МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Кафедра вычислительной математики и информатики**

**КУРСОВАЯ РАБОТА**

**Динамическое изображение. Свободное падение листа**

Работу выполнила Куницын В. И.

 (подпись, дата)

Факультет Математики и компьютерных наук курс 2 Направление 02.03.01 Математика и компьютерные науки

Научный руководитель

преподаватель К.Ю. Дуриндина

 (подпись, дата)

Нормоконтролер

преподаватель А.А. Цыбенко

 (подпись, дата)

Краснодар 2018

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 4](#_Toc512460463)

[1. История языка Pascal и PascalABC.NET 5](#_Toc512460464)

[1.1. История развития языка Pascal 5](#_Toc512460465)

[1.2. PascalABC.NET и особенности его развития 7](#_Toc512460466)

[1.3. Процедуры и функции модуля GraphABC 9](#_Toc512460467)

[2. Разработка алгоритма 13](#_Toc512460468)

[3. Тестовые примеры 18](#_Toc512460469)

[ЗАКЛЮЧЕНИЕ 24](#_Toc512460470)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 25](#_Toc512460471)

## **ВВЕДЕНИЕ**

Была поставлена задача: сконструировать изображение падения пера.

Цель:составить алгоритм для реализации программного кода, решающего поставленную задачу, и разработать сопроводительную документацию к нему.

Задание включает в себя следующие виды работ:

1) Пошаговая разработка алгоритма и его описание.

2) Описание алгоритма на языке программирования.

3) Отладка программы.

4) Демонстрация возможностей программы.

5) Составление документации.

В качестве языка программирования был выбран язык PascalABC.NET 3.3.

## **История языка Pascal и PascalABC.NET**

## **История развития языка Pascal**

Язык программирования Pascal был создан Никлаусом Виртом, и назван в честь французского философа и математика XVII в. Блеза Паскаля. Сам Никлаус Вирт был профессором информатики в Федеральном техническом университете в Швейцарии и нуждался в языке, с помощью которого можно было обучать студентов навыкам программирования.

Концепция Паскаля была разработана Н. Виртом примерно в 1970 году и Pascal быстро получил широкое распространение благодаря легкости его изучения, наглядности составленных на нем текстов программ. Поскольку Pascal послужил основой для разработки других языков программирования, таких как Ада и Модула-2, и поскольку многие языки содержат аналогичные Pascal структуры, знание Pascal является солидной базой для изучения других языков программирования.

В середине 70-х годов была создана попытка разработать международный стандарт на Pascal. В результате в 1982 году появился стандарт ИСО 7185.

Язык Pascal стараниями Андерса Хейлсберга превратился в мощную профессиональную систему программирования Turbo Pascal. После просуществовавшей сравнительно недолго и не получившей широкое распространения версии 1.0 в середине 1984 года появляется версия 2.0. Распространение которой пошло стремительными темпами. К осени 1985 года появляется версия 3.0, отличающаяся от версии 2.0 следующими особенностями: компилятор и редактор стали работать существенно быстрее, появилась возможность передачи параметров в программу с помощью команды RUN, стал возможным вызов MS-DOS из программы, стала удобной работа с файлами. С начала 1988 года начинает распространяться версия 4.0.

Здесь Турбо Паскаль представлен в совершенно новой форме. Не только становится еще более быстрым компилятором, но и появляется совершенно новое программное окружение. Появилась возможность разбиения программы на части (модули), компилируемые по отдельности. К осени 1988 года появилась версия 5.0 с еще белее развитым программным окружением. Здесь заслуживает внимания прежде всего встроенный отладчик. Турбо Паскаль может использоваться в большинстве существующих для персональных компьютеров операционных систем.

Появление инструментальных средств Borlаnd Pascal with Objects и Delphi для разработки программ в среде Windows лишний раз показала какие по истине неисчерпаемые возможности таит в себе Паскаль. Borland Pascal и используемый в Delphi язык Object Pascal основываются на Turbo Pascal и развивают его идеи.

