```c
// $Id: vstring.h,v 1.2 2021-12-20 12:42:21-08 - - $

//
// Buffer to accumulate a variable sized character string.
//
// vstring_new - allocate a new vstring
// vstring_delete - delete the vstring
// vstring_clear - set the vstring to the empty string
// vstring_append - append a character to the vstring
// vstring_peek - return a peek of the contents of the string
//

#ifndef VSTRING_H
#define VSTRING_H

typedef struct vstring vstring;

vstring* vstring_new (void);
void vstring_delete (vstring*);
void vstring_clear (vstring*);
void vstring_append (vstring*, char);
const char* vstring_peek (vstring*);

#endif

```

```c
// $Id: wordlist.h,v 1.2 2021-12-20 12:42:21-08 - - $

//
// Sorted list of word and frequency counts.
//
// wordlist_new - allocate a new list
// wordlist_delete - free the list
// wordlist_incr - increment the word count, insert if not there
//

#ifndef WORDLIST_H
#define WORDLIST_H

typedef struct wordlist wordlist;

wordlist* wordlist_new (void);
void wordlist_delete (wordlist*);
void wordlist_incr (wordlist*, const char*);
void wordlist_print (wordlist* list);

#endif

```

```c
// $Id: vstring.c,v 1.1 2019-12-17 14:53:52-08 - - $

#include <assert.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include "vstring.h"

#define INIT_SIZE 16
struct vstring {
   char* data;
   size_t strlen;
   size_t size;
};

vstring* vstring_new (void) {
   vstring* vstr = malloc (sizeof (vstring));
   assert (vstr != NULL);
   vstr->strlen = 0;
   vstr->size = INIT_SIZE;
   vstr->data = malloc (vstr->size);
   assert (vstr->data != NULL);
   vstr->data[0] = '\0';
   return vstr;
}

void vstring_delete (vstring* vstr) {
   assert (vstr != NULL);
   free (vstr->data);
   free (vstr);
}

void vstring_clear (vstring* vstr) {
   assert (vstr != NULL);
   vstr->strlen = 0;
   vstr->data[0] = '\0';
}

void vstring_append (vstring* vstr, char chr) {
   assert (vstr != NULL);
   assert (vstr->data != NULL);
   assert (vstr->strlen < vstr->size);
   vstr->data[vstr->strlen++] = chr;
   if (vstr->strlen == vstr->size) {
      vstr->size *= 2;
      vstr->data = realloc (vstr->data, vstr->size);
      assert (vstr->data != NULL);
   }
   vstr->data[vstr->strlen] = '\0';
}

const char* vstring_peek (vstring* vstr) {
   assert (vstr != NULL);
   return vstr->data;
}

```

```c
// $Id: wordlist.c,v 1.1 2019-12-17 14:53:52-08 - - $

#include <assert.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include "wordlist.h"

typedef struct node node;
struct node {
   char* word;
   size_t count;
   node* link;
};
struct wordlist {
   node* head;
};

wordlist* wordlist_new (void) {
   wordlist* list = malloc (sizeof (wordlist));
   assert (list != NULL);
   list->head = NULL;
   return list;
}

void wordlist_delete (wordlist* list) {
   while (list->head != NULL) {
      node* tmp = list->head;
      list->head = tmp->link;
      free (tmp->word);
      free (tmp);
   }
   free (list);
}

void wordlist_incr (wordlist* list, const char* word) {
   node** currp = &(list->head);
   while (*currp != NULL && strcmp ((*currp)->word, word) < 0) {
      currp = &(*currp)->link;
   }
   if (*currp == NULL || strcmp (word, (*currp)->word) < 0) {
      node* link = *currp;
      *currp = malloc (sizeof (node));
      assert (*currp != NULL);
      (*currp)->word = strdup (word);
      assert ((*currp)->word != NULL);
      (*currp)->count = 0;
      (*currp)->link = link;
   }
   ++(*currp)->count;
}

void wordlist_print (wordlist* list) {
   for (node* itor = list->head; itor != NULL; itor = itor->link) {
      printf ("%s %zd\n", itor->word, itor->count);
   }
}
```

```c
// $Id: main.c,v 1.1 2019-12-17 14:53:52-08 - - $

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

#include "vstring.h"
#include "wordlist.h"

void countwords (FILE* infile, wordlist* list) {
   vstring* vstr = vstring_new();
   bool in_word = false;
   for (;;) {
      int chr = fgetc (infile);
      if (isalpha (chr)) {
         in_word = true;
         vstring_append (vstr, tolower (chr));
      }else {
         if (in_word) wordlist_incr (list, vstring_peek (vstr));
         in_word = false;
         vstring_clear (vstr);
      }
      if (chr == EOF) break;
   }
   vstring_delete (vstr);
}

int main (int argc, char** argv) {
   int exit_status = EXIT_SUCCESS;
   const char* exec_name = basename (argv[0]);
   wordlist* list = wordlist_new();
   if (argc == 1) {
      countwords (stdin, list);
   }else {
      for (int argi = 1; argi != argc; ++argi) {
         const char* filename = argv[argi];
         FILE* infile = fopen (filename, "r");
         if (infile == NULL) {
            exit_status = EXIT_FAILURE;
            fprintf (stderr, "%s: %s: %s\n",
                     exec_name, filename, strerror (errno));
         }else {
            countwords (infile, list);
            fclose (infile);
         }
      }
   }
   wordlist_print (list);
   wordlist_delete (list);
   return exit_status;
}

```

```
# $Id: Makefile,v 1.2 2021-08-11 21:56:43-07 - - $

GCC     = gcc -g -Wall -Wextra -std=gnu11
GRIND   = valgrind --leak-check=full --show-reachable=yes
NODEPS  = ${filter ci clean spotless tar, ${MAKECMDGOALS}}
MKTAR   = gtar --create --verbose --gzip

H_FILES = vstring.h wordlist.h
C_FILES = vstring.c wordlist.c main.c
OBJECTS = ${C_FILES:.c=.o}
EXECBIN = countwords
SOURCES = ${H_FILES} ${C_FILES} Makefile data


all : ${EXECBIN}

${EXECBIN} : ${OBJECTS}
        ${GCC} -o $@ $^

%.o : %.c
        - checksource $<
        ${GCC} -c $<

ci : ${SOURCES}
        - checksource $^
        cid -is $^

clean :
        - rm --force ${OBJECTS} test.log test.out test.err

lis : ${SOURCES} Makefile.deps
        mkpspdf Listing.ps $^

spotless : clean
        - rm --force ${EXECBIN} Listing.{ps,pdf} Makefile.deps

tar : ${SOURCES}
        ${MAKE} --no-print-directory spotless
        ( DIRNAME=$$(basename $$(pwd)) \
        ; cd .. \
        ; ${MKTAR} --exclude=RCS --file=countwords.tar.gz $$DIRNAME \
        )

test : ${EXECBIN}
        ${GRIND} --log-file=test.log \
                 ${EXECBIN} ${SOURCES} 1>test.out 2>test.err

Makefile.deps :
        ${GCC} -MM ${C_FILES} >Makefile.deps

ifeq (${NODEPS}, )
include Makefile.deps
endif

```

```
This is a test of data.
Data of is this a test?
Data of is this a test?
Data of is this a test?
what is this testing?
what is this testing?
$Id: data,v 1.2 2022-03-21 13:33:14-07 - - $
```

```
vstring.o: vstring.c vstring.h
wordlist.o: wordlist.c wordlist.h
main.o: main.c vstring.h wordlist.h
```