Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
Институт космических и информационных технологий

Кафедра вычислительной техники

УТВЕРЖДАЮ
Заведующий кафедрой
_____ О.В. Непомнящий
подпись
« _____» _____ 2019 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.01 – «Информатика и вычислительная техника»

Интерактивное электронное учебное пособие по основам алгоритмизации

Руководитель	_____ подпись, дата	ст. преподаватель	К.В. Пушкарев
Выпускник	_____ подпись, дата		А.С. Груздев
Консультант	_____ подпись, дата	канд. техн. наук, доцент	Л.И. Покидышева
Нормоконтролер	_____ подпись, дата		В.И. Иванов

Красноярск 2019

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Обычно, когда ученик начинает учиться программировать, он использует один из основных текстовых языков программирования. В зависимости от языка и его синтаксиса это может быть, как легко, так и сложно. Многие языки требуют написания строк с запутанным кодом только для отображения текста «Hello world!». Используя блок-схемы, можно сосредоточиться на концепциях программирования, вместо нюансов типичного языка программирования [1].

Целью выпускной квалификационной работы является разработка кроссплатформенного приложения, позволяющего составлять алгоритмы в виде блок-схем и выполнять их, а также включающего в себя тестовый режим.

Программа обладает следующими особенностями:

- возможность автоматического и пошагового выполнения алгоритма с отображением текущего состояния алгоритма и переменных;
- добавление различных блоков в схему алгоритма;
- наличие тестового режима, в котором у пользователя будет спрашиваться содержимое какой-либо переменной или выходные данные по завершении программы.

Для достижения цели в работе решаются следующие задачи:

- анализ задания на выпускную квалификационную работу;
- проектирование;
- реализация приложения.

1 Анализ задания на выпускную квалификационную работу

1.1 Обзор существующих решений

В соответствии с заданием на выпускную квалификационную работу необходимо разработать интерактивное пособие по основам алгоритмизации. Для того, чтобы выполнить требования, в первую очередь, был произведен обзор существующих решений для выявления типичных для таких программ функций и возможностей.

На сегодняшний день существует ряд приложений графического программирования, которые позволяют составлять из блоков схему, которую можно выполнить в программе. Рассмотрим несколько таких приложений. В таблице 1 приведены основные сведения о них.

Таблица 1 – Основные сведения о приложениях

Название	Язык	Лицензия	Выполнение алгоритма	Тестовый режим
Flowgorithm	C#	Freeware	+	-
DRAKON Editor	Tcl	Public domain [2]	+	-
Visual Logic	Не указано	Shareware	+	-
RAPTOR	A# и C#	UML Designer - GPL-3.0, остальное Public domain [3]	+	-
Scratch	React [4]	New BSD License [5]	+	-
Afce	C++	GPL-3.0 [6]	-	-

1.1.1 Flowgorithm

Flowgorithm – это язык программирования для начинающих, основанный на простых блок-схемах. Вы можете запускать свои программы прямо в Flowgorithm [7]. Интерфейс программы изображен на рисунке 1.

Программа работает только на операционных системах семейства Windows с разрядностью x64 и x32, а точнее Windows 10, 8, 7 и более старые версии. Для запуска необходим .NET версии 4.6 для Windows 10 и 8 или версии 3.5 для Windows 7 и более старых версий Windows [8].

Flowgorithm распространяется бесплатно, но с определенными условиями.

Проблемы лицензии Flowgorithm:

- она не свободная, так как содержит запрет на изучение устройства программы (п. 1.4);

- она содержит запрет на коммерческое распространение, воспроизведение и аренду (п. 1.3) [9].

Готовые схемы программа сохраняет в формате «.fprg», в котором находится код формата XML.

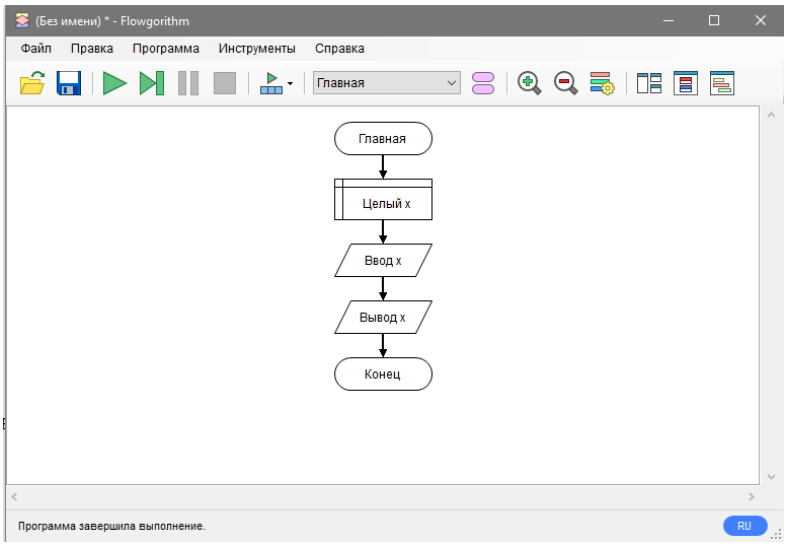

Рисунок 1 – Интерфейс Flowgorithm

1.1.2 DRAKON Editor

DRAKON Editor – это редактор для рисования алгоритмов на языке ДРАКОН, у которого есть два предназначения:

- создавать диаграммы для иллюстрирования и документации;
- программировать с помощью ДРАКОН-схем [2].

Диаграммы в нём составляются из отрезков и блоков.

Для запуска необходима программа Tcl. Пользователи Linux устанавливают следующие пакеты: tcl8.5, tk8.5, tcllib, libsqlite3-tcl, libtk-img. Поддерживаемые операционные системы: Windows, Mac, Linux [2].

Программа находится в общественном достоянии [2], распространяется бесплатно, исходный код находится в открытом доступе.

Готовые схемы сохраняются в формате «.drn», в котором находится база данных SQLite 3.6.

Программисты могут при помощи DRAKON Editor'a генерировать исходный код из ДРАКОН-схем. Поддерживаются такие языки, как Java, Python, Tcl, Javascript, Erlang, Lua, C#, C и C++.

Кроме собственно редактирования диаграмм имеются следующие возможности:

- проверка графического синтаксиса языка ДРАКОН;
- экспорт в PDF;
- экспорт в PNG [2].

Интерфейс DRAKON Editor изображен на рисунке 2.

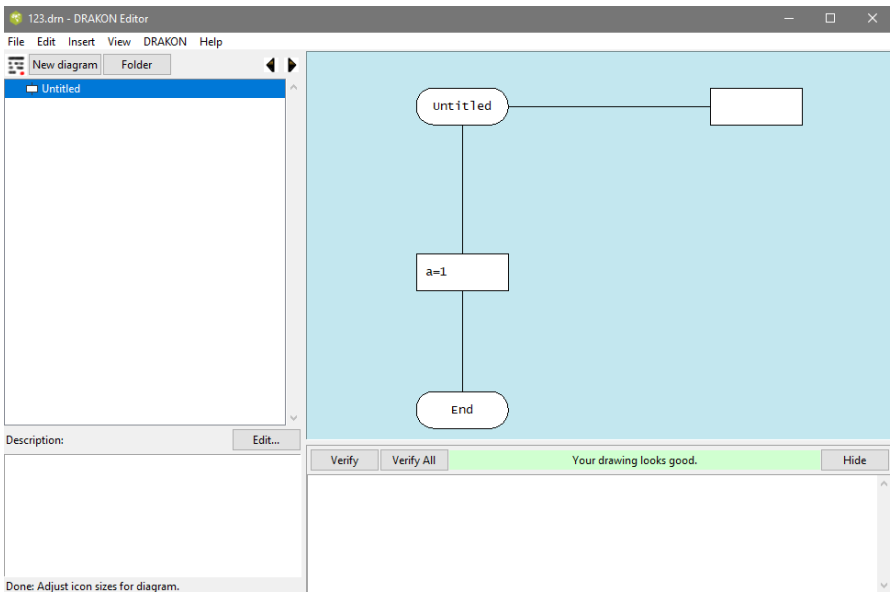

Рисунок 2 – Интерфейс DRAKON Editor

1.1.3 Visual Logic

Visual Logic – это графический инструмент разработки, использующий блок-схемы. Используется для обучения вводным аспектам программирования [10].

