МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

РЫБИНСКИЙ ГОСУДАРСТВЕННЫЙ АВИАЦИОННЫЙ
ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
имени П. А. Соловьева

Факультет радиоэлектроники и информатики
Кафедра МПО ЭВС

Специальность
Программная инженерия

СПЕЦИФИКАЦИЯ ТРЕБОВАНИЙ К ПО

интерпретатор функционального языка программирования

«ЛИСПА»

Студента гр. ПИМ-20 Смирнов М. Ю.

Научный руководитель Пинаев В. Н.

Рыбинск 2020

1. Наименование программы

1.1 Полное наименование программы

Интерпретатор функционального языка программирования «ЛИСПА», клона языка «ЛИСП».

1.2 Краткое наименование программы

«ЛИСПА».

2. Плановые сроки начала и окончания работ

Дата начала: 01 ноября 2020 г.

Дата окончания: 30 декабря 2020 г.

3. Назначение и цели создания программы

3.1 Назначение программы

Разбор программы, написанной на функциональном языке программирования «ЛИСПА», и ее выполнение.

3.2 Цели создания программы

Изучение принципов построения интерпретаторов для функциональных языков программирования. Применение полученных знаний на практике.

4. Требования к программе

4.1 Требования к программе в целом

4.1.1 Требования к структуре и функционированию системы

4.1.1.1 Общее описание программы

Разрабатываемая программа позволит выполнять программы, написанные на функциональном языке программирования «ЛИСПА». РБНФ языка «ЛИСПА» представлен в приложении А; служебные слова с описанием выполняемых функций представлены в приложении Б; синтаксис языка «ЛИСПА» описан в приложении В.

Входными данными для программы являются файлы, содержащие программы, написанные на языке программирования «ЛИСПА». Передача программы для обработки интерпретатором происходит через командную строку при запуске интерпретатора. Важно отметить, что передается путь до файла с программой, а не текст программы. Расширение файла с программой должно быть «lispa». Данный файл является текстовым файлом.

После загрузки программы интерпретатор производит разбор программы и ее выполнение. В случае возникновения ошибки, она будет выведена на экран консоли. При этом дальнейшее выполнение программы будет прервано.

Результатом работы интерпретатора является вывод информации, которую выводит исполняемая программа. И дополнительно результат работы программы.

4.1.1.2 Описание режимов функционирования

Предусмотрен только один режим функционирования. При запуске интерпретатору передается путь до файла с программой. Далее производится загрузка программы интерпретатором и ее выполнение. Интерпретатор должен работать в консольном режиме.

4.1.1.3. Структура программы

4.2 Требования к функциям, выполняемыми системой

4.2.1 Требования к функциональным возможностям

4.2.1.1 Технология работы программы

Необходимо реализовать четыре регистра:

- Регистр стека

Содержит аргументы для вызова функций, либо аргументы, являющиеся результатом вызова функции.

- Регистр окружения

Является ассоциативным массивом, содержащим связь между символьными названиями объектов и самими объектами.

- Регистр команд

Содержит код на языке программирования «ЛИСПА».

- Регистр состояний

Содержит тройки регистров стека, окружения и команд. Необходим для возможности откатить интерпретатор к предыдущему состоянию регистров.

Регистры представляют собой стековую структуру данных.

Изначально регистр стека, регистр окружения и регистр состояний пусты. В регистре команд содержится вся исходная программа на языке программирования «ЛИСПА».

При корректной исходной программе, после ее выполнения в регистре стека должно остаться единственное значение, являющееся результатом работы программы. Регистр команд и регистр состояний должны оказаться пустыми.

4.2.1.2 Функциональные возможности модулей

В программе следует реализовать следующие модули:

1. Модуль считывания исходной программы и ее разбиение на токены.

Отвечает за работу с файлом, содержащим исходную программу на языке ЛИСПА. Производит считывание указанного файла. Считанный файл разбивается на токены. Производит первичную проверку синтаксиса и семантики.

