Триус Ю. В., Дяченко А. Ю.

Програма-інтерпретатор алгоритмічних систем Маркова, Тьюрінга, Поста AlgoMachines

У роботі [9] відмічається, що “поняття “алгоритм” давно уже стало звичним не лише для математиків: воно є концептуальною основою різноманітних процесів обробки інформації; саме наявність відповідних алгоритмів і забезпечує можливість автоматизації таких процесів. Разом з математичною логікою теорія алгоритмів утворює теоретичний фундамент сучасних комп’ютерних наук”. З іншого боку завдяки розвитку інформатики та інформаційних технологій вдається доводити алгоритмічну розв’язність або нерозв’язність масових математичних проблем, виконувати автоматичне доведення математичних тверджень, досліджувати складність і ефективність алгоритмів тощо. Тому невипадково до навчальних планів спеціальностей, що відносяться до освітніх напрямів “прикладна математика” і “комп’ютерні науки”, включено дисципліну “Математична логіка і теорія алгоритмів”. На жаль, як показує досвід навчання цього курсу, студенти, “зачаровані” магією інформаційних технологій, без великого ентузіазму опановують фундаментальні основи математики та інформатики. Одним з шляхів подолання цього негативного явища, підвищення інтересу студентів до фундаментальних наук і їх навчально-пізнавальної активності є, на нашу думку, створення і впровадження в освітній процес комп’ютерно-орієнтованих методичних систем навчання.

У Черкаському національному університеті створено комп’ютерно-орієнтований навчально-методичний комплекс (НМК) з курсу “Математична логіка і теорія алгоритмів” для студентів математичних і комп’ютерних спеціальностей [8]. До складу НМК входять:

- навчальні посібники (у друкованому та електронному варіантах);
- навчально-методичні матеріали для забезпечення лекційних, практичних і лабораторних занять, самостійної та індивідуальної роботи студентів (у друкованому та електронному варіантах);
- навчальні інструментально-контролюючі програмні засоби: Master of Logic (для розв’язування і дослідження задач алгебри висловлень), DrawShem (для аналізу і синтезу комбінаційних схем), Boolean Minimiser (для побудови мінімальних нормальних форм формул алгебри висловлень), AlgoMachines (програма-інтерпретатор алгоритмічних систем А. Тьюрінга, Е. Поста, А. А. Маркова), створені студентами та аспірантами університету;
- банк завдань для самостійного виконання;
- банк завдань і тестів для поточного і підсумкового контролів.

У даній статті розглядається особливості використання програмного продукту AlgoMachines при навчанні основ теорії алгоритмів.

1. Загальні відомості про програму AlgoMachines

До складу програмного продукту AlgoMachines (розробник – Дяченко А. Ю., науковий керівник – Тирус Ю. В.) входять:

- виконуваний файл AlgoMachines.exe, що реалізує середовище програмування в алгоритмічних системах Маркова, Тьюрінга, Поста;
- підкаталог Examples, що містить бібліотеку готових до використання алгоритмів у внутрішньому форматі середовища AlgoMachines;
- підкаталог Help, який містить систему допомоги у форматі html;
- файл шрифтів Shemfont.ttf;
- файл AlgoMachines.cfg – файл з опціями програми, в тому числі паролем доступу до режиму підготовки контрольних завдань (у зашифрованому вигляді);
- файл p.dat – файл з протоколом виконання контрольних завдань (у зашифрованому вигляді);
- файли *.tst – файли з контрольними завдання.

Програма має два основні режими: *інструментальний* та *контролюючий*. В інструментальному режимі за допомогою програми користувач може конструювати алгоритми розв'язування різноманітних задач в алгоритмічних системах А. Тьюрінга, Е. Поста та А. А. Маркова і виконувати їх як в автоматичному, так і в покроковому режимах. При цьому користувач звільняється від виконання громіздких і рутинних операцій, які на певному етапі навчання є несуттєвими. Користувач також може одержувати необхідні теоретичні відомості через контекстну систему допомоги.

В **інструментальному режимі** за допомогою програмного продукту можна:

- створювати нові алгоритми в одній з алгоритмічних систем;
- відкривати вже існуючі алгоритми та коригувати їх;
- виконувати алгоритми при будь-яких вхідних даних, заданих у певному алфавіті;
- аналізувати роботу алгоритмів у режимі покрокового виконання;
- зберігати алгоритми;
- змінювати під час роботи мову інтерфейсу програми (передбачено використання української, російської та англійської мов);
- подавати алгоритми різними способами, зокрема алгоритми Маркова можна подати у вигляді граф-схеми, скороченої граф-схеми та блок-схеми, а програму машини Тьюрінга у вигляді таблиці чи послідовності команд.

Програма має синтаксичний аналізатор, який контролює правильність написання допустимих команд і повідомляє про допущені помилки. Також передбачені засоби проти зациклювання алгоритмів при їх виконанні.

Система параметрів програми призначена для швидкого налагодження середовища під специфіку задачі, що розв'язується.

Для введення вхідних даних можна використовувати як клавіатуру, так і спеціальну панель набору.

В **режимі підготовки контрольних завдань** програмного продукту AlgoMachines викладач може:

- підготувати файли з контрольними завданнями;
- встановити параметри контролю, зокрема, час проходження контрольних завдань, включення підтвердження відповіді, показ правильної відповіді, формування протоколу проходження контролю, встановлення захисту від несанкціонованого входження до режиму підготовки контрольних завдань або виходу з режиму контролю;
- встановлення параметрів оцінювання результатів.

Режим контролю призначений для перевірки умінь і навичок студентів аналізувати готові алгоритми і створювати власні алгоритми розв'язування задач обчислювального характеру в алгоритмічних системах Тьюрінга, Поста, Маркова.

2. Структура меню програми AlgoMachines

Управління роботою програми здійснюється через головне меню програми, яке містить такі основні пункти: *файл*, *редагування*, *опції*, *алгоритм*, *контроль*, *вікно*, *допомога*.

Пункт меню **"Файл"** містить команди для роботи з файлами, які є стандартними для більшості програм:

- "*Новий алгоритм*" – створити новий алгоритм в одній з алгоритмічних систем, в підпункті даного меню обирається тип системи ("Маркова", "Тюрінга", "Поста");
- "*Відкрити*" – відкрити вже існуючий алгоритм;
- "*Зберегти*" – зберегти алгоритм;
- "*Зберегти як*" – зберегти алгоритм у файлі з вказаним ім'ям;
- "*Закрити*" – закрити поточне вікно з алгоритмом;
- "*Закрити всі*" – закрити всі вікна з алгоритмами;

– “*Вихід*” – завершення роботи з програмним продуктом.

У залежності від типу алгоритмічної системи файли мають наступні розширення:

“.nam” – алгоритм в алгоритмічній системі Маркова;

“.tur” – алгоритм в алгоритмічній системі Тьюрінга;

“.pst” – алгоритм в алгоритмічній системі Поста.

Пункт меню **“Редагування”** містить такі команди:

– “*Вирізати*” – видалити і розмістити в буфері обміну виділений фрагмент тексту;

– “*Копіювати*” – копіювати у буфер обміну виділений фрагмент тексту, коли курсор знаходиться в полі тексту. Якщо ж курсор знаходиться на таблиці виконання, то копіюється таблиця, а якщо користувач проглядає в цей час блок-схему чи (скорочену) граф-схему, то у буфер обміну копіюється відповідне зображення схеми;

– “*Вставити*” – вставка з буферу обміну тексту;

– “*Вставити* хххх” – вставка в поточне вікно з текстом відповідного символу хххх з множини символів $\{\rightarrow, \Lambda, \alpha, \beta, \gamma\}$.

Пункт меню **“Опції”** містить такі параметри програми:

– “*Мова*” – зміна мови інтерфейсу програми (англійська, російська, українська). Зміна мови відбувається без перезавантаження програми;

– “*Використовувати алфавіт*” – встановлення використання множини символів для алгоритмічної системи Маркова. Якщо ця опція не встановлена, то як символи вхідного алфавіту використовуються всі доступні в програмі символи, якщо ж вона встановлена, то символи алфавіту треба вводити через кому і алфавіт в цьому випадку зберігається в тому ж файлі, що й алгоритм;

– “*Панелі набору*” – прибрати / показати панель набору програми, зокрема панелі: “Літери”, “Цифри”, “Операції”, “Редагування”, “Спеціальні символи”. Панелі набору є загальними для всіх режимів програми і не залежать від виду обраної для роботи алгоритмічної системи. Тому вони зберігаються в загальних опціях програми – файлі “AlgoMachines.cfg”;

– “*Показувати*” – налаштування виду вінка з алгоритмом, зокрема таких його елементів: програми (алгоритму), умови задачі, коментарів, протоколу виконання програми (алгоритму), панелі виконання, початкового положення каретки, поля результату, інформаційної стрічки. Дані опції індивідуальні для кожної алгоритмічної системи і зберігаються безпосередньо в файлі з алгоритмом;

– “*Параметри машини Тьюрінга*” – встановлення параметрів машини Тьюрінга, зокрема: символу позначення внутрішнього стану, максимальної кількості кроків виконання алгоритму, кольору рядка нумерації, кольору інформаційної стрічки, кольору каретки, кольору символів на стрічці та кольору номерів комірок, при цьому ці опції індивідуальні для кожної машини Тьюрінга і зберігаються безпосередньо в файлі з алгоритмом;

– “*Параметри машини Поста*” – встановлення параметрів машини Поста відбувається у вікні, яке практично співпадає з аналогічним вікном для машини Тьюрінга (відмінність полягає лише в обранні символу для позначення мітки, замість символу позначення внутрішнього стану).

