```
// $Id: oclib.oh,v 1.24 2011-11-18 17:34:29-08 - - $

#ifndef __OCLIB_OH__
#define __OCLIB_OH__

#ifdef __OCLIB_C__
#   define __(ID)          __##ID
#   define NONE__          void
#   define VOID__(ID)      void __##ID
#   define BOOL__(ID)      ubyte __##ID
#   define CHAR__(ID)      ubyte __##ID
#   define INT__(ID)       int __##ID
#   define STRING__(ID)    ubyte *__##ID
#   define STRINGS__(ID)   ubyte **__##ID
#   define null            0
#   define false           0
#   define true            1
typedef unsigned char ubyte;
void *xcalloc (int nelem, int size);
#else
#   define EOF             (-1)
#   define __(ID)          ID
#   define NONE__
#   define VOID__(ID)      void ID
#   define BOOL__(ID)      bool ID
#   define CHAR__(ID)      char ID
#   define INT__(ID)       int ID
#   define STRING__(ID)    string ID
#   define STRINGS__(ID)   string[] ID
VOID__(__assert_fail) (STRING__(expr), STRING__(file), INT__(line));
#endif

VOID__(putb) (BOOL__(b));
VOID__(putc) (CHAR__(c));
VOID__(puti) (INT__(i));
VOID__(puts) (STRING__(s));
VOID__(endl) (NONE__);
INT__(getc) (NONE__);
STRING__(getw) (NONE__);
STRING__(getln) (NONE__);
STRINGS__ (getargv) (NONE__);
VOID__(exit) (int status);
#define assert(expr) \
        {if (! (expr)) __(__assert_fail) (#expr, __FILE__, __LINE__);}

#endif

```

```
// $Id: oclib.c,v 1.45 2012-11-16 21:10:41-08 - - $

#include <ctype.h>
#include <libgen.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define __OCLIB_C__
#include "oclib.oh"

ubyte **oc_argv;

void ____assert_fail (char *expr, char *file, int line) {
   fflush (NULL);
   fprintf (stderr, "%s: %s:%d: assert (%s) failed.\n",
            basename ((char *) oc_argv[0]), file, line, expr);
   fflush (NULL);
   abort();
}

void *xcalloc (int nelem, int size) {
   void *result = calloc (nelem, size);
   assert (result != NULL);
   return result;
}

void __ocmain (void);
int main (int argc, char **argv) {
   argc = argc; // warning: unused parameter 'argc'
   oc_argv = (ubyte **) argv;
   __ocmain();
   return EXIT_SUCCESS;
}

```

```

ubyte *scan (int (*skipover) (int), int (*stopat) (int)) {
   int byte;
   do {
      byte = getchar();
      if (byte == EOF) return NULL;
   } while (skipover (byte));
   ubyte buffer[0x1000];
   ubyte *end = buffer;
   do {
      *end++ = byte;
      assert (end < buffer + sizeof buffer);
      *end = '\0';
      byte = getchar();
   }while (byte != EOF && ! stopat (byte));
   ubyte *result = (ubyte *) strdup ((char *) buffer);
   assert (result != NULL);
   return result;
}

int isfalse (int byte)   { return 0 & byte; }
int isnl (int byte)      { return byte == '\n'; }
void __putb (ubyte byte) { printf ("%s", byte ? "true" : "false"); }
void __putc (ubyte byte) { printf ("%c", byte); }
void __puti (int val)    { printf ("%d", val); }
void __puts (ubyte *str) { printf ("%s", str); }
void __endl (void)       { printf ("%c", '\n'); fflush (NULL); }
int __getc (void)        { return getchar(); }
ubyte *__getw (void)     { return scan (isspace, isspace); }
ubyte *__getln (void)    { return scan (isfalse, isnl); }
ubyte **__getargv (void) { return oc_argv; }
void __exit (int status) { exit (status); }

```

```
// $Id: 00-trivial.oc,v 1.1 2011-09-15 18:50:16-07 - - $
//
// This program does nothing then returns an exit status of 0.
//
```

```
// $Id: 01-hello.oc,v 1.1 2011-09-15 18:50:16-07 - - $
// Simple hello world program.

#include "oclib.oh"

puts ("Hello, world!\n");
```

```
// $Id: 03-test3.oc,v 1.2 2012-12-03 13:17:30-08 - - $

#include "oclib.oh"

int a = 3;
int b = 8;
int c = a + b;
a = b + c;
puti (a);
putc ('\n');

```

```
// $Id: 04-test4.oc,v 1.1 2011-09-15 18:50:16-07 - - $

#include "oclib.oh"

struct foo {
   int a;
}

int a = 6;
foo b = new foo ();
b.a = 8;
a = a * b.a + 6;;

puti (a);
putc (' ');
puti (b.a);
endl ();