2. Модуль обработки ошибок и логирования.

Производит логирование действий интерпретатора. Логи хранятся в специальной файле.

Все ошибки можно разделить на предупреждения и критические ошибки. Предупреждения помещаются в лог и дублируются в консоли. При возникновении критической ошибки происходит прерывание выполнения программы. Ошибка фиксируется в логах и дублируется в консоли.

3. Модуль ядра интерпретатора.

Контролирует изменения в регистрах. Производит вторичную проверку кода исполняемой программы. Данная проверка производится непосредственно перед выполнением команды. Выполняет программу на языке ЛИСПА. Результаты работы программы выводит в консоль операционной системы.

4.2.2 Требования к интерфейсам

4.2.2.1 Общие требования

Программа должна работать в консольном режиме. Шрифт и размер текста определяется стандартными настройками системы.

4.2.2.2 Макеты разделов

Макеты отсутствуют, так как интерпретатор запускается в стандартной консоли операционной системы.

Приложение А. РБНФ языка ЛИСПА

```
Prog := S_expr {S_expr}
S_expr := Atom | List;
List := ‘()’ | ‘(‘S_expr, {S_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;
```

VARIABLE – переменная объявленная стандартным образом: может начинаться с буквы или символа «_», далее могут следовать буквы, цифры и символы «_».

Приложение Б. Описание зарезервированных символов и слов

Таблица 1 – Арифметические функции

Символ/слово функции	Описание
+	Производит сложение переданных аргументов.
-	Последовательно вычитает из первого переданного аргумента все последующие аргументы.
*	Производит перемножение всех переданных аргументов.
/	Последовательно вещественно делит первый переданный аргумент на все последующие аргументы.

Таблица 2 – Логические функции

Символ/слово функции	Описание
<; >; <=; >=	Сравнивает первый переданный аргумент со всеми остальными указанной операцией. Возвращает истину, если все сравнения были истины; ложь в противном случае. Пример: (> 5 4 3 2 1). Последовательно сравниваем 5 со всеми остальными числами. Так 5 больше всех остальных чисел, результатом функции будет истина.
=	Сравнивает переданные аргументы на равенство. Возвращает истину, если все переданные аргументы равны между собой; ложь в противном случае.
!	Возвращает обратный результат от переданного логического аргумента. Может быть передан только 1 аргумент.

Таблица 3 – Функции для обработки элементов списка

Символ/слово функции	Описание
car	Возвращает первый элемент списка. Может быть передан только 1 аргумент. Если список пуст, то возвращается NIL.
cdr	Возвращает остаток списка. Может быть передан только 1 аргумент. Результатом работы может являться NIL, если список содержит не более одного элемента. Если список является NIL, то возвращается NIL.
cons	Соединяет переданные аргументы в список. Передается два аргумента. Результатом работы функции является список, в котором первый аргумент является car, второй – cdr. Результатом работы функции является новый список. Данной функции всегда передается два аргумента.
null	Функция для проверки списка на пустоту. Возвращает истину, если список равен nil.

Таблица 4 – Определение функция и переменных

Символ/слово функции	Описание
defun	Служебное слово для объявления пользовательской функции. Синтаксис детально описан в приложении В.
set	Возможно 2 вида поведения: 1.Объявляет переменную, инициализируя ее. 2.Заменяет значение существующей переменной.