Пункт меню **“Алгоритм”** містить такі команди:

– “*Виконати*” – застосувати введений алгоритм до заданого вхідного слова, при цьому будується таблиця виконання, в якій фіксується команда, що виконується, та результат її застосування на відповідному кроці роботи алгоритму. Останній рядок таблиці містить результат роботи алгоритму. Команда “Виконати” може бути недоступна користувачу внаслідок неправильно заданого алфавіту, неправильно введеного алгоритму (наприклад якщо використовуються символи не з алфавіту) чи неправильно заданого вхідного слова;

– “*Пауза*” – призупинити виконання алгоритму без завершення процесу виконання. Одночасно при цьому відбувається перехід до режиму покрокового виконання і подальше виконання алгоритму можливе як у покроковому, так і в автоматичному режимах;

– “*Виконати покроково*” – покомандне виконання алгоритму при заданому вхідному слові, при цьому перехід до наступного кроку відбувається за вказівкою користувача. Команда, що буде виконуватися наступною, виділяється червоною рамкою. Протокол виконання теж заповнюється покроково. З режиму покрокового виконання можливий перехід до режиму автоматичного виконання;

– “*Припинити виконання*” – “аварійна” зупинка виконання алгоритму, при цьому в якості результату роботи алгоритму подається поточний стан інформаційної стрічки (для машин Тьюрінга та Поста) або поточне слово (для алгоритмів Маркова).

Доступ до пунктів меню “Алгоритм” можливий також через кнопки панелі виконання алгоритмів (рис. 1), котрі розташовані безпосередньо в робочих вікнах, та клавіші “швидкого” доступу:

– виконати – F9,
– пауза – CTRL+F3,
– виконати покроково – F8,
– припинити виконання – CTRL+F2.

Рис. 1. Панель виконання алгоритмів

Пункт меню **“Вікна”** містить команди для роботи з вікнами, які є стандартними для багатовіконних (MDI) програм:

– “*Наступне*” – перехід до наступного вікна з алгоритмом, порядок розташування вікон залежить від порядку створення вікон;
– “*Попереднє*” – перехід до попереднього вікна з алгоритмом;
– “*Каскадом*” – розташувати всі вікна з алгоритмами каскадом;
– “*Розділити горизонтально*” – показати всі вінка з алгоритмами, при цьому розділення вікон відбувається горизонтально;
– “*Розділити вертикально*” – показати всі вінка з алгоритмами, при цьому розділення вікон відбувається вертикально.

Крім того, цей пункт містить список імен всіх алгоритмів, які завантажені в середовищі.

Пункт меню **“Контроль”** містить такі підпункти:

– “*Параметри режиму контролю*” – перехід до режиму розробки файлу з контрольними завданнями;
– “*Розпочати контроль*” – перехід до режиму контролю.

Пункт меню **“Допомога”** містить такі підпункти:

– “*Теоретичні відомості*” – виклик довідкової системи програми **AlgoMachines**, яка, містить такі розділи: характеристика програми; теоретичні відомості про алгоритмічні системи А. Тьюрінга, Е. Поста, А. А. Маркова; приклади розв'язування задач за допомогою програми **AlgoMachines;**
– “*Про програму*” – відомості про авторів програми.

3. Структура робочого вікна програми AlgoMachines

Правила роботи з програмним продуктом і теоретичні відомості про реалізовані в ній алгоритмічні системи наведено в розділі допомоги, яка викликається за допомогою пунктів меню “*Допомога* \ *Робота з програмою*” та “*Допомога* \ *Теоретичні відомості*”.

Робота з програмним продуктом починається із завантаження головного модуля програми AlgoMachines.exe.

Для організації роботи користувача з програмою AlgoMachines прийнятий практично однаковий вид робочого вікна для різних алгоритмічних систем. Так на рис. 2 подано загальний вигляд вікна “Машина Тьюрінга”.

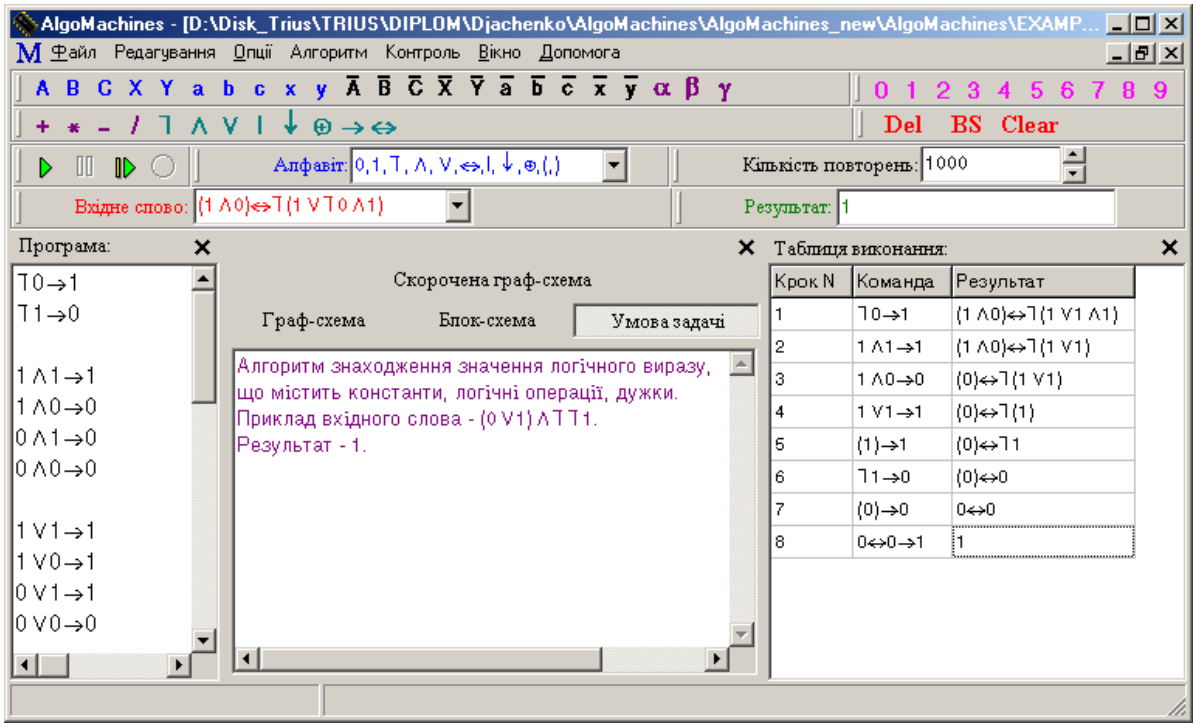

Рис. 2. Вікно програми AlgoMachines в режимі “Машина Тьюрінга”

У верхній частині вікна розташовані: панель виконання, поле введення алфавіту, поле для введення вхідного слова, поле результату, поле вибору початкового положення читаючого елементу та інформаційна стрічка (для алгоритмічних систем Тьюрінга і Поста). Дані елементи можна довільно переміщувати, змінювати їх розмір і приховувати. Ця панель налаштовується індивідуально для кожного алгоритму.

В лівій частині вікна знаходиться алгоритм, у центрі – умова задачі та коментарі до неї, у правій частині – протокол виконання алгоритму. Перелічені елементи теж можна показувати/приховувати в залежності від специфіки задачі, що розв'язується.

4. Особливості реалізації алгоритмічної системи Маркова

4.1. Загальна характеристика алгоритмічної системи Маркова.

Алгоритмічна система Маркова [1, 2, 4–7] будується за загальною схемою побудови алгоритмічних систем. Спочатку задається абстрактний алфавіт U. Вхідна дані задачі подається у вигляді слів в алфавіті U. Система допустимих операцій містить дві операції:

– операцію підстановки (операцію дії);
– операцію входження (операцію розпізнавання).

Операція *підстановки* записується у вигляді:

$$A \to B,$$

де A і B – слова в алфавіті U, при цьому знак $\to$ не входить до алфавіту U.

Застосування операції підстановки $A \to B$ до даного слова X в алфавіті U полягає в тому, що перше зліва входження слова A в дане слово X замінюється на слово B.

Застосування операції підстановки $A \to B$ до даного слова X можливе лише за умови, якщо слово A входить у дане слово X. У цьому випадку кажуть, що підстановка $A \to B$ *застосована* до слова X.

Якщо ліва частина підстановки $A \to B$, тобто слово A, не входить до слова X, то підстановка $A \to B$ *незастосовна* до слова X.

Частинними випадками підстановок є дві підстановки:

$$\Lambda \to B,\ A \to \Lambda,$$

де символ Λ означає порожнє слово. За замовчуванням вважається, що для будь-якого слова X в будь-якому алфавіті U має місце подання $\Lambda X\Lambda$.

У підстановці $\Lambda\to B$ порожнє слово замінюється на слово B, а в підстановці $A\to\Lambda$ слово A замінюється на порожнє слово.

Розглянуті підстановки називаються *звичайними підстановками*. При побудові алгоритмів в алгоритмічних системах передбачено обов'язкова присутність операції закінчення роботи алгоритму.

У теорії нормальних алгоритмів роль такої операції виконує *заключна підстановка* виду

$$A\to .B,$$

після застосування якої алгоритмічний процес зупиняється (при цьому символ "**.**" не входить до алфавіту U).

Застосування операції підстановки не можливе без іншої операції – *операції входження*.

Операція *входження* має вигляд

$$A\subset X,$$

де символ "$\subset$" не входить до алфавіту U. Це предикат, значенням якого може бути "хиба" (0) або "істина" (1). Операція входження визначає можливість застосування операції підстановки $A\to B$ до слова X.

4.2. Алгоритмічний процес в системі Маркова.

Скінченну упорядковану послідовність операцій входження і відповідних їм операцій підстановок, виконання яких призводить до розв'язування поставленої задачі в деякому алфавіті, називають ***нормальним алгоритмом***.

Необхідною умовою можливості побудови нормальних алгоритмів, які реалізують будь-який конструктивно заданий процес перетворення слів, є використання обох видів підстановок як звичайних, так і заключних, яких може бути кілька в одному алфавіті.

Сконструювати нормальний алгоритм означає записати в певному порядку одну операцію за іншою.

Суть алгоритмічного процесу в системі Маркова полягає в тому, щоб застосувати нормальний алгоритм до заданого вхідного слова за таким правилом:

– виконати першу підстановку алгоритму, яку можна застосувати до заданого вхідного слова;

– процес застосування підстановок вести доти, поки може бути застосована до поточного слова хоча б одна із звичайних підстановок алгоритму або доти, поки буде застосована перший і єдиний раз заключна підстановка, при цьому щоразу перегляд підстановок відбувається з першої операції в алгоритмі.

Якщо на деякому етапі алгоритмічного процесу жодна підстановка алгоритму незастосовна до поточного слова, то алгоритм завершує свою роботу і результатом вважається одержане на цей момент слово.

