VG101 — Intoduction to Computers & Programming

Lab 6

Instructor: Manuel Charlemagne

TA: Yihao Liu – UM-JI (Summer 2018)

Goals of the lab

- Loop and Control Statements
- File I/O in C
- Arrays and Pointers
- Usage of new Library

1 Introduction

In the Shanghai International Film Festival, Krystor, Frank and Simon enjoyed the films very much. However, they can't always find good seats in them. After going to many different cinemas, they decided to design a seat selecting system to simulate the procedure of selecting seats and discover how they can find the best seat.

2 Working Flow

2.1 Select Seats

In Figure 1, the structure of a typical cinema hall is shown, and three seats are selected.

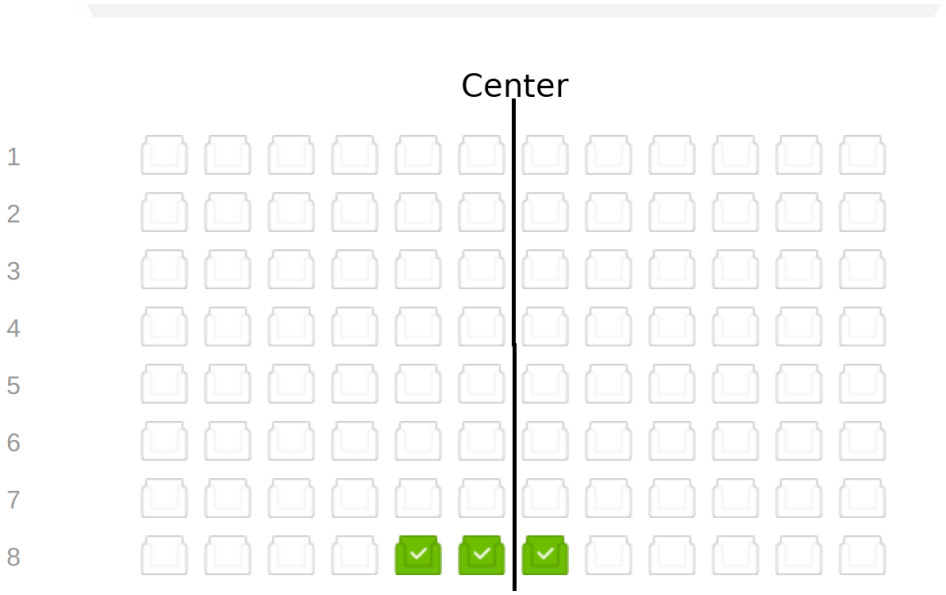

Figure 1: Structure of a typical cinema hall.

When people are going to see a movie, they always want to sit together, and prefer the seats in the middle and back of the cinema. In Figure 1, Krystor, Frank and Simon selected the three best seat in the hall. The rule how they select the seats is:

- They always select the n consecutive seats on a row, where n is the number of them. The seats are consecutive means that they are next to each other on the axis.
- They always select the seats closest to the middle back of the cinema hall, which means, the distance between the centroid of the seats and the middle back point (row 8, column between 6 and 7 in Figure 1) should be smallest.
- When two selections on different rows have the same distance, choose the row with the larger row number.
- When two different selections on one row have the same distance, choose the centroid on the left.
- If they can't find n consecutive seats, they will leave the cinema directly.

For example, in Figure 1, the centroid is $(8, 6)$, the middle back point is $(8, 6.5)$, so the distance is 0.5, which is the smallest in all selections.

Hint: don't use square root when comparing the distance, it may lose some accuracy.

Hint: this part is the most difficult part in the lab, don't forget what we've said when doing exams!

2.2 Read all Files in a Directory

The information about the halls in the cinema are stored in a directory `halls`, there are several files in it, but you don't know the file names. Fortunately, a header file `<dirent.h>` can provide an iteration through a directory. Note that is is not in the C standard, you can consider it as a library that provide some functions to you.

