

---

# 1

---

# Introdução

---

Este capítulo mostra detalhes e características básicas de operação da linguagem de programação LISP. São apresentadas informações sobre a obtenção, instalação e uso dos programas CLISP e SBCL, além de seus respectivos ambientes interativos de utilização. É demonstrado de forma simples o uso de algumas operações de cálculos matemáticos básicos e apresentação de cadeias de caracteres.

## 1.1 LINGUAGEM LISP

---

A linguagem LISP sigla de **LISt Processing** (processamento de listas) foi desenvolvida pelo matemático e cientista da computação John McCarthy no verão de 1956 durante a realização do projeto de pesquisa sobre Inteligência Artificial no Dartmouth College, do qual era seu líder tendo sido inicialmente implementada em 1958.

LISP foi desenvolvida para ser usada como linguagem de processamento de listas algébricas para computação simbólica usada em sistemas de Inteligência Artificial. Um ano após seu lançamento é apresentada a primeira versão da linguagem chamada LISP 1, depois em 1962 é apresentada LISP 1.5 e em 1964 é apresentada LISP 2 que apesar de diversas melhorias sobre LISP 1.5 não foi amplamente usada. Aproximadamente 22 anos após o lançamento de LISP 1.5 é lançada em 1984 a versão COMMON LISP (ou CL tema desta obra) e em 1994 é lançada o padrão ANSI COMMON LISP identificado como ANSI X3.226-1994 e atualmente o padrão ANSI INCITS 226-1994, tendo sua documentação um total de 1153 páginas.

LISP é a segunda linguagem de programação de alto nível mais antiga em uso até os dias atuais, depois de FORTRAN (**F**ormula **T**RANslator) lançada em 1954.

LISP influenciou o desenvolvimento de diversas outras linguagens, tornando-se na verdade representante de uma família de diversas linguagens de programação, destacando-se: LISP 1.5 (1955); MacLISP (1965); LOGO (1968); InterLISP (1970); SmallTalk (1971); ZetaLISP, Scheme e NIL de **New Implementation of LISP** (1975); COMMON LISP (1980); CCL de **Closure CL** (1984), não confundir **Closure** com **Clojure**; Emacs LISP, AutoLISP e AutoLISP da série de programas AutoCAD (1985); CLOS de **Common Lisp Object System** (1989) incorporada a linguagem CL; EuLISP, Racket e ISLISP (1990); SBCL (1999); ACL2 (2005); Clojure (2005) e da brasileira Run (2017), entre outras aqui não citadas.

Entre os períodos de 1956 e 1984 ocorreram o lançamento de diversos dialetos da linguagem LISP que se tornaram incompatíveis entre si, apesar da versão LISP 1.5 ter sido a mais amplamente aceita. No início da década de 1970 dois eram os dialetos LISP mais populares: MACLISP desenvolvido pelo projeto MAC (**Mathematics and Computation**) do MIT (**Massachusetts Institute of Technology**) em 1965 e INTERLISP desenvolvida pela empresa BBN Technologies em 1967, posteriormente chamado de BBN LIST.

Tanto MACLISP como INTERLISP estenderam LISP 1.5 com funcionalidades que acabaram mais tarde influenciando o desenvolvimento do padrão CL.

O primeiro esforço de padronização da linguagem ocorreu em 1969, mas um efetivo trabalho só veio em 1984 com a publicação do documento "**Common Lisp: the Language**" de Guy L. Steele que formou a base para o padrão CL, tendo este trabalho sido agenciado pela agência DARPA (**Defense Advanced Research Projects Agency**).

O passo seguinte ocorreu em 1990 quando do lançamento da CLOS (**Common Lisp Object System**) com suporte a orientação a objetos e em 1994 ocorre a publicação da segunda edição do documento "**Common Lisp: the Language**".

Embora os padrões ANSI COMMON LISP e COMMON LISP possuam diferenças a especificação ANSI COMMON LISP reconhece o padrão COMMON LISP publicado em 1990 e 1994.

Em 1987 é iniciado o trabalho de padronização ISO (**International Organization for Standardization**) formado pelo grupo SC22 WG16 (LISP WG) que estabeleceu um padrão mínimo e compacto para a linguagem LISP diferentemente do padrão adotado pelo CL.

## 1.2 OBTEÇÃO E INSTALAÇÃO DO CLISP

O programa CLISP (de **COMMON LISP**) inclui para uso um interpretador, um compilador de **bytecode**, um depurador e os complementos CLOS e MOP (**Meta Object Protocol**), além de possibilitar o uso de expressões regulares POSIX e Perl entre outros recursos. CLISP pode ser executado em diversos sistemas operacionais, tais como: Linux, FreeBSD, NetBSD, OpenBSD, Solaris, Tru64, HP-UX, BeOS, IRIX, AIX, Mac OS X e Windows necessitando apenas de 4 MB (**megabyte**) de memória RAM (**Random Access Memory**), tendo sido criado por Bruno Haible e Michael Stoll inicialmente em 1987 na Alemanha. CLISP é um programa de uso livre sob a Licença Pública Geral GNU. O único inconveniente atualmente é o fato de parecer o projeto congelado, pois não a atualização da ferramenta desde 23 de abril de 2010.

Para obter informações diversas sobre o programa CLISP acesse o sítio oficial do projeto <http://clisp.org> como mostra a figura 1.1.

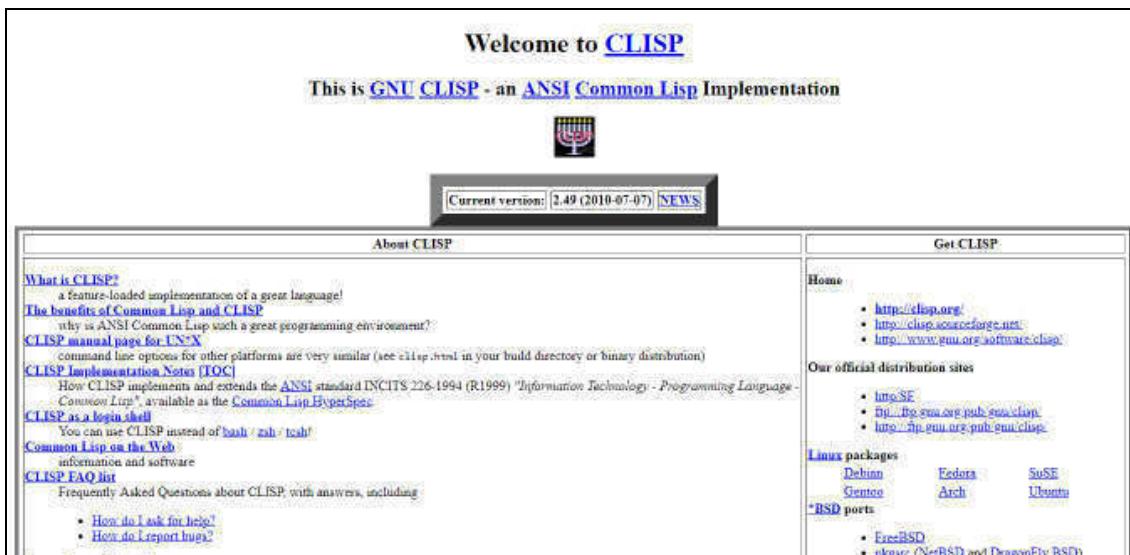


Figura 1.1 - Tela da página do projeto CLISP.

Para obter cópia do programa é necessário fazer acesso fora do sítio oficial. Assim sendo, acesse o sítio <https://sourceforge.net/projects/clisp/>, como apresenta a figura 1.2.

Na tela apresenta ação com um clique do ponteiro do **mouse** o botão verde denominado Download, aguarde o arquivo de programa `clisp-2.49-win32-mingw-big.exe` ser copiado em seu sistema, normalmente para a pasta de Downloads.

Após a cópia do programa `clisp-2.49-win32-mingw-big.exe` faça sua execução a partir de seu acionamento com um duplo clique do ponteiro do **mouse**. Se apresentada a caixa de diálogo de controle de acesso e usuário, confirme com o acionamento do botão **Sim**.

