

```

/*
  LISP-HEADER.h was originally written by Daniel N. Ozick, 8-91 and appeared in
  Dr. Dobb's Journal. Later modified by Douglas Chubb 1991-92 to handle LISP
  atomic
  property lists, lists with NULL and most COMMON LISP predicates. C-LISP
  garbage
  collector has been completely reworked and improved.
*/

```

```

/* Lisp-Style Library for C (Main Header File */

```

```

/* Constants */

```

```

/* Array Sizes */

```

```

#define MAXSTRING 128          /* size of standard character array      */
#define MAXLINE 256          /* size of text line character array     */
#define HASH_TABLE_SZ 211    /* size of HASH_TABLE -- should be prime */
    */

```

```

/* Characters */

```

```

#define EOS '\0'

```

```

#define TAB '\t'

```

```

#define NEWLINE '\n'

```

```

#define DOS_EOF 26

```

```

#define SPACE 32

```

```

#define FORMFEED '\f'

```

```

#define BELL 7

```

```

#define BACKSPACE 8

```

```

#define RETURN 13

```

```

#define LINEFEED 10

```

```

#define ESCAPE 27

```

```

#define DOT '.'

```

```

#define PERIOD '.'

```

```

#define BACKSLASH '\\'

```

```

#define SINGLE_QUOTE '\"'

```

```

#define DOUBLE_QUOTE '\"'

```

```

#define LEFT_PAREN '('

```

```

#define RIGHT_PAREN ')'

```

```

#define LINE_SPLICE (-2)

```

```

#define CHAR_BYTE_SIZE 0 /* See usage description below. */

```

/* CHAR_BYTE_SIZE = n-1 bytes of computer word, but depends upon how memory is addressed, e.g., for Mac IIfx, n=1; for Sun SPARCII, n=4.

Note: Object_Type may be coded to use only one byte for 'char' expression,

such as the 68000-based computers. Other computers may assign full 4-byte words (32 bits) for 'char' CHAR_BYTE_SIZE is used in "SOT" function. */

/** Types **/

/* Object_Type -- values for first component of 'Object' (self-id tag) */

typedef char Object_Type;

/* General Types */

#define UNDEFINED 0

#define SYMBOL 1

#define STRING 2

#define INTEGER 3

#define FUNCTION 4

#define PAIR 5

/* Char Object_Type bits will be assigned as follows: nP, Cnn, TTT
and decoded as follows:

P = garbage-collector 'protection' bit

C = garbage-collector 'change' bit

n = not currently used

T = Object_Type designator (0-7) */

/* Built-in C Structures */

/* Pointer -- 'Generic *' : what's pointed to is unknown at compile time */

typedef void *Pointer;

/* Object -- pointer to self-identified object (starts with Object_Type) */

typedef Object_Type *Object;

/* Function -- pointer to function of ? arguments returning Object */

typedef Object (*Function)(Object, ...);

/* Function_0 -- pointer to function of 0 arguments return Object */

typedef Object (*Function_0)(void);

/* Function_1 -- pointer to function of 1 Object return Object */

typedef Object (*Function_1)(Object);

/* Function_1v -- pointer to function of 1 Object, return void */

typedef void (*Function_1v)(Object);

```

/* Symbol_Entry -- the attributes of a symbol (entered into Symbol_Table) */
typedef struct
{
    char *print_name;      /* printed representation and lookup key */
    Object value;         /* value of global variable named by symbol */
    Object plist;        /* symbol indicator-property list */
} Symbol_Entry;

/* Pair -- a Lisp 'cons' cell for creating linked lists */
typedef struct
{
    Object car;           /* any Object */
    Object cdr;          /* PAIR Object or NULL (usually) */
} Pair;

/* Hash_Table -- an array of hash-bucket lists used for symbol tables */
typedef Object Hash_Table [HASH_TABLE_SZ];

/** Object Components */
/* SOT -- size of 'Object Type' (bytes used by type tag) */
#define SOT sizeof (Object_Type) + CHAR_BYTE_SIZE

/* type -- return the object's self-identification (Object_Type) */
#define type(object)      *((Object_Type *) object)

#define ntype(object)     ('\007' & (type(object)))

/* symbol -- return the address of symbol's name and value (Symbol_Entry) */
#define symbol(object)    ((Symbol_Entry *) (object + SOT))
/* symbol_value -- return the value assigned to a symbol */
#define symbol_value(object) (symbol (object)->value)
/* string -- return the address of (the first char of) standard C string */
#define string(object)   ((char *) (object + SOT))
/* integer -- return an 'int' */
#define integer(object)  *((int *) (object + SOT))
/* function -- return the address of a function that returns Object */
#define function(object) *((Function *) (object + SOT))
/* pair -- return the address of a Lisp-style CONS cell */
#define pair(object)     ((Pair *) (object + SOT))
/* first -- return first element of a list (LISP CAR) */
#define first(object)    (pair(object)->car)
/* but_first -- return list less its first element (LISP CDR) */
#define but_first(object) (pair(object)->cdr)
/* second -- C Macro for LISP CADR */
#define second(object) first(but_first(object))