Поддерживаемая операционная система – Windows.

Лицензия Shareware – в программе присутствует демо-режим, в котором нельзя сохранять готовые схемы. Исходного кода в открытом доступе нет.

Готовые схемы сохраняются в формате «.vls».

Программа позволяет выполнять алгоритм целиком и пошагово.

На рисунке 3 изображен интерфейс программы.

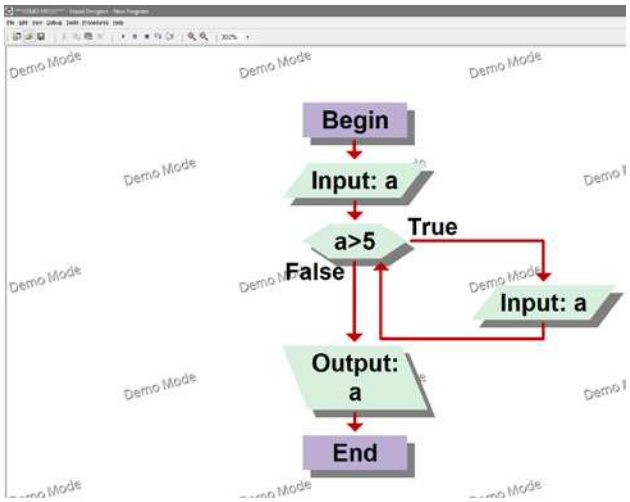

Рисунок 3 – Интерфейс Visual Logic

1.1.4 RAPTOR

RAPTOR – это среда программирования на основе блок-схем, разработанная специально для того, чтобы помочь студентам визуализировать свои алгоритмы и избежать проблем с синтаксисом [3].

Поддерживаемые операционные системы – Windows 10, 8, 7, XP. Для запуска необходим .NET Framework версии 4.5.

RAPTOR основан на NClass у которого лицензия GNU General Public License v3, остальная часть программы находится в общественном достоянии, распространяется бесплатно. Исходный код в открытом доступе [3].

Готовые схемы сохраняются в формате «.rap».

Программа позволяет выполнять алгоритм целиком и пошагово, а также транслировать ее в такие языки программирования, как Ada, C#, C++, Java, VBA.

Интерфейс RAPTOR изображен на рисунке 4.

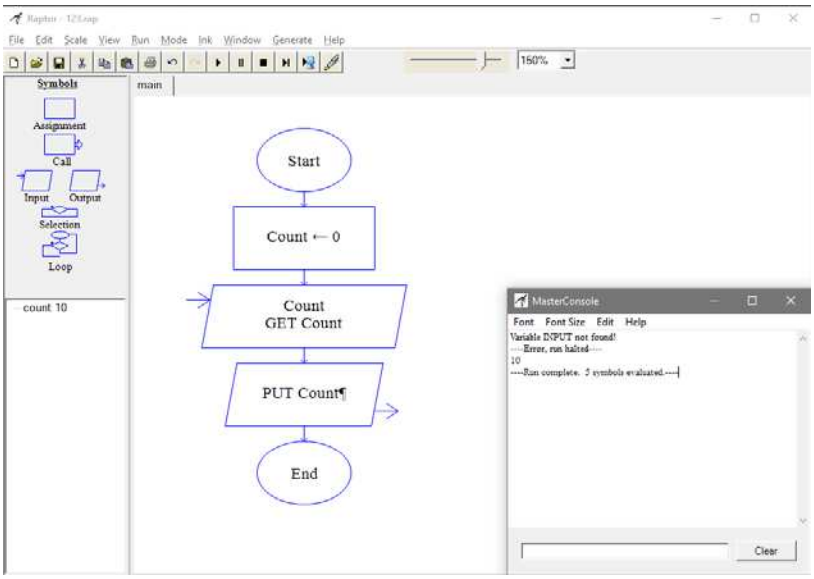

Рисунок 4 – Интерфейс RAPTOR

1.1.5 Scratch

Scratch – визуальная событийно-ориентированная среда программирования для обучения детей от 8 до 16 лет [11].

В программе присутствует область скрипта, где с помощью различных блоков (блоки из списка блоков переносятся в область скриптов) пользователь задает поведение спрайтов в окне визуального отображения алгоритма.

Приложение имеет клиент, написанный на React, который доступен на всех платформах через браузер [4].

Приложение полностью бесплатно. Исходный код находится в открытом доступе под лицензией New BSD License [5].

Готовые схемы сохраняются в форматах «.sb2» и «.sb3». Сохраняемые файлы представляют собой архив, в котором находится набор изображений и файл json [12].

Интерфейс Scratch 3.0 изображен на рисунке 5.

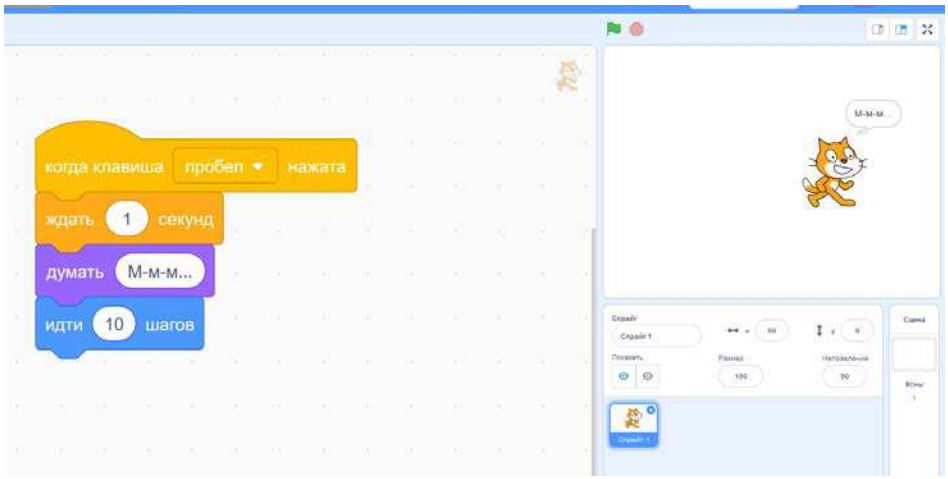

Рисунок 5 – Интерфейс Scratch 3.0

1.1.6 Afce

Afce – это редактор блок-схем с генерацией кода и векторной графики [13]. Проект кроссплатформенный и может быть собран для Microsoft Windows, GNU/Linux и MacOS.

Исходный код написан на языке C++ с использованием Qt.

Лицензия - GNU General Public License v3 [6].

Программа позволяем собирать схемы, но не запускать их на выполнение алгоритма.

Интерфейс изображен на рисунке 6.