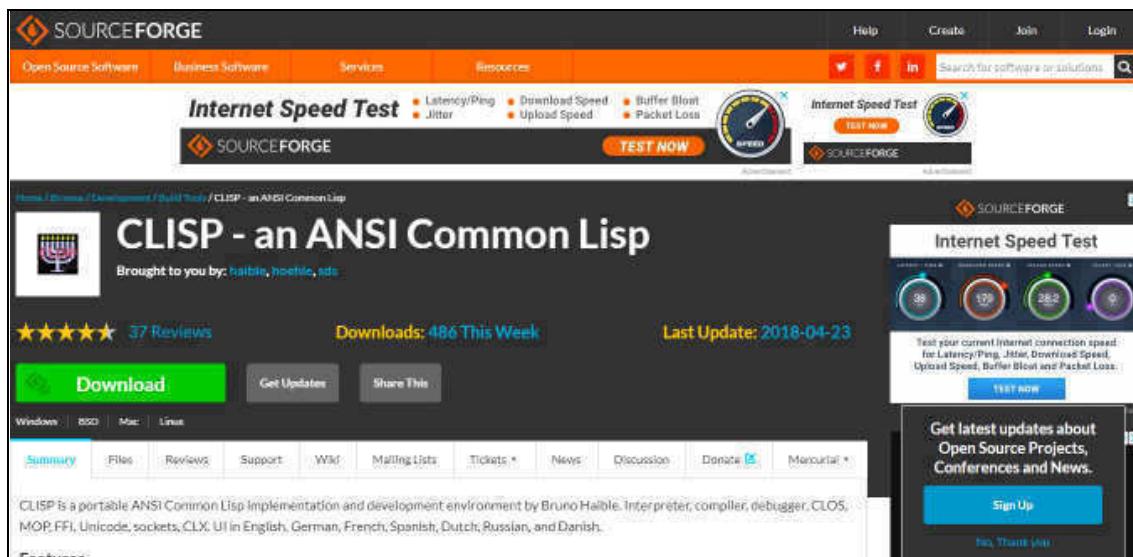


Figura 1.2 - Tela da página do projeto SourdeForge.

Em seguida é apresentada a caixa de diálogo GNU CLISP 2.49 Setup, como mostra a figura 1.3.

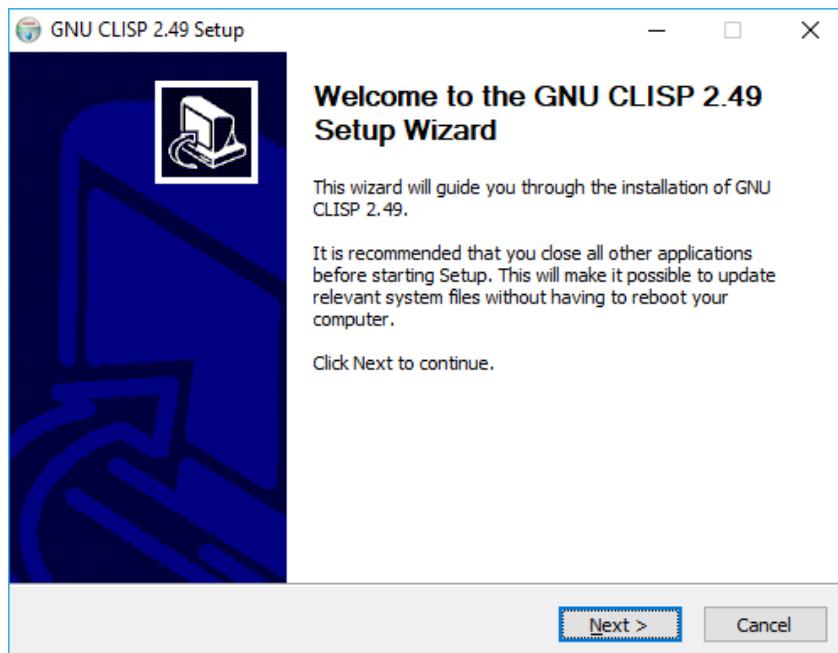


Figura 1.3 - Caixa de diálogo: GNU CLISP 2.49 Setup.

Acione o botão **Next** e será apresentada a tela License Agreement, como mostra a figura 1.4, após proceder a leitura e concordando com os termos acione o botão **I**

Agree para que o processo de instalação seja continuado. Caso selecione o botão Cancel o processo de instalação é interrompido.

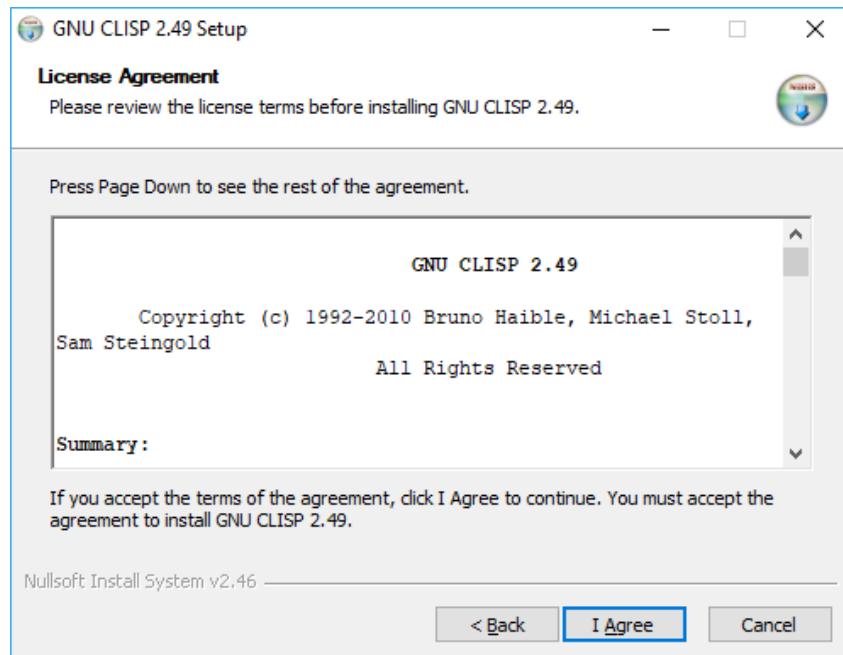


Figura 1.4 - Caixa de diálogo: License Agreement.

Após aceitar os termos do contrato e acionar o botão I Agree é apresentada a tela Choose Components como apresenta a figura 1.5. Mantenha as opções apresentadas e acione na sequência o botão Next para a próxima etapa.

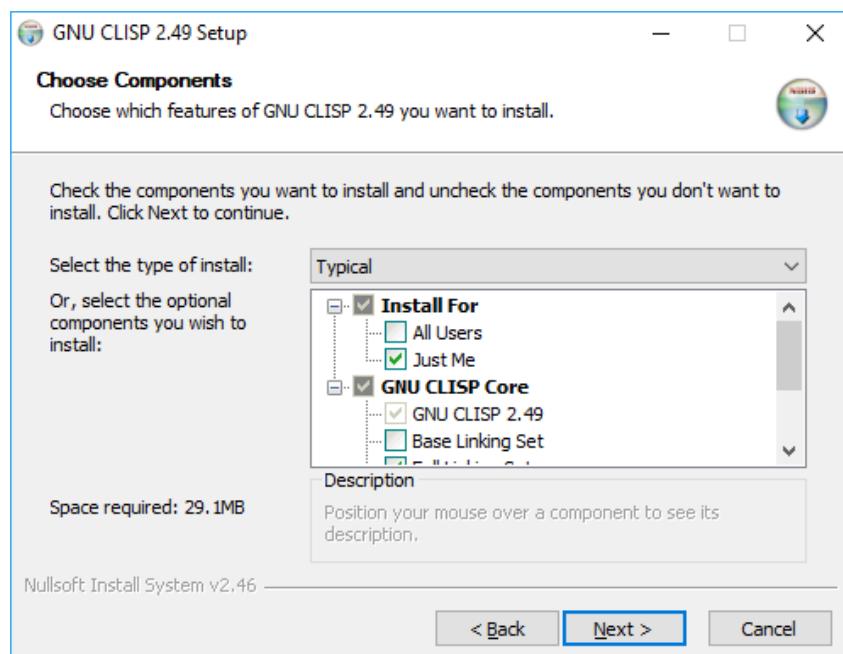


Figura 1.5 - Caixa de diálogo: Choose Components.

Na tela Choose Install Location, como indica a figura 1.6, mantenha a indicação de local apresentado e acione o botão Next.

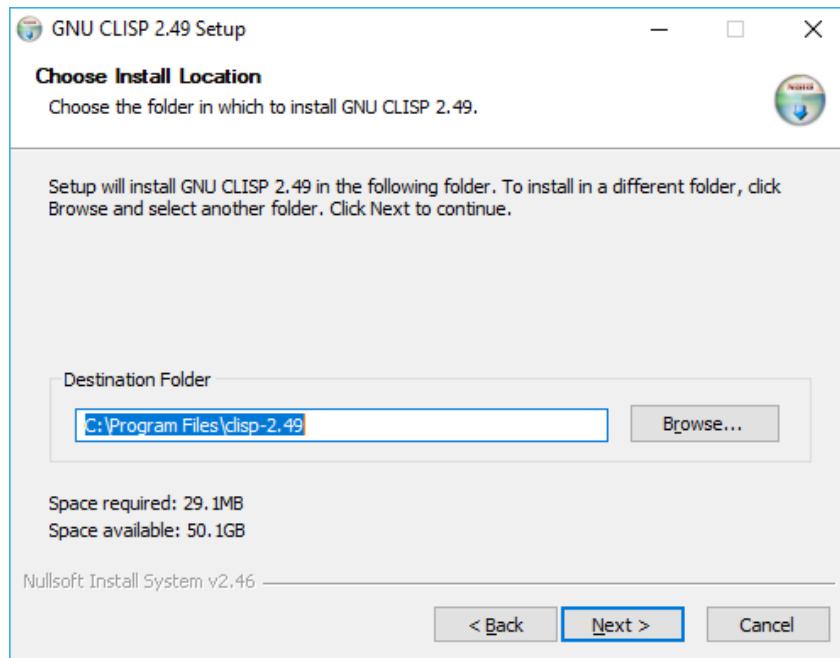


Figura 1.6 - Caixa de diálogo: tela Choose Install Location.