Кажуть, що нормальний алгоритм Ω застосовний до слова X, якщо процес перетворення слова X закінчується після скінченого числа кроків яким-небудь словом Y. В цьому випадку кажуть, що алгоритм Ω перетворює слово X в слово Y:

$$\Omega\ (X)=Y.$$

Якщо процес перетворення слова X алгоритмом Ω ніколи не закінчується, то кажуть, що алгоритм Ω *незастосовний* до слова X.

Зауважимо, що якщо жодна підстановка алгоритму Ω незастосовна до вхідного слова X, то результатом роботи алгоритму будемо вважати саме слово X, тобто $\Omega(X)=X$.

Для скороченого запису нормальних алгоритмів використовуються так звані *схеми нормальних алгоритмів*, які містять лише операції підстановки, а відповідні їм операції входження опускаються. Для подання нормальних алгоритмів використовують блок-схеми, граф-схеми або скорочені граф-схеми [1].

Нормальний алгоритм називається *замкненим*, якщо його схема містить хоча б одну заключну підстановку виду $A \to .B$. Будь-який нормальний алгоритм можна зробити замкненим, якщо вкінці алгоритму добавити підстановку виду $\Lambda \to .\Lambda$.

Замкнений нормальний алгоритм позначають $\Omega\cdot$. Очевидно, що незамкнений алгоритм Ω і замкнений алгоритм $\Omega\cdot$, одержаний доповненням підстановкою $\Lambda \to .\Lambda$, еквівалентні, тобто $\Omega(X)=\Omega\cdot(X)$ для будь-якого слова X в алфавіті U.

Розглянемо особливості реалізації алгоритмічної системи Маркова в програмі AlgoMachines і наведемо декілька прикладів розв'язування задач за її допомогою.

Приклад 1. Нехай задано алфавіт U={a, b} і схема нормального алгоритму:

$$A_{ab} = \begin{cases} ab \to ba \\ ba \to \Lambda \\ b \to .a \end{cases}$$

Побудувати граф-схему алгоритму і застосувати його до заданого вхідного слова: X="bbaba".

Для виконання даного завдання створюємо новий алгоритм в алгоритмічній системі Маркова (меню "*Файл \ Нова машина \ Маркова*") і в полі "Алфавіт" задаємо вхідний алфавіт. Потім у редакторі програми набираємо текст програми за допомогою панелі набору або безпосередньо з клавіатури. Обравши закладку "Граф-схема", одержуємо граф-схему алгоритму, подану на рис. 3. Після цього в полі "Вхідне слово:" треба ввести вхідне слово в заданому алфавіті. Для застосування створеного алгоритму до вхідного слова обирається пункт меню "*Алгоритм \ Виконати*" (чи "*Алгоритм \ Виконати покроково*"). Після цього в полі "Результат:" одержується результат застосування алгоритму. Кожний крок виконання алгоритму фіксується в "*Таблиці виконання*".

Застосувавши до вхідного слова X=bbbaa алгоритм A_{ab}, одержимо: X_1=bba; X_2=b; X_3=a – результат (див. рис. 3).

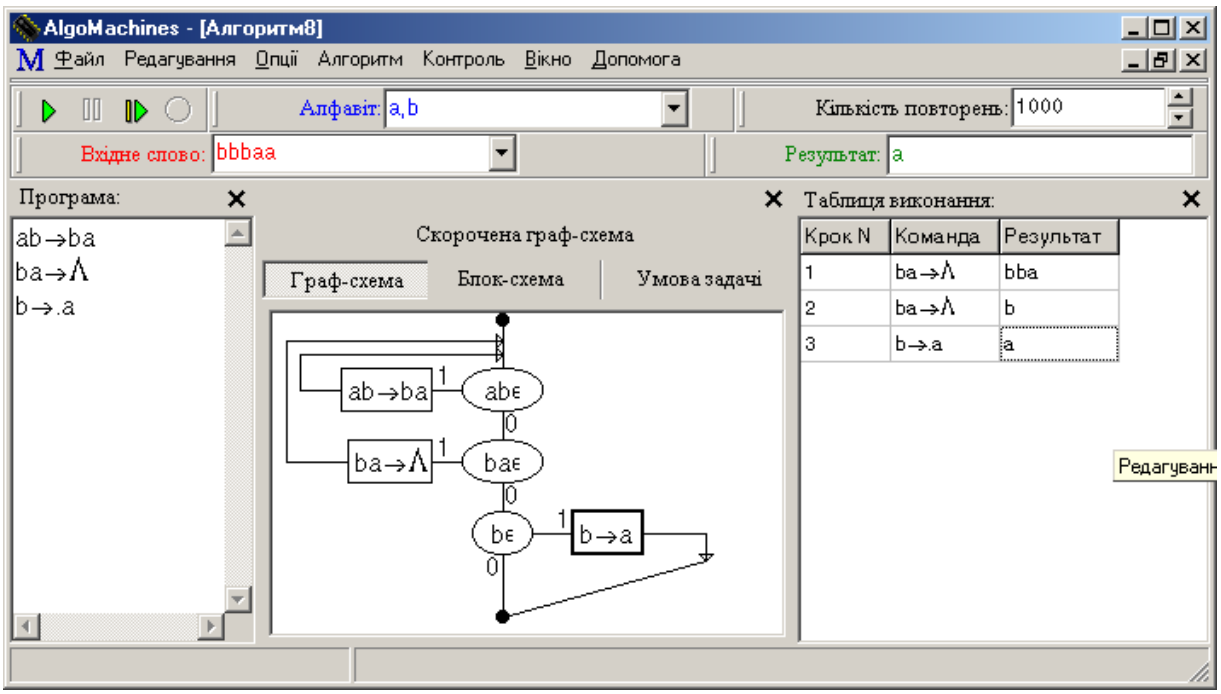

Рис. 3. Робоче вікно системи Маркова з граф-схемою алгоритму

Приклад 2. Побудувати нормальний алгоритм для знаходження суми двох натуральних чисел.

Нехай задано два натуральних числа n і m. Подамо ці числа у вигляді слів в алфавіті U={|}. У цьому алфавіті натуральні числа подаються у вигляді слів, які містять відповідну кількість букв "|". Наприклад, 3=|||, 5=|||||.

Для подання вхідних даних задачі доповнимо алфавіт U символом "+". Тоді вхідне слово в розширеному алфавіті $U' = U \cup \{+\} = \{|, +\}$, яке відповідає виразу "$n+m$", наприклад при $n=3$ і $m=5$, буде мати такий вигляд: ||||+|||.

Легко бачити, що нормальний алгоритм для розв'язування поставленої задачі буде мати таку просту схему

$$A_{+} = \{+ \to .\Lambda$$

Для розв'язування поставленої задачі за допомогою програми AlgoMachines виконуємо дії, аналогічні наведеним у прикладі 1. Блок-схема алгоритму і результат його виконання при $n=3$ і $m=5$ подано на рис. 4.

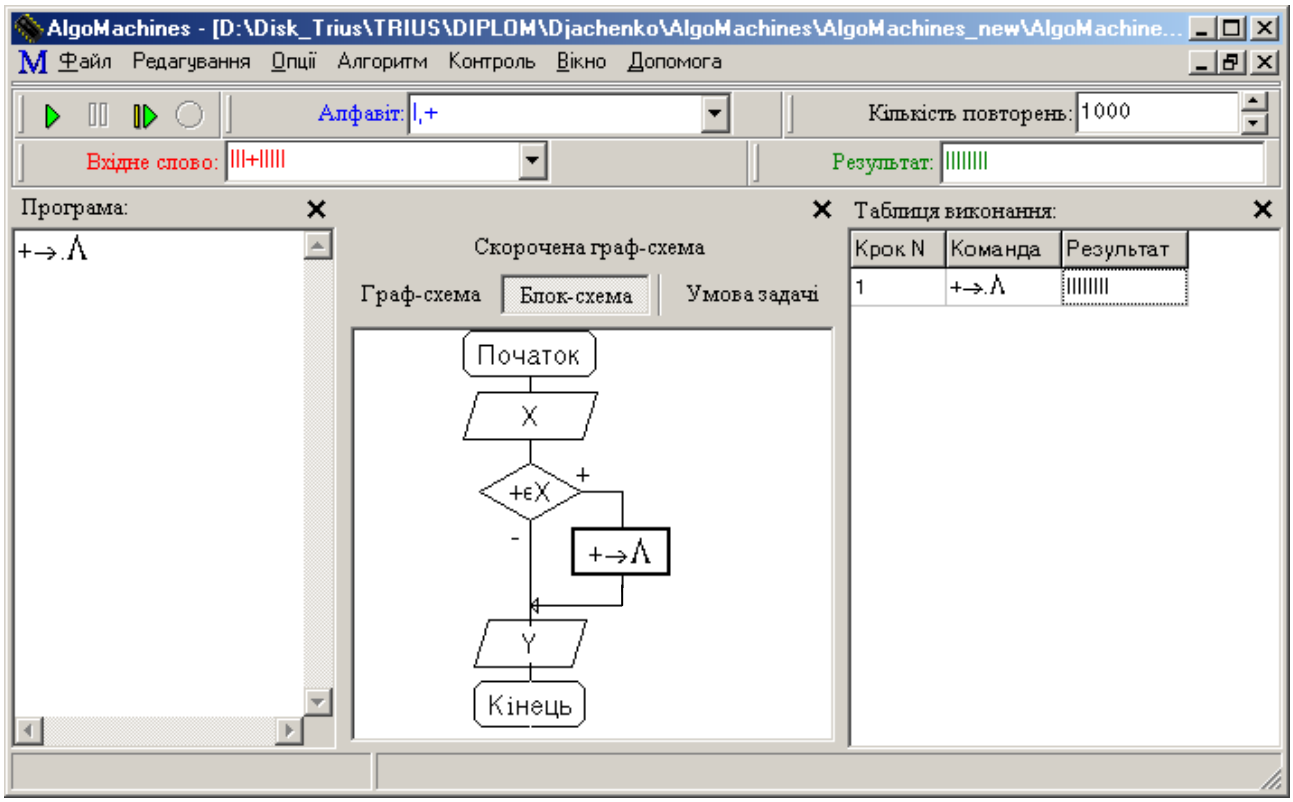

Рис. 4. Алгоритм знаходження суми двох натуральних чисел

Приклад 3. Побудувати нормальний алгоритм для знаходження різниці двох довільних натуральних чисел.

