公開

002

TR-IT-0289

リスト処理用ライブラリ (LinC) 説明書

西村 仁志 *　　隅田 英一郎
Hitoshi NISIMURA and Eiichiro SUMITA

1999 年 2 月 26 日

概 要

本書は C 言語用リスト処理ライブラリ List processing in C （以降 LinC ）のマニュアルである。LinC は Common Lisp の主要な関数やマクロに相当するルーチンを実装している。このライブラリは、リスト処理を多用するシステムなどを C 言語で作成するのに活用でき、実際、C 版変換主導翻訳システムで利用されている。

エイ・ティ・アール音声翻訳通信研究所
ATR Interpreting Telecommunications Research Laboratories

---

* (株) 国際電気通信基礎技術研究所

もくじ

表一覧

図一覧

第 1 章

システム概要

本章ではLinCの稼働環境、特徴、ライブラリのC言語に対する位置付けについて説明する。

1.1 稼働実績環境

表 1.1: LinCの稼働実績環境

機種	OS	コンパイラ	シェル
SUN WorkStation	SunOS4.1.3	gcc 2.7.2.2, g++ 2.7.2.2	tcsh
DEC Alpha Station	DigitalUnixV3.2	gcc 2.7.2.2, g++ 2.7.2.2	tcsh
KSR	KSROS	cc	tcsh
DOS/V	Linux	gcc 2.7.2.2, g++ 2.7.2.2	tcsh
DOS/V	windows95	VisualC++ 4.0	-

1.2 特徴

LinCの特徴は、次のとおりである。

1. Common Lispと同機能の関数がたくさん利用できる。
2. 本ライブラリを利用して作ったリスト処理プログラムは全体がCになるので処理速度がLISPより速い。
3. 本ライブラリを利用すれば、既存のリスプで記述されたプログラムを容易にCにコンバートできる。(ほとんど、単純作業でできる。)
4. トレースが出力できるので、プログラムのデバッグも容易である。
5. 本ライブラリの引数及びリターン値は、型OBJECT* で統一されているので、プログラマーはデータの型を意識する必要がない。

6. ユーザ固有の OBJECT 型の構造体が定義できるマクロが用意されている。

1.3 本ライブラリの C 言語に対する位置付け

本ライブラリの C 言語に対する位置付けを以下に示す。

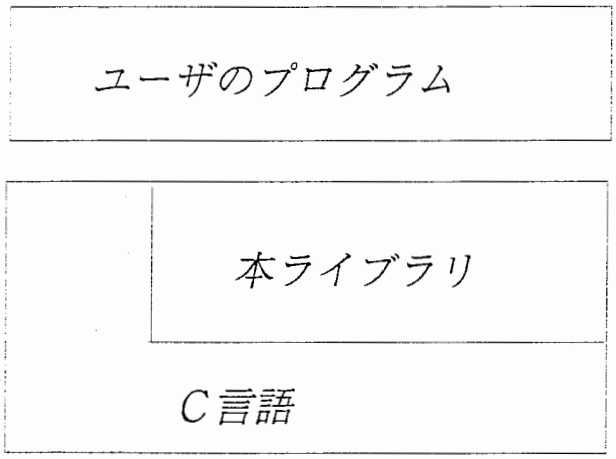

図 1.1: 本ライブラリの C 言語に対する位置付け

第 2 章

利用方法

本章では、LinC の利用方法について説明する。

2.1 LinC と gc をインストールする

1. 配布媒体から LinC と gc を、インストールするディレクトリにコピーする（gc, LinC の 2 つのディレクトリがコピーされる）
   例： cp -r /export/* /home/.

2. ゴミ集めライブラリ（gc）をインストールする
   (a) ゴミ集めライブラリの READM(例:/home/gc/README) を見て、エディタで開き、実行ファイルを作る。

3. リスト処理ライブラリ（LinC）をインストールする
   (a) Unix の場合
      i. bison(gnu 版 yacc) と flex(gnu 版 lex) をインストールする。(LinC 自身を修正しない時は必要ない)
      ii. リスト処理ライブラリの Makefile(例:/home/LinC/Makefile) をエディタで開き、必要に応じ修正する
      iii. LinC をコンパイルし、実行ファイルを作る
   (b) Windows95 の場合
      i. VisualC++ を起動する。
      ii. 「プロジェクトワークスペース」で、Static Library として、プロジェクトを作る。

iii. ヘッダファイルと machine_win32.c 以外のファイルをプロジェクトへ追加する。

iv. 「警告レベル」をなしにする

v. 「プリプロセッサの定義」で変数 WIN32 と GARBAGE を追加する。

vi. 「インクルードファイルのパス」に gc と LinC のパス名をセットする。(例：/home/gc, /home/LinC/win95)

vii. コンパイルする。

2.2 プログラミング手順

本節では、『リストの個数を数える (本ライブラリの length_o と同じ機能)』プログラムを例としてプログラミング手順を以下の順序で説明する。

1. コーディング
2. コンパイル
3. デバッグ
4. メモリ管理

2.2.1 コーディング

1. ヘッダを入れる

```
#include "lisp_like_function.h"
```

2. 本ライブラリの初期化をおこなう。本ライブラリの関数を使用する前に関数 init_lisp_like_function() を呼ぶ必要がある。

```
init_lisp_like_function();
```

3. ユーザ固有の部分をコーディングする

以下に、リストの個数を数えるプログラムのコーディング例 (リカーシブとループの 2 種類) を示す。

```
#define DEVELOP

#include "lisp_like_function.h"

OBJECT *length_list_1 (OBJECT *object)
{
     /*** リカーシブを用いる方法 ***/
     trace_o(length_list_1);
     trace_print_o(object);

     if (atom_o(object)){
  return_o(read_from_fix_value(0));
     }
     else
     {
  return_o(read_from_fix_value
   (1 + get_fix_value(length_list_1(cdr_o(object)))));
     }
}


OBJECT *length_list_2 (OBJECT *object)
{
     /*** ループを用いる方法 ***/
     int i=0;
     dolist_o(obj, object, {
          i++;
          debug_print_o(read_from_fix_value(i));
     });
     return(read_from_fix_value(i));
}
```

次ページにつづく

ここで、read_from_fix_value は C の int 型の数値を OBJECT 型にする関数であり、get_fix_value は OBJECT 型を C の int 型にする関数である。

```
void main (int argc, char **argv)
{
     OBJECT *length;
     OBJECT *input;
     OBJECT *memory_area;

     init_lisp_like_function();

     memory_area = make_memory_area(100);

     with_set_memory_area(memory_area, {
          input = quote_o((1 2 3 a b 3.5));

          printf("\ninput=");
          princ_o(input, T);

          length = length_list_1(input);
          printf("\nlength_1 =");
          princ_o(length, T);
          clear_memory_area();
          length = length_list_2(quote_o((1 2 3 a b 3.5)));
          printf("\nlength_2 =");
          princ_o(length, T);
     });
}
```

2.2.2 コンパイル

マクロの展開

本ライブラリでは、マクロを多用している。そのため gdb などのデバッガでデバッグをするとき、マクロ内でプログラムが落ちた場合、その場所がわからないことがある。

そのため、コンパイル時にマクロを展開されたソースが残るようにコンパイルし、展開されたソースでエラー箇所を見つけれるようにすることが望ましい。

本ライブラリをリンクする

本ライブラリの"lisp_like_function.o" y.tab.o , lex.yy.o , lisp_like_function.o , init_lisp_symbol_s.o , lisp_number.o , machines.o , ../gc/gc.a をリンクする。

```
/*** マクロの展開 ***/
gcc -E -P sample_program.c -o sample_program.i

/*** ライブラリのリンク ***/
gcc -o sample_program sample_program.i y.tab.o lex.yy.o \
lisp_like_function.o init_lisp_symbol_s.o \
lisp_number.o machines.o ../gc/gc.a
```

2.2.3 デバッグ

本ライブラリにはデバッグするための機能としてトレースとデバッグプリントがある。以下に、それらの説明を示す。

なお、以下のマクロ(トレースとデバッグプリント)を有効にするには、変数DEVELOP [1]をファイルの先頭(#include "lisp_like_function.h"より前)で宣言する必要がある。

逆に、この宣言をしなければ、無効になる。

```
#define DEVELOP
#include "lisp_like_function.h"
```

trace の出力

リスプと同様のトレースを出力することができる。具体的には、trace_o(関数名), trace_print_o(関数の引数) [2], return_o(リターン値)の3つのマクロをトレース表示したい関数に以下のようにコーディングすればよい。

また、非表示にするには、trace_o(関数名)をuntrace_o(関数名)とすればよい。

[1] この定義はファイル単位で有効である

[2] trace_print_o を記述すると関数の引数のメモリが使用可能どうかのチェックを行なう

```
OBJECT *length_list_1 (OBJECT *object)
{
     /*** リカーシブを用いる方法 ***/
     trace_o(length_list_1);
     trace_print_o(object);

     if (atom_o(object)){
          return_o(read_from_fix_value(0));
     }
     else
     {
          return_o(read_from_fix_value
            (1 + get_fix_value(length_list_1(cdr_o(object)))));
     }
}
```

トレース結果を以下に示す。

```
||(1)TRACE||-----s------length_list_1
    ||TRACE||object = ( 1  2  3  A  B  3.5000000000)
  ||(2)TRACE||-----s------length_list_1
      ||TRACE||object = ( 2  3  A  B  3.5000000000)
    ||(3)TRACE||-----s------length_list_1
        ||TRACE||object = ( 3  A  B  3.5000000000)
      ||(4)TRACE||-----s------length_list_1
          ||TRACE||object = ( A  B  3.5000000000)
        ||(5)TRACE||-----s------length_list_1
            ||TRACE||object = ( B  3.5000000000)
          ||(6)TRACE||-----s------length_list_1
              ||TRACE||object = ( 3.5000000000)
            ||(7)TRACE||-----s------length_list_1
                ||TRACE||object = nil
            ||(7)TRACE||<---read_from_fix_value(0) =  0
          ||(6)TRACE||<---read_from_fix_value (1 + get_fix_value(length_list_1(cdr_o(object)))) =  1
        ||(5)TRACE||<---read_from_fix_value (1 + get_fix_value(length_list_1(cdr_o(object)))) =  2
      ||(4)TRACE||<---read_from_fix_value (1 + get_fix_value(length_list_1(cdr_o(object)))) =  3
    ||(3)TRACE||<---read_from_fix_value (1 + get_fix_value(length_list_1(cdr_o(object)))) =  4
  ||(2)TRACE||<---read_from_fix_value (1 + get_fix_value(length_list_1(cdr_o(object)))) =  5
||(1)TRACE||<---read_from_fix_value (1 + get_fix_value(length_list_1(cdr_o(object)))) =  6
```