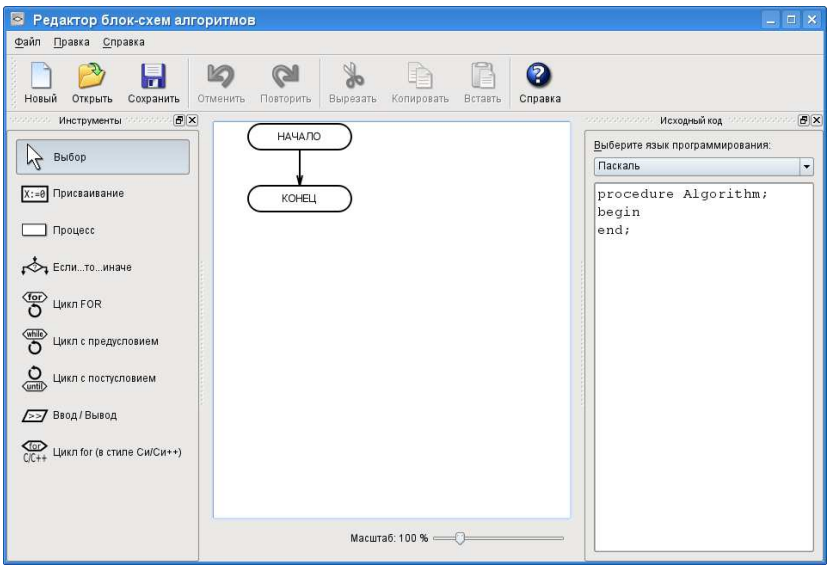

Рисунок 6 – Интерфейс Afce

1.1.7 Итоги обзора

Из обзора существующих решений, представленных в пунктах 1.1.1-1.1.6, можно сделать вывод, что все рассмотренные аналоги предназначены для обучения студентов основам алгоритмизации.

Каждое из рассмотренных решений позволяет создавать схему алгоритма в редакторе и выполнять его (кроме afce), получая определенный результат. Также во всех решениях, кроме Scratch, присутствует пошаговое выполнение алгоритма, которое помогает лучше понять работу алгоритма, поэтому этот элемент очень важен в подобных программах.

Из пяти программ в Scratch лучше всего выглядит визуальная часть алгоритма, но так как блоки в нем не соответствуют никаким стандартам блок-схем, а алгоритмы собираются в упрощенной форме, то это решение больше подходит для детей средней и начальной школы, а не для студентов ВУЗа. Среди остальных решений только в Flowgorithm визуальное отображение алгоритма наиболее близко к ГОСТ 19.701-90.

В RAPTOR и Flowgorithm присутствует окно с переменными алгоритма, что даёт наглядность, способствующую пониманию алгоритма студентами. Также в этих двух программах можно транслировать алгоритм в программный

код. В Visual Logic, RAPTOR и Flowgorithm присутствует консоль, которая предназначена для ввода и вывода значений во время выполнения алгоритма.

Рассмотренные системы направлены только на составление и выполнение алгоритма и не имеют никаких средств, которые можно было бы использовать для создания тестовых заданий с дальнейшей возможностью их проверки.

Было решено в процессе разработки заимствовать часть кода из приложения afce. В случае использования afce с лицензией GPL-3.0, необходимо будет предоставлять другим свободный доступ к программе и ее исходному коду. Таким образом, пользователи получат полный комплекс прав на использование, распространение и модификацию приложения.

1.2 Интерфейс программы

На основе задания на ВКР был сделан макет интерфейса, который должен быть в конечном приложении. На рисунке 7 изображен этот макет.

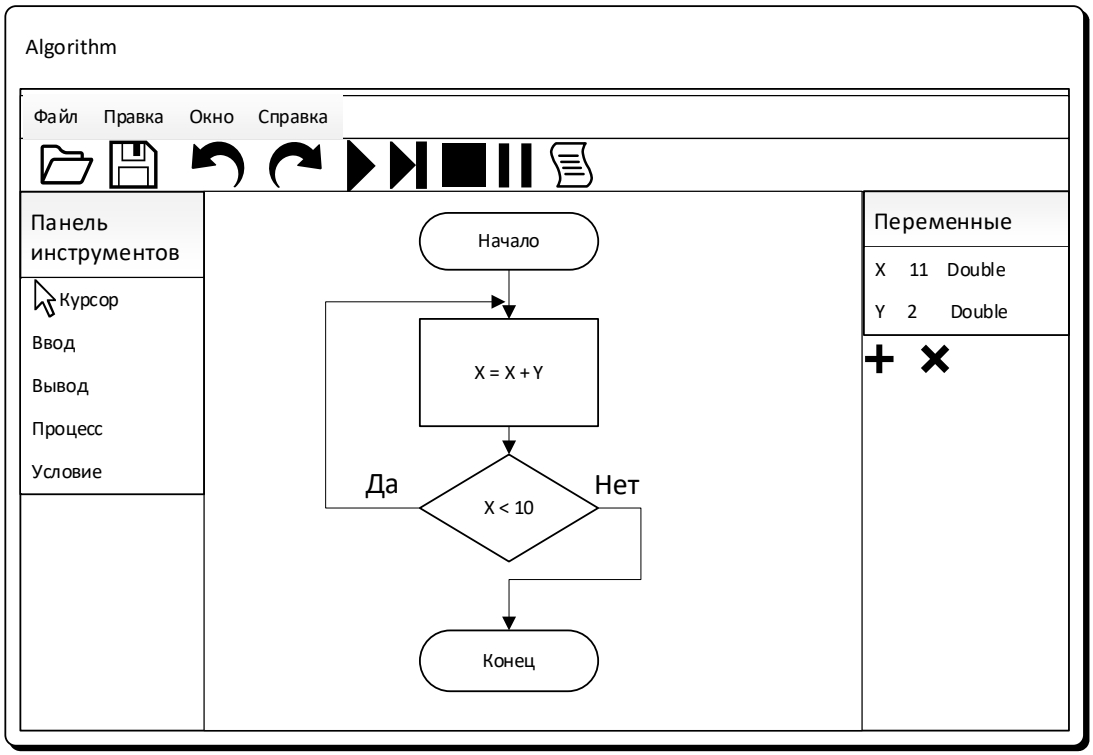

Рисунок 7 – Макет интерфейса

1.3 Диаграмма прецедентов

На рисунке 8 изображена диаграмма прецедентов.

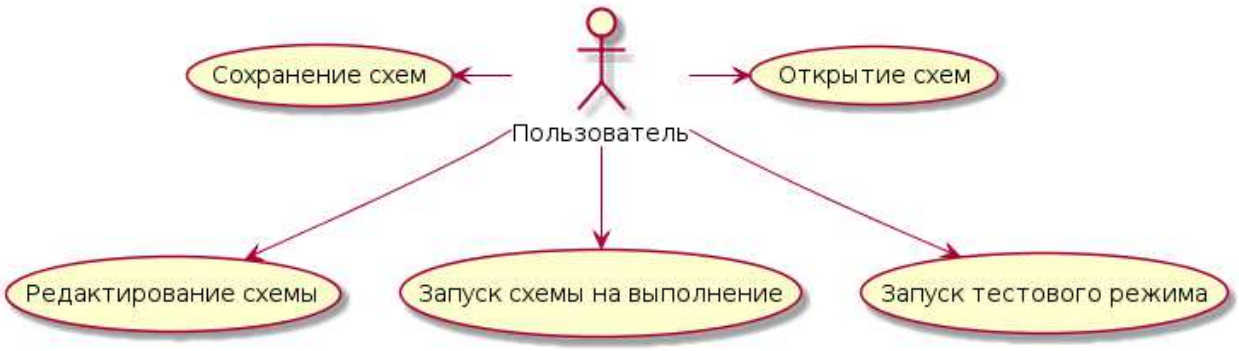

Рисунок 8 – Диаграмма прецедентов

Текстовое описание прецедентов:

Название прецедента: Редактирование схемы.

Цель сценария: добавить блок на схему алгоритма/редактировать блок.

Предусловия: пользователь находится на главном экране, выбирает определенный блок и нажимает на точку в поле с алгоритмом.

Основной сценарий:

A. Создается объект класса, соответствующего типу выбранного блока.

B. Обновляется отображение блоков на схеме.

Постусловия: в алгоритме становится на блок больше.

