|  |  |
| --- | --- |
|  | |
| МИНОБРНАУКИ РОССИИ | |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | |
| Институт информационных технологий | |
| Кафедра корпоративных информационных систем  **КУРСОВАЯ РАБОТА**  по дисциплине  Структура и алгоритмы обработки данных  **Тема курсовой работы**: «Создание декларативного языка для описания GUI с автоматической генерацией кода для GUI-библиотек» | |
| Студент группы ИКБО-07-18 | Басыров Сергей Амирович |
|  |  |
|  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студента) |
|  |  |
| Руководитель курсовой работы | Советов Пётр Николаевич |
|  |  |
|  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись руководителя) |
|  |  |
| Работа представлена к защите | «20» декабря 2019 г |
|  |  |
| Допущен к защите | «23» декабря 2019 г. |

# содержание

# ВВедение

Предметной областью данного курсового проекта является графические интерфейса пользователя с использованием GUI-библиотеки в рамках какого-либо высокоуровневого языка программирования.

Основные задачи, возникающие при создании GUI как правило, являются: изучение GUI-библиотеки и реализация графического интерфейса. Первая задача как правило реализуема в относительно короткие сроки, однако, вторая задача может потребовать значительных временных затрат, ввиду специфики языка программирования. Если такое сравнение уместно, то, например, реализация сложного графического интерфейса на базе Qt в С++ будет быстрее и не трудоёмкой задачей, нежели на базе Tkinter в Python.

Исходя из темы курсовой работы поставим вопрос, возможно ли создать декларативный язык для описания графического интерфейса, который одновременной решал бы и первую задачу и вторую, за короткие сроки? Чтобы время на его изучение было минимальным и при этом реализация графического интерфейса не уступала бы библиотечным аналогам.

Одним из вариантов решения является создание языка на базе существующей GUI-библиотеки. Под этим подразумевается создание предметно-ориентированного языка декларативного стиля решающий задачу source-to-source генерации. Иначе говоря, мы описываем графический интерфейс на понятном для разработчика языке, а получаем сгенерированный код под GUI-библиотеку. Для решения такой задачи необходимо:

1. проектирование языка;
2. спроектировать и разработать транслятор;
3. спроектировать и разработать генератор кода;
4. спроектировать и разработать консольное приложение, которое позволит получать сгенерированный код.

Для реализации транслятора, генератора и приложения, в рамках курсового проекта был выбран язык Python, по требованию технического задания.

В качестве GUI-библиотеки, для которой будет генерироваться код, был выбран Tkinter, так как данная библиотека является «визитной карточкой» Python для создания графических приложений и входит в стандартную библиотеку языка.

Создаваемый декларативный язык для описания графического интерфейса будет назван Lui. Название в свою очередь является сокращением предложения – «Language of user interface».

В итоге целью курсового проекта является автоматизация создания графического интерфейса с помощью декларативного языка. А непосредственно актуальность заключается в том, что разработчик, не знающий GUI-библиотеку Tkinter сможет для своего приложения без графического интерфейса реализовать графическую часть, в короткие сроки.

# основная часть

## 0 Описание декларативного подхода и существующих решений

Декларативный язык программирования – это язык, с помощью которого решение задачи происходит посредством описания решаемой задачи, а не указания конкретной последовательности действий, приводящей к выполнению задачи.

Одним из примеров декларативного языка, с помощью которого описывается графический интерфейс является QML (Qt Modeling Language). Данный язык используется в рамках Qt Quick для создания динамичных приложений. Он предоставляет разработчику следующие возможности:

* разметка GUI в виде компонентов;
* создание адаптивного GUI;
* создание анимации;
* обработка событий;
* вставка JavaScript-кода;
* интеграция с С++ логикой.

Достоинствами языка являются:

* удобочитаемость кода;
* интегрируемость с JavaScript;
* интегрируемость с Qt/C++.