デバッグプリント

OBJECT 型の変数は debug_print_o を使えば、その内容を変数名付きで表示できる。以下に、その例を示す。

```
OBJECT *length_list_2 (OBJECT *object)
{
     /*** ループを用いる方法 ***/
     int i=0;
     dolist_o(obj, object, {
          i++;
          debug_print_o(read_from_fix_value(i));
     });
     return(read_from_fix_value(i));
}
```

デバッグプリント結果を以下に示す。

```
read_from_fix_value(i)= 1
read_from_fix_value(i)= 2
read_from_fix_value(i)= 3
read_from_fix_value(i)= 4
read_from_fix_value(i)= 5
read_from_fix_value(i)= 6
```

2.2.4 メモリ管理

関数 make_memory_area を使えば、予め、メモリを確保できるので処理時間を短くすることができる。また、関数 clear_memory_area を使えば、すでにアロケーションしたメモリを再利用することができる。

```
     OBJECT *memory_area;
     memory_area = make_memory_area(100); /*** メモリを確保する ***/
     with_set_memory_area(memory_area, {
          input = quote_o((1 2 3 a b 3.5));

          printf("\ninput=");
          princ_o(input, T);

          length = length_list_1(input);
          printf("\nlength_1 =");
          princ_o(length, T);
          clear_memory_area();      /*** メモリの初期化をする ***/
          length = length_list_2(quote_o((1 2 3 a b 3.5)));
          printf("\nlength_2 =");
          princ_o(length, T);
     });
```

2.2.5 サンプルプログラムの実行例

```
input=(1 2 3 A B 3.5000000000)
  ---(1)TRACE-------s---length'list'1
     ---TRACE---object=(1 2 3 A B 3.5000000000)
   ---(2)TRACE-------s---length'list'1
      ---TRACE---object=(2 3 A B 3.5000000000)
    ---(3)TRACE-------s---length'list'1
       ---TRACE---object=(3 A B 3.5000000000)
      ---(4)TRACE-------s---length'list'1
         ---TRACE---object=(A B 3.5000000000)
        ---(5)TRACE-------s---length'list'1
           ---TRACE---object=(B 3.5000000000)
          ---(6)TRACE-------s---length'list'1
             ---TRACE---object=(3.5000000000)
            ---(7)TRACE-------s---length'list'1
               ---TRACE---object=NIL
            ---(7)TRACE---;---read'from'fix'value(0) = 0
          ---(6)TRACE---;---read'from'fix'value (1 + get'fix'value(length'list'1(cdr'o(object)))) = 1
        ---(5)TRACE---;---read'from'fix'value (1 + get'fix'value(length'list'1(cdr'o(object)))) = 2
      ---(4)TRACE---;---read'from'fix'value (1 + get'fix'value(length'list'1(cdr'o(object)))) = 3
    ---(3)TRACE---;---read'from'fix'value (1 + get'fix'value(length'list'1(cdr'o(object)))) = 4
  ---(2)TRACE---;---read'from'fix'value (1 + get'fix'value(length'list'1(cdr'o(object)))) = 5
 ---(1)TRACE---;---read'from'fix'value (1 + get'fix'value(length'list'1(cdr'o(object)))) = 6
length'1 =6
read'from'fix'value(i)=1
read'from'fix'value(i)=2
read'from'fix'value(i)=3
read'from'fix'value(i)=4
read'from'fix'value(i)=5
read'from'fix'value(i)=6
length'2 =6
```

2.3 処理速度を速くするためのキーポイント

本ライブラリのマクロ quote_o は Common Lisp の quote とは違い、コンパイル時に評価されずに実行時に評価される。そのため、quote_o がループの中にあると処理速度が遅くなってしまう。それを防ぐには、quote_o をループの外へ出すと速くなる。

```
dolist_o(obj, object, {
      ..............
      x = quote_o((a b c));
      ..............
});
```

```
 y = quote_o((a b c));
dolist_o(obj, object, {
      ..............
      x = y;
      ..............
});
```

2.4 ユーザ固有の構造体を定義する方法

ユーザ固有の OBJECT 型の構造体[3]を定義するときの手順を以下に示す。

1. 変数の宣言

(a) 型名を typedef する。(本節の説明で出てくる『XXXXX』はこの型名を意味する。)

```
/*** TARGET を typedef している ***/
typedef struct Target TARGET;
```

(b) ユーザ固有の構造体を定義する。(宣言した構造体のメンバーを定義する) 構造体の先頭には、OBJECT_MEMBER を入れる (セミコロンなし)

```
struct Target {
     OBJECT_MEMBER
     OBJECT *target;
     OBJECT *alternative;
     OBJECT *examples;
     OBJECT *local_dic;
};
```

(c) ユーザ固有の構造体に関係するシステムの変数及び関数をマクロ p_define_alloc_object_area を使って宣言する。

```
p_define_alloc_object_area(TARGET);
```

本マクロにより、

```
EXTERN OBJECT *type##_MEMORY_AREA;
EXTERN OBJECT *type##_SYSTEM_MEMORY_AREA;
type *alloc_##type();
OBJECT *make_default_##type##_MEMORY_AREA(OBJECT*);
OBJECT *set_default_##type##_MEMORY_AREA(void);
OBJECT *type##_p();
EXTERN P_int type##_TYPE;
OBJECT *get_##TYPE##_AREA(void);
OBJECT *set_##TYPE##_AREA(OBJECT *area);
/*** (XXXXX は typedef で宣言した型名) ***/
```

[3]システムで定義されている構造体には、SYMBOL, LIST, STRING, NUMBER などがある。

が宣言される。

(d) ユーザ固有の構造体を作る関数を宣言する。

```
OBJECT *make_target ();
```

(e) ユーザ固有の構造体の print_function を宣言する。

```
OBJECT *target_print ();
```

2. 関数の定義

(a) ユーザ固有の構造体の print_function を定義する。

```
OBJECT *target_print (target)
OBJECT *target;
{
     TARGET *w_target;
     OBJECT *car_examples;
     w_target = (TARGET *)target;
     car_examples = car_o(w_target->examples);
     printf("#<");print(w_target->target);printf(" ");
     print(car_examples);printf("...>");
     return(target);
}
```

(b) ユーザ固有の構造体に関係するシステムの関数をマクロ define_alloc_object_function を使って定義する。

本マクロにより、関数 XXXXX *alloc_XXXXX() と OBJECT *set_default_XXXXX_MEMORY_AREA() と OBJECT *make_XXXXX_AREA(int size) と OBJECT *type_p(OBJECT *object) OBJECT *clear_XXXXX_AREA() と OBJECT *get_XXXXX_AREA() と OBJECT *set_XXXXX_AREA(OBJECT *area) とが定義される。

/*** (XXXXX は型名) ***/

```
define_alloc_object_function(TARGET, target_print);
```

(c) ユーザ固有の構造体を作る関数を関数 alloc_XXXXX() を使い定義する。

```
OBJECT *make_target (OBJECT *in_target, OBJECT *alternative,
                     OBJECT *examples, OBJECT *local_dic)
{
     TARGET *target;
     target = alloc_TARGET();
     target->target = in_target;
     target->alternative = alternative;
     target->examples = examples;
     target->local_dic = local_dic;
     return((OBJECT *)target);
}
```

3. 関数の初期化

定義された構造体をアロケーションしたり、使用する前に、set_default_XXXXX_MEMORY_AREA を実行しなければならない。

```
set_default_TARGET_MEMORY_AREA();
```

4. メモリの再利用

アロケーションしたメモリを clear_XXXXX_AREA を使って再利用することが出来る。

```
OBJECT *clear_XXXXX_AREA()
```

第 3 章

システム設計

本章では、本ライブラリの外部仕様 (本ライブラリの使用者が意識する箇所の仕様) を説明する。

3.1 基本的設計思想

『LISP プログラマーが C 言語でリスト処理プログラムを容易に作成できるライブラリにする』ということが基本的にある。

3.2 関数と引数

Common Lisp と同機能、同一インタフェイスになるように努めてある。

3.3 メモリ管理

処理時間を節約するため、 push_o や list_o などでアロケーションされるメモリを、関数 make_memory_area で予め、確保することができる。ここで、宣言したメモリを全て使ってしまうと、自動的に宣言した数だけ、更に、アロケーションするようになっている。

また、関数 clear_memory_area を使えば、マクロ with_set_memory_area でアロケーションしたメモリを再利用することができる。

```
OBJECT *memory_area = make_memory_area(100);

with_set_memory_area(memory_area, {
     ..................
     push_o, list_o などのリスト処理関数
     ..................
     clear_memory_area();
     ..................
})
```

3.4 t と nil(T と NULL)

Common Lisp であらかじめ用意されているアトムである t と nil には、グローバル変数の T と NULL が対応している。

表 3.1: t と nil(T と NULL)

Lisp	C
t	T
nil	NULL

3.5 型の統一

本ライブラリで使用する関数の引数やリターン値の型でプログラマを悩ませないように、OBJECT * 型に統一されている。

3.6 プリント関数

本ライブラリで使用するデータは Common Lisp と同様すべてプリント関数をもつ。そのため、データの内容を関数 print_o 1 つだけで、標準出力へ表示できる。そのため、デバッグが容易である。

