Multi-paradigm programming environments is when the systems expand the capabilities of the shells in various directions. One of the example of the multi-paradigm programming languages is Pascal. Pascal are also known as the typical block-structured algorithmic language. Due to many cases, it may be implemented very efficiently on the conventional computer hardware. Conventional programming language such as Pascal are aimed at creation of simple language that will be suitable for scientific computation and construction of system software.

Current Expert System programming language vs Very high-level languages

|  |  |  |
| --- | --- | --- |
| **Language Features** | **High Level** | **Very High Level** |
| Data structure | Lists | collections |
| Decomposition | Selectors | Pattern match |
| Database | Property list | Association assertions |
| Program specification | functions | Pattern-action rules |
| Control structure | Recursion | Non-determinism |

Pascal does not have the programming features as mentioned above but it does have some features for the design and also the constructions of the expert system. Example of typical high level language is LISP and PROLOG, in some sense may regard as something close to very high level Expert System language.

The main drawback of Pascal like languages will be their inflexible control structures besides they have very limited or almost none inference power.

Compare LISP with Prolog

**DATA STRUCTURE**

LISP : list

List can be explained as 1D array with the following properties which is having a dynamically alterable length of array, the elements may have heterogenous types, composed by means of constructor functions to carry elements and decomposed by selectors functions rather that position Index

Prolog: Collections

In a very high level languages, other data structures other than list also available

**DECOMPOSITION METHOD**

Lisp: By function CAR and CDR

Very simple mainly by the means of functions

Prolog: pattern matching

AI language usually have one critical property which is pattern matching because it will be needed when there is explicit structure decomposition that will also be used for associative retrieval and pattern action rules.

**DATA BASE**

Lisp: property lists

Property lists used to characterize a sysmbol with arbitrary number of properties. The property list for the symbol S can be implemented as the even-length list associated with S by hash-coding.

Prolog: Associative assertions

For a very high ES language, the facts about the symbol can be written as assertions. The set of assertions will have the associative data base that will allow access with more ease and natural. They can also used to deduce new facts.

**PROGRAM SPECIFICATION**

**Lisp:** functinos

In lisp, function will be the only language concepts on which everything else must be build and there are three kind of function in LISP which is transformation functions, predicate and special functions.

Transformation functions are used to perform expressions into other expressions. Predicate is to test the expresions for their properties. Special function is used to evaluate expressions in the non-standard ways

**CONTROL STRUCTURE**

LISP: recursion

Most complex programs are written using recursion and it is very important in programming

PROLOG : non-determinism

Most important feature of high level ES language. Used to abstract points within unnecessary details and is one of the very few ways to cope with complex system. The existence quantifier of the logic non-determinism will permit the abstraction from the explicit disjuncts only one of which holds true, non-determinism control the permits an abstraction from a set of computation where only one of which need to succeed.

PROLOG is obviously better than LISP

Pascal is simple to write in program but no always a good choice to implement ES.

LISP and prolog are suitable languages to write in when building a ES but only a well trained programmer can build the system. LISP and PROLOG are both less famous compared to Pascal because one of the factor is the availability of the hardware. LISP and prolog were not available for mainframe until recently.

|  |  |  |  |
| --- | --- | --- | --- |
| property | Lisp | prolog | pascal |
| List |  |  |  |
| Decomposability |  |  |  |
| Flexible control structure |  |  |  |
| Interactiveness |  |  |  |
| Efficiency |  |  |  |
| Late binding time patter matching for data and control |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |