УДК 517.62

Применение синтаксического анализатора при построении графиков функций

Application for parser plotting functions

Адекенова А.Н., Устинова Л.В.

Карагандинский государственный университет им. Е.А.Букетова (E-mail: adekenova@mail.ru)

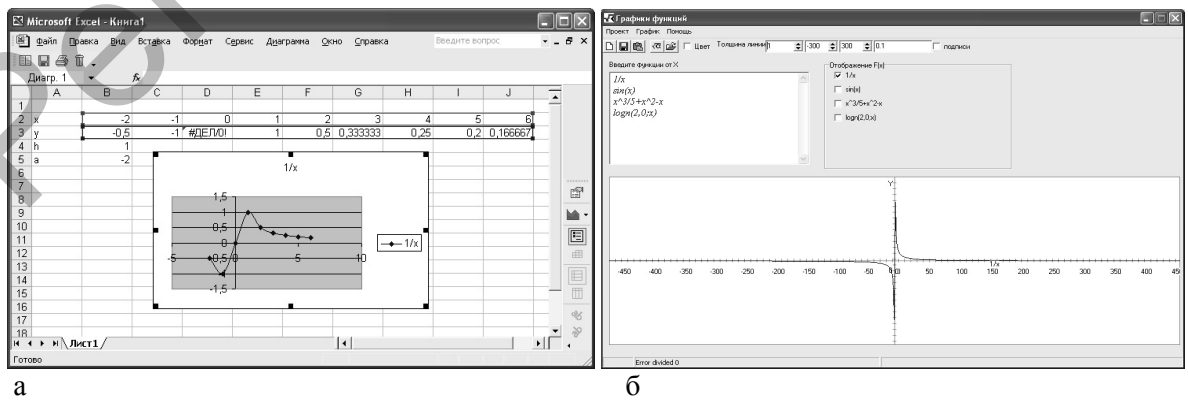
Мақалада функциялардың графиктерін салу барысында синтаксистік талдауышты қолдану мүмкіндігі көрсетілген. Қолданбалы математика және информатика кафедрасының мұғалімдерімен екі негізгі модельдерден тұратын программа құрылған: синтаксистік талдауыш пен функциялардың графиктерін салу. Сонымен қатар математикалық өрнек құрамның сипатталуы мен араларындағы қарым-қатынастары көрсетілген синтаксистік талдауының негізгі үлгісі құрастырылған. Программа жеке модульдерден тұратын болған соң, оларды өңдеуге мүмкіндік бар, сондықтан оларды басқа программаларда да қолдануға болады. Жұмыс барысында математикалық өрнекті талдау алгоритмдері құрылған. Берілген программаны информатика мен жоғарғы сыныптарда математика сабақтарында қолдануға болады.

This article discusses the use of the parser to construct graphs of functions. Team of trainers program was established, which consists of two main modules: the parser and graphing functions. In this case, was developed by the general scheme of parsing a mathematical expression describing the structures and their interactions. Since the program is divided into modules, thus can be easily modified, allowing the use of modules developed in other programs. When working on the project were created algorithms to produce parse mathematical expressions. The presented program can be used as the science lessons, and on the lessons of algebra in high school.

В школьном курсе элементарной алгебры тема анализа функций является одной из самых сложных. При изучении данной темы школьники должны научиться исследовать и строить графики функций одной переменной, используя все известные характеристические точки функции, включая корни, точки разрыва первого и второго рода и т.д.

Существует программное обеспечение, которое может решать подобные задачи, например, «Excel» [1]. Оно имеет понятный пользовательский интерфейс, ориентированный на среднего пользователя, что делает использование подобных средств не всегда доступным. Кроме того, при построении функций, имеющих точки разрыва, данные программы не обрабатывают исключительные ситуации, а требуется вмешательство пользователя (рис. 1).

Существующее программное обеспечение, которое может решать подобные задачи, является универсальным, например Eurica или MathCad. Оно имеет сравнительно сложный пользовательский интерфейс, ориентированный на пользователя, прослушавшего, как минимум, институтский курс высшей математики, что делает использование подобных средств школьниками невозможным. Разрабатываемая программа позволит школьникам проверить свои знания при изучении указанной темы. Дополнительно созданный компонент синтаксического анализатора может использоваться студентами при создании приложений в курсовых проектах по предметам «Программирование на Delphi», «Компьютерная графика» [2].



а б

Рис. 1. Результат построения графика функции f=1/x: *а* — MS Excel; *б* — приложение «Построение графика функции»

Сформулируем функциональные требования к программе:
– ввод аналитического представления функции одной переменной и длительное хранение его в системе;
– ввод и изменение интервала определения функции;
– ввод и корректировка шага аргумента;
– построение таблицы значений функции на заданном интервале или изображение графика функции на заданном интервале, включая условие, что на указанном интервале она может иметь точки разрыва;
– сохранение таблицы значений функции на заданном интервале в текстовый файл;
– сохранение графических результатов в файл;
– построение графиков выделенных функций;
– изменение параметров построения — изменение цвета и толщины линии графиков;
– сохранение результатов табулирования функций;
– определение синтаксических ошибок.