Условие ввода в действие альтернативных сценариев

Условие 1. Пользователь два раза подряд нажал на блок.

Основной сценарий:

A. Появляется окно, в котором можно изменить атрибуты блока.

B. После нажатия в окне кнопки «ОК», данные сохраняются и обновляется отображение блоков на схеме.

Постусловия: изменяются атрибуты блока.

Название прецедента: Запуск схемы на выполнение.

Предусловия: пользователь нажимает кнопку «Запустить пошагово».

Основной сценарий:

A. В качестве активного блока выбирается первый блок в схеме.

B. При нажатии кнопки «Сделать шаг», выполняется действие, описанное в блоке, и активным становится следующий блок.

Постусловия: алгоритм переходит в состояние выполнения.

Условие ввода в действие альтернативных сценариев

Условие 1. Запустить полностью

Предусловия: пользователь нажимает кнопку «Запустить».

A. Алгоритм выполняется целиком.

Постусловия: алгоритм выполнится.

Название прецедента: Запуск тестового режима.

Предусловия: пользователь нажимает кнопку «Запустить тест».

Основной сценарий:

A. Пользователь выбирает тип вопроса «Значение переменной».

B. Пользователю задается вопрос: «Чему будет равна переменная в конце выполнения алгоритма?».

C. При правильном ответе появится сообщение об этом.

Постусловия: пользователь ответил на вопрос теста.

Условие ввода в действие альтернативных сценариев

Условие 1. Второй тип вопроса

Предусловия: Пользователь выбирает тип вопроса «Вывод в консоль».

A. Пользователю задается вопрос: «Что программа выведет в консоль?».

B. При правильном ответе появится сообщение об этом.

Постусловия: пользователь ответил на вопрос теста.

Условие 2. Случайный тип вопроса

Предусловия: Пользователь выбирает тип вопроса «Случайный».

A. Программа выбирает случайный вопрос из двух других.

Постусловия: выбирается тип вопроса теста.

Название прецедента: Сохранение схем.

Предусловия: пользователь нажимает кнопку «Сохранить».

Основной сценарий:

А. Алгоритм конвертируется в формат xml и сохраняется на диск.

Постусловия: пользователь сохранил схему.

Название прецедента: Открытие схем.

Предусловия: пользователь нажимает кнопку «Открыть».

Основной сценарий:

А. В окне проводника пользователь находит файл, который хочет открыть и дважды нажимает на него.

Постусловия: загрузится алгоритм из файла.

1.4 Выбор инструментов

Приложение должно работать на платформах Windows и Linux и обеспечивать графический интерфейс пользователя.

Так как необходимо написать кроссплатформенное приложение, выбор стоял между следующими языками и фреймворками:

- Java с библиотекой JavaFX;
- C++ с библиотекой Qt;
- Python с библиотекой Qt.

Java – это язык программирования, который следует парадигме объектно-ориентированного программирования. JavaFX является платформой, которая позволяет создавать десктопные приложения с графическим интерфейсом. Одной из особенностей JavaFX является то, что дизайн приложения можно описывать с помощью CSS [14].

C++ – это компилируемый язык программирования со статической типизацией. C++ включает в себя поддержку объектно-ориентированного программирования.

Qt включает в свой состав графический фреймворк, который позволяет эффективно построить графическое отображение алгоритма. Также он

включает в себя библиотеки для работы с XML, форматом в котором планируется хранить сохраненные схемы.

Python – высокоуровневый язык программирования с динамической типизацией. Одним из преимуществ Python является четкий синтаксис, благодаря которому исходный код на Python легко читаем. Для работы с qt в Python есть библиотека PySide2, которая официально поддерживается разработчиком qt [15].

C++ в отличие от Java, который использует JVM, и Python, который использует CPython, является компилируемым языком программирования, что ускоряет работу программы. В итоге, так как C++ является одним наиболее производительным языком из вышеперечисленных, выбор был сделан в пользу Qt и C++.

1.5 Итоги анализа задания

На основе анализа аналогов, сделаны выводы:

1. Не было найдено ни одного приложения, полностью удовлетворяющего требованиям задания. Ни в одном из найденных приложений не было обнаружено тестового режима.

2. В качестве основы для графического отображения алгоритма целесообразно обратиться к проекту afce.

Были сформулированы функциональные требования к разрабатываемому приложению в виде диаграммы прецедентов, и их текстового описания. В качестве инструмента разработки был выбран C++17 с библиотекой Qt.

2 Проектирование и реализация приложения

2.1 Структура приложения

На рисунке 9 представлена диаграмма классов. При объектно-ориентированном проектировании были выделены ключевые интерфейсы, что даёт возможность в дальнейшем расширять систему без изменения существующего программного кода. Повторное использование кода реализуется через отношение наследования.

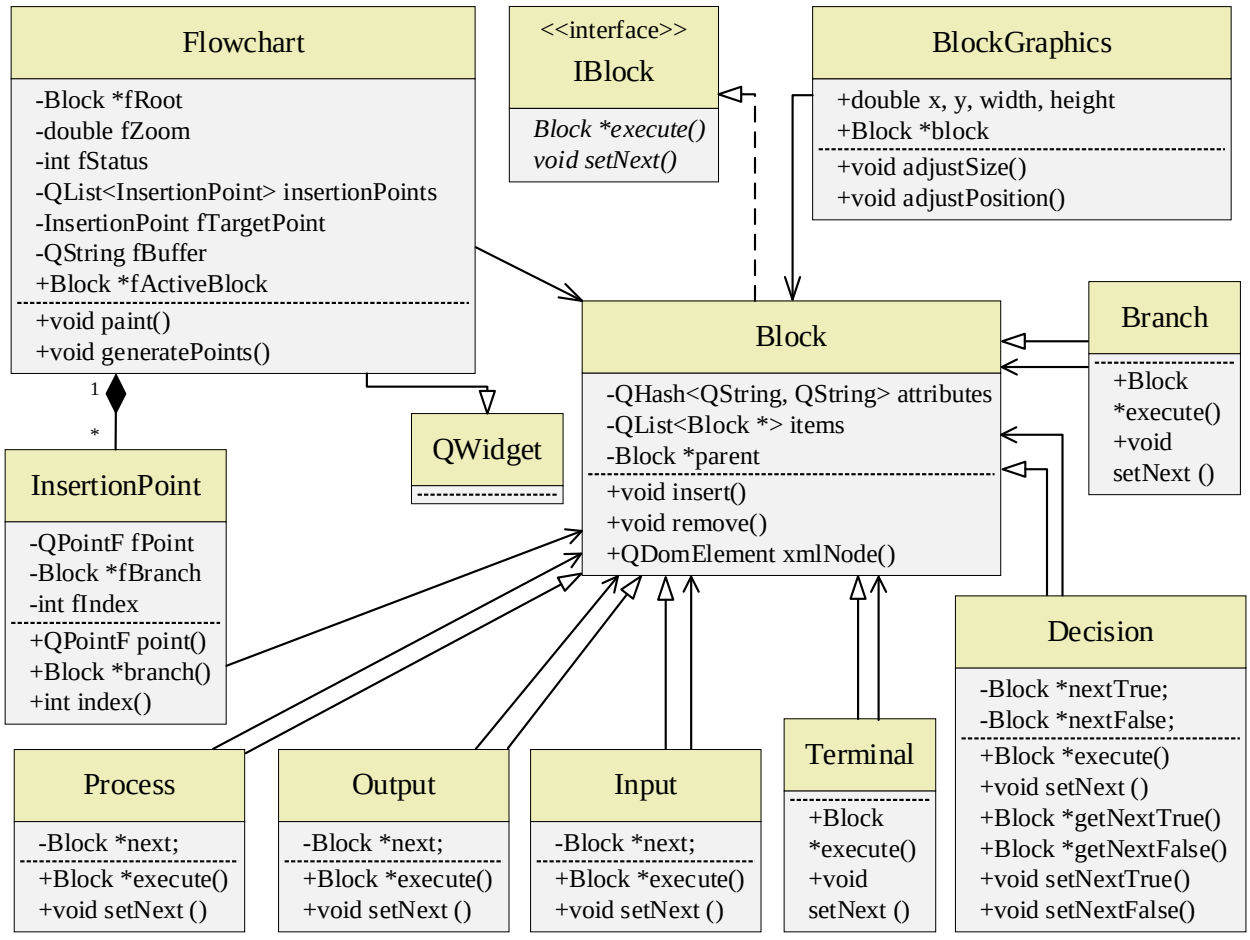

Рисунок 9 – Диаграмма классов

Был реализован интерфейс IBlock с помощью абстрактного класса, т.к. в C++ нет специального синтаксиса для представления интерфейсов. Данный интерфейс используется для реализации блока в алгоритме. В нем

