Principles of Programming Languages

A Report for Common Lisp Language

Team members:

42110161, Ansar Ausama A. A. Yousif

42110142, Rawan El-Sayed

42110099, Reem Ayman

-Developing history of Common Lisp:

Common Lisp (CL) is derived from the Lisp programming language. Work began on CL in 1981, after the participation of the ARPA director, *Bob Engelmore*, to develop the Lisp dialect.

Common Lisp was published in the ANSI standard document. Much of the language design was implemented and initialized via e-mail in 1982. Guy L. Steele Jr. gave the first overview of Common Lisp at the 1982 ACM Symposium on “LISP and functional programming”.

The first language documentation was published in 1984 as “Common Lisp the Language” (known as CLtL1), first edition. A second edition, known as CLtL2 published in 1990, included many changes to the language that were made during the ANSI Common Lisp standardization process, such as: the extended loop syntax, the Common Lisp object system and the conditional system for error handling, but the second does not describe the final ANSI common lisp standard and is, therefore, not according to ANSI Common Lisp. The final standard was finally published in 1994 and since then, no updates to the standard have been published, but many additions and improvements have been added

to CL; for example: Unicode, Concurrency, CLOS-based IO have been provided by implementations and libraries, and developed as a unified and improved language for Maclisp. Common Lisp sought to unify and extend the features of Maclisp dialects.

Common Lisp is not an application, but rather a general-purpose, multi-paradigm programming language. A dynamic programming language that facilitates the development of developmental and incremental programs. Incremental development is often done interactively without interrupting the application while it is running.

Some common information about the common language is that:

a. It is first from the Lisp family.
b. It is a model (general, procedural, functional, reflective, meta, object-oriented).
c. It was designed by Scott Fahlman, Richard P. Gabriel, David A. Moon, Kent Pitman, Guy Steele, Dan Weinreb.
d. The developer committee (ANSI X3J13 committee), and it appeared for the first time in 39 years,1984 and since 29 years for ANSI Common Lisp.
e. Type discipline (strong dynamic)

f. Scope (lexical, optionally dynamic)
g. Filename extensions (.lisp, .lsp, .l, .cl, .fasl), OS across platforms, of course have a website “common-lisp.net”
h. Main application: Allegro CL, ABCL, Clasp, CLISP, Clozure CL, CMUCL, ECL, GCL, LispWorks, Scieneer CL, SBCL, Symbolics Common Lisp.
i. Dialects: CLtL1, CLtL2, ANSI Common Lisp.
j. Affected by: Lisp, Lisp Machine Lisp, Maclisp, Scheme, Interlisp.

There are 24 built-in Common Lisp special functions, which are: Block Compiler-LET, Declare, Eval-when, FLET, Function, Go, If, Labels, LET, LET*, Macrolet, Multiple-Value-Call, Mutiple-Value-PROG1, PROGN, PROGV, Quote, Return-From, SETQ, TAGBODY, The, Throw, and, UNWIN D-Protect.

There are many working environments supported by Common Lisp, but the most important and most important of them is Portacle, which is a complete working environment for Common Lisp. It is a multi-platform programming language and can be taken with you wherever you are, for ease of dealing with. It is very suitable for beginners who do not want to spend time preparing everything. It can be run on any operating system, like Windows, OS X, and Linux.

-Domain and Category of Common Lisp:

Common Lisp was mainly influenced by were lisp machine lisp, Maclisp, NIL, S-1 Lisp, spice lisp, and Scheme. It has quite a lot of Lisp machine Lisp. But it's known to be Maclisp inheritor it also provides partial backwards compatibility with it. So being inspired by all of these languages it got a lot of features as it has many built-in data types, functions, macros and other language elements, and an object system having its name and mainly borrowing certain features from Scheme such as lexical scoping and lexical closure. Common Lisp is counted as extensible through its standard features, for example: - code transformations and input parsers for characters. Common Lisp was made to be effectively implementable on any computer weather it's personal or for workstation as well as different platforms and operating systems.

So, since Lisp was originally created as a practical Mathematical notation for Computer Programs then it was used in Ai researches.