Исходные данные:
– аналитическое задание функции;
– интервал определения функции;
– шаг изменения аргумента, определяющий количество точек на интервале;
– текстовый файл, содержащий функции, шаг табулирования, интервал построения.

Требования к надежности:
– предусмотреть контроль вводимой информации;
– предусмотреть блокировку некорректных действий пользователя при работе с системой.

Тогда результатом работы программы будет:
– построение графиков функций на экране;
– сохранение заданных графиков функции в графический файл;
– сохранение результатов табулирования функции с заданными параметрами.

Разработаем функциональные диаграммы уточнения спецификации программы для построения таблиц/графиков функций одной переменной.

Диаграмма, показанная на рисунке 2 а, является диаграммой верхнего уровня. На ней хорошо видно, что является исходными данными для программы и каких результатов работы от нее ожидают [3].

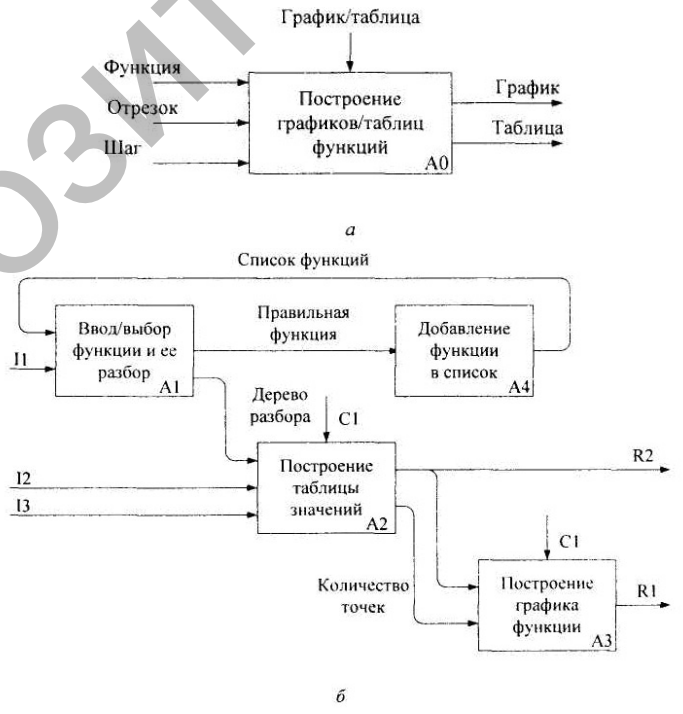


Рис. 2. Функциональные диаграммы для системы исследования функций: *а* — диаграмма верхнего уровня; *б* — уточняющая диаграмма

Диаграмма, представленная на рисунке 2 б, уточняет функции программы. На ней показаны четыре блока: Ввод/выбор функции и ее разбор, Добавление функции в список, Построение таблицы значений и Построение графика функции. Для каждого блока определены исходные данные, управляющие воздействия и результаты. Согласно правилам наименования входов/выходов, имеющих продолжение на родительской диаграмме, на диаграмме использованы следующие обозначения:

– I1 — функция;
– I2 — отрезок;
– I3 — шаг;
– C1 — вид график/таблица;
– R1 — график функции на отрезке;
– R2 — таблица значений функции на отрезке.

Детализируя эту диаграмму, получаем три процесса: Ввод/выбор функции и ее разбор, Построение таблицы значений функции и Построение графика функции. Для хранения функций добавляем хранилище функций. Затем определяем потоки данных.

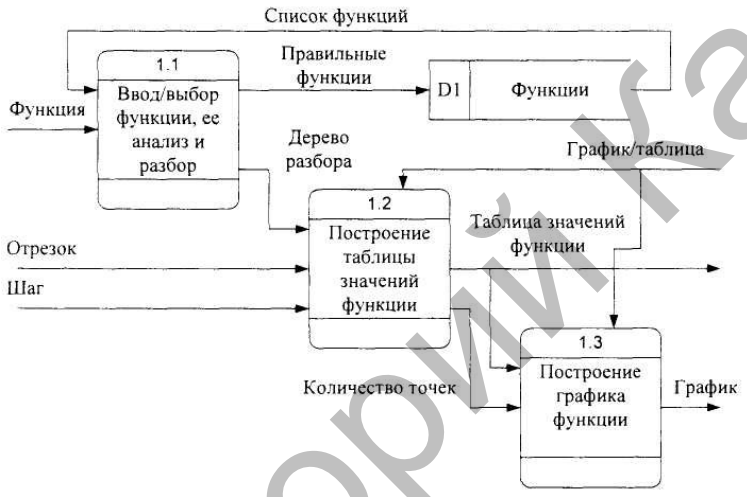


Рис. 3. Детализирующая диаграмма потоков данных системы исследования функций (нотация Гейна-Сарсона)

На этапе составления программы и плана отладки используем метод пошаговой детализации для проектирования структуры программного обеспечения (рис. 3).