第 4 章

プログラム設計

本章では、LinC の内部仕様を説明する。

4.1 本ライブラリで使用する主な構造体

本ライブラリでは、Common Lisp と同様に、数字、文字列などもすべて構造体として持っている。

そこで、使用している主な構造体を以下に示す。

1. SYMBOL

```
struct Symbol {
        OBJECT_MEMBER
        OBJECT *name;
        OBJECT *value;
        OBJECT *package;
        OBJECT *(*function)();
        OBJECT *p_list;
};
```

2. LIST

```
struct List {
        OBJECT_MEMBER
        OBJECT   *car;
        OBJECT *cdr;
};
```

3. STRING

```
struct String {
     OBJECT_MEMBER
  char *string;
};
```

4. NUMBER

```
struct Number {
     OBJECT_MEMBER
  union body{
       int fix;
       float flt;
  } value;
};
```

4.2 マクロの利用

関数で記述できる箇所[1]でも、処理時間を短くするためにマクロで記述してある。

[1] マクロ alloc_memory など

第 5 章

コーディングの基本方針

本表示システムのプログラムをコーディングするに当たり、決めた(標準化した)ことを以下に示す。

5.1 ネーミングコンベンション

ネーミングの標準化を以下のように決めた。

1. デファインマクロで定義した変数
   すべて大文字で、区切りはアンダースコアー"_"。

```
例:
 #define ACTIVE_ON 1
```

2. typedef した変数名
   すべて大文字で、区切りは"_"。

```
例:
 typedef struct Symbol SYMBOL;
```

3. 構造体名
   先頭または"_" のつぎの 1 文字が大文字で、その他は小文字。区切りは"_"。

```
例:
struct Memory_Area {
..............
};
```

4. 関数名

全て、小文字。

(a) Common Lisp と同機能で引数とリターン値の型が OBJECT の場合

変数名の語尾に"_o" をつける

例:

```
car_o
```

(b) Common Lisp と同機能でリターン値の型が OBJECT で、引数が OBJECT でないの場合

変数名の語尾に"_i" をつける

例:

```
nth_i
```

(c) Common Lisp と同機能でリターン値の型が int で、引数が OBJECT の場合

変数名の語尾に"_d" をつける

例:

```
position_d
```

(d) Common Lisp と同機能でリターン値の型が char で、引数が OBJECT の場合

変数名の語尾に"_s" をつける

例:

```
string_s
```

第 6 章

ファイル構成

ファイル構成とその内容を表 6.1に示す。

表 6.1: ファイル構成

ファイル名	内容
ヘッダー	
lisp_like_function.h	リスト処理関数本体の関数の宣言及びマクロ定義
machines.h	マシン固有部のインターフェイス
本体	
lisp_like_function.c	リスト処理本体
read_s_exp.y	Lisp の s 式の読みとり (文法記述)
read_s_exp.l	Lisp の s 式の読みとり (語彙部記述)
lisp_number.c	数字の英語表示
init_lisp_symbol_s.c	シンボルの初期化
machines_unix.c	マシン固有部 (UNIX)
machines_win32.c	マシン固有部 (Windows95)
サンプルプログラム	
sample_program.c	本ライブラリの関数やマクロを利用したサンプルプログラム
メイクファイル	
Makefile	本ライブラリとサンプルプログラムをメイクする Make ファイル
MakefileSample	サンプルプログラムをメイクする Make ファイル

第 7 章

関数の説明

本ライブラリの関数またはマクロはCommon Lispのものとほとんど一対一に対応する。それゆえ、本ライブラリの関数またはマクロがどのCommon Lisp関数に対応するかを示すことで、関数の説明とすることにする。

ただし、それだけでは不十分だと思われるものについては捕捉をつけるものとする。

OBJECT *init_lisp_like_function *()* [*Function*]

OBJECT *LISP_return_o *(OBJECT *value)* [*Function*]

OBJECT *abs_o *(OBJECT *number)* [*Function*]

OBJECT *acons_o *(OBJECT *key, OBJECT *datum, OBJECT *a_list)* [*Function*]

OBJECT *active_p_check_function *(OBJECT *object)* [*Function*]

---

char *add_escape *(char *string)* [*Function*]

---

char *add_escape2 *(char *string)* [*Function*]

OBJECT *adjoin_o *(OBJECT *item, OBJECT *list, OBJECT *in_test, OBJECT *in_key_function)* [*Function*]

---

void *alloc_char_memory *(MEMORY_AREA *memory_area, int alloc_size)* [*Function*]

OBJECT **alloc_hash_item_s *(int hash_size)* [*Function*]

HASH_TABLE *alloc_hash_table *(int hash_size)* [*Function*]

MEMORY_AREA *alloc_memory_area *(void)* [*Function*]

OBJECT *alloc_object_memory_o *(int type)* [*Function*]

OBJECT *alpha_char_p_o *(OBJECT *character)* [*Function*]

OBJECT *and_p_o *(OBJECT *s_exp)* [*Function*]

OBJECT *append2_o *(OBJECT *list_1, OBJECT *list_2)* [*Function*]

OBJECT *append3_o *(OBJECT *list_1, OBJECT *list_2, OBJECT *list_3)* [*Function*]

OBJECT *append_array_o *(OBJECT *array1, OBJECT *array2)* [*Function*]

OBJECT *append_o *(OBJECT *first, ...)* [*Function*]

char *append_string *(char *s, char *t)* [*Function*]

OBJECT *append_string_o *(OBJECT *s, OBJECT *t)* [*Function*]

OBJECT *apply_o *(OBJECT *func_symbol, ...)* [*Function*]

OBJECT *aref2_o *(OBJECT *array, OBJECT *arg1)* [*Function*]

OBJECT *aref3_o *(OBJECT *array, OBJECT *arg1, OBJECT *arg2)* [*Function*]

OBJECT *aref_array *(OBJECT *array, OBJECT **args)* [*Function*]

OBJECT *aref_o *(OBJECT *array, ...)* [*Function*]

OBJECT *aref_string *(OBJECT *array, OBJECT **args)* [*Function*]

OBJECT *array_dimension_o *(OBJECT *array, OBJECT *axis_number)* [*Function*]

OBJECT *array_dimensions_o *(OBJECT *array)* [*Function*]

OBJECT *arrayp_o *(OBJECT *object)* [*Function*]

OBJECT *ash_o *(OBJECT *integer, OBJECT *count)* [*Function*]

OBJECT *assoc_function2_o *(OBJECT *obj, OBJECT *a_list)* [*Function*]

OBJECT *assoc_o *(register OBJECT *obj, register OBJECT *a_list, OBJECT *in_test, OBJECT *in_key_function)* [*Function*]

OBJECT *atom_o *(OBJECT *object)* [*Function*]

OBJECT *both_car_comp_greater *(OBJECT *object1, OBJECT *object2)* [*Function*]

OBJECT *bq_append1_o *(OBJECT *object1)* [*Function*]

OBJECT *bq_append2_o *(OBJECT *object1, OBJECT *object2)* [*Function*]

OBJECT *bq_append3_o *(OBJECT *object1, OBJECT *object2, OBJECT *object3)* [*Function*]

OBJECT *bq_list1_o *(OBJECT *object1)* [*Function*]

OBJECT *bq_list2_o *(OBJECT *object1, OBJECT *object2)* [*Function*]

OBJECT *bq_list3_o *(OBJECT *object1, OBJECT *object2, OBJECT *object3)* [*Function*]

OBJECT *bq_list4_o *(OBJECT *object1, OBJECT *object2, OBJECT *object3, OBJECT *object4)* [*Function*]

OBJECT *bq_list_ast2_o *(OBJECT *object1, OBJECT *object2)* [*Function*]

OBJECT *bq_list_ast3_o *(OBJECT *object1, OBJECT *object2, OBJECT *object3)* [*Function*]

OBJECT *bq_list_ast4_o *(OBJECT *object1, OBJECT *object2, OBJECT *object3, OBJECT *object4)* [*Function*]

OBJECT *bq_list_ast5_o *(OBJECT *object1, OBJECT *object2, OBJECT *object3, OBJECT *object4, OBJECT *object5)* [*Function*]

OBJECT *butlast_o *(OBJECT *object)* [*Function*]

---

static int calc_gcd *(int x, int y)* [*Function*]

OBJECT *car_comp_greater *(OBJECT *object1, OBJECT *object2)* [*Function*]

OBJECT *car_comp_greater_equal *(OBJECT *object1, OBJECT *object2)* [*Function*]

OBJECT *car_o *(OBJECT *list)* [*Function*]

OBJECT *car_string_equal *(OBJECT *a, OBJECT *b)* [*Function*]

OBJECT *catch_o_internal *(OBJECT *tag, OBJECT *result, int TRACE_ON)* [*Function*]

OBJECT *catch_throw_cancel_p *(OBJECT *catch_throw_control)* [*Function*]

OBJECT *catch_throw_throw_p *(OBJECT *catch_throw_control)* [*Function*]

OBJECT *cdr_o *(OBJECT *list)* [*Function*]

OBJECT *char_bits_o *(OBJECT *character)* [*Function*]

int char_code_d *(OBJECT *character)* [*Function*]

OBJECT *char_code_o *(OBJECT *character)* [*Function*]

OBJECT *char_downcase_o *(OBJECT *character)* [*Function*]

OBJECT *char_font_o *(OBJECT *character)* [*Function*]

OBJECT *char_list_to_string_o *(OBJECT *list)* [*Function*]

OBJECT *char_o *(OBJECT *string, OBJECT *index)* [*Function*]

OBJECT *char_to_string *(OBJECT *ch)* [*Function*]

OBJECT *char_upcase_o *(OBJECT *character)* [*Function*]

OBJECT *character_i *(int code)* [*Function*]

OBJECT *character_o *(OBJECT *code)* [*Function*]

OBJECT *characterp_o *(OBJECT *object)* [*Function*]

OBJECT *clear_char_AREA *(void)* [*Function*]