Натуральні числа n, m і вхідні слова будемо зображати в алфавіті $U=\{|,-\}$. Тоді вхідне слово, яке відповідає виразу "$n-m$", наприклад при наприклад при $n=4$ і $m=2$, буде мати такий вигляд: ||||–||.

Оскільки результат віднімання двох довільних натуральних чисел може бути цілим числом, то число 0 будемо вважати порожнім словом Λ, а від'ємні числа будемо зображати як слова в алфавіті U зі знаком "–" попереду.

Розглянемо два випадки:

а) $n \geq m$;

б) $n < m$.

У випадку а) до нормального алгоритму треба ввести таку підстановку, яка видаляє по одній букві "|" в зменшуваному і від'ємнику одночасно, і залишає знак "–": $|-| \to -$.

Знак "–" треба залишати в слові, що перетворюється, доти, поки від'ємник має ненульову довжину. Щоб одержати різницю після закінчення дії першої підстановки, треба ввести ще підстановку, яка видаляє символ "–". Тоді нормальний алгоритм віднімання двох натуральних чисел в алфавіті $U=\{|, -\}$ за умови, що $n \geq m$, має вигляд:

$$A_{-}^{\geq} = \begin{cases} |-| \to - \\ - \to .\Lambda \end{cases}$$

Для випадку б) алгоритм дещо ускладнюється, оскільки треба проаналізувати процес завершення у випадку, коли після дії підстановки $|-|\to-$ поточне слово буде починатися з підслова "–|". Оскільки, якщо залишити алгоритм без зміни, то він буде обчислювати значення виразу $|n-m|$.

Для одержання результату у вигляді від'ємного числа треба до алгоритму ввести підстановку $-|\to.-|$ перед підстановкою $-\to.\Lambda$. Тоді схема нормального алгоритму матиме вигляд:

$$A_{-}=\begin{cases}|-|\to-\\ -|\to.-|\\ -\to.\Lambda\end{cases}$$

Результат виконання алгоритму A_{-} в середовищі програми Algomachines при *n*=5 і *m*=3 подано на рис. 5.

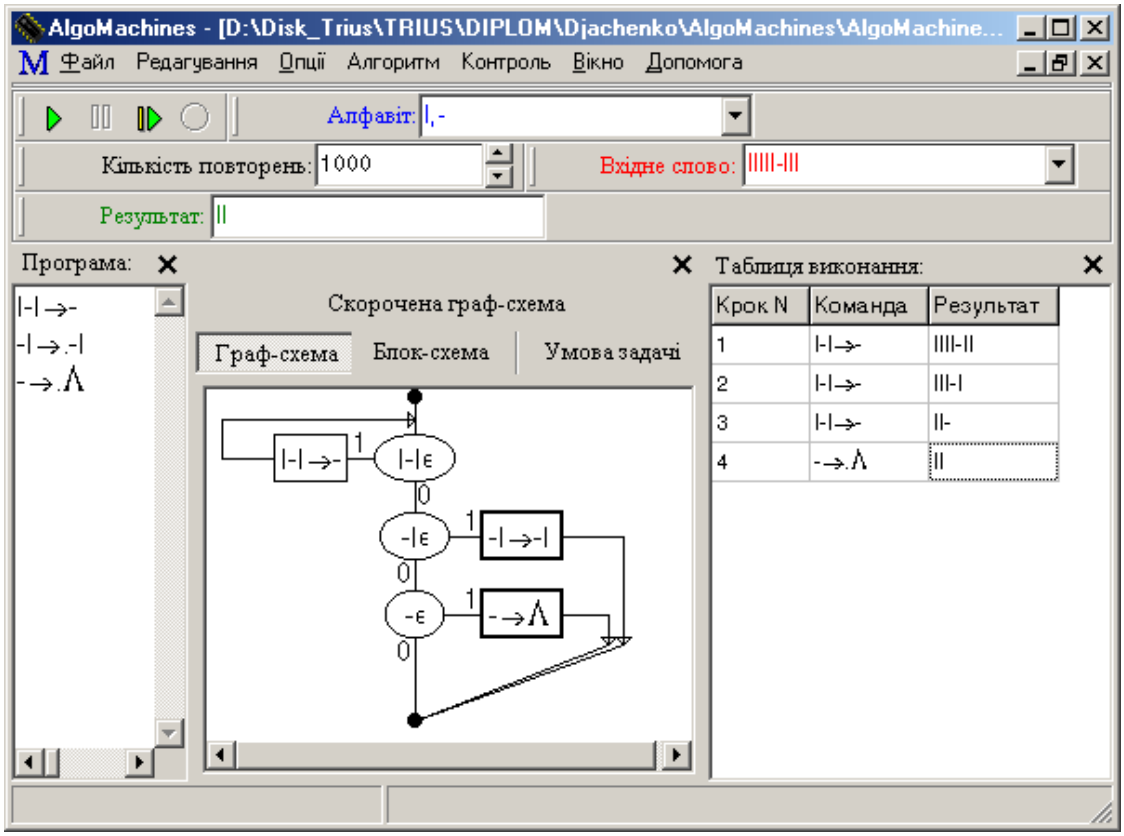

Рис. 5. Алгоритм знаходження різниці двох натуральних чисел

5. Особливості реалізації алгоритмічної системи Тьюрінга

5.1. Опис пристроїв машини Тьюрінга.

Машина Тьюрінга – ще один спосіб подання алгоритму [1–5, 7]. Як вважав А. Тьюрінг, розв'язати задачу означає, керуючись певною ідеєю, сконструювати деяку машину, яка буде розв'язувати цю задачу при будь-яких вхідних даних.

Ідея машини Тьюрінга ґрунтується на загальному аналізі процесу знаходження значень функції обчислювачем. При цьому, як і в теорії нормальних алгоритмів, має місце основна гіпотеза теорії алгоритмів: *будь-який конструктивно заданий алгоритм може бути реалізований машиною Тьюрінга.*

Машина Тьюрінга має такі основні складові:

– інформаційна стрічка (блок зовнішньої пам'яті (БЗП));
– блок внутрішньої пам'яті (БВП);
– читаючий елемент (ЧЕ);

– «механічний» пристрій (МП);

– функціональний (програмний) блок (ФБ).

Інформаційна стрічка (блок зовнішньої пам'яті) являє собою скінченну кількість послідовно розташованих комірок. В процесі роботи машини Тьюрінга до існуючих комірок стрічки за допомогою «механічного» пристрою можуть добудовуватися нові комірки як зліва, так і справа (тобто стрічку можна вважати потенційно необмеженою в обидві сторони, але в кожний конкретний момент часу вона складається з скінченної кількості комірок).

Вхідні дані задачі кодуються за допомогою скінченної кількості букв (станів) зовнішнього алфавіту $A=\{a_0, a_1, \dots, a_m\}$, кожна з яких зберігається в одній з комірок БЗП, при цьому букву a_0 (порожній знак) будемо вважати ознакою порожньої комірки.

Кожна комірка в певний такт роботи машини Тьюрінга може містити тільки одну букву. В комірках зовнішньої пам'яті, що не зайняті іншими буквами зовнішнього алфавіту, за замовчуванням розміщується буква a_0.

Якщо до стрічки добавляються нові комірки, то вони за замовчуванням містять «порожній символ».

В процесі роботи машини Тьюрінга комірки зовнішньої пам'яті можуть змінювати свій стан, або залишатися незмінними.

Інформаційна стрічка вважається направленою і її комірки в кожний момент часу упорядковані зліва направо.

Тобто, якщо в якийсь момент часу (такт роботи) j стрічка має r комірок, то стан стрічки повністю описується скінченною послідовністю букв алфавіту A:

$$a_{j1}a_{j2}\dots a_{jk}\dots a_{jr},$$

де a_{jk} – стан комірки з порядковим номером k ($k=\overline{1,r}$).

Блок внутрішньої пам'яті або просто **внутрішня пам'ять** – це деякий «пристрій», який в кожний момент часу знаходиться в певному стані. Стани машини Тьюрінга кодуються за допомогою букв внутрішнього алфавіту

$$Q=\{q_0, q_1, \dots, q_n\}.$$

При цьому букви q_i не повинні входити до зовнішнього алфавіту A для будь-якого $i \in \overline{0,n}$. Стан внутрішньої пам'яті часто називають *внутрішнім станом* машини Тьюрінга.

Один з таких станів називається *початковим* (в цьому стані машина Тьюрінга починає працювати) і позначається q_1 (*старт-символ*). Серед станів машини Тьюрінга також виділяється *заключний* стан, який означає закінчення роботи машини (зупинку машини). Для позначення цього стану як правило використовують символ q_0 (*стоп-символ*).

Читаючий елемент – деякий «пристрій», який може переміщуватися вздовж стрічки так, що в кожний такт роботи він «оглядає» певну комірку стрічки. Іноді вважають, що читаючий елемент є нерухомим, а переміщується інформаційна стрічка за допомогою механічного пристрою.

Механічний пристрій – особливий механізм, який в залежності від стану, комірки, що оглядається, і стану внутрішньої пам'яті одночасно може змінити або стан комірки, і (або) перемістити читаючий елемент у сусідню праву (ліву) комірку.

Якщо читаючий елемент оглядає крайню праву комірку і в результаті роботи машина Тьюрінга повинна перемістити його в сусідню (неіснуючу) справа комірку, то вважається, що цей механічний пристрій добудовує порожню комірку. Аналогічна процедура відбувається, якщо читаючий елемент знаходиться у крайній лівій комірці.

Якщо в деякий момент часу внутрішній стан переходить в заключний стан q_0, то робота машини Тьюрінга припиняється і результатом роботи вважається поточний стан інформаційної стрічки.

Якщо при якому-небудь внутрішньому стані q_i ($i\neq 0$) в машині Тьюрінга не відбувається ніяких змін, то вважають, що машина Тьюрінга продовжує працювати нескінченно довго.

