

Programming Methodology - SE1012

Lab 10

IT23613522

Question 1: String Comparison

```
vboxuser@Ubuntu: ~/IT23613522
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#define MAX_ITEMS 128
#define MAX_LEN 128
static void trim(char *s) {
    char *start = s;
    while (*start && isspace((unsigned char)*start)) start++;
    if (start != s) memmove(s, start, strlen(start) + 1);
    size_t len = strlen(s);
    while (len > 0 && isspace((unsigned char)s[len - 1])) s[--len] = '\0';
}
static void unquote(char *s) {
    size_t len = strlen(s);
    if (len >= 2 && ((s[0] == '"' && s[len-1] == '"') || (s[0] == '\'' && s[len-1] == '\''))) {
        memmove(s, s + 1, len - 2);
        s[len - 2] = '\0';
    }
}
int cmpstr(const void *a, const void *b) {
    const char *pa = *(const char **)a;
    const char *pb = *(const char **)b;
    return strcmp(pa, pb);
}
int main(void) {
    char line[4096];
    if (!fgets(line, sizeof line, stdin)) return 0;
    line[strlen(line)] = '\0';
    char *tokens[MAX_ITEMS];
    char *buf = malloc(strlen(line) + 1);
    if (!buf) return 1;
    strcpy(buf, line);
    int count = 0;
    char *tok = strtok(buf, ",");
    while (tok && count < MAX_ITEMS) {
        trim(tok);
        unquote(tok);
        count++;
    }
    return 0;
}
-- INSERT --
```

1,19 Top

```
vboxuser@Ubunt: ~/IT23613522
int cmpstr(const void *a, const void *b) {
    const char *pa = *(const char **)a;
    const char *pb = *(const char **)b;
    return strcmp(pa, pb);}
int main(void) {
    char line[4096];
    if (!fgets(line, sizeof line, stdin)) return 0;
    line[strcspn(line, "\n")] = '\0';
    char *tokens[MAX_ITEMS];
    char *buf = malloc(strlen(line) + 1);
    if (!buf) return 1;
    strcpy(buf, line);
    int count = 0;
    char *tok = strtok(buf, ",");
    while (tok && count < MAX_ITEMS) {
        trim(tok);
        unquote(tok);
        size_t len = strlen(tok);
        char *stored = malloc(len + 1);
        if (!stored) break;
        strcpy(stored, tok);
        tokens[count++] = stored;
        tok = strtok(NULL, ",");
    }
    if (count == 0) {
        free(buf);
        return 0;}
    qsort(tokens, count, sizeof(char *), cmpstr);
    for (int i = 0; i < count; i++) {
        if (i) printf(", ");
        printf("%s", tokens[i]);
        free(tokens[i]);
    }
    printf("\n");
    free(buf);
    return 0;}
-- INSERT --
```

51,15 Bot

```
vboxuser@Ubunt: ~/IT23613522
vboxuser@Ubunt:~/IT23613522$ gcc s.c -o s
vboxuser@Ubunt:~/IT23613522$ ./s
banana,apple,cherry
apple, banana, cherry
vboxuser@Ubunt:~/IT23613522$
```