OBJECT *close_o *(OBJECT *stream)* [*Function*]

void clrhash *(register Hash ***h_tbl_p)* [*Function*]

OBJECT *clrhash_o *(OBJECT *hash_table)* [*Function*]

OBJECT *code_char_i *(int code)* [*Function*]

OBJECT *code_char_o *(OBJECT *code, OBJECT *bit, OBJECT *font)* [*Function*]

OBJECT *coerce_o *(OBJECT *object, OBJECT *result_type)* [*Function*]

OBJECT *comp_caar_greater *(OBJECT *object1, OBJECT *object2)* [*Function*]

OBJECT *comp_equal_o *(OBJECT *object1, OBJECT *object2)* [*Function*]

OBJECT *comp_greater2_o *(OBJECT *object1, OBJECT *object2)* [*Function*]

OBJECT *comp_greater_equal2_o *(OBJECT *object1, OBJECT *object2)* [*Function*]

OBJECT *comp_greater_equal_o *(OBJECT *first, ...)* [*Function*]

OBJECT *comp_greater_o *(OBJECT *first, ...)* [*Function*]

OBJECT *comp_less2_o *(OBJECT *object1, OBJECT *object2)* [*Function*]

OBJECT *comp_less_equal2_o *(OBJECT *object1, OBJECT *object2)* [*Function*]

OBJECT *comp_less_equal_o *(OBJECT *first, ...)* [*Function*]

OBJECT *comp_less_o *(OBJECT *first, ...)* [*Function*]

OBJECT *comp_not_equal_o *(OBJECT *object1, OBJECT *object2)* [*Function*]

OBJECT *compile_file_o *(OBJECT *file_name)* [*Function*]

OBJECT *concatenate2_o *(OBJECT *output_type_spec, OBJECT *object)* [*Function*]

OBJECT *concatenate3_o *(OBJECT *output_type_spec, OBJECT *object1, OBJECT *object2)* [*Function*]

OBJECT *concatenate4_o *(OBJECT *output_type_spec, OBJECT *object1, OBJECT *object2, OBJECT *object3)* [*Function*]

OBJECT *concatenate_o *(OBJECT *output_type_spec, OBJECT *rest)* [*Function*]

---

char* concatenate_string *(char* first, ...)* [*Function*]

OBJECT *cons_o *(OBJECT *obj, OBJECT *list)* [*Function*]

OBJECT *consp_o *(OBJECT *object)* [*Function*]

OBJECT *copy_STRING *(OBJECT *object)* [*Function*]

OBJECT *copy_list_o *(OBJECT *object)* [*Function*]

OBJECT *copy_seq_o *(OBJECT *sequence)* [*Function*]

OBJECT *copy_string_o *(OBJECT *object)* [*Function*]

OBJECT *copy_tree_all *(OBJECT *object)* [*Function*]

OBJECT *copy_tree_o *(OBJECT *object)* [*Function*]

---

int count_back_slash *(char *string)* [*Function*]

---

int count_escape *(char *string)* [*Function*]

OBJECT *count_o *(OBJECT *obj, OBJECT *list, OBJECT *in_test)* [*Function*]

OBJECT *decf_o *(OBJECT **number, OBJECT *delta)* [*Function*]

OBJECT *declare_o *(OBJECT *object)* [*Function*]

OBJECT *default_key_function *(OBJECT *object)* [*Function*]

OBJECT *default_stream *(OBJECT *stream)* [*Function*]

#define define_char_comp_cond_function *(NAME, FUNCTION)* [*Function*]

#define define_char_cond_function *(NAME, FUNCTION)* [*Function*]

#define define_cxxr_function *(x1, x2)* [*Function*]

#define define_cxxxr_function *(x1, x2, x3)* [*Function*]

#define define_cxxxxr_function *(x1, x2, x3, x4)* [*Function*]

#define define_delete_function *(FUNC_TYPE, ARGS, ARGS_INTERNAL, CONDITION_OF_STRING, CONDITION_OF_LIST)* [*Function*]

#define define_position_function *(FUNC_TYPE, ARGS, ARGS_INTERNAL, CONDITION_OF_STRING, CONDITION_OF_LIST)* [*Function*]

#define define_string_cond_function *(NAME, FUNCTION, TYPE)* [*Function*]

---

char *del_back_slash *(char *string)* [*Function*]

---

char *del_escape *(char *string)* [*Function*]

OBJECT *delete_duplicates_internal_o *(OBJECT *list, OBJECT *test, OBJECT *key)* [*Function*]

OBJECT *delete_duplicates_o *(OBJECT *list, OBJECT *test, OBJECT *key)* [*Function*]

OBJECT *delete_list_o *(register OBJECT *obj, register OBJECT *list, OBJECT *in_test, OBJECT *in_key_function)* [*Function*]

OBJECT *delete_o *(OBJECT *obj, OBJECT *sequence, OBJECT *in_test, OB-*

*JECT *in_key_function)* [*Function*]

OBJECT *delete_string_o *(OBJECT *obj, OBJECT *string_obj, OBJECT *in_test, OBJECT *in_key_function)* [*Function*]

OBJECT *digit_char_p_o *(OBJECT *char1, OBJECT *radix)* [*Function*]

---

dir_error *(char *path)* [*Function*]

OBJECT *display_lisp_memory_area *(void)* [*Function*]

OBJECT *divide2_o *(OBJECT *number1, OBJECT *number2)* [*Function*]

OBJECT *divide3_o *(OBJECT *number1, OBJECT *number2, OBJECT *number3)* [*Function*]

OBJECT *divide_o *(OBJECT *number1, OBJECT *number2)* [*Function*]

---

float double_to_float *(double a1)* [*Function*]

OBJECT *eighth_o *(OBJECT *list)* [*Function*]

OBJECT *elt_o *(OBJECT *sequence, OBJECT *index)* [*Function*]

OBJECT *enough_namestring_o *(OBJECT *pathname, OBJECT *default_pathname_defaults)* [*Function*]

OBJECT *eq_o *(void *a, void *b)* [*Function*]

OBJECT *eql_o *(OBJECT *object1, OBJECT *object2)* [*Function*]

OBJECT *equal_o *(register OBJECT *object1, register OBJECT *object2)* [*Function*]

OBJECT *equalp_o *(OBJECT *object1, OBJECT *object2)* [*Function*]

OBJECT *error_o *(OBJECT *msg, OBJECT *rest)* [*Function*]

int error_trap *(char *message)* [*Function*]

OBJECT *escape_char_p *(char ch)* [*Function*]

OBJECT *eval_and_o *(OBJECT *s_exp)* [*Function*]

OBJECT *eval_atom_o *(OBJECT *atom)* [*Function*]

OBJECT *eval_func_o *(OBJECT *s_exp)* [*Function*]

OBJECT *eval_function_o *(OBJECT *s_exp)* [*Function*]

OBJECT *eval_o *(OBJECT *s_exp)* [*Function*]

OBJECT *eval_or_o *(OBJECT *s_exp)* [*Function*]

OBJECT *eval_quote_o *(OBJECT *s_exp)* [*Function*]

OBJECT *every_o *(OBJECT *object, OBJECT *list)* [*Function*]

OBJECT *every_o_1 *(OBJECT *function, OBJECT *list)* [*Function*]

OBJECT *exp_o *(OBJECT *object)* [*Function*]

OBJECT *export_o *(OBJECT *symbols, OBJECT *in_package)* [*Function*]

OBJECT *expt_o *(OBJECT *a, OBJECT *b)* [*Function*]

OBJECT *fifth_o *(OBJECT *list)* [*Function*]

OBJECT *file_namestring_o *(OBJECT *pathname)* [*Function*]

OBJECT *file_write_date_o *(OBJECT *file_name)* [*Function*]

OBJECT *find_keyword_pair *(OBJECT *original_arg, OBJECT *key_args)* [

Function]

OBJECT *find_list_o *(register OBJECT *obj, register OBJECT *list, OBJECT *in_test, register OBJECT *in_key_function)* [*Function*]

OBJECT *find_o *(register OBJECT *obj, register OBJECT *sequence, OBJECT *in_test, register OBJECT *in_key_function)* [*Function*]

OBJECT *find_package_o *(OBJECT *package_name_object)* [*Function*]

OBJECT *find_string_o *(register OBJECT *obj, register OBJECT *string_obj, OBJECT *in_test, register OBJECT *in_key_function)* [*Function*]

OBJECT *finish_output_o *(OBJECT *stream)* [*Function*]

OBJECT *first_o *(OBJECT *list)* [*Function*]

OBJECT *first_object *(OBJECT *object)* [*Function*]

OBJECT *float_o *(OBJECT *object)* [*Function*]

OBJECT *format_o *(OBJECT *stream, OBJECT *format, OBJECT *arg_list)* [*Function*]

OBJECT *format_radix_o *(OBJECT *arg, OBJECT *stream, OBJECT *cardinal_flg)* [*Function*]

OBJECT *fourth_o *(OBJECT *list)* [*Function*]

OBJECT *free_object *(OBJECT *obj)* [*Function*]

OBJECT *fs_fflush *(OBJECT *stream)* [*Function*]

OBJECT *fs_printf *(OBJECT *stream_obj, char *format, ...)* [*Function*]

OBJECT *fs_tab_printf *(OBJECT *stream, int space_number)* [*Function*]

OBJECT *funcall10_o *(OBJECT *(*func)(void), OBJECT *arg1, OBJECT *arg2, OBJECT *arg3, OBJECT *arg4, OBJECT *arg5, OBJECT *arg6, OBJECT *arg7, OBJECT *arg8, OBJECT *arg9)* [*Function*]

OBJECT *funcall11_o *(OBJECT *(*func)(void), OBJECT *arg1, OBJECT *arg2, OBJECT *arg3, OBJECT *arg4, OBJECT *arg5, OBJECT *arg6, OBJECT *arg7, OBJECT *arg8, OBJECT *arg9, OBJECT *arg10)* [*Function*]