Delphi - компилятор языка Pascal. Delphi 1 был первым инструментарием разработки Windows приложений, объединившим в себе оптимизирующий компилятор, визуальную среду программирования и мощные возможности работы с базами данных. Годом позже Delphi 2 предложил все то же, но на новом уровне современной 32-битной операционной системы Windows 95 и Windows NT. Кроме того, Delphi 2 предоставил программисту 32-битовый компилятор, создававший более быстрые и эффективные приложения, мощные библиотеки объектов.

Продолжительная работа команды разработчиков Delphi привела к появлению в третьей версии продукта расширенного набора инструментов для создания приложений, возможности использования технологий COM для разработки приложений WWW и многих других современных технологий программирования.

Delphi 4 является очередным шагом в эволюции компиляторов Паскаля с тех времен, когда более 16 лет назад Андерс Хейлсберг создал первый компилятор Turbo Pascal.

## **PascalABC.NET и особенности его развития**

**PascalABC.NET** — язык программирования Паскаль нового поколения, включающий классический Паскаль, большинство возможностей языка [Delphi](https://ru.wikipedia.org/wiki/Delphi_%28%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F%29%22%20%5Co%20%22Delphi%20%28%D1%8F%D0%B7%D1%8B%D0%BA%20%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F%29), а также ряд собственных расширений. Он реализован на платформе [Microsoft.NET](https://ru.wikipedia.org/wiki/.NET_Framework), которая обеспечивает язык **PascalABC.NET** огромным количеством стандартных библиотек и позволяет легко сочетать его с другими .NET-языками: C#, Visual Basic.NET, управляемый C++, Oxygene и др., а также содержит все современные языковые средства: [классы](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B0%D1%81%D1%81_%28%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%29), [перегрузку операций](https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B5%D0%B3%D1%80%D1%83%D0%B7%D0%BA%D0%B0_%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%82%D0%BE%D1%80%D0%BE%D0%B2), [интерфейсы](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%28%D0%BE%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%29), [обработку исключений](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0_%D0%B8%D1%81%D0%BA%D0%BB%D1%8E%D1%87%D0%B5%D0%BD%D0%B8%D0%B9), [обобщенные классы и подпрограммы](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D0%BE%D0%B1%D1%89%D1%91%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5), [сборку мусора](https://ru.wikipedia.org/wiki/%D0%A1%D0%B1%D0%BE%D1%80%D0%BA%D0%B0_%D0%BC%D1%83%D1%81%D0%BE%D1%80%D0%B0_%28%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%29), лямбда-выражения, средства [параллельного программирования](https://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D1%80%D0%B0%D0%BB%D0%BB%D0%B5%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5).

PascalABC.NET имеет ряд преимуществ по сравнению с языком предыдущего поколения:

1. мультипарадигменный язык: на нём можно программировать в [структурном](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5), [объектно-ориентированном](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) и [функциональном](https://ru.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) стилях.
2. простая и мощная интегрированная среда разработки, поддерживающая технологию [IntelliSense](https://ru.wikipedia.org/wiki/IntelliSense%22%20%5Co%20%22IntelliSense), содержащая средства автоформатирования, встроенный отладчик и встроенный дизайнер форм. Кроме того, консольный компилятор PascalABC.NET функционирует на [Linux](https://ru.wikipedia.org/wiki/Linux%22%20%5Co%20%22Linux) и [MacOS](https://ru.wikipedia.org/wiki/MacOS%22%20%5Co%20%22MacOS) под [Mono](https://ru.wikipedia.org/wiki/Mono%22%20%5Co%20%22Mono).
3. В среду **PascalABC.NET** встроен также *дизайнер форм*, позволяющий создавать полноценные оконные приложения в стиле RAD (Rapid Application Development - быстрое создание приложений).
4. В отличие от многих профессиональных сред, среда разработки **PascalABC.NET** не имеет громоздкого интерфейса и не создает множество дополнительных вспомогательных файлов на диске при компиляции программы. Для небольших программ это позволяет соблюсти принцип "Одна программа - один файл на диске".
5. Модуль растровой графики GraphABC и модуль векторных графических объектов ABCObjects могут быть использованы для создания простейших графических, а также интерактивных анимационных приложений, управляемых событиями.

