1.Аннотация дисциплины «**Функциональное и логическое программирование»**

Задачи искусственного интеллекта, проблемы и особенности решения. Современные парадигмы программирования задач искусственного интеллекта. Представление знаний. Методологии программирования, методологии логического и функционального программирования. Процедурные, аппликативные и декларативные языки программирования.

Языки функционального программирования. Язык Lisp, его диалекты. Преимущества и недостатки функционального подхода в программировании.

Основные элементы языка Lisp. Атомы, точечные пары, S-выражения, списки. Атомы и константы. Правила записи идентификаторов. Правила записи списков. Вычислимые S-выражения. Внутреннее представление списков. Поняие функции. Функции в Lisp-е. Селекторы, конструкторы, предикаты. Особенность работы предикатов сравнения. Функции работы со списками. Использование функционалов и рекурсии. Использование параметров при определении функций. Функции более высокого прядка.

Отношение, как способ фиксации знаний. Доказательства. Исчисление высказываний и исчисление предикатов. Кванторные операции. Принцип резолюции. Основные элементы и основные структуры языка Пролог (факты, правила и вопросы). Простейшие логические программы. Понятие домена. Стандартные домены. Структура программы на языке Пролог. Простейший интерпретатор логических программ. Протокол интерпретатора. Виды импликации.

Термы как структуры данных. Функторы и предикатные символы. Использование переменных в фактах, правилах и вопросах. Подстановки и примеры термов. Отождествление и сопоставление. Правила отождествления. Механизм унификации. Конъюнкция и дизъюнкция целей.

Декларативная и процедурная семантика логических программ. Понятие процедуры в логической программе. Иллюстрация различия декларативной и процедурной семантики программ. Механизм возврата. Построение дерева вывода. Стандартные предикаты cat и fail.

Рекурсивные процедуры и их применение. Граничные условия. Порядок утверждений в логической программе и поиск решений. Список. Формы записи списков в Пролог. Работа со списками и рекурсивные процедуры их обработки средствами Пролога. Примеры программ.

2. Проектируемые (планируемые) результаты освоения содержания дисциплины

2.1 Студент должен знать

- способы обобщения, анализа, критического восприятия информации, постановки цели и выбору путей её достижения (ОК-1);
- социальную значимость своей будущей профессии, (ОК-7)
- возможности кооперации с коллегами, работе в коллективе (ОК-13);
- основные законы естественнонаучных дисциплин в профессиональной деятельности, применять методы математического анализа и моделирования, теоретического и экспериментального исследования (ОК-15)
- основные понятия и методы математической логики, теории вычислений и алгоритмов, методы вычислительной математики (ПК-2);
- существующие методы решения поставленной задачи (ПК-4);
- основы информатики и программирования (ПК-11);
- способы описания требуемой функциональности ПО и подготовки технического задания (ПК-14);

- способность формализовать предметную область программного проекта и разработать спецификации для компонентов программного продукта (ПК-15).

- **Студент должен уметь**

- использовать методы и приемы философского анализа проблем (ОК-10);
- применять основы информатики и программирования к проектированию, конструированию и тестированию программных продуктов (ПК-11);
- использовать основные методы и нотации, применяющиеся для проектирования ПО (ПК-17);
- использовать компиляторы и интерпретаторы, средства тестирования и другие средства разработки ПО (ПК-30).

2.3Студент должен иметь навыки

- владения культурой мышления, способностью к обобщению, анализу, критическому восприятию информации, постановке цели и выбору путей её достижения (ОК-1);
- описания математической модели заданной предметной области (ПК-5);
- моделирования, анализа и использования формальных методов конструирования программного обеспечения (ПК-12);
- описания требуемой функциональности ПО и подготовки технического задания (ПК-14).
- проектирования модели данных и выбора структур данных (ПК-18);

описания и анализа используемых алгоритмов (ПК-19)

ВВЕДЕНИЕ

Методологии программирования и языки , их поддерживающие

Можно выделить:

Императивная методология (процедурная);
Объектно-ориентированная методология;
Функциональная методология;
Логическая методология;
Методология параллельного программирования (Многопроцессорная).

