UKOOPSPILKA HIGHER EDUCATIONAL INSTITUTION
“POLTAVA UNIVERSITY OF ECONOMICS AND TRADE”

EDUCATIONAL AND SCIENTIFIC INSTITUTE OF BUSINESS AND MODERN TECHNOLOGIES

FORM OF EDUCATION FULL-TIME
DEPARTMENT OF MATHEMATICAL MODELING AND SOCIAL INFORMATICS

Allowed for protection

Head of Department _____Iemets Oleg
(signature)

«_____»_____2021

EXPLANATORY NOTE TO BACHELOR'S WORK

on the topic

DEVELOPMENT OF SOFTWARE FOR THE SIMULATOR ON THE TOPIC «PREDICTIVE PARSING: SCHEME, PRINCIPLE OF OPERATION, APPLICATION» OF THE DISTANCE LEARNING COURSE «PROGRAMMING THEORY»

in the specialty 122 "Computer Science"

Executor of work Adnan Muhammad

_____ «___»_____2021
(signature)

Supervisor Ph.D., Assoc. Chernenko Oksana

_____ «___»_____2021
(signature)

POLTAVA 2021
CONTENT

LIST OF SYMBOLS, SYMBOLS, UNITS, ABBREVIATIONS, TERMS

Symbols, symbols, abbreviations, terms	Explanation of symbols, abbreviations, symbols
Predictive parser	a parser that works with recursive descent
FIRST(α)	set of terminals of the start line, derived from α. If $\alpha \overset{*}{\Rightarrow} \varepsilon$, then ε is also belongs to FIRST(α).
FOLLOW(A)	for nonterminal A it is a set of terminals a that can appear directly to the right from A in some sentient form, so the set of terminal a such that there is a generation $S\alpha \Rightarrow Aa\beta$ for some α and β.

INTRODUCTION

Modern telecommunication, information and computer technologies, first of all - electronic simulators are widely used for training. Therefore, the development of simulators is relevant now.

Simulators are used to train typical problem-solving skills.

They provide:

- sequential display of tasks of a given complexity on the selected topic;
- control over the user's actions to solve the proposed task;
- instant reaction to wrong actions;
- correction of user errors;
- demonstration of the correct solution of the problem;
- output of the final message about the results of the user's work (perhaps with recommendations or advice).

With the help of simulators it is possible to implement an individual approach in the organization of training. Thus, the advantages of this form of work include the following:

- generation of learning tasks and examples;
- the possibility of step-by-step control of the task;
- the possibility of working with the simulator in extracurricular activities;
- formation of tasks of different levels of complexity;
- lack of control by the teacher (control is provided to the information system), which significantly increases the efficiency of the teacher.

The purpose of the work is to develop of the simulator software on the topic "Predictive parsing: scheme, principle of operation, application" of the distance learning course "Programming Theory".

The object of development is the process of distance learning in mathematical disciplines.

The subject of development - the simulator on the topic "Predictive parsing: scheme, principle of operation, application".

The list of methods used is the use of predictive parsing. The Java programming language is used to develop the program.

The work consists of four sections. In the first section the problem statement is considered. The second section describes the relevance of using simulators, the distance learning solutions. The third section presents an overview of the material on the topic of work, algorithm of the simulator, justification of the choice of software. The forth section presents development of a block diagram, description of software implementation, description of software implementation

Volume of explanatory note: 61 pages, incl. main part - 48 pages, appendices - 1 page, sources - 12 titles.

1. STATEMENT OF THE PROBLEM

Project implementation is an important stage in specialist training.

In the process of project implementation, it is important that the student demonstrates the methods and approaches necessary for professional activity. The supervisor needs to focus the student on the task so that the practical part of the work prevails over the formal. It is necessary to focus the student's attention on the interpretation of the results.

Execution of the project allows to strengthen readiness of students for independent work in modern conditions.

The implementation of the project provides students with mastery of competencies, production functions, typical tasks and skills that must be possessed by a specialist in the specialty "Computer Science".

In many cases, careful developing of grammar, removal of it’s left recursion and it left factorization allow to receive the grammar that can be analyzed by parser, which uses a method of recursive descent and it doesn’t require rollback (predictive analyzer).

Non-recursive predictive analyzer can be constructed using explicit use of the stack instead implicit in the recursive calls. The key problem of predictive analysis is to identify products that need to be applied to non-terminal as well. A parse table can be used to find products.

The model of predictive parser which operates by table is shown in Fig. 1.1.

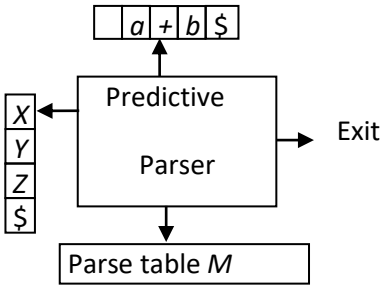


Figure 1.1 – Predictive analyzer scheme.

Analyzer has an input buffer, a stack , a parse table, and an output stream. Buffer contains the input line after which follows the right end marker $ - a sign of the end of the line. The stack contains a sequence of grammar symbols from $ on the bottom. In the beginning, the stack contains the initial grammar symbol directly above the character $. The parse table is a two-dimensional array $M[A,a]$, where A is non-terminal; a - terminal or symbol $.

Formulation of the problem:

1. Given an analyzer, use it to analyze the input stream. Namely: to learn to determine the products that need to be used at a certain step.
2. Learn to read and be able to compile a spreadsheet.
3. Create an algorithm for using a predictive analyzer.
4. Develop an algorithm simulator.

To provide a clear interface of the simulator, you must use the following structure:

- start page:
  - the theme of the simulator;
  - information about the developer;
  - button to go to the theoretical material;
  - button to go to training;
- passing the example:
  - condition of the problem;
  - tasks;
  - choice of answer;
  - go to the next step;
- result:
  - list of steps taken;
  - button to go to the home page;
  - exit button.

For the simulator to function, the following functions must be implemented:

- go from the home page to solve the example;
- moving on to the next question;
- checking the correctness of the answer;
- output error message;
- transition to the results of the simulator.

2. INFORMATION REVIEW

2.1. Relevance of using simulators

The simulator program provides:

- sequential display of tasks;
- control over the user's actions to solve the proposed task;
- instant reaction to wrong actions;
- demonstration of the correct solution of the problem;
- output of the final message about the results of the user's work.

Assessment of academic achievement is the purpose of monitoring programs. Such programs are also called test programs because they control on the basis of tests.

A test is a set of tasks of a special form designed to test the mastery of the material of a particular topic or several topics.

Test tasks differ from the usual ones in that they are short in content, require a clear answer and are formulated in such a way that it is not necessary to analyze its meaning to check the correctness of the answer. This is what allows you to use a computer for testing.

For example, most often a test task is provided together with several numbered options for ready-made answers. The user needs to specify the number of the answer, which, in his opinion, is correct. Checking the correctness of the task is to compare the specified number with a known number of the correct answer.

You can provide several correct answers to the task, then it is checked whether all of them are selected by the user.

If the task involves a short exact answer, for example in the form of a number or a word, the answer options are not given and the test is performed as a comparison of the provided and correct answers - whether they match or not.

Testing is a convenient method of mass verification, it is used without a computer. According to the same test for all, according to the same rules, at the same time the level of academic achievement of each student is determined [2].

The computer allows you to automate all stages of testing: the program registers the student, displays the tasks one by one, takes answers, checks their correctness and displays the final result (grade).

The use of a computer gives the test speed: the test result can be seen on the screen immediately after the last task.

Computer testing takes place while the user is working with the program, any interference from another person is completely excluded. The tests used for computer testing are performed by experienced professionals. Thus, computer testing allows us to get an objective assessment of our achievements.

The computer stores all the results that accompany the test: a list of suggested tasks; answers provided; time spent on each task, etc. If testing is done systematically, such results accumulate and can be tracked as the student progresses. So, computer testing provides a lot of useful information, it is informative.

Another advantage of computer-based testing is that the user is usually provided with all sorts of conveniences: it is enough to point to the mouse to select the answer; you can change the answer to another, the answer is accepted only on the alert; there is approximate information on the screen - how much time is left, how many tasks are left, etc [3].

Software training, systems learning, and desktop simulations are all synonymous. Every software application requires that users learn enough to employ its features and avoid frustration. There is no software that does everything for everybody and so there is always something more to learn, especially because new updates and releases occur; sometimes sporadically, sometimes frequently.

In many instances, it may be enough to show users how to perform a task when it is simple and intuitive. When the task requires multiple steps, however,

users will usually need to practice those steps in a safe environment. However, learners can always benefit by practicing the software tasks they need to perform rather than just watch a demo.

Today, there are many ways to create software simulations that demonstrate a series of steps. However, few of them allow you to easily let learners practice those steps. Many are free; others are included as part of an authoring tool; and some dedicated software simulation applications are powerful and more expensive. Which do you choose? It all depends on what your learners need to use the software that is required they know. You need to find the balance between not giving them enough and going overboard.

A simple online search will show that there are many available tools that will record your screen while you go through the steps you need to show learners. However, almost all of them don't create simulations that test learners on which steps to take; they just record the screen and produce a linear video. That can be very helpful but may not be enough [4].

True authoring tools for which software simulations are just one feature among many can allow:

- The creation of software simulations in more than one mode, including at least:
    - Demonstration, which may also be called View
    - Training, also called Practice or Try
    - Test, also called Assessment
- The ability to add interactions, shapes, images, and more to the simulation
- Publishing to HTML5
- Communication with a learning management system or learning record store

2.2. Distance learning solutions

The list of educational applications, platforms and resources below aim to help parents, teachers, schools and school administrators facilitate student learning and provide social care and interaction during periods of school closure. Most of the solutions curated are free and many cater to multiple languages. While these solutions do not carry UNESCO's explicit endorsement, they tend to have a wide reach, a strong user-base and evidence of impact. They are categorized based on distance learning needs, but most of them offer functionalities across multiple categories [5].

Digital learning management systems

- CenturyTech – Personal learning pathways with micro-lessons to address gaps in knowledge, challenge students and promote long-term memory retention.
- ClassDojo – Connects teachers with students and parents to build classroom communities.
- Edmodo – Tools and resources to manage classrooms and engage students remotely, offering a variety of languages.
- Edraak – Arabic language online education with resources for school learners and teachers.
- EkStep – Open learning platform with a collection of learning resources to support literacy and numeracy.
- Google Classroom – Helps classes connect remotely, communicate and stay-organized.
- Moodle – Community-driven and globally-supported open learning platform.
- Nafham – Arabic language online learning platform hosting educational video lessons that correspond with Egyptian and Syrian curricula.

- Paper Airplanes – Matches individuals with personal tutors for 12-16 week sessions conducted via video conferencing platforms, available in English and Turkish.
- Schoology – Tools to support instruction, learning, grading, collaboration and assessment.
- Seesaw – Enables the creation of collaborative and sharable digital learning portfolios and learning resources.
- Skooler – Tools to turn Microsoft Office software into an education platform.

Systems built for use on basic mobile phones

- Cell-Ed – Learner-centered, skills-based learning platform with offline options.
- Eneza Education - Revision and learning materials for basic feature phones.
- Funzi – Mobile learning service that supports teaching and training for large groups.
- KaiOS – Software that gives smartphone capabilities to inexpensive mobile phones and helps open portals to learning opportunities.
- Ubongo – Uses entertainment, mass media, and the connectivity of mobile devices to deliver localized learning to African families at low cost and scale,available in Kiswahili and English.
- Ustad Mobile – Access and share educational content offline.

Systems with strong offline functionality

- Kolibri – Learning application to support universal education, available in more than 20 languages.
- Rumie – Education tools and content to enable lifelong learning for underserved communities.
- Ustad Mobile – Access and share educational content offline.

Massive Open Online Course (MOOC) Platforms

- Alison – Online courses from experts, available in English, French, Spanish, Italian and Portuguese
- Canvas Network – Course catalogue accessible for free for teachers in order to support lifelong learning and professional development.
- Coursera – Online courses taught by instructors from well-recognized universities and companies.
- European Schoolnet Academy – Free online professional development courses for teachers in English, French, Italian and other European languages.
- EdX – Online courses from leading educational institutions.
- iCourse – Chinese and English language courses for university students.
- Future Learn – Online courses to help learners study, build professional skills and connect with experts.
- Icourses – Chinese language courses for university students.
- TED-Ed Earth School – Online lessons about nature made available continuously during a 5-week period between Earth Day (April 22nd) and World Environment Day (June 5th).
- Udemy – English, Spanish and Portuguese language courses on ICT skills and programming.
- XuetangX – Online courses provided by a collection of universities on different subjects in Chinese and English.

Self-directed learning content

- ABRA - Selection of 33 game-like activities in English and in French to promote reading comprehension and writing skills of early readers.
- British Council – English language learning resources, including games, reading, writing and listening exercises.