OBJECT *funcall2_o *(OBJECT *object, OBJECT *arg1)* [*Function*]

OBJECT *funcall2_o_1 *(OBJECT *func, OBJECT *arg1)* [*Function*]

OBJECT *funcall3_o *(OBJECT *object, OBJECT *arg1, OBJECT *arg2)* [*Function*]

OBJECT *funcall4_o *(OBJECT *(*func)(void), OBJECT *arg1, OBJECT *arg2, OBJECT *arg3)* [*Function*]

OBJECT *funcall5_o *(OBJECT *(*func)(void), OBJECT *arg1, OBJECT *arg2, OBJECT *arg3, OBJECT *arg4)* [*Function*]

OBJECT *funcall6_o *(OBJECT *(*func)(void), OBJECT *arg1, OBJECT *arg2, OBJECT *arg3, OBJECT *arg4, OBJECT *arg5)* [*Function*]

OBJECT *funcall7_o *(OBJECT *(*func)(void), OBJECT *arg1, OBJECT *arg2, OBJECT *arg3, OBJECT *arg4, OBJECT *arg5, OBJECT *arg6)* [*Function*]

OBJECT *funcall8_o *(OBJECT *(*func)(void), OBJECT *arg1, OBJECT *arg2, OBJECT *arg3, OBJECT *arg4, OBJECT *arg5, OBJECT *arg6, OBJECT *arg7)* [*Function*]

OBJECT *funcall9_o *(OBJECT *(*func)(void), OBJECT *arg1, OBJECT *arg2, OBJECT *arg3, OBJECT *arg4, OBJECT *arg5, OBJECT *arg6, OBJECT *arg7, OBJECT *arg8)* [*Function*]

OBJECT *funcall_o *(OBJECT *func, ...)* [*Function*]

OBJECT *function_call *(OBJECT *function, OBJECT **object, int argno)* [*Function*]

OBJECT *function_p_o *(OBJECT *s_exp)* [*Function*]

OBJECT *functionp_o *(OBJECT *object)* [*Function*]

OBJECT *get_char_AREA *(void)* [*Function*]

int get_fix_value *(OBJECT *number)* [*Function*]

float get_float_value *(OBJECT *number)* [*Function*]

OBJECT *get_function_args_info *(OBJECT *function_name)* [*Function*]

OBJECT *get_global_value_o *(OBJECT *global_value_stack)* [*Function*]

OBJECT *get_o *(OBJECT *symbol, OBJECT *indicator, OBJECT *default1)* [*Function*]

OBJECT *get_output_stream_to_string_o *(OBJECT *stream)* [*Function*]

OBJECT *get_symbol *(void *string, Hash **symbol_table)* [*Function*]

OBJECT *get_symbol_from_package *(char *string, OBJECT *package)* [*Function*]

float get_wall_time *(struct timeval *start, struct timeval *end)* [*Function*]

void *gethash *(void *key, Hash *h_tbl[])* [*Function*]

OBJECT *gethash_o *(OBJECT *key, OBJECT *hash_table, OBJECT *default_value)* [*Function*]

int hash_atom_fn *(OBJECT *key)* [*Function*]

int hash_fix_number_fn *(OBJECT *object)* [*Function*]

int hash_float_number_fn *(OBJECT *object)* [*Function*]

int hash_fraction_number_fn *(OBJECT *object)* [*Function*]

---

int hash_object_eql_fn *(OBJECT *key)* [*Function*]

---

int hash_object_equal_fn *(OBJECT *key)* [*Function*]

---

int hash_object_fn *(register OBJECT *key, OBJECT *function, OBJECT *hash_table)*
[*Function*]

---

int hash_pointer_fn *(OBJECT *object)* [*Function*]

---

int hash_string_func *(void* ptr, int size)* [*Function*]

---

int hash_string_func_internal *(void* ptr)* [*Function*]

OBJECT *hash_table_count_o *(OBJECT *hash_table)* [*Function*]

OBJECT *hash_table_item_list_o *(OBJECT *number, OBJECT *hash_table)* [
Function]

---

int hash_table_size_d *(OBJECT *hash_table)* [*Function*]

OBJECT *hash_table_size_o *(OBJECT *hash_table)* [*Function*]

OBJECT *identity_o *(OBJECT *obj)* [*Function*]

OBJECT *ignore1_o *(OBJECT *object)* [*Function*]

OBJECT *ignore2_o *(OBJECT *object1, OBJECT *object2)* [*Function*]

OBJECT *ignore3_o *(OBJECT *object1, OBJECT *object2, OBJECT *object3)* [*Function*]

OBJECT *import_o *(OBJECT *symbols, OBJECT *in_package)* [*Function*]

OBJECT *incf_o *(OBJECT **number, OBJECT *delta)* [*Function*]

OBJECT *init_lisp_like_function *(void)* [*Function*]

OBJECT *init_lisp_memory_area *(void)* [*Function*]

OBJECT *init_memory_area *(MEMORY_AREA *memory_area)* [*Function*]

OBJECT *init_object_memory_area *(MEMORY_AREA *memory_area)* [*Function*]

OBJECT *integerp_o *(OBJECT *number)* [*Function*]

OBJECT *intersection_o *(OBJECT *list_1, OBJECT *list_2, register OBJECT *test, register OBJECT *key)* [*Function*]

---

int key_cmp *(void *target, void *source)* [*Function*]

---

int key_cpy *(void *target, void *source)* [*Function*]

---

int key_size *(void *key)* [*Function*]

OBJECT *keywordp_o *(OBJECT *object)* [*Function*]

OBJECT *lambda_member_7_1 *(OBJECT *a, OBJECT *b)* [*Function*]

OBJECT *last_o *(OBJECT *list)* [*Function*]

int length_array_d *(OBJECT *array)* [*Function*]

OBJECT *length_array_o *(OBJECT *array)* [*Function*]

int length_d *(OBJECT *object)* [*Function*]

int length_list_d *(OBJECT *list)* [*Function*]

OBJECT *length_o *(OBJECT *object)* [*Function*]

int length_string_d *(OBJECT *object)* [*Function*]

OBJECT *list0_o *(void)* [*Function*]

OBJECT *list1_o *(void *object)* [*Function*]

OBJECT *list2_o *(OBJECT *object1, OBJECT *object2)* [*Function*]

OBJECT *list3_o *(OBJECT *object1, OBJECT *object2, OBJECT *object3)* [*Function*]

OBJECT *list4_o *(OBJECT *object1, OBJECT *object2, OBJECT *object3, OBJECT *object4)* [*Function*]

OBJECT *list5_o *(OBJECT *object1, OBJECT *object2, OBJECT *object3, OBJECT *object4, OBJECT *object5)* [*Function*]

OBJECT *list_o *(OBJECT *first, ...)* [*Function*]

OBJECT *listp_car *(OBJECT *object)* [*Function*]

OBJECT *listp_car_eq *(OBJECT *object1, OBJECT *object2)* [*Function*]

OBJECT *listp_car_eql *(OBJECT *object1, OBJECT *object2)* [*Function*]

OBJECT *listp_car_equal *(OBJECT *object1, OBJECT *object2)* [*Function*]

OBJECT *listp_cdr_equal *(OBJECT *object1, OBJECT *object2)* [*Function*]

OBJECT *listp_o *(OBJECT *object)* [*Function*]

OBJECT *load_o *(OBJECT *file_name)* [*Function*]

OBJECT *log_o *(OBJECT *object)* [*Function*]

OBJECT *logand_o *(OBJECT *integer1, OBJECT *integer2)* [*Function*]

OBJECT *logbitp_o *(OBJECT *index, OBJECT *integer)* [*Function*]

OBJECT *logior_o *(OBJECT *integers)* [*Function*]

OBJECT *make_CATCH_THROW_CONTROL *(int return_reason, OBJECT *return_value)* [*Function*]

OBJECT *make_STRING *(char *char_string)* [*Function*]

OBJECT *make_SYMBOL *(char *string, OBJECT *package)* [*Function*]

OBJECT *make_arg_list *(OBJECT *function_name, OBJECT *args)* [*Function*]

OBJECT *make_array_o *(OBJECT *dimensions, OBJECT *initial_element, OBJECT *initial_contents)* [*Function*]

OBJECT *make_char_AREA *(int size)* [*Function*]

OBJECT *make_char_array *(void)* [*Function*]

OBJECT *make_closure *(OBJECT *function, OBJECT *s_exp, OBJECT *set_value_info)*
[*Function*]

OBJECT *make_fraction *(long int num, long int den)* [*Function*]

OBJECT *make_function *(char *func_name, void *function, OBJECT *lisp_args)*
[*Function*]

Hash **make_hash_table *(void)* [*Function*]

OBJECT *make_hash_table_o *(OBJECT *in_test, OBJECT *in_size)* [*Function*]

OBJECT *make_lisp_function *(char *string, void *function, OBJECT *lisp_args)*
[*Function*]

OBJECT *make_lisp_functions *(void)* [*Function*]

OBJECT *make_lisp_package *(void)* [*Function*]

OBJECT *make_lisp_variable *(char *string)* [*Function*]

OBJECT *make_lisp_variables *(void)* [*Function*]

OBJECT *make_list_i *(int length, OBJECT *initial_element)* [*Function*]

OBJECT *make_list_o *(OBJECT *length, OBJECT *initial_element)* [*Function*]

OBJECT *make_memory_area *(char *memory_name, int memory_size, int max_number, int memory_object_type, OBJECT *(*memory_print_function)(OBJECT*, OBJECT*, OBJECT*))* [*Function*]

OBJECT *make_number_array *(void)* [*Function*]

OBJECT *make_object_memory_area *(char *memory_name, int memory_object_type, OBJECT *(*memory_print_function)(OBJECT*, OBJECT*, OBJECT*), int memory_size, int max_number)* [*Function*]

OBJECT *make_package_o *(OBJECT *package_name_object, OBJECT *use)* [*Function*]

---

char *make_pathname *(char *dic, char *file)* [*Function*]

