МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение  
высшего образования

**«Дальневосточный федеральный университет»**

**ШКОЛА ЕСТЕСТВЕННЫХ НАУК**

**Кафедра прикладной математики, механики, управления и программного обеспечения**

КОЛЛЕКТИВНАЯ РАЗРАБОТКА ПРОЕКТА ТРАНСЛЯТОРА, ПЕРЕВОДЯЩЕГО ПОДМНОЖЕСТВО ЯЗЫКА GO В ЭКВИВАЛЕНТНОЕ ПОДМНОЖЕСТВО ЯЗЫКА С

**КУРСОВАЯ РАБОТА**

по дисциплине «Технологии коллективной промышленной разработки информационных систем» по образовательной программе подготовки бакалавров по направлению 09.03.04 Программная инженерия

Выполнили студенты гр. Б8117-09.03.04

\_\_\_\_\_\_\_\_\_\_\_ Копосова Д. А.

\_\_\_\_\_\_\_\_\_\_\_ Кузьменко Д. А.

\_\_\_\_\_\_\_\_\_\_\_ Макарова О. А.

\_\_\_\_\_\_\_\_\_\_\_ Сахаров И. А.

Руководитель: доктор технических наук Гриняк В.М.

\_\_\_\_\_\_\_\_\_\_\_\_ «\_\_\_» \_\_\_\_\_\_\_\_ 2021 г.

(подпись)

Защищён с оценкой \_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

(подпись) И.О. Фамилия

«\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2021 г.

г. Владивосток

2021 г.

**Оглавление**

[Введение 4](#_Toc55052933)

[1 Регламент проведения инспекции 5](#_Toc55052934)

[1.1 Процедура организации и проведения формальной инспекции 5](#_Toc55052935)

[1.2 Процедура организации и проведения неформальной инспекции 5](#_Toc55052936)

[1.3 Роли участников инспекции 6](#_Toc55052937)

[1.4 Порядок организации инспекции 6](#_Toc55052938)

[1.5 Порядок подготовки к инспекции 7](#_Toc55052939)

[1.6 Перечень статусов и степени важности замечаний 7](#_Toc55052940)

[1.7 Метрики, характеризующие эффективность инспекций 8](#_Toc55052941)

[2 Модель состояний задач 9](#_Toc55052942)

[2.1 Состояния задач 9](#_Toc55052943)

[2.2 Создание новой задачии 9](#_Toc55052944)

[2.3 Перехода задачи из состояния в состояние 10](#_Toc55052945)

[3 План проекта 10](#_Toc55052946)

[3.1 Проектная команда 10](#_Toc55052947)

[3.2 График разработки 11](#_Toc55052948)

[4 Требования к проекту 12](#_Toc55052949)

[4.1 История изменений документа 12](#_Toc55052950)

[4.2 Подсистемы проекта 12](#_Toc55052951)

[4.3 Требования к подсистеме «Пользовательский интерфейс» 12](#_Toc55052952)

[4.4 Требования к подсистеме «Лексический анализатор» 15](#_Toc55052953)

[4.5 Требования к подсистеме «Синтаксический анализатор» 16](#_Toc55052954)

[4.6 Требования к подсистеме «Семантический анализатор» 16](#_Toc55052955)

[5 Архитектура проекта 17](#_Toc55052956)

[6 Программа измерений проекта 17](#_Toc55052957)

[7 Перечень задач проекта 17](#_Toc55052958)

[8 Рекомендации по кодированию 17](#_Toc55052959)

[9 План тестирования проекта 17](#_Toc55052960)

[Заключение 18](#_Toc55052961)

[Список литературы 19](#_Toc55052962)

Введение

Из-за больших объемов проектов разработка программного обеспечения ведется коллективом специалистов. Работая в коллективе, отдельные специалисты должны взаимодействовать друг с другом, обеспечивая целостность проекта, что при отсутствии удовлетворительных средств описания поведения сложных систем, упоминавшемся выше, достаточно сложно. Причем чем больше коллектив разработчиков, тем сложнее организовать процесс работы

Рано или поздно каждый программист садится писать свой компилятор. Это интересная и довольно непростая задача. В большинстве случаев разработка компилятора – это коллективная разработка.