- Byju's – Learning application with large repositories of educational content tailored for different grades and learning levels.
- Code It – Helps children learn basic programming concepts through online courses, live webinars and other kid-friendly material. Available in English and German.
- Code.org – Wide range of coding resources categorized by subject for K12 students offered for free by a non-profit.
- Code Week – List of online resources to teach and learn computer coding, available in all EU languages.
- Discovery Education – Free educational resources and lessons about viruses and outbreaks for different grade levels.
- Duolingo – Application to support language learning. Supports numerous base and target languages.
- Edraak - A variety of resources for K-12 education in Arabic, targeting students, parents and teachers.
- Facebook Get Digital - Lesson plans, conversation starters, activities, videos and other resources for students to stay connected
- Feed the Monster – Android application in multiple languages to help teach children the fundamentals of reading, available in 48 languages.
- History of Africa – A nine-part BBC documentary series on the history of Africa based on UNESCO's General History of Africa book collection.
- Geekie – Portuguese language web-based platform that provides personalized educational content using adaptive learning technology.
- Khan Academy – Free online lessons and practice in math, sciences and humanities, as well as free tools for parents and teachers to track student progress. Available in 40+ languages, and aligned to national curriculum for over 10 countries.

- KitKit School - Tablet-based learning suite with a comprehensive curriculum spanning early childhood through early primary levels.
- LabXchange – Curated and user-created digital learning content delivered on an online platform that enables educational and research experiences.
- Madrasa – Resources and online lessons for STEM subjects in Arabic
- Mindspark – Adaptive online tutoring system that helps students practice and learn mathematics.
- Mosoteach – Chinese language application hosting cloud classes.
- Music Crab – Mobile application accessible for music education.
- OneCourse – Child-focused application to deliver reading, writing and numeracy education.
- Profuturo – Resources in different subject areas for students in English, Spanish, French and Portuguese.
- Polyup – Learning content to build math and gaining computational thinking skills for students in primary and early secondary school.
- Quizlet – Learning flashcards and games to support learning in multiple subjects, available in 15 languages.
- SDG Academy Library - A searchable library of more than 1,200 educational videos on sustainable development and related topics.
- Siyavula – Mathematics and physical sciences education aligned with South African curriculum.
- Smart History – Art history site with resources created by historians and academic contributors.
- YouTube – Huge repository of educational videos and learning channels.

Mobile reading applications

- African Storybook - Open access to picture storybooks in 189 African languages.
- Biblioteca Digital del Instituto Latinoamericano de la Comunicación Educativa – Offers free access to Spanish language works and book collections for students and teaching staff in schools and universities
- Global Digital Library – Digital storybooks and other reading materials easily accessible from mobile phones or computers. Available in 43 languages.
- Interactive Learning Program – Mobile app in Arabic to advance reading, writing and numeracy skills created by the United Nations Relief and Works Agency.
- Reads – Digital stories with illustrations in multiple languages.
- Room to Read – Resources to develop the literacy skills of children and youth with specialized content to support girls.
- StoryWeaver – Digital repository of multilingual stories for children.
- Worldreader – Digital books and stories accessible from mobile devices and functionality to support reading instruction. Available in 52 languages.

Collaboration platforms that support live-video communication

- Dingtalk – Communication platform that supports video conferencing, task and calendar management, attendance tracking and instant messaging.
- Lark – Collaboration suite of interconnected tools, including chat, calendar, creation and cloud storage, in Japanese, Korean, Italian and English
- Hangouts Meet – Video calls integrated with other Google's G-Suite tools.

- Teams – Chat, meet, call and collaboration features integrated with Microsoft Office software.
- Skype – Video and audio calls with talk, chat and collaboration features.
- WeChat Work – Messaging, content sharing and video/audio-conferencing tool with the possibility of including max. 300 participants, available in English and Chinese.
- WhatsApp – Video and audio calls, messaging and content sharing mobile application.
- Zoom – Cloud platform for video and audio conferencing, collaboration, chat and webinars.

Tools for teachers to create of digital learning content

- Thinglink – Tools to create interactive images, videos and other multimedia resources.
- Buncee – Supports the creation and sharing visual representations of learning content, including media-rich lessons, reports, newsletters and presentations.
- EdPuzzle – Video lesson creation software.
- EduCaixa - Courses in Spanish language to help teachers develop the skills and competencies of learners in areas such as communication, entrepreneurship, STEM and big data.
- Kaltura – Video management and creation tools with integration options for various learning management systems.
- Nearpod – Software to create lessons with informative and interactive assessment activities.
- Pear Deck – Facilitates the design of engaging instructional content with various integration features.
- Squigl – Content creation platform that transforms speech or text into animated videos.

- Trello - A visual collaboration tool used by teachers and professors for easier coursework planning, faculty collaboration, and classroom organization.

External repositories of distance learning solutions

- Brookings – A catalogue of nearly 3,000 learning innovations. Not all of them are distance learning solutions, but many of them offer digital education content.
- Common Sense Education – Tips and tools to support school closures and transitions to online and at-home learning.
- Commonweatlh of Learning – List of resources for policymakers, school and college administrators, teachers, parents and learners that will assist with student learning during the closure of educational institutions.
- Education Nation – Nordic countries have opened up their learning solutions for the world for free, supporting teachers and learners during the school closures.
- EdSurge – Community-driven list of edtech products, including many distance learning resources for students, teachers and schools, covering primary to post-secondary education levels.
- European Commission Resources – A collection of online platforms for teachers and educators, available in 23 EU languages.
- GDL Radio: a collection of radio and audio instruction resources.
- Global Business Coalition for Education – List of e-learning platforms, information sharing platform and communication platforms.
- Keep Learning Going – Extensive collection free tools, strategies, tips and best practices for teaching online from a coalition of USA-based education organizations. Includes descriptions of over 600+ digital learning solutions.

- Koulu.me – A collection of apps and pedagogical solutions curated by Finnish edtech companies to facilitate distance for pre-primary to upper secondary learners.
- Organisation internationale de la Francophonie: Resources for primary and secondary school students and teachers for learning and teaching French.
- Profuturo Resources: Spanish language resources in different subject areas for primary and secondary school students.
- UNEVOC Resources – Tools, guides, MOOCS and other resources collected by UNESCO's International Centre for Technical and Vocational Education and Training for continued learning in the area of TVET.
- UNHCR – An extensive list of over 600 distance learning solutions from the United Nations agency for refugees.

3. THEORETICAL PART

3.1. Review of material on the topic of work

Predictive parser is a parser that works with recursive descent. This is possible if the left recursion is removed from the grammar and it is left-factorized.

The analyzer runs by a program that works like that way. The program considers X - the symbol on the top of stack and a - the current input character. These two characters define analyzer actions. There are three options.

1. If $X = a = \$$ then the analyzer stops and reports of successful parse finish.

2. If $X = a \neq \$$, then the analyzer removes X from stack and promotes pointer of input flow to the next character.

3. If X - nonterminal, then the program considers record $M[X, a]$ from parse table M. This recording is X - products grammar, or recording error. For example, if $M[X, a] = \{X \to UVW\}$ then the analyzer replaces X on the top of stack into WVU (with U on top). We believe, that the analyzer just printed used products on exit. If, $M[X, a] = error$ the analyzer calls subprogram for error analysis [6].

Algorithm 1. Non-recursive predictive analysis

First, the analyzer is in the configuration with $\$S$ (on the top there is a start symbol S of grammar G) and a line $w\$$ in the input buffer.

To set up the indicator ip to the first symbol $w\$$;

repeat

Denote the symbol X at the top of stack,

and a - the symbol that is pointing to ip.

if X is non-terminal or $\$$ **then**

if $X = a$ **then** delete X from stack and move to ip

else error

else /* $X - non - teminal$ */

if $M[X,a] = X \to Y_1 Y_2 ... Y_k$ **then begin**

To delete X from stack; to put in stack

Y_k ,Y_{k-1} ,...,Y_1, Y_1 on the top of stack;

Output the products $X \to Y_1 Y_2 ... Y_k$

end

else error /* *entrance of table M is empty* */

until $X = \$$ /**stack is empty* */

Example 1

Consider the grammar of arithmetic expressions:

$E \to TE'$, $E' \to +TE' \mid \varepsilon$,

$T \to FT'$, $T' \to *FT' \mid \varepsilon$,

$F \to (E) \mid \mathbf{id}$.

For the input stream **id**+**id*****id**, the predictive parser makes the sequence of steps that has shown in Table 3.1.

The index entry points to the first character in the left column "Login". If you attentively analyze the actions of the analyzer, you can see that its output coincides with the sequence of products used in the left generation. If to the already read input character add symbols in a stack (from top to bottom), then we get left-sentient form in generation [7].

When you try to build predictive analyzer there are two useful functions connected with grammar G. These functions FIRST and FOLLOW also allow to build a table of predictive analysis for G, of course, if it's possible. The sets generated by these functions can also use in recovery after error.

Let be α an arbitrary line of grammar characters. **Function FIRST**(α) it is set of terminals of the start line, derived from α. If $\alpha \overset{*}{\Rightarrow} \varepsilon$, then ε is also belongs to FIRST(α) [6].

Table 3.1. Input Syntax

Stack	Entrance	Exit
$\$E$	**id**+**id***id$	
$\$E'$ T	**id**+**id***id$	$E \to TE'$
$\$E'$ T' F	**id**+**id***id$	$T \to FT'$
$\$E'$ T' **id**	**id**+**id***id$	$F \to$ **id**
$\$E'$ T'	+**id***id$	
$\$E'$	+**id***id$	$T' \to \varepsilon$
$\$E'$ T+	+**id***id$	$E' \to +TE'$
$\$E'$ T	**id***id$	
$\$E'$ T' F	**id***id$	$T \to FT'$
$\$E'$ T' **id**	**id***id$	$F \to$ **id**
$\$E'$ T'	***id$	
$\$E'$ T' F*	***id$	$T' \to *FT'$
$\$E'$ T' F	**id**$	
$\$E'$ T'**id**	**id**$	$F \to$ **id**
$\$E'$ T'	$	
$\$E'$	$	$T' \to \varepsilon$
$	$	$E' \to \varepsilon$

To construct $\mathrm{FIRST}(X)$ for all symbols of grammar X, we apply the following algorithm.

Algorithm 2. Construction of set FIRST **for grammar symbols.**

Step 1. If X is terminal then $\mathrm{FIRST}(X)=\{X\}$; if X is non-terminal then $\mathrm{FIRST}(X)=\{\}$.

Step 2. If there is a product $X \to \varepsilon$ then you need to add ε to $\mathrm{FIRST}(X)$.

Step 3 If X is non-terminal and we have a product $X \to Y_1Y_2...Y_k$, then you need to add a in $\mathrm{FIRST}(X)$, if for some i a $\mathrm{FIRST}(Y_i)$ and ε belongs to all sets $\mathrm{FIRST}(Y_1),...,\mathrm{FIRST}(Y_{i-1})$ that is $Y_1...Y_{i-1} \Rightarrow \varepsilon$. If ε belongs to $\mathrm{FIRST}(Y_i)$ for all $i=1,2,...,k,$ that is to add ε to $\mathrm{FIRST}(X)$

For example, everything that belongs to $\mathrm{FIRST}(Y_1)$ also belongs to $\mathrm{FIRST}(X)$. If we can't receive ε from Y_1, then we add nothing to $\mathrm{FIRST}(X)$, but if $Y_1 \Rightarrow \varepsilon$, then we add $\mathrm{FIRST}(Y_2)$ and so on.

Now we can calculate FIRST for any line $X_1X_2...X_n$ in such way.

$\mathrm{FIRST}(X_1X_2...X_n)=\{\}$.

Add to FIRST($X_1X_2...X_n$) all not ε –symbols FIRST(X_1).

Add also all non ε-symbols with FIRST(X_2), if $\varepsilon \in$ FIRST(X_1), all non ε-symbols with FIRST(X_3), if ε belongs as FIRST(X_1), as FIRST(X_2), and so on.

Finally, add ε to FIRST($X_1X_2...X_n$), if for all i FIRST(X_i) contains ε [7].

Function FOLLOW(A) for nonterminal A it is a set of terminals a that can appear directly to the right from A in some sentient form, so the set of terminal a such that there is a generation $S\alpha \Rightarrow Aa\beta$ for some α and β. Note that between A and a in the process of ejecting can appear nonterminals from which is derived ε . If A is first symbol from the right side of some sentient form, then \$ belongs to FOLLOW(A) [6].

To calculate FOLLOW(A) for the non-terminal A, we apply the following algorithm .

Algorithm 3 Building of FOLLOW(X) **for all *X*-non-terminals.**

Step 1. Place \$ in FOLLOW(S), where S is initial symbol and \$ is right end marker.

Step 2. If there is a product $A \to \alpha B\beta$, then all elements from the set FIRST(β), except of ε, add to FOLLOW(B).

Step 3. If there is a product $A \to \alpha B$ or $A \to \alpha B\beta$, where FIRST(β) contains ε ($\beta \overset{*}{\Rightarrow} \varepsilon$), then all elements of the set FOLLOW(A) add to FOLLOW(B).

Example 2.

We have a functions FIRST and FOLLOW for grammar from example 1.

$E \to TE'$, E' $\to$ +TE' $|\varepsilon$,

$T \to FT'$, $T' \to *FT' \,|\varepsilon$,

$F \to (E) \,|$**id**.

For it:

$\text{FIRST}(E) = \text{FIRST}(T) = \text{FIRST}(F) = \{(, \textbf{id}\}$;

$\text{FIRST}(E') = \{+, \varepsilon\}$;

$\text{FIRST}(T') = \{*, \varepsilon\}$;

$\text{FOLLOW}(E) = \text{FOLLOW}(E') = \{), \$\};$

$\text{FOLLOW}(T) = \text{FOLLOW}(T') = \{+,), \$\};$

$\text{FOLLOW}(F) = \{+, *,), \$\}.$

For example, **id** and left bracket added to FIRST(F) on the 3rd step of algorithm for FIRST , as and in accordance with FIRST(**id**) = {**id**} and FIRST('(') = { (} accordingly to step 1. On the 3rd step $i = 1$ when accordingly to product $T \to FT'$ to FIRST(T) add also **id** and left bracket. On the 2nd step FIRST(E') is included in ε.