OBJECT *make_pathname_o *(OBJECT *directory, OBJECT *name, OBJECT *type)* [*Function*]

OBJECT *make_quote *(char *string, char *package_name)* [*Function*]

OBJECT *make_string_input_stream_o *(OBJECT *string)* [*Function*]

OBJECT *make_string_o *(OBJECT *size, OBJECT *initial_element)* [*Function*]

OBJECT *make_string_output_stream_o *(void)* [*Function*]

---

make_symbol_for_make_arg_list *(void)* [*Function*]

OBJECT *map_list_to_string_o *(OBJECT *function, OBJECT *list)* [*Function*]

OBJECT *map_o *(OBJECT *output_type_spec, OBJECT *function, OBJECT *sequence)* [*Function*]

OBJECT *map_string_to_list_o *(OBJECT *function, register OBJECT *string)* [*Function*]

OBJECT *mapc2_o *(OBJECT *object, OBJECT *list)* [*Function*]

OBJECT *mapc2_o_1 *(OBJECT *function, OBJECT *list)* [*Function*]

OBJECT *mapc3_o *(OBJECT *object, OBJECT *list1, OBJECT *list2)* [*Function*]

OBJECT *mapc3_o_1 *(OBJECT *function, OBJECT *list1, OBJECT *list2)* [*Function*]

OBJECT *mapc4_o *(OBJECT *object, OBJECT *list1, OBJECT *list2, OBJECT *list3)* [*Function*]

OBJECT *mapc4_o_1 *(OBJECT *function, OBJECT *list1, OBJECT *list2, OBJECT *list3)* [*Function*]

OBJECT *mapc_o *(OBJECT *function, OBJECT *list)* [*Function*]

OBJECT *mapcar2_o *(OBJECT *object, OBJECT *list)* [*Function*]

OBJECT *mapcar2_o_1 *(OBJECT *function, OBJECT *list)* [*Function*]

OBJECT *mapcar3_o *(OBJECT *object, OBJECT *list1, OBJECT *list2)* [*Function*]

OBJECT *mapcar3_o_1 *(OBJECT *function, OBJECT *list1, OBJECT *list2)* [*Function*]

OBJECT *mapcar_o *(OBJECT *function, ...)* [*Function*]

OBJECT *mapcar_o_1 *(OBJECT *function, OBJECT **list)* [*Function*]

OBJECT *maphash_o *(OBJECT *object, OBJECT *list)* [*Function*]

OBJECT *maphash_o_1 *(OBJECT *function, OBJECT *hash_table)* [*Function*]

OBJECT *maplist_o *(OBJECT *function, ...)* [*Function*]

OBJECT *maplist_o_1 *(OBJECT *function, OBJECT **list)* [*Function*]

OBJECT *match_char_string *(OBJECT *char_bag, OBJECT *ch)* [*Function*]

OBJECT *max2_o *(OBJECT *number1, OBJECT *number2)* [*Function*]

OBJECT *max3_o *(OBJECT *number1, OBJECT *number2, OBJECT *number3)* [*Function*]

OBJECT *max_o *(OBJECT *number, OBJECT *rest_numbers)* [*Function*]

OBJECT *member_o *(OBJECT *obj, OBJECT *list, OBJECT *in_test, OBJECT *in_key_function)* [*Function*]

OBJECT *merge_pathnames_o *(OBJECT *in_pathname, OBJECT *in_defaults)*
[*Function*]

OBJECT *min2_o *(OBJECT *number1, OBJECT *number2)* [*Function*]

OBJECT *min3_o *(OBJECT *number1, OBJECT *number2, OBJECT *number3)*
[*Function*]

OBJECT *min_o *(OBJECT *first, ...)* [*Function*]

OBJECT *minus2_o *(OBJECT *number1, OBJECT *number2)* [*Function*]

OBJECT *minus3_o *(OBJECT *number1, OBJECT *number2, OBJECT *number3)* [*Function*]

OBJECT *minus_o *(OBJECT *number1, OBJECT *number2)* [*Function*]

OBJECT *minusp_o *(OBJECT *number)* [*Function*]

OBJECT *mod_o *(OBJECT *number1, OBJECT *number2)* [*Function*]

#define multiple_value_setq2_object_int_o *(object1, object2, value_function)* [*Function*]

OBJECT *namestring_o *(OBJECT *pathname)* [*Function*]

OBJECT *nconc_o *(register OBJECT *list_1, OBJECT *list_2)* [*Function*]

OBJECT *ninth_o *(OBJECT *list)* [*Function*]

OBJECT *not_define_symbol_info *(char *string, OBJECT **value_variable_pt, OBJECT *(**function_variable_pt)(void))* [*Function*]

OBJECT *not_o *(OBJECT *object)* [*Function*]

OBJECT *nreverse_o *(OBJECT *list)* [*Function*]

OBJECT *nsublis_internal *(OBJECT *alist, OBJECT **tree, OBJECT *in_test, OBJECT *in_key_function)* [*Function*]

OBJECT *nsublis_o *(OBJECT *alist,* [*Function*]

OBJECT *nsubst_internal *(OBJECT *new_obj, register OBJECT *obj,* [*Function*]

OBJECT *nsubst_o *(OBJECT *new_obj, OBJECT *obj,* [*Function*]

OBJECT *nsubstitute_list_o *(register OBJECT *new_obj, register OBJECT *obj,* [*Function*]

OBJECT *nsubstitute_o *(OBJECT *new_obj, OBJECT *obj, OBJECT *sequence, OBJECT *in_test, OBJECT *in_key_function)* [*Function*]

OBJECT *nth_i *(int n, OBJECT *list)* [*Function*]

OBJECT *nth_o *(OBJECT *n_o, OBJECT *list)* [*Function*]

OBJECT *nthcdr_i *(int n, OBJECT *list)* [*Function*]

OBJECT *nthcdr_o *(OBJECT *n, OBJECT *list)* [*Function*]

OBJECT *null_o *(OBJECT *object)* [*Function*]

OBJECT *numberp_o *(OBJECT *object)* [*Function*]

OBJECT *oddp_o *(OBJECT *object)* [*Function*]

#define offset_m *(offset) (((offset) + WORDSIZE -1) & -WORDSIZE)* [*Function*]

OBJECT *one_minus_o *(OBJECT *number)* [*Function*]

OBJECT *one_plus_o *(OBJECT *number)* [*Function*]

OBJECT *open_o *(OBJECT *filename, OBJECT *direction, OBJECT *if_does_not_exists, OBJECT *if_exists)* [*Function*]

OBJECT *or_p_o *(OBJECT *s_exp)* [*Function*]

OBJECT *package_use_list_o *(OBJECT *package)* [*Function*]

OBJECT *parse_integer_o *(OBJECT *string, OBJECT *junk_allowed)* [*Function*]

OBJECT *pathname_directory_o *(OBJECT *pathname)* [*Function*]

OBJECT *pathname_name_o *(OBJECT *pathname)* [*Function*]

OBJECT *pathname_o *(OBJECT *object)* [*Function*]

OBJECT *pathname_string_o *(OBJECT *string)* [*Function*]

OBJECT *pathname_type_o *(OBJECT *pathname)* [*Function*]

OBJECT *pathnamep_o *(OBJECT *object)* [*Function*]

char *plain_string_s *(OBJECT *atom)* [*Function*]

OBJECT *plus2_o *(OBJECT *number1, OBJECT *number2)* [*Function*]

OBJECT *plus3_o *(OBJECT *number1, OBJECT *number2, OBJECT *number3)* [*Function*]

OBJECT *plus_o *(OBJECT *first, ...)* [*Function*]

OBJECT *plusp_o *(OBJECT *number)* [*Function*]

OBJECT *pop_global_value_o *(OBJECT **p_global_value_stack)* [*Function*]

OBJECT *pop_o *(OBJECT **list_p)* [*Function*]

---

int position_d *(void *obj, OBJECT *list, OBJECT *in_test)* [*Function*]

OBJECT *pprint_o *(OBJECT *object, OBJECT *stream)* [*Function*]

OBJECT *prin1_internal *(OBJECT *object, OBJECT *stream)* [*Function*]

OBJECT *prin1_o *(OBJECT *object, OBJECT *stream)* [*Function*]

OBJECT *princ_internal *(OBJECT *object, OBJECT *stream)* [*Function*]

OBJECT *princ_o *(OBJECT *object, OBJECT *stream)* [*Function*]

OBJECT *print *(OBJECT *object, OBJECT *stream, OBJECT* level)* [*Function*]

OBJECT *print_ARRAY *(OBJECT *array, OBJECT *stream, OBJECT *level)* [*Function*]

OBJECT *print_CATCH_THROW_CONTROL *(OBJECT *catch_throw_control, OBJECT *stream, OBJECT* level)* [*Function*]

OBJECT *print_CHARACTER *(OBJECT *character, OBJECT *stream, OBJECT *level)* [*Function*]

OBJECT *print_CLOSURE *(OBJECT *closure, OBJECT *stream, OBJECT *level)* [*Function*]

OBJECT *print_FRACTION *(OBJECT *fraction, OBJECT *stream, OBJECT *level)* [*Function*]

OBJECT *print_LIST *(OBJECT *list, OBJECT *stream, OBJECT *level)* [*Function*]

OBJECT *print_MEMORY_AREA *(OBJECT *memory_area, OBJECT *stream, OBJECT *level)* [*Function*]

OBJECT *print_NUMBER *(OBJECT *number, OBJECT *stream, OBJECT *level)* [*Function*]

OBJECT *print_PACKAGE *(OBJECT *package, OBJECT *stream, OBJECT *level)* [*Function*]

OBJECT *print_PATHNAME *(OBJECT *pathname, OBJECT *stream, OBJECT *level)* [*Function*]

OBJECT *print_STREAM *(OBJECT *stream, OBJECT *out_stream, OBJECT *level)* [*Function*]

OBJECT *print_STRING *(OBJECT *string, OBJECT *stream, OBJECT *level)* [*Function*]