```

```
// $Id: 06-test6.oc,v 1.3 2012-12-03 13:19:06-08 - - $

#include "oclib.oh"

struct foo {}
struct bar {}

int f0 ();
int f1 (int a);
int f2 (int a, int b);
int f3 (string a, string b, string c);
int f4 (foo a, bar b);
string s = "a";
string[] sa = new string[10];

```

```
// $Id: 07-assert.oc,v 1.2 2012-11-27 13:38:26-08 - - $

#include "oclib.oh"
assert ("null" == null);
```

```
// $Id: 10-hundred.oc,v 1.1 2011-09-15 18:50:16-07 - - $

#include "oclib.oh"

int count = 0;
while (count <= 100) {
   count = count + 1;
   puti (count);
   endl ();
}
```

```
// $Id: 11-numbers.oc,v 1.1 2011-09-15 18:50:16-07 - - $

#include "oclib.oh"

int number = 1;
while (number > 0) {
   puti (number);
   putc ('\n');
   number = number + number;
}
puti (number);
putc ('\n');

```

```
// $Id: 12-elseif.oc,v 1.2 2012-12-03 13:21:16-08 - - $

#include "oclib.oh"

int a = 3;

if (a == 1) puts ("one");
else if (a == 2) puts ("two");
else if (a == 3) puts ("three");
else puts ("many");
endl ();
```

```
// $Id: 13-assertfail.oc,v 1.3 2011-11-08 14:53:37-08 - - $

#undef __OCLIB_OH__
#include "oclib.oh"

puts (getargv()[0]);
puts (" was compiled ");
puts (__DATE__);
puts (" @ ");
puts (__TIME__);
endl ();
assert ("assert" == "fail");

```

```
// $Id: 14-ocecho.oc,v 1.2 2011-11-16 23:08:30-08 - - $

#include "oclib.oh"

string[] argv = getargv ();
int argi = 1;
while (argv[argi] != null) {
   if (argi > 1) putc (' ');
   puts (argv[argi]);
   argi = argi + 1;
}
endl ();

```

```
// $Id: 20-fib-array.oc,v 1.3 2012-12-03 13:22:58-08 - - $
//
// Put Fibonacci numbers in an array, then print them.
//

#include "oclib.oh"

#define FIB_SIZE 30
int[] fibonacci = new int[FIB_SIZE];

fibonacci[0] = 0;
fibonacci[1] = 1;

int index = 2;
while (index < FIB_SIZE) {
   fibonacci[index] = fibonacci[index - 1] + fibonacci[index - 2];
   index = index + 1;
}

index = 0;
puts ("Numeri di figlio Bonacci\n");
while (index < FIB_SIZE) {
   puts ("fibonacci[");
   puti (index);
   puts (" = ");
   puti (fibonacci[index]);
   endl ();
   index = index + 1;
}
```

```
// $Id: 21-eratosthenes.oc,v 1.2 2011-09-19 14:25:40-07 - - $

#include "oclib.oh"
#define SIZE 100
#define LOWPRIME 2

bool[] sieve = new bool[SIZE];
int index = LOWPRIME;

while (index < SIZE) {
   sieve[index] = true;
   index = index + 1;
}

int prime = LOWPRIME;
while (prime < SIZE) {
   if (sieve[prime]) {
      index = prime * 2;
      while (index < SIZE) {
         sieve[index] = false;
         index = index + prime;
      }
   }
   prime = prime + 1;
}

index = LOWPRIME;
while (index < SIZE) {
   if (sieve[index]) {
      puti (index);
      endl ();
   }
}

```

```
// $Id: 23-atoi.oc,v 1.7 2012-12-03 13:21:36-08 - - $

#include "oclib.oh"

int atoi (string str) {
   assert (str != null);
   bool neg = false;
   int num = 0;
   int digit = 0;
   if (str[0] != '\0') {
      if (str[0] == '-') {
         digit = digit + 1;
         neg = true;
      }
      bool contin = true;
      while (contin) {
         if (str[digit] == '\0') {
            contin = false;
         }else {
            char c = str[digit];
            digit = digit + 1;
            if (c < '0') contin = false;
            else if (c > '9') contin = false;
            else num = num * 10 + ord c - ord '0';
         }
      }
      if (neg) num = - num;
   }
   return num;
}

int argi = 0;
string[] argv = getargv ();
while (argv[argi] != null) {
   string arg = argv[argi];
   puts (arg);
   puts (" = ");
   puti (atoi (arg));
   endl ();
}

```

```
// $Id: 30-fac-fnloop.oc,v 1.4 2011-11-16 20:18:50-08 - - $
//
// Function uses a loop to compute factorial.
//

#include "oclib.oh"

int fac (int n) {
   int f = 1;
   while (n > 1) {
      f = f * n;
      n = n - 1;
   }
   return f;
}

int n = 1;
while (n <= 5) {
   puti (fac (n));
   endl ();
   n = n + 1;
}