Although Common Lisp may give the sensations of being an extension language. But there are now 7M+ LOC in Common Lisp for that application. Actually, common Lisp is not only the extension language.

Common Lisp also got used in commercial applications including Grammarly, which uses AI to analyze text and suggest improvements, and Boeing, which uses a server written in the Lisp variant Common lisp also get used in different fields

ranging from dynamic web applications and Natural Language Processing to systems engineering infrastructure in container clusters (DNS, dynamic load balancer). And the list of common lisp's implementations goes on, thanks to its features.

-Goals & Concerns:

Just like any language Common lisp goal was to maintain a very flexible programming language. Providing a lot of features like error handling and allowing a wide range of programming paradigms, including functional, object-oriented, and procedural programming.

Common lisp was made to be a highly expressive language, meaning that it allows writing code in a very compact way to write and it's also easy to read.

Also, Common Lisp was designed to be able to work on different platforms giving it portability, which means that code written in Common Lisp can be run on a wide range of systems.

They also gave Common Lisp a high-performance meaning that the language is designed to be very efficient allowing programmers to make fly changes and experiment with the language. It has several advanced optimization techniques allowing Common lisp to be used for large-scale applications.

So, at the end Common Lisp as a versatile and powerful programming language it is well-suited for a wide range of programming tasks and programming applications.

-Contribution:

Common Lisp has made a number of great assistances to computer science and programming filed, including:

1) **Dynamic programming**: Common Lisp was first to introduce the concept of dynamic programming, allowing programs to modify their own code at runtime. Which was proven to be very useful in a wide range of applications, including machine learning, natural language processing, and expert systems.

2) **Object-oriented programming**: Common Lisp was one of the first to introduce object-oriented programming concepts, such as classes, inheritance, and polymorphism. Which had a great impact on the development of object-oriented programming languages such as Java and Python.

3) **Functional programming**: Common Lisp was first to introduce a number of functional programming concepts, such as higher-order functions, closures, and recursion. These concepts have posteriorly been used in a wide range of programming languages.

4) **Macros**: it's widely known for common lisp its macro system, which allows programmers to define their own language constructs. This has made it possible to create domain-specific languages (DSLs) that are tailored to specific tasks and applications.

To finish this, Common Lisp has taken a great part in upgrading the field of computer science and programming, and it left a

mark that can be seen in many modern programming languages and technologies.

-What is new in Common Lisp:

There are many achievements, including the development of SBCL, which we can now call “a shared library”. A new 3D graphics project Kons-9. The project builds on the strengths of the Common Lisp, CLOG Common GUI. It's like a GUI framework for creating web applications. It provides a lightweight abstraction for creating fully dynamic web applications.

-Overview of Common Lisp:

Common Lisp is a high-level, dynamic programming language that was developed for use in artificial intelligence and other complex applications. It is a dialect of Lisp, which is a family of programming languages known for their expressive syntax and powerful metaprogramming capabilities. Common Lisp is designed to be a very flexible and extensible language, allowing programmers to create their own data types, operators, and control structures.

One of the key features of Common Lisp is its support for macros. Macros are a way of extending the language by

defining new syntactic constructs that are translated into standard Lisp code at compile time. This allows programmers to create domain-specific languages that are optimized for their application domain.

Here's an example of a simple Common Lisp program that adds two number:

```
(defun add (x y)
    (+ x y))
(add 5 7) ;returns 12
```

This program defines (*defun*) a function called (*add*), which takes two arguments (*x*) and (*y*) and returns their sum. Common Lisp uses the prefix typing (*+ x y*) which means x + y. The program then calls the (*add*) function with the arguments 5 and 7, which returns 12.

-Comments:

There are two ways to write comments:

1) Line comments: Line comments start with a semicolon (;) and continue to the end of the line. Anything following the semicolon on the line is considered a comment and is ignored by the Lisp compiler or interpreter. The previous example shows the usage of this type.

