```
 1: // $Id: oclib.h,v 1.6 2019-04-17 13:44:05-07 - - $
 2:
 3: // Bilingual file useable as a header file for both oc and c++.
 4:
 5: #ifndef __OCLIB_H__
 6: #define __OCLIB_H__
 7:
 8: #ifdef __cplusplus
 9:    using string = char*;
10:    extern "C" {
11:    #define END_EXTERN_C }
12: #else
13:    #define END_EXTERN_C
14: #endif
15:
16: #define SUCCESS 0
17: #define FAILURE 1
18: #define TRUE 1
19: #define FALSE 0
20: #define EOF (-1)
21:
22: #define assert(expr) if (not (expr)) fail (#expr, __FILE__, __LINE__);
23:
24: void fail (string expr, string file, int line);
25:
26: void putchr (int c);
27: void putint (int i);
28: void putstr (string s);
29:
30: int getchr();
31: string getstr();
32: string getln();
33:
34: END_EXTERN_C
35:
36: #endif
37:
```

```cpp
 1: // $Id: ocdefs.h,v 1.4 2019-04-17 18:51:14-07 - - $
 2:
 3: // C++ template file used to compile oc programs with c++ compiler.
 4:
 5: #ifndef __OCDEFS_H__
 6: #define __OCDEFS_H__
 7:
 8: #include <type_traits>
 9:
10: using string = char*;
11:
12: template <typename type>
13: using ptr = type*;
14:
15: template <typename type>
16: struct array {
17:    using array_value_type = type;
18:    type* data {};
19:    array() = default;
20:    array (type* that) { data = that; }
21:    array& operator= (type* that) { data = that; return *this; }
22:    type& operator[] (int i) { return data[i]; }
23: };
24:
25: template <typename type>
26: typename std::enable_if<std::is_class<type>::value,ptr<type>>::type
27: alloc() {
28:    return new type();
29: }
30:
31: template <typename type>
32: array<typename type::array_value_type>
33: alloc (int size) {
34:    auto result = new typename type::array_value_type [size] {};
35:    using result_t = array<typename type::array_value_type>*;
36:    return *reinterpret_cast<result_t> (&result);
37: }
38:
39: template <typename type>
40: typename std::enable_if<std::is_same<type,string>::value,string>::type
41: alloc (int size) {
42:    return new char[size] {};
43: }
44:
45: #endif
46:
```

```
 1: // $Id: oclib.c,v 1.3 2019-04-17 13:47:02-07 - - $
 2:
 3: #include <stdio.h>
 4: #include <stdlib.h>
 5: #include <string.h>
 6:
 7: #define not !
 8: #define nullptr 0
 9: #define string char*
10:
11: #include "oclib.h"
12:
13: void fail (string expr, string file, int line) {
14:     fprintf (stderr, "%s:%d: assert (%s) failed\n", file, line, expr);
15:     abort();
16: }
17:
18: void* xcalloc (int nelem, int size) {
19:     void* result = calloc (nelem, size);
20:     assert (result != nullptr);
21:     return result;
22: }
23:
24: void putchr (int chr) { printf ("%c", chr); }
25: void putint (int num) { printf ("%d", num); }
26: void putstr (string str) { printf ("%s", str); }
27:
28: int getchr() { return getchar(); }
29:
30: string getstr (void) {
31:     static char buffer[0x1000];
32:     static char format[16];
33:     sprintf (format, "%%%zds", sizeof buffer - 1);
34:     int count = scanf (format, buffer);
35:     return count != 1 ? nullptr : strdup (buffer);
36: }
37:
38: string getln (void) {
39:     static char buffer[0x1000];
40:     string result = fgets (buffer, sizeof buffer, stdin);
41:     return result == nullptr ? nullptr : strdup (result);
42: }
43:
```

```
1: // $Id: 00-trivial.oc,v 1.1 2019-04-16 12:14:45-07 - - $
2:
3: int main(){}
```

```
 1: // $Id: 01-hello.oc,v 1.1 2019-04-16 12:14:45-07 - - $
 2: // Simple hello world program.
