Лексический анализатор

1) Реализовать функцию readFile, читающую посимвольно текстовые файлы. Интерфейс:
   std::string readFile(std::string fileName).

2) Реализовать функцию, осуществляющую лексический разбор текста согласно спецификации прототипа. В случае обнаружения ошибки прерывать разбор с помощью исключения, содержащего информацию о месте возникновения ошибки и о её причине. Интерфейс:
   std::vector<Token> makeTokens(std::string text)

Синтаксический анализатор

1) Реализовать функцию, осуществляющую синтаксический разбор массива токенов согласно правилам разбора из спецификации, и строящую в результате IR. В случае обнаружения ошибки прерывать разбор с помощью исключения, содержащего информацию о месте возникновения ошибки и о её причине. Интерфейс:
   IR* parseProgram(std::vector<Token> tokens)

2) Реализовать функцию семантической проверки. В случае ошибки проверки прерывать компиляцию с помощью исключения. Минимальный набор правил для проверки:
   a) Ни в одной функции нет двух переменных с одинаковыми именами
   b) Все использующиеся переменные определены
   c) Все вызываемые функции существуют

   Интерфейс:
   void checkCorrectness(IR* ir)

Генерация и сериализация байткода

1) Реализовать функцию, строящую байткод по IR. Интерфейс:
   Bytecode* generateBytecode(IR* ir)

2) Реализовать функцию, печатающую байткод в текстовом представлении в заданный файл. Интерфейс:
   void writeBytecode(Bytecode* bc, std::string fileName)

Десериализация и исполнение байткода

1) Реализовать функцию, читающую байткод из текстового представления. Интерфейс:

   Bytecode* readBytecode(std::string fileName)

2) Реализовать функцию, интерпретирующую байткод. Интерпретацию выполнять в соответствии с семантикой, заданной в спецификации. Интерфейс:

   void interpret(Bytecode* bytecode)