Метод пошаговой детализации реализует нисходящий подход и базируется на основных конструкциях структурного программирования. Он предполагает пошаговую разработку алгоритма. Каждый шаг при этом включает разложение функции на подфункции. Так, на первом этапе описывают решение поставленной задачи, выделяя общие подзадачи, на следующем аналогично описывают решение подзадач, формулируя при этом подзадачи следующего уровня. Таким образом, на каждом шаге происходит уточнение функций проектируемого программного обеспечения. Процесс продолжают, пока не доходят до подзадач, алгоритмы решения которых очевидны.

Декомпозируя программу методом пошаговой детализации, следует придерживаться основного правила структурной декомпозиции, следующего из принципа вертикального управления: в первую очередь, детализировать управляющие процессы декомпозируемого компонента, оставляя уточнение операций с данными напоследок. Это связано с тем, что приоритетная детализация управляющих процессов существенно упрощает структуру компонентов всех уровней иерархии и позволяет не отделять процесс принятия решения от его выполнения: так, определив условие выбора некоторой альтернативы, сразу же вызывают модуль, ее реализующий.

Разработаем алгоритм программы построения графиков функций одной переменной на заданном интервале изменения аргумента [X1, X2]. Для того чтобы программировать построение графиков функции с точками разрыва первого и второго рода, необходимо аналитически исследовать заданные функции. Данная задача решается при использовании синтаксического анализатора, в нашем случае приведение к ошибке 102 означает разрыв функции, и на данном интервале построений не происходит.

Программа будет взаимодействовать с пользователем через традиционное иерархическое меню, которое содержит пункты: Функция, Отрезок, Шаг, Вид результата, Выполнить и Выход. Для каждого пункта этого меню необходимо реализовать сценарий [4].

Определим, какие фрагменты программы оформим подпрограммами. Во-первых, фрагменты Разбор формулы, Расчет значений функции, Построение графика и Вывод таблицы, так как это достаточно сложные операции. Это — подпрограммы первого уровня, которые определяют структуру программы (рис. 4).

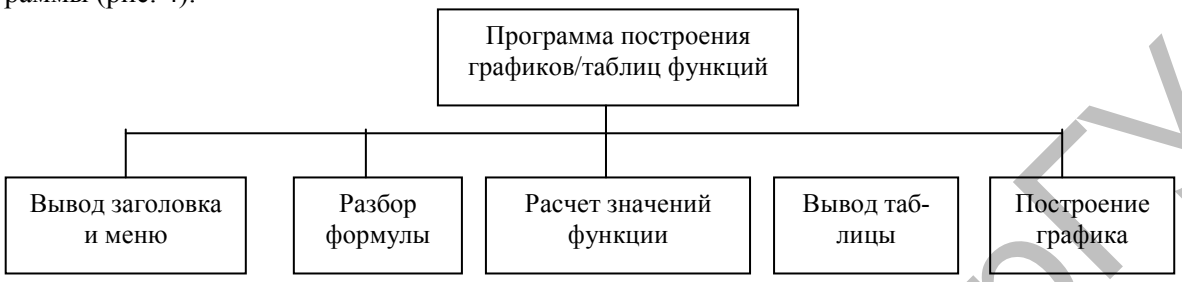


Рис. 4. Структурная схема программы построения графиков/таблиц функций

На следующих шагах необходимо выполнить детализацию алгоритмов подпрограмм. Один из возможных вариантов полной структурной схемы данной программы показан на рисунке 5.

Использование метода пошаговой детализации обеспечивает высокий уровень технологичности разрабатываемого программного обеспечения, так как он позволяет использовать только структурные способы передачи управления.