Поэтому **цель курсовой работы:** коллективная разработка проект транслятора, который преобразует программу, содержащую подмножество языка Go в программу, содержащую подмножество языка C++, генерируя эквивалентный исходный код. Язык задаётся множеством цепочек, принадлежащих ему.

**Задачи курсовой работы:**

* Разработать регламент проведения инспекции;
* Разработать модель состояний задач;
* Разработать план проекта;
* Разработать требования к проекту;
* Разработать архитектуру проекта;
* Разработать программу измерений проекта;
* Разработать перечень задач проекта;
* Разработать рекомендации по кодированию;
* Разработать план тестирования проекта.

1. Регламент проведения инспекции

1.1 Процедура организации и проведения формальной инспекции

Формальной инспекции подвергаются готовые части рабочих продуктов, где необходимо проверить документы дизайна или работоспособность кода с прилагающимися тестами для проверки. В процедуре формальной инспекции должны принимать участие 3 члены команды.

**Этапы формальной инспекции:**

1.Планирование инспекции;

2.Назначение инспекции;

3.Собрания по инспекции;

4.Завершение инспекции (распространение результатов, проверка исправления недостатков в рабочем продукте).

**Назначение инспекции**

Функции автора – инициировать формальную инспекцию рабочего продукта. Для этого автор должен оповестить руководителя проекта

Функции председателя – убедиться, что рабочий продукт удовлетворяет критерию готовности к формальной инспекции.

Функции инспектора – самостоятельно независимо от других участников инспекции изучить предоставленный для инспекции рабочий продукт, заполнить необходимые поля протокола подготовки к формальной инспекции

1.2 Процедура организации и проведения неформальной инспекции

Неформальные инспекции подвергаются маленькие части рабочих продуктов, где необходимо проверить работоспособность кода.

В процедуре формальной инспекции могут принимать участие 2 члена команды.

**Этапы неформальной инспекции:**

1. Назначение инспекции.

2. Собрания по инспекции.

3.Завершение инспекции (распространение результатов, проверка исправления недостатков в рабочем продукте).

1.3 Роли участников инспекции

1. **Автор (Author)** - сотрудник, разработавший инспектируемый рабочий продукт, либо сделавший инспектируемые изменения в существующем рабочем продукте.

**В роли могут быть: Кузьменко Дмитрий;**

1. **Председатель (Moderator)** - ответственный сотрудник, выполняющий роль председателя инспекции.

**В роли могут быть: Копосова Дарья, Сахаров Игорь;**

1. **Ведущий (Presenter)** - сотрудник, представляющий рабочий продукт инспекторам.

**В роли могут быть: Макарова Ольга;**

1. **Инспектор (Inspector)** - сотрудник, ответственный за эффективную

проверку инспектируемого рабочего продукта.

**В роли могут быть: Сахаров Игорь, Копосова Дарья, Макарова Ольга;**

1. **Инженер по качеству** – сотрудник, ответственный за анализ метрик после инспекции.

**В роли могут быть: Копосова Дарья.**

1.4 Порядок организации инспекции

1. Планирование инспекции. **Автор оповещает о необходимости проведения инспекции в специальном чате, где присутствуют все участники команды.**
2. Назначение инспекции. **Определяется количество участников (председателей и инспекторов) и назначается дата и время проведения инспекции.**
3. Собрания по инспекции. **В назначенное время по прибытию всех членов инспекции в указанное время проводится инспекция части продукта. (Возможно использование вспомогательного инструмента для проведения инспекции Discord)**
4. Завершение инспекции (распространение результатов, проверка исправления недостатков в рабочем продукте). **Обсуждение недочетов и моментов, которые необходимо исправить**

1.5 Порядок подготовки к инспекции

Время и дата проведения инспекции устанавливается на этапе назначения инспекции в зависимости от количества участников инспекции выбирается наиболее подходящие сроки проведения для всех членов команды в пределах 2 дней после планирования инспекции.