Question 2: Arrays of Pointers

```
vboxuser@Ubuntu: ~/IT23613522
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <stdlib.h>
#define MAX_ITEMS 256
#define MAX_LEN 64
#define BUF_SIZE 4096
static void trim(char *s) {
    char *start = s;
    while (*start && isspace((unsigned char)*start)) start++;
    if (start != s) memmove(s, start, strlen(start) + 1);
    size_t len = strlen(s);
    while (len > 0 && isspace((unsigned char)s[len - 1])) s[--len] = '\0';
}
static void unquote(char *s) {
    size_t len = strlen(s);
    if (len >= 2 && ((s[0] == '"' && s[len-1] == '"') || (s[0] == '\'' && s[len-1] == '\''))) {
        memmove(s, s + 1, len - 2);
        s[len - 2] = '\0';
    }
}
int count_occurrences(char words[][MAX_LEN], int n, const char *target) {
    int count = 0;
    char (*p)[MAX_LEN] = words;
    for (int i = 0; i < n; i++, p++) {
        if (strcmp(*p, target) == 0) count++;
    }
    return count;
}
int main(void) {
    char buf[BUF_SIZE];
    if (!fgets(buf, sizeof buf, stdin)) return 0;
    buf[strcspn(buf, "\n")] = '\0';
    char *copy = malloc(strlen(buf) + 1);
    if (!copy) return 1;
    1,1 Top
```

```
vboxuser@Ubunt: ~/IT23613522
int count_occurrences(char words[][MAX_LEN], int n, const char *target) {
    int count = 0;
    char (*p)[MAX_LEN] = words;
    for (int i = 0; i < n; i++, p++) {
        if (strcmp(*p, target) == 0) count++;
    }
    return count;
}

int main(void) {
    char buf[BUF_SIZE];
    if (!fgets(buf, sizeof buf, stdin)) return 0;
    buf[strcspn(buf, "\n")] = '\0';
    char *copy = malloc(strlen(buf) + 1);
    if (!copy) return 1;
    strcpy(copy, buf);
    char words[MAX_ITEMS][MAX_LEN];
    int count = 0;
    char *tok = strtok(copy, ",");
    while (tok && count < MAX_ITEMS) {
        trim(tok);
        unquote(tok);
        strncpy(words[count], tok, MAX_LEN - 1);
        words[count][MAX_LEN - 1] = '\0';
        count++;
        tok = strtok(NULL, ",");
    }
    free(copy);
    if (!fgets(buf, sizeof buf, stdin)) return 0;
    buf[strcspn(buf, "\n")] = '\0';
    trim(buf);
    unquote(buf);
    int occurrences = count_occurrences(words, count, buf);
    printf("The word '%s' appears %d %s.\n", buf, occurrences, occurrences == 1 ? "time" : "times");
    return 0;
}
```

56,1 Bot

```
vboxuser@Ubunt: ~/IT23613522
vboxuser@Ubunt:~/IT23613522$ vim s.c
vboxuser@Ubunt:~/IT23613522$ gcc s.c -o s
vboxuser@Ubunt:~/IT23613522$ ./s
apple, apple, banana, apple
apple
The word 'apple' appears 3 times.
vboxuser@Ubunt:~/IT23613522$
```

Question 3: Character Operations

```
vboxuser@Ubuntu: ~/IT23613522

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

int main(void) {
    char s[1024];
    if (!fgets(s, sizeof s, stdin)) return 0;
    s[strcspn(s, "\n")] = '\0';
    int alphabets = 0;
    int digits = 0;
    int special = 0;
    for (size_t i = 0; s[i]; i++) {
        unsigned char c = (unsigned char)s[i];
        if (isalpha(c)) alphabets++;
        else if (isdigit(c)) digits++;
        else if (ispunct(c)) special++;
    }
    printf("Alphabets: %d, Digits: %d, Special characters: %d\n", alphabets, digits, special);
    return 0;
}
```

```
vboxuser@Ubuntu: ~/IT23613522

vboxuser@Ubuntu:~/IT23613522$ vim s.c
vboxuser@Ubuntu:~/IT23613522$ gcc s.c -o s
vboxuser@Ubuntu:~/IT23613522$ ./s
Hello123!
Alphabets: 5, Digits: 3, Special characters: 1
vboxuser@Ubuntu:~/IT23613522$
```