Стоит заметить, что язык **PascalABC.NET** близок к реализации Delphi (Object Pascal). В нем отсутствует ряд [специфических языковых конструкций Delphi](http://pascalabc.net/downloads/pabcnethelp/scr/Common/PABCNETvsDelphi.html), некоторые конструкции изменены. Кроме этого, добавлен ряд возможностей: имеется автоопределение типа при описании, можно описывать переменные внутри блока, имеются операции +=, -=, \*=, /=, методы можно описывать непосредственно в теле класса или записи, можно пользоваться встроенными в стандартные типы методами и свойствами, память под объекты управляется [сборщиком мусора](http://pascalabc.net/downloads/pabcnethelp/scr/LangGuide/MemoryManagement/index_memory.html) и не требует явного освобождения, множества **set** могут быть созданы на основе произвольных типов, введен операторы **foreach**, переменные циклов **for** и **foreach** можно описывать непосредственно в заголовке цикла, имеются обобщенные классы и подпрограммы (generics), лямбда-выражения и др.

В среде **PascalABC.NET** большое внимание уделено связи запущенной программы с оболочкой: консольная программа, запущенная из-под оболочки, осуществляет ввод-вывод в специальное окно, встроенное в оболочку. Можно также запустить несколько программ одновременно - все они будут контролироваться оболочкой.

Язык **PascalABC.NET** позволяет программировать в классическом *процедурном стиле*, в *объектно-ориентированном стиле* и содержит множество элементов для программирования в *функциональном стиле*. Выбор стиля или комбинации этих стилей - дело вкуса программиста, а при использовании в обучении - методический подход преподавателя.

Сочетание богатых и современных языковых средств, возможностей выбора разных траекторий обучения позволяет рекомендовать **PascalABC.NET** с одной стороны, как язык для обучения программированию (от школьников до студентов младших и средних курсов), с другой - как язык для создания проектов и библиотек средней сложности.

## **Процедуры и функции модуля GraphABC**

Модуль GraphABC представляет собой простую графическую библиотеку и предназначен для создания несобытийных графических и анимационных программ в процедурном и частично в объектном стиле. Рисование осуществляется в специальном *графическом окне*, возможность рисования в нескольких окнах отсутствует. Кроме этого, в модуле GraphABC определены простейшие события мыши и клавиатуры, позволяющие создавать элементарные событийные приложения. Основная сфера использования модуля GraphABC - обучение.

Модуль GraphABC основан на графической библиотеке GDI+, но запоминает текущие перо, кисть и шрифт, что позволяет не передавать их в качестве параметров при вызове графических примитивов. К свойствам пера, кисти и шрифта можно получать доступ как в процедурном, так и в объектном стиле. Например, для доступа к цвету текущего пера используется процедура SetPenColor(c) и функция PenColor, а также свойство Pen.Color.

В модуле GraphABC можно управлять самим графическим окном и компонентом GraphABCControl, на котором осуществляется рисование. По умолчанию компонент GraphABCControl занимает всю клиентскую часть графического окна, однако, на графическое окно можно добавить элементы управления, уменьшив область, занимаемую графическим компонентом (например, так сделано в модулях Robot и Drawman).

Для работы с рисунками используется класс Picture, позволяющий рисовать на себе те же графические примитивы, что и на экране.

Режим блокировки рисования на экране (LockDrawing) позволяет осуществлять прорисовку лишь во внеэкранном буфере, после чего с помощью метода Redraw восстанавливать все графическое окно. Данный метод используется для ускорения анимации и создания анимации без мерцания.