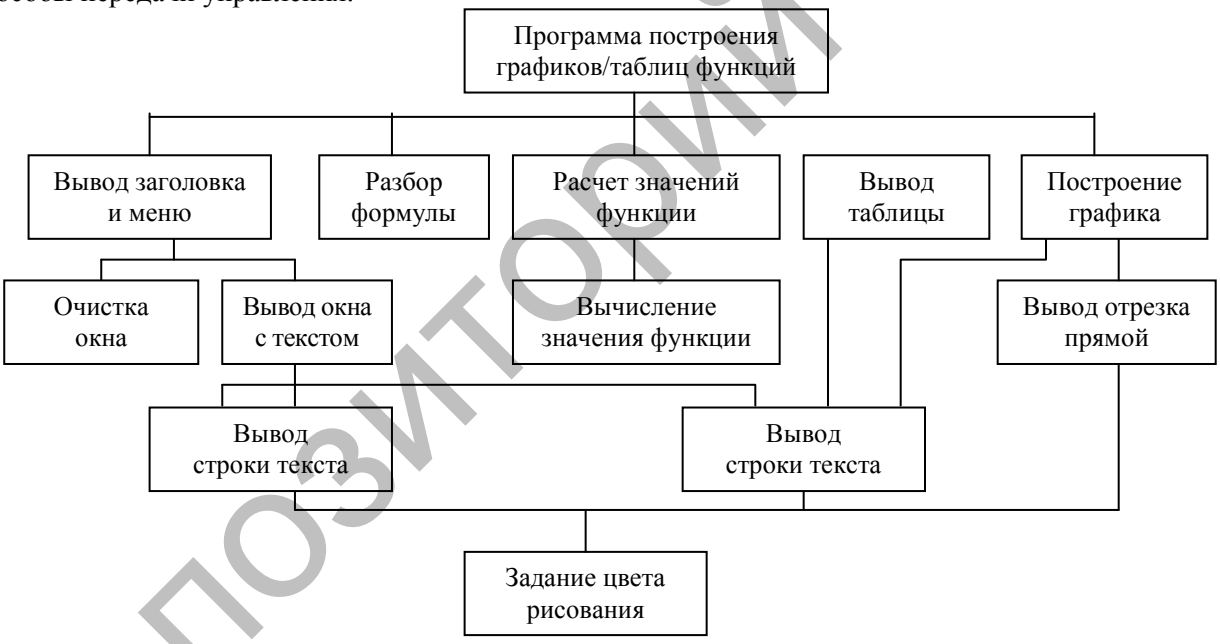


Рис. 5. Полная многоуровневая структурная схема программы построения графиков/таблиц

Для анализа технологичности полученной иерархии модулей целесообразно использовать структурные карты Константайна или Джексона. На структурной карте отношения между модулями представляют в виде графа, вершинам которого соответствуют модули и общие области данных, а дугам — межмодульные вызовы и обращения к общим областям данных.

а б в г

Рис. 6. Обозначения вершин по стандартам IBM, ISO и ANSI: а — модуль, б — подсистема, в — библиотека, г — область данных

Различают четыре типа вершин (рис. 6):

– модуль — подпрограмма;
– подсистема — программа;
– библиотека — совокупность подпрограмм, размещенных в отдельном модуле;
– область данных — специальным образом оформленная совокупность данных, к которой возможно обращение извне.

При этом отдельные части программной системы (программы, подпрограммы) могут вызываться последовательно, параллельно или как сопрограммы (рис. 7).

Чаще всего используют последовательный вызов, при котором модули, передав управление, ожидают завершения выполнения вызванной программы или подпрограммы, чтобы продолжить прерванную обработку.

Под параллельным вызовом понимают распараллеливание вычислений на нескольких вычислителях, когда при активизации другого процесса данный процесс продолжает работу (рис. 8 а). На однопроцессорных компьютерах в мультипрограммных средах в этом случае начинается попеременное выполнение соответствующих программ. Параллельные процессы бывают синхронные и асинхронные. Для синхронных процессов определяют точки синхронизации — моменты времени, когда производится обмен информацией между процессами.

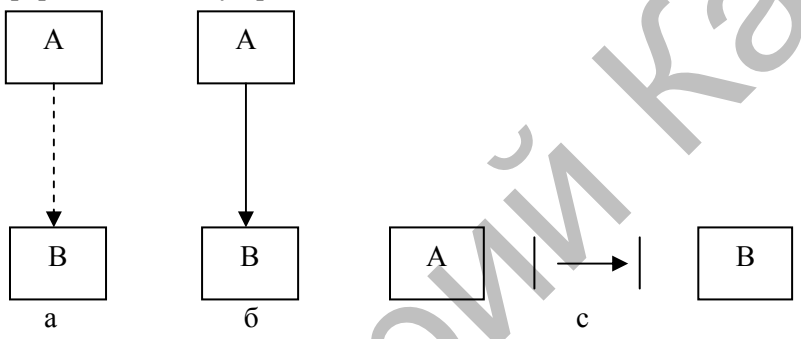


Рис. 7. Обозначения типа вызова: *а* — последовательный вызов; *б* — параллельный вызов, *в* — вызов сопрограммы

Асинхронные процессы обмениваются информацией только в момент активизации параллельного процесса.

Под вызовом сопрограммы понимают возможность поочередного выполнения двух одновременно запущенных программ: например, если одна программа подготовила пакет данных для вывода, то вторая может ее вывести, а затем перейти в состояние ожидания следующего пакета. Причем в мультипрограммных системах основная программа, передав данные, продолжает работать, а не переходит в состояние ожидания, как изображено на рис. 8 б.