Дата инспекций не назначается на субботу или воскресенье или на праздничные дни.

1.6 Перечень статусов и степени важности замечаний

**Допустимые значения статуса замечания**

* **Дефект (Defect)** - проблема, которая найдена на фазе, отличной от той, на которой внесена.
* **Ошибка (Error)** – проблема, которая найдена на той же фазе, на которой внесена.
* **Комментарий (Comment)** – это наблюдение, предложение, рекомендация или улучшение, предложенное для будущего выпуска рабочего продукта или вопрос, требующий разъяснения.
* **Замечание для исследования (Investigate)** – проблема, природа которой не может быть определена на собрании и требует дополнительного исследования.

**Допустимые значения степени серьёзности замечания**

* Особо важная (Major)
* Средняя (Moderate)
* Мелкая, незначительная (Minor)
* Другие (Other)

**Порядок верификации учёта замечаний**

* Особо важная (Major) – неисправность или ошибки в программном продукте более чем 50%;
* Средняя (Moderate) - неисправность или ошибки в программном продукте более чем 20 %;
* Мелкая, незначительная (Minor) - ошибки в программном продукте более чем 10%;
* Другие (Other) – в иных случаях.

1.7 Метрики, характеризующие эффективность инспекций

Метрики по инспекциям собираются инженером по качеству **Сахаровым Игорем** после каждой инспекции и анализируются с помощью контрольных карт

**Метрики по инспекциям:**

* **Inspection Fault Density (IFD)**

IFD = (Количество найденных ошибок / Размер рабочего продукта)

Стратегическая цель метрики – повысить качество разрабатываемого ПО.

Изучаемый объект метрики – инспекция, измеряемый атрибут – плотность найденных в ходе инспекции ошибок.

Единица измерения – ошибка / <страница, требование, LOC, тест>.

* **Inspection Preparation Rate (IPR)**

IPR = (Количество инспекторов \* Размер продукта) / Общее время подготовки;

Стратегическая цель метрики – повысить качество разрабатываемого ПО.

Изучаемый объект метрики – подготовка к инспекции, измеряемый атрибут – производительность подготовки к инспекции.

Единица измерения –<страница, требование, LOC, тест>/ час.

* **Inspection Rate (IR)**

IR = Размер продукта / Общее время инспектирования;

Стратегическая цель метрики – повысить качество разрабатываемого ПО.

Изучаемый объект метрики – проведение инспекции, измеряемый атрибут – производительность инспектирования.

Единица измерения –<страница, требование, LOC, тест>/ час.

1. Модель состояний задач

2.1 Состояния задач

**Перечень возможных состояний задачи и их интерпретация:**

* New (Новая задача). Аналитиками создана новая задача;
* Task assignment (Задача назначена). Аналитик назначает задачу на конкретного автора;
* Coding (Задача в разработке). Задача в разработке у автора;
* Сheck or testing (Проверка или тестирование). Задача находится в состоянии проверки или тестируется;
* Closed (Задача выполнена). Задача закрыта.

2.2 Создание новой задачии

**Правила создания новой задачи:**

Список сотрудников, которые могут создавать новые задачи: **Копосова Дарья, Сахаров Игорь**

Новые задачи могут создаваться в случае выдвижения новых требований клиентом (Артемьевой Ириной Леонидовной).

2.3 Перехода задачи из состояния в состояние

**Правила перехода задачи из состояния в состояние:**

* **New** →**Task assignment**. Переход в случае назначения аналитиком автора на задачу;
* **Task assignment**→ **Coding.** Переход, если назначенный автор начал работать над задачей
* **Coding** → **Сheck or testing.** Переход в случае, если автор закончил работу над задачей и можно провести тестирование.
* **Сheck or testing** → **Closed.** Переход в случае, если задача протестирована и 80% тестов прошли удачно.