Read and run the code below and try to read all files in the `halls` directory.

```c
#include <stdio.h>
#include <stdlib.h>
#include <dirent.h>

int main() {
    DIR *dir = opendir(".");
    while (dir) {
        struct dirent *file = readdir(dir);
        if (file == NULL) {
            break;
        } else if (file->d_name[0] != '.') {
            printf("%s\n", file->d_name);
        }
    }
    closedir(dir);
}
```

The documentation: http://pubs.opengroup.org/onlinepubs/007908799/xsh/dirent.h.html

Think about the questions:

- How to set the working directory?

- What's `file->d_name` in the loop?
- How to use `strcat` or `strcpy` to concat strings?

Add some lines to the above code so that you can generate strings like `halls/[filename]` in the loop.

2.3 Input Files Format

There are at most 10 cinema halls, the format of the cinema halls definitions is:

- the hall name on the first line
- the row number m and column number n on the second line
- n boolean values on each of the next m lines, 1 represents that the seat exists and 0 represents that the seat doesn't exist
- if the column number is even, the center line is between column $m/2$ and $m/2+1$; otherwise, the center line is column $(m+1)/2$ (columns start from 1)

The information of the hall in Figure 1 is:

```
apple
8 12
1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1
```

You can define your hall as follows:

```c
typedef struct hall {
    char name[100];
    size_t m, n;
    int **data;
} hall_t;
```

2.3.1 FILE I/O in C

Read and run the code below.

```c
FILE *fd = fopen("vg101.txt", "w");
fprintf(fd, "%s is easy", "vg101");
fclose(fd);

fd = fopen("vg101.txt", "r");
char str[100];
fscanf(fd, "%s", str);
printf("%s\n", str);
fclose(fd);
```

Think about the questions:

- What is `FILE *fd`?
- Why the file should be closed? Try to remove `fclose` and find what happens.

2.3.2 Two-dimensional Dynamic Array

Read and run the code below.

```
size_t n;
scanf("%zu", &n);
int **a = malloc(n * sizeof(int *));
for (int i = 0; i < n; i++) {
    a[i] = malloc(n * sizeof(int));
}
// Do something
//
for (int i = 0; i < n; i++) {
    free(a[i]);
}
free(a);
```

Think about the questions:

- What are `size_t` and `%zu`?
- How to use the elements in `int **a`?
- Why should `a[i]` be freed?

Then write a function which takes a filename as a string and return a struct initialized with this file. Also write a function which can free the memory in that struct.

```
void init_hall(hall_t *hall, const char *filename) {
    // Do something
}

void free_hall(hall_t *hall) {
    // Do something
}
```

2.4 Input Format

The first line of the input is an integer n.
The next n lines contains a cinema hall name, a customer name and an integer m which is the number of a group of people.

If the hall doesn't exist, or they can't find seats there, they just leave the cinema; otherwise, select the seats for them and record in your system.

The input of the hall in Figure 1 is:

```
1
apple krystor 3
```

2.5 Output Files Format

For each cinema hall, you should output a file describing the seat status. First, print a graph about the seats and then print the customer names, row, column and number by the order in the input. The output file name should be the same as the cinema hall name.

The output file `apple` in Figure 1 is:

```
[ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
[ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
[ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
[ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
[ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
[ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
[ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
[ ] [ ] [ ] [ ] [x] [x] [x] [ ] [ ] [ ] [ ] [ ]
krystor 8 5 3
```

2.6 Submission Format

You only need to submit one file `l6.c`, write your code in the function `void l6()`.

A sample `l6.c` file

```c
#include "l6.h"

void l6() {
    // Do something
}
```

A sample `l6.h` file (no need to submit)

```c
#ifndef PROJECT_L6_H
#define PROJECT_L6_H

void l6();

#endif //PROJECT_L6_H
```

A sample `main.c` file (no need to submit)

```c
#include "l6.h"

int main() {
    l6();
    return 0;
}
```