In step 1, the calculation of sets FOLLOW to FOLLOW(E) include $. In step 2, using a product $F \to (E)$, to FOLLOW(E) a right bracket is also added . In step 3, the applied to the rule $E \to TE'$, in FOLLOW(E') are included $ and the right bracket. As $E' \Rightarrow \varepsilon$, they are also going in FOLLOW(T) . According to product $E \to TE'$ in step 2 in FOLLOW(T) include all from the FIRST(E') differently from ε [7].

To construct tables of predictive analysis in G grammar can be used an algorithm that is based on such idea. Suppose that $A \to \alpha$ - the production of grammar and $a \in \text{FIRST}(\alpha)$. Then the analyzer does deployment of A on α if the input symbol is a. Difficulties arise when $\alpha = \varepsilon$ either $\alpha \Rightarrow \varepsilon$. In this case, you need to deploy A into α, if the current input symbol belongs to FOLLOW(A), or if from the input stream received $ and $\$ \in \text{FOLLOW}(A)$.

Algorithm 4. Building a tables of predict analysis.

For each product of grammar $A \to \alpha$ to do steps 1st and 2nd.

Step 1. For each terminal a with $\text{FIRST}(\alpha)$ add $A \to \alpha$ in $M[A, a]$.

Step 2. If $\varepsilon \in \text{FIRST}(\alpha)$, then add $A \to \alpha$ in $M[A, b]$ for each terminal b with FOLLOW(A). If $\varepsilon \in \text{FIRST}(\alpha)$ and $\$ \in \text{FOLLOW}(A)$, add $A \to \alpha$ in $M[A, \$]$

Step 3 To accept that all uncertain M table inputs indicate the error.

Example 3.

We apply the algorithm 4 to the grammar of Example 1. Whereas $\text{FIRST}(TE') = \text{FIRST}(T) = \{(, \mathbf{id}\}$, in accordance to product $E \to TE'$ inputs $M[E,(]$ and $M[E,\mathbf{id}]$ are equal to $E \to TE'$.

According to the product $E' \to +TE'$ the entrance $M[E',+]$ is equal to $E' \to +TE'$. In accordance to product $E' \to \varepsilon$ inputs $M[E',)]$ and $M[E',\$]$ equal because $\text{FOLLOW}(E') = \{), \$\}$.

Table 3.2. Predictive analyzer for grammar from example 1.

Non-terminals	Input symbol					
	id	+	*	(	)	$
E	$E \to TE'$			$E \to T\ E'$		
E'		$E' \to +TE'$			$E' \to \varepsilon$	$E' \to \varepsilon$
T	$T \to FT'$			$T \to F\ T'$		
T'		$T' \to \varepsilon$	$T' \to *F\ T'$		$T' \to \varepsilon$	$T' \to \varepsilon$
F	$F \to \mathbf{id}$			$F \to (E)$		

This empty cells mean error non-empty containing products, through which replaced non-terminal on top of the stack [7].

3.2. Algorithmization of the problem

The main page displays the topic of the simulator, information about the developer and manager, it is suggested to open the theoretical material or go to the tests.

Step 1. Task: The analyzer is controlled by a program that works in this way. The program considers the X – symbol at the top of the stack and a – the current input symbol. These two symbols determine the action of the analyzer. There are three possibilities.

1. If $X=a=\$$,

Answer options:

- The analyzer stops and announces the successful completion of the analysis.
- The analyzer removes *X* from the stack and moves the input stream pointer to the next character.
- The program considers the record *M(X,a)* from the parsing table M. This entry is either an *X*-output grammar or an error entry. If *M(X,a)=error*, the analyzer calls the error analysis routine.

If the answer is the first option, then go to the next step. If not - there is a transition to reference information.

Step 2. Task: The analyzer is controlled by a program that works in this way. The program considers the X – symbol at the top of the stack and a – the current input symbol. These two symbols determine the action of the analyzer. There are three possibilities.

2. If *X=a≠$,*

Answer options:

- The analyzer stops and announces the successful completion of the analysis.
- The analyzer removes *X* from the stack and moves the input stream pointer to the next character.
- The program considers the record *M(X,a)* from the parsing table M. This entry is either an *X*-output grammar or an error entry. If *M(X,a)=error*, the analyzer calls the error analysis routine.

If the answer is the second option, then go to the next step. If not - there is a transition to reference information.

Step 3. Task: The analyzer is controlled by a program that works in this way. The program considers the X – symbol at the top of the stack and a – the current input symbol. These two symbols determine the action of the analyzer. There are three possibilities.

3. If *X* is a non-terminal,

Answer options:

- The analyzer stops and announces the successful completion of the analysis.
- The analyzer removes *X* from the stack and moves the input stream pointer to the next character.
- The program considers the record *M(X,a)* from the parsing table M. This entry is either an *X*-output grammar or an error entry. If *M(X,a)=error*, the analyzer calls the error analysis routine.

If the answer is the third option, then go to the next step. If not - there is a transition to reference information.

Step 4. Task: Consider the grammar of arithmetic expressions:

$E \to TE'$, $E' \to +TE' \mid \varepsilon$,

$T \to FT'$, $T' \to *FT' \mid \varepsilon$,

$F \to (E) \mid \mathbf{id}$.

For the input stream $\mathbf{id} + \mathbf{id} * \mathbf{id}$, the predictive parser makes the sequence of steps that has shown in Table.

The index entry points to the first character in the left column "Login". If you attentively analyze the actions of the analyzer, you can see that its output coincides with the sequence of products used in the left generation. If to the already read input character add symbols in a stack (from top to bottom), then we get left-sentient form in generation.

Fill in the column Exit:

Stack	Entrance	Exit
\$*E*	**id**+**id*****id**\$	
\$*E*’ *T*	**id**+**id*****id**\$	
\$*E*’ *T*’ *F*	**id**+**id*****id**\$	
\$*E*’ *T*’ **id**	**id**+**id*****id**\$	
\$*E*’ *T*’	+**id*****id**\$	
\$E’	+**id*****id**\$	
\$*E*’ *T*+	+**id*****id**\$	
\$*E*’ *T*	**id*****id**\$	
\$*E*’ *T*’ *F*	**id*****id**\$	
\$*E*’ *T*’ **id**	**id*****id**\$	

$*E*' *T*'	***id**$	
$*E*' *T*' *F* *	***id**$	
$*E*' *T*' *F*	**id**$	
$*E*' *T*'**id**	**id**$	
$*E*' *T*'	$	
$*E*'	$	
$	$	

You can view for help Algorithm 1. Non-recursive predictive analysis.

If the answer is correct, then go to the next step. If not - there is a transition to reference information.

Step 5. Task: Everything that belongs to FIRST(Y_1) also belongs to FIRST(X). If we can't receive ε from Y_1, then we add nothing to FIRST(X), but if $Y_1 \Rightarrow \varepsilon$, then we add FIRST($Y_2$) and so on.

Now we can calculate FIRST for any line $X_1X_2...X_n$ in such way.

Set the sequence:

1. $\text{FIRST}(X_1X_2...X_n) = \{\}$.
2. Add to FIRST($X_1X_2...X_n$) all not ε –symbols FIRST(X_1).
3. Add also all non ε-symbols with FIRST(X_2), if $\varepsilon \in$ FIRST(X_1), all non ε-symbols with FIRST(X_3), if ε belongs as FIRST(X_1), as FIRST(X_2), and so on.
4. Add ε to FIRST($X_1X_2...X_n$), if for all i FIRST(X_i) contains ε.

You can view for help Algorithm 2. Construction of set FIRST(X) for grammar symbols.

If the answer is correct, then go to the next step. If not - there is a transition to reference information.

Step 6. Task: We have a functions FIRST and FOLLOW for grammar.

$E \to TE'$, E' $\to$ +TE' $|\varepsilon$,

$T \to FT'$, $T' \to *FT' \mid \varepsilon$,

$F \to (E) \mid$ **id**.

For it fill the cells:

$\text{FIRST}(E) = \text{FIRST}(T) = \text{FIRST}(F) = \{\_\_\_\_\_\};$

$\text{FIRST}(E') = \{\_\_\_\_\_\};$

$\text{FIRST}(T') = \{\_\_\_\_\_\};$

$\text{FOLLOW}(E) = \text{FOLLOW}(E') = \{\_\_\_\_\_\};$

$\text{FOLLOW}(T) = \text{FOLLOW}(T') = \{\_\_\_\_\_\};$

$\text{FOLLOW}(F) = \{\_\_\_\_\_\}.$

You can view for help Algorithm 3. Building of $\text{FOLLOW}(X)$ for all X-non-terminals.

If the answer is correct, then go to the next step. If not - there is a transition to reference information.

Step 7. Task: We apply the algorithm 4 to the grammar

$E \to TE'$, $E' \to +TE' \mid \varepsilon$,

$T \to FT', T' \to *FT' \mid \varepsilon$,

$F \to (E) \mid \mathbf{id}$.

Whereas $\text{FIRST}(TE') = \text{FIRST}(T) = \{(, \mathbf{id}\}$, in accordance to product $E \to TE'$ inputs $M[E,(]$ and $M[E, \mathbf{id}]$ are equal to $E \to TE'$.

According to the product $E' \to +TE'$ the entrance $M[E',+]$ is equal to $E' \to +TE'$. In accordance to product $E' \to \varepsilon$ inputs $M[E',)]$ and $M[E',\$]$ equal because $\text{FOLLOW}(E') = \{), \$\}$.

Fill the cells in table:

Non-terminals	Input symbol					
	id	+	*	(	)	*$*
E	$E \to \_\_\_$			$E \to \_\_\_$		
E'		$E' \to \_\_\_$			$E' \to \_\_\_$	$E' \to \_\_$
T	$T \to \_\_\_$			$T \to \_\_\_$		
T'		$T' \to \_\_\_$	$T' \to \_\_\_$			
F	$F \to \_\_\_$			$F \to \_\_\_$	$T' \to \_\_\_$	$T' \to \_\_$

You can view for help Algorithm 4. Building a tables of predict analysis.

If the answer is correct, then go to the next step. If not - there is a transition to reference information.

Step 8. Completion message: "Congratulations! You have completed the simulator on the topic "Predictive parsing: scheme, principle of operation, application" of the distance learning course "Programming Theory".

The start window opens.

3.3. Justification of the choice of software

Java is not only a language but an ecosystem of tools covering almost everything you may need for Java development. This includes:

Java Development Kit (JDK) – with that and a standard Notebook app you can write and run/compile Java code

Java Runtime Environment (JRE) – software distribution tool containing a stand-alone Java Virtual Machine, the Java standard library (Java Class Library), and a configuration tool

Integrated Development Environment (IDE) – tools that help you run, edit, and compile your code. IntelliJ IDEA, Eclipse, and NetBeans are the most popular ones [8].

Java can be found anywhere you look. It’s a primary language for Android development. You will find it in web applications, governmental websites, and big data technologies such as Hadoop and Apache Storm. And it’s also a classic choice for scientific projects, especially natural language processing. Java was dominating mobile even in pre-smartphone days – first mobile games in the early 2000s were mostly made in Java. So, it’s fair to say that Java, thanks to its long history, has earned its place in the Programming Hall of Fame. TIOBE index, one of the most reputable programming rankings in the world, uses search engine results for calculation. Despite the growing

popularity of Go and Python, Java has remained at the top of the list for more than a decade.

Though no longer the only officially supported language for Android development and, of course, far from the only choice for web programming, Java keeps pace with the alternatives. And since that's not only thanks to its respectable age, let's explore what advantages Java has to offer [9].

Object-oriented programming

Java embraces object-oriented programming (OOP) – a coding concept in which you not only define the type of data and its structure, but also the set of functions applied to it. This way, your data structure becomes an object that can now be manipulated to create relationships between different objects.

In contrast to another approach – procedural programming – where you have to follow a sequence of instructions using variables and functions, OOP allows you to group these variables and functions by context thus labeling them and referring to functions in the context of each specific object.

Why is OOP an advantage?

- You can easily reuse objects in other programs
- It prevents errors by having objects hide some information that shouldn't be easily accessed
- It makes programs more organized and pre-planned, even the bigger ones
- It offers simple maintenance and legacy code modernization

High-level language with simple syntax and a mild learning curve

Java is a high-level language, meaning that it closely resembles human language. In contrast to low-level languages that resemble machine code, high-level languages have to be converted using compilers or interpreters. This simplifies development, making a language easier to write, read, and maintain.

Standard for enterprise computing

Enterprise applications are Java's greatest asset. It started back in the 90s when organizations began looking for robust programming tools that weren't C.

Java supports a plethora of libraries – building blocks of any enterprise system – that help developers create any function a company may need. The vast talent pool also helps – Java is the language used for introduction to computer programming in most schools and universities. Besides, its integration capabilities are impressive as most of the hosting providers support Java. Last but not least, Java is comparatively cheap to maintain since you don't have to depend on a specific hardware infrastructure and can run your servers on any type of machine you may have.

Shortage of security risks

You may encounter the notion that Java is a secure language but that's not entirely true. The language itself doesn't protect you from vulnerabilities, but some of its features can save you from common security flaws. First, compared to C, Java doesn't have pointers. A pointer is an object that stores the memory address of another value that can cause unauthorized access to memory. Second, it has a Security Manager, a security policy created for each application where you can specify access rules. This allows you to run Java applications in a "sandbox," eliminating risks of harm.