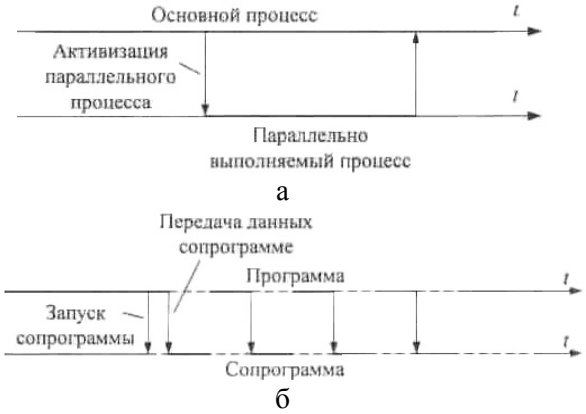


Рис. 8. Диаграммы реализации параллельного вызова *(а)* и вызова сопрограммы *(б)*: — — выполнение; — - — - ожидание

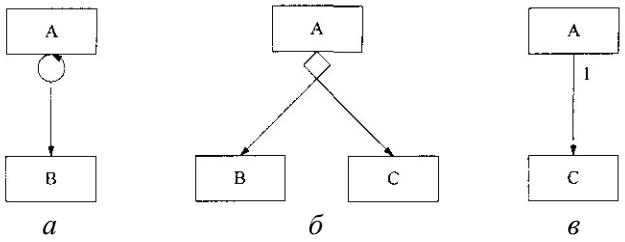


Рис. 9. Обозначения особых условий вызова: *а* — циклический, *б* — условный; *в* — однократный

При необходимости на структурной карте можно уточнить особые условия вызова (рис. 9): циклический, условный и однократный вызов — при повторном вызове основного модуля однократно вызываемый модуль не активизируется [5].

Связи по данным и управлению обозначают стрелками, параллельными дуге вызова, направление стрелки указывает на направление связи (рис. 10).

Структурные карты Константайна позволяют наглядно представить результат декомпозиции программы на модули и оценить ее качество, т.е. соответствие рекомендациям структурного программирования (сцепление и связность) (рис. 10).

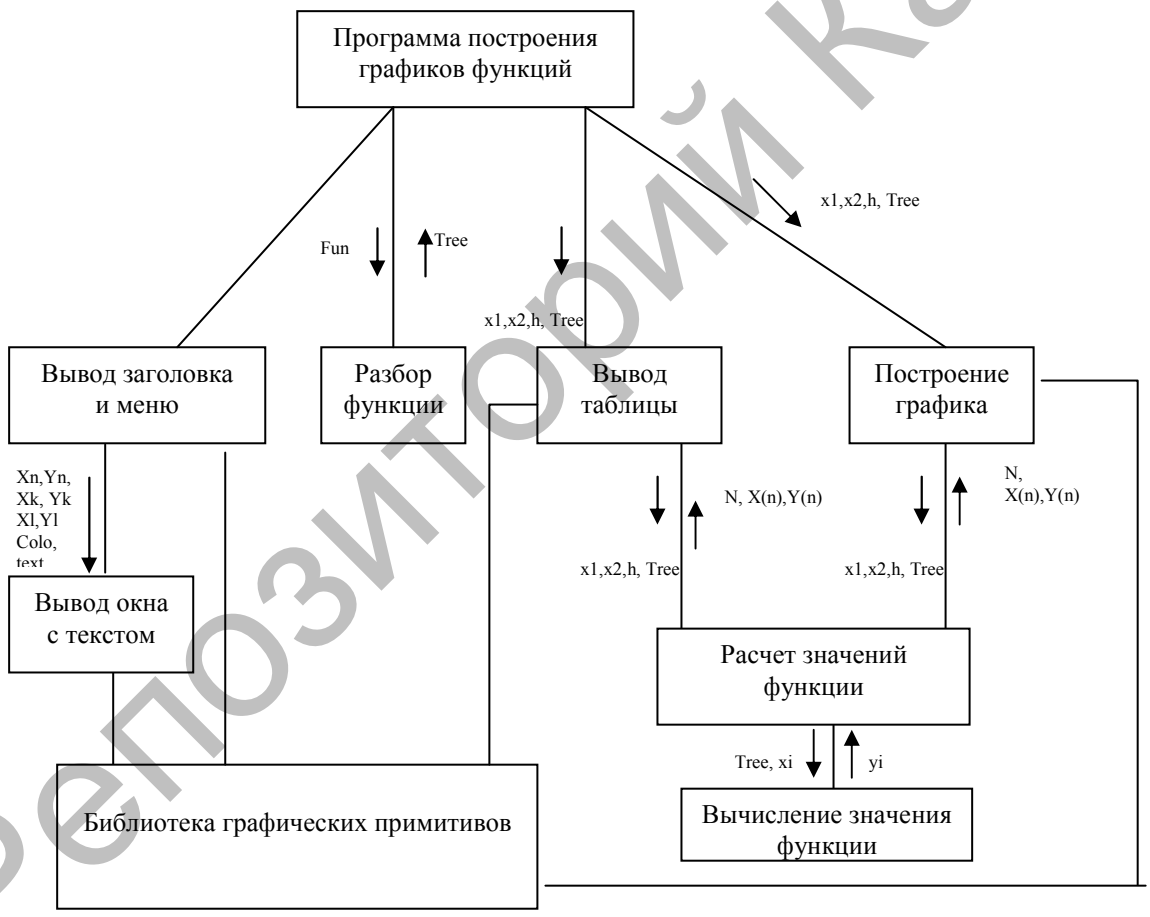


Рис. 10. Структурная карта Константайна для программы построения таблицы/графика функции