 3:
 4: #include "oclib.h"
 5:
 6: int main() {
 7:    putstr ("Hello, world!\n");
 8:    return SUCCESS;
 9: }
10:
```

```
 1: // $Id: 03-test3.oc,v 1.1 2019-04-16 12:14:45-07 - - $
 2:
 3: #include "oclib.h"
 4:
 5: int main() {
 6:    int a = 3;
 7:    int b = 8;
 8:    int c = a + b;
 9:    a = b + c;
10:    putint (a);
11:    putchr ('\n');
12:    return SUCCESS;
13: }
14:
```

```
 1: // $Id: 04-test4.oc,v 1.1 2019-04-16 12:14:45-07 - - $
 2:
 3: #include "oclib.h"
 4:
 5: struct foo {
 6:     int a;
 7: };
 8:
 9: int main() {
10:     int a = 6;
11:     ptr<struct foo> b = alloc<struct foo>();
12:     b->a = 8;
13:     a = a * b->a + 6;;
14:     putint (a);
15:     putchr (' ');
16:     putint (b->a);
17:     putchr ('\n');
18:     return SUCCESS;
19: }
20:
```

```
 1: // $Id: 06-test6.oc,v 1.1 2019-04-16 12:14:45-07 - - $
 2:
 3: #include "oclib.h"
 4:
 5: struct foo {};
 6: struct bar {};
 7:
 8: int f0();
 9: int f1 (int a);
10: int f2 (int a, int b);
11: int f3 (string a, string b, string c);
12: int f4 (ptr<struct foo> a, ptr<struct bar> b);
13:
14: int main() {
15:    string s = "a";
16:    array<string> sa = alloc<array<string>>(10);
17:    return SUCCESS;
18: }
19:
```

```
1: // $Id: 07-assert.oc,v 1.2 2019-04-16 13:14:01-07 - - $
2:
3: #include "oclib.h"
4:
5: int main() {
6:    assert ("nullptr" == nullptr);
7:    return FAILURE;
8: }
9:
```

```
 1: // $Id: 10-hundred.oc,v 1.2 2019-04-17 13:23:14-07 - - $
 2:
 3: #include "oclib.h"
 4:
 5: int main() {
 6:    int count = 0;
 7:    while (count < 16) {
 8:       count = count + 1;
 9:       putint (count);
10:       putchr ('\n');
11:    }
12:    return SUCCESS;
13: }
14:
```

```
 1: // $Id: 11-numbers.oc,v 1.1 2019-04-16 12:14:45-07 - - $
 2:
 3: #include "oclib.h"
 4:
 5: int main() {
 6:     int number = 1;
 7:     int done = FALSE;
 8:     while (not done) {
 9:         putint (number);
10:         putchr ('\n');
11:         if (number <= 0) done = TRUE;
12:         number = number + number;
13:     }
14:     return SUCCESS;
15: }
16:
```

```
 1: // $Id: 12-elseif.oc,v 1.1 2019-04-16 12:14:45-07 - - $
 2:
 3: #include "oclib.h"
 4:
 5: int main (int argc) {
 6:    if (argc == 1) putstr ("one");
 7:    else if (argc == 2) putstr ("two");
 8:    else if (argc == 3) putstr ("three");
 9:    else putstr ("many");
10:    putchr ('\n');
11:    return SUCCESS;
12: }
13:
```

```
 1: // $Id: 13-macros.oc,v 1.2 2019-04-16 13:12:54-07 - - $
 2:
 3: #include "oclib.h"
 4:
 5: int main (int argc, array<string> argv) {
 6:     putstr (argv[0]);
 7:     putstr (" ");
 8:     putstr (__FILE__);
 9:     putchr ('[');
10:     putint (__LINE__);
11:     putstr ("] compiled ");
12:     putstr (__DATE__);
13:     putstr (" ");
14:     putstr (__TIME__);
15:     putstr ("\n");
16:     return SUCCESS;
17: }
18:
```

```
 1: // $Id: 14-ocecho.oc,v 1.1 2019-04-16 12:14:45-07 - - $
 2:
 3: #include "oclib.h"
 4:
 5: int main(int argc, array<string> argv) {
 6:    int argi = 1;
 7:    while (argi < argc) {
 8:       if (argi > 1) putchr (' ');
 9:       putstr (argv[argi]);
10:       argi = argi + 1;
11:    }
12:    putstr ("\n");
13:    return SUCCESS;
14: }
15:
```

```
 1: // $Id: 20-fib-array.oc,v 1.2 2019-04-17 13:23:14-07 - - $
 2: //
 3: // Put Fibonacci numbers in an array, then print them.
