VE482 Lab 4

Liu Yihao 515370910207

1 Layer programming

- The program can be divided into three layers, what are they?
  The kernel layer, the logic layer and the interface layer.
- Split the program into files according to the defined layers.
  The kernel layer in list.c/.h, the logic layer in logic.c/.h and the interface layer in interface.c/.h.
- Create the appropriate corresponding header files.
  list.h

```
//
// Created by liu on 2017/10/15.
//

#ifndef PROJECT_LIST_H
#define PROJECT_LIST_H

#include <stdio.h>

typedef struct node {
    char *str;
    void *data;
    struct node *next;
} node_t;

typedef struct list {
    struct node *first;
    size_t length;
} list_t;

void list_init(list_t **list);

void list_free(list_t *list);

node_t *list_insert(list_t *list, node_t *node, char *str, void *data);

node_t *list_insert_first(list_t *list, char *str, void *data);

const node_t *list_search(list_t *list, const node_t *first, const void
 ↪  *data, int (*cmp)(const void *, const void *));
```

```
30
31  void list_sort(list_t *list, int (*cmp)(const void *, const void *));
32
33  void list_print(const list_t *list, FILE *file, void(*print)(FILE* file,
    ↪  const void *));
34
35  #endif //PROJECT_LIST_H
```

logic.h

```
1   //
2   // Created by liu on 2017/10/20.
3   //
4
5   #ifndef PROJECT_API_H
6   #define PROJECT_API_H
7
8   #include "list.h"
9
10  typedef enum {
11      VAR_INT,
12      VAR_DOUBLE,
13      VAR_STRING,
14      VAR_SIZE
15  } VAR_TYPE;
16
17  typedef enum {
18      SORT_INC,
19      SORT_DEC,
20      SORT_RAND,
21      SORT_SIZE
22  } SORT_TYPE;
23
24  static const char *TYPE_NAME[VAR_SIZE] = {
25          "int.txt", "double.txt", "string.txt"
26  };
27
28  static const char *SORT_NAME[SORT_SIZE] = {
29          "inc", "dec", "rand"
30  };
31
32  void generate_filename(char *buffer, VAR_TYPE var_type, SORT_TYPE
    ↪  sort_type);
33
34  VAR_TYPE get_var_type(const char *filename);
35
36  SORT_TYPE get_sort_type(const char* str);
37
38  void read_and_sort(VAR_TYPE var_type, SORT_TYPE sort_type);
39
```

```
#endif // PROJECT_API_H
```

- If necessary rewrite functions such that no call is emitted from lower level functions to upper level functions.

  list.c

```
//
// Created by liu on 2017/10/15.
//

#include <stdlib.h>
#include <string.h>
#include "list.h"

void list_init(list_t **list) {
    *list = malloc(sizeof(list_t));
    (*list)->first = NULL;
    (*list)->length = 0;
}

void list_free(list_t *list) {
    node_t *temp = list->first;
    for (int i = 0; i < list->length; i++) {
        temp = temp->next;
        free(list->first->str);
        free(list->first->data);
        free(list->first);
        list->first = temp;
    }
    free(list);
}

node_t *list_insert(list_t *list, node_t *node, char *str, void *data) {
    node_t *new_node = malloc(sizeof(node_t));
    new_node->str = str;
    new_node->data = data;
    list->length++;
    new_node->next = node->next;
    node->next = new_node;
    return new_node;
}

node_t *list_insert_first(list_t *list, char *str, void *data) {
    node_t *new_node = malloc(sizeof(node_t));
    new_node->str = str;
    new_node->data = data;
    list->length++;
    if (list->first) {
        new_node->next = list->first;
    } else {
```

```
45          new_node->next = NULL;
46      }
47      list->first = new_node;
48      return new_node;
49  }
50
51  const node_t *list_search(list_t *list, const node_t *first, const void
    ↪  *data, int (*cmp)(const void *, const void *)) {
52      if (list->first == NULL) return NULL;
53      if (first == NULL) first = list->first;
54      else first = first->next;
55      while (first) {
56          if (cmp(first->data, data)) return first;
57          first = first->next;
58      }
59      return first;
60  }
61
62  void list_sort(list_t *list, int (*cmp)(const void *, const void *)) {
63      if (list->length == 0)return;
64      node_t *arr = malloc(sizeof(node_t) * list->length);
65      node_t *temp = list->first;
66      for (size_t i = 0; i < list->length; i++) {
67          memcpy(arr + i, temp, sizeof(node_t));
68          temp = temp->next;
69      }
70      qsort(arr, list->length, sizeof(node_t), cmp);
71      temp = list->first;
72      for (size_t i = 0; i < list->length; i++) {
73          temp->str = arr[i].str;
74          temp->data = arr[i].data;
75          temp = temp->next;
76      }
77      free(arr);
78  }
79
80  void list_print(const list_t *list, FILE *file, void(*print)(FILE *file,
    ↪  const void *)) {
81      node_t *temp = list->first;
82      for (size_t i = 0; i < list->length; i++) {
83          fprintf(file, "%s=", temp->str);
84          print(file, temp->data);
85          fprintf(file, "\n");
86          temp = temp->next;
87      }
88  }
```