Platform-independency (Write Once Run Anywhere)

Write Once Run Anywhere (WORA) is a popular programming catchphrase introduced by Sun Microsystems to describe Java's cross-platform capabilities. It meant you could create a Java program on, let's say, Windows, compile it to bytecode, and run the application on any other platform that supports a Java Virtual Machine (JVM). In this case, a JVM serves as an abstraction level between the code and the hardware.

All major operating systems including Windows, Mac OS, and Linux support JVM. And unless you're writing a program that relies mostly on platform-specific features and UI, you can share – maybe not all – but a big chunk of bytecode.

Distributed language for easy remote collaboration

Java was designed as a distributed language meaning that it has an integrated mechanism for sharing data and programs among multiple computers for improved performance and efficiency.

Automatic memory management

Java developers don’t have to worry about manually writing code for memory management tasks thanks to automatic memory management (AMM), also used in the Swift programming language, and garbage collection, an application that automatically handles allocation and deallocation of memory. What exactly does it mean?

A program’s effectiveness is directly linked to memory – and memory is limited. By using languages with manual management, developers risk forgetting to allocate memory resulting in increased memory footprint and lagging. A garbage collector can locate objects that are no longer referenced by your program and remove them. Despite the fact that it affects your program’s CPU, you can reduce or prevent it with smart optimization and tuning.

Multithreading

In programming, a thread is the smallest unit of processing. To maximize utilization of CPU time, Java allows you to run these threads simultaneously – in a process called multithreading.

Threads share the same memory area so switching between them takes little time. They are also independent, so if one thread faces exception, it doesn’t affect other threads. This is especially useful for gaming and animation-heavy programs.

Stability and massive community

Java has survived to a respectable age thanks to the community, Oracle’s support, and the cornucopia of applications and languages that keep running on JVM. Besides, new versions of Java are regularly released with fresh, interesting features [10].

4. PRACTICAL PART

4.1. Development of a block diagram to be programmed

Figure 4.1 shows a block diagram of the algorithm of the simulator.

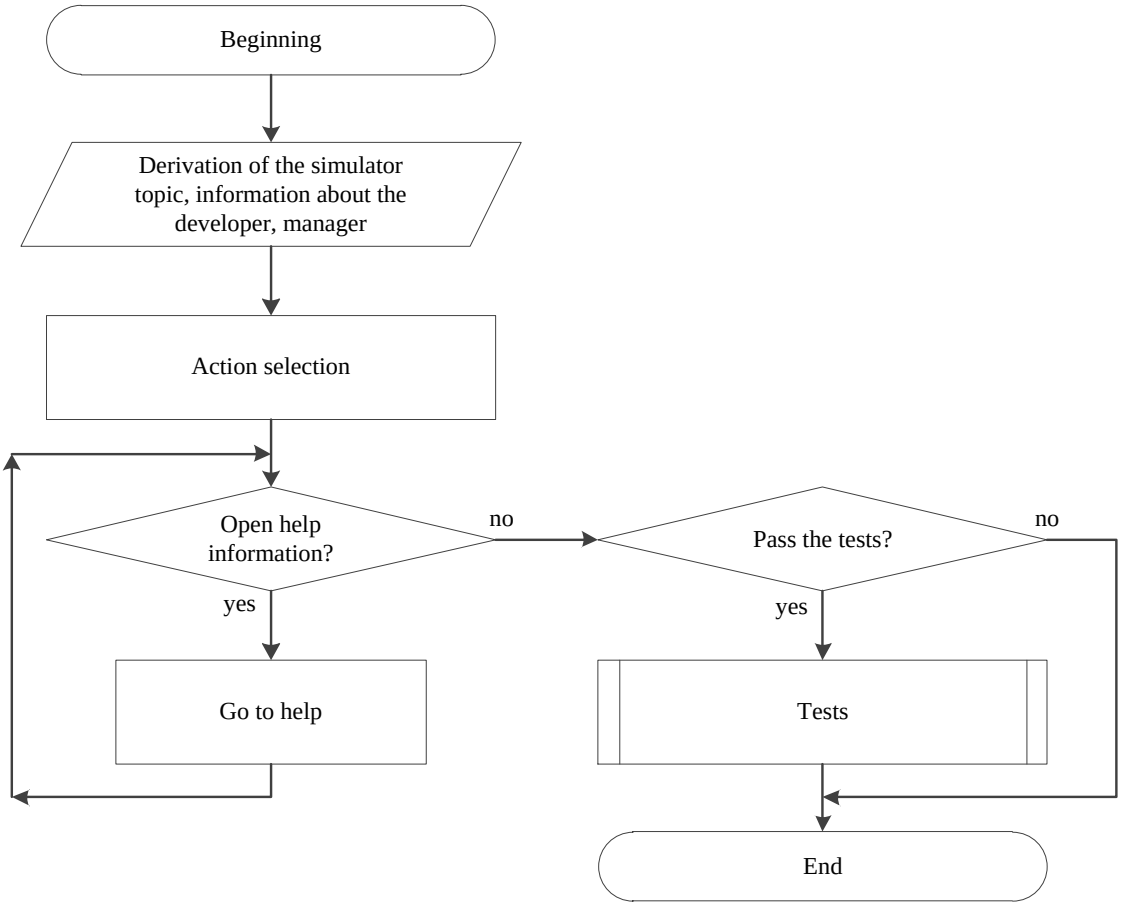


Figure 4.1 - Block diagram of the algorithm of the simulator

Figures 4.2 – 4.5 shows a block diagram of the algorithm for passing the tests.

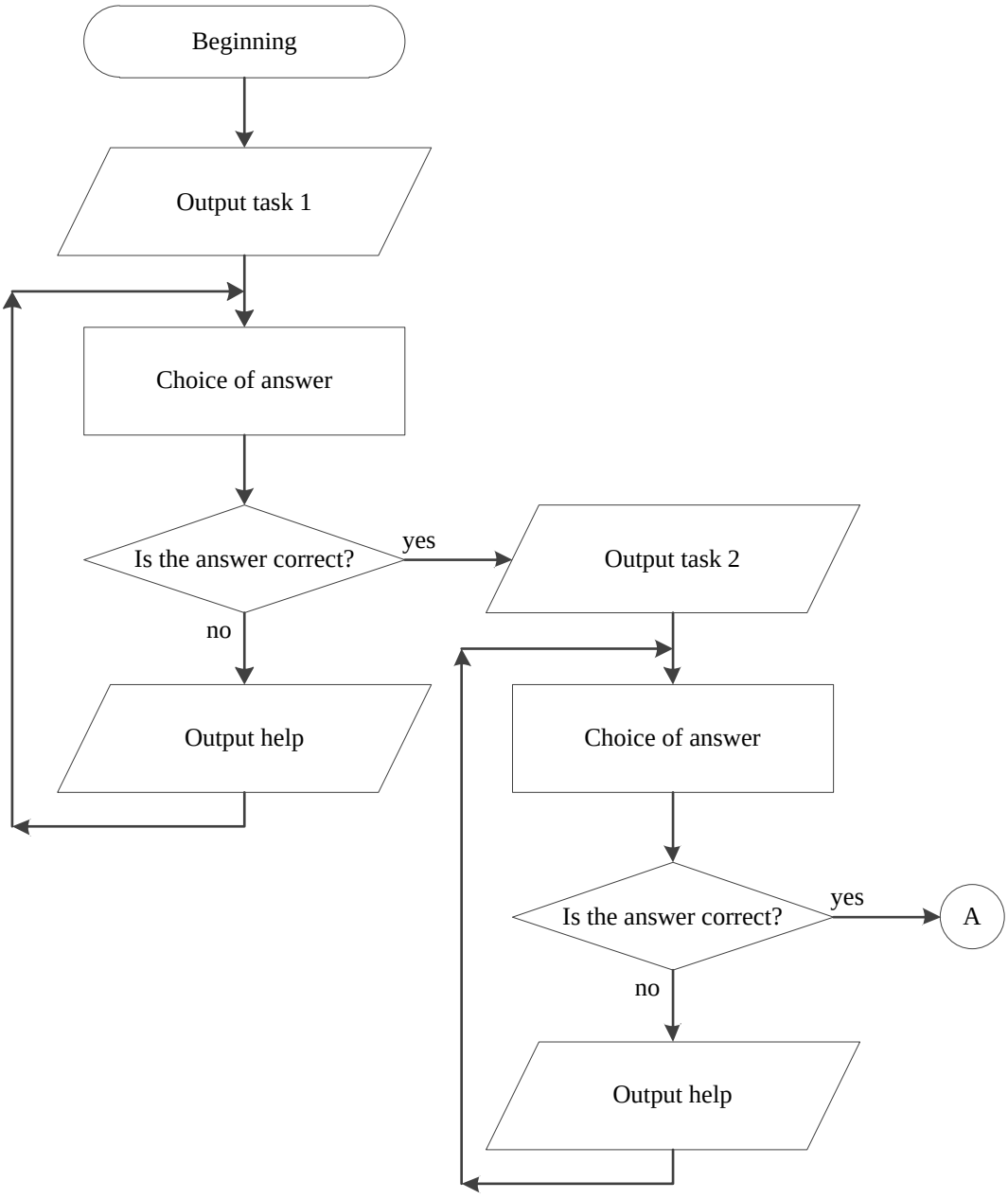


Figure 4.2 - Block diagram of the algorithm for passing the tests

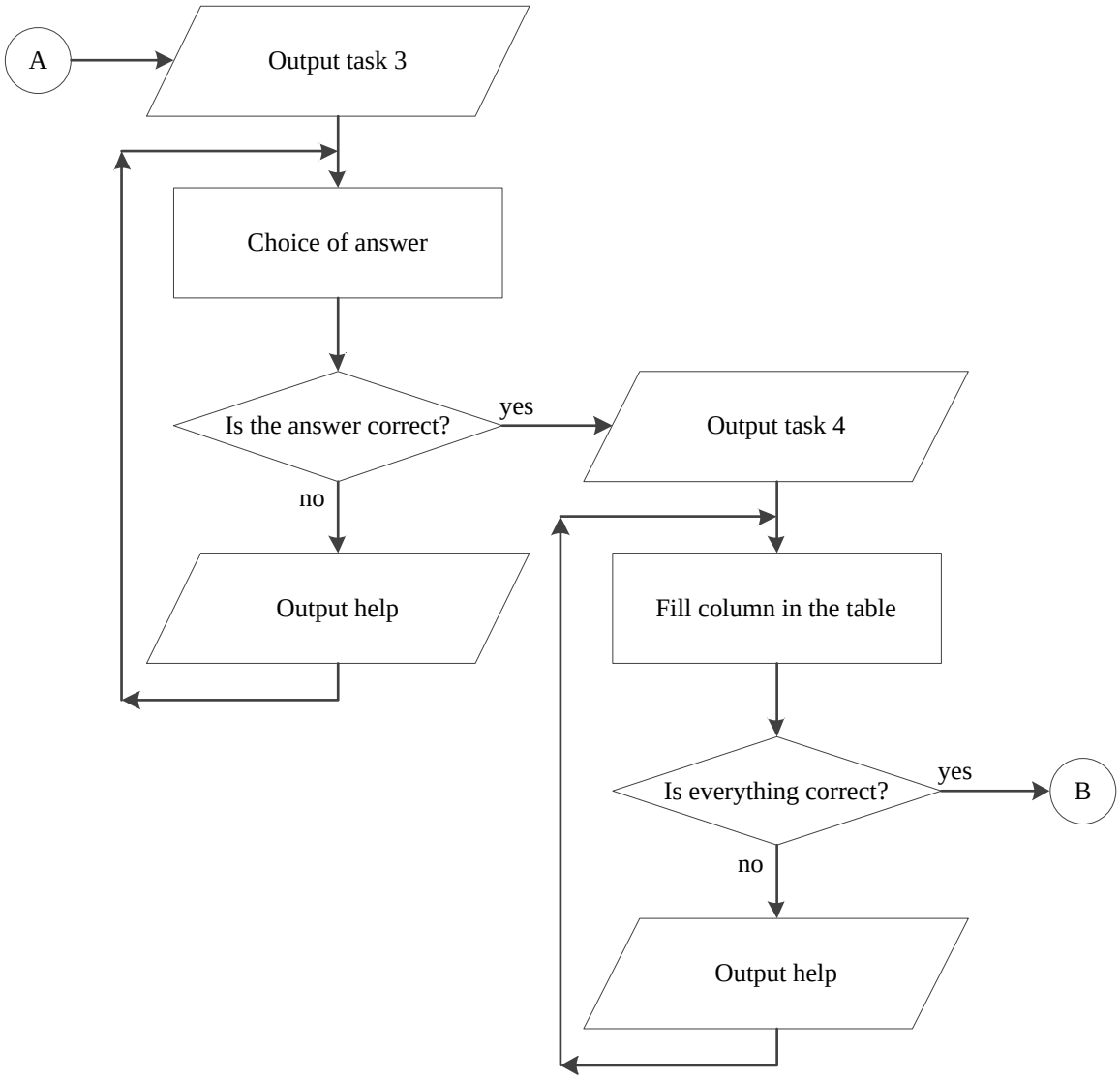


Figure 4.3 - Continuation of block diagram of the algorithm for passing the tests