Процедуры и функции, представленные на рисунке 10, оформлены в отдельные модули:

– UExpressions — модуль синтаксического анализатора выражений (Разбор функции);
– UOperateur — модуль, содержащий описание операций и функций;
– UPile — модуль лексического анализатора;
– UMain — модуль главной формы, содержащий процедуры, Построение графика, Вывод таблицы, Сохранение результатов табулирования.

Перейдем к этапу проектирования интерфейсов — разработке конкретных операций ввода-вывода для каждого диалога с учетом специфики формы интерфейса.

Приложение «Построение графиков функций» реализовано средствами Delphi 7.0. Интерфейс программы содержит следующие элементы (рис. 11):

– главное меню;
– панель инструментов;
– поле ввода функций;
– панель включения отображения функции;
– область вывода графиков функций;
– строка состояния.

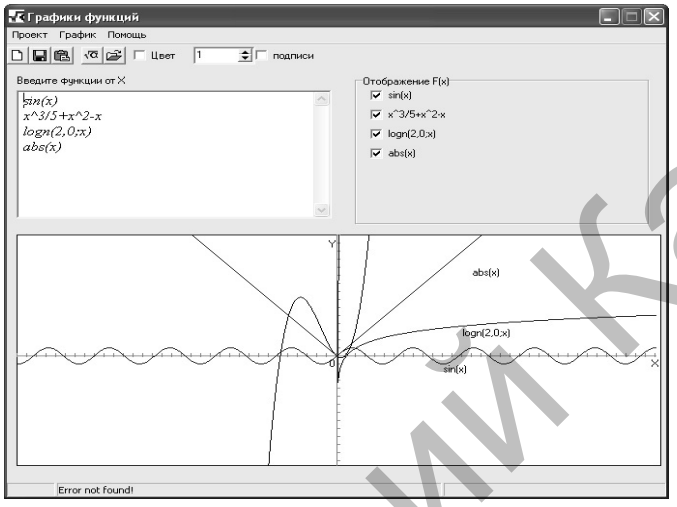


Рис. 11. Интерфейс приложения «Построение графиков функций»

Вариант меню для данной системы представлен на рисунке 12.

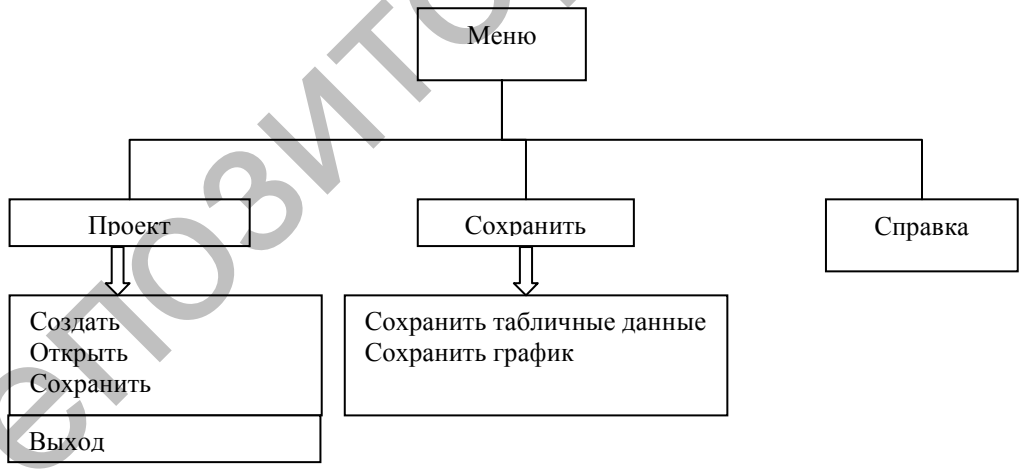


Рис. 12. Меню системы «Построение графиков функций»

Главное меню содержит основные команды работы приложения: Проект, График, Справка. Пункт меню «Проект» предназначен для создания, сохранения, открытия файла, содержащего список функций. Пункт меню «График» используется для сохранения графиков функции в наиболее используемых графических форматах — *bmp, ico, wmf, emf* (рис. 13), а также результатов табулирования функций с заданным шагом (рис. 14). Пункт «Справка» содержит описание приложения.