 4: //
 5:
 6: #include "oclib.h"
 7:
 8: #define FIB_SIZE 30
 9:
10: int main() {
11:    int index = 2;
12:    array<int> fibonacci = alloc<array<int>> (FIB_SIZE);
13:    fibonacci[0] = 0;
14:    fibonacci[1] = 1;
15:    while (index < FIB_SIZE) {
16:       fibonacci[index] = fibonacci[index - 1] + fibonacci[index - 2];
17:       index = index + 1;
18:    }
19:    index = 0;
20:    putstr ("Numeri di figlio Bonacci\n");
21:    while (index < FIB_SIZE) {
22:       putstr ("fibonacci[");
23:       putint (index);
24:       putstr ("] = ");
25:       putint (fibonacci[index]);
26:       putchr ('\n');
27:       index = index + 1;
28:    }
29:    return SUCCESS;
30: }
31:
```

```
 1: // $Id: 21-eratosthenes.oc,v 1.1 2019-04-16 12:14:45-07 - - $
 2:
 3: #include "oclib.h"
 4:
 5: #define SIZE 100
 6: #define LOWPRIME 2
 7:
 8: int main() {
 9:     int prime = LOWPRIME;
10:     int index = LOWPRIME;
11:     array<int> sieve = alloc<array<int>>(SIZE);
12:
13:     while (index < SIZE) {
14:         sieve[index] = TRUE;
15:         index = index + 1;
16:     }
17:
18:     while (prime < SIZE) {
19:         if (sieve[prime]) {
20:             index = prime * 2;
21:             while (index < SIZE) {
22:                 sieve[index] = FALSE;
23:                 index = index + prime;
24:             }
25:         }
26:         prime = prime + 1;
27:     }
28:
29:     index = LOWPRIME;
30:     while (index < SIZE) {
31:         if (sieve[index]) {
32:             putint (index);
33:             putchr ('\n');
34:         }
35:         index = index + 1;
36:     }
37:
38:     return SUCCESS;
39: }
40:
```

```
 1: // $Id: 23-atoi.oc,v 1.1 2019-04-16 12:14:45-07 - - $
 2:
 3: #include "oclib.h"
 4:
 5: int atoi (string str) {
 6:     int neg = FALSE;
 7:     int num = 0;
 8:     int digit = 0;
 9:     int contin = FALSE;
10:     int c = 0;
11:     assert (str != nullptr);
12:     if (str[0] != '\0') {
13:        if (str[0] == '-') {
14:           digit = digit + 1;
15:           neg = TRUE;
16:        }
17:        contin = TRUE;
18:        while (contin) {
19:           if (str[digit] == '\0') {
20:              contin = FALSE;
21:           }else {
22:              c = str[digit];
23:              digit = digit + 1;
24:              if (c < '0') contin = FALSE;
25:              else if (c > '9') contin = FALSE;
26:              else num = num * 10 + c - '0';
27:           }
28:        }
29:        if (neg) num = - num;
30:     }
31:     return num;
32: }
33:
34: int main (int argc, array<string> argv) {
35:     int argi = 1;
36:     string arg = nullptr;
37:     while (argi < argc) {
38:        arg = argv[argi];
39:        putstr (arg);
40:        putstr (" = ");
41:        putint (atoi (arg));
42:        putchr ('\n');
43:        argi = argi + 1;
44:     }
45:     return SUCCESS;
46: }
47:
```

```
 1: // $Id: 30-fac-fnloop.oc,v 1.1 2019-04-16 12:14:45-07 - - $
 2: //
 3: // Function uses a loop to compute factorial.
 4: //
 5:
 6: #include "oclib.h"
 7:
 8: int fac (int n) {
 9:    int f = 1;
10:    while (n > 1) {
11:       f = f * n;
12:       n = n - 1;
13:    }
14:    return f;
15: }
16:
17: int main() {
18:    int n = 1;
19:    while (n <= 5) {
20:       putint (fac (n));
21:       putchr ('\n');
22:       n = n + 1;
23:    }
24:    return SUCCESS;
25: }
26:
```

```
 1: // $Id: 31-fib-2supn.oc,v 1.1 2019-04-16 12:14:45-07 - - $
 2: //
 3: // Very slow program, computes Fibonacci numbers with O(2^n) speed.