1. План проекта

3.1 Проектная команда

Таблица 1. Проектная команда

|  |  |
| --- | --- |
| **Проектная команда** | **Члены команды** |
| Тимлид | Макарова Ольга |
| Кодер 1 | Кузьменко Дмитрий |
| Кодер 2 | Макарова Ольга |
| Инженер по качеству | Сахаров Игорь |
| Технический писатель | Копосова Дарья |
| Тестировщик 1 | Копосова Дарья |
| Тестировщик 2 | Сахаров Игорь |

3.2 График разработки

Таблица 2. График разработки

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Дата**  **Разработка** | **14.9.2020-21.9.2020** | **22.9.2020-30.9.2020** | **1.10.2020-12.10.2020** | **13.10.2020-21.10.2020** | **22.10.2020-31.10.2020** | **1.11.2020-15.11.2020** | **16.11.2020-31.11.2020** | **1.12.2020-10.12.2020** | **11.12.2020-20.12.2020** | **21.12.2020-31.12.2020** | **1.1.2021-10.1.2021** | **11.1.2021-20.1.2021** | **21.1.2021-10.1.2021** |
| **План проекта программного Обеспечения** |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Спецификация маркетинговых требований** |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Спецификация требований к программному обеспечению** |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Спецификация проектирования Программного Обеспечения** |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Разработка грамматики для лексического анализатора** |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Разработка лексического анализатора** |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Тестирование лексического анализатора** |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Разработка грамматики для синтаксического анализатора** |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Разработка синтаксического анализатора** |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Тестирование синтаксического анализатора** |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Разработка семантического анализатора** |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Тестирование семантического анализатора** |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Составление документации** |  |  |  |  |  |  |  |  |  |  |  |  |  |

1. Требования к проекту

4.1 История изменений документа

Таблица 3. История изменений

|  |  |  |
| --- | --- | --- |
| **Дата** | **Автор** | **Внесённые изменения** |
| 22.09.2020 | Д.А. Копосова | Исходная версия требований |
| 25.09.2020 | Д.А. Копосова | Добавлены требования к подсистеме лексического анализатора |
| 1.10.2020 | И.А. Сахаров | Добавлены требования к пользовательскому интерфейсу |
| 7.10.2020 | Д.А. Копосова | Добавлены требования к подсистеме синтаксического анализатора |
| 12.10.2020 | Д.А. Копосова | Добавлены требования к подсистеме семантического анализатора |

Программный продукт *GoToC* предназначен для компиляции программ, написанных на языке Go, в программу на языке С.

4.2 Подсистемы проекта

Программный продукт *GoToC* состоит из следующих подсистем:

1. Пользовательский интерфейс;
2. Подсистема лексического анализатора;
3. Подсистема синтаксического анализатора;
4. Подсистема семантического анализатора.

4.3 Требования к подсистеме «Пользовательский интерфейс»

**Требование REQ\_UI\_001.1**

Функция загрузки кода из файла на языке Go должна быть доступна для пользователя в ячейке БТ1 с надписью «Загрузить файл» (Рис. 1).



Рисунок 1 – Панель основных функций

**Требование REQ\_UI\_001.2**

Функция сохранения компилируемого кода должна быть доступна для пользователя в ячейке БТ2 с надписью «Сохранить» и по нажатии сочетания клавиш «Ctrl + F1» (Рис. 1).

**Требование REQ\_UI\_001.3**

Функция компиляция кода должна быть доступна для пользователя в ячейке БТ3 с надписью «Компилировать» и по нажатии сочетания клавиш «Ctrl + F5» (Рис. 1).

**Требование REQ\_UI\_001.4**

Функция сохранения скомпилируемого кода должна быть доступна для пользователя в ячейке БТ4 с надписью «Компилировать» и по нажатии сочетания клавиш «Ctrl + F2» (Рис. 1).