Será então apresentada a tela Choose Start Menu Folder onde poderá ser selecionado a pasta de gravação do programa, como mostra a figura 1.7. Nesta etapa mantenha a informação apresentada e acione simplesmente o botão Install para que o processo de instalação seja iniciado.

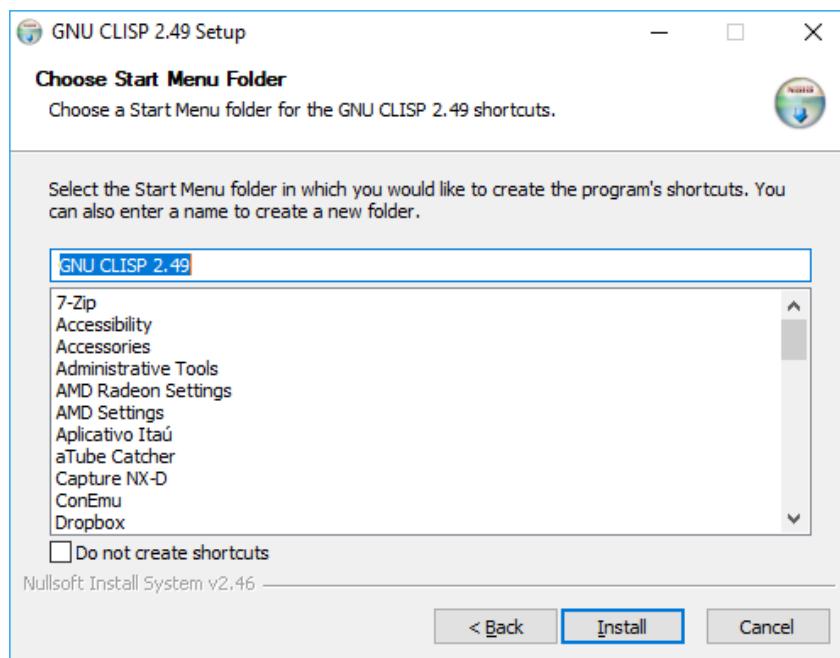
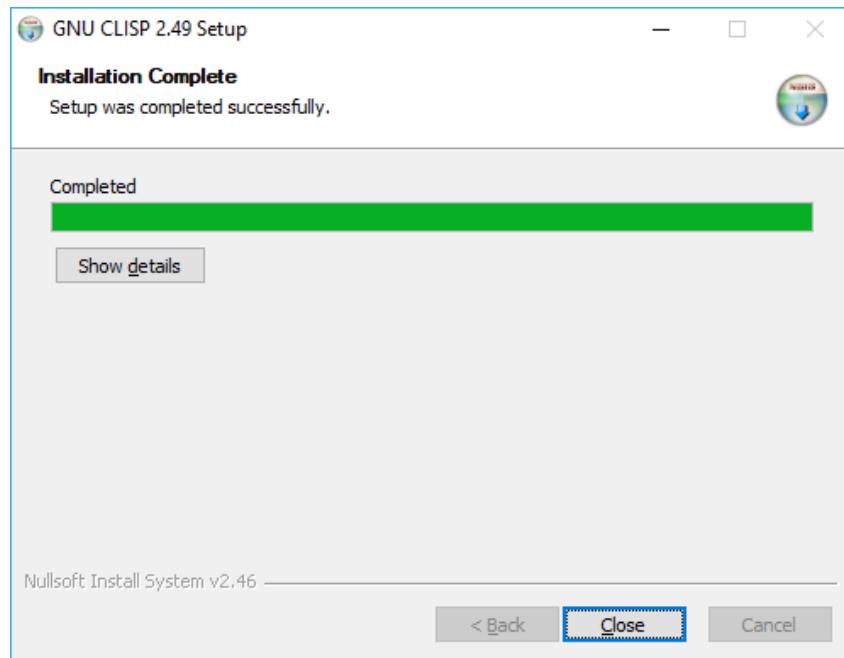


Figura 1.7 - Caixa de diálogo: tela Choose Start Menu Folder.

Aguarde o término da instalação e ao final deste processo acione o botão Close da tela Installation Complete como indica a figura 1.8.



*Figura 1.8 - Caixa de diálogo: tela Installation Complete.*

Após a conclusão da instalação você notará a criação de um ícone de atalho para acesso chamado GNU CLISP 2.49 na área de trabalho do sistema. Assim sendo, acione com um duplo clique do ponteiro do **mouse** o ícone indicado e veja a tela do ambiente de programação LISP apresentado como mostra a figura 1.9.

```
GNU CLISP 2.49
i i i i i i      00000   0       0000000  00000  00000
I I I I I I I I    8     8   8       8     8   0   8   8
I \ `+' / I       8     8       8     8       8   8
\ `--+` /          8     8       8       00000  80000
\ `--+` /          8     8       8       8   8
|   |   |           8     0   8       8     0   8   8
-----+----- 00000   8000000  0008000  00000   8

Welcome to GNU CLISP 2.49 (2010-07-07) <http://clisp.cons.org/>

Copyright (c) Bruno Haible, Michael Stoll 1992, 1993
Copyright (c) Bruno Haible, Marcus Daniels 1994-1997
Copyright (c) Bruno Haible, Pierpaolo Bernardi, Sam Steingold 1998
Copyright (c) Bruno Haible, Sam Steingold 1999-2000
Copyright (c) Sam Steingold, Bruno Haible 2001-2010

Type :h and hit Enter for context help.

[1]> -
```

Figura 1.9 - Tela do ambiente CLISP a partir do ícone da área de trabalho.

Há duas outras formas de fazer uso do ambiente CLISP no Windows. A partir das teclas <Win> + <r>, que apresenta a figura 1.10, informe no campo Abrir da caixa de diálogo Executar a chamada de um dos programas de processamento de comandos CMD ou POWERSHELL e acione o botão OK.

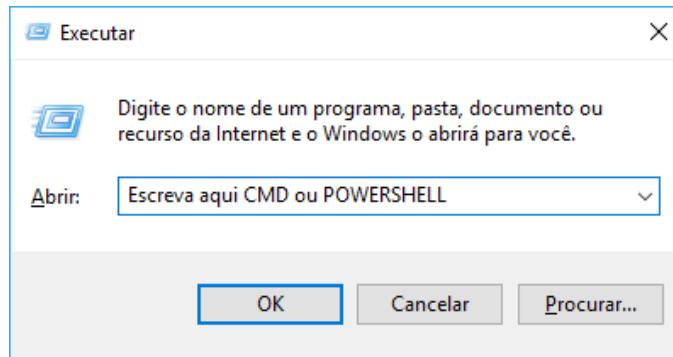


Figura 1.10 - Caixa de diálogo Executar.

Assim que o ***prompt*** do processador de comandos dos programas CMD/POWERSHELL for apresentado efetue a execução do comando CLISP e acione a tecla <Enter>.

Se efetuada a execução do ambiente CMD a apresentação da tela de operação do ambiente CLISP assemelha-se a imagem da figura 1.9 como mostra a figura 1.11.

