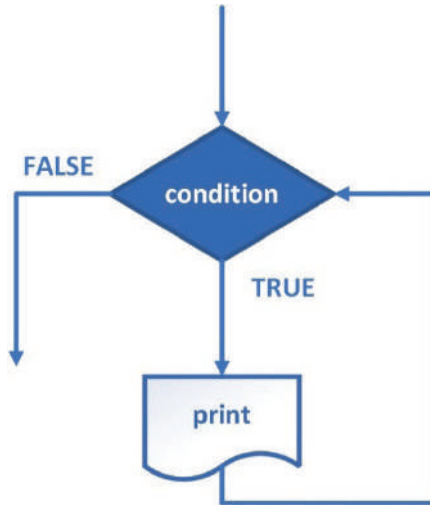


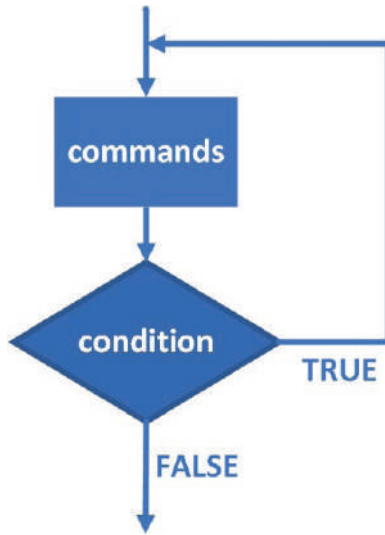
Figure 6. Diagram of the “while” loop in programming

Table 16. Code Representation of the “while” Loop in Different Programming Languages

<pre>while(condition) {     commands(); }</pre>	<pre>counter = 0 while counter&lt;determined_number:     commands()     print("operation value", i)</pre>
The Use of the “while” Loop Block in the C Programming Language	The Use of the “while” Loop Block in the Python Programming Language
<pre>while(condition) {     commands(); }</pre>	<pre>while(condition) {     commands(); }</pre>
The Use of the “while” Loop Block in the C++ Programming Language	The Use of the “while” Loop Block in the C# Programming Language

Representations of the “while” loop block were given in Figure 6 and Table 16. Figure 6 shows the way to use the “while” loop block. In Table 16, the use of the “while” control block in four separate programming languages (C, C++, C#, and Python) is shown with examples.

With the “while” loop, a command or command block can be used in more than one repetition operation. For example, with the while(1) statement, the state of the condition constantly returns TRUE. When the condition state is FALSE, the repetition of the loop stops.



*Figure 7. The diagram of the “do/while” Loop in Programming*

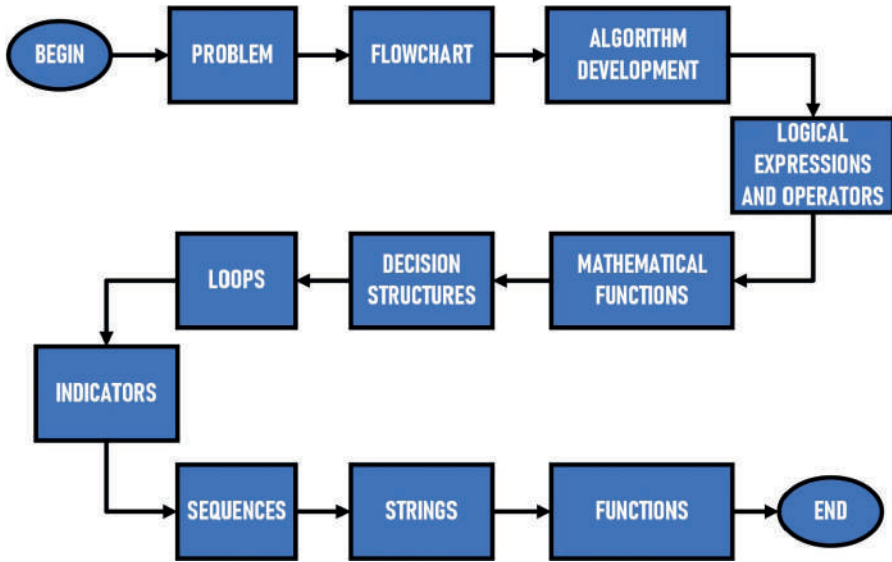
**Table 17. Code Representation of the “do/while” Loop in Different Programming Languages**