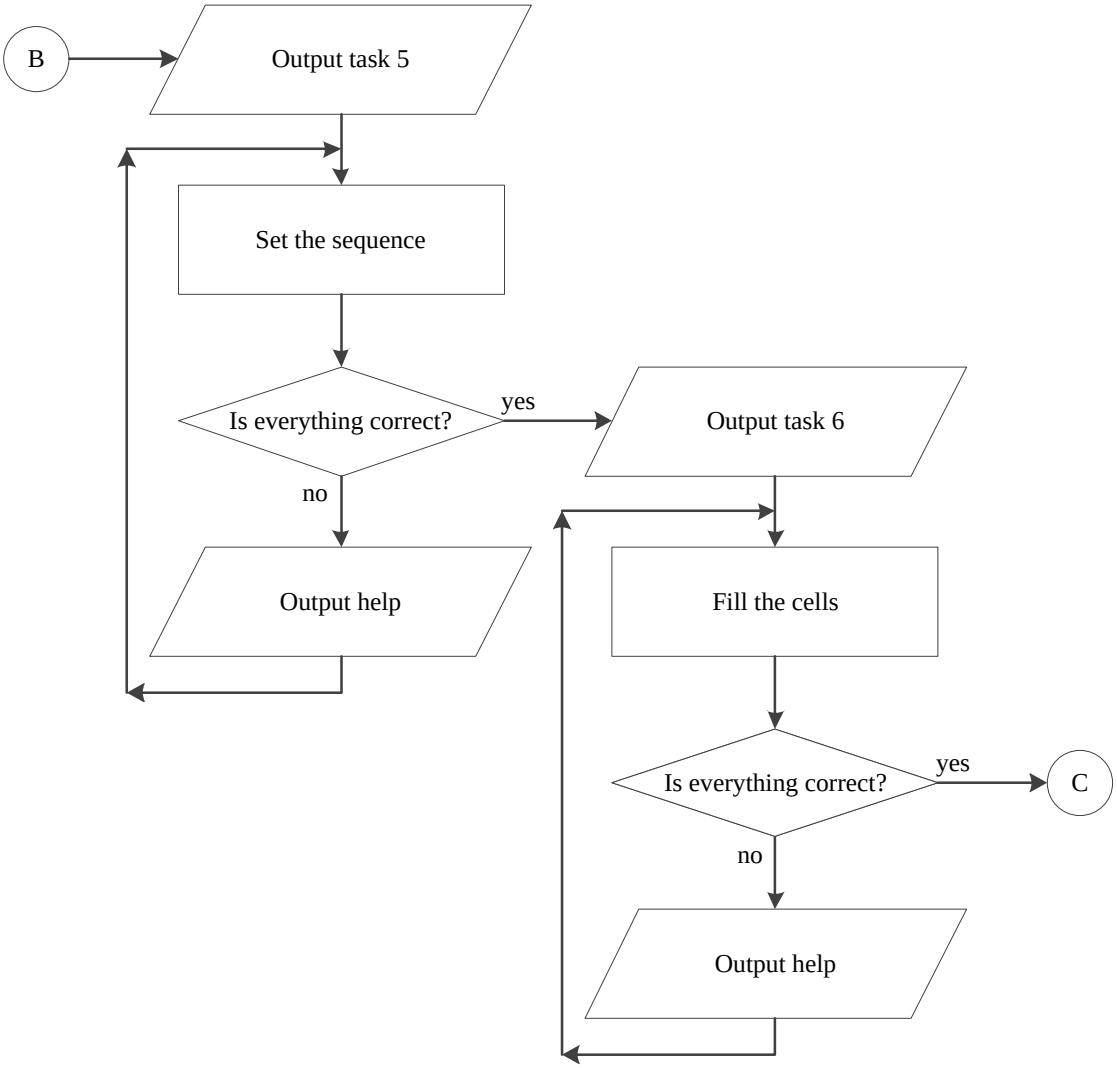


Figure 4.4 - Continuation of block diagram of the algorithm for passing the tests

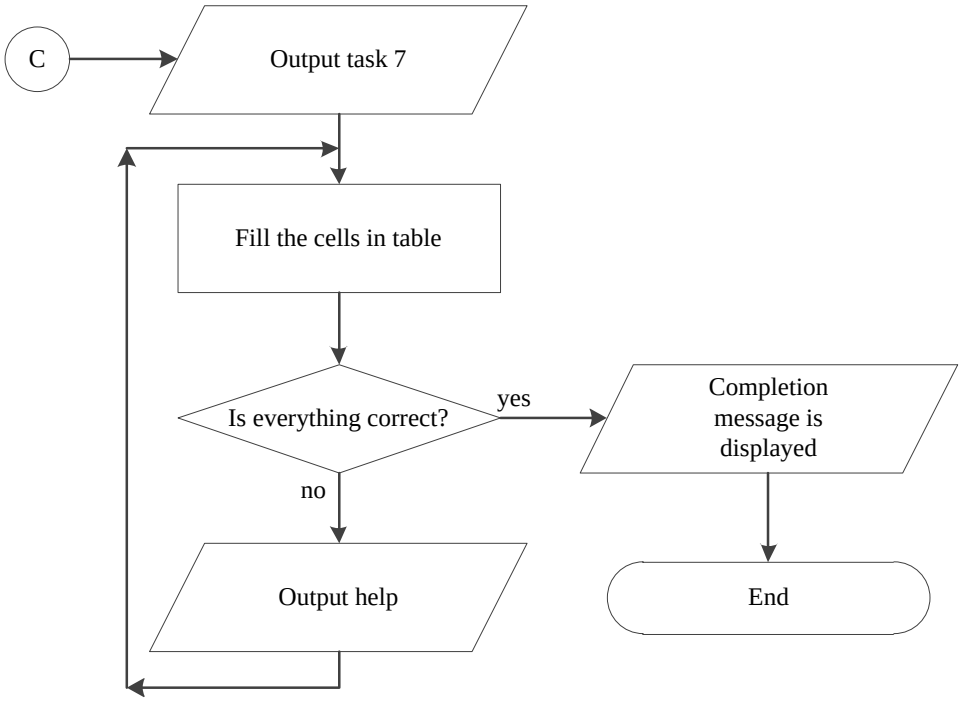


Figure 4.5 - Continuation of block diagram of the algorithm for passing the tests

4.2. Description of software implementation

After creating the project of the program, Package images was added, in which all pictures are stored, and Applet.java, in which you can create a graphic view of the simulator and develop its functionality(Fig. 4.3) [11].

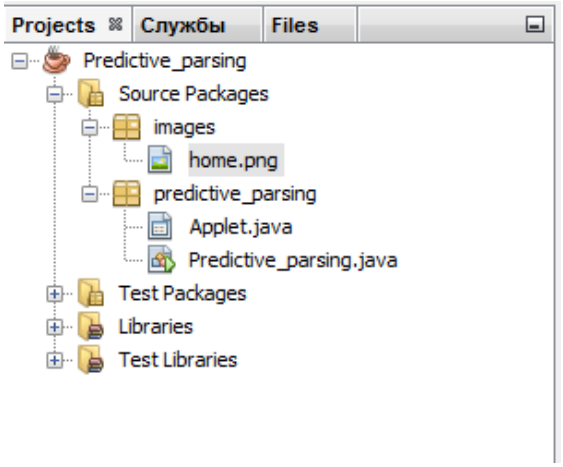


Figure 4.3 – Projects files

Now JPanel panels are created on the JApplet form, on which graphic elements will be placed. Also set the Layout property of the panels to be switched, set the Card Layout (Fig. 4.4) [11].

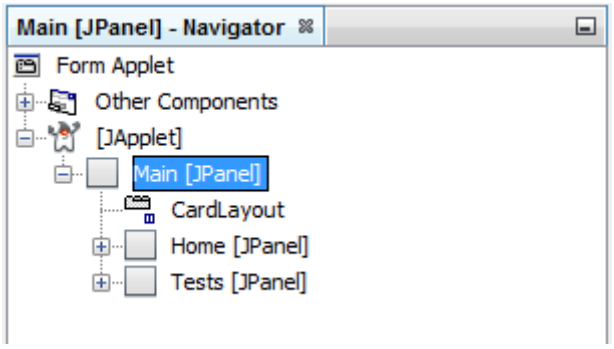


Figure 4.4 - Navigator

Using the graphic elements Label and Button (Fig. 4.5) was made the main page of the simulator (Fig. 4.6).

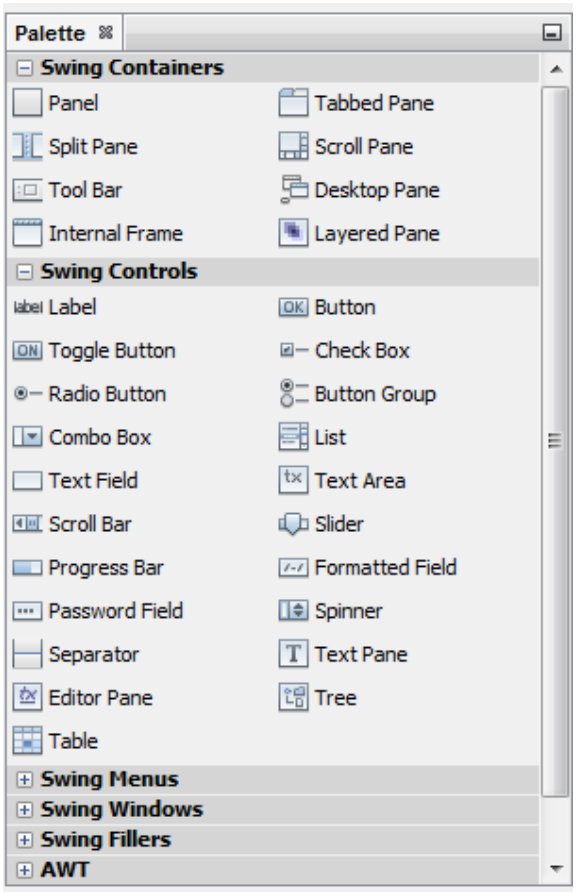


Figure 4.5 - Palette

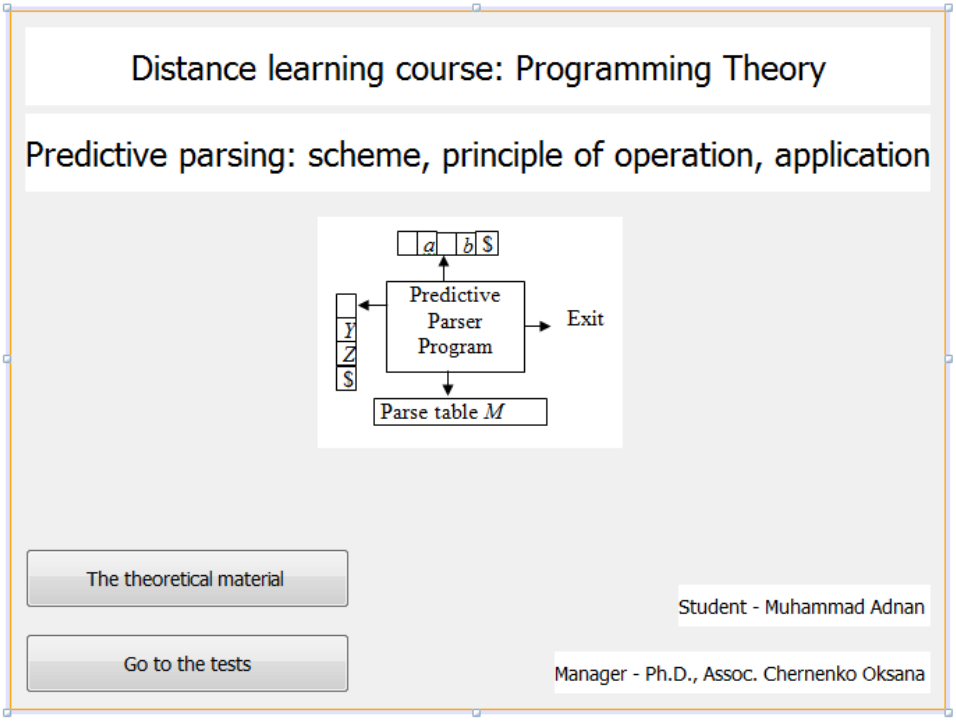


Figure 4.6 – Panel Home

To further display the steps of the algorithm, a panel is placed and Card Layout is assigned to it. Buttons are created to switch between steps (Fig. 4.7).

