Daily Planner Project, CS 135, Fall 2024

Tong Yi

In this project, we read a daily planner in **csv** (comma separated values) format, list all the tasks by its priority, and tally the number of hours for each priority category.

Warning:

1. This is copyrighted materials; you are not allowed to upload to the Internet.
2. Our project is more complicate than similar projects in the Internet and uses a different approach.
   (a) Ask help only from teaching staff of this course.
   (b) Use solutions from ChatGPT or online tutoring websites like, but not limited to, chegg.com violates academic integrity and is not allowed.

1 Time in 24-hour Notation and 12-hour Notation

A time of day is written in the 24-hour notation in the form hh:mm (for example 01:23), where hh (00 to 23) is the number of full hours that have passed since midnight, mm (00 to 59) is the number of full minutes that have passed since the last full hour. For more details, see wikipedia link.

The following figure is taken from https://www.britannica.com/topic/24-hour-clock.

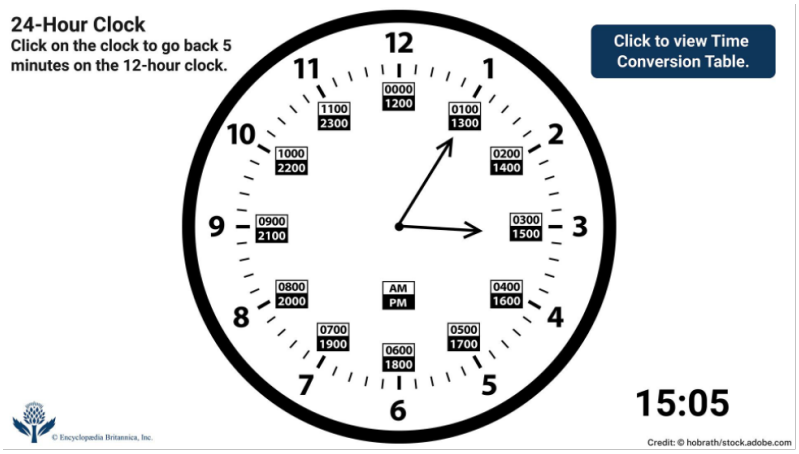

Figure 1: convertion between 24-hour notation and 12-hour notation

2 Task A: Convert Time from 24-hour Notation to 12-hour Notation

Relation between 24-hour notation and 12-hour notation is shown in the following table. Let mm be minutes.

24-hour notation	12-hour notation
00:mm	12:mm AM
01:mm	1:mm AM
02:mm	2:mm AM
03:mm	3:mm AM
04:mm	4:mm AM
05:mm	5:mm AM
06:mm	6:mm AM
07:mm	7:mm AM
08:mm	8:mm AM
09:mm	9:mm AM
10:mm	10:mm AM
11:mm	11:mm AM
12:mm	12:mm PM
13:mm	1:mm PM
14:mm	2:mm PM
15:mm	3:mm PM
16:mm	4:mm PM
17:mm	5:mm PM
18:mm	6:mm PM
19:mm	7:mm PM
20:mm	8:mm PM
21:mm	9:mm PM
22:mm	10:mm PM
23:mm	11:mm PM

From the above table, we notice the processing of 24-hour format can be categorized by four categories, depending on the value of hour. Each category is represented by different color.

In Task A, input time in 24-hour notation and print out the corresponding 12-hour notation. There are several steps.

1. Name your source code `convert_24_to_12.cpp`
2. Input time in 24-hour notation to a string variable.
3. Extract values of hour and minute from input.
   (a) Use `find` method of `string` class to find out the index of colon (:), a character separating hour and minute.
   (b) Use `substr` method of `string` class to get the substring representing hour and minute.
   (c) Use `stoi` function to convert a string to the corresponding integers. For example, `stoi("12")` returns 12.
4. If hour is not in [0, 23] (that is, hour is smaller than 0 or hour is larger than 23) or minute is not in [0, 59], then print "invalid input" and return -1. The word "invalid" cannot be missed.
5. Depending on the values of hour, print out the corresponding 12-hour notation.
6. Here are some sample input/output.
   (a) When hour or minute is not valid.

```
Enter time (hh:mm) in 24-hour notation (for example, 12:56): 5:61
invalid format
```

(b) When input format is 00:mm. The highlights are input and key part of output.

```
Enter time (hh:mm) in 24-hour notation (for example, 12:56): 00:21
24-hour notation 00:21 in 12-hour notation is 12:21 AM
```