Функціональний (програмний) блок являє собою прямокутну таблицю (з двома входами), в якій зберігаються відомості про ті операції, які повинен виконувати механічний

пристрій в залежності від внутрішнього стану машини Тьюрінга і зовнішнього стану комірок інформаційної стрічки.

5.2. Поняття команди, програми, конфігурації машини Тьюрінга.

Операції, які може виконувати машина Тьюрінга, називаються *командами*.

Команди машина Тьюрінга в загальному випадку мають такий вигляд:

$$T(i, j) = q_i a_j \to q_k a_l X\,, \qquad (1)$$

де $a_j, a_l \in A$, $j = \overline{0,m}$, $l = \overline{0,m}$, $q_i, q_k \in Q$, $i = \overline{1,n}$, $k = \overline{0,n}$, $X \in \{R, L, \Lambda\}$, R – right (право), L – left (ліво), Λ – порожній символ.

Отже, команди машини Тьюрінга можуть бути трьох видів:

$q_i a_j \to q_k a_l \Lambda$ або $q_i a_j \to q_k a_l$ – ця команда означає, що до виконання команди машина Тьюрінга знаходиться у стані q_i і читаючий елемент "оглядає" комірку у стані a_j, а потім машина Тьюрінга переходить до стану q_k і ту ж саму комірку переводить у стан a_l;

$q_i a_j \to q_k a_l R$ – теж саме, що й в попередній команді, але читаючий елемент зміщується на одну комірку вправо від попередньої;

$q_i a_j \to q_k a_l L$ – теж саме, але читаючий елемент зміщується на одну комірку вліво.

Таким чином, допустимими елементарними операціями (мікрооперціями) в машині Тьюрінга є такі три операції:

1. Запис у фіксовану комірку зовнішньої пам'яті певної букви зовнішнього алфавіту (при цьому попередній вміст комірки стирається);
2. Перехід до нового внутрішнього стану (який може збігатися з попереднім станом). При цьому в БВП вміщується відповідна буква внутрішнього алфавіту Q;
3. Фіксування комірки пам'яті за однією з адрес: R, L, Λ. При цьому відбувається зміна положення ЧЕ. Точніше, за адресою Λ положення ЧЕ не змінюється, за адресою R ЧЕ зсувається вправо на одну комірку, за адресою L – вліво.

Команди машини Тьюрінга вміщуються в комірках функціонального блоку, за таким принципом: в комірці з ім'ям «$q_i a_j$» знаходиться вираз «$q_k a_l X$».

Скінченну сукупність команд виду (1)*, які повинна виконувати машина Тьюрінга для розв'язання поставленої задачі, називають* ***програмою (алгоритмом)*** *або* ***функціональною схемою.***

Конфігурацією машини Тьюрінга (на j-ому такті роботи) називають сукупність, яка утворена:

- послідовністю всіх букв зовнішнього алфавіту, які знаходяться в комірках інформаційної стрічки:

$$a_{j1} a_{j2} \ldots a_{jk} \ldots a_{jr}\,,$$

- станом внутрішньої пам'яті q_i;
- номером k–ої комірки, яку оглядає читаючий елемент.

Сукупність цих даних можна записати одним словом:

$$a_{j1} a_{j2} \ldots q_i\, a_{jk} \ldots a_{jr},$$

яке називається *машинним словом*, що описує стан машини Тьюрінга у даний такт її роботи.

5.3. Алгоритмічний процес в системі Тьюрінга.

Робота машини Тьюрінга полягає в тому, що вона з даного «стану» після одного такту роботи механічного пристрою переходить в наступний стан, що визначається певною командою і т.д.

Або, іншими словами, робота машина Тьюрінга полягає в переході від одного машинного слова (конфігурації) до іншого машинного слова, яке описує наступний стан машини Тьюрінга.

Отже, машина Тьюрінга повністю визначається:

а) зовнішнім алфавітом $A=\{a_0, a_1, \dots, a_m\}$ (a_0 – порожній символ);

б) внутрішнім алфавітом $Q=\{q_0, q_1, \dots, q_n\}$ (q_0 – заключний стан, q_1 – початковий стан);

в) програмою (алгоритмом), яка містить команди виду (1).

Зауваження. Ні які дві команди виду $q_i a_j \to q_k a_l X$, $q_p a_r \to q_s a_t X$, що входять до програми, не повинні мати однакові ліві частини.

Говорять, що слово α "переробляється" машиною Тьюрінга в слово β, якщо від слова α машина Тьюрінга після виконання скінченної кількості команд приходить до слова β, яке сприймається в положенні «стоп».

Приклад 4. Для двох довільних натуральних чисел *n* і *m*, записаних за допомогою символу "1", побудувати машину Тьюрінга, яка знаходить їх суму, при цьому машина повинна починати і закінчувати свою роботу, оглядаючи ліву крайню (непорожню) комірку.

Для розв'язування поставленої задачі створюється нова машина Тьюрінга за допомогою послідовності команд "*Файл* \ *Нова машина* \ *Тьюрінга*". Вводиться зовнішній алфавіт $A=\{1,+, a\}$, де символом "а" за замовчуванням заповнюються порожні комірки стрічки машини Тьюрінга, та задаються три внутрішніх стани: q_0, q_1, q_2. Після цього у вікні редагування алгоритму вводиться алгоритм (рис. 6). Введення команд алгоритму здійснюється за допомогою панелі набору, що викликається натисненням правої кнопки миші на відповідній клітинці алгоритму, або ж за допомогою безпосереднього ("ручного") введення команди.

Після цього робота алгоритму перевіряється на контрольних вхідних даних. Результат застосування кожної команди виводиться в "Таблиці виконання" (рис. 6).

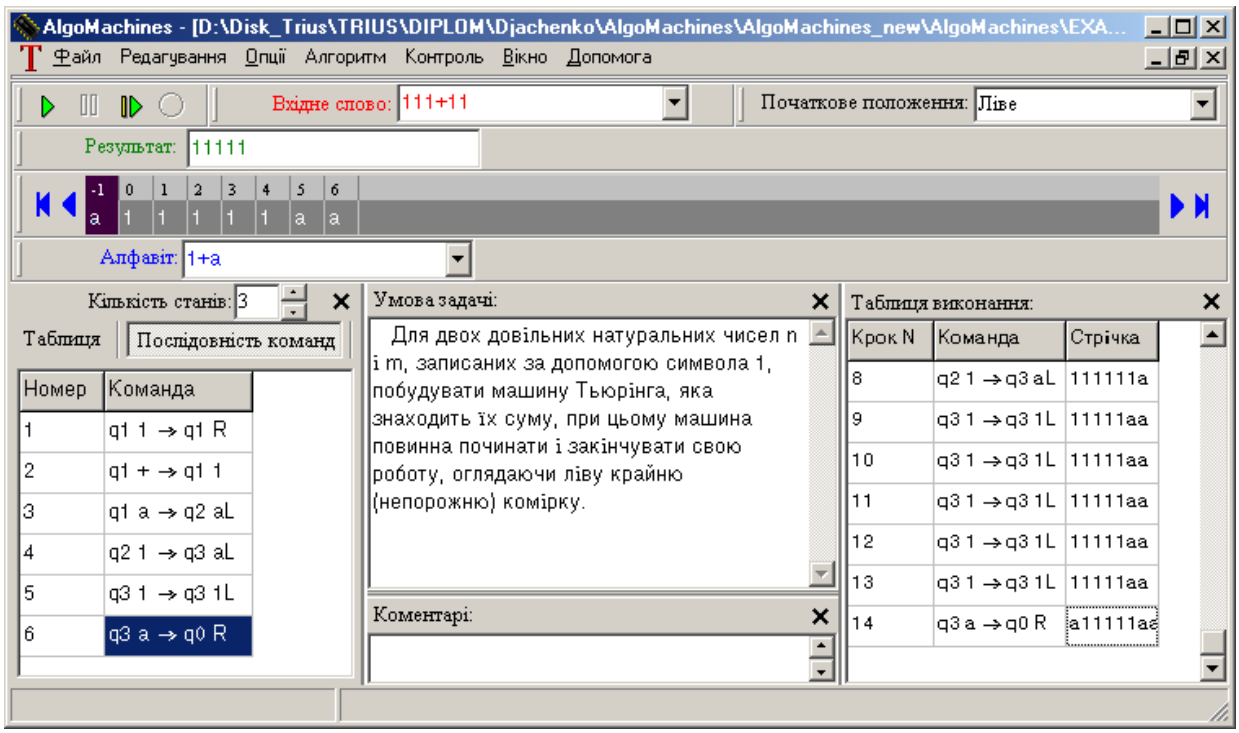

Рис. 6. Алгоритм знаходження суми двох натуральних чисел

Приклад 5. Довести, що функція $f(x)=x-2$, де $x \in N_0=\{0\} \cup N$, обчислювальна за Тьюрінгом.

Для доведення необхідно побудувати машину Тьюрінга, що обчислює значення даної функції при $x \geq 2$, тобто коли значення існують, і працює нескінченно довго в протилежному випадку:

$$f(x)=\begin{cases} \text{не існує, якщо } x=0 \vee x=1, \\ x-2, \text{ якщо } x \geq 2. \end{cases}$$

Будемо вважати, що початкова конфігурація має вигляд

$$q_1 01^x 0 = q_1 01\underbrace{1...1}_{x}0,$$

а заключна конфігурація при $x \geq 2$ повинна мати такий вигляд:

$$q_0 01^{x-2} 0 = q_0 0001\underbrace{1...1}_{x-2}0.$$

Отже, треба побудувати машину Тьюрінга, що починає свою роботу з лівого крайнього положення і таку, що

$$T(01^x0) = 0001^{x-2}0$$

для будь-яких $x \geq 2$, а для $x < 2$ вона буде працювати нескінченно довго.

Програма машини Тьюрінга для обчислення значень даної функції та результати обчислень при $x = 5$ і $x = 0$ подані відповідно на рис. 7 і рис. 8.