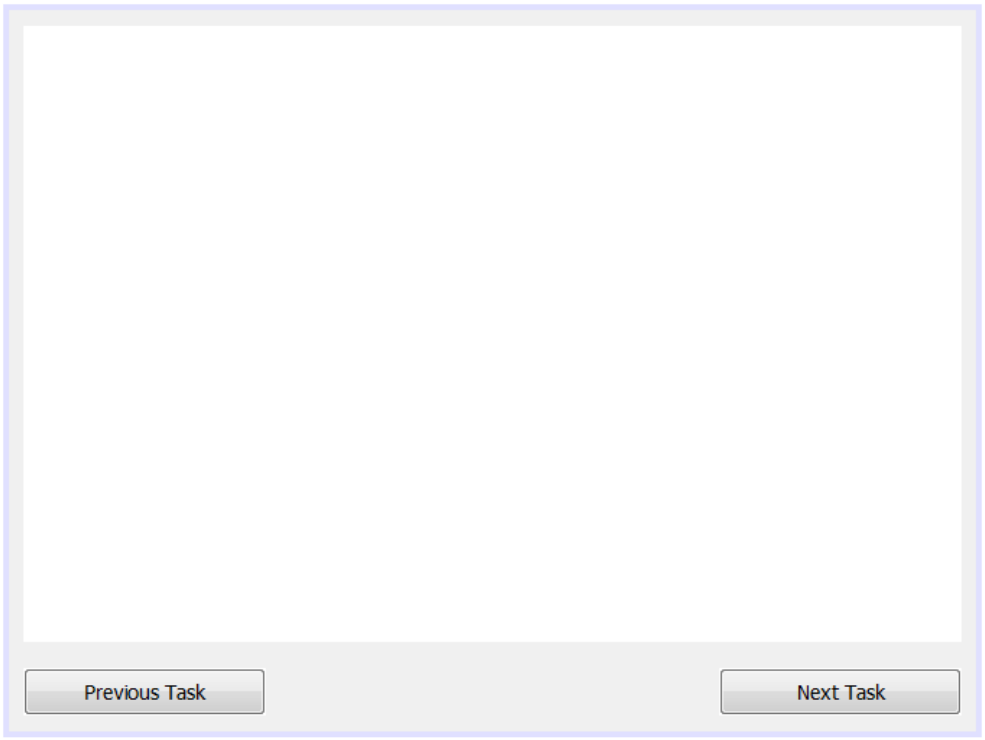


Figure 4.7 – Panel Tests

The applet is run with this code:

```
try {
    java.awt.EventQueue.invokeAndWait(new Runnable() {
        public void run() {
            initComponents();
        }
    });
} catch (Exception ex) {
    ex.printStackTrace();
}
```

Theoretical material opens with an event

```
    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        BufferedInputStream in = null;
        FileOutputStream fout = null;
        try {
            in = new
BufferedInputStream(getClass().getResource("theory.pdf").openStream());
            fout = new FileOutputStream("theory.pdf");
            byte data[] = new byte[1024];
            int count;
            while ((count = in.read(data, 0, 1024)) != -1) {
                fout.write(data, 0, count);
            }
        } catch (IOException ex) {
            Logger.getLogger(Applet.class.getName()).log(Level.SEVERE,
null, ex);
        } finally {
            try {
                if ( in != null)
                    in.close();
```

```
                if (fout != null) {
                    fout.close();
                    if (Desktop.isDesktopSupported()) {
                        File file = new
File(System.getProperty("user.dir")+"\\theory.pdf");
                        Desktop.getDesktop().open(file);
                    }
                }
            }
            catch (IOException ex) {
                Logger.getLogger(Applet.class.getName()).log(Level.SEVERE,
null, ex);
            }
        }
    }
```

Switching between panels is due to

```
CardLayout cl =(CardLayout) Main.getLayout();
cl.show(Main, "tests");
```

Variables have been created to determine the current step and the ability to switch between panels.

```
    int s=1;
    int c=0;
    String[] w = {"step1","step2","step3","step4","step4-
1","step5","step6","step7","step7-1"};
```

Also for some steps arrays for comparison of the given answer are specified.

```
    String[] t1 =
{"","E>TE'","T>FT'","F>id","","T'>e","E'>+TE'","","T>FT'","F>id","","T'>*FT
'","","F>id","","T'>e","E'>e"};
    String[][] t2 = new String[][]{
        {"E>TE'","","","E>TE'","","",""},
        {"","E'>+TE'","","","E'>e","E'>e"},
        {"T>FT'","","","T>FT'","","",""},
        {"","T'>e","T'>*FT'","","T'>e","T'>e"},
        {"F>id","","","F>(E)","","",""}
    };
```

To check the answer at each step, the Check () function is created. In case of an error, the help is displayed (see Appendix A).

Оскільки в тренажері передбачено повторне проходження, то після переходу до тестів додатково очищуються всі поля.

```
    private void StartActionPerformed(java.awt.event.ActionEvent evt) {
        CardLayout cl =(CardLayout) Main.getLayout();
        cl.show(Main, "tests");
        cl =(CardLayout) Steps.getLayout();
        cl.show(Steps, w[8]);
```

```
        s=1;
        c=0;
        buttonGroup1.clearSelection();
        buttonGroup2.clearSelection();
        buttonGroup3.clearSelection();
        jComboBox1.setSelectedIndex(0);
        jComboBox2.setSelectedIndex(0);
        jComboBox3.setSelectedIndex(0);
        jComboBox4.setSelectedIndex(0);
        jTextField1.setText("");
        jTextField2.setText("");
        jTextField3.setText("");
        jTextField4.setText("");
        jTextField5.setText("");
        jTextField6.setText("");
        for(int i = 0; i < jTable1.getRowCount(); i++){
            jTable1.setValueAt("", i, 2);
        }
        for(int i = 0; i < jTable2.getRowCount(); i++){
            for(int j = 1; j < jTable2.getColumnCount(); j++){
                jTable2.setValueAt("", i, j);
            }
        }
    }
```

The answer is checked before proceeding to the next step. If the last task was completed, a completion message is displayed.

```
    private void NextActionPerformed(java.awt.event.ActionEvent evt) {
        CardLayout cl;
        if(c<s-1) {
            c++;
        } else {
            if(Check()) {
                s++;
                c++;
            }
        }
        if(s<10) {
            cl =(CardLayout) Steps.getLayout();
            cl.show(Steps, w[c]);
        } else {
            JLabel label= new JLabel();
            label.setFont(new Font("Tahoma", Font.PLAIN, 14));
            label.setText("<html><center>You have completed the simulator
on the topic <br>\"Predictive parsing: scheme, principle of operation,
application\"<br> of the distance learning course <br>\"Programming
Theory\".</center>");
            JOptionPane.showMessageDialog(Main, label,
"Congratulations!",JOptionPane.PLAIN_MESSAGE);
            cl =(CardLayout) Main.getLayout();
            cl.show(Main, "home");
        }
    }
```

Going to the next step is possible only to the first task.

```
    private void PrevActionPerformed(java.awt.event.ActionEvent evt) {
        if(c>0)
        {
```

```
            c--;
            CardLayout cl =(CardLayout) Steps.getLayout();
            cl.show(Steps, w[c]);
        }
    }
```

All help and tips are as follows.

```
    private void Help1_1ActionPerformed(java.awt.event.ActionEvent evt) {
        Algorithm1.setSize(635, 480);
        Algorithm1.setResizable(false);
        Algorithm1.setLocationRelativeTo(null);
        Algorithm1.setVisible(true);
    }
```

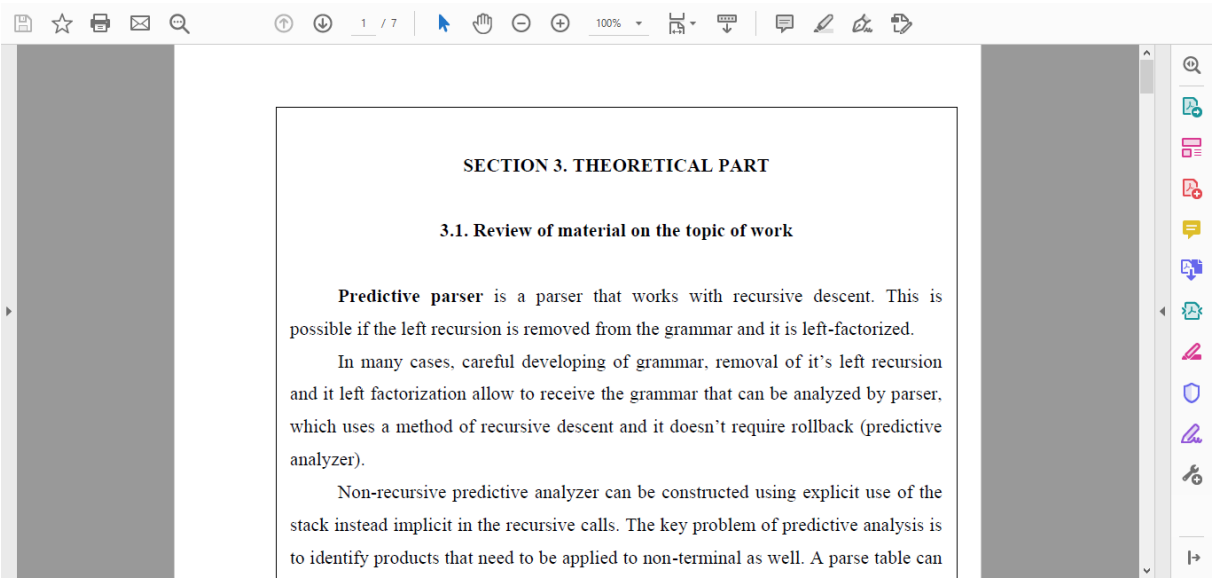
In the main class of the Predictive_parsing project the standard code of start of the program is registered.

```
public class Predictive_parsing {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        JFrame frame = new JFrame("Simulator");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        Applet applet = new Applet();
        frame.getContentPane().add(applet);
        applet.init();
        applet.start();
        frame.pack();
        frame.setSize(670, 500);
        frame.setResizable(false);
        frame.setLocationRelativeTo(null);
        frame.setVisible(true);
    }
}
```

4.3. Description of the program

On the start window there is a simulator theme, developer and manager (Fig. 4.6). You can open the theoretical material on the topic by clicking "The theoretical material" (Fig. 4.8). The "Go to the tests" button opens a window with the first task (Fig. 4.9).



SECTION 3. THEORETICAL PART

3.1. Review of material on the topic of work

Predictive parser is a parser that works with recursive descent. This is possible if the left recursion is removed from the grammar and it is left-factorized.

In many cases, careful developing of grammar, removal of it's left recursion and it left factorization allow to receive the grammar that can be analyzed by parser, which uses a method of recursive descent and it doesn't require rollback (predictive analyzer).

Non-recursive predictive analyzer can be constructed using explicit use of the stack instead implicit in the recursive calls. The key problem of predictive analysis is to identify products that need to be applied to non-terminal as well. A parse table can

Figure 4.8 – The theoretical material

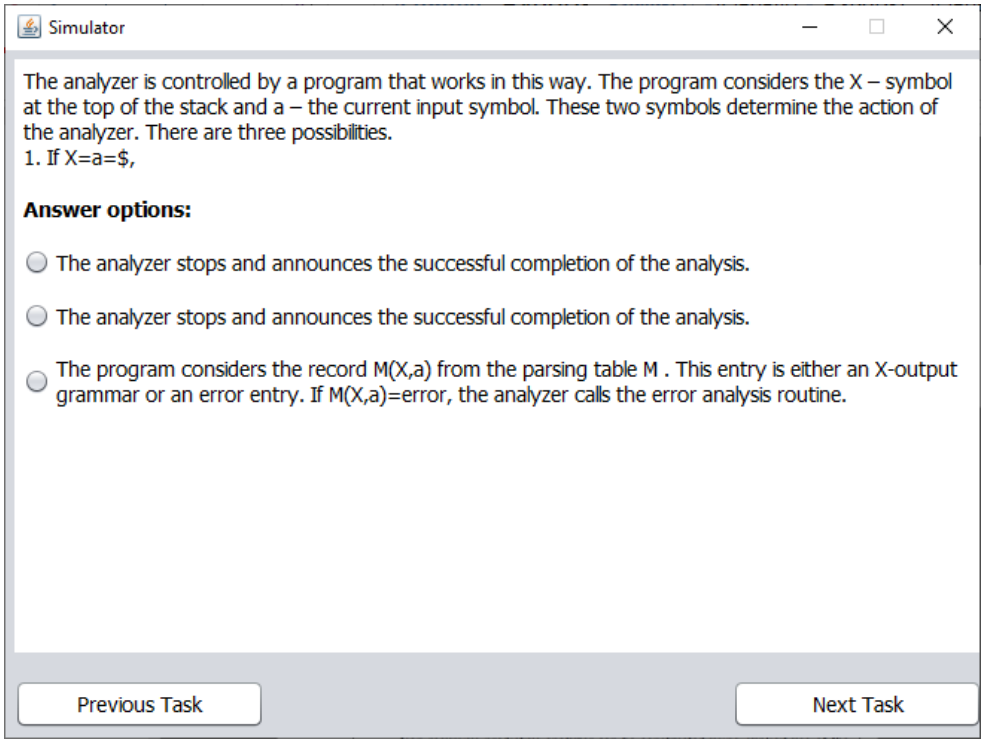


Figure 4.9 – First step

You need to select the answer and click "Next Task" (Fig. 4.10).

If the answer is incorrect, a help will be displayed (Fig. 4.11).