2) Block comments: Block comments start with (#|) and end with (|#). Anything between them is considered a comment and is ignored by the Lisp compiler or interpreter. Here's an example of a block comment:

```
(defun cube (x)
    #|
    This function computes the cube of a number.
    It does so by multiplying the number by itself three times.
    |#
    (* x x x))
```

-Major Commands in Common Lisp:

Some of the major commands in Common Lisp with a brief explanation of what they do:

1) (*defun*): Defines a new function with a given name and argument list.

2) (*let*): Binds a set of variables to values within a given scope.

3) (*if*): Implements conditional branching based on a Boolean test.

4) (*cond*): Implements multi-way conditional branching based on a series of tests.

5) (*case*): Implements conditional branching based on a set of predefined values.

6) (*loop*): Provides a powerful looping construct with a wide range of options.

7) (*setf*): Assigns a new value to a variable or other data structure.

8) (*setq*): Assigns a new value to a global variable.

9) (*format*): Provides formatted output to a stream or string.

10) (*read*): Reads input from a stream or string into a Lisp data structure.

11) (*write*): Writes a Lisp data structure to a stream or string.

-Examples on some of the previous commands:

1) The *if* function:

```
(defun absolute-value (number)
    "Return the absolute value of a number."
    (if (< number 0)
        (- number)
        number))
```

In this example, the (*if*) command is used to implement a simple function that returns the absolute value of a number. The function takes a single argument, (*number*), and uses the (*if*) command to check whether the number is less than zero. If it is, the function returns the negation of the number (i.e., its absolute value). If not, the function returns the number itself.

Note that the last line is considered as (*else*) in C++.

Here's an example of how this function could be used:

```
(absolute-value -5)  ; returns 5
(absolute-value 5)  ; returns 5
```

2) The (*cond*) function:

```
(defun compare (a b)
    "Compare two numbers and return a string indicating their relationship."
    (cond ((< a b) "a is less than b")
        ((> a b) "a is greater than b")
        (t "a and b are equal")))
```

In this example, the (*cond*) command is used to implement a function that compares two numbers and returns a string indicating their relationship. The function takes two arguments, (*a*) and (*b*), and uses (*cond*) to check whether (*a*) is less than,

greater than, or equal to (*b*). The first condition that evaluates to true will cause the corresponding string to be returned.

Here's an example of how this function could be used:

```
(compare 2 3) ; returns "a is less than b"
(compare 3 2) ; returns "a is greater than b"
(compare 2 2) ; returns "a and b are equal"
```

3) The (*loop*) function:

```
(defun sum-of-squares (n)
    "Compute the sum of the squares of the first N positive integers."
    (loop for i from 1 to n
        summing (* i i)))
```

In this example, the (*loop*) command is used to implement a function that computes the sum of the squares of the first (*n*) positive integers. The function takes a single argument, (*n*), which is expected to be a positive integer. The (*loop*) command is used to iterate over the integers from (*1*) to (*n*), squaring each integer and summing up the results.

-Comments and evaluation:

Common Lisp is a powerful and flexible programming language that has been widely used in industry, academia, and research for decades. Its key strengths include:

- **Expressive syntax**: Common Lisp's syntax is designed to be highly expressive, allowing for concise and readable code.

- **Dynamic programming**: Common Lisp's dynamic typing and metaprogramming capabilities make it a powerful tool for rapid prototyping, experimentation, and exploration.

- **Powerful standard library**: Common Lisp includes a rich set of built-in functions and data types, as well as a comprehensive standard library that provides support for a wide range of tasks.

- **Extensibility**: Common Lisp is highly extensible, with support for macros, reader macros, and other metaprogramming features that allow developers to customize and extend the language to suit their needs.

- **CLOS**: Common Lisp's object system, CLOS, is highly flexible and extensible, allowing for a wide range of OOP programming styles.

However, Common Lisp also has some weaknesses, including:

- **Complexity**: Common Lisp's power and flexibility come at the cost of complexity, which can make it difficult for beginners to learn and use effectively.

- **Performance**: Common Lisp is generally slower than low-level languages like C or C++, which can make it less suitable for performance-critical applications.