```

```

/* third -- return third element of list object */
#define third(object) first(but_first(but_first(object)))

/** Type Predicates: */
#define is_null(object)      (object == NULL)
#define is_symbol(object)   (ntype(object) == SYMBOL)
#define is_pair(object)     (ntype(object) == PAIR)
#define is_atom(object)     (is_null(object) || (ntype(object) != PAIR))
#define is_list(object)    (is_null(object) || is_pair(object))
#define is_string(object)   (ntype(object) == STRING)
#define is_integer(object)  (ntype(object) == INTEGER)
#define is_function(object) (ntype(object) == FUNCTION)

/* declare_symbol -- declare extern var with same name as interned symbol */
#define declare_symbol(name, type)      extern Object name;
/* symbol_plist -- returns property list of object */
#define symbol_plist(object) (symbol(object) -> plist)

/* List-Based Stacks */
/* push -- push an object onto a (list-based) stack */
#define push(location, object) location = first_put (object, location)
/* pop -- pop an object off of a (list-based) stack, NULL if stack empty */
#define pop(location) \
( (location != NULL) ? pop_f (&location) : NULL)

/* LISP Function Prototypes */
void error (char *fstr, ...);
Object first_put (Object item, Object list);
Object last_put (Object item, Object list);
Object last (Object lst);
Object list (Object item, ...);
Object append (Object list1, Object list2);
Object nconc (Object list_1, Object list_2);
Object lisp_union (Object list1, Object list2);
Object get_prop (Object sym, char *str);
void put_prop (Object sym, char *str, Object property);
void free_structure (Object obj);
void remprop (Object sym, char *str);
Object gensym (char *ppp);
Object make_indic (char *str);
Object make_prop (char *str);

Object reverse (Object list);
void mapc (Function_1 f, Object list);
Object mapcar (Function_1 f, Object list);

```

```
void mapl (Function_1 f, Object arg_list);
Object map_no_nils (Function_1 f, Object list);
Object nth (Object list, int n);
Object assoc (Object key, Object a_list);
Object pop_f (Object *location);
int length (Object list);
Object is_member (Object obj, Object list);
Object member (Object obj, Object list);
Object lisp_equal (Object obj1, Object obj2);
int index (Object element, Object list);
Object set_difference (Object list1, Object list2);
Object intersection (Object lst1, Object lst2);
Object remove_item (Object item, Object sequence);
Object remove_duplicates (Object obj);
char *make_c_string (char *str);
Object make_symbol (char *name);
Object make_string (char *s);
Object make_integer (int n);
Object make_function (Function f);
```

```
/* Input-Output */
```

```
void write_object (Object obj);
Object read_object (void);
Object lookup (char *str);
Object intern (char *str);
Object install_with_value (char *str, Object val);
Object set_symbol_value (Object sym, Object val);
void install_internal_symbols (void);
```

```
/* Garbage Collection functions */
```

```
void initialize_garbage_collector (void);
void push_memory_pointer (Pointer p);
Pointer pop_memory_pointer (void);
Pointer safe_malloc (size_t size);
void safe_free (void *p);
void push_temp_pointer (Pointer p);
Pointer pop_temp_pointer (void);
void collect_garbage (void);
void mark_object (Object obj);
void mark2_object (Object obj);
void unmark_object (Object obj);
```

```
void init_internal_read_table (void);
void set_internal_reader (void);
```