 4: //
 5:
 6: #include "oclib.h"
 7:
 8: int fibonacci (int n) {
 9:    if (n < 2) return n;
10:    return fibonacci (n - 1) + fibonacci (n - 2);
11: }
12:
13: int main() {
14:    int n = 0;
15:    while (n < 10) {
16:       putstr ("fibonacci(");
17:       putint (n);
18:       putstr (") = ");
19:       putint (fibonacci (n));
20:       putchr ('\n');
21:       n = n + 1;
22:    }
23:    return SUCCESS;
24: }
25:
```

```
 1: // $Id: 40-arraystack.oc,v 1.2 2019-04-17 13:23:14-07 - - $
 2:
 3: #include "oclib.h"
 4:
 5: #define EMPTY (-1)
 6:
 7: struct stack {
 8:    array<string> data;
 9:    int size;
10:    int top;
11: };
12:
13: ptr<struct stack> new_stack (int size) {
14:    ptr<struct stack> stack = alloc<struct stack>();
15:    stack->data = alloc<array<string>> (size);
16:    stack->size = size;
17:    stack->top = EMPTY;
18:    return stack;
19: }
20:
21: void push (ptr<struct stack> stack, string str) {
22:    assert (stack->top < stack->size - 1);
23:    stack->top = stack->top + 1;
24:    stack->data[stack->top] = str;
25: }
26:
27: string pop (ptr<struct stack> stack) {
28:    string tmp = stack->data[stack->top];
29:    assert (stack->top != EMPTY);
30:    stack->top = stack->top - 1;
31:    return tmp;
32: }
33:
34: int empty (ptr<struct stack> stack) {
35:    return stack->top == EMPTY;
36: }
37:
38: int main (int argc, array<string> argv) {
39:    ptr<struct stack> stack = new_stack (100);
40:    int argi = 0;
41:    while (argi < argc) {
42:       push (stack, argv[argi]);
43:       argi = argi + 1;
44:    }
45:    while (not empty (stack)) {
46:       putstr (pop (stack));
47:       putchr ('\n');
48:    }
49:    return SUCCESS;
50: }
51:
```

```
 1: // $Id: 41-linkedstack.oc,v 1.2 2019-04-17 13:23:14-07 - - $
 2:
 3: #include "oclib.h"
 4:
 5: struct node {
 6:    string data;
 7:    ptr<node> link;
 8: };
 9:
10: struct stack {
11:    ptr<node> top;
12: };
13:
14: int empty (ptr<struct stack> stack) {
15:    assert (stack != nullptr);
16:    return stack->top == nullptr;
17: }
18:
19: ptr<struct stack> new_stack() {
20:    ptr<struct stack> stack = alloc<struct stack>();
21:    stack->top = nullptr;
22:    return stack;
23: }
24:
25: void push (ptr<struct stack> stack, string str) {
26:    ptr<node> tmp = alloc<node>();
27:    assert (stack != nullptr);
28:    tmp->data = str;
29:    tmp->link = stack->top;
30:    stack->top = tmp;
31: }
32:
33: string pop (ptr<struct stack> stack) {
34:    string tmp = stack->top->data;
35:    assert (stack != nullptr);
36:    assert (not empty (stack));
37:    stack->top = stack->top->link;
38:    return tmp;
39: }
40:
41: int main (int argc, array<string> argv) {
42:    int argi = 0;
43:    ptr<struct stack> stack = new_stack();
44:    while (argi < argc) {
45:       push (stack, argv[argi]);
46:       argi = argi + 1;
47:    }
48:    while (not empty (stack)) {
49:       putstr (pop (stack));
50:       putchr ('\n');
51:    }
52:    return SUCCESS;
53: }
54:
```

```
 1: // $Id: 42-viiiqueens.oc,v 1.1 2019-04-16 12:14:45-07 - - $
 2:
 3: #include "oclib.h"
 4:
 5: #define BOARD_SIZE 8
 6: array<int> board = nullptr;
 7:
 8: int is_safe (int newcol) {
 9:     int col = 0;
10:     int diagonal = 0;
11:     while (col < newcol) {
12:         if (board[col] == board[newcol]) return FALSE;
13:         diagonal = board[col] - board[newcol];
14:         if (diagonal == col - newcol) return FALSE;