**Требование REQ\_UI\_001.5**

Функция отменить (undo) действие должна быть доступна для пользователя в ячейке БТ5 с надписью «Компилировать» и по нажатии сочетания клавиш «Ctrl + U» (Рис. 1).

**Требование REQ\_UI\_001.6**

Функция повторно выполнить (redo) должна быть доступна для пользователя в ячейке БТ6 с надписью «Компилировать» и по нажатии сочетания клавиш «Ctrl + R» (Рис. 1).

**Требование REQ\_UI\_002.1**

Текст компилированного текста должен отображаться в textbox для исходного кода на Go (Рис. 2).

**Требование REQ\_UI\_002.2**

Текст скомпилированного текста должен отображаться в textbox для целевого кода на С++ (Рис. 2).

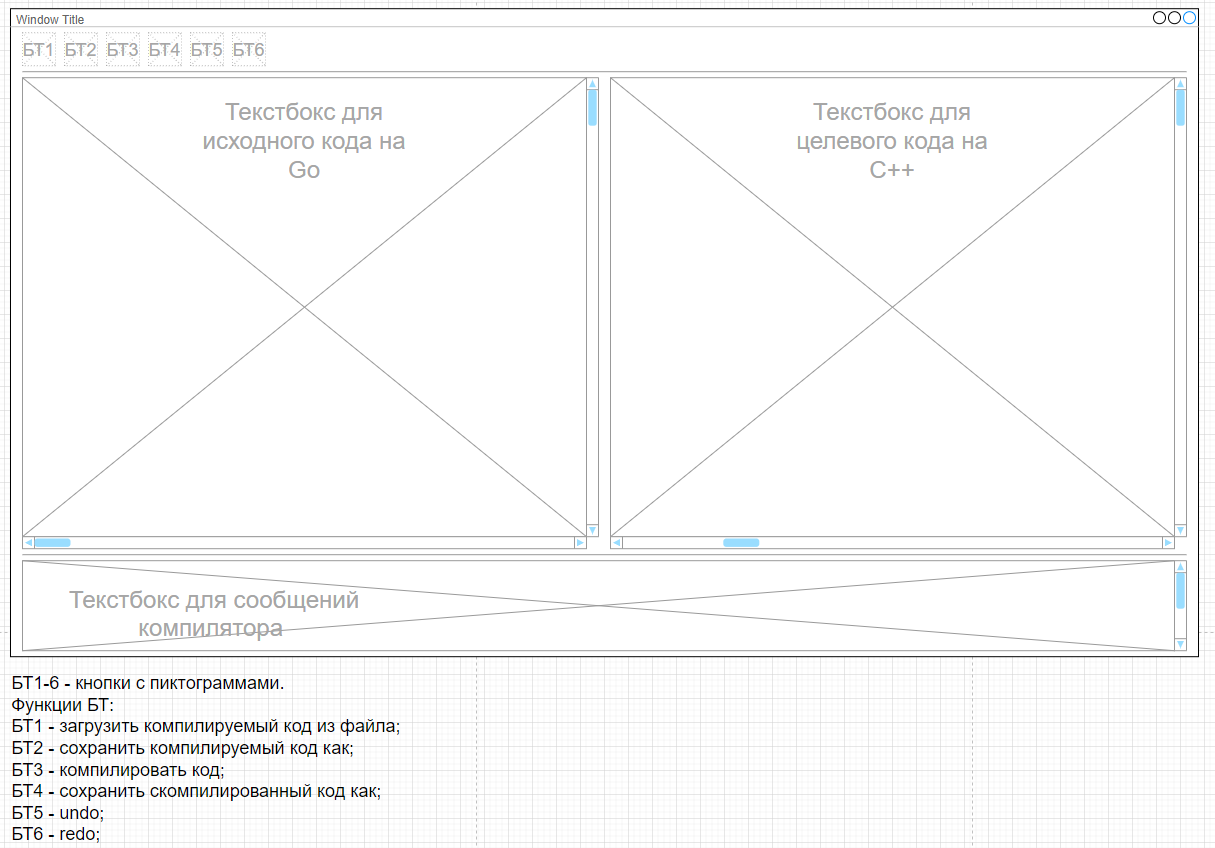


Рисунок 2 – Textbox исходного и целевого кода

**Требование REQ\_UI\_003**

Текст сообщений компилятора должен отображаться в textbox для сообщений компилятора (Рис. 3).