logic.c

```
1   //
```

```c
// Created by liu on 2017/10/20.
//

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <assert.h>
#include "logic.h"

int int_inc(const void *a, const void *b) {
    int _a = *(int *) (((node_t *) a)->data);
    int _b = *(int *) (((node_t *) b)->data);
    if (_a > _b)return 1;
    if (_a < _b)return -1;
    return 0;
}

int int_dec(const void *a, const void *b) {
    return int_inc(b, a);
}

int double_inc(const void *a, const void *b) {
    double _a = *(double *) (((node_t *) a)->data);
    double _b = *(double *) (((node_t *) b)->data);
    if (_a > _b)return 1;
    if (_a < _b)return -1;
    return 0;
}

int double_dec(const void *a, const void *b) {
    return double_inc(b, a);
}

int string_inc(const void *a, const void *b) {
    return strcmp((char *) (((node_t *) a)->data), (char *) (((node_t *)
    ↪  b)->data));
}

int string_dec(const void *a, const void *b) {
    return strcmp((char *) (((node_t *) b)->data), (char *) (((node_t *)
    ↪  a)->data));
}

int all_rand(const void *a, const void *b) {
    return (rand() % 2) * 2 - 1;
}

int (*const cmp[VAR_SIZE][SORT_SIZE])(const void *, const void *) = {
```

```
        {int_inc,    int_dec,    all_rand},
        {double_inc, double_dec, all_rand},
        {string_inc, string_dec, all_rand}
};

void int_print(FILE *file, const void *a) {
    fprintf(file, "%d", *(int *) (a));
}

void double_print(FILE *file, const void *a) {
    fprintf(file, "%lf", *(double *) (a));
}

void string_print(FILE *file, const void *a) {
    fprintf(file, "%s", (char *) (a));
}

void (*const print[VAR_SIZE])(FILE *file, const void *) = {
        int_print, double_print, string_print
};

void generate_filename(char *buffer, VAR_TYPE var_type, SORT_TYPE sort_type)
 ↪ {
    strcpy(buffer, SORT_NAME[sort_type]);
    size_t length = strlen(buffer);
    buffer[length] = '_';
    strcpy(buffer + length + 1, TYPE_NAME[var_type]);
}

VAR_TYPE get_var_type(const char *filename) {
    char buffer[100] = {};
    VAR_TYPE var_type = 0;
    for (; var_type < VAR_SIZE; var_type++) {
        generate_filename(buffer, var_type, SORT_RAND);
        if (strcmp(buffer, filename) == 0) {
            break;
        }
    }
    return var_type;
}

SORT_TYPE get_sort_type(const char *str) {
    SORT_TYPE sort_type = 0;
    for (; sort_type < SORT_SIZE; sort_type++) {
        if (strcmp(SORT_NAME[sort_type], str) == 0) {
            break;
        }
    }
    return sort_type;
```

```
97  }
98
99  void read_and_sort(VAR_TYPE var_type, SORT_TYPE sort_type) {
100     char filename[20] = {0};
101     generate_filename(filename, var_type, SORT_RAND);
102     FILE *input = fopen(filename, "r");
103     if (!input) return;
104     printf("reading %s\n", filename);
105     char buffer[1024] = {0};
106     list_t *list1;
107     list_init(&list1);
108     while (!feof(input)) {
109         fgets(buffer, 1024, input);
110         char *pos = strchr(buffer, '=');
111         if (!pos) continue;
112         size_t length = pos - buffer;
113         *pos = '\0';
114         char *str = (char *) malloc(sizeof(char) * (length + 1));
115         strcpy(str, buffer);
116         void *data;
117         switch (var_type) {
118         case VAR_INT:
119             data = malloc(sizeof(int));
120             *((int *) data) = strtol(pos + 1, NULL, 10);
121             break;
122         case VAR_DOUBLE:
123             data = malloc(sizeof(double));
124             *((double *) data) = strtod(pos + 1, NULL);
125             break;
126         case VAR_STRING:
127             length = strlen(pos + 1);
128             data = malloc(sizeof(char) * (length + 1));
129             strcpy(data, pos + 1);
130             pos = data + strlen(data) - 1;
131             while (*pos == '\n') *(pos--) = '\0';
132             break;
133         default:
134             assert(0);
135             break;
136         }
137         list_insert_first(list1, str, data);
138     }
139     fclose(input);
140
141     printf("sorting elements\n");
142     list_sort(list1, cmp[var_type][sort_type]);
143
144     generate_filename(filename, var_type, sort_type);
145     printf("writing %s\n", filename);
```

```
146     FILE *output = fopen(filename, "w");
147     list_print(list1, output, print[var_type]);
148     fclose(output);
149     list_free(list1);
150 }
```

- The initial program implements a command line interface, write a “Menu interface” which (i) wel- comes the user, (ii) prompts him for some task to perform, and (iii) runs it. When a task is completed the user should (i) be informed if it was successful and then (ii) be displayed the menu. From the menu he should be able to exit the program.