15:         if (diagonal == newcol - col) return FALSE;
16:         col = col + 1;
17:     }
18:     return TRUE;
19: }
20:
21: void printqueens() {
22:     int col = 0;
23:     while (col < BOARD_SIZE) {
24:         putchr (board[col] + '1');
25:         col = col + 1;
26:     }
27:     putchr ('\n');
28: }
29:
30: void queens (int newcol) {
31:     int row = 0;
32:     if (newcol == BOARD_SIZE) printqueens();
33:     else {
34:         while (row < BOARD_SIZE) {
35:             board[newcol] = row;
36:             if (is_safe (newcol)) queens (newcol + 1);
37:             row = row + 1;
38:         }
39:     }
40: }
41:
42: int main() {
43:     board = alloc<array<int>> (BOARD_SIZE);
44:     queens (0);
45:     return SUCCESS;
46: }
47:
```

```
 1: // $Id: 44-dot-product.oc,v 1.1 2019-04-16 12:14:45-07 - - $
 2:
 3: #include "oclib.h"
 4:
 5: int dot_product (int size, array<int> vec1, array<int> vec2) {
 6:     int index = 0;
 7:     int dot = 0;
 8:     while (index < size) {
 9:         dot = dot + vec1[index] * vec2[index];
10:         index = index + 1;
11:     }
12:     return dot;
13: }
14:
15: #define SIZE 10
16:
17: int main() {
18:     array<int> vec1 = alloc<array<int>> (SIZE);
19:     array<int> vec2 = alloc<array<int>> (SIZE);
20:     int i = 0;
21:     while (i < SIZE) {
22:         vec1[i] = i + 10;
23:         vec2[i] = i * 10;
24:         i = i + 1;
25:     }
26:     putint (dot_product (SIZE, vec1, vec2));
27:     putchr ('\n');
28:     return SUCCESS;
29: }
30:
```

```
 1: // $Id: 45-towers-of-hanoi.oc,v 1.1 2019-04-16 12:14:45-07 - - $
 2:
 3: #include "oclib.h"
 4:
 5: void move (string src, string dst) {
 6:    putstr ("Move a disk from the ");
 7:    putstr (src);
 8:    putstr (" to the ");
 9:    putstr (dst);
10:    putstr (".\n");
11: }
12:
13: void towers (int ndisks, string src, string tmp, string dst) {
14:    if (ndisks < 1) return;
15:    towers (ndisks - 1, src, dst, tmp);
16:    move (src, dst);
17:    towers (ndisks - 1, tmp, src, dst);
18: }
19:
20: int main() {
21:    towers (4, "source", "temporary", "distination");
22:    return SUCCESS;
23: }
24:
```

```
 1: // $Id: 51-stringcat.oc,v 1.1 2019-04-16 12:14:45-07 - - $
 2: //
 3: // Allocate and concatenate strings.
 4: //
 5:
 6: #include "oclib.h"
 7:
 8: int strlen (string str) {
 9:    int len = 0;
10:    while (str[len] != '\0') len = len + 1;
11:    return len;
12: }
13:
14: void strcat (string dest, string src) {
15:    int pos = strlen (dest);
16:    int srcix = 0;
17:    while (src[srcix] != '\0') {
18:       dest[pos] = src[srcix];
19:       pos = pos + 1;
20:       srcix = srcix + 1;
21:    }
22:    dest[pos] = '\0';
23: }
24:
25: int main (int argc, array<string> argv) {
26:    int length = 0;
27:    int argi = 1;
28:    string concat = nullptr;
29:    while (argi < argc) {
30:       length = length + strlen (argv[argi]) + 2;
31:       argi = argi + 1;
32:    }
33:    putstr ("concat length = ");
34:    putint (length);
35:    putchr ('\n');
36:    concat = alloc<string> (length);
37:    argi = 1;
38:    putchr ('[');
39:    while (argi < argc) {
40:       strcat (concat, "(");
41:       strcat (concat, argv[argi]);
42:       strcat (concat, ")");
43:       argi = argi + 1;
44:    }
45:    putstr (concat);
46:    putstr ("]\n");
47:    return SUCCESS;
48: }
```

```
 1: // $Id: 53-insertionsort.oc,v 1.3 2019-04-16 13:44:49-07 - - $
 2: //
 3: // Use insertion sort to print argv in sorted order.