<pre>do {   counter_value++;   commands(); } while (counter_value &lt; determined_ number);</pre>	<pre>counter = 1 while True:   print(counter)   counter = counter + 1 if(counter &gt; determined_number):   break</pre>
The Use of the “do/while” Loop Block in the C Programming Language	The Use of the “do/while” Loop Block in the Python Programming Language
<pre>do {   counter_value++;   commands(); } while(counter_value &lt; determined_ number);</pre>	<pre>do {   counter_value++;   commands(); } while(counter_value &lt; determined_ number);</pre>
The Use of the “do/while” Loop Block in the C++ Programming Language	The Use of the “do/while” Loop Block in the C# Programming Language

The use of the “do/while” loop block were shown in Figure 7 and Table 17. Figure 7 shows the general usage of the “do/while” loop block. In Table 17, the use of the “do/while” control block in four different programming languages (C, C++, C#, and Python) is shown with codes. As long as the condition in operation is TRUE, the loop block or commands() function is executed. In the do/while loop, commands() execute the function once, regardless of the state of the condition (TRUE or FALSE). In short, regardless of the state of the condition, the commands() execute the function or loop block once.

## 7. CONCLUSION AND RECOMMENDATIONS

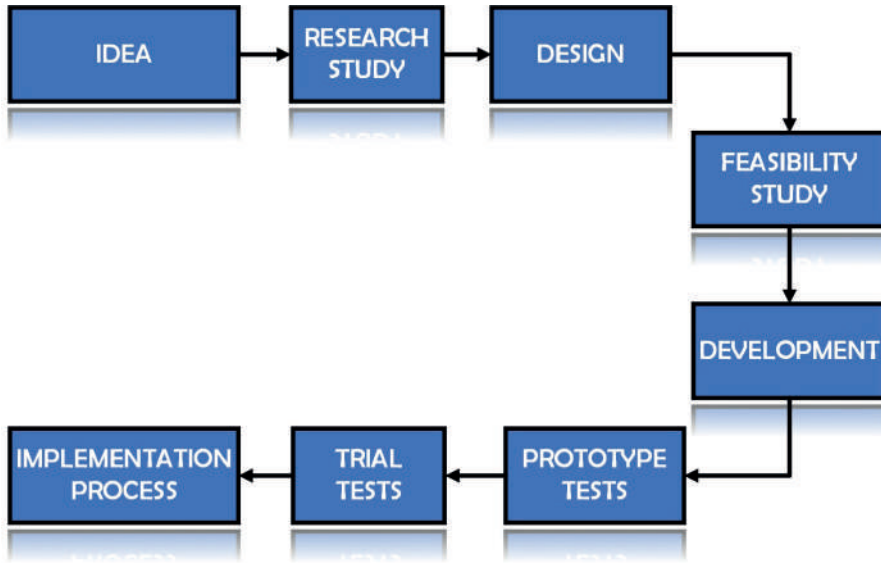
Algorithm design, which is considered together with the literature studies examined within the scope of this study, is an important topic when learning a programming language. Regardless of the programming language, the diagram of the steps to be followed when learning a language with an algorithm is given in Figure 8.



*Figure 8. Steps to follow when learning a programming language*

Based on Figure 8, it can be said that when learning a programming language, it is necessary to determine exactly what the problem means first in a work within the scope of the designs handled by algorithms. In order to address this problem, the algorithm development process begins with a flowchart to be developed. While an algorithm is being developed, logical expressions of the existing parameters and operators should be determined. Before the specified operator and expressions are processed on the algorithm, decision structures are processed by mathematical operations. Repetitions of the results of decision structures are created with loops created on the system. Then the creation of integrated functions related to the learning of indicators, arrays, and strings on the programming language is carried out. The gradual learning process in programming languages is completed by creating and running the related functions.