присутствуют два виртуальных метода: execute() для выполнения действий, предписанных блоку и setNext() для установки следующего узла. От IBlock наследует абстрактный класс Block, в котором определены дополнительные атрибуты, необходимые блоку. Список указателей на Block items хранит дочерние блоки, которых может быть несколько, а parent указатель на один родительский блок. Например, блок класса Decision будет иметь два дочерних блока Branch.

В поле attributes типа QHash хранятся атрибуты вида <ключ:значение>. Например, в классе Output там будет храниться имя выводимой переменной. Класс содержит методы для удаления добавления дочерних элементов, а также методы для получения всех приватных полей.

В классе BlockGraphics хранятся данные, необходимые для графического отображения блока, которые включают в себя координаты x и y, ширину и высоту блока.

От класса Block наследуют 6 классов: Decision, Process, Input, Output, Branch, Terminal. В классе Decision присутствует 2 указателя на следующие узлы. В классах Process, Input и Output указатель один, и так же как и в классе Decision присутствует метод execute который выполняет определенную последовательность действий. Последние два класса не имеют указателя на следующий блок и не имеют реализации метода execute, т.к. в этом нет необходимости.

Класс InsertionPoint используется для точек между блоками, куда можно вставить новые блоки.

Класс Variable содержит в себе 3 поля: тип, название и значение. Список объектов класса Variable, который изменяется во время выполнения программы, является глобальной переменной.

Flowchart является наследником класса QWidget и используется для отображения всех блоков, а также он хранит в себе ссылку на корневой блок, класса Terminal. Помимо этого, он хранит в себе данные о текущем активном блоке.

2.2 Графический интерфейс

Для реализации графической части был использован фреймворк Qt. На рисунке 10 изображена структура классов, которые используются для графической части приложения.

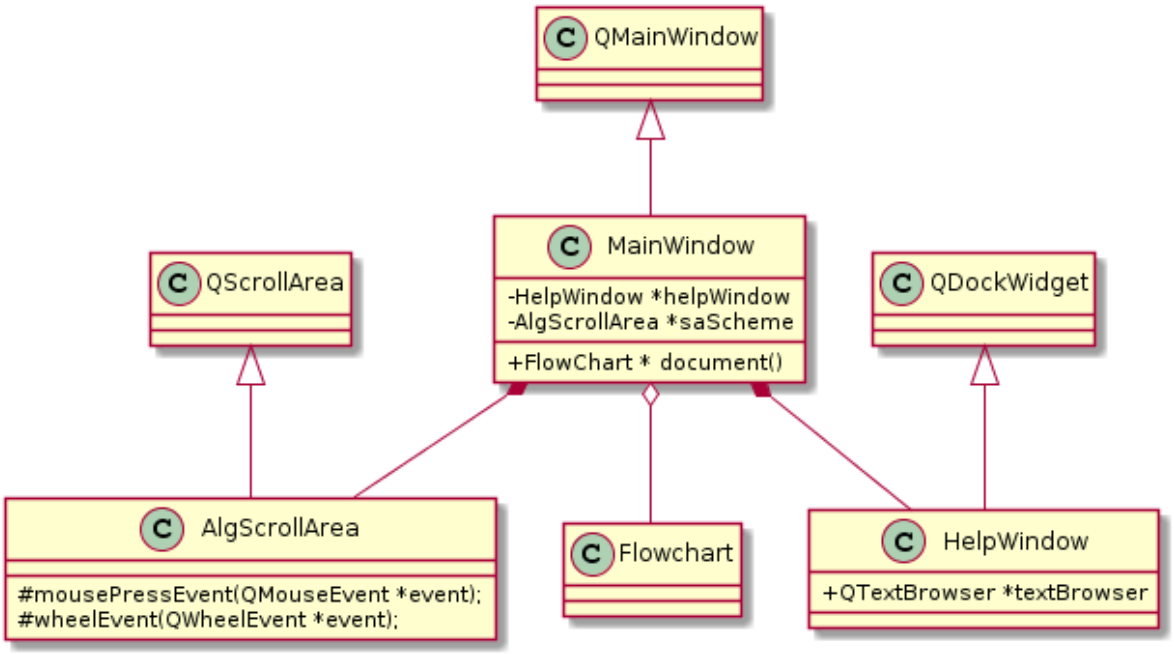

Рисунок 10 – Диаграмма графических классов

Виджет MainWindow содержит в себе все графические элементы программы. В частности, это объект класса AlgScrollArea, который представляет собой поле, на котором рисуется виджет Flowchart. Также MainWindow содержит в себе окно со списком блоков и списком переменных, а также окно помощи. Список блоков, список переменных и окно помощи помещены в объекты класса QDockWidget, который позволяет удобно располагать элементы в главном окне.

Для разработки интерфейса использовались стандартные кнопки QPushButton. Для выравнивая компонентов и масштабирования использованы классы QGridLayout, QVBoxLayout, QHBoxLayout, позволяющие удобно располагать виджеты.

Чтобы пользователь смог разобраться в функционале приложения и в полной мере его использовать, в приложение встроена справка (класс HelpWindow). Для ее отображения применен стандартный класс QTextBrowser, позволяющий обрабатывать некоторые теги HTML.

На рисунке 11 изображен интерфейс главного экрана приложения. На рисунках 12 и 13 показаны примеры диалоговых окон.