Question 4: Using Strings in Programs

```
vboxuser@Ubuntu: ~/IT23613522

vboxuser@Ubuntu:~/IT23613522$ vim s.c
vboxuser@Ubuntu:~/IT23613522$ gcc s.c -o s
vboxuser@Ubuntu:~/IT23613522$ ./s
first line
second line
third line

1. first line
2. second line
3. third line
vboxuser@Ubuntu:~/IT23613522$
```

```
vboxuser@Ubuntu: ~/IT23613522
#include <ctype.h>
#define MAX_LINES 100
#define MAX_LEN 1024
void trim_newline(char *s) {
    size_t l = strlen(s);
    while (l > 0 && (s[l-1] == '\n' || s[l-1] == '\r')) { s[l-1] = '\0'; l--; }
}
void trim_spaces(char *s) {
    char *start = s;
    while (*start && isspace((unsigned char)*start)) start++;
    if (start != s) memmove(s, start, strlen(start) + 1);
    size_t len = strlen(s);
    while (len > 0 && isspace((unsigned char)s[len-1])) s[--len] = '\0';
}
int main(void) {
    char lines[MAX_LINES][MAX_LEN];
    char buf[MAX_LEN];
    int count = 0;
    while (count < MAX_LINES) {
        if (!fgets(buf, sizeof buf, stdin)) break;
        trim_newline(buf);
        trim_spaces(buf);
        if (buf[0] == '\0') break;
        strncpy(lines[count], buf, MAX_LEN - 1);
        lines[count][MAX_LEN - 1] = '\0';
        count++;
    }
    for (int i = 0; i < count; i++) {
        char out[MAX_LEN + 32];
        sprintf(out, "%d. %s", i + 1, lines[i]);
        printf("%s\n", out);
    }
    return 0;
}
```

36,1 Bot

Question 5: The Nature of Recursion

```
vboxuser@Ubuntu: ~/IT23613522
#include <stdio.h>
#include <stdlib.h>

int sum_digits(int n) {
    if (n < 0) n = -n;
    if (n == 0) return 0;
    return (n % 10) + sum_digits(n / 10);
}

int main(void) {
    int n;
    if (scanf("%d", &n) != 1) return 0;
    printf("Sum of digits: %d\n", sum_digits(n));
    return 0;
}
```

15,1 All

```
vboxuser@Ubuntu: ~/IT23613522
vboxuser@Ubuntu:~/IT23613522$ vim s.c
vboxuser@Ubuntu:~/IT23613522$ gcc s.c -o s
vboxuser@Ubuntu:~/IT23613522$ ./s
345
Sum of digits: 12
vboxuser@Ubuntu:~/IT23613522$
```

Question 6: Sum of Array Elements Using Recursion (With Tracing)

```
vboxuser@Ubuntu: ~/IT23613522
#include <stdio.h>

int sumArray(int arr[], int index, int size) {
    printf("Entering sumArray with index = %d, size = %d\n", index, size);
    if (index == size) return 0;
    int s = arr[index] + sumArray(arr, index + 1, size);
    printf("sumArray(index = %d, size = %d) returning: sum = %d\n", index, size, s);
    return s;
}

int main(void) {
    int arr[] = {1, 2, 3, 4, 5};
    int size = sizeof arr / sizeof arr[0];
    int total = sumArray(arr, 0, size);
    printf("Sum of array elements: %d\n", total);
    return 0;
}

17,1 All
```

```
vboxuser@Ubuntu: ~/IT23613522
vboxuser@Ubuntu:~/IT23613522$ vim s.c
vboxuser@Ubuntu:~/IT23613522$ gcc s.c -o s
vboxuser@Ubuntu:~/IT23613522$ ./s
Entering sumArray with index = 0, size = 5
Entering sumArray with index = 1, size = 5
Entering sumArray with index = 2, size = 5
Entering sumArray with index = 3, size = 5
Entering sumArray with index = 4, size = 5
Entering sumArray with index = 5, size = 5
sumArray(index = 4, size = 5) returning: sum = 5
sumArray(index = 3, size = 5) returning: sum = 9
sumArray(index = 2, size = 5) returning: sum = 12
sumArray(index = 1, size = 5) returning: sum = 14
sumArray(index = 0, size = 5) returning: sum = 15
Sum of array elements: 15
vboxuser@Ubuntu:~/IT23613522$
```