Figure 9 shows an idea study in terms of its operations on algorithms. In the flowchart given in Figure 9, research studies are started with the creation of the idea. Initial designs are created with the research carried out, and a feasibility study should be carried out in order to match the algorithm with the idea in terms of the continuity of the algorithm process. After the feasibility processes, prototype tests and trial tests should be performed along with the development of the idea. With the establishment of the implementation process, the relevant work is concluded.



*Figure 9. Creation of an algorithm process of an idea by flowcharts*

When the topics discussed in this chapter and the related literature are examined, it is seen that algorithms have an important role in the operation of a system. The selection of the best algorithm for a particular problem can be carried out with advanced programming knowledge and learning of algorithm design. In terms of their use on information and data, the importance of algorithms on integrated systems is gradually increasing today. It is observed that due to this increasing importance, systems become dependent on algorithms. The importance of algorithms, developed over time for integrated systems, in ensuring the control and security of the system both electronically and software is gradually increasing. States should use artificial intelligence in the use and development of algorithms on systems in terms of both technology security and software security. Considering the sectoral distribution of artificial intelligence, regardless of the sector, it has the opportunity to be applied in every field, from cyber security to autonomous vehicles, from virtual servers to mobile systems, from satellite communications to language processing applications and image processing.



## REFERENCES

- Deitel, H. M., & Deitel, P. J. (2004). C: How to program. Pearson Educación.
- Kelly, S. (2016). What Is Python? In Python, PyGame and Raspberry Pi Game Development (pp. 3-5). Apress, Berkeley, CA.
- Chollet, F. (2021). Deep learning with Python. Simon and Schuster.
- Hejlsberg, A., Wiltamuth, S., & Golde, P. (2003). C# language specification. Addison-Wesley Longman Publishing Co., Inc..
- Arnold, K., Gosling, J., & Holmes, D. (2005). The Java programming language. Addison Wesley Professional.
- Gavrilović, N., Arsić, A., Domazet, D., & Mishra, A. (2018). Algorithm for adaptive learning process and improving learners' skills in Java programming language. Computer Applications in Engineering Education, 26(5), 1362-1382.
- Namlı, N. A., & Şahin, M. C. (2017). Algoritma eğitiminin problem çözme becerisi üzerine etkisi. Recep Tayyip Erdoğan Üniversitesi Sosyal Bilimler Dergisi, 3(5), 135-153.
- Zurnacı, B., & Turan, Z. (2022). Türkiye'de okul öncesinde kodlama eğitimine ilişkin yapılan çalışmaların incelenmesi. Kocaeli Üniversitesi Eğitim Dergisi, 5(1), 258-286.
- Bui, N. D., Yu, Y., & Jiang, L. (2019). Bilateral dependency neural networks for cross-language algorithm classification. In 2019 IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER) (pp. 422-433). IEEE.
- Akçay, A., & Çoklar, A. N. (2016). Bilişsel becerilerin gelişimine yönelik bir öneri: Programlama eğitimi. Eğitim teknolojileri okumaları, 121-139.
- Türk, A. M., & Bilge, A. (2018). Bilgisayar Mühendisliği Eğitiminde Teknoloji Eğilimlerinin Takip Edilmesi. arXiv preprint arXiv:1807.07571.
- Çevik, K. K., & Dandıl, E. (2012). Yapay sinir ağları için net platformunda görsel bir eğitim yazılımının geliştirilmesi. Bilişim Teknolojileri Dergisi, 5(1), 19-28.
- Alaybeyoğlu, A., & Morkaya, Ö. (2006). Ülkemizdeki Bilgisayar Mühendisliği Lisans Eğitimi ile Yazılım Mühendisliği Lisans Eğitiminin Karşılaştırılması. Proceedings of the 3. Ulusal Elektrik Elektronik Bilgisayar Mühendislikleri Eğitimi Sempozyumu.
- Özden, N. (2008). A comparison of the misconceptions about the time-efficiency of algorithms by various profiles of computer-programming students. Computers & Education, 51(3), 1094-1102.
- Ozoran, D., Cagiltay, N., & Topalli, D. (2012). Using scratch in introduction to programming course for engineering students. In 2nd International Engineering Education Conference (IEEC2012) (Vol. 2, pp. 125-132).

