```c
/* ==========================================================================
 * Fibonacci numbers, computed inefficiently and only for a small range.
 * This program is meant to show the use of different kinds of memory.
 * ==========================================================================
 */
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>

static const unsigned short max_input = 23;

static unsigned short* fibo_cache = NULL;
static size_t          fibo_cache_size = 0;

/* compute Fibonacci numbers recursively, optionally using a cache */
static unsigned short fibo(unsigned short n) {
  if ((n < 0) || (n > max_input)) {
    fprintf(stderr, "Error %s() argument %u > %u\t%p\n",
            __func__, n, max_input, (void*) &max_input);
    return 0;
  }

  if (n < 2)
     return 1;
  if (fibo_cache && (n < fibo_cache_size) && fibo_cache[n])
     return fibo_cache[n];

  fprintf(stderr, ">>> %s(%2u)\t\t%p\n", __func__, n, (void*)&n);

  const unsigned short result = fibo(n-1) + fibo(n-2);

  if (fibo_cache && (n < fibo_cache_size))
     fibo_cache[n] = result;

  fprintf(stderr, "<<< %s(%2u) =%5u\n", __func__, n, result);
  return result;
}


/* set up a cache of the required size, keep existing cache if big enough */
static void prepare_cache(unsigned short index) {
  if (index < fibo_cache_size)
     return;

  if (fibo_cache)
     free(fibo_cache);

  // need one element more to access fibo_cache[index]
  const size_t size = index+1;

  fibo_cache = calloc(size, sizeof(unsigned short));
  if (fibo_cache) {
     fibo_cache_size = size;
     fprintf(stderr, "fibo_cache allocated\t%p\nfibo_cache_size %zu\t%p\n",
             (void*)fibo_cache, size, (void*) &fibo_cache_size);
  } else {
    fibo_cache_size = 0; // memory allocation failed
    fprintf(stderr, "fibo_cache allocation failed\n");
  }
}
```

```
/* command-line arguments should be integer numbers
 * >=0  computes the respective Fibonacci number, if in range
 * < 0  sleeps for the given number of seconds
 */
int main(int argc, const char* argv[]) {
  for (int i = 1; i < argc; i++) {
    char* end = NULL;
    long argument = strtol(argv[i], &end, 0);
    if (*end != 0) {
      fprintf(stderr, "invalid argument #%i: '%s'\n", i, argv[i]);
      continue;
    }
    if (argument < 0) {
      fprintf(stderr, "\nsleeping %li seconds, so you can look at:\n"
              "cat /proc/%u/maps\n", -argument, getpid());
      sleep(-argument);
      continue;
    }
    unsigned short input = argument;
    prepare_cache(input);
    unsigned short result = fibo(input);
    printf("%2u -> %5u\n", input, result);
  }
}

/*
compiling:
gcc   -Wall fibonacci.c -o fibonacci
clang -Wall fibonacci.c -o fibonacci

example invocation:
./fibonacci 8 -300

source to PDF:
enscript -GEc -M A4 -f Courier11 fibonacci.c -o - | ps2pdf - fibonacci.c.pdf
*/
```