 A screenshot of a Windows command-line window titled 'Processador de comandos do Windows - clisp'. The window shows the output of the 'clisp' command. It starts with the Windows copyright notice, followed by the command 'C:\Windows\System32>clisp'. The screen then displays a 10x10 grid of characters ('i', ' ', '8', 'o', etc.) that form a stylized logo. Below this, the text 'Welcome to GNU CLISP 2.49 (2010-07-07) <http://clisp.cons.org/>' is shown, followed by a series of copyright notices for Bruno Haible, Michael Stoll, Marcus Daniels, Pierpaolo Bernardi, and Sam Steingold from 1993 to 2010. The prompt '[1]>' is at the bottom left.
 

```

Processador de comandos do Windows - clisp
Microsoft Windows [versão 10.0.17134.590]
(c) 2018 Microsoft Corporation. Todos os direitos reservados.

C:\Windows\System32>clisp
i i i i i   ooooo   o       ooooooo   ooooo   ooooo
I I I I I I   8   8       8   8   o   8   8
I \ '+' / I   8   8       8   8       8   8
\ '-' /      8   8       8   ooooo   8oooo
 \_ | _      8   8       8   8       8   8
      |      8   o   8       8   o   8   8
-----+----- ooooo   8oooooo   ooo8ooo   ooooo   8

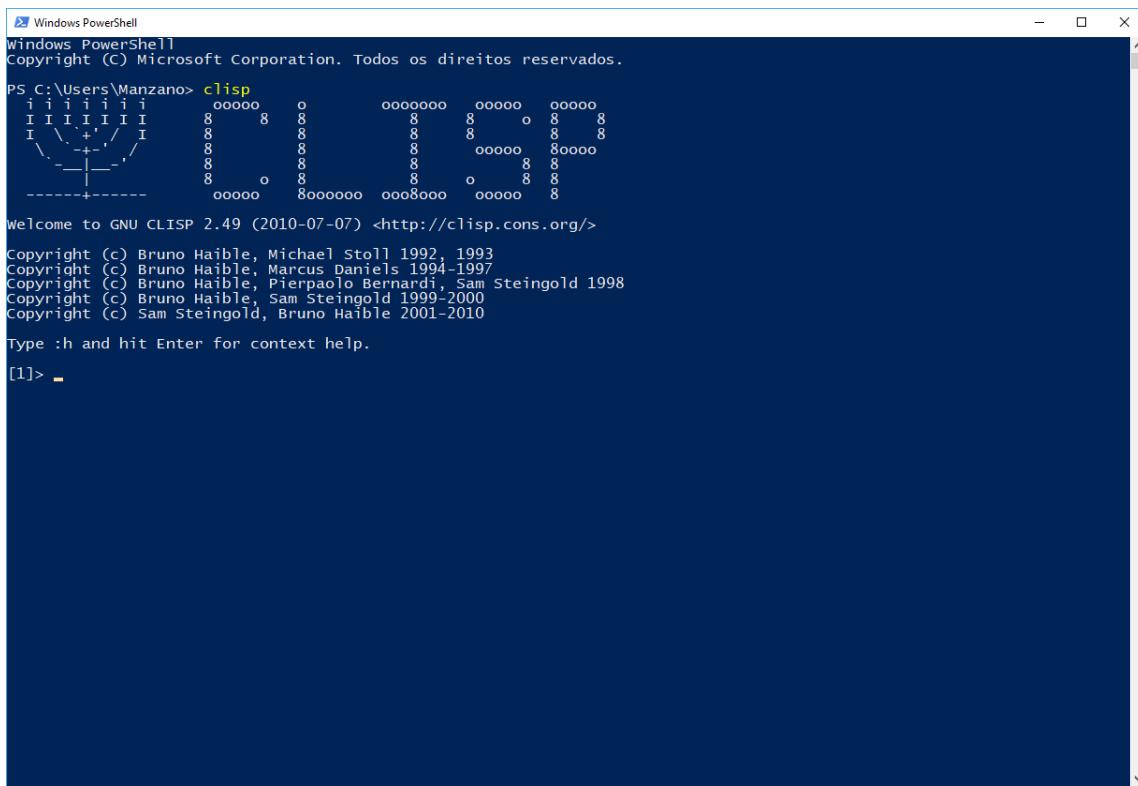
Welcome to GNU CLISP 2.49 (2010-07-07) <http://clisp.cons.org/>
Copyright (c) Bruno Haible, Michael Stoll 1992, 1993
Copyright (c) Bruno Haible, Marcus Daniels 1994-1997
Copyright (c) Bruno Haible, Pierpaolo Bernardi, Sam Steingold 1998
Copyright (c) Bruno Haible, Sam Steingold 1999-2000
Copyright (c) Sam Steingold, Bruno Haible 2001-2010

Type :h and hit Enter for context help.

[1]> -
  
```

Figura 1.11 - Tela do ambiente CLISP a partir do programa CMD.

Se efetuada a execução do ambiente POWERSHELL a apresentação da tela de operação do ambiente CLISP assemelha-se a figura 1.12.



The screenshot shows a Windows PowerShell window titled "Windows PowerShell". The command "clisp" is run from the path "PS C:\Users\Manzano>". The output displays the CLISP logo, which consists of a grid of characters resembling letters and numbers. Below the logo, the text "Welcome to GNU CLISP 2.49 (2010-07-07) <http://clisp.cons.org/>" is shown, followed by a series of copyright notices for various contributors from 1992 to 2001. At the bottom, it says "Type :h and hit Enter for context help." and "[1]> -".

Figura 1.12 - Tela do ambiente CLISP a partir do programa POWERSHELL.

Para encerrar a execução do ambiente CLISP execute um dos comandos:

(EXIT) ou (exit)  
(BYE) ou (bye)  
(QUIT) ou (quit)

Atente para o uso obrigatório dos parentes circundando o comando.

## 1.3 OBTEÇÃO E INSTALAÇÃO DO SBCL

---

O programa SBCL (de **Steel Bank COMMON LISP**) é um interpretador e compilador de alto desempenho, distribuído no regime **free software** a partir da licença padrão BSD (para alguns subsistemas) e como domínio público (para o resto do sistema). É ideal que se faça a leitura da licença disponibilizada no sítio do projeto. O SBCL é executado em diversos sistemas operacionais padrão POSIX e no Windows, ainda de forma experimental.

O programa SBCL passa por constantes revisões e atualizados lançamentos, sendo descente do código do programa CMU CL, criado na universidade Carnegie Mellon para sistemas operacionais destinados a plataforma UNIX.

Para obter informações diversas sobre o programa SBCL acesse o sítio oficial do projeto <http://sbcl.org/> como mostra a figura 1.13.

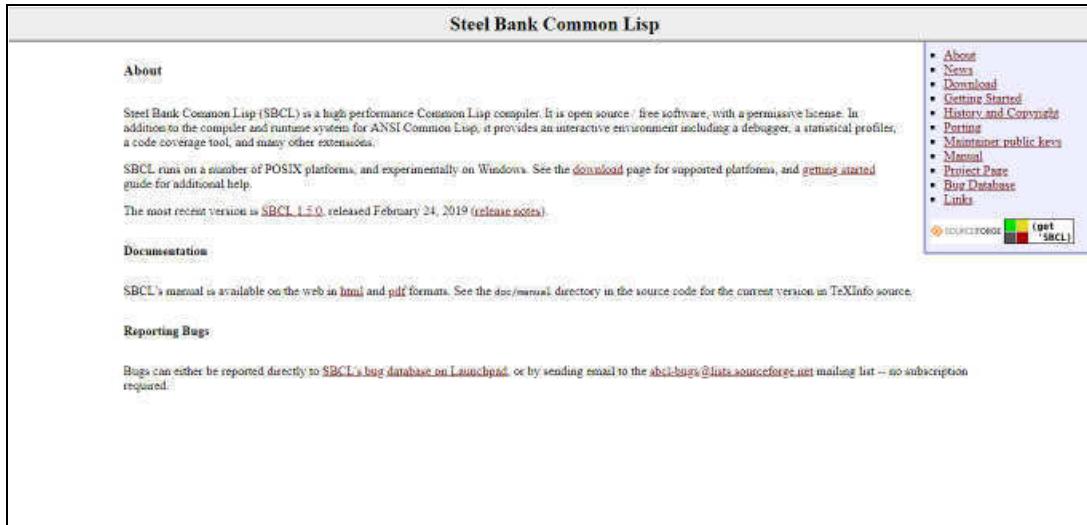


Figura 1.13 - Tela da página do projeto CLISP.

A obtenção de uma cópia para instalação pode ser obtida a partir do sítio do projeto a partir da seleção da opção de menu Download. No quadro apresentado na seção Binaries selecione na linha que indica o sistema operacional Windows a opção da coluna x86 para Windows de 32 bit ou AMD64 para Windows de 64 bits.

Na sequência é apresentada a página do projeto SourceForge indicando o nome do arquivo selecionada e poucos segundos depois apresenta uma caixa de diálogo mostrando o arquivo de programa selecionado para cópia. Neste instante, ação o botão Salvar arquivo para copiar o programa de instalação para seu computador.

Se selecionada a opção da coluna x86 será indicado a cópia do arquivo de instalação `sbcl-999-x86-windows-binary.msi`, mas, se selecionada a opção da coluna AMD64 será indicado a cópia do arquivo de instalação `sbcl-999-x86-64-windows-binary.msi`, onde 999 é o número de identificação da versão mais recente disponibilizada para aquisição.

Após a cópia do arquivo de instalação vá ao local copiado em com um duplo clique do ponteiro do mouse selecione o programa para instalação. Em seguida se apresentada a caixa de diálogo de aviso de segurança solicitando autorização para iniciar a instalação proceda com a seleção do botão Executar.

Em seguida é apresentada a caixa de diálogo de início de instalação Steel Bank Common Lisp 1.4.14 (X86) Setup, semelhante a imagem indicada na figura 1.14.