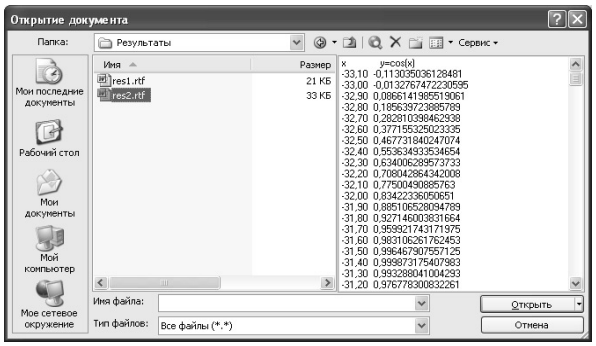


Рис. 13. Результаты табулирования функции

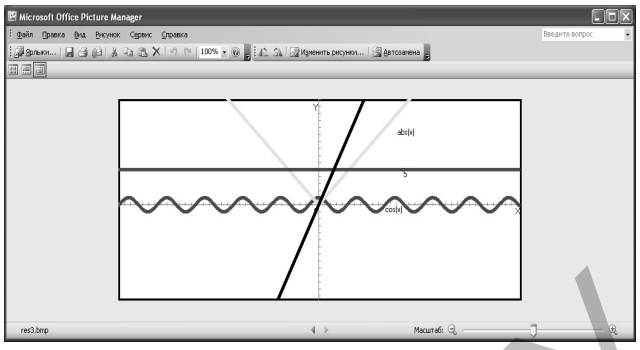


Рис. 14. Просмотр графика функции — файл res2.bmp

Панель инструментов, представленная на рисунке 15, используется для быстрого доступа к основным пунктам главного меню, а также настройки параметров построения.

Рис. 15. Панель инструментов «Построение графиков функций»

При запуске программы Поле ввода функции содержит тестовый пример основных типов функций. При изменении функций результат построения графиков интерактивно отображается в области вывода. При добавлении функции в панели «Отображения» добавляется флажок с соответствующей надписью. Введенные функции автоматически прорисовываются с параметрами, определенными в панели инструментов. На поле ввода функции наложено ограничение — количество функций. В Строке состояния при вводе некорректно заданного выражения выводится сообщение об ошибке (рис. 16).

Рис. 16. Строка состояния при вводе выражения sinx

Программа предусматривает возможность задания цвета и толщины линии вывода графиков функций. Панель «Отображение» позволяет управлять режимом включения/отключения вывода графиков отдельных функций (рис. 17).

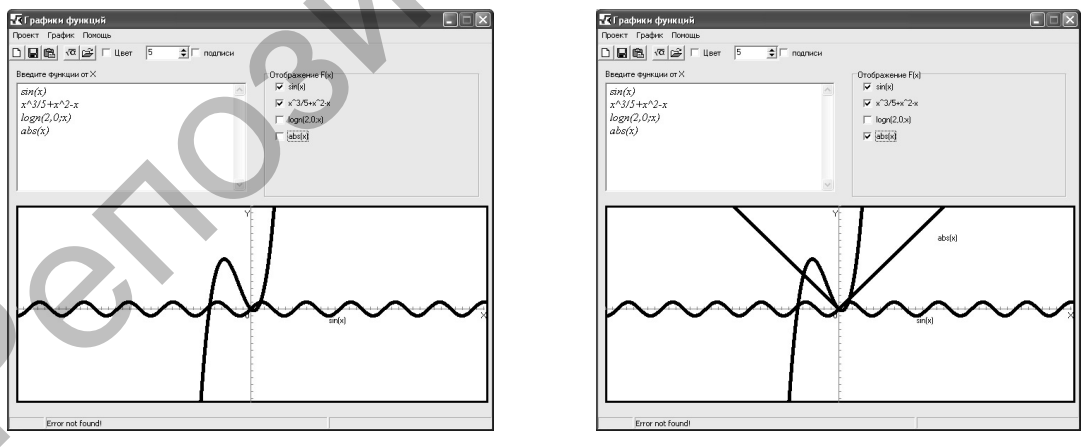


Рис. 17. Изменение параметров отображения

Параметры построения можно изменять в процессе работы с программой.

При закрытии программы пользователю предлагается сохранить полученные результаты (рис. 18).

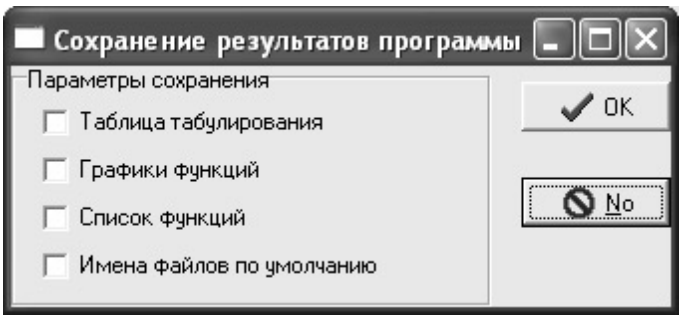


Рис. 18. Запрос на сохранение результатов поиска

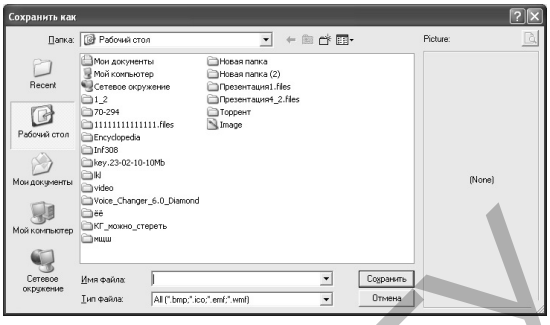


Рис. 19. Форма, реализующая диалог «Сохранить как…»

В системе «Построение графиков функций» реализованы стандартные диалоги открытия и сохранения файла (рис. 19).

