# Chapter 12. The concept of an algorithm

## 12.1 The concept of an algorithm

Algorithms are one of the tools of modern technologies for modeling and visualizing processes and phenomena.

An algorithm is a sequence of actions leading from the source data to a given result.

Features of algorithms:

*Determinism - unambiguity:* each step of the algorithm is clearly (unambiguously) defined and does not allow ambiguous interpretation. After each step of the algorithm is completed, the next step is determined or a stop command is given, signaling the completion of the algorithm.

*Mass character - versatility:* algorithms are capable of solving not only one specific task, but also a whole class of similar tasks that differ in the values of the source data.

*Finiteness - limitation:* The algorithms guarantee to get the result in a finite time. However, there are algorithmically unsolvable problems for which it is impossible to build an algorithmic solution. For example, the problem of angle trisection (dividing an angle into three equal parts using a compass and ruler) [4].

Forms of representation of algorithms:

*Verbal description:* a textual explanation of the sequence of actions necessary to achieve the result.

*A program in a high-level algorithmic language:* a set of instructions written in a programming language.

*Flowcharts:* A graphical representation of the algorithm based on the use of various blocks and arrows that indicate the sequence of operations. Flowcharts are a visual block logical structure of an algorithm that facilitates its analysis and understanding. Below is a verbal algorithm for the transition of an avenue divided by a square, which divides the avenue into two one-way traffic sections. The transition takes place along the street that intersects the avenue with the square. The algorithm does not correspond to the properties of determinism and mass character, since it is applicable only for one intersection:

1. Look at the traffic light: green - see step 3, otherwise see step 2

2. We stand and wait

3. We look at the transport on the left and go to the middle of the road

4. Look at the traffic light: green - see step 6, otherwise see step 5

5. We stand and wait

6. We look at the transport on the right and go to the end of the road

7. The end of the algorithm

The efficiency and ease of use of algorithms are provided by the requirements for the forms – models of algorithms that they are represented by:

*Simplicity:* models should be clear and easy to understand.

*Versatility:* models must solve a wide class of problems. They must be flexible and adaptive to cope with different types of data and conditions, which ensures their wide application in various fields.

## 12.2 Normal Markov algorithms

Normal Markov algorithms (NAM) are a rigorous mathematical form of writing algorithms for processing character strings, which can be used to prove the solvability or insolvability of various problems. Normal Markov algorithms are rules for converting words in a given alphabet.

The alphabet is a nonempty finite set of characters. It can include numbers, punctuation marks.

A word is an arbitrary sequence of characters included in the alphabet.

For the correct operation of the algorithm, the following is set:

1. The alphabet of the algorithm
2. A finite set of acceptable substitutions – rules
3. The procedure for applying the rules

The substitution formula is the entry shown in formula (1) (reads "alpha replace with beta") α → β, where α is the left part of the rule, β is the right part of the rule. The rule prescribes to replace the alpha variable with the value of the beta variable.

Rules of operation of the Markov algorithm:

1. At each step, the substitution rules included in the algorithm are viewed from top to bottom.
2. The first of the rules applicable to the input word is selected. That is, a sequence of characters equal to the left side of the rule is found in the source word.
3. The part of the word equal to the left part of the rule is replaced by the right part of the rule.
4. The rules are again viewed from top to bottom to find the applicable rules.
5. If no rule can be applied to a word, the algorithm is considered complete.
6. The result of the algorithm is considered to be the word obtained after applying the rules.

An example of the Markov algorithm is presented below, where the given word "aaaa" must be processed using two rules and get the word "dd":

Word: aaaa

Rules: аа → ac, ac → d;

Solution: aaaa = acaa = acac = dac = dd

Using rules in Markov algorithms:

1. If the left part of a certain rule enters a word more than once, then only the leftmost occurrence is replaced at one step.
2. The right side of the rule may be empty. The result of applying such a rule is the removal of a sequence of characters from a word that match the left side of the rule.
3. The left side of the rule may be empty. In this case, the right part of the rule is added to the beginning of the word. Such a transformation is considered applicable to any word, so it must always be the last in the list of rules, otherwise all subsequent actions will be blocked. If such a rule exists, then at least one terminal rule is needed to complete the transformations [10].

A rule is called a terminal rule if, after its execution, the transformations stop, and the algorithm is considered completed. The terminal rule has its own designation: ↦. There may be several terminal rules in one algorithm, or none. The algorithm can also have one terminal rule. The example shows the operation of the Markov algorithm:

Word: aa

Rules: \*а → аа\*, \* ↦ , → \*;

Solution: аа = \*аа = аа\*а = аааа\* = аааа

Figure 12.1 shows second example of a normal Markov algorithm, in which eight rules (substitutions) work. There is an initial word Ferrari, you need to get the word Bugatti. After seven applications of the rules, the algorithm execution stops and the final word is obtained.