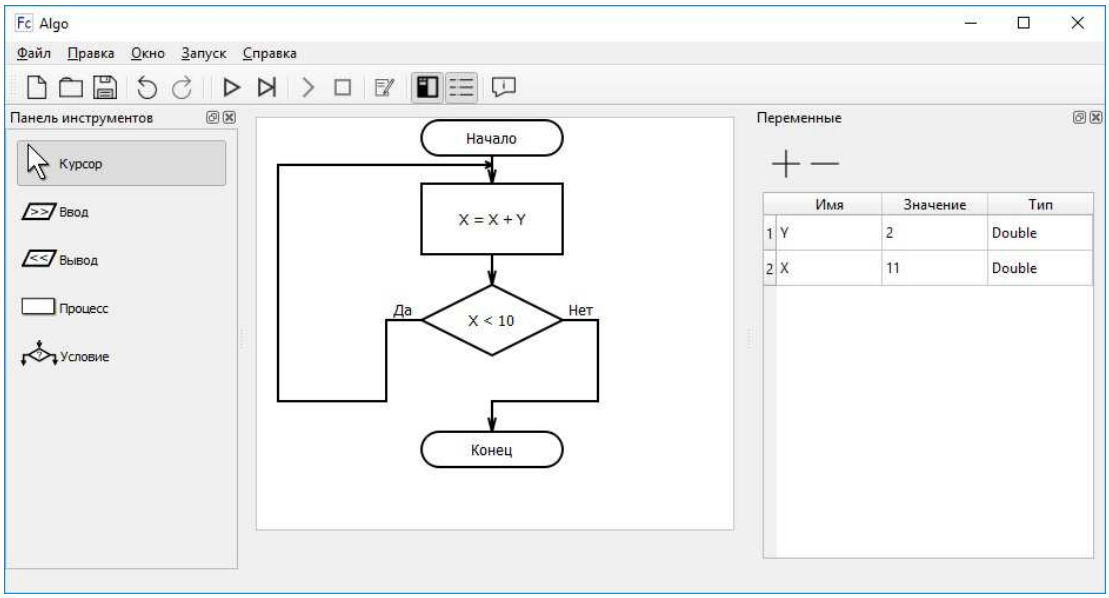

Рисунок 11 – Главный экран

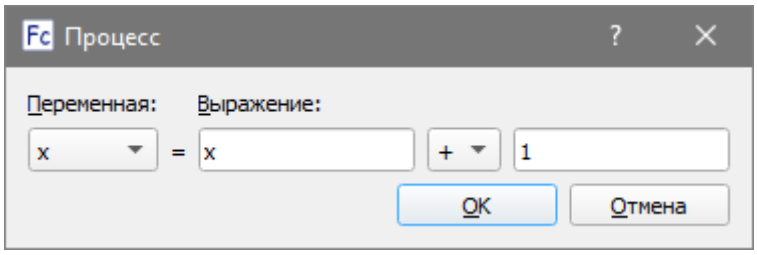

Рисунок 12 – Изменение содержимого блока

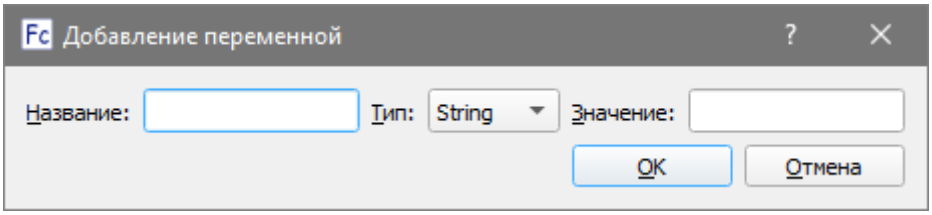

Рисунок 13 – Добавление переменной

2.3 Сохранение и загрузка схем алгоритмов

В приложении для сохранения и загрузки готовых схем алгоритмов используется формат XML. Для работы с форматом XML в Qt есть библиотека QDomDocument, которая позволяет легко собирать данные в формат XML. Пример содержимого файла расположен в листинге 1.

Листинг 1 – Содержимое файла с алгоритмом

```
<?xml version="1.0" encoding="utf-8"?>
<algorithm>
  <terminal id="0">
    <branch id="1">
      <if id="2" cond="x &lt; 10" true="2">
        <branch id="3">
          <process variable="x" id="5" first="x" second="1" sign="+"/>
        </branch>
        <branch id="4"/>
      </if>
    </branch>
  </terminal>
  <variables>
    <variable value="0" name="x" varType="Double"/>
  </variables>
</algorithm>
```

При переносе данных в формат XML, сначала берётся корневой блок алгоритма и создается элемент XML с типом, соответствующем типу блока. Далее те же действия проделываются со всеми дочерними элементами.

В таблице 2 показано соответствие типов элементов XML классам приложения.

Таблица 2 – Соответствие классов и элементов XML

Класс	Название элемента
Terminal	terminal
Branch	branch
Process	process
Decision	if
Input	io
Output	ou

У элемента terminal в качестве дочернего выступает только элемент branch. Branch может иметь любые дочерние элементы, кроме terminal. Также 2 дочерних элемента branch имеет блок if. Остальные элементы не могут иметь дочерних элементов.

Элемент terminal с дочерним элементом branch являются обязательными в любой схеме.

Все атрибуты блока, а также его id, записываются в атрибуты элемента.

У блоков Terminal и Branch отсутствуют дополнительные атрибуты.

Все атрибуты предоставлены строковыми значениями.

Для блока типа Process, атрибуты записываются следующим образом:

- variable – название изменяемой переменной;
- first – первое слагаемое в выражении;
- sign – знак в выражении;
- second – второе слагаемое в выражении.

У блока типа Decision несколько другие атрибуты:

- cond – выражения условия;
- id следующих блоков слева и справа в атрибуты true и false соответственно. Например, в листинге 1 у блока Decision левая ветвь указывает на этот же блок, таким образом образуя цикл.

У блоков класса Input и Output в атрибут var запишется имя переменной.

Получившееся дерево XML присоединяется к элементу algorithm, как дочерний элемент.

После этого записываются все переменные. Для записи переменных используется элемент variables, к которому, в качестве дочерних элементов, присоединяется каждая переменная как элемент с типом variable. Атрибуты следующие:

- value используется для значения переменной;
- name для имени;
- varType для типа.

2.4 Тестовый режим

В программе был реализован тестовый режим, при включении которого пользователь может выбрать тип вопроса, как изображено на рисунке 14. Реализовано три типа вопросов:

1. Чему будет равна переменная в конце выполнения программы? При выборе этого пункта выберется случайная переменная, и после ввода пользователя, алгоритм выполнится и значение переменной сравнится со значением, которое ввел пользователь.

2. Что выведет программа в консоль? При этом типе вопроса алгоритм выполнится, и вывод в консоль всех блоков вывода сравнится со значением написанным пользователем.

3. Случайный вопрос, при выборе которого вопрос случайно выберется из списка вопросов.

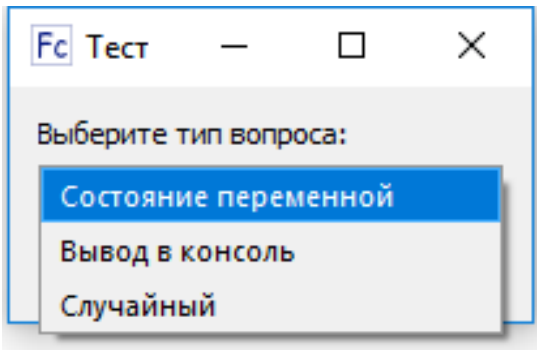

Рисунок 14 – Типы вопросов

После выбора типа вопроса пользователь переходит в окно с вопросом, где расположено поле ввода для ответа. Пример окна с вопросом изображен на рисунке 15.

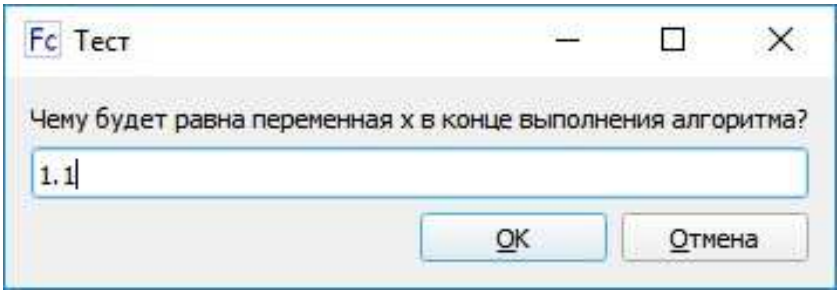

Рисунок 15 – Вопрос теста

После того, как пользователь ввел ответ и нажал кнопку «OK», алгоритм выполнится и появится диалоговое окно, в котором будет написано правильно ли ответил пользователь (рисунок 16).

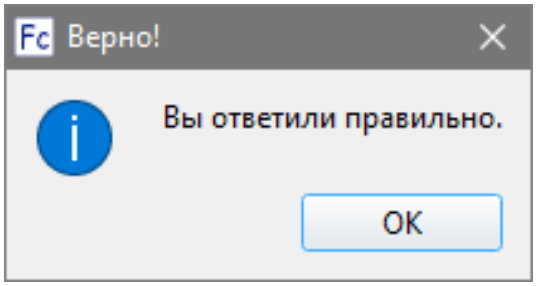

Рисунок 16 – Правильный ответ

3 Инструкции

3.1 Инструкция пользователя

Приложение предназначено для обучения основам алгоритмизации с помощью составления блок-схем алгоритмов и последующего их выполнения.