Базисом (основой) любой методологии является совокупность основных принципов и методов управления работой техники (аппаратуры)

Исторически первой появилась, и до сих пор используется императивная методология программирования. Программа на императивном языке — это последовательность команд, каждая из которых приводит к изменению состояния вычислителя. Состояние вычислителя — это состояние его памяти. Поэтому базовой командой, использующейся любым императивным языком программирования, является оператор присваивания. Порядок работы вычислителя определяют принципы Фон-Неймана. Дискретное изменение состояния вычислителя определяется последовательностью (порядком) команд программы, а программисту, особенно разыскивая и устраняя ошибки, приходится контролировать изменение состояния его памяти. В больших программах команды группируются в подпрограммы (процедуры и функции) — структурный подход к программированию.

В объектно-ориентированном программировании - базовым понятием является объект. Объект обладает свойствами (данными) и методами (подпрограммами). Принципиальное отличие от императивной методологии - появление объекта, некоторой новой конструкции, объединяющей данные и программы (подпрограммы) и особых способов работы с ними (описание объектов и классов объектов, создание новых классов и объектов, на базе описанных ранее). Использование понятия объект, привело к появлению

более емких описаний и команд программы. С точки зрения же реализации программы, новых методов работы с информацией это не потребовало.

Функциональное (аппликативное) программирование - базовым понятием является функция (математическое понятие); за состоянием памяти следить не нужно - важен результат (возврат), а не процесс (состояние памяти). Операторов и команд нет в принципе.
Логическое программирование - поддерживается декларативными языками программирования. Они основаны на системе правил. В отличие от императивного, порядок действий определяется набором правил и не определяется порядком записи в тексте программы.

LISP

ANSI Common LISP - Пол Грэм
Основы функционального программирования - Л В Городняя
Мир Лиспа, введение в язык и ФП (2 части) - Э Хювёнен, Й Сяпянен

Функциональное программирование ориентировано на символьную обработку данных. Предполагается, что любую информацию можно свести к символьной. Слово «символ» здесь близко к понятию «знак».

Базис Lisp образуют:
атомы, структуры, базовые функции, базовые функционалы.

«Данные»

Вся информация (данные и программы) в Lisp представляется в виде символьных выражений – **S-выражений**. По определению

S-выражение ::= <атом> | <точечная пара>.

Элементарные конструкции «данных»:

Атомы:

- **символы** (идентификаторы) – синтаксически – набор литер (букв и цифр), начинающихся с буквы;
- **специальные символы – {T, Nil}** (используются для обозначения логических констант);
- **самоопределимые атомы** – атомы, по умолчанию связанные со значением: натуральные числа, дробные числа (например 2/3), вещественные числа, строки – последовательность символов, заключенных в двойные апострофы (например “abc”);

Более **сложные конструкции — списки** и **точечные пары** (**структуры**) строятся из унифицированных структур – блоков памяти – бинарных узлов.

Запишем определения:

Точечные пары ::= (<атом>.<атом>) | (<атом>.<точечная пара>) |
(<точечная пара>.<атом>) | (<точечная пара>.<точечная пара>);

Список ::= <пустой список> | <непустой список>, где

<пусой сисок> ::= ( ) | Nil,
<непустой список>::= (<первый элемент> . <хвост>),
<первый элемент> ::= <S-выражение>,
<хвост> ::= <список>.

Синтаксически:

любая структура в Lisp (точечная пара или список) заключается в круглые скобки

(A . B) – точечная пара, (A) - список из одного элемента,

пустой список изображается как **Nil** или **( );**

непустой список по определению может быть изображен:

(A . (B . (C . (D ())))), допустимо изображение списка последовательностью атомов, разделенных пробелами – **(A B C D)**.

Элементы списка могут, в свою очередь, быть списками (любой список заключается в круглые скобки),

например – **(A (B C) (D (E))).** Таким образом, синтаксически наличие скобок является признаком структуры — списка или точечной пары.

Любая непустая **структура Lisp** в памяти представляется списковой ячейкой, хранящей два указателя на голову (первый элемент) и хвост — все остальное.