 4: //
 5:
 6: #include "oclib.h"
 7:
 8: int strcmp (string s1, string s2) {
 9:     int index = 0;
10:     int contin = TRUE;
11:     int s1c = 0;
12:     int s2c = 0;
13:     int cmp = 0;
14:     while (contin) {
15:         s1c = s1[index];
16:         s2c = s2[index];
17:         cmp = s1c - s2c;
18:         if (cmp != 0) return cmp;
19:         if (s1c == '\0') contin = FALSE;
20:         if (s2c == '\0') contin = FALSE;
21:         index = index + 1;
22:     }
23:     return 0;
24: }
25:
26: void insertion_sort (int size, array<string> array) {
27:     int sorted = 1;
28:     int slot = 0;
29:     string element = nullptr;
30:     int contin = FALSE;
31:     while (sorted < size) {
32:         slot = sorted;
33:         element = array[slot];
34:         contin = TRUE;
35:         while (contin) {
36:             if (slot == 0) {
37:                 contin = FALSE;
38:             }else if (strcmp (array[slot - 1], element) <= 0) {
39:                 contin = FALSE;
40:             }else {
41:                 array[slot] = array[slot - 1];
42:                 slot = slot - 1;
43:             }
44:         }
45:         array[slot] = element;
46:         sorted = sorted + 1;
47:     }
48: }
49:
```

```
50:
51: void print_array (int size, array<string> array) {
52:     int index = 0;
53:     while (index < size) {
54:         putstr (array[index]);
55:         putchr ('\n');
56:         index = index + 1;
57:     }
58: }
59:
60: int read_words (int size, array<string> words) {
61:     int count = 0;
62:     string word = nullptr;
63:     while (TRUE) {
64:         if (count == size) return count;
65:         word = getstr();
66:         if (word == nullptr) return count;
67:         words[count] = word;
68:         count = count + 1;
69:     }
70: }
71:
72: int main() {
73:     int count = 100;
74:     array<string> words = alloc<array<string>>(count);
75:     count = read_words (count, words);
76:     insertion_sort (count, words);
77:     print_array (count, words);
78:     return SUCCESS;
79: }
80:
```

```
 1: # $Id: Makefile,v 1.15 2019-04-17 13:44:05-07 - - $
 2:
 3: UTILDIR = /afs/cats.ucsc.edu/courses/cmps104a-wm/bin
 4:
 5: NOWARN  = -Wno-write-strings -Wno-main
 6: OCGPP   = g++ ${NOWARN} -x c++ -include ocdefs.h
 7:
 8: OCSRC   = ${sort ${wildcard *.oc}}
 9: LIBSRC  = oclib.c
10: LIBOBJ  = ${LIBSRC:.c=.o}
11: HEADERS = oclib.h ocdefs.h
12: OCOBJ   = ${OCSRC:.oc=.o}
13: EXECS   = ${OCSRC:.oc=.elf}
14: ALLSRC  = ${HEADERS} ${LIBSRC} ${OCSRC} Makefile
15: LISTING = Listing.oc-programs
16:
17: all : ${EXECS}
18:
19: %.elf : %.o ${LIBOBJ}
20:         g++ $< ${LIBOBJ} -o $@
21:
22: %.o : %.oc ${HEADERS}
23:         ${OCGPP} -c $<
24:
25: ${LIBOBJ} : ${LIBSRC}
26:         gcc -c $<
27:
28: spotless : clean
29:         - rm ${LISTING}.{ps,pdf} ${EXECS}
30:
31: clean :
32:         -rm ${OCOBJ} ${LIBOBJ}
33:
34: ci :
35:         ${UTILDIR}/cid + ${ALLSRC}
36:
37: lis :
38:         ${UTILDIR}/checksource ${ALLSRC}
39:         ${UTILDIR}/mkpspdf ${LISTING}.ps ${ALLSRC}
40:
41: again :
42:         make --no-print-directory clean ci all lis
```