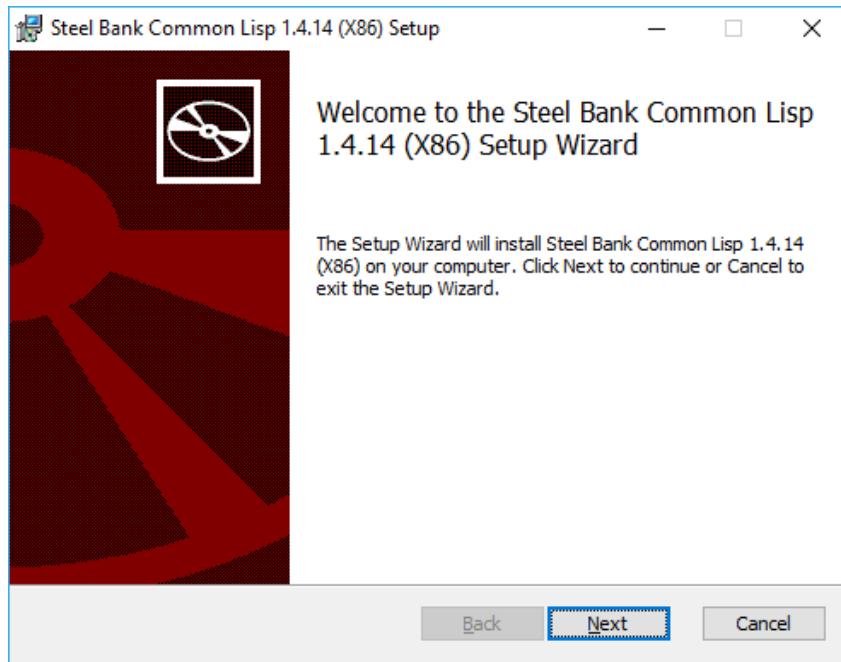


Figura 1.14 - Caixa de diálogo: Steel Bank Common Lisp 1.4.14 (X86) Setup.

Acione o botão Next e será apresentada a tela End-User License Agreement, como mostra a figura 1.15, após proceder a leitura e concordando com os termos acione a caixa de opção I accept the terms in the License Agreement e selecione na sequência o botão Next para que o processo de instalação seja continuado. Caso selecione o botão Cancel o processo de instalação é interrompido.

Após aceitar os termos do contrato é apresentada a tela Custom Setup como apresenta a figura 1.16. Mantenha as opções apresentadas e acione na sequência o botão Next para a próxima etapa.

Na tela Ready to install Steel Bank Common Lisp 1.4.14 (X86), como indica a figura 1.17, acione o botão Install.

Aguarde o término da instalação e ao final deste processo acione o botão Finish da tela Completed the Steel Bank Common Lisp 1.4.14 (X86) Setup Wizard como indica a figura 1.18.

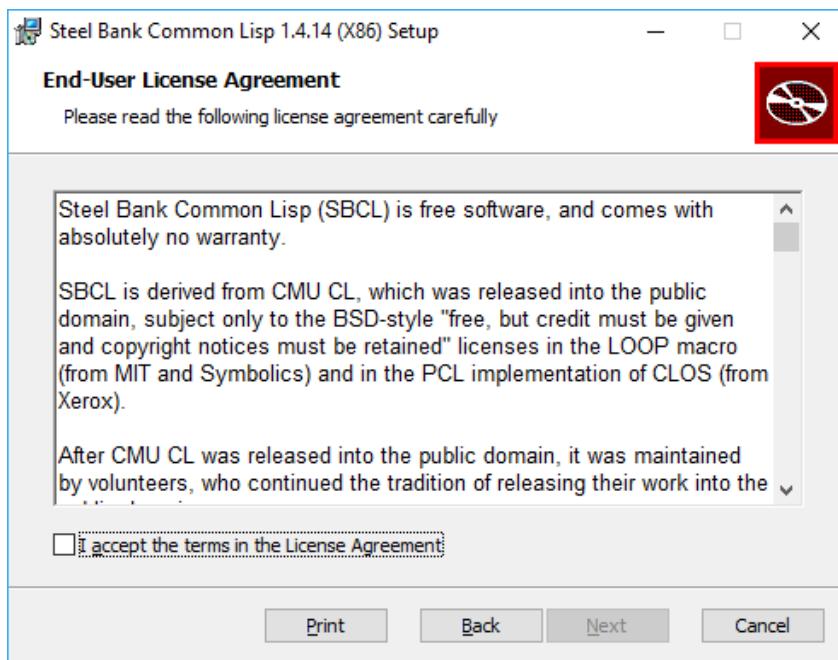


Figura 1.15 - Caixa de diálogo: *End-User License Agreement*.

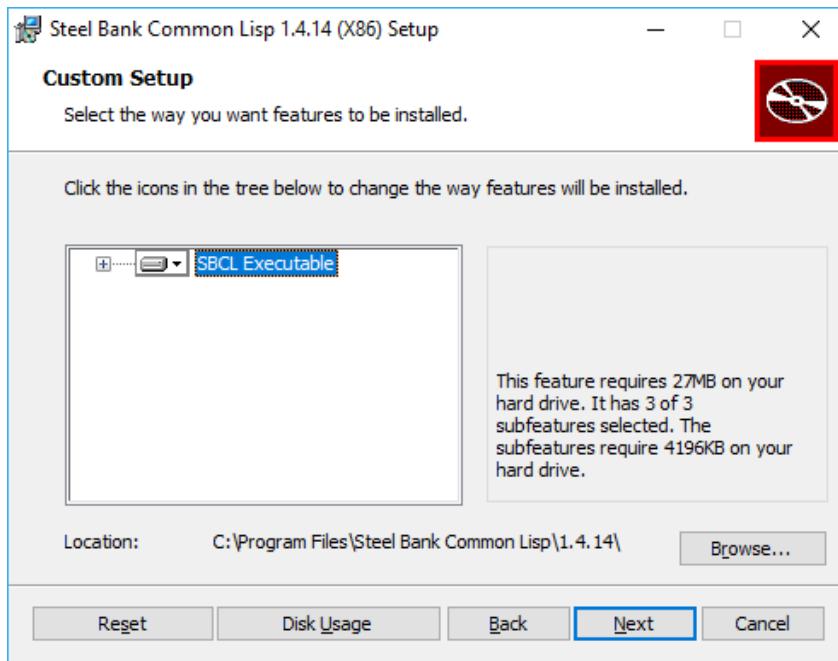


Figura 1.16 - Caixa de diálogo: *Custom Setup*.

Há duas outras formas de fazer uso do ambiente SBCL no Windows. A partir das teclas <Win> + <r>, que apresenta a figura 1.10, informe no campo Abrir da caixa de diálogo Executar a chamada de um dos programas de processamento de comandos CMD ou POWERSHELL e ação o botão OK.

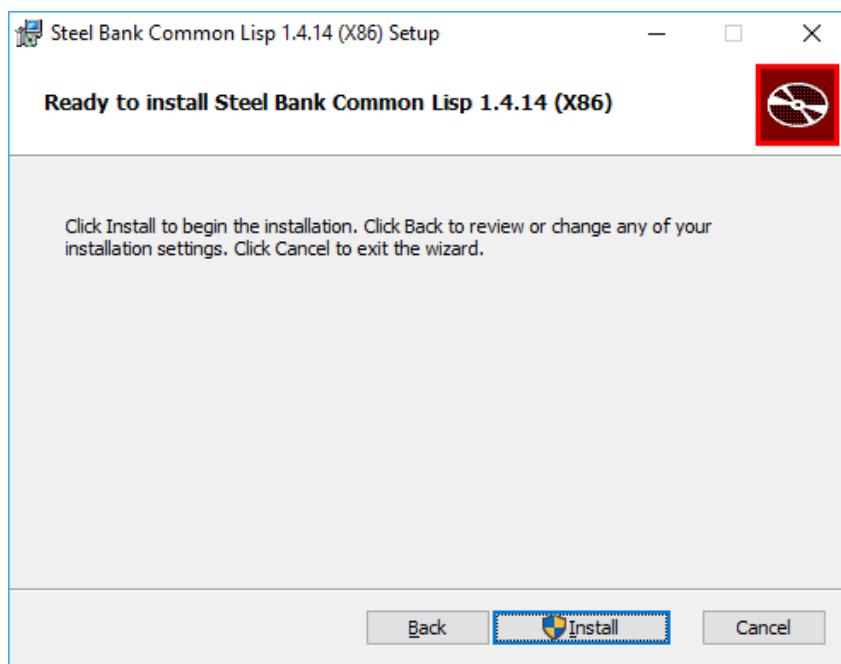


Figura 1.17 - Caixa de diálogo: Ready to install Steel Bank Commom Lisp 1.4.14 (X86).