```

```
// $Id: 31-fib-2supn.oc,v 1.1 2011-10-20 21:31:44-07 - - $
//
// Very slow program, computes Fibonacci numbers with O(2^n) speed.
//

#include "oclib.oh"

int fibonacci (int n) {
   if (n < 2) return n;
   return fibonacci (n - 1) + fibonacci (n - 2);
}

// Main program.

int n = 0;
while (n < 10) {
   puts ("fibonacci(");
   puti (n);
   puts (" = ");
   puti (fibonacci (n));
   endl ();
}

```

```
// $Id: 40-arraystack.oc,v 1.6 2012-12-03 13:23:28-08 - - $

#include "oclib.oh"

#define EMPTY (-1)

struct stack {
   string[] data;
   int size;
   int top;
}

stack new_stack (int size) {
   stack stack = new stack (); // Zeros out both fields.
   stack.data = new string[size]; // Array of null pointers.
   stack.size = size;
   stack.top = EMPTY;
   return stack;
}

void push (stack stack, string str) {
   assert (stack.top < stack.size - 1);
   stack.top = stack.top + 1;
   stack.data[stack.top] = str;
}

string pop (stack stack) {
   assert (stack.top != EMPTY);
   string tmp = stack.data[stack.top];
   stack.top = stack.top - 1;
   return tmp;
}

bool empty (stack stack) {
   return stack.top == EMPTY;
}

// Main program.
string[] argv = getargv ();
stack stack = new_stack (100);

int argi = 0;
while (argv[argi] != null) {
   push (stack, argv[argi]);
   argi = argi + 1;
}

while (! empty (stack)) {
   puts (pop (stack));
   endl ();
}

```

```
// $Id: 41-linkedstack.oc,v 1.7 2012-12-03 13:24:26-08 - - $

#include "oclib.oh"

struct node {
   string data;
   node link;
}

struct stack {
   node top;
}

bool empty (stack stack) {
   assert (stack != null);
   return stack.top == null;
}

stack new_stack () {
   stack stack = new stack ();
   stack.top = null;
   return stack;
}

void push (stack stack, string str) {
   assert (stack != null);
   node tmp = new node ();
   tmp.data = str;
   tmp.link = stack.top;
   stack.top = tmp;
}

string pop (stack stack) {
   assert (stack != null);
   assert (! empty (stack));
   string tmp = stack.top.data;
   stack.top = stack.top.link;
   return tmp;
}

// Main program.

string[] argv = getargv ();
stack stack = new_stack ();
int argi = 0;

while (argv[argi] != null) {
   push (stack, argv[argi]);
   argi = argi + 1;
}

while (! empty (stack)) {
   puts (pop (stack));
   endl ();
}

```

```
// $Id: 42-viiiqueens.oc,v 1.4 2012-12-03 13:24:51-08 - - $

#include "oclib.oh"

#define BOARD_SIZE 8
int[] board = new int[BOARD_SIZE];

bool is_safe (int newcol) {
   int col = 0;
   while (col < newcol) {
      if (board[col] == board[newcol]) return false;
      int diagonal = board[col] - board[newcol];
      if (diagonal == col - newcol) return false;
      if (diagonal == newcol - col) return false;
      col = col + 1;
   }
   return true;
}

void printqueens () {
   int col = 0;
   while (col < BOARD_SIZE) {
      putc (chr (board[col] + ord '1'));
      col = col + 1;
   }
   putc ('\n');
}

void queens (int newcol) {
   if (newcol == BOARD_SIZE) printqueens ();
   else {
      int row = 0;
      while (row < BOARD_SIZE) {
         board[newcol] = row;
         if (is_safe (newcol)) queens (newcol + 1);
         row = row + 1;
      }
   }
}

// Main program.
queens (0);

```

```
// $Id: 44-dot-product.oc,v 1.3 2012-12-03 13:25:15-08 - - $

#include "oclib.oh"

int dot_product (int size, int[] vec1, int[] vec2) {
   int index = 0;
   int dot = 0;
   while (index < size) {
      dot = dot + vec1[index] * vec2[index];
      index = index + 1;
   }
   return dot;
}

#define SIZE 10
int[] vec1 = new int[SIZE];
int[] vec2 = new int[SIZE];
int i = 0;
while (i < SIZE) {
   vec1[i] = i + 10;
   vec2[i] = i * 10;
}
puti (dot_product (SIZE, vec1, vec2));
endl ();
```

```
// $Id: 45-towers-of-hanoi.oc,v 1.3 2012-12-03 13:26:14-08 - - $

#include "oclib.oh"

void move (string src, string dst) {
   puts ("Move a disk from ");
   puts (src);
   puts (" to ");
   puts (dst);
   puts (".\n");
}

void towers (int ndisks, string src, string tmp, string dst) {
   if (ndisks < 1) return;
   towers (ndisks - 1, src, dst, tmp);
   move (src, dst);
   towers (ndisks - 1, tmp, src, dst);
}

towers (4, "Source", "Temporary", "Destination");

