Нумерация:

Номер задачи пишется в квадратных скобках и состоит из следующих частей:

1. сокращение IS;
2. название подсистемы. Возможные значения: FRONT – задачи, относящиеся к интерфейсу программы, IO – задачи, относящиеся к подсистеме ввода и вывода, TRANSLATE – задачи, относящиеся к подсистеме трансляции;
3. порядковый номер. Для каждой подсистемы используется отдельный порядковый числовой номер для требования.

Между частями номера требования пишется символ «\_». После номера задачи пишутся заголовок задачи, развёрнутое описание и ссылки на требования, по которым составлена задача.

Задачи по разработке графического интерфейса пользователя:

Разработать пользовательский интерфейс

Разработать пользовательский интерфейс, готовый к подключению к нему компонентов транслятора. В интерфейсе должны присутствовать все предполагаемые элементы управления. Также требуется продумать цветовую схему.

Ссылки: REQ\_FRONT\_1 – 16.

Разработать класс элемента управления «Кнопка»

Написать класс пользовательского элемента управления, выполняющего функцию кнопки.

Ссылки: REQ\_FRONT\_2.

Разработать класс элемента управления «Поле ввода»

Написать класс пользовательского элемента управления, выполняющего функцию поля для ввода текста. Поле должно иметь возможность блокировать изменение текста и показывать плейсхолдер, должно обладать рамкой, меняющей цвет при получении/потере фокуса ввода.

Ссылки: REQ\_FRONT\_2, REQ\_FRONT\_3, REQ\_FRONT\_5.

Диалоги для чтения из файла и сохранения в файл

Добавить два диалога для работы с файловой системой: один для чтения кода на языке C# из файлов формата TXT и CS, второй для сохранения кода на языке Java в файлы формата TXT и JAVA.

Ссылки: REQ\_FRONT\_6, REQ\_FRONT\_7, REQ\_IO\_1, REQ\_IO\_3.

Подтверждение выхода из программы

Добавить окно с подтверждением выхода, появляющееся при наличии несохранённых изменений.

Ссылки: REQ\_FRONT\_16.

Задачи по разработке подсистемы ввода/вывода:

Проверка входного файла на пустоту

Добавить проверку файла, выбранного с помощью диалога, на пустоту. Если файл пустой, выводить соответствующее предупреждение в виде окна с сообщением.

Ссылки: REQ\_IO\_2.

Задачи по разработке подсистемы трансляции:

Разработать лексический анализатор

Разработать лексический анализатор, выполняющий разбиение входного кода на языке C# на лексемы. Выходными данными анализатора должен быть массив структур вида

* текст лексемы;
* тип лексемы;
* номер строки в тексте, содержащей эту лексему;
* номер столбца в тексте, в котором расположен первый символ лексемы.

Возможные типы лексем: идентификатор, оператор, оператор цикла, оператор ветвления, тип данных. Тип лексемы представить в виде статической целочисленной константы (например, идентификатор – static int TOKEN\_TYPE\_IDENTIFIER = 0).

Лексический анализатор должен выполнять проверку кода на наличие следующих ошибок:

* символ, отсутствующий в алфавите входного языка;
* лексема, отсутствующая во входном языке.

Ссылки: REQ\_TRANSLATE\_1, REQ\_TRANSLATE\_2, REQ\_TRANSLATE\_4.

Разработать синтаксический анализатор

Разработать синтаксический анализатор, выполняющий построение абстрактного дерева разбора на основе массива структур, полученного из лексического анализатора. Синтаксический анализатор должен проверить исходный код по следующим критериям:

* наличие непарных скобок, то есть ( { [;
* отсутствие круглых скобок с логическим выражением в операторах циклов и ветвления;
* правильность математических и логических выражений;
* наличие оператора else без соответствующего оператора if;
* отсутствие точки с запятой в конце строки и т.д.

Ссылки: REQ\_TRANSLATE\_1, REQ\_TRANSLATE\_2, REQ\_TRANSLATE\_4.

Разработать семантический анализатор

Разработать семантический анализатор, получающий на вход абстрактное дерево разбора, сгенерированное синтаксическим анализатором и выполняющий проверку исходного кода на наличие следующих ошибок:

* использование необъявленного идентификатора;
* использование идентификатора до его объявления;
* несоответствие типов данных;
* переобъявление существующего идентификатора;
* несоответствие количества операндов в объявлении и вызове функции.

Ссылки: REQ\_TRANSLATE\_1, REQ\_TRANSLATE\_2, REQ\_TRANSLATE\_4.

Разработать генератор кода

Разработать генератор кода, получающий на вход абстрактное дерево разбора и генерирующий на его основе исходный код на языке Java. Генератор кода должен заменять несуществующие лексемы в языке Java на их аналоги. Например, «nuint» на «int».

Ссылки: REQ\_TRANSLATE\_1, REQ\_TRANSLATE\_2, REQ\_TRANSLATE\_3, REQ\_TRANSLATE\_4.