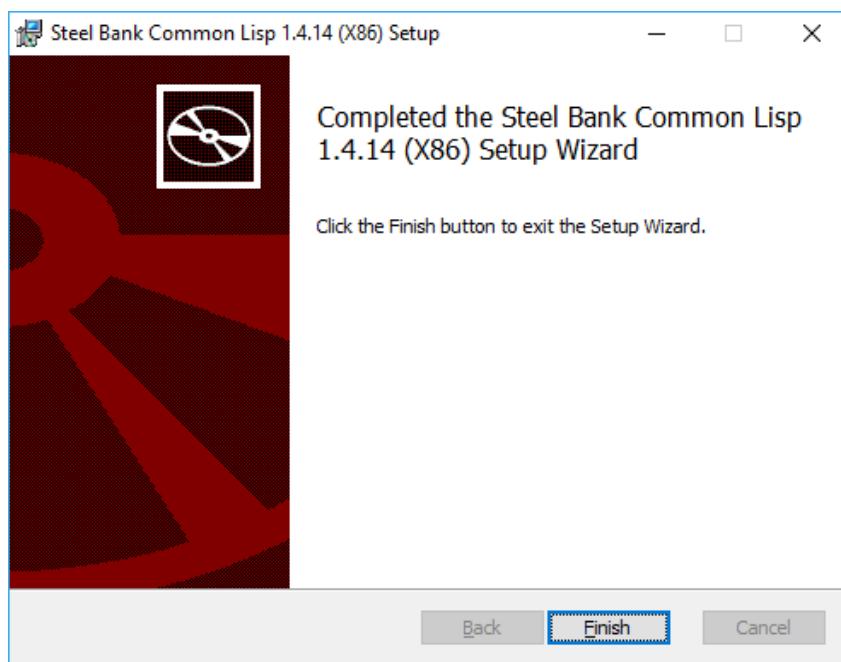
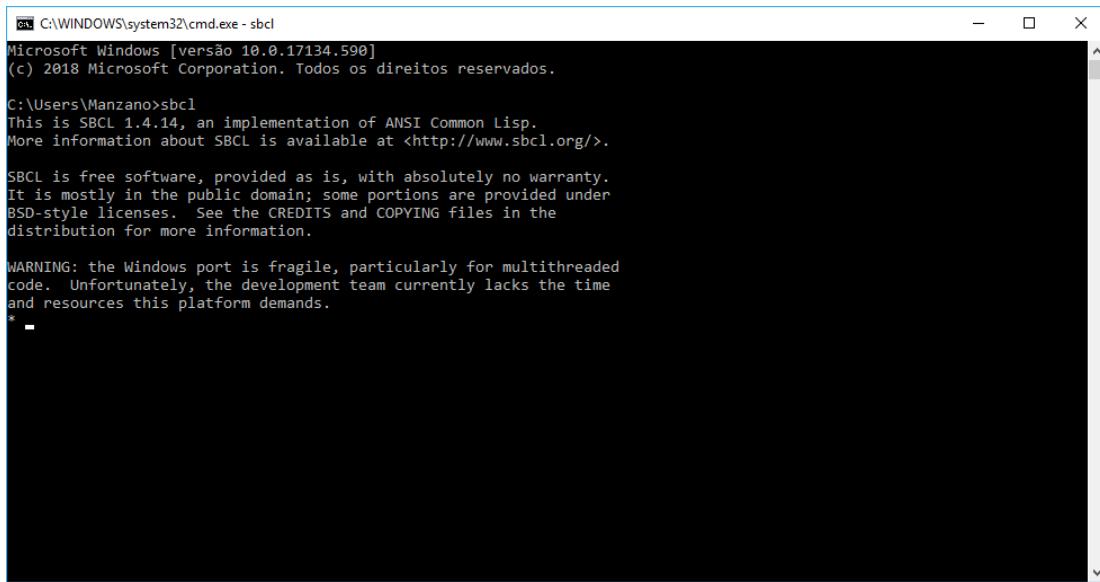


Figura 1.18 - Caixa de diálogo: tela Completed the Steel Bank Commom Lisp 1.4.14 (X86) Setup Wizard.

Assim que o **prompt** do processador de comandos dos programas CMD/POWERSHELL for apresentado efetue a execução do comando SBCL e acione a tecla <Enter>.

Se efetuada a execução do ambiente CMD a apresentação da tela de operação do ambiente SBCL como mostra a figura 1.19.



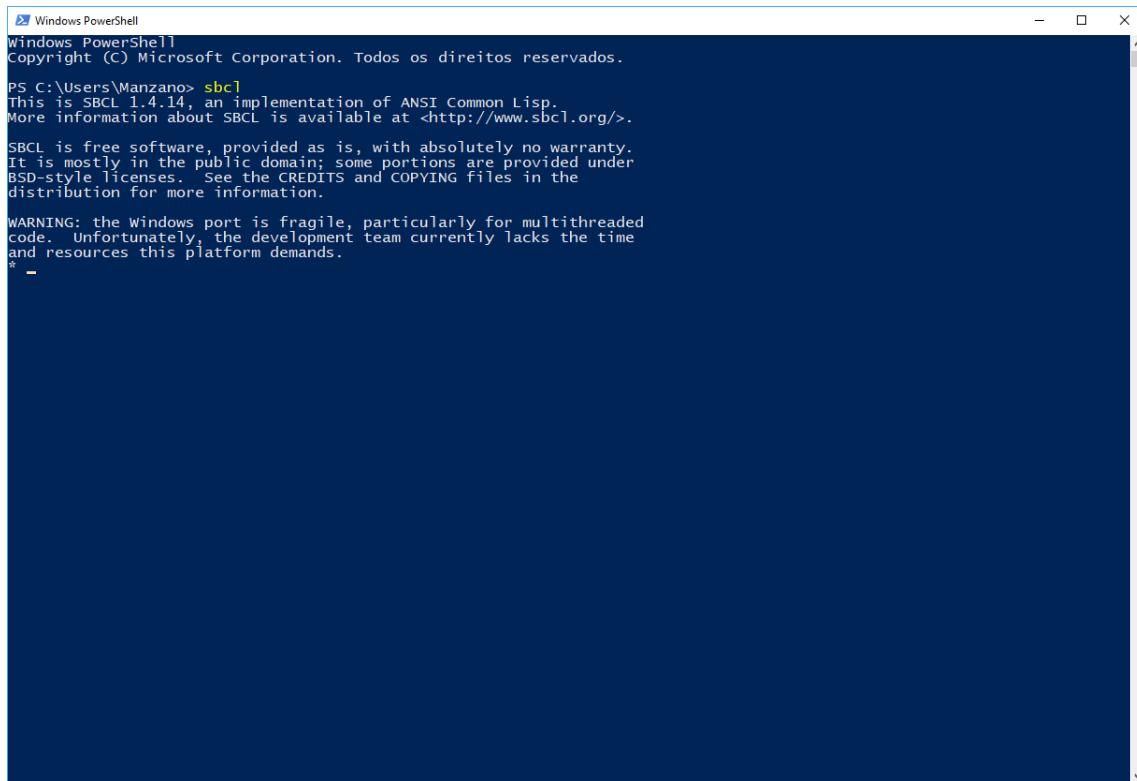
```
C:\WINDOWS\system32\cmd.exe - sbcl
Microsoft Windows [versão 10.0.17134.590]
(c) 2018 Microsoft Corporation. Todos os direitos reservados.

C:\Users\Manzano>sbcl
This is SBCL 1.4.14, an implementation of ANSI Common Lisp.
More information about SBCL is available at <http://www.sbcl.org/>.

SBCL is free software, provided as is, with absolutely no warranty.
It is mostly in the public domain; some portions are provided under
BSD-style licenses. See the CREDITS and COPYING files in the
distribution for more information.

WARNING: the Windows port is fragile, particularly for multithreaded
code. Unfortunately, the development team currently lacks the time
and resources this platform demands.
*
```

Figura 1.19 - Tela do ambiente SBCL a partir do programa CMD.



```
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos os direitos reservados.

PS C:\Users\Manzano> sbcl
This is SBCL 1.4.14, an implementation of ANSI Common Lisp.
More information about SBCL is available at <http://www.sbcl.org/>.

SBCL is free software, provided as is, with absolutely no warranty.
It is mostly in the public domain; some portions are provided under
BSD-style licenses. See the CREDITS and COPYING files in the
distribution for more information.

WARNING: the Windows port is fragile, particularly for multithreaded
code. Unfortunately, the development team currently lacks the time
and resources this platform demands.
*
```

Figura 1.20 - Tela do ambiente SBCL a partir do programa POWERSHELL.

Se efetuada a execução do ambiente POWERSHELL a apresentação da tela de operação do ambiente SBCL assemelha-se a figura 1.20.

Para encerrar a execução do ambiente SBCL execute o comando:

(QUIT) ou (quit)

Atente para o uso obrigatório dos parentes circundando o comando.

## 1.4 QUAL AMBIENTE USAR: CLISP OU SBCL?

---

Nos dois tópicos anteriores foram apresentadas duas opções de programas para execução e aprendizagem do código LISP indicado neste trabalho. Ambas as ferramentas são aqui descritas devido ao fato de serem ambientes populares.

O programa CLISP foi atualizado pela última vez (até a publicação deste trabalho) no ano de 2010 enquanto que o programa SBCL é atualizado com mais frequência. Segundo aponta o grupo **Common Lisp Brasil** (<https://lisp.com.br/>) a ferramenta CLISP é considerada um artefato histórico, ou seja, é uma ferramenta obsoleta e indica entre outras três opções o uso da ferramenta SBCL apontando esta como a mais recomendada.

A despeito das considerações apresentadas ambas as ferramentas são boas candidatas para o uso de iniciantes na linguagem e sua aprendizagem. O que deve ser considerado é que para aplicações robustas a ferramenta SBCL se apresenta mais adequada que a ferramenta CLISP. É importante salientar que CLISP em alguns momentos é mais permissivo que SBCL permitindo escrever operações que em SBCL geram certas mensagens de advertência, o que faz do SBCL uma ferramenta mais adequada, mas não desmerece o CLISP.