- Write two main functions, one which will “dispatch” the work to another function which will run the command line user interface and a second one which will “dispatch” the work to the Menu user interface.

  interface.h

```
1  //
2  // Created by liu on 2017/10/20.
3  //
4
5  #ifndef PROJECT_INTERFACE_H
6  #define PROJECT_INTERFACE_H
7
8  int dispatch_cli(int argc, char *argv[]);
9
10 int dispatch_ui(int argc, char *argv[]);
11
12 #endif //PROJECT_INTERFACE_H
```

  interface.c

```
1  //
2  // Created by liu on 2017/10/20.
3  //
4
5  #include <stdio.h>
6  #include <stdlib.h>
7  #include <string.h>
8  #include <time.h>
9
10 #include "interface.h"
11 #include "logic.h"
12
13 int dispatch_cli(int argc, char *argv[]) {
14     srand(time(NULL));
15     if (argc < 3)return 0;
16     VAR_TYPE var_type = get_var_type(argv[1]);
17     SORT_TYPE sort_type = get_sort_type(argv[2]);
18     if (var_type < VAR_SIZE && sort_type < SORT_SIZE) {
19         read_and_sort(var_type, sort_type);
20     }
```

```
21      return 0;
22  }
23
24  int dispatch_ui(int argc, char *argv[]) {
25      srand(time(NULL));
26      char buffer[1000];
27      printf("Welcome to the menu interface!\n");
28      int exit_flag = 0;
29      while (!exit_flag) {
30          VAR_TYPE var_type = VAR_SIZE;
31          SORT_TYPE sort_type = SORT_SIZE;
32          printf("Please input one of the strings:\n");
33          printf("- rand_int.txt (read random integers)\n");
34          printf("- rand_double.txt (read random doubles)\n");
35          printf("- rand_string.txt (read random strings)\n");
36          printf("- exit (exit the program)\n");
37          while (var_type == VAR_SIZE) {
38              printf("> ");
39              fgets(buffer, 999, stdin);
40              buffer[strlen(buffer) - 1] = '\0';
41              if (strcmp(buffer, "exit") == 0) {
42                  exit_flag = 1;
43                  break;
44              }
45              var_type = get_var_type(buffer);
46              if (var_type == VAR_SIZE) {
47                  printf("Invalid input, please retry!\n");
48              }
49          }
50          if (exit_flag) break;
51          printf("Please input one of the strings:\n");
52          printf("- inc (output in increasing order)\n");
53          printf("- dec (output in decreasing order)\n");
54          printf("- rand (output in random order)\n");
55          printf("- exit (exit the program)\n");
56          while (sort_type == SORT_SIZE) {
57              printf("> ");
58              fgets(buffer, 999, stdin);
59              buffer[strlen(buffer) - 1] = '\0';
60              if (strcmp(buffer, "exit") == 0) {
61                  exit_flag = 1;
62                  break;
63              }
64              sort_type = get_sort_type(buffer);
65              if (sort_type == SORT_SIZE) {
66                  printf("Invalid input, please retry!\n");
67              }
68          }
69          if (exit_flag) break;
```

```
70          read_and_sort(var_type, sort_type);
71          printf("The operation is successful!\n\n");
72      }
73      return 0;
74  }
```

ui.c

```
1  #include "interface.h"
2
3  int main(int argc, char *argv[]) {
4      return dispatch_ui(argc, argv);
5  }
```

cli.c

```
1  #include "interface.h"
2
3  int main(int argc, char *argv[]) {
4      return dispatch_cli(argc, argv);
5  }
```

2 Libraries

- What are the three stages performed when compiling a file?

  Preprocess, compilation and link.

- Briefly describe each of them.
  - Preprocess: substitute definitions and proceed with pragmas
  - Compilation: compile source code into binary files and generate symbol table
  - Link: link binary files according symbol table
- Search more details on how to proceed.

  It's very easy to use CMake to generate static libraries with the "add_library" command.

- Create two static libraries, one for each of the two lowest layers in the previous program.

```
1  add_library(l4_list_static STATIC list.c)
2  add_library(l4_logic_static STATIC logic.c)
```

- Compile the command line version of the program using these two static libraries.

```
1  add_executable(l4_cli_static cli.c interface.c)
2  target_link_libraries(l4_cli_static l4_logic_static l4_list_static)
```

- Generate two dynamic libraries, one for each of the two lowest layers in the previous program.

```
1  add_library(l4_list_dynamic SHARED list.c)
2  add_library(l4_logic_dynamic SHARED logic.c)
3  target_link_libraries(l4_logic_dynamic l4_list_dynamic)
```

- Compile the whole program

```
add_executable(l4_cli cli.c interface.c logic.c list.c)
add_executable(l4_ui ui.c interface.c logic.c list.c)
```

- Compile the Menu version of the program using these two dynamic libraries.

```
add_executable(l4_ui_dynamic ui.c interface.c)
target_link_libraries(l4_ui_dynamic l4_api_dynamic l4_list_dynamic)
```