Недостатками языка являются:

* (don’t know)

(Можно вставить диаграмму по тому как работает QML)

Пример кода на QML приведён в таблице 1

Таблица 1 – Примеры кода на QML

|  |
| --- |
| **Пример кода на QML** |
| [Rectangle](https://doc.qt.io/qt-5/qml-qtquick-rectangle.html) {  width: 200  height: 100  color: "red"  [Text](https://doc.qt.io/qt-5/qml-qtquick-text.html) {  anchors.centerIn: parent  text: "Hello, World!"  }  } |

При проектировании языка за ориентир будет взять именно QML.

## 1 Проектировании языка

Проектирование языка включается в себя следующие задачи:

1. описание синтаксиса;
2. описание структуры кода;
3. описание ключевых слов;
4. описание РБНФ.

## 1.1 Описание синтаксиса

Синтаксис представляет собой последовательное описание вложенных друг в друга компонентов, которые впоследствии будут отражены в графическом интерфейсе.

Описание компонентов начинается с названия компонента, с заглавной буквы. После чего описываются свойства и вложенные компоненты.

Для наглядности приведём пример кода, который выглядит следующим образом (Таблица 2).

Таблица 2 – Пример кода на языке Lui

|  |
| --- |
| **Пример кода языка Lui** |
| Window {  title: "Window app"  width: 200  height: 200    Label {  caption: "It's a label caption"  }  } |

В приведённом коде идёт описание компонента Window, который является основным окном приложения, и указание свойств:

1. title – заголовок окна;
2. width – ширина окна;
3. height – высота окна.

После чего, внутри компонента Window, описывается компонент Label, который является надписью, и для которого указывается свойство caption, что является самим текстом надписи.

Таблица 3 – Структура кода языка Lui

|  |
| --- |
| **Структура кода языка Lui** |
| Window {  title: "Window app"  [property: value]  Label {  caption: "It's a label caption"  }    [Component: { [property: value] }]  } |

Такая структура удобна для восприятия и не должна вызывать трудности в её понимании. Так как это понимается чуть ли не интуитивно. Вложенная структура друг за другом следующих элементов и свойства, характеризующие тот или иной компонент.

## 1.2 Описание ключевых слов

В качестве ключевых слов будут выступать имена компонентов и их свойства. Все компоненты можно разделить на четыре категории:

1. компоненты окон;
2. компоненты разметок;
3. компоненты ввода;
4. компоненты вывода.

## 1.3 Описание РБНФ

РБНФ (Расширенные формы Бэкуса-Наура) – это форма записи грамматики контекстно-свободных языков.

Приведём синтаксические уравнения языка Lui в таблице 4.

Таблица 4 – РБНФ языка Lui

|  |  |
| --- | --- |
| **№** | **Синтаксическое уравнение** |
| 1 | syntax = mainComponent "{" {property} {component} "}" |
| 2 | component = componentName "{" {property} {component} "}" |
| 3 | property = propertyName":" string | number |
| 4 | componentName = string | {string} |
| 5 | propertyName = string | {string} |
| 6 | mainComponent = "Window" |
| 7 | string = char | {char} |
| 8 | number = digit | {digit} |
| 9 | char = "A" | ... | "Z" |
| 10 | digit = "0" | ... | "9" |

Учитывая, что синтаксические уравнения описывают рекурсивную структуру, то для реализации синтаксического анализатора, распознающего вышеописанные РБНФ, будет использоваться метод рекурсивного спуска. Псевдокод приведён в таблице …

## 2 Изучение абстрактного синтаксического дерева

Абстрактное синтаксическое дерево – это дерево, которое строится в процессе синтаксического анализа и затем преобразовывается в целевой код с помощью генератора.

По описанным синтаксическим уравнениям в разделе 2.1 построим синтаксическое дерево. Корнем такого дерева будет является один из оконных компонентов. Дочерними элементами будут:

* список свойств в формате словаря
* список вложенных компонентов

Соответственно каждый элемент списка компонентов будет иметь такую же древовидную структуру. По описанию построим пример общего синтаксического дерева.