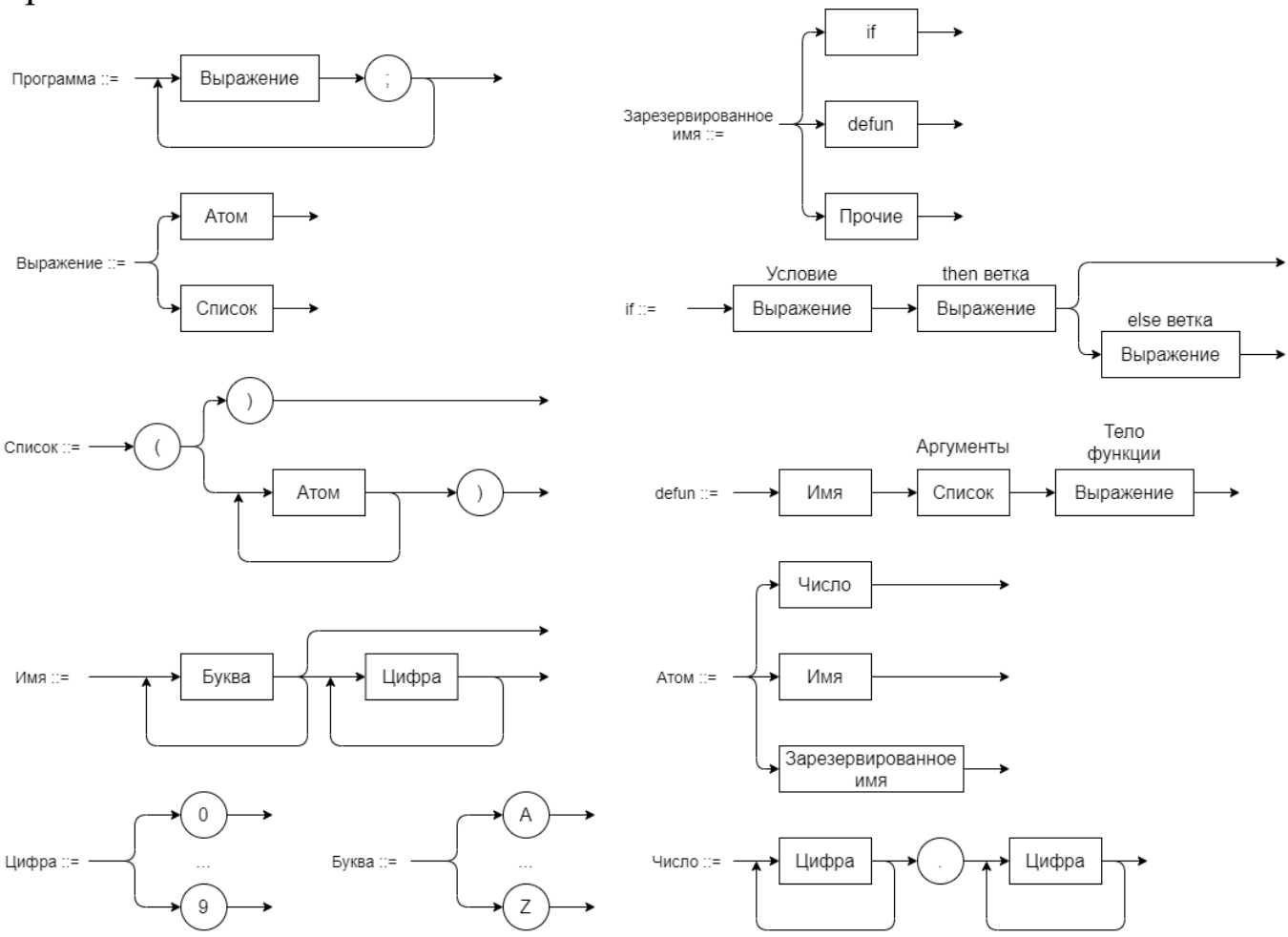
Таблица 5 – Условный оператор

Символ/слово функции	Описание
if	Условный оператор. Аргументами должны являться логические переменные. В случае истинности условия происходит переход на ветку then; в противном случае на ветку else.

Таблица 6 – Зарезервированные слова

Символ/слово функции	Описание
T	Зарезервированная логическая константа равная значению истины.
Nil	Зарезервированная константа. Означает пустой список.

Приложение В. Семантика языка ЛИСПА



Прочие в зарезервированных именах – это функции, для которых синтаксическая диаграмма тривиальна. В эту группы входят функции: car, cdr, cons, null, set, арифметические операции, логические операции.