В модуле GraphABC определен ряд констант, типов, процедур, функций и классов для рисования в *графическом окне*. Они подразделяются на следующие группы:

|  |  |  |
| --- | --- | --- |
| * Графические примитивы
* Функции для работы с цветом
* Цветовые константы
* Действия с пером: процедуры и функции
* Действия с пером: объект Pen
* Стиль пера
 | * Действия с кистью: объект Brush
* Стили кисти
* Стили штриховки кисти
* Действия со шрифтом: процедуры и функции
* Действия со шрифтом: объект Font
* Стили шрифта
* Действия с рисунками: класс Picture
 | * Действия с графическим окном: процедуры и функции
* Действия с графическим окном: объект Window
* Действия с системой координат: процедуры и функции
* Действия с системой координат: объект Coordinate
* Блокировка рисования и ускорение анимации
* Режимы рисования
* События GraphABC
 |

Так как при написании кода самого курсового проекта используются объекты группы «графические примитивы», то приведём примеры некоторых встроенных подпрограмм данной группы.

**procedure PutPixel(x, y: integer; c: Color)-** Заливает пиксель с координатами (x, y) цветом c.

**procedure Line(x, y, x1, y1: integer) -** Рисует линию от точки с координатами (x, y) до точки с координатами (x1, y1) текущим цветом кисти.

**procedure Line(x, y, x1, y1: integer; c: Color) -** Рисует линию от точки с координатами (x, y) до точки с координатами (x1, y1) цветом c.

**procedure FillCircle(x, y, r: integer) -** Заполняет окружность с координатами центра (x, y) и радиусом r текущим цветом кисти.

**procedure DrawCircle(x, y, r: integer) -** Рисует контур окружности с координатами центра (x, y) и радиусом r текущим цветом кисти.

**procedure** FillEllipse(x, y, x1, y1: integer) - Заполняет эллипс, который ограничен прямоугольником с координатами противоположных вершин (x, y) и (x1, y1), текущим цветом кисти.

**procedure** DrawEllipse(x, y, x1, y1: integer) - Рисует контур эллипса, который ограничен прямоугольником с координатами противоположных вершин (x, y) и (x1, y1), текущим цветом кисти.

**procedure** FillRectangle(x, y, x1, y1: integer) ­- Заполняет прямоугольник, который задан координатами противоположных вершин (x, y) и (x1, y1), текущим цветом кисти.

**procedure** Arc(x, y, r, c1, c2: integer) - рисует дугу окружности с координатами центра (x, y) и радиусом r, ограниченную двумя лучами, угол поворота которых определен c1 и c2 против часовой стрелки.

**procedure ClearWindow() -** Очищает окно белым цветом по умолчанию.

**procedure Clear(c: Color) -** Очищает окно цветом ,заданным пользователем в переменной c.

**procedure SetWindowIsFixedSize(s: boolean) ­-** Зафиксирует размер графического окна если s = true, в противном случае - пользователь сможет изменять размеры окна.

**procedure SetWindowWidth(w: integer) -** Установит ширину графического окна, равную w.

**procedure SetWindowHeight(h: integer) -** Установит высоту графического окна, равную h.

## **Разработка алгоритма**

В процессе исследования было выделено несколько подходов к созданию динамических изображений:

1. Стереть изображение полностью, затем перерисовать его заново, но уже с необходимыми изменениями, которые и будут создавать динамический эффект.
2. Закрасить ненужные элементы цветом фона, перерисовывая их в другом месте, что даст иллюзию их перемещения.

В программе используется первый способ рисования динамического изображения.

Для удобства разобьем код на несколько основных этапов:

1. Интерактивное общение с пользователем.
2. Рисование фигуры «перо».
3. «Падение пера» с имитацией колебательных движений.