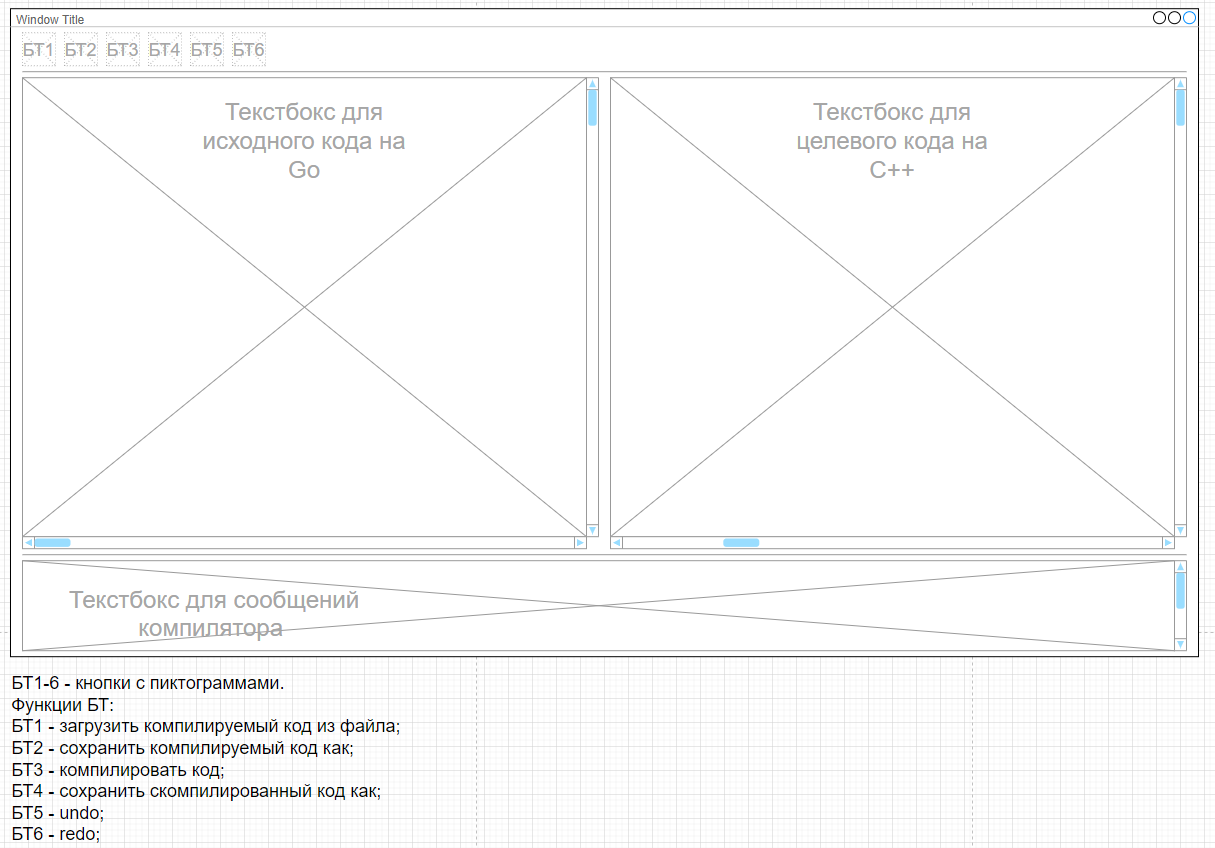


Рисунок 3 - Textbox для сообщений компилятора

**Требование REQ\_UI\_004**

На рисунке 4 отображены общие требования к интерфейсу.