При открытии программы появится главное окно программы. На рисунке 17 показан интерфейс программы с подписанными основными элементами.

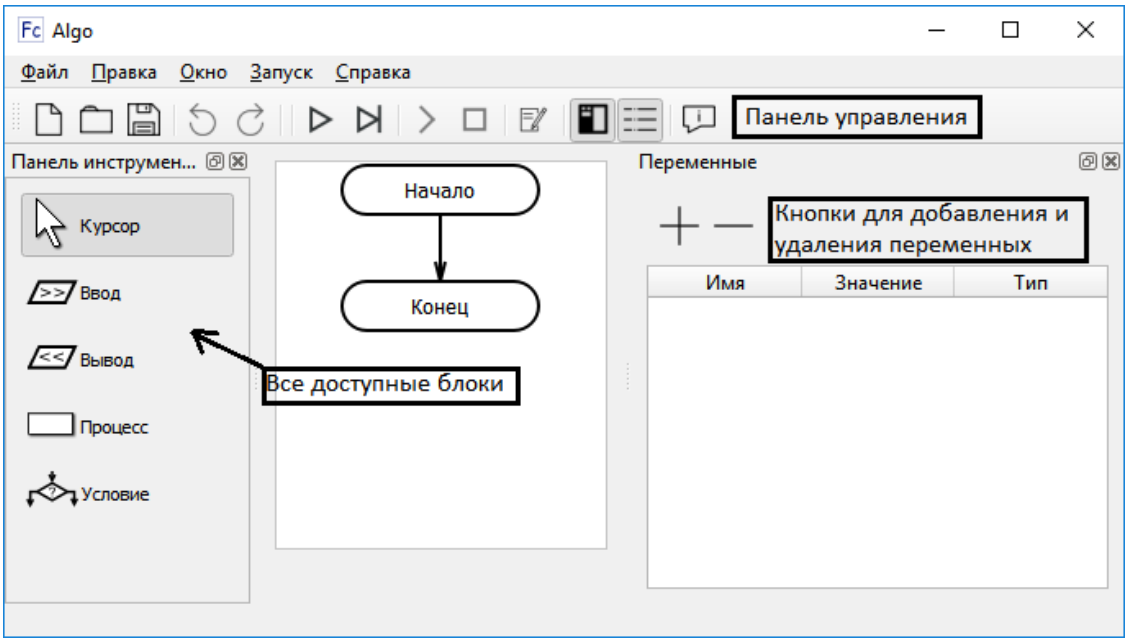

Рисунок 17 – Интерфейс с пояснениями

Слева расположены четыре типа блоков: Процесс, Условие, Ввод и Вывод. При нажатии на один из них на схеме алгоритма в центре экрана появятся точки, в которые можно поставить этот блок (рисунок 18).

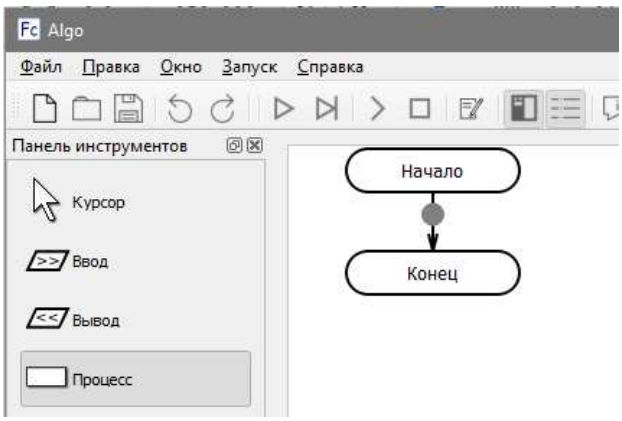

Рисунок 18 – Добавление блока

При двойном нажатии на блок (после размещения его в алгоритме) откроется окно в котором можно изменить свойства этого блока. В блоке условия, также присутствуют кнопки для изменения направления левой и правой ветвей. Пример изменения содержимого блока есть на рисунке 19.

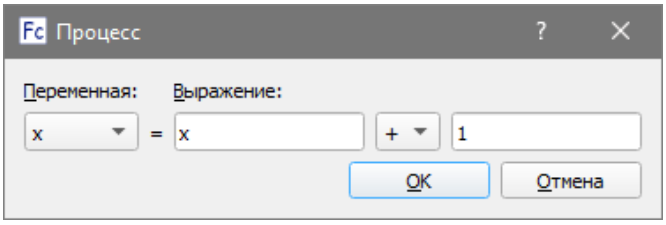

Рисунок 19 – Изменение содержимого блока типа Процесс

Для работы любого из блоков, необходимо наличие хотя бы одной переменной. Для этого в главном окне, слева находится список переменных, а также клавиши для их удаления и добавления. Можно добавить переменные двух типов: String для строк и Double для чисел с плавающей запятой. В названии переменных допустимы буквы русского и английского алфавита, цифры и знак подчеркивания, остальные символы недопустимы. На рисунке 20 показан пример окна добавления переменной.

Рисунок 20 – Добавление переменной типа String

Для запуска алгоритма на выполнение сверху на панели управления присутствуют пункты «Запуск» и «Запуск пошагово». При нажатии кнопки «Запуск пошагово» станут доступны кнопки «Сделать шаг» и «Остановить». На рисунке 21 показан алгоритм в процессе выполнения.

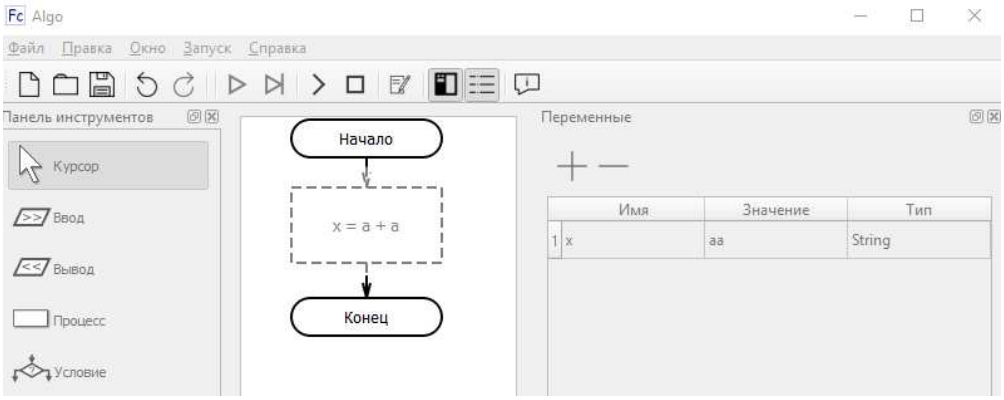

Рисунок 21 – Процесс выполнения

При использовании блоков ввода или вывода во время выполнения появится окно консоли, как на рисунке 22. При выводе необходимо нажать клавишу Enter, чтобы закрыть консоль, и программа продолжила свою работу. При вводе нужно ввести значение для переменной и также нажать клавишу Enter.

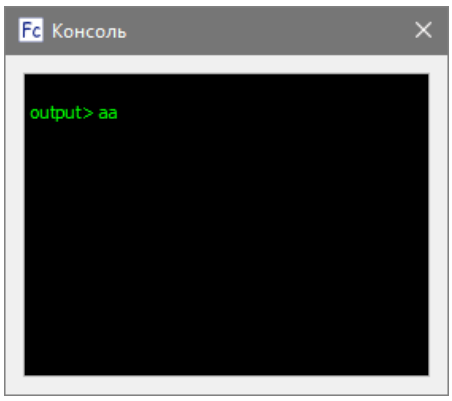

Рисунок 22 – Вывод в консоль

Для запуска тестового режима на панели управления присутствует кнопка «Тест». При нажатии кнопки появится окно, где можно выбрать определенный тип вопросов (рисунок 23).