```

```
// $Id: 53-insertionsort.oc,v 1.4 2012-12-03 13:26:42-08 - - $
//
// Use insertion sort to print argv in osrted order.
//

#include "oclib.oh"

int strcmp (string s1, string s2) {
   int index = 0;
   bool contin = true;
   while (contin) {
      char s1c = s1[index];
      char s2c = s2[index];
      int cmp = ord s1c - ord s2c;
      if (cmp != 0) return cmp;
      if (s1c == '\0') contin = false;
      index = index + 1;
   }
   return 0;
}

void insertion_sort (int size, string[] array) {
   int sorted = 1;
   while (sorted < size) {
      int slot = sorted;
      string element = array[slot];
      bool contin = true;
      while (contin) {
         if (slot == 0) {
            contin = false;
         }else if (strcmp (array[slot - 1], element) <= 0) {
            contin = false;
         }else {
            array[slot] = array[slot - 1];
            slot = slot - 1;
         }
      }
      array[slot] = element;
      sorted = sorted + 1;
   }
}

void print_array (string label, int size, string[] array) {
   endl ();
   puts (label);
   puts (":\n");
   int index = 0;
   while (index < size) {
      puts (array[index]);
      endl ();
      index = index + 1;
   }
}

string[] argv = getargv ();
int argc = 0;
while (argv[argc] != null) argc = argc + 1;
print_array ("unsorted", argc, argv);
```

```
insertion_sort (argc, argv);
print_array ("sorted", argc, argv);

```

```
char O[9];Q(l,b,d){int o=8,p=1,q=1<<
l|1<<22-l;for(;l>7?!write(1,O,9):o--
;)O[l]=56-o,b&p|d&q||Q(l+1,b|p,d|q),
p*=2,q*=2;}main(){O[8]=10;Q(0,0,0);}
```

```
// $Id: 91-typecheck.oc,v 1.1 2011-11-07 12:09:34-08 - - $
//
// This file should scan and parse correctly,
// but fail to type check.
//

int[] a = null;
reference[] a = new string[10];
void foo ();
void foo (int a);
void foo (int[] a, int[] b) {int x = a + b;}
struct foo { int a; int b; }

a + b;
f ();
f (x, y+3, z);
foo + bar;
a = b = c = d;
test = abc + def + ghi;
this + 23 * a + "hello";
while (a < b) f = f + 1;
return 3 + 4;
a[i] = b[j];
return;
while (true) {a = 3; b = 4; }
if (a == b) f (x);
if (a != b) y = 3; else f (y, z);

```

```
/*
This is an unterminated comment.
It would cause cpp to error out.
When cpp returns a non-zero exit code,
so should your compiler.
$Id: 92-uncomment.oc,v 1.1 2011-09-15 18:50:16-07 - - $

int main (int argc, char **argv) {
```

```
// $Id: 93-semantics.oc,v 1.1 2011-11-01 22:02:19-07 - - $
// This code should scan and parse correctly,
// but fail to type check.
int[] a = null;
int[] b = null;
int c = a + b; // can’t add arrays
void[] f() {}; // can’t hae void[]
void n = null; // can’t have void vars
bool x = a < b; // can’t compare pointers <
bool y = a==b; // this is ok
```

```
// $Id: 94-syntax.oc,v 1.1 2011-11-01 22:02:19-07 - - $
k
int f() {
int a = ;
return foo;
public static void main (String[] args) {
   System.exit (255);
}

```

```
// $Id: 95-cobol.oc,v 1.1 2011-11-01 22:02:19-07 - - $

000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID.     HELLOWORLD.
000300
000400*
000500 ENVIRONMENT DIVISION.
000600 CONFIGURATION SECTION.
000700 SOURCE-COMPUTER. RM-COBOL.
000800 OBJECT-COMPUTER. RM-COBOL.
000900
001000 DATA DIVISION.
001100 FILE SECTION.
001200
100000 PROCEDURE DIVISION.
100100
100200 MAIN-LOGIC SECTION.
100300 BEGIN.
100400     DISPLAY " " LINE 1 POSITION 1 ERASE EOS.
100500     DISPLAY "Hello world!" LINE 15 POSITION 10.
100600     STOP RUN.
100700 MAIN-LOGIC-EXIT.
100800     EXIT.
```

```
// Unterminated strings.
// $Id: 96-unterminated.oc,v 1.3 2012-12-03 13:27:56-08 - - $

string t = "\*/";
string s = "abc;
char c = 'a;
s = "abcd\";
s = "abc|\
;
int 23foobar;
```