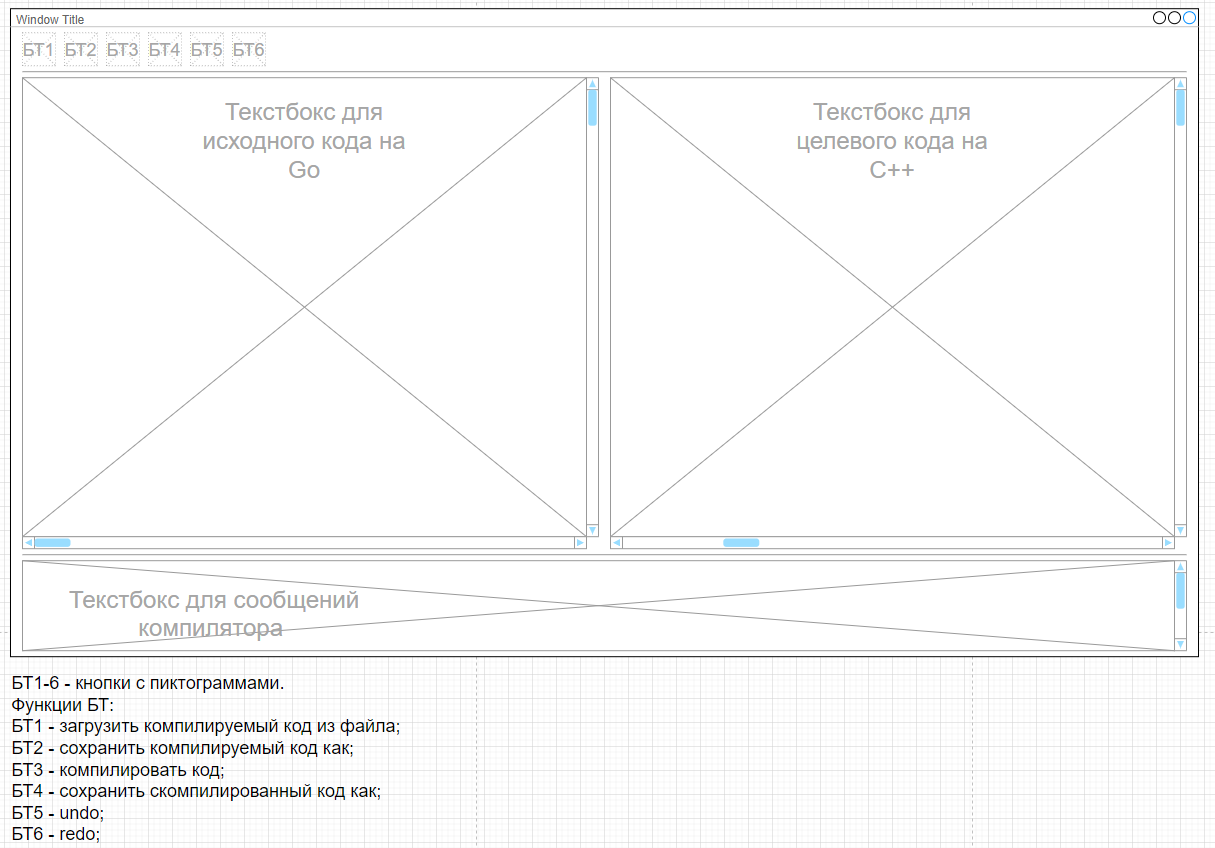


Рисунок 4 – Пользовательский интерфейс.

**Требование REQ\_UI\_005**

Если в textbox для исходного кода был введен код с ошибкой или код на другом язык, то в textbox для сообщений компилятора будет выводиться сообщения ошибке.

4.4 Требования к подсистеме «Лексический анализатор»

**Требование REQ\_LA\_001**

Подсистема лексический анализатор должна подготовить входную последовательность из textbox для исходного кода для подсистемы синтаксический анализатор.

**Требование REQ\_LA\_002**

Подсистема должна считывать входной поток символов из textbox для исходного кода.