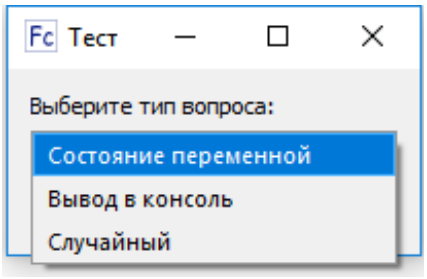

Рисунок 23 – Окно с типами вопросов

После этого появится окно с вопросом и полем ввода, куда нужно ввести ответ (рисунок 24). При правильном ответе на вопрос появится окно с сообщением об этом.

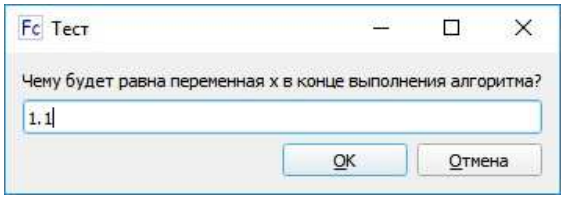

Рисунок 24 – Окно с вопросом теста

Для того, чтобы сохранить готовую схему на панели управления есть кнопка «Сохранить». Для того, чтобы открыть готовую схему есть кнопка «Открыть».

3.2 Инструкция разработчика

Для компиляции исходного кода необходима библиотека Qt версии 5.12 или новее. В качестве IDE лучше использовать Qt Creator. Загрузить Qt вместе с Qt Creator можно с официального сайта Qt [16]. В качестве компилятора лучше выбрать GCC версии 7.3.0 или более позднюю версию. После запуска Qt Creator достаточно нажать кнопку «Запустить» и программа должна запуститься.

Список файлов и их назначение:

-flowchart.h и flowchart.cpp – описание и реализация таких классов как Flowchart, InsertionPoint, Block и все наследующие от него классы;

- flowchartstyle.h и flowchartstyle.cpp – настройка отображения блоков;

- console.h и console.cpp – реализация окна консоли;

- helpwindow.h и helpwindow.cpp – реализация окна помощи;

- mainwindow.h и mainwindow.cpp – реализация класса MainWindow, а также тестов;

- globvars.h – класс Variable;

- diagramscene.pro – информация для сборки проекта с помощью qmake.

В файле flowchart.h находится структурное описание алгоритма, где можно добавить/изменить определенные поля у классов, реализующих блоки, или добавить свои классы для реализации другого вида блоков. Например, чтобы добавить новый вид блоков, нужно создать класс, который будет наследовать от Block, а также реализовывать функции Execute и setNext() интерфейса IBlock.

Функция для графического отображения блоков, под названием paint, расположена в классе Flowchart, там же расположены и функции для сохранения в xml.

Все элементы меню, кнопок и других графических элементов описаны в файлах mainwindow.h и mainwindow.cpp. Например, для изменения вопросов в тестовом режиме вам понадобится функция slotRunTest().

ЗАКЛЮЧЕНИЕ

В результате проделанной работы было спроектировано и реализовано интерактивное электронное учебное пособие по основам алгоритмизации.

Был разработан графический интерфейс программы, структура приложения. Реализованы четыре типа блоков: условие, процесс, ввод и вывод. Было сделано пошаговое и полное выполнение алгоритма, а также сохранение и загрузка из XML. Был сделан тестовый режим, который содержит в себе два типа вопросов, а также вопрос случайного типа.

Возможные направления для дальнейшей доработки:

- возможность генерации файлов с готовыми алгоритмами;
- более разнообразные тесты;
- больше видов блоков;
- реализация функций в алгоритмах.

Последняя версия приложения размещена в git-репозитории [17].

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Flowgorithm - Flowchart Programming Language [Электронный ресурс]. – Режим доступа: http://www.flowgorithm.org/ (дата обращения: 27.04.2019).
2. DRAKON Editor [Визуальный язык ДРАКОН] [Электронный ресурс]. – Режим доступа: https://drakon.su/drakon_editor (дата обращения: 27.04.2019).
3. RAPTOR - Flowchart Interpreter [Электронный ресурс]. – Режим доступа: https://raptor.martincarlisle.com/ (дата обращения: 27.04.2019).
4. LLK/scratch-www: Standalone web client for Scratch [Электронный ресурс]. – Режим доступа: https://github.com/LLK/scratch-www (дата обращения: 27.04.2019).
5. scratch-www/LICENSE at develop · LLK/scratch-www [Электронный ресурс]. – Режим доступа: https://github.com/LLK/scratch-www/blob/develop/LICENSE (дата обращения: 27.04.2019).
6. afce/LICENSE at Master · viktor-zin/afce · GitHub [Электронный ресурс]. – Режим доступа: https://github.com/viktor-zin/afce/blob/master/LICENSE (дата обращения: 27.04.2019).
7. Flowgorithm - About [Электронный ресурс]. – Режим доступа: http://www.flowgorithm.org/about/index.htm (дата обращения: 27.04.2019).
8. Flowgorithm - Download [Электронный ресурс]. – Режим доступа: http://www.flowgorithm.org/download/index.htm (дата обращения: 27.04.2019).
9. Flowgorithm - EULA.pdf [Электронный ресурс]. – Режим доступа: http://flowgorithm.org/about/Flowgorithm%20-%20EULA.pdf (дата обращения: 27.04.2019).
10. Visual Logic [Электронный ресурс]. – Режим доступа: http://www.visuallogic.org/FAQ.html (дата обращения: 27.04.2019).
11. Scratch - О проекте [Электронный ресурс]. – Режим доступа: https://scratch.mit.edu/about (дата обращения: 27.04.2019).

12. Scratch File Format - Scratch Wiki [Электронный ресурс]. – Режим доступа: https://en.scratch-wiki.info/wiki/Scratch_File_Format (дата обращения: 27.04.2019).

13. GitHub - viktor-zin/afce : Flowchart editor [Электронный ресурс]. – Режим доступа: https://github.com/viktor-zin/afce (дата обращения: 27.04.2019).

14. javafx.css (JavaFX 12) [Электронный ресурс]. – Режим доступа: https://openjfx.io/javadoc/12/javafx.graphics/javafx/css/package-summary.html (дата обращения: 05.06.2019).

15. Qt For Python – Qt For Python [Электронный ресурс]. – Режим доступа: https://doc.qt.io/qtforpython/ (дата обращения: 05.06.2019).

16. Download Qt: Choose commercial or open-source [Электронный ресурс]. – Режим доступа: https://www.qt.io/download (дата обращения: 29.04.2019).

17. Gruzdevas / build – Bitbucket [Электронный ресурс]. – Режим доступа: https://bitbucket.org/gruzdevas/build/src/master/ (дата обращения: 27.04.2019).

Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
Институт космических и информационных технологий
Кафедра вычислительной техники

УТВЕРЖДАЮ
Заведующий кафедрой
_____ О.В. Непомнящий
подпись
« 27 » 06 2019 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.01 – «Информатика и вычислительная техника»

Интерактивное электронное учебное пособие по основам алгоритмизации

Руководитель	24.06.19 подпись, дата	ст. преподаватель	К.В. Пушкарев
Выпускник	24.06.19 подпись, дата		А.С. Груздев
Консультант	26.06.19 подпись, дата	канд. техн. наук, доцент	Л.И. Покидышева
Нормоконтролер	26.06.19 подпись, дата		В.И. Иванов

Красноярск 2019