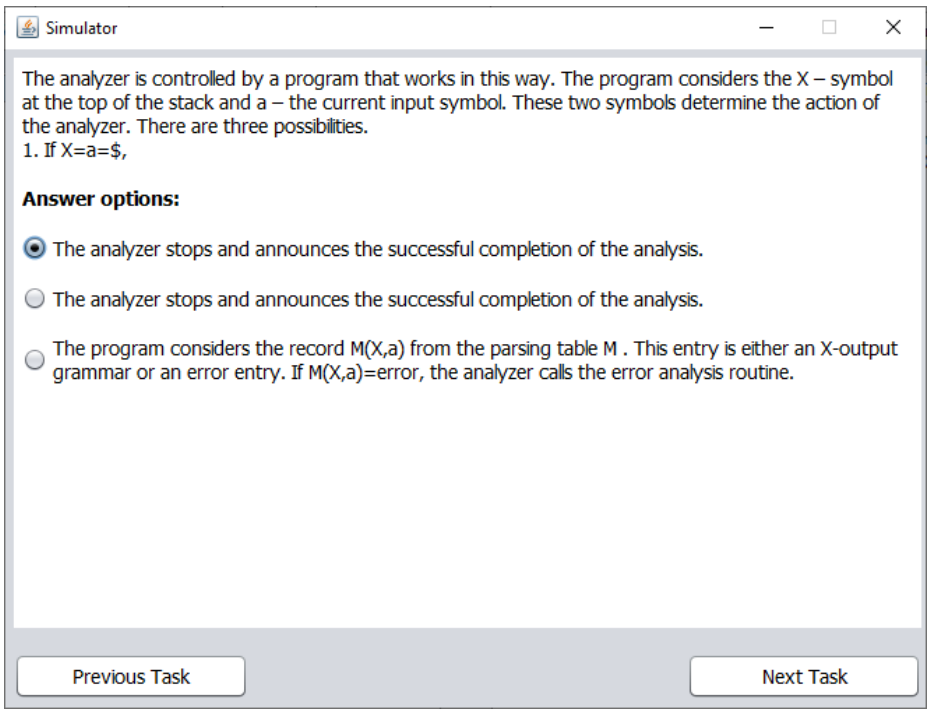


Figure 4.10 – Select the answer

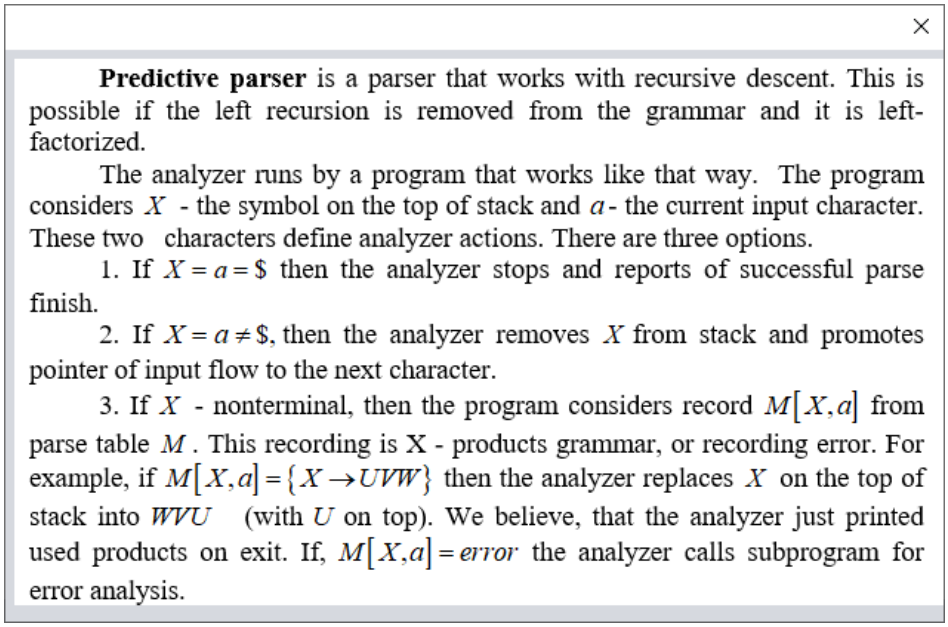


Figure 4.11 – Help

If the answer is correct, the next step is displayed (Fig. 4.12). You can return to the previous step at any time by clicking "Previous Task".

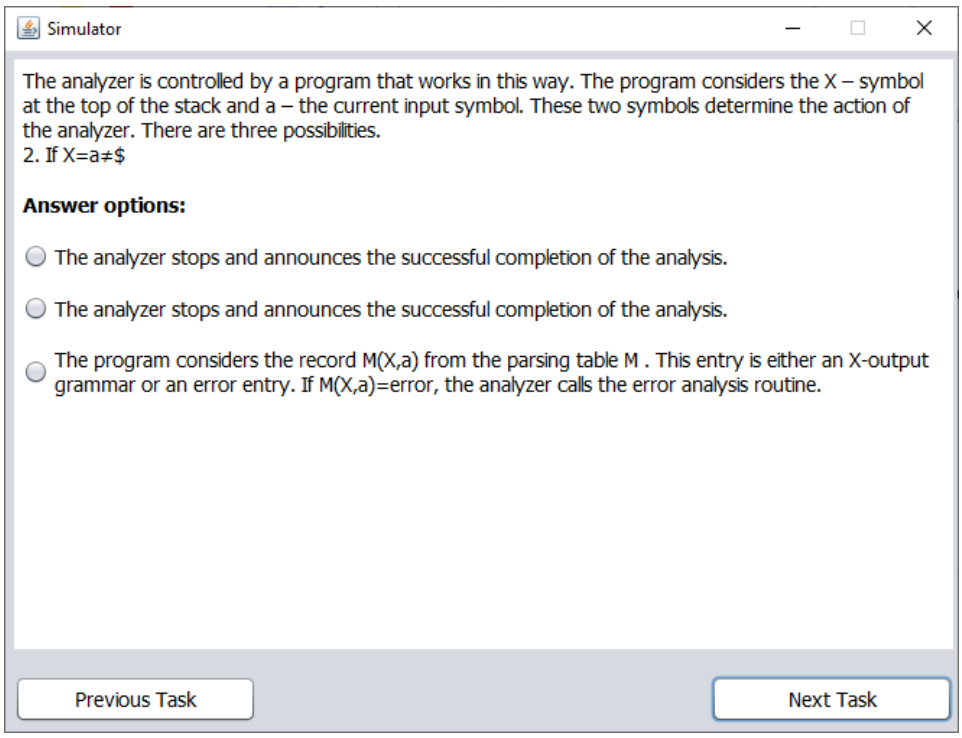


Figure 4.12 – Second step

The next steps require you to fill in a table column. First, the task is displayed, then you need to specify the answers (Fig. 4.13-4.14).

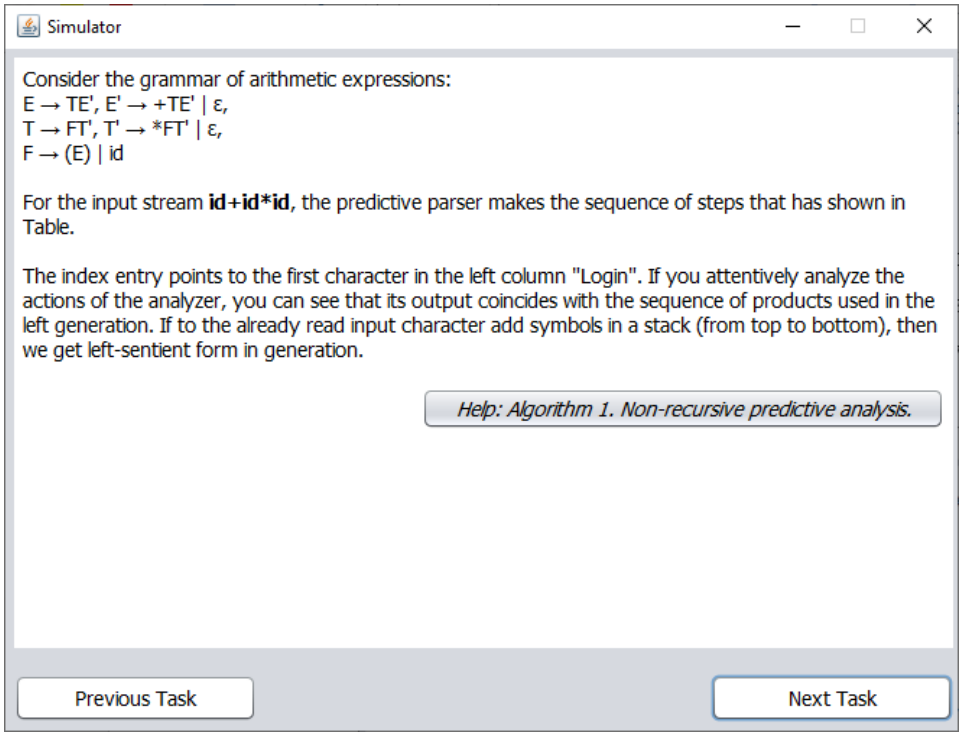


Figure 4.13 – Task

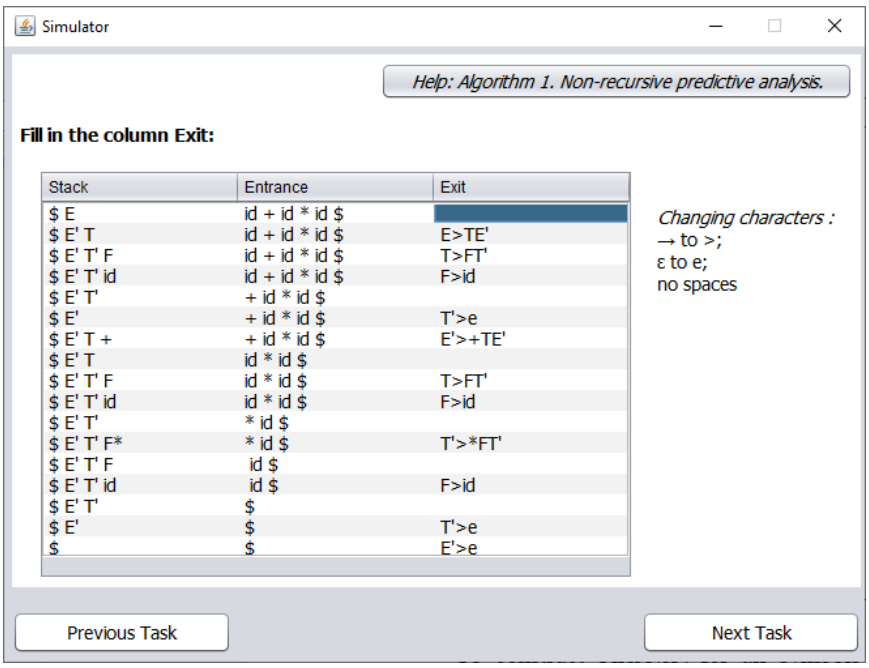


Figure 4.14 – Specify the answers

You can also open the help in these steps if the need arises (Fig. 4.15). To do this, click "Help".

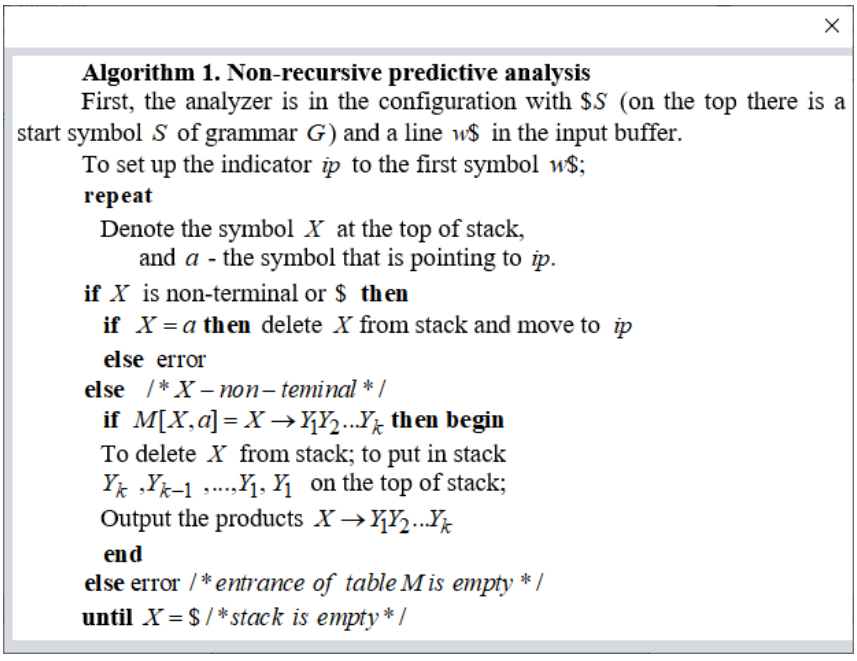


Figure 4.15 – Help with Algorithm

The next step is to establish the sequence. Next to each sentence you need to set their number (Fig. 4.16).

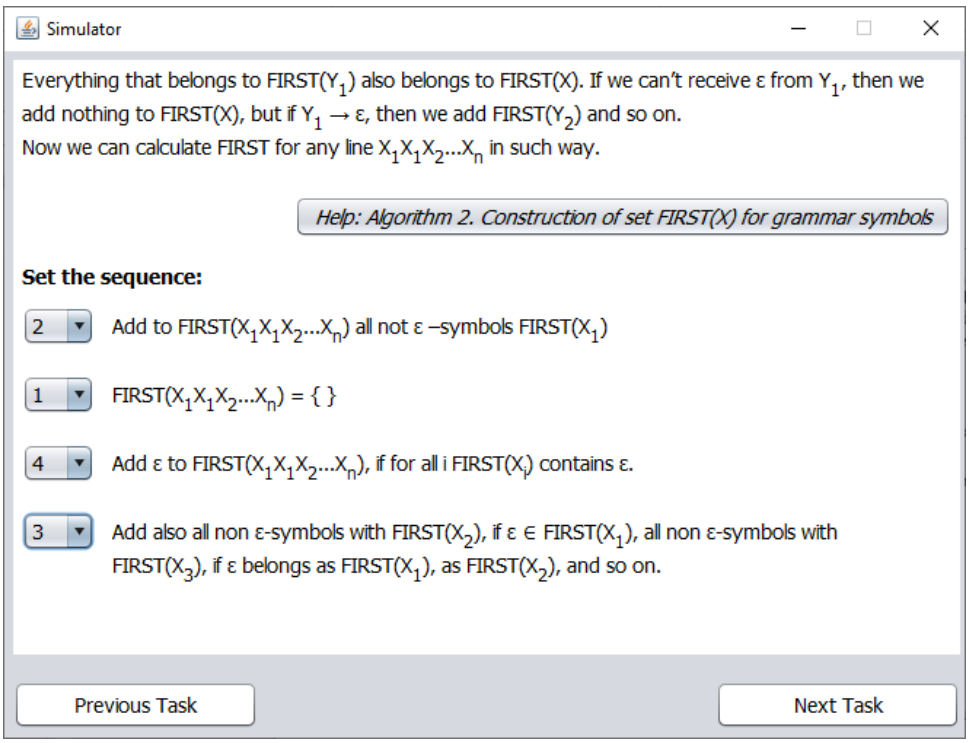


Figure 4.16 – Establish the sequence

Next you need to fill in the fields (Fig. 4.17).