- **Tooling and ecosystem**: While Common Lisp has a rich standard library and many third-party libraries, its tooling and ecosystem are not as well developed as some other languages. This can make it harder to find support, documentation, and tools for specific tasks.

Overall, Common Lisp is a powerful and flexible language that is well-suited for a wide range of tasks, from scripting to AI programming. Its expressive syntax, dynamic programming features, and powerful standard library make it a popular choice for many developers. However, its complexity and performance limitations may make it less suitable for some applications, and its tooling and ecosystem may require more effort to work with.

References:

https://portacle.github.io/ https://en.wikipedia.org/wiki/Common_Lisp https://www.bing.com/ck/a?!&&p=5d159cec2f11649aJmltdHM9MTY4NDQ1NDQwMCZpZ3VpZD0zMmU2ZjI4ZS0yZWY0LTZlZjYtMzA3MC1lMGMyMmY4ODZmNjQmaW5zaWQ9NTQ1MA&ptn=3&hsh=3&fclid=32e6f28e-2ef4-6ef6-3070-e0c22f886f64&psq=common+lisp+in+arabic+wikipedia&u=a1aHR0cHM6Ly93d3cud2lraWRhdGEub3JnL3dpa2kvUTg0OTE0Ng&ntb=1 https://riptutorial.com/common-lisp/example/11553/global-special-variables-are-special-everywhere https://lisp-lang.org/style-guide/ https://www.bing.com/ck/a?!&&p=fc4e7ab253df4fcaJmltdHM9MTY4NDQ1NDQwMCZpZ3VpZD0zMmU2ZjI4ZS0yZWY0LTZlZjYtMzA3MC1lMGMyMmY4ODZmNjQmaW5zaWQ9NTE5Mw&ptn=3&hsh=3&fclid=32e6f28e-2ef4-6ef6-3070-e0c22f886f64&psq=what+is+special+or+new+in+the+common+lisp&u=a1aHR0cHM6Ly9zdGFja292ZXJmbG93LmNvbS9xdWVzdGlvbnMvNjg4Njg5NTgvYXMtb2YtMjAyMS13aGF0LWFyZS1hbGwtY29tbW9uLWxpc3Atc3BlY2lhbC1mdW5jdGlvbnM&ntb=1 These Years in Common Lisp: 2022 in review - Lisp journey (lisp-journey.gitlab.io)

http://www.lispworks.com/products/lisp-overview.html

https://en.wikipedia.org/wiki/Common_Lisp_Object_System

https://www.cs.ucdavis.edu/~vemuri/classes/ecs170/lispintro.htm

https://en.wikipedia.org/wiki/Functional_programming

https://www.tutorialspoint.com/lisp/lisp_basic_syntax.htm

https://learnxinyminutes.com/docs/common-lisp/

https://gigamonkeys.com/book/syntax-and-semantics.html

https://chatgptonline.ai/chat/

https://poe.com/ChatGPT

Lisp History (GNU Emacs Lisp Reference Manual)

Companies using Lisp (common-lisp.net)

Closure (computer programming) - Wikipedia

Category:Common Lisp (programming language) software - Wikipedia

https://stackshare.io/common-lisp -:~:text=As%20one%20of%20the%20earliest%20programming%20languages%2C%20Lisp,in%20the%20Languages%20category%20of%20a%20tech%20stack.

https://www.cliki.net/Common+Lisp+implementation

https://stackoverflow.com/questions/4164336/does-common-lisp-has-great-legacy-is-it-better-to-learn-common-lisp-or-a-more -:~:text=Common%20Lisp%27s%20design%20goals%20a%20decade%20later%20were,libraries%20or%20programs%20would%20not%20start%20at%20zero.

https://lisp-journey.gitlab.io/blog/state-of-the-common-lisp-ecosystem-2020/
https://www.britannica.com/technology/artificial-intelligence-programming-language

https://project-mage.org/why-common-lisp

https://lispcookbook.github.io/cl-cookbook/cl21.html

https://stackoverflow.com/questions/794450/what-is-lisp-used-for-today-and-where-do-you-think-its-going

https://en.wikipedia.org/wiki/Lisp_(programming_language)