Министерство образования и науки РФ
РЫБИНСКИЙ ГОСУДАРСТВЕННЫЙ АВИАЦИОННЫЙ
ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
имени П. А. Соловьева

Факультет радиоэлектроники и информатики
Кафедра МПО ЭВС

Специальность
Программная инженерия

*КУРСОВОЙ ПРОЕКТ*

по курсу: «Конструирование компиляторов»

на тему:
«Реализация интерпретатора ядра Лисп на C++»

Техническое задание

Студенты гр. ПИМ-20 Смирнов М. Ю.

Руководитель:

Рыбинск 2020

## РБНФ разрабатываемого языка:

```
Prog := S_expr, {S_expr};

S_expr := Atom | List;

List := ‘()’ | ‘(‘S_expr, PS_expr}’)’;

Atom := Numb | Symbol;

BasicFunctions := ArifmF | LogicalF | ListF | DefinesF | IfF | SpecialS;

ArifmF := + | - | * | /;

LogicalF := < | <= | == | != | > | >=;

ListF := car | cdr | cons;

DefineF := defun | set;

IfF := if;

SpecialS := T | Nil;

Symbol := BasicFunctions | Variable;
```

## Функциональные требования

Выполнение программы на интерпретаторе может быть реализовано двумя способами: прямое вычисление и ленивое вычисление. При ленивом вычислении, значения аргументов не вычисляются, а передаются как выражения в функцию. Непосредственное вычисление происходит только после полного формирования выражения. При прямом вычислении аргументы функции перед передачей вычисляются. В функцию при этом передается уже вычисленное значение аргумента. При реализации ядра Лиспа необходимо реализовать прямой метод вычислений.

Существуют базовый функции, которые уже определены и которые необходимо реализовать:

1. Арифметические функции:
    a. Сложение (+).
    b. Вычитание (-).
    c. Умножение (*).
    d. Вещественное деление (/).
2. Логические функции:
    a. Меньше/больше (< / >).
    b. Меньше либо равно/больше либо равно (<= / >=).
    c. Равенство (==).
    d. Неравенство (!=).
3. Функции для обработки элементов списка:
    a. Получение первого элемента списка (car).
    b. Получение остатка списка (cdr).
    c. Объединение элементов в список (cons).
4. Определение функций и переменных:
    a. Объявление функции (defun).
    b. Объявление переменной с инициализацией (set).
5. Условный оператор (if):
6. Зарезервированные слова:

a. T.
b. Nil.

Интерпретатор должен считывать программу на разрабатываемом языке программирования из файла.

## Входные данные

Разрабатываемый интерпретатор должен производить считывание из файла. Имя файла передается в качестве аргумента при запуске исполняемого файла.

## Выходные данные

Результатом работы программы является вывод в консоль всего, что выводит программа, написанная на разрабатываемом языке программирования.

## Язык программирования

В качестве языка программирования для реализации следует использовать C++.

## Предполагаемые проблемы в процессе разработки

Анализ предметной области позволил выявить следующие проблемы:

1. Списки хранятся в виде двух указателей car и cdr. Функция cons производит создание новой ячейки и создает связи с переданными ей указателями. Списки могут содержать различные элементы внутри себя. Для поддержания этого принципа требуется разработать специальную структуру данных. Предположительно это будет древовидная структура, состоящая из экземпляров класса Cell. Такой подход позволит использовать встроенный механизм полиморфизма языка C++ для хранения различных объектов.
2. Необходимо разработать структуру для хранения окружения Environment. Это структура хранит имена переменных или функций и их значения или определения. Предположительно следует использовать структуру словаря, то есть хранить ключ и значение. Неоднозначным моментов является следующее: необходимо ли создавать новое окружение при заходе в функцию? Скорее всего, это необходимо, однако как именно организовать создание нового окружения и как к нему получать доступ, неочевидно. Также непонятно следует ли учитывать вложенность окружений друг в друга. Предположительно будет деление на глобальный и локальный контексты. Глобальный доступен всегда. Локальным считается окружение, в котором происходит текущее вычисление. Возможно, следует сделать дерево окружений для создания вложенности локальных окружений друг в друга. В этом случае следует использовать правило ближайшего.
3. Каким образом хранить пользовательские функцию в окружении? Возможны два варианта решения:
    a. Хранить в виде исходного кода.
    b. Преобразовать к внутренним объектам и хранить в них.

    Автор склоняется к первому варианту.
4. Как лучше организовать передачу результата функции наружу? Пока идея только одна, а именно передавать возвращаемое значение через стек. Аналогичным образом передавать и аргументы в функцию. Следовательно, перед вызовом функции необходимо поместить вычисленные аргументы в стек. Передать управление функции (предположительно с созданием нового окружения). При завершении функции, ее результат помещается на вершину стека.