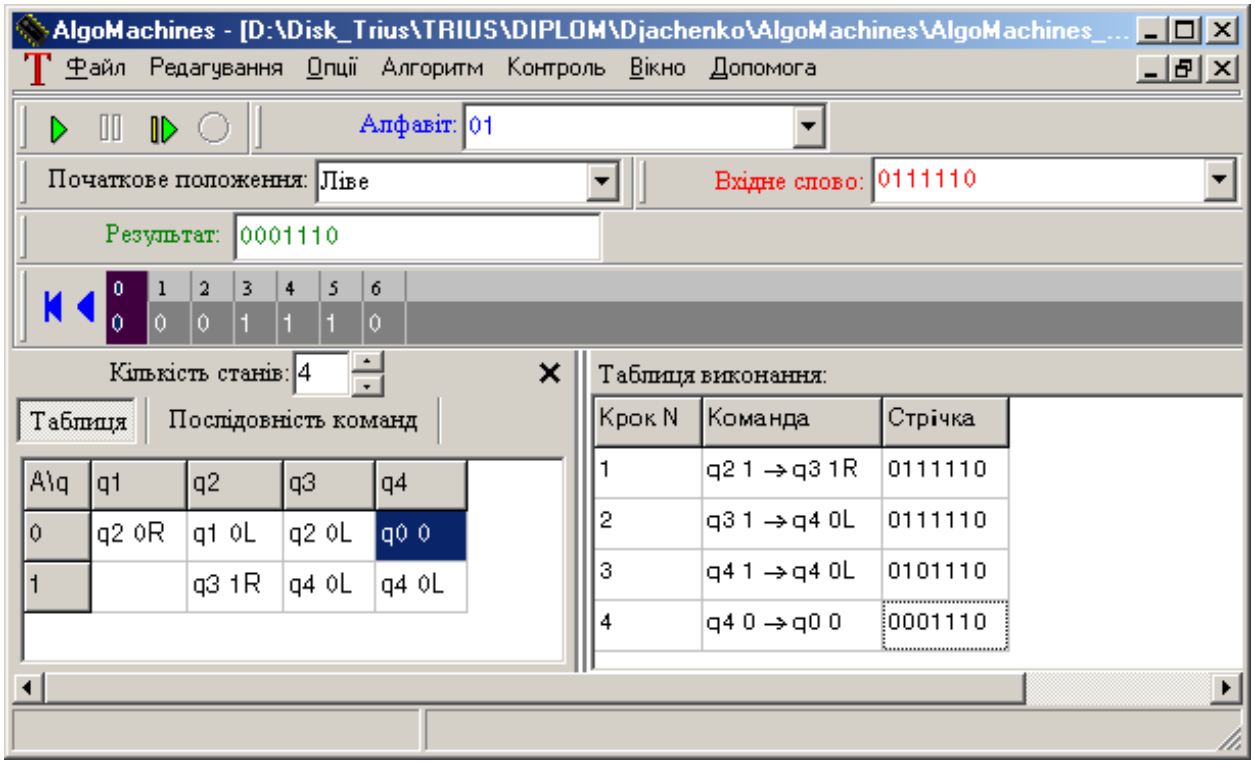

Рис. 7.

6. Особливості реалізації алгоритмічної системи Поста

6.1. Будова машини Поста.

Машина Поста [2, 10] – абстрактна машина, яка складається з стрічки і каретки (читаючий і записуючий пристрій). Стрічка потенційно нескінченна і розділена на комірки однакового розміру. У кожну комірку стрічки заноситься один символ двійкового алфавіту. Один із символів двійкового алфавіту називається *міткою* і традиційно позначається «V» (або іншим символом, наприклад «1»), другий символ – *порожній символ* (пропуск) (або «0»).

Якщо в комірку занесена мітка, то ця комірка називається *відміченою*, якщо в комірці мітки немає, то вона вважається порожньою або *невідміченою*.

Каретку можна переміщувати вздовж стрічки вліво і вправо. Коли вона нерухома, вона знаходиться проти однієї з комірок стрічки. Говорять, що каретка “оглядає” цю комірку і така комірка називається *поточною*.

За один такт роботи машини каретку можна переміщувати на одну комірку вліво або вправо. Крім того, каретка може “розпізнати” наявність чи відсутність мітки в поточній комірці, або встановити мітку в порожній комірці, або видалити мітку з відміченої комірки.

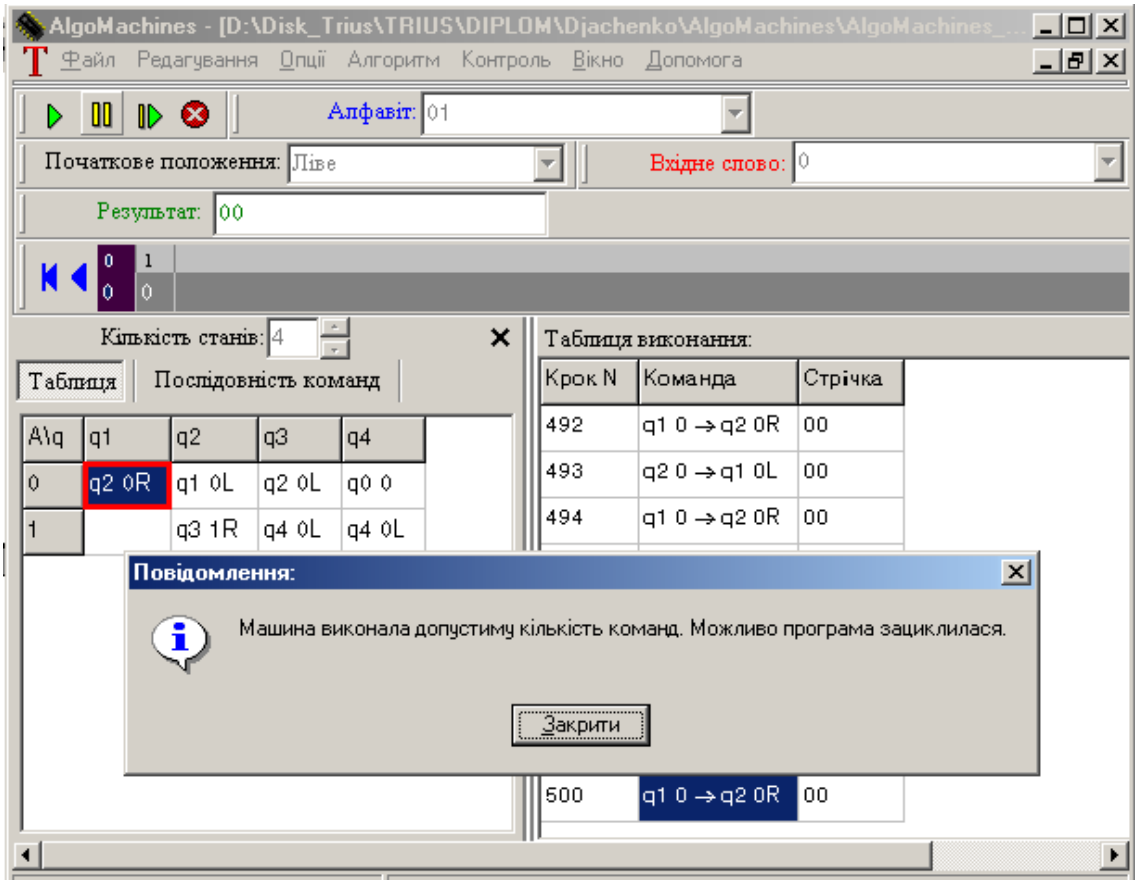

Рис. 8.

6.2. Система команд і робота машини Поста.

Систему команд машини Поста подано в таблиці 1, де *n* – номер поточної команди, *m*, *m*1, *m*2 – *посилання*: номери команд, що будуть виконуватися наступними.

Таблиця 1

Система команд машини Поста

Команда			**Призначення команди**
Номер поточної команди	**Позначення команди**	**Посилання**	
n	→	*m*	Перемістити каретку на одну комірку вправо і перейти до виконання команди з номером *m*.
n	←	*m*	Перемістити каретку на одну комірку вліво і перейти до виконання команди з номером *m*.
n	V	*m*	У поточну комірку занести мітку і перейти до виконання команди з номером *m*. Виконання цієї команди можливе лише в тому випадку, коли поточна комірка порожня. У протилежному випадку команда вважається невиконуваною.
n	β	*m*	Видалити мітку з поточної комірки і перейти до виконання команди з номером *m*. Виконання цієї команди можливе лише в тому випадку, коли поточна комірка містить мітку. В протилежному випадку команда вважається невиконуваною.

n	?	*m*1, *m*2	Команда *переходу*. Якщо поточна комірка не містить мітку, то перейти до виконання команди з номером *m*1, інакше, тобто якщо комірка містить мітку, – на команду з номером *m*2.
n	!		Команда *зупинки* машини Поста.

Робота машини Поста полягає в тому, що каретка переміщується вздовж стрічки і встановлює або видаляє мітки у відповідних комірках. Щоб машина Поста працювала, потрібно задати деякий алгоритм, у комірках інформаційної стрічки розмістити певним чином мітки (зокрема можна всі комірки залишити порожніми) і вказати початкове положення каретки (розташувати каретку проти однієї з комірок).

Упорядковану скінченну сукупність команд, які повинна виконувати машина Поста для розв'язання поставленої задачі, називають ***алгоритмом*** **(*програмою*).**

Послідовність команд машини Поста буде алгоритмом, якщо:

1) на *n*-му місті цієї послідовності стоїть команда з номером *n*;
2) кожному посиланню, що входить до складу команди, відповідає конкретна команда заданої послідовності.

Робота машини Поста за заданим алгоритмом відбувається в такий спосіб.

Машина переводиться в початковий стан і виконує першу команду алгоритму. Ця команда виконується за один крок, після чого машина виконує ту команду, номер якої відповідає посиланню першої команди. Ця команда також виконується за один крок, після чого виконується та команда, номер якої відповідає посиланню попередньої команди і так далі доти, поки не буде здійснено перехід на команду зупинки. Якщо для деякого початкового стану в процесі роботи машини Поста ніколи не буде здійснено перехід на команду зупинки, то в такому випадку кажуть, що до даного початкового стану задана машина незастосовна і машина працює нескінченно довго.

Взагалі кожна команда виконується за один крок, а перехід від виконання однієї команди до виконання іншої відбувається за правилом:

нехай на *k*-ому кроці виконувалася команда з номером *n*, тоді:

1) якщо ця команда має єдине посилання *m*, то на (*k*+1)-ому кроці виконується команда з номером *m*;

2) якщо ця команда має два посилання *m*1 і *m*2, то на (*k*+1)-ому кроці виконується одна з двох команд – з номером *m*1 або з номером *m*2 в залежності від наявності чи відсутності мітки у поточній комірці.