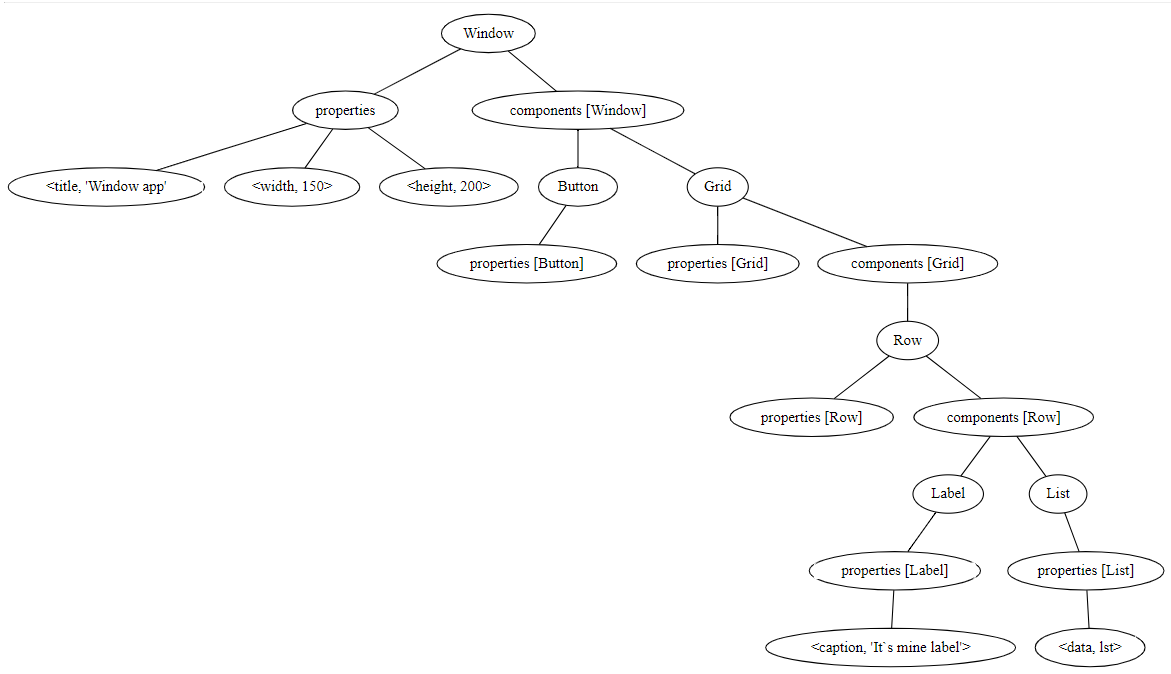


Рисунок 1 – Общее синтаксическое дерево

Для наглядности приведём пример кода (таблица ).

|  |
| --- |
|  |
| Window {  ….  } |

По приведённому коду построим синтаксическое дерева, получаемое при синтаксическом анализе. На рисунке (…) изображено дерево, а на рисунке (…) результат работы синтаксического анализатора, реализованного на Python.



(Вставить рисунок с итоговым результатом Syntaxer)

## 3 Проектирование и реализация транслятора

## 4 Изучение GUI-библиотеки Tkinter

## 5 Проектирование и реализация генератора

## 6 Проектирование и реализация приложения

В качестве итого приложения будет исполняемый файл, который принимает команды. Описание команд приведено в таблице ниже.

|  |  |
| --- | --- |
| **Команда** | **Описание** |
| --help | Выводит справку по командам |
| --file=[filename] | Путь до файла \*.lui файла |
| --debug | Включает режим отладки, записывая все шаги в файл |
| --version | Выводит текущую версию приложения |

# заключение

Я молодец!

# список используемых источников

Нормативные документы:

ГОСТ 7.32-2017

Приказ о курсовом проектировании

Книги:

Н. Вирт "Построение компиляторов"

Н. Вирт "Алгоритмы и структуры данных"

Электронные ресурсы:

<https://en.wikipedia.org/wiki/Widget_(GUI)>

<http://effbot.org/tkinterbook/>

<https://docs.python.org/3.7/library/tkinter.html>

<https://docs.python.org/3.7/library/unittest.html>

<https://docs.python.org/3.7/library/pydoc.html>

https://doc.qt.io/qt-5/qtqml-syntax-basics.html