OBJECT *print_SYMBOL *(OBJECT *symbol, OBJECT *stream, OBJECT* level)* [*Function*]

print_bits_d *(f)* [*Function*]

int print_detail_lisp_memory_area *(void)* [*Function*]

int print_detail_memory_area *(OBJECT *memory_area)* [*Function*]

OBJECT *print_hash_info *(OBJECT *hash_table)* [*Function*]

OBJECT *print_hash_table *(OBJECT *hash_table, OBJECT *stream, OBJECT* *level)* [*Function*]

OBJECT *print_list_internal *(OBJECT *list, OBJECT *stream)* [*Function*]

OBJECT *print_o *(OBJECT *object, OBJECT *stream)* [*Function*]

OBJECT *probe_file_o *(OBJECT *file_name)* [*Function*]

OBJECT *push_global_value_o *(OBJECT *value, OBJECT **p_global_value_stack)* [*Function*]

OBJECT *push_list_list *(OBJECT *obj, register OBJECT *list_list)* [*Function*]

OBJECT *push_o *(void *obj, OBJECT **list_p)* [*Function*]

OBJECT *pushnew_o *(OBJECT *object, OBJECT **listp, OBJECT *in_test)* [*Function*]

OBJECT *quote_p_o *(OBJECT *s_exp)* [*Function*]

OBJECT *rassoc_function2_o *(register OBJECT *obj, register OBJECT *a_list)* [*Function*]

OBJECT *rassoc_o *(OBJECT *obj, OBJECT *a_list, OBJECT *in_test, OBJECT *in_key_function)* [*Function*]

OBJECT *read_char_o *(OBJECT *stream, OBJECT *flg, void *eof_return_value)* [*Function*]

OBJECT *read_line_from_file *(OBJECT *stream, OBJECT *flg, OBJECT *eof_return_value)* [*Function*]

OBJECT *read_line_from_string *(OBJECT *stream, OBJECT *flg, OBJECT *eof_return_value)* [*Function*]

OBJECT *read_line_o *(OBJECT *stream, OBJECT *flg, OBJECT *eof_return_value)*

[*Function*]

OBJECT *reduce_o *(OBJECT *function, OBJECT *sequence)* [*Function*]

OBJECT *rem_o *(OBJECT *number1, OBJECT *number2)* [*Function*]

OBJECT *remove_duplicates_o *(OBJECT *list, OBJECT *test, OBJECT *key, OBJECT *from_end)* [*Function*]

OBJECT *remove_if_not_o *(OBJECT *in_test, OBJECT *sequence, OBJECT *in_key_function)* [*Function*]

OBJECT *remove_if_o *(OBJECT *in_test, OBJECT *sequence, OBJECT *in_key_function)* [*Function*]

OBJECT *remove_o *(OBJECT *obj, OBJECT *sequence, OBJECT *in_test, OBJECT *in_key_function)* [*Function*]

OBJECT *rest_o *(OBJECT *a)* [*Function*]

OBJECT *reverse_list_o *(OBJECT *list)* [*Function*]

OBJECT *reverse_o *(OBJECT *sequence)* [*Function*]

OBJECT *reverse_string_o *(OBJECT *string)* [*Function*]

OBJECT *rplaca_o *(OBJECT *list, OBJECT *object)* [*Function*]

OBJECT *rplacd_o *(OBJECT *list, OBJECT *object)* [*Function*]

OBJECT *search_list_o *(OBJECT *list1, OBJECT *list2, OBJECT *from_end, OBJECT *test, OBJECT *key, OBJECT *start1, OBJECT *start2, OBJECT *end1, OBJECT *end2)* [*Function*]

OBJECT *search_o *(OBJECT *sequence1, OBJECT *sequence2, OBJECT *from_end, OBJECT *test, OBJECT *key, OBJECT *start1, OBJECT *start2, OBJECT *end1, OBJECT *end2)* [*Function*]

OBJECT *search_string_o *(OBJECT *string1, OBJECT *string2, OBJECT *from_end, OBJECT *test, OBJECT *key, OBJECT *start1, OBJECT *start2, OBJECT *end1, OBJECT *end2)* [*Function*]

OBJECT *second_o *(OBJECT *list)* [*Function*]

OBJECT *second_object *(OBJECT *object)* [*Function*]

OBJECT *set_array_contents_internal *Function* [ ]

---

set_bit_array *(void)* [*Function*]

OBJECT *set_char_AREA *(OBJECT *area)* [*Function*]

OBJECT *set_difference_o *(OBJECT *list1, OBJECT *list2, OBJECT *in_test, OBJECT *in_key_function)* [*Function*]

OBJECT *set_function_args_info *(OBJECT *function_name, OBJECT *symbol_info)* [*Function*]

OBJECT *set_global_value_pointer *(OBJECT *global_value, OBJECT **value_variable_pt)* [*Function*]

---

void set_new_memory *(MEMORY_AREA *memory_area)* [*Function*]

OBJECT *set_o *(OBJECT *a, OBJECT *b)* [*Function*]

---

void *set_old_memory *(MEMORY_AREA *memory_area)* [*Function*]

OBJECT *set_symbol *(void* string, OBJECT *symbol, int state, Hash ***symbol_table)* [*Function*]

OBJECT *set_symbol_to_package *(char* string, OBJECT *symbol, int state, OBJECT *package)* [*Function*]

OBJECT *set_type_relation *(int type_id, char *type_name, OBJECT *type_list)* [*Function*]

OBJECT *set_type_relation_s *(void)* [*Function*]

OBJECT *set_value_closure *(OBJECT *closure, OBJECT *set_value_info)* [*Function*]

OBJECT *setf_aref_array_o *(OBJECT *array, OBJECT **args, int last_argno)* [*Function*]

OBJECT *setf_aref_o *(OBJECT *array, ...)* [*Function*]

OBJECT *setf_aref_string_o *(OBJECT *array, OBJECT **args)* [*Function*]

OBJECT *setf_cadr_list *(OBJECT *list, OBJECT *object)* [*Function*]

OBJECT *setf_car_o *(OBJECT *list, OBJECT *object)* [*Function*]

OBJECT *setf_cdr_o *(OBJECT *list, OBJECT *object)* [*Function*]

OBJECT *setf_eighth_o *(OBJECT *list, OBJECT *object)* [*Function*]

OBJECT *setf_elt_o *(OBJECT *sequence, OBJECT *index, OBJECT *object)* [*Function*]

OBJECT *setf_fifth_o *(OBJECT *list, OBJECT *object)* [*Function*]

OBJECT *setf_first_o *(OBJECT *list, OBJECT *object)* [*Function*]

OBJECT *setf_fourth_o *(OBJECT *list, OBJECT *object)* [*Function*]

OBJECT *setf_get_o *(OBJECT *symbol, OBJECT *indicator, OBJECT *value)* [*Function*]

OBJECT *setf_gethash_o *(OBJECT *key, OBJECT *hash_table, OBJECT *content)* [*Function*]

OBJECT *setf_global_value_o *(OBJECT *global_value_stack, OBJECT *value)* [*Function*]

OBJECT *setf_ninth_o *(OBJECT *list, OBJECT *object)* [*Function*]

OBJECT *setf_nth_i *(register int n, register OBJECT *list, OBJECT *object)* [*Function*]

OBJECT *setf_nth_o *(OBJECT *n_o, OBJECT *list, OBJECT *object)* [*Function*]

OBJECT *setf_second_o *(OBJECT *list, OBJECT *object)* [*Function*]

OBJECT *setf_seventh_o *(OBJECT *list, OBJECT *object)* [*Function*]

OBJECT *setf_sixth_o *(OBJECT *list, OBJECT *object)* [*Function*]

OBJECT *setf_symbol_value_o *(OBJECT *object, OBJECT *value)* [*Function*]

OBJECT *setf_tenth_o *(OBJECT *list, OBJECT *object)* [*Function*]

OBJECT *setf_third_o *(OBJECT *list, OBJECT *object)* [*Function*]

void sethash *(void *key, void *content, Hash ***h_tbl_p)* [*Function*]

OBJECT *sethash_o *(OBJECT *key, OBJECT *content, OBJECT *hash_table)* [*Function*]

OBJECT *seventh_o *(OBJECT *list)* [*Function*]

OBJECT *sixth_o *(OBJECT *list)* [*Function*]

---

void sort2_o_internal *(register OBJECT **list_array, register int left, register int right, register OBJECT *test, register OBJECT *(*key_function)(OBJECT*))* [*Function*]

OBJECT *sort_o *(OBJECT *list, OBJECT *in_test, OBJECT *in_key_function)* [*Function*]

OBJECT *stable_sort_o *(OBJECT *list, OBJECT *in_test, OBJECT *in_key_function)* [*Function*]

---

char *string_alloc *(int size)* [*Function*]

---

char *string_capitalize *(char *in_string, OBJECT *start, OBJECT *end)* [

Function]

OBJECT *string_capitalize_o *(OBJECT *object, OBJECT *start, OBJECT *end)*
[*Function*]

OBJECT *string_comp_equal_o *(register OBJECT *object1, register OBJECT *object2)* [*Function*]

---

char *string_downcase *(char *in_string)* [*Function*]

OBJECT *string_downcase_o *(OBJECT *object)* [*Function*]

---

char *string_downcase_s *(OBJECT *object)* [*Function*]

---

int string_equal *(register char *string1, register char *string2)* [*Function*]

OBJECT *string_equal11_o *(register OBJECT *object1, register OBJECT *object2)* [*Function*]

OBJECT *string_equal2_o *(OBJECT *string1, OBJECT *string2, OBJECT *start1, OBJECT *start2, OBJECT *end1, OBJECT *end2)* [*Function*]

OBJECT *string_equal_o *(OBJECT *string1, OBJECT *string2)* [*Function*]

---

int string_greater *(char *string1, char *string2)* [*Function*]

OBJECT *string_greater11_o *(OBJECT *object1, OBJECT *object2)* [*Function*]