При открытии проекта используется фильтр (рис. 20) для блокировки некорректных действий, например, выбор исполняемого файла, со стороны пользователя.

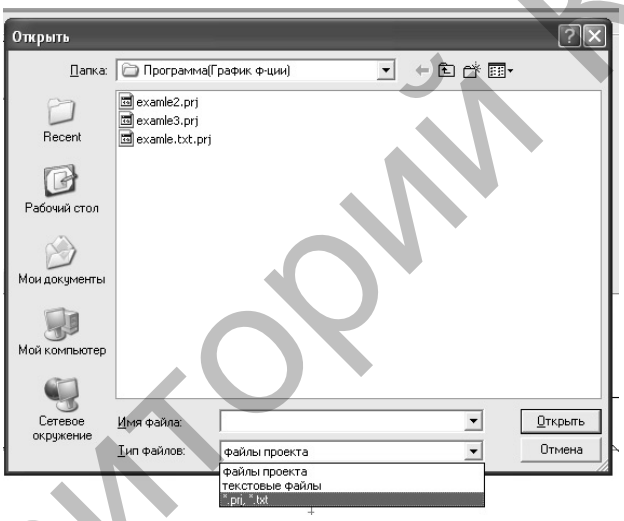


Рис. 20. Использование фильтра при открытии файла

В программе предусматривается несколько процедур блокировки некорректных действий пользователя:

– ограничение на ввод данных: шаг построения не может быть отрицательным числом, количество вводимых функций ограничено, толщина пера варьируется в пределах от 1..5;
– в том случае, если пользователь некорректно ввел функцию, данная функция игнорируется, в Строке состояния выводится сообщение об ошибке;
– основной целью применения синтаксического анализатора в программе является достижение универсальности при построении графиков функций, а также возможность вывода графиков с точками разрыва [6].

Результатом проделанной работы явилось создание программы «Построение графиков функций», являющейся исполняемым приложением, не требующей установки. С программой может работать пользователь, незнакомый со сложными математическими пакетами и языками программирования. Работа с этой программой не требует лицензионного соглашения.

Программа состоит из двух основных модулей: синтаксического анализатора и построения графиков функций. При этом была разработана общая схема синтаксического анализа математического выражения с описанием структур и их взаимодействия. Так как программа разбита на модули, следовательно, легко может быть модифицирована, что дает возможность использования разработанных

модулей в других программах. Например, компонент синтаксического разбора арифметического выражения может использоваться также при построении графиков функции с двумя и более переменными [7].

При работе над проектом были созданы алгоритмы, позволяющие производить синтаксический разбор математического выражения.

К числу основных отличительных особенностей созданного программного продукта можно отнести следующие:

– небольшой объем оперативной и дисковой памяти, требуемой для работы с данной программой;
– дружественный интерфейс программы и наличие справки;
– возможность сохранения результатов в различных форматах;
– поддержка интеллектуального ввода данных;
– универсальность задания функций;
– поддержка вывода функций с точками разрыва;
– дополнительные настройки параметров вывода графиков функций
– автомасштабирование области вывода графиков функции;
– возможность использования модуля синтаксического анализатора в других приложениях.

Представленная программа может быть использована как на уроках информатики, так и на уроках алгебры в старших классах.

References

1. *Ellen M.* Voorhes, Donna Harman. Overview of Sixth Text Retrieval Conference (TREC-6) // National Institute of Standards and Technology Gaithersburg, MD 20899, 1998.
2. *Belousov A.I., Tkachev S.B.* Discrete mathematics: a textbook // Ed. VS Zarubina, AP Kritsenko. — M.: MSTU. Bauman, 2001. — P. 744.
3. *Ermakov A.E.* Part-time parsing of the text in a retrieval system // Computational Linguistics and Intelligent Technologies: Proceedings of the International Workshop Dialogue — 2002.
4. *Ermakov A.E.* Full text search problems and their solution // PC World. — № 5. — 2001. — P. 24–27.
5. *Novikov F.A.* Discrete mathematics for programmers. — St. Petersburg: Piter, 2000. — P. 304.
6. *Kuznetsov O.P., Adelson-Velsky G.M.* Discrete mathematics for the engineer. — M.: Energoatomizdat, 1988. — P. 480.
7. *Gordeev A.V., Molchanov, A.Yu.* System software. — St.: Peter, 2002. — P. 736.