- Özyurt, Ö., & Özyurt, H. (2016). Using Facebook to enhance learning experiences of students in computer programming at Introduction to Programming and Algorithm course. *Computer Applications in Engineering Education*, 24(4), 546-554.
- Balreira, D. G., Silveira, T. L. D., & Wickboldt, J. A. (2023). Investigating the impact of adopting Python and C languages for introductory engineering programming courses. *Computer Applications in Engineering Education*, 31(1), 47-62.
- Shnaider, P., Chernysheva, A., Khlopotov, M., & Babayants, C. (2023). Generation of Course Prerequisites and Learning Outcomes Using Machine Learning Methods. In *Artificial Intelligence in Education Technologies: New Development and Innovative Practices: Proceedings of 2022 3rd International Conference on Artificial Intelligence in Education Technology* (pp. 34-46). Singapore: Springer Nature Singapore.
- Chen, L., & Zhang, X. (2023). Research On the Teaching Method of Programming Course by Using Computational Thinking. *IC-ICAIE 2022, AHCS 9*, pp.383-389, 2023, DOI: 10.2991/978-94-6463-040-4\_58
- Gökoğlu, S. (2017). Programlama eğitiminde algoritma algısı: Bir metafor analizi. *Cumhuriyet International Journal of Education*, 6(1), 1-14.
- Akkaya, A., & Öztürk, G. (2020). Algoritma yazma ve öğrenimi hakkında meslek yüksekokulu öğrencilerinin görüşleri. *Balıkesir Üniversitesi Fen Bilimleri Enstitüsü Dergisi*, 22(1), 367-380.
- Alper, A., & Öztürk, S. (2019). Programlama öğretimindeki ters-yüz öğretim yönteminin öğrencilerin başarılarına, bilgisayara yönelik tutumuna ve kendi kendine öğrenme düzeylerine etkisi. *Bilim Eğitim Sanat ve Teknoloji Dergisi*, 3(1), 13-26.
- Yukselturk, E., & Altıok, S. (2016). Bilişim teknolojileri öğretmen adaylarının programlama öğretiminde Scratch aracının kullanımına ilişkin algıları. *Mersin Üniversitesi Eğitim Fakültesi Dergisi*, 12(1).
- Yalçınkaya, B., Dönmez, A. H., Aydın, F., Kayalı, N., & Sönmez, A. R., (2018). İlköğretim Çocuklarının Kodlama Algısı Üzerine Empirik Bir Analiz Çalışması ve Çocuk Kütüphanelerinde Uygulanmasının Önemi. 1. Uluslararası Çocuk Kütüphaneleri Sempozyumu (pp.126-138). Nevşehir, Turkey
- Karaman, U., & Filiz, S. (2019). Kodlama eğitimine yönelik tutum ölçeği'nin (KEYTÖ) geliştirilmesi. *Gelecek Vizyonlar Dergisi (fvj: Future Visions Journal)*, 3(2), 36-47.
- Pala, F. K., & Mıhçı-Türker, P. (2019). Öğretmen adaylarının programlama eğitimine yönelik görüşleri. *Journal of Theoretical Educational Science*, 12(1), 116-134.
- Aytekin, A., Sönmez Çakır, F., Yücel, Y. B. & Kulaözü, İ. (2018). Algoritmaların Hayatımızdaki Yeri ve Önemi. *Avrasya Sosyal ve Ekonomi Araştırmaları Dergisi*, 5 (7), 143-150 .

- Deniz, G., & Eryılmaz, S. (2019). Türkiye’de programlama eğitimi ile ilgili yapılan çalışmaların incelenmesi: Bir betimsel analiz çalışması. *Eğitimde Kuram ve Uygulama*, 15(4), 319-338.
- Kaban, A. (2021). Mobil Programlama Dersini Alan Öğretmen Adaylarının Mobil Programlama Öğrenimine Yönelik Görüşleri. *Ahi Evran Üniversitesi Kırşehir Eğitim Fakültesi Dergisi*, 22(1), 497-520.
- Bayraktar, A. (2021). Türkiye’de Bilgi ve Belge Yönetimi Müfredatlarında Okutulan Bilgisayar Programlarına ve Programlama Dillerine Yönelik Dersler. *Bilgi Ve Belge Araştırmaları*, (16), 103-131 .