Можливі такі випадки зупинки машини Поста:

1) при виконанні алгоритму на деякому кроці буде здійснено перехід на команду зупинки; в цьому випадку алгоритм вважається виконаним і робота машини припиняється – відбувається результативне завершення роботи машини;

2) при виконанні алгоритму на деякому кроці буде здійснено перехід на команду, яка є невиконуваною (див. табл. 1); у цьому випадку відбувається безрезультативне завершення роботи машини.

Розглянемо використання програми AlgoMachines для створення і виконання алгоритмів в алгоритмічній системі Поста на прикладі.

Приклад 6. Побудувати алгоритм для машини Поста, за допомогою якого знаходиться різниця двох натуральних чисел *n* і *m*, де *n* – зменшуване, *m* –від'ємник і $n \geq m$. При цьому між словами, що є поданням чисел *n* і *m* (від'ємник) на інформаційній стрічці, знаходиться лише одна порожня комірка. Початкове положення каретки відповідає крайньому правому положенню.

Для побудови алгоритму можна скористатися, наприклад такими міркуваннями: необхідно організувати роботу машини так, щоб каретка послідовно видаляла мітки вкінці слова, що відповідає поточному стану від'ємника, і на початку слова, що відповідає

поточному стану зменшуваного, доти, поки слово, що відповідає поточному стану від'ємника, не стане порожнім.

Для розв'язування поставленої задачі необхідно створити нову машину Поста за допомогою послідовності команд "*Файл* \ *Нова машина* \ *Поста*". Після цього ввести вхідне слово, задати початкове положення каретки, ввести алгоритм і виконати його. Введення та редагування команд алгоритму здійснюється за допомогою меню, що з'являється при натисненні правої кнопки мишки.

Один з варіантів алгоритму розв'язування поставленої задачі та результат її виконання при n=3 і m=2 подано на рис. 9.

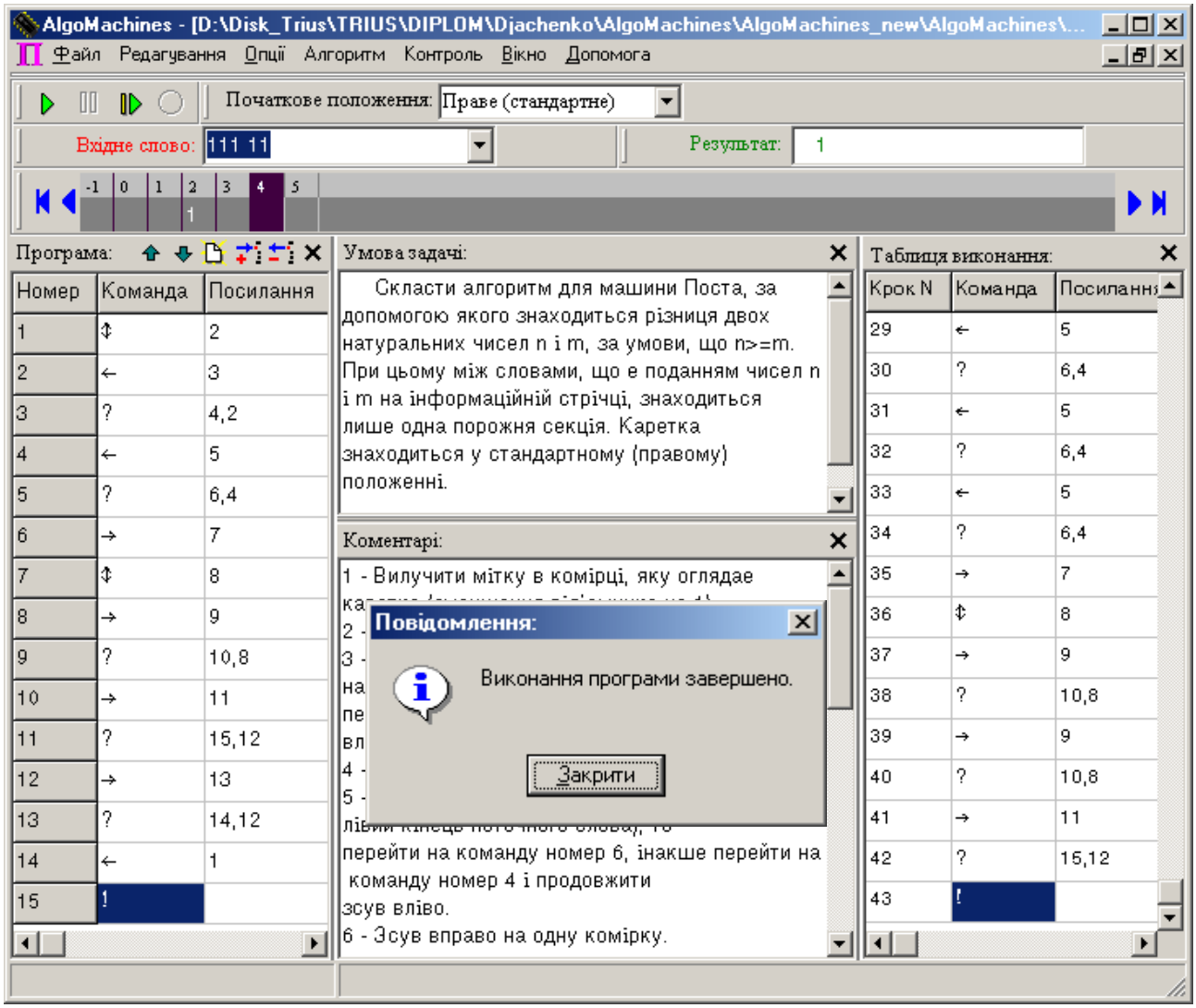

Рис. 9. Алгоритм знаходження різниці двох натуральних чисел в системі Поста

7. Режим підготовки контрольних завдань

Режим підготовки контрольних завдань призначений для викладача. Доступ до даного режиму захищений паролем.

Завдання для контролю являють собою послідовність завдань на *аналіз* чи *синтез* (побудову) алгоритмів розв'язування задач в алгоритмічних системах Маркова, Тьюрінга, Поста. Ця послідовність зберігається в окремому файлі з розширенням *.tst.

Вікно редагування контрольних завдань містить дві закладки: "*Оцінювання*" та "*Завдання*".

Закладка “*Оцінювання*” містить параметри, загальні для всієї послідовності завдань файлу контролю (рис. 10), проте індивідуальні для кожного файлу з контрольним завданням (*.tst). Зокрема це такі параметри як:

– “*Система оцінювання*” – режим обрання системи оцінювання (двобальної, чотирибальної, дванадцятибальної тощо), при цьому діапазон балів для відповідної оцінки розраховується автоматично після введення викладачем максимальної кількості відсотків для даної оцінки. Загальна сума балів та кількість контрольних завдань відображається у верхній частині вікна. Замість числових значень оцінок можна вводити текстові значення, наприклад “зараховано”, “незараховано”;

– “*Запит на підтвердження відповіді*” – якщо ця опція встановлена, то після виконання контрольного завдання буде виводитися запит на підтвердження відповіді;

– “*Показувати правильні відповіді*” – якщо ця опція встановлена, то у випадку неправильної відповіді на завдання, буде виводитися правильна відповідь;

– “*Використовувати інтерпретатор при розв'язуванні задач синтезу*” – якщо ця опція встановлена, то при виконанні контрольного завдання на синтез алгоритму студент має можливість виконувати перевірку алгоритму на правильність за допомогою інтерпретатора AlgoMachines, інакше він може лише вводити алгоритм;

– “*Запит паролю на вихід з режиму контролю*” – якщо ця опція встановлена, то для виходу з режиму контролю потрібно ввести пароль;

– “*Час на виконання завдань контролю*” – встановлюється максимальний час на виконання контрольних завдань.