Nesta obra as duas ferramentas foram usadas e nelas foram testados cada um dos exemplos e exercícios desenvolvidos. Fica ao seu critério a opção em escolher a ferramenta que lhe seja mais agradável e adequada.

Na interface de comunicação dos ambientes a um ***prompt*** que indica a prontidão de uso. O símbolo de ***prompt*** é a forma com que a interface avisa que está pronta para receber determinado comando.

No programa CLISP o indicativo de prontidão é sinalizado com o símbolo “[n]>”, onde “n” indica um valor numérico sequencial informando as interações realizadas até então. Já no programa SBCL o indicativo de prontidão é sinalizado com o símbolo “\*” asterisco.

Devido à particularidade operacional das formas de indicação de ***prompts*** de cada ferramenta e para homogeneização das operações apresentadas nesta obra será usado como indicativo de ***prompt*** nos exemplos e exercícios de demonstração o símbolo “>>>” para as ações que se comportam igualmente em ambos os programas.

Quando uma ação interativa é executada em LISP a resposta poderá ocorrer em uma ou duas linhas imediatamente a indicação da ação. Em alguns casos a resposta apresentada poderá ocorrer diferentemente entre os dois programas, mesmo sendo a mesma a resposta. Quando esses efeitos ocorrerem serão indicados os ***prompts*** originais de cada uma das ferramentas.

## 1.5 CARACTERÍSTICAS BÁSICAS SOBRE LISP

---

A linguagem LISP é uma linguagem de programação dinâmica que permite a construção de programas na forma de funções simples (expressões simbólicas), onde, cada expressão mais genérica pode ser combinada para obter-se um programa com funcionalidade completa.

Para dar ideia sobre o potencial da linguagem LISP o programa AutoCAD da Autodesk possui internamente como extensão de suas funcionalidades uma versão da linguagem LISP chamada AutoLISP, além do editor de textos Emacs e do programa de reserva de passagens aéreas Orbitz, entre outros escritos em LISP.

Pelo fato de ser LISP uma linguagem simples esta permite ao programador estender suas funcionalidades permitindo usar a linguagem como suporte a operação de outros paradigmas de programação. Originalmente LISP é uma linguagem pertencente ao paradigma declarativo funcional, mas devido a sua característica básica operacional permite que esta incorpore diversos outros paradigmas. Isso faz com que LISP consiga permanecer em uso atualmente, além de outras linguagens que surgiram e desapareceram.

LISP como uma das suas maiores características possui a capacidade de ser uma linguagem interativa, além da parte programada que possui. Cada função ou expressão criada para um programa pode ser testada, depurada e corrigida separadamente de forma rápida, sem que para isso seja necessário rever o programa todo ou mesmo compila-lo se for este o caso. LISP é uma linguagem incremental que facilita a construção de grandes programas com grande facilidade.

O menor programa LISP pode ser escrito sem nenhum conteúdo definido dentro de um par de parênteses na forma de lista que após execução retorna como resposta o resultado NIL indicando que nada foi realizado. NIL é um constante especial que representa uma resposta com valor falso após a execução de uma expressão, se uma ação retornar verdadeiro a constante especial usada é T. Tanto NIL como T podem ser expressos em caracteres minúsculos ou maiúsculos.

( )

NIL

O contexto definido dentro de parênteses é chamado em LISP de **expressão**, e por esta razão qualquer coisa escrita em LISP que esteja entre parênteses, até mesmo um programa inteiro é considerada uma expressão. Veja o grau de simplicidade que isso sugere. Todo comando LISP é por sua natureza uma função, ou seja, uma expressão indicada sempre dentro de parênteses que retorna um valor como resposta, mesmo quando o retorno é NIL ao indicar que nada foi realizado ou que o retorno é falso. É pertinente salientar que qualquer elemento informado dentro de parênteses ao interpretador LISP é chamado de **form**.

Um programa LISP mais sofisticado é baseado em uma estrutura de lista configurada a partir da sintaxe:

**(ação [<argumento1> [<argumento2> [... <argumentoN>]]])**

Onde o indicativo **ação** refere-se a definição de alguma operação a ser realizada pela linguagem sobre o(s) argumento(s) fornecido(s); **argumento1**, **argumento2** e **argumento** indicam a definição do que deverá se processado na linguagem. A lista de argumentos fornecidos a uma ação é variável e depende da regra de uso da função aplicada.

**OBS:** Os argumentos indicados nesta obra que se encontram entre os símbolos “<” e “>” caracterizam-se por serem obrigatórios e quando entre os símbolos “[” e “]” caracterizam-se por serem opcionais. De qualquer forma não é para proceder ao uso desses caracteres apenas dos conteúdos indicados por eles quando assim for necessário.

As expressões LISP podem ser definidas de forma aninhada, uma dentro da outra, o que obriga a se ter alto grau de atenção para indicar o fechamento dos diversos parênteses utilizados em determinada expressão. Essa ocorrência é tão preocupante que ao longo dos anos alguns programadores LISP começar a dizer que o significado da sigla LISP é **Lots of Irritant Stupid Parents**, ou seja, muitos parênteses estupidamente irritantes.

Uma linguagem de programação, seja ela qual for, opera a partir de dois elementos distintos: os dados e o código do programa. Os dados representam os valores a serem processados e o código representa o processamento dos dados em si. Em LISP o código do programa pode ser a chamada da **ação** e os dados pode ser a execução dos **argumentos**. No entanto, por vezes, o **argumento** pode ser definido como **ação** e a **ação** pode ser definida como **argumento**, ou seja, em LISP é pos-

sível implementar novos recursos a partir das estruturas de dados existentes ou definidas pelo programador, permitindo grande extensão de seus recurso a um nível alto de abstração. Quando uma linguagem de programação permite que uma ação possa ser interpretada como argumento e que um argumento possa ser interpretado como ação linguagem opera **homoiconicidade**, sendo LISP uma linguagem desta categoria.

Na linguagem LISP os dados a serem usados são considerados a partir de valores numéricos, variáveis e constantes. O código, quanto processamento, estabelece as regras de instruções que visam manipular os dados em memória (regras de negócio), sendo estes baseados em elementos primitivos da linguagem como a execução de operações matemáticas a partir de expressões aritméticas ou algébricas ou mesmo de expressões lógicas.

Como comentado a base da programação LISP é a definição de expressões simbólicas na forma de ações delimitadas entre parênteses que sempre retornam algum valor como resposta a ação. Uma expressão simbólica LISP é também referenciada como **sexp**.

A ação delimitada entre parênteses é composta de uma operação e seus argumentos representados pelos parâmetros fornecidos para a execução de certa ação.

A operação definida a uma expressão como ação é estabelecida a partir do uso de certo recurso, como: função (rotina de programa que retorna um valor como resposta a uma ação específica), macro (recurso que amplia certa função agregando mais funcionalidades a sua ação original), predicado (função que testa a validade de seus argumentos para a execução de condições específicas), construtor (função que cria dados compostos a partir de seus elementos mais simples), seletor (função que recebe um dado composto e devolve suas partes em separado), reconhecedor (função que identifica certo detalhe especial do tipo de dado definido) e teste (função de comparação de dados). Os recursos função, macro, predicado, construtor, seletor, reconhecedor e teste formam a base de definição para uma interface entre o uso e a implementação de certos dados, caracterizando-se a estrutura para definição de tipos abstratos de dados e informações em LISP.

Outra característica operacional da linguagem LISP é o fato de ser operada a partir do uso de notação pré-fixa (notação polonesa) que garante a definição de operações não ambíguas.

Se há o desejo de efetuar a expressão aritmética  $2 + 5$  para obter o resultado 7 deve-se escrevê-la na forma pré-fixada (notação polonesa) como:

(+ 2 5)

Dependendo de como os valores de certa expressão aritmética são dispostos e calculados  $2 + 3 * 5$  pode-se obter o resultado 17, se realizada  $2 + (3 * 5)$  ou pode-se obter o resultado 25, se for realizada  $(2 + 3) * 5$ . A expressão aritmética  $2 + 3 * 5$  caracteriza-se por ser uma expressão ambígua e uma forma de eliminar sua ambiguidade é expressá-la explicitamente da maneira que se deseja calcular como:

(+ 2 (\* 3 5))  
(\* 5 (+ 2 3))

Observe nos exemplos indicados que em LISP cada expressão simbólica é representada por uma lista contendo como primeiro elementos um **operador** aplicado a uma lista de um ou mais **operandos**. Veja que em LISP os operandos são avaliados e computados antes de serem passados para o operador que executa a ação matemática desejada. O estilo (operador operandos) é conhecido na linguagem LISP como **avaliação estrita**.

Um recurso existente em LISP é sua capacidade de efetuar o tratamento de seu código como se esse fosse um dado e vice-versa como normalmente ocorre em linguagens funcionais, caracterizando-se desta forma operações com funções de primeira ordem.