OBJECT *string_greater2_o *(OBJECT *string1, OBJECT *string2, OBJECT *start1, OBJECT *start2, OBJECT *end1, OBJECT *end2)* [*Function*]

---

int string_less *(char *string1, char *string2)* [*Function*]

OBJECT *string_less_o *(OBJECT *object1, OBJECT *object2)* [*Function*]

OBJECT *string_o *(OBJECT *object)* [*Function*]

---

char *string_s *(OBJECT *atom)* [*Function*]

OBJECT *string_stream_p *(OBJECT *stream)* [*Function*]

OBJECT *string_to_char_list_o *(OBJECT *string)* [*Function*]

OBJECT *string_trim_o *(OBJECT *char_bag, OBJECT *string)* [*Function*]

char *string_upcase *(char *in_string)* [*Function*]

OBJECT *string_upcase_o *(OBJECT *object)* [*Function*]

char *string_upcase_s *(OBJECT *object)* [*Function*]

OBJECT *stringp_o *(OBJECT *object)* [*Function*]

OBJECT *sublis_o *(OBJECT *alist, OBJECT *tree, OBJECT *in_test, OBJECT *in_key_function)* [*Function*]

OBJECT *subseq_array_i *(OBJECT *array, int start_point, int end_point)* [*Function*]

OBJECT *subseq_i *(OBJECT *object, int start_point, int end_point)* [*Function*]

OBJECT *subseq_list_i *(register OBJECT *list, register int start_point, register int end_point)* [*Function*]

OBJECT *subseq_o *(OBJECT *object, OBJECT *start_point, OBJECT *end_point)* [*Function*]

OBJECT *subseq_string_i *(OBJECT *in_string_obj, int start_point, int end_point)* [*Function*]

OBJECT *subst_o *(OBJECT *new_obj, register OBJECT *obj, OBJECT *tree, OBJECT *in_test, OBJECT *in_key_function)* [*Function*]

OBJECT *substitute_o *(OBJECT *new_obj, OBJECT *obj,* [*Function*]

OBJECT *subtypep_o *(OBJECT *x, OBJECT *y)* [*Function*]

---

void swap2_object *(OBJECT **list_array, int i, int j)* [*Function*]

OBJECT *symbol_name_o *(OBJECT *object)* [*Function*]

OBJECT *symbol_p_list_o *(OBJECT *symbol)* [*Function*]

OBJECT *symbol_package_o *(OBJECT *object)* [*Function*]

OBJECT *symbol_value_o *(OBJECT *object)* [*Function*]

OBJECT *symbolp_o *(OBJECT *object)* [*Function*]

OBJECT *tenth_o *(OBJECT *list)* [*Function*]

OBJECT *terpri_o *(OBJECT *stream)* [*Function*]

---

void test_lisp_like_function *(void)* [*Function*]

OBJECT *third_o *(OBJECT *list)* [*Function*]

OBJECT *times2_o *(OBJECT *number1, OBJECT *number2)* [*Function*]

OBJECT *times3_o *(OBJECT *number1, OBJECT *number2, OBJECT *number3)* [*Function*]

OBJECT *times4_o *(OBJECT *number1, OBJECT *number2, OBJECT *number3, OBJECT *number4)* [*Function*]

OBJECT *times_o *(OBJECT *first, ...)* [*Function*]

OBJECT *truncate_o *(OBJECT *number1, OBJECT *number2)* [*Function*]

OBJECT *type_of_o *(OBJECT *object)* [*Function*]

OBJECT *typep_o *(OBJECT *object, OBJECT *type)* [*Function*]

OBJECT *union_o *(OBJECT *list_1, OBJECT *list_2, register OBJECT *test, register OBJECT *key)* [*Function*]

OBJECT *values0_o *(void)* [*Function*]

OBJECT *values1_o *(OBJECT *object1)* [*Function*]

OBJECT *values2_o *(OBJECT *object1, OBJECT *object2)* [*Function*]

OBJECT *values3_o *(OBJECT *object1, OBJECT *object2, OBJECT *object3)* [*Function*]

OBJECT *values4_o *(OBJECT *object1, OBJECT *object2, OBJECT *object3, OBJECT *object4)* [*Function*]

OBJECT *values5_o *(OBJECT *object1, OBJECT *object2, OBJECT *object3, OBJECT *object4, OBJECT *object5)* [*Function*]

OBJECT *values6_o *(OBJECT *object1, OBJECT *object2, OBJECT *object3, OBJECT *object4, OBJECT *object5, OBJECT *object6)* [*Function*]

OBJECT *values_o *(OBJECT *object1, OBJECT *object2)* [*Function*]

OBJECT *write_char_o *(OBJECT *character, OBJECT *stream)* [*Function*]

OBJECT *write_to_string_o *(OBJECT *object)* [*Function*]

OBJECT *zerop_o *(OBJECT *number)* [*Function*]

第 8 章

おわりに

著者は、本ライブラリを利用して、Common Lisp で書かれた約５０００行のプログラムをＣに変換し、ほとんどが単純作業でできることを確かめた。

更に、筆者は本ライブラリを基にして、「Lisp からＣへの自動変換ツール（L2C）」[4] を作成した。

最後に、開発に当たり貴重な意見を下さった ATR 音声翻訳通信研究所第三研究室の諸氏に感謝致します。

付録 A

サンプルプログラム

A.1 サンプルプログラムのソースリスト

```
#define DEVELOP

#include "lisp_like_function.h"

OBJECT *length_list_1 (OBJECT *object)
{
     /*** リカーシブを用いる方法 ***/
     trace_o(length_list_1);
     trace_print_o(object);

     if (atom_o(object)){
  return_o(read_from_fix_value(0));
     }
     else
     {
  return_o(read_from_fix_value
   (1 + get_fix_value(length_list_1(cdr_o(object)))));
     }
}


OBJECT *length_list_2 (OBJECT *object)
{
     /*** ループを用いる方法 ***/
     int i=0;
     dolist_o(obj, object, {
  i++;
          debug_print_o(read_from_fix_value(i));
     });
     return(read_from_fix_value(i));
}

void main (int argc, char **argv)
{
     OBJECT *length;
     OBJECT *input;
     OBJECT *memory_area;
```

```
    init_lisp_like_function();

    memory_area = make_memory_area(100);

    with_set_memory_area(memory_area, {
  input = quote_o((1 2 3 a b 3.5));

  printf("\ninput=");
  princ_o(input, T);

  length = length_list_1(input);
  printf("\nlength_1 =");
  princ_o(length, T);
  clear_memory_area();
  length = length_list_2(quote_o((1 2 3 a b 3.5)));
  printf("\nlength_2 =");
  princ_o(length, T);
    });
}
```

A.2 サンプルプログラムの Makefile

```
##
## Makefile
##

CC              = gcc
LinC_DIR = /home/as32/nisimura/System/export-L2C/LinC

CFLAGS          = -g
SYSLIBS         = -lm -L/usr/5lib

PROG = sample_program

OBJS =  $(LinC_DIR)/y.tab.o $(LinC_DIR)/lex.yy.o $(LinC_DIR)/lisp_like_function.o \
$(LinC_DIR)/init_lisp_symbol_s.o $(LinC_DIR)/lisp_number.o $(LinC_DIR)/machines.o

$(PROG):$
$(CC) $(CFLAGS) -I$(LinC_DIR) -o $@ $(PROG).c -L$(LinC_DIR) $(OBJS) -lm $(LinC_DIR)/../gc/gc.a

clean:
rm -f $(PROG)
```

A.3 サンプルプログラムの実行結果

```
input=(1 2 3 A B 3.5000000000)
  ||(1)TRACE||-----s------length_list_1
       ||TRACE||object=(1 2 3 A B 3.5000000000)
    ||(2)TRACE||-----s------length_list_1
         ||TRACE||object=(2 3 A B 3.5000000000)
      ||(3)TRACE||-----s------length_list_1
           ||TRACE||object=(3 A B 3.5000000000)
        ||(4)TRACE||-----s------length_list_1
             ||TRACE||object=(A B 3.5000000000)
          ||(5)TRACE||-----s------length_list_1
               ||TRACE||object=(B 3.5000000000)
            ||(6)TRACE||-----s------length_list_1
                 ||TRACE||object=(3.5000000000)
              ||(7)TRACE||-----s------length_list_1
                   ||TRACE||object=NIL
              ||(7)TRACE||<---read_from_fix_value(0) = 0
            ||(6)TRACE||<---read_from_fix_value (1 + get_fix_value(length_list_1(cdr_o(object)))) = 1
          ||(5)TRACE||<---read_from_fix_value (1 + get_fix_value(length_list_1(cdr_o(object)))) = 2
        ||(4)TRACE||<---read_from_fix_value (1 + get_fix_value(length_list_1(cdr_o(object)))) = 3
      ||(3)TRACE||<---read_from_fix_value (1 + get_fix_value(length_list_1(cdr_o(object)))) = 4
    ||(2)TRACE||<---read_from_fix_value (1 + get_fix_value(length_list_1(cdr_o(object)))) = 5
  ||(1)TRACE||<---read_from_fix_value (1 + get_fix_value(length_list_1(cdr_o(object)))) = 6
length_1 =6
read_from_fix_value(i)=1
read_from_fix_value(i)=2
read_from_fix_value(i)=3
read_from_fix_value(i)=4
read_from_fix_value(i)=5
read_from_fix_value(i)=6
length_2 =6
```

参考文献

[1] 小西弘一, 清水剛：“C プログラムブック III”,(株) アスキー,(1986-9)

[2] 後藤英一, 井田昌之：“COMMON LISP 言語使用書”, 共立出版 (株),(1986-3)

[3] B.W. カーニハン /D.M. リッチー 著、石田晴久 訳：“プログラミング言語 C 第 2 版 (ANSI 規格準拠)”, 共立出版 (株),(1989)

[4] 西村仁志、隅田：“TR-IT-0292　リスプから C への自動変換ツール (L2C)”, (株) ATR 音声翻訳通信研究所,(1999-2)