**Требование REQ\_LA\_003**

Подсистема должна распознавать лексемы в контексте грамматики путем классификации по типам: BoolKeyword,IntKeyword, FloatKeyword, StringKeyword, BoolLiteral, IntLiteral, FloatLiteral, StringLiteral, Identifier, AssignmentInFor, Assignment, Semicolon, Colon, Comma, OpenningRoundBracket, ClosingRoundBracket, OpenningCurlyBracket, ClosingCurlyBracket, Comparison, Arithmetic, Multiply, ForKeyword, ReturnKeyword, SwitchKeyword, CaseKeyword, DefaultKeyword, IfKeyword, ElseKeyword, FuncKeyword, VarKeyword, ConstKeyword, EndOfStatement, Undefined)

**Требование REQ\_LA\_004**

В случае, когда подсистема не может идентифицировать лексему, она должна классифицировать ее как специальный токен-ошибку.

**Требование REQ\_LA\_005**

Каждая лексема должна представлять собой структуру, которая содержит идентификатор лексемы, последовательность символов лексем, выделанной из входного потока и порядковый номер лексемы.

**Требование REQ\_LA\_006**

Подсистема лексического анализатора на вход в подсистему синтаксического анализатора подает таблицу из лексем и их идентификаторов.

4.5 Требования к подсистеме «Синтаксический анализатор»

**Требование REQ\_STA \_001**

Подсистема синтаксический анализатор должна проверить входную последовательность из лексического анализатора и подать проверенную последовательность на вход семантическому анализатору.

**Требование REQ STA \_002**

Подсистема по средствам входных данных из лексического анализатор должна построить дерево разбора.

**Требование REQ\_STA \_003**

С помощью дерева разбора подсистема должна проверить синтаксическую корректность кода.

**Требование REQ\_STA \_004**

В случае нахождения ошибки при построении дерева разбора система должна вывести сообщения об ошибках в textbox для сообщения компилятора.

**Требование REQ\_STA \_005**

Подсистема синтаксического анализатора на вход в подсистему семантического анализатора подает дерево разбора в случае успешного его построения.

4.6 Требования к подсистеме «Семантический анализатор»

**Требование REQ\_SMA \_001**

Подсистема семантический анализатор должна проверить входную последовательность из синтаксического анализатора и подать проверенную последовательность на вход генератору кода.

**Требование REQ\_SMA \_002**

Подсистема по средствам входных данных из синтаксического анализатора должна провести проверку контекстных условий.

**Требование REQ\_SMA \_003**

В случае нахождения ошибки при проверки контекстных условий система должна вывести сообщения об ошибках в textbox для сообщения компилятора.

**Требование REQ\_SMA \_004**

Подсистема семантического анализатора после успешной проверки условий должна подать проверенную последовательность на генератор кода.

1. Архитектура проекта
2. Программа измерений проекта
3. Перечень задач проекта
4. Рекомендации по кодированию
5. План тестирования проекта

Заключение

В ходе курсовой работы были решены следующие задачи:

* Разработан регламент проведения инспекции;
* Разработана модель состояний задач;
* Разработан план проекта;
* Разработаны требования к проекту;
* Разработана архитектуру проекта;
* Разработана программа измерений проекта;
* Разработан перечень задач проекта;
* Разработаны рекомендации по кодированию;
* Разработан план тестирования проекта.

Список литературы

1. Ахо, А. Компиляторы: принципы, технологии и инструментарий компиляторов / И. В. Красиков. – М.: Вильямс, 1986. – 1184 с. ;
2. Бруно, К. Л. LLVM: инфраструктура для разработки компиляторов / К. Л. Бруно, А. Рафаэль. – М.: ДМК Пресс, 2015. – 342 с.;
3. Вирт, Н. Построение компиляторов / Н. Вирт. – М.: ДМК Пресс, 2010. – 192 с.;
4. Racket Documentation [Электронный ресурс]. URL: <https://docs.racket-lang.org/> (дата обращения 16.05.2020).