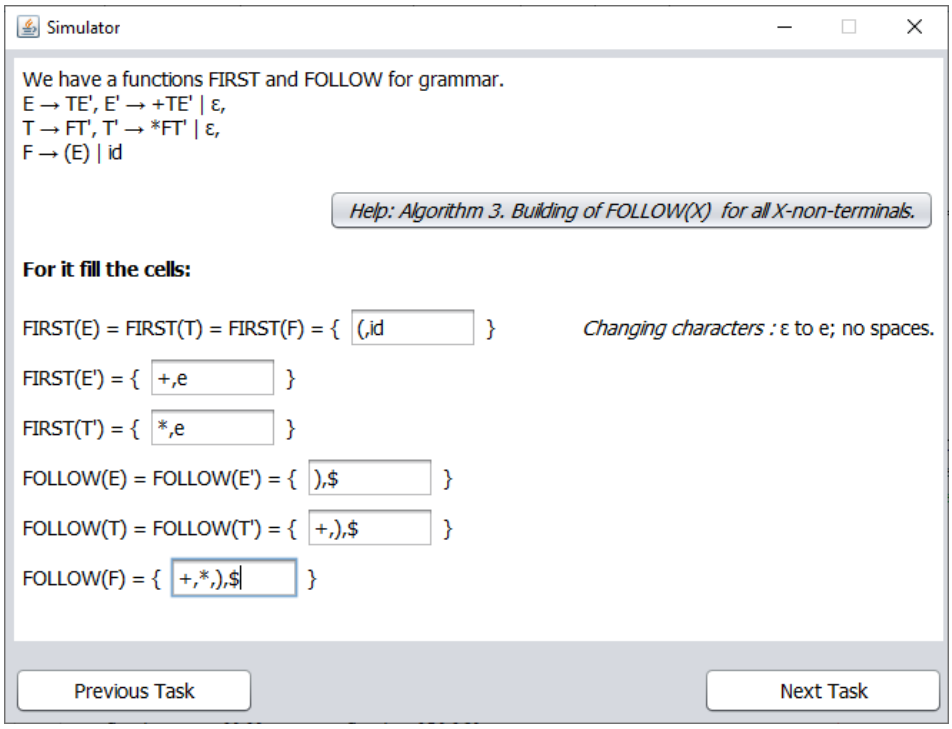


Figure 4.17 – Fill in the fields

In the last step, the task is displayed again, and then you need to fill in the table (Fig. 4.18-4.19).

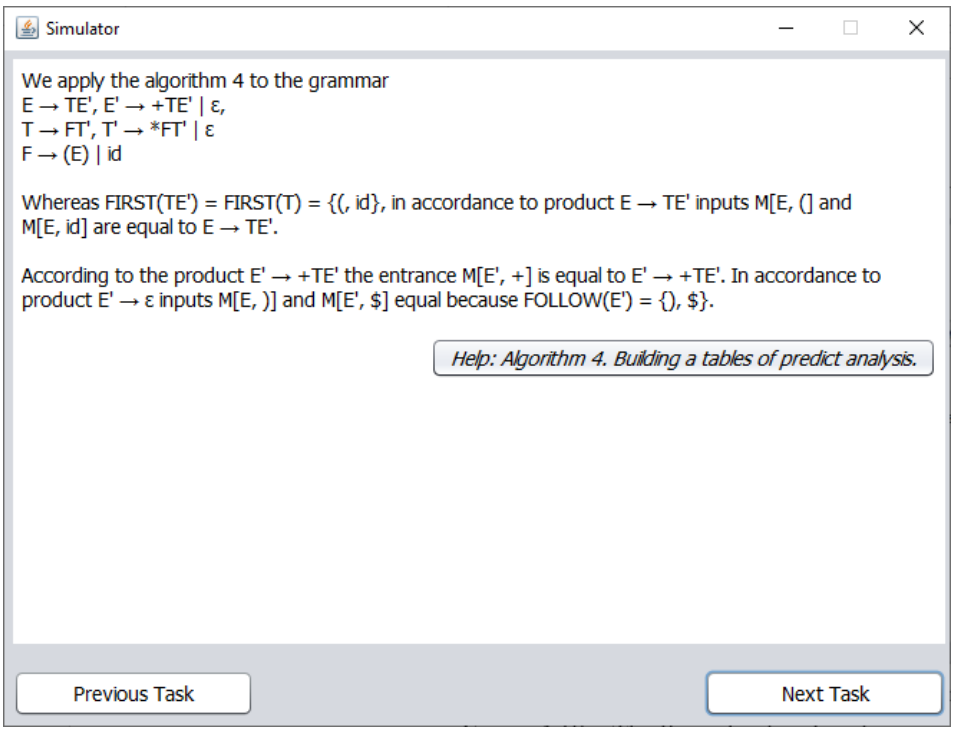


Figure 4.18 – Task to step

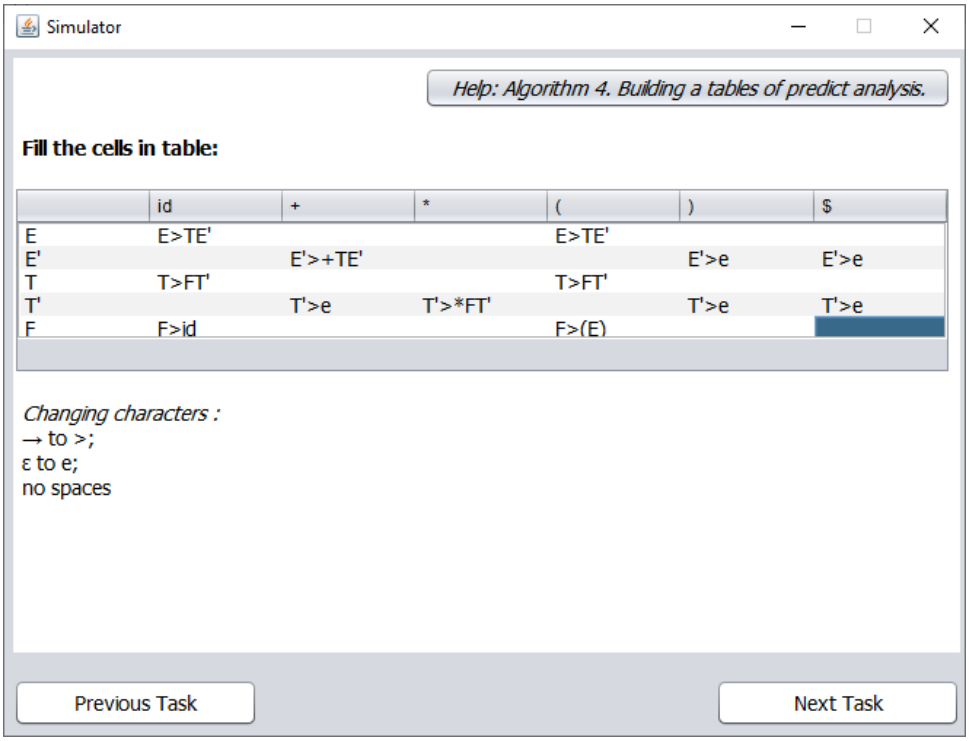


Figure 4.19 – Fill in the table

A completion message is then displayed (Fig. 4.20).

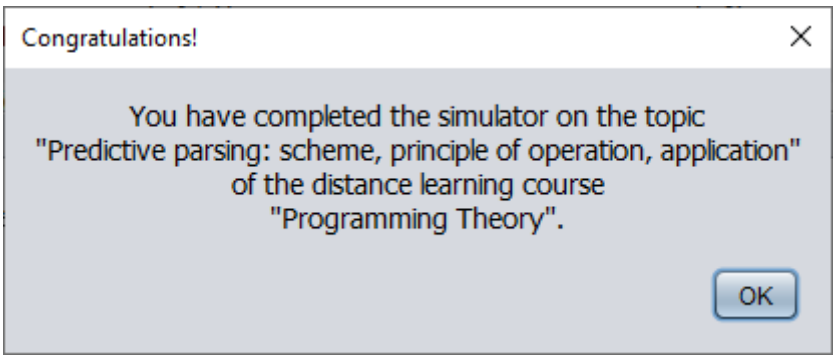


Figure 4.20 – A completion message

Then there is a transition to the home page.

CONCLUSIONS

The purpose of the work is the algorithm of the simulator and the simulator on the topic "Predictive parsing: scheme, principle of operation, application" distance learning course "Programming Theory".

The algorithm outlines the main steps of constructing predicative parsers for further software implementation.

The list of the main tasks and reference information on them is given:

- operation of the analyzer program;
- algorithm 1. Non-recursive predictive analysis;
- lgorithm 2. Construction of the set FIRST for grammar symbols;
- algorithm 3. Construction of FOLLOW (X) for all X-nonterminals of grammar;
- algorithm 4. Construction of predictive analysis tables.

To provide a clear and user-friendly interface, you must use the following structure:

- start page:
    - the theme of the simulator;
    - information about the developer;
    - button to go to the theoretical material;
    - button to go to training;
- passing the example:
    - condition of the problem;
    - tasks;
    - choice of answer;
    - go to the next step;
- result:
    - list of steps taken;
    - button to go to the home page;
    - exit button.

If the answer is incorrect, the output of reference information is implemented.

Creating simulators for distance learning - this opens up a new way for us to study for students of distance (distance) form of education. The advantage of simulators is that they can be used both for student training and for self-study.

The main results of the work:

1) Created theoretical material for the simulator;

2) Created tests for the simulator;

3) Developed an algorithm for the simulator on the topic "Predictive parsing: scheme, principle of operation, application" of the distance learning course "Programming Theory";

4) Developed a simulator;

5) His work was protested.

After passing the completion message is displayed, the start page opens.

REFERENCES

1. Ємець О. О. Методичні рекомендації до виконання бакалаврської роботи для студентів за освітньою програмою «Комп'ютерні науки» спеціальності 122 «Комп'ютерні науки та інформаційні технології» галузь знань - 12 «Інформаційні технології»» / О.О.(Олег) Ємець. – Полтава : РВВ ПУЕТ, 2011. – 71 с.
2. Морзе Н.В. Методика навчання інформатики. Частина І: Загальна методика навчання інформатики / Н.В. Морзе.– Київ: Навчальна книга, 2003
3. Морзе Н.В. Методика навчання інформатики. Частина ІІ: Методика навчання інформаційних технологій / Н.В. Морзе.– Київ: Навчальна книга, 2003
4. Toolkit: Tools for Creating Software Simulations [digital resource] / Joe Ganci. – Available from: https://learningsolutionsmag.com/articles/toolkit-tools-for-creating-software-simulations
5. Distance learning solutions [digital resource] / WWW.UNESCO.ORG. – Available from: https://en.unesco.org/covid19/educationresponse/solutions
6. Бабій М.С. Теорія програмування: Навчальний посібник [Електронний ресурс] / М.С. Бабій, О.П. Чекалов.– Суми: Вид-во СумДУ, 2009. – 181 с.
7. Нікітченко М.С. Теоретичні основи програмування: Навчальний посібник [Електронний ресурс] / М.С. Нікітченко. – Київ: КНУ ім. Т.Г. Шевченка, 2009. – 200 с. – Режим доступу: http://ttp.unicyb.kiev.ua/doc/TOP.pdf.
8. The Good and the Bad of Java Programming [digital resource] / AltexSoft – Available from: https://www.altexsoft.com/blog/engineering/pros-and-cons-of-java-programming/
9. Features of Java – Learn Why Java Is Important [digital resource] / DataFlair – Available from: https://data-flair.training/blogs/features-of-java/

10. David Flanagan Java™ in a Nutshell, Third Edition / David Flanagan. – Sebastopol: O'Reilly & Associates , Inc., 2001 – Available from: docstore.mik.ua/orelly/java-ent/jnut/copyrght.htm
11. Designing a Swing GUI in NetBeans IDE [digital resource] / Apache NetBeans 12.3 – Available from: https://netbeans.apache.org//kb/docs/java/quickstart-gui.html
12. Бібліографічний запис. Бібліографічний опис. Загальні вимоги та правила складання: ДСТУ 7.1-2006. – [Чинний від 2007-07-01]. – К. : Держспоживстандарт України, 2007. – 47 с.

APPENDIX A. PROGRAM CODE