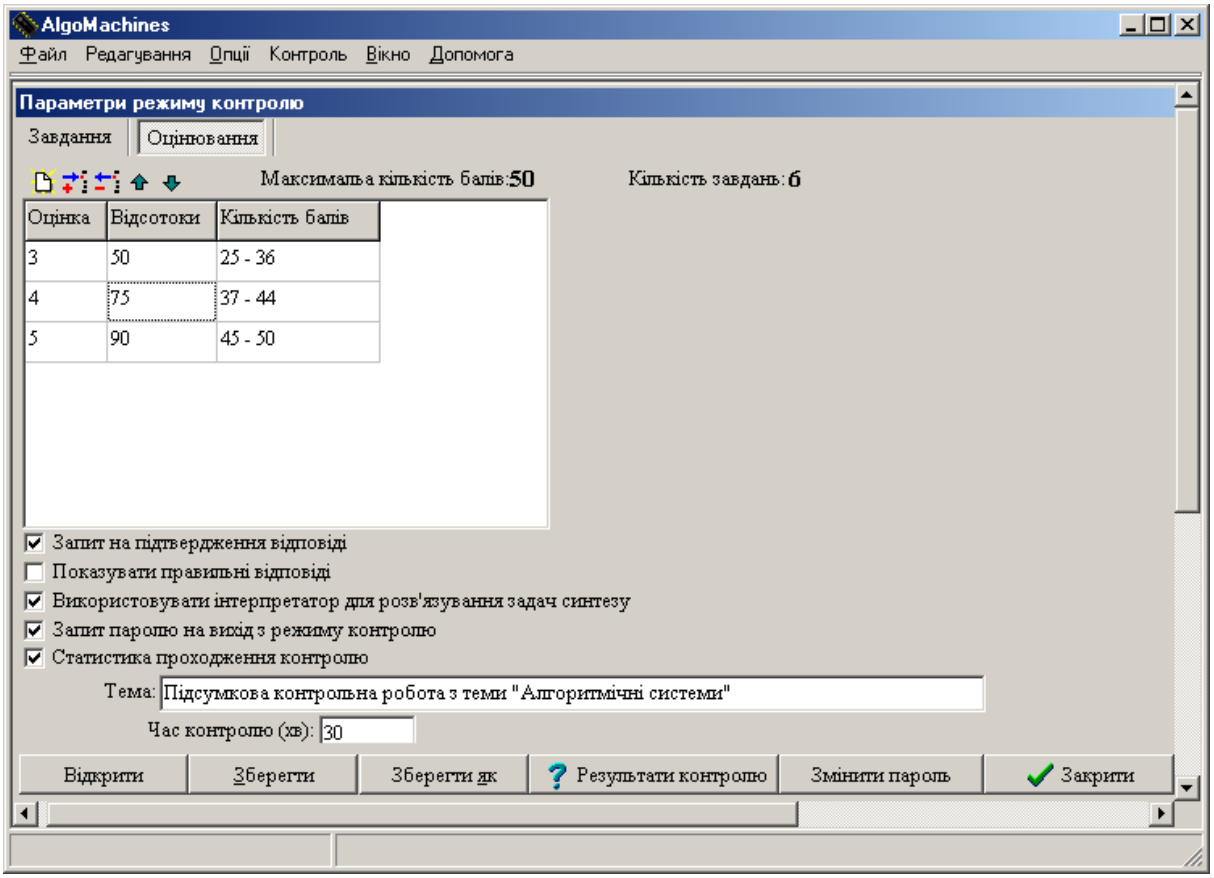

Рис. 10. Закладка “Оцінювання” з параметрами режиму контролю

Закладка “*Завдання*” (рис. 11) містить послідовність контрольних завдань на аналіз і синтез алгоритмів у заданих алгоритмічних системах. При цьому можна комбінувати завдання на синтез і аналіз алгоритмів у різних алгоритмічних систем, або навпаки, створювати завдання виключно на синтез (аналіз) в одній з алгоритмічних систем.

Для завдань аналізу готових алгоритмів встановлюються наступні параметри:

– кількість балів за завдання;

– початкове положення читаючого елементу (тільки для машин Тьюрінга і Поста);
– алгоритм для аналізу задається вказівкою на файл (типу nam, tur, pst), для зміни цього параметру є кнопка –“Вибрати”, а для редагування алгоритму засобами середовища – кнопка – “Відкрити”;
– множина вхідних слів.

Завдання студента полягає в знаходженні результату застосування алгоритму до кожного вхідного слова. При цьому відповідь вважається правильною, якщо правильно визначені всі результати.

Для завдань синтезу алгоритмів встановлюються наступні параметри:

– кількість балів за завдання;
– алфавіт, який треба використовувати при створенні алгоритму (тільки для алгоритмічних систем Маркова та Тьюрінга);
– множина вхідних слів та відповідних результатів виконання алгоритму (дана множина призначена для перевірки відповідності введеного студентом алгоритму розв’язування поставленій задачі, тому для якісної перевірки правильності алгоритму слід задавати якомога більше вхідних слів);
– варіант правильної відповіді задається вказівкою на файл (типу nam, tur, pst), для зміни цього параметру є кнопка “Вибрати”, а для його редагування засобами середовища – кнопка “Відкрити”;
– умова завдання, яке видається студенту при розв'язуванні завдання.

Завдання студента полягає в створенні алгоритму розв'язування поставленої задачі.

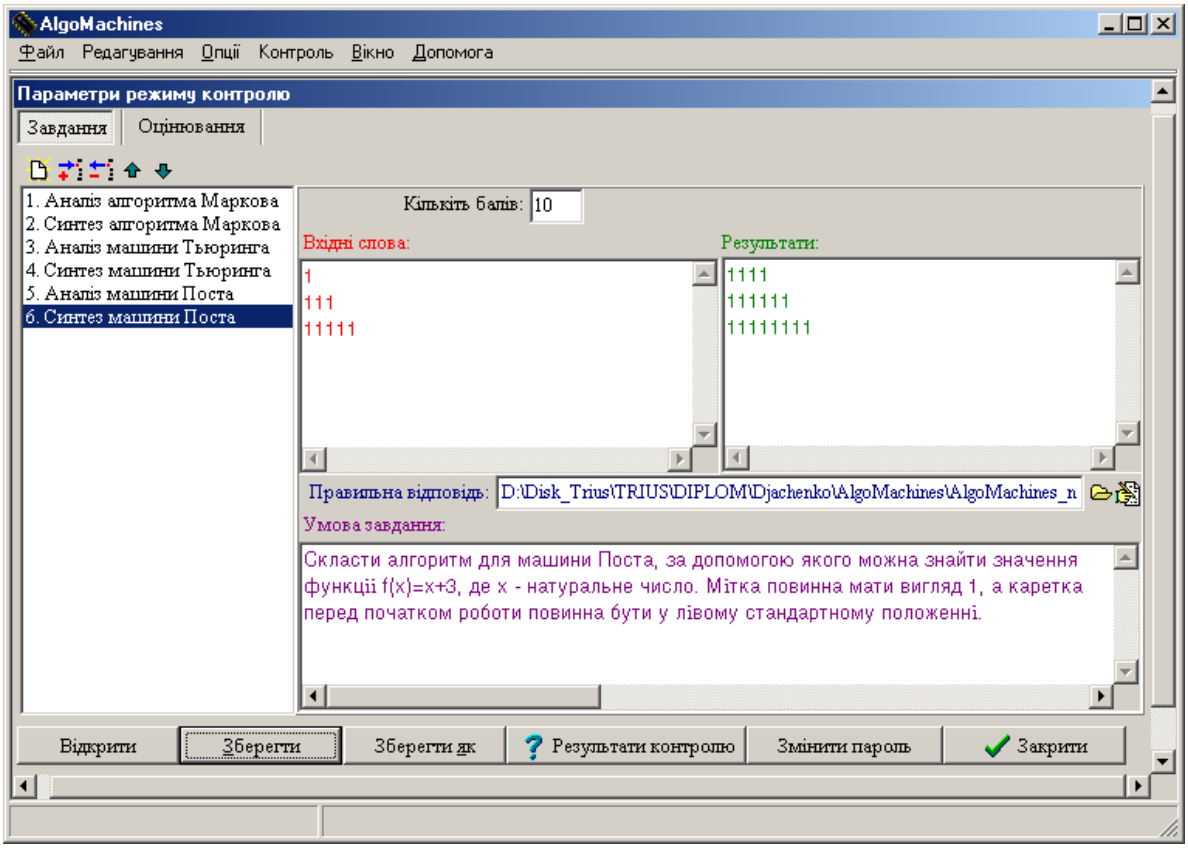

Рис. 11. Закладка “Завдання” режиму контролю

8. Режим контролю

Доступ до режиму контролю здійснюється вибором пункту меню “*Контроль \ Розпочати контроль*”. Перед початком контролю студент повинен ввести своє прізвище, ім'я, курс та групу, де він навчається.

Потім студент за вказівкою викладача обирає файл з контрольними завданнями.

Далі студент послідовно виконує контрольні завдання. Для прийняття програмою відповіді студента на поточне завдання і переходу до наступного завдання треба натиснути кнопку “Ввести”.

Після виконання всіх контрольних завдань (або завершення контролю за часом) виводиться вікно з результатами контролю (рис. 12).

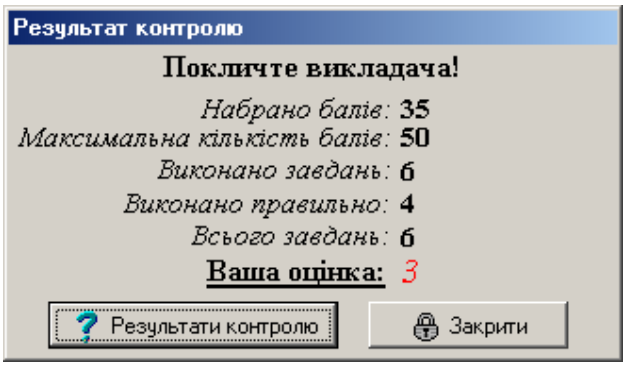

Рис. 12. Вікно з результатами контролю

Перед виходом з режиму контролю студент, натиснувши на кнопку “Результати контролю”, може проглянути таблицю з підсумковими результатами і протокол виконання контрольних завдань, який містить відомості про зроблені помилки і правильні відповіді на всі завдання.

Висновки

Використання у навчальному процесі програми AlgoMachines сприяє підвищенню інтересу студентів математичних і комп’ютерних спеціальностей до таких фундаментальних математичних дисциплін, як математична логіка, теорія алгоритмів, дискретна математика; надає викладачам зазначених дисциплін можливість збільшити долю самостійної роботи студентів, автоматизувати поточний і підсумковий контроль при навчанні теорії алгоритмів.

Література

1. Калужнін Л. А., Королюк В. С. Алгоритми і математичні машини. – К.: Рад. школа, 1964. – 283 с.
2. Капітонова Ю. В., Кривий С. Л., Летичевський О. А., Луцький Г. М., Печурін М. К. Основи дискретної математики: Підручник. Том 1.– К.: В-во “ЛітСофт”, 2000. – 380 с.
3. Капітонова Ю. В., Кривий С. Л., Летичевський О. А., Луцький Г. М., Печурін М. К. Основи дискретної математики: Підручник. Том 2. – К.: В-во “ЛітСофт”, 2000. – 370 с.
4. Лиман Ф. М. Математична логіка і теорія алгоритмів: Навчальний посібник. – Суми: Видавництво “Слобожанщина”, 1998. – 152 с.
5. Мальцев М. И. Алгоритмы и рекурсивные функции. – М.: Наука, 1986. – 368 с.
6. Марков А. А., Нагорный Н. М. Теория алгорифмов. – М.: Наука,1984. – 432 с.
7. Мендельсон Э. Введение в математическую логику. – М.: Наука, 1976. – 320 с.
8. Триус Ю. В. Комп’ютерно-орієнтований навчально-методичний комплекс курсу “Математична логіка і теорія алгоритмів” // Матеріали міжнародної науково-методичної конференції “Інформатизація освіти України: стан, проблеми, перспективи”: Тези доповідей. – Херсон: Айлант, 2005. – С. 94–95.
9. Успенський В.А., Семенов А.Л. Теория алгоритмов: основные открытия и приложения. – М.: Наука, Гл. ред. физ.-мат. лит., 1987. – 288 с.
10. Успенский В. А. Машина Поста. – М.: Наука, 1988. – 96 с.