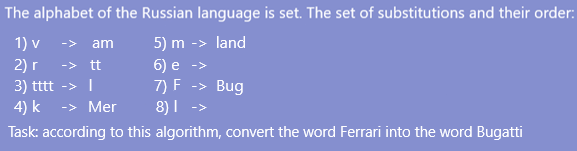


Figure 12.1 – An example of how the Markov algorithm works

## 12.3 Abstract Turing Machine

A Turing machine is a computing machine with linear memory that, according to specified transition rules, transforms input data using a sequence of elementary actions.

The main elements of the Turing machine (figure 12.2):

* 1. Control device
  2. The head of the machine
  3. The performing tape

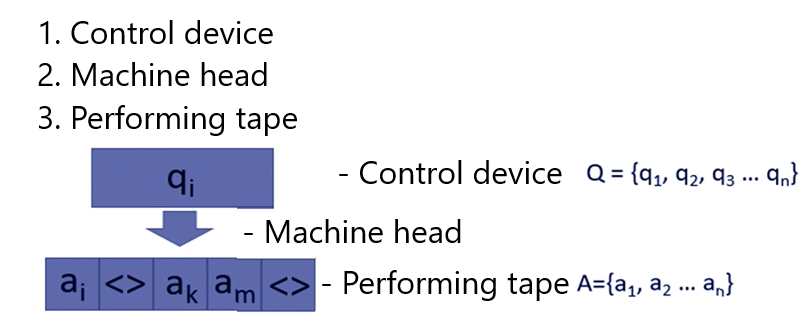
****

Figure 12.2 – The main elements of the Turing machine

The machine works as follows: the head can read the contents of a specific cell, erase, overwrite, and move a cell to the left or right and repeat reading or writing. The actions that the machine can perform are recorded in the form of a table, where the rows and columns are the external and internal alphabets, respectively. The external alphabet is a finite set, the elements of which are called letters or symbols (they can be initially displayed on the ribbon). The internal alphabet is a finite set of machine head states. The command that the machine must execute is determined by the intersection of the external and internal alphabets in the table. An example of a Turing machine program is shown in figure 12.3.

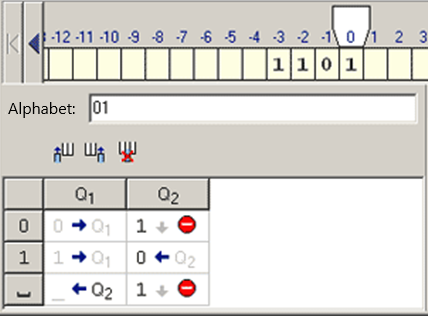


Figure 12.3 – An example of an abstract Turing machine

Before starting the work of the presented Turing machine, its head is above the first digit of the number on the left.

Control device Q = {q1, q2}. Alphabet A = {0, 1}.

The algorithm must add one to the number in the binary number system. All actions must be performed in the specified number system.

To add one to the number, you need to move the head of the car so that it is above the last digit of the number. To do this, there is a state q1, in which the head of the car will move to the right until it is outside the number. In this case, the head will be shifted to the left by one position and the q2 state will be triggered. In the case when the last digit of the number is zero, the machine overwrites it to one and shuts down, since changing the higher digits of the number is not required (110012 + 12 = 1100112). If the last digit of a number is one, the algorithm overwrites it to zero and processes the adjacent left digit in the same way. If the algorithm has processed all digits, including the most senior one (this is possible when all digits of the number are equal to one), one is written to the left of it (1112 + 12 = 10002).

Task: to create an algorithm for the operation of the Turing machine, which replaces all zeros with ones, and all ones with zeros. Before the machine starts working, its head is above the first digit on the right of the number. The original line on the tape = 101001 Control device Q = {q1} Alphabet = {0, 1} The algorithm is shown in figure 12.4.

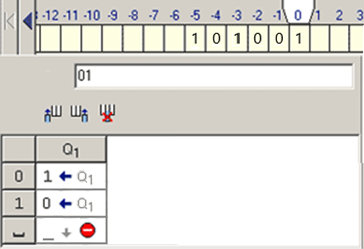


Figure 12.4 – Turing machine algorithm

The principle of operation of the Turing machine algorithm: the machine has a single state q1. It is responsible for changing the symbols and for moving the car's head to the adjacent left position. Initially, the head of the machine is located above the rightmost digit of the number on the tape. By processing the first character (rightmost), the algorithm checks it for zero: if the equality is true, the algorithm overwrites zero by one and moves the machine's head to the left by one position. If there was one instead of zero in the initial state of the cell, the algorithm overwrites it to zero and similarly shifts the head to the left by one position. When the algorithm has processed all the characters and the machine's head is over an empty cell, the algorithm will shut down.

Thus, studying the basic Markov and Turing algorithms allows the user to better understand the principles and methods of algorithms in programming. The normal Markov algorithm and the abstract Turing machine represent the basic foundation of algorithms and are the basis for developing more complex and efficient programming solutions.