Опишем более подробно этапы программы:

1. Интерактивное общение с пользователем.

Запуск программы сопровождается вызовом окна, обращающегося к пользователю с предложением: «Введите количество колебаний (отличное от нуля), возникающих в результате падения пера, которые Вы хотите увидеть:».

1. Рисование фигуры «перо».

Используя процедуру Ellipse рисуем «тело» (контур) пера. В качестве бородок пера используем процедуру построения отрезка ­­– Line.

1. «Падение пера» с имитацией колебательных движений.

Используя процедуру очищения графического окна, стираем изображения и перерисовываем, создавая динамический эффект.

Инициируем графический модуль:

Uses GraphAbc.

Описываем процедуру рисования пера:

**procedure** Pero(x, y: integer; u: real);

**begin**

Ellipse(x - 60, y + 100, x - 100, y + 110);

 Line(x - 70, y + 105, x - 110, y + 105);

 Line(x - 65, y + 102, x - 70, y + 105);

 Line(x - 75, y + 102, x - 80, y + 105);

 Line(x - 85, y + 102, x - 90, y + 105);

 Line(x - 65, y + 107, x - 70, y + 105);

 Line(x - 75, y + 107, x - 80, y + 105);

 Line(x - 85, y + 107, x - 90, y + 105);

**end**;

Описываем процедуру обеспечивающую движение пера в направлении часовой стрелки:

**procedure** Pochas(**var** u, j: real; **var** x, y: integer; s: integer);

**begin**

clearwindow;

 x := round(300 - round(100 \* sin(u)));

 y := s + round(100 + round(100 \* cos(u)));

 Pero(x, y, u);

 sleep(100);

 redraw;

 u += 0.126

**end**;

Описываем процедуру обеспечивающую движение пера в направлении обратном движению часовой стрелки:

**procedure** Protchas(**var** u, j: real; **var** x, y: integer; s: integer);

**begin**

clearwindow;

 x := round(300 - round(100 \* sin(u)));

 y := s + round(100 + round(100 \* cos(u)));

 Pero(x, y, u);

 sleep(100);

 redraw;

**end**;

 Обозначаем переменные

**Var** x, y, k, l, s: **integer**;

u, j, t, z: real;

 Задаем заголовок графического окна:

Window.Title := 'Падающее перо';

write('Введите количество колебаний(отличное от нуля),возникающих в результате падения пера,которые Вы хотите увидеть: ');

Считываем выбор пользователя:

 readln(button).

Задаём размеры клиентской части графического окна и цвет заливки графического окна:

**begin**

**InitWindow**(0, 0, 500, 500, clwhite);

Устанавливаем начальное положение пера по координатным осям:

x := 100;

y := 100;

 Задаем переменную, определяющую начальное значение угла, необходимого для параметризации окружности, как траектории движения пера:

 u := -3 / 2 \* pi;

 Блокируем рисование в графическом окне и зададим параметры, задающие границы колебания:

lockdrawing;

j := 0;

z := 2.86;

t := 9.46;

 Выполняем начальное падение пера с начальной высоты:

**while** u < -pi \* z / 3 - j **do**

 **begin**

j += 0.211;

 clearwindow;

 x := round(300 - round(100 \* sin(u)));

 y := round(100 + round(100 \* cos(u)));

 Pero(x, y, u);

 sleep(30);

 redraw;

 u += -0.126;

 **end**;

 Обеспечиваем дальнейший «спуск» пера:

s := 30; //сдвиг по оси OY

Определяем количество повторений «колебаний» и условия воспроизведения имитации движения в каждом направлении:

**for** k := 1 **to** button **do**

 **begin**

 **if** k **mod** 2 = 0 **then**

 **while** (u > (-pi \* z / 3) - j) **do**

 **begin**

u += -0.126;

 Protchas(u, j, x, y, s);

 u += -0.126;