Em especial o padrão COMMON LISP dá suporte ao paradigma da programação orientada a objetos com algumas peculiaridades como a definição de métodos não associados a certa classe dando suporte a classe em si, herança e o uso de sobre-carga. É pertinente salientar que a orientação a objetos em LISP se difere dos estilos usados em linguagens como C++, Java ou C# e não será tratado neste trabalho por fugir de seu escopo, merecendo uma obra particularizada para o tema.

Além dos detalhes apresentados LISP aceita o empacotamento (**packaging**) de programas em módulos, permite definição de macros, uso de funções genéricas, efetua o gerenciamento automático de memória com coleta de lixo.

O objetivo desta obra é apresentar de forma simples e introdutória a linguagem LISP para iniciantes na linguagem funcional de computadores. Desta forma, serão abordados durante as próximas páginas diversos exemplos de códigos que fornecerão ao final do estudo uma base para a compreensão mais acentuada de programas escritos em LISP.

## **1.6 INTERAÇÃO BÁSICA (PRIMEIRO CONTATO)**

Com o objetivo de proporcionar visão introdutória a linguagem LISP e algumas de suas funcionalidades efetue a chamada do ambiente CLISP como já orientado para realizar algumas operações básicas. O programa CLISP é uma ferramenta conhecida pela alcunha de **avaliador** que tem por finalidade interagir com o usuário.

O primeiro exercício é a apresentação da mensagem “Alo, mundo!”. Assim sendo, execute a instrução:

```
(print "Alo, mundo!")
```

Ao se executada a ação anterior ocorre um salto de linha em branco e a apresentação da mensagem duas vezes com as aspas delimitadas. A primeira apresentação da mensagem refere-se a apresentação da mensagem em si (retorno primário da função), a segunda apresentação refere-se ao retorno secundário da função `print`, como mostram as figuras 1.21 e 1.22 respectivamente para os programas CLISP e SBCL.

```
PS C:\Users\Augusto Manzano> clisp
i i i i i i i   oooooo   o   ooooooo  ooooo   ooooo
I I I I I I I   8   8   8   8   8   8   o   8   8
I \ + , / I     8   8   8   8   8   8   ooooo   8ooooo
I \ - , / I     8   8   8   8   8   8   o   8   8
----- oooooo   8ooooooo  ooo8ooo   ooooo   8
Welcome to GNU CLISP 2.49 (2010-07-07) <http://clisp.cons.org/>
Copyright (c) Bruno Haible, Michael Stoll 1992, 1993
Copyright (c) Bruno Haible, Marcus Daniels 1994-1997
Copyright (c) Bruno Haible, Pierpaolo Bernardi, Sam Steingold 1998
Copyright (c) Bruno Haible, Sam Steingold 1999-2000
Copyright (c) Sam Steingold, Bruno Haible 2001-2010

Type :h and hit Enter for context help.

[1]> (print "Alo, mundo!")

"Alo, mundo!"
"Alo, mundo!"
[2]>
```

Figura 1.21 - Tela com resultado da ação da função “print” no CLISP.

Note que inicialmente o ambiente CLISP mostra o ***prompt*** [1]> e após a execução da instrução o ***prompt*** é indicado como [2]>. A cada execução efetuada o ***prompt*** será atualizado com um novo valor sequencial informado a quantidade de entradas efetuadas. Ao sair do ambiente e fazer seu retorno o contador de entrada é sempre inicializado.

```
This is SBCL 1.4.14, an implementation of ANSI Common Lisp.  
More information about SBCL is available at <http://www.sbcl.org/>.  
SBCL is free software, provided as is, with absolutely no warranty.  
It is mostly in the public domain; some portions are provided under  
BSD-style licenses. See the CREDITS and COPYING files in the  
distribution for more information.  
  
WARNING: the Windows port is fragile, particularly for multithreaded  
code. Unfortunately, the development team currently lacks the time  
and resources this platform demands.  
* (print "Alo, mundo!")  
"Alo, mundo!"  
"Alo, mundo!"  
* -
```

Figura 1.22 - Tela com resultado da ação da função “print” no SBCL.

Similarmente a função `print` há a função `write` que efetua apresentação de conteúdo semelhante sem o salto antecipado de linha. Assim sendo execute a instrução:

```
(write "Alo, mundo!")
```

Observe a apresentação da mensagem “Alo, mundo!” sem o salto antecipado de linha passando o **prompt** a ser indicado como [3]>.

Além das funções `print` e `write` há uma terceira maneira de se realizar apresentações com um controle diferenciado por meio da função `format`. Assim sendo execute a instrução:

```
(format t "~%Alo, mundo!~%")
```

Neste caso ocorre com o uso do especificador % (porcentagem) um efeito de salto de linha antes da apresentação da mensagem e após a apresentação da mensagem. A mensagem é apresentada sem o uso dos símbolos de aspas.

Antes do especificador de salto de linha é usado a diretiva ~ (til) que indica o uso de certo efeito de apresentação junto a cadeia de caracteres delimita entre os símbolos de aspas inglesas.

Após a apresentação da mensagem ocorre o retorno do valor NIL indicando retorno de valor lógico como falso. A função `format` possui de forma simplificada a estrutura operacional (será a função documentada com mais detalhes no próximo capítulo).

```
(format <destino> <cadeia>)
```

O argumento **destino** indica o modo de apresentação do conteúdo no terminal de vídeo indicado em **cadeia**, podendo ser definido como nil (falso), t (verdadeiro) ou a indicação de uso de um arquivo. O argumento **cadeia** pode ser definido com o auxílio de códigos de formatação para estabelecer suas diretivas de ação.

O destino de saída t refere-se ao uso do fluxo de saída \*STANDARD-OUTPUT\* indicando que a saída deve ocorrer no terminal. O valor NIL indica que não deve ser apresentado absolutamente nada como resposta da função, mas indica que deve ocorrer a apresentação da cadeia de caracteres. A referência de uso de um arquivo indica que a saída deve ser realizada no arquivo e não no terminal.

A função **format** para a apresentação de dados fornece mais funcionalidades que as funções **print** e **write** a partir da aplicação de diversos especificadores, dos quais, alguns serão apresentados ao longo desta obra.

As figuras 1.23 e 1.24 apresentam trechos de tela com os resultados obtidos com o uso das funções **print**, **write** e **format** nos ambientes CLISP e SBCL.

```
[1]> (print "Alo, mundo!")
"Alo, mundo!"
[Alo, mundo!
[2]> (write "Alo, mundo!")
"Alo, mundo!"
[Alo, mundo!
[3]> (format t "~%Alo, mundo!~%")
Alo, mundo!
NIL
[4]> _
```

Figura 1.23 - Tela com resultado da ação das funções “print”, “write” e “format” no CLISP.

```
* (print "Alo, mundo!")
"Alo, mundo!"
[Alo, mundo!
* (write "Alo, mundo!")
"Alo, mundo!"
[Alo, mundo!
* (format t "~%Alo, mundo!~%")
Alo, mundo!
NIL
*
```

Figura 1.24 - Tela com resultado da ação das funções “print”, “write” e “format” no SBCL.

Observe na sequência a execução de algumas operações aritméticas simples de adição (+), subtração (-), multiplicação (\*) e divisão (/). Não entre no prompt o

símbolo “>>>” apenas informe a operação delimita entre parênteses incluindo-se os parênteses.

```
>>> (+ 1 2)
```

```
3
```

```
>>> (+ 1 2 3)
```

```
6
```

```
>>> (+ 1.5 2.3)
```

```
3.8
```

```
>>> (- 5 3)
```

```
2
```

```
>>> (- 5 3 1)
```

```
1
```

```
>>> (* 2 3)
```

```
6
```

```
>>> (* 1.5 3)
```

```
4.5
```

```
>>> (* 1.5 3 2)
```

```
9.0
```

```
>>> (/ 5.0 2)
```

```
2.5
```

```
>>> (/ 5.0 2 .5)
```

```
5.0
```

As operações com mais de dois argumentos são efetuadas sucessivamente não importando a função de operação utilizada (+, -, \* ou /). Atente para a indicação do ***prompt*** [*n*], onde “*n*” representa um valor numérico sequencial apresentado.

Além das funções (+, -, \* e /), tem-se um grande conjunto de funções primitivas (funções internas) para a realização de diversas operações. Veja alguns exemplos de funções para os cálculos de exponencial (exp), potência (expt), raiz quadrada (sqrt), maior valor (max), menor valor (min) e valor absoluto (abs).

```
>>> (exp 1)
```

```
2.7182817
```

```
>>> (exp 2.5)
```

```
12.182494
```

```
>>> (expt 2 3)
```

```
8
```

```
>>> (expt 2.5 3.8)
```

```
32.521606
```

```
>>> (sqrt 25)
```

```
5
```

```
>>> (max 1 2 3)
```

```
3
```

```
>>> (min 1 2 3)
```

```
1
```

```
>>> (abs -5)
```

```
5
```

A partir desta exposição você já possui o mínimo necessário para aprofundar seu conhecimento em LISP. Até o próximo capítulo.