 **end**;

 **if** (k **mod** 2 <> 0) **and** (k <> button) **then**

 **while** (u < (-pi \* t / 3) + j) **do**

 **begin**

u += 0.126;

 Pochas(u, j, x, y, s);

 u += 0.126;

 **end**;

Обеспечим дальнейший спуск и «угасание» колебаний:

 s += 15;

 z += -0.02;

 t += -0.024;

 **end**;

**end.**

## **Тестовые примеры**

Пример правильного выполнения алгоритма:

 **Шаг первый:** обращение к пользователю



Рисунок 1 – Пример «обращение к пользователю 1»

**Шаг второй**: После выбора пользователя программа рисует изображение.



Рисунок 2 – Динамическое изображение 1

**Шаг третий:** Перо начинает движение по траектории и через некоторое время имеем следующее изображение:



Рисунок 3 – Динамическое изображение 2

**Шаг четвёртый**: С целью демонстрации окончательного результата исполнения программы было введено количество колебаний равное 12, что привело к следующему результату:



Рисунок 4 – Динамическое изображение 3

В процессе отладки работы программы были выявлены некоторые проблемы с алгоритмом, которые были успешно устранены. В качестве примера приведу следующее:

Если в процессе выполнения цикла, обеспечивающего имитацию колебательного движения пера в процессе его падения не предусмотреть изменение параметров $z$ и $t$, отвечающих за определение границ каждого этапа колебательного движения, то получится следующий результат:

**Шаг первый:** Обращение к пользователю



Рисунок 5 – Пример «обращение к пользователю 2»

**Шаг второй**: После выбора пользователем количества колебаний получаем следующее:



Рисунок 6 – Динамическое изображение 4

Сразу после этого программа начинала осуществлять колебания, нарушающие законы физики, что не соответствует заявленным требованиям:

****

Рисунок 7 – Динамическое изображение 5

При сравнении с предыдущим результатом очевидно, что произошло отклонение вправо на гораздо большую величину, чего не могло бы произойти.

Для наглядности было введено значений количество колебаний равное 12, после выполнения которых был получен следующий результат:



Рисунок 8 – Динамическое изображение 6

Данную ошибку алгоритма легко можно исправить, если в тело цикла добавить операторы изменения значений параметров $z$ и $t $соответственно.

## **ЗАКЛЮЧЕНИЕ**

Графические средства PascalABC.NET недостаточно широки, однако, позволяют демонстрировать графические объекты разного уровня сложности. В графическом режиме можно выводить большое количество объектов и изображений, в том числе изменяя их так, чтобы создавался динамический эффект.

В процессе исследования был проведен анализ поставленной задачи и разработан алгоритм решения задачи, описанный на языке программирования PascalABC.NET 3.3.

Созданная программа рисует изображение пера и имитирует его движение в процессе падения.

Возможно и дальнейшее усовершенствование программы:

* + - 1. В дальнейшем возможно добавление в программу блоков программного кода, учитывающих различные параметры пера, позволяющие построить более точную физико-математическую модель движения тела;
			2. На основании данной программы можно построить приложение для моделирования процессов падения различных физических тел;
			3. Возможно создание более широкого пользовательского интерфейса, предусматривающего как выбор параметров исследуемого объекта, так и его типа.

## **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Ерёмин О.Ф. Pascal ABC. Методическое пособие по программированию на языке Pascal ABC. – Моздок, 2009. – 50с.
2. Меженный О.А. Самоучитель Turbo Pascal/ О.А. Меженный. – СПБ, Диалектика, 2008. – 330с.
3. Цветков А.С. Язык программирования Pascal. Система программирования Pascal ABC. Учебное пособие для школьников 7-9 класс– СПБ, 2012. –46с.
4. Поляков К.Ю. Информатика. Углубленный уровень. Учебник для 11 класса в 2 частях. Часть 2. / Ерёмин Е.А.­­­ –Москва: Бином, 2016. – 312с.