(c) When input hour is in [1, 11]. The highlights are input and key part of output.

```
Enter time (hh:mm) in 24-hour notation (for example, 12:56): 11:05
24-hour notation 11:05 in 12-hour notation is 11:05 AM
```

(d) When input format is 12:mm. The highlights are input and key part of output.

```
Enter time (hh:mm) in 24-hour notation (for example, 12:56): 12:05
24-hour notation 12:05 in 12-hour notation is 12:05 PM
```

(e) When input hour is in [13, 23]. The highlights are input and key part of output.

```
Enter time (hh:mm) in 24-hour notation (for example, 12:56): 17:51
24-hour notation 17:51 in 12-hour notation is 5:51 PM
```

Here are references for `https://cplusplus.com/reference/string/string/find/` and `https://cplusplus.com/reference/string/string/substr/` methods of `string` class.

2.1 example of using find and substr methods of string class

Input a name in the format of LastName,FirstName. Extract first name and last name.

```cpp
#include <iostream>
#include <string>
using namespace std;

//purpose: enter a full name, extract last and first name.
//Sample input/output:
//Enter your full name (LastName,FirstName): Yi,Tong
//Your last name is Yi.
//Your first name is Tong.
int main() {
    cout << "Enter your full name (LastName,FirstName): ";
    string fullName;
    cin >> fullName;

    //find out the index of character ',' in fullName and save it in commaIndex.
    int commaIndex = fullName.find(',');

    //(1) The index of the first character of a string is 0.
    //(2) By Statement,
    //        int commaIndex = fullName.find(',');
    //    commaIndex saves the index of character ',' in string fullName.
```

```
    //(3) fullName.substr(0, commaIndex)
    //    extracts a substring from string fullName,
    //    starting from the first character -- whose index is 0 --
    //    all the way to the character just BEFORE the character at index commaIndex,
    //    which is character ','.
    //    Between index 0 and index commaIndex -1,
    //    there are a total of commaIndex -1 - 0 + 1 = commaIndex characters,
    //    that is why the value for the second
   parameter of substr method is commaIndex, as shown in
    //    fullName.substr(0, commaIndex).
    //
    //    So fullName.substr(0, commaIndex) returns the substring
    //    from the first letter of fullName all the way to character just BEFORE ','
    string lastName = fullName.substr(0, commaIndex);
    cout << "Your last name is " << lastName << "." << endl;

    //Note that the following version of substr has only one parameter.
    //fullName.substr(commaIndex+1) extracts a substring from fullName,
    //starting from the character at commaIndex+1,
    //which is the character right AFTER ',' character,
    //all the way to the last character in fullName.
    string firstName = fullName.substr(commaIndex+1);
    cout << "Your first name is " << firstName << "." << endl;
    return 0;
}
```

3 Task B: convert 12-hour time notation to 24-hour time notation

Input time in 12-hour notation and print out the corresponding 24-hour notation. There are several steps.

1. Name your source code `convert_12_to_24.cpp`
2. Input time in 12-hour notation to a string variable.
   (a) The format of 12-hour notation is `hh:mm AM` or `hh:mm PM`, where hh is in [1, 12] and mm is in [0, 59], the input may contain space.
   We cannot use `cin >> variableName;` to read the input, since `>>` stops at the first space or return key, whichever is encounted first.
   To read input with spaces, use `getline(cin, variableName);` read a line, and put the input line to `veriableName`.
   Operator `>>` read an int, a double number, a string, $\cdots$, to the corresponding variable.
   Function `getline` read a line and save that line to a string variable.
3. Extract the last two characters from input, it should be either AM or PM.
4. Extract values of hour and minute from input.
   (a) Use `find` method of `string` class to find out the index of colon (:), a character separating hour and minute.

(b) Use `substr` method of `string` class to get the substring representing hour and minute.

(c) Use `stoi` function to convert a string to the corresponding integers. For example, `stoi("12")` returns 12.

5. If hour is not in [1, 12] (that is, hour is smaller than 0 or hour is larger than 12) or minute is not in [0, 59], then print "invalid input" and return -1. The word "invalid" cannot be missed.

   (a) Optional (will not test in gradescope script): test whether input is ended with either "AM" or "PM".

6. Depending on the values of suffix "AM" or "PM" and hour, print out the corresponding 24-hour notation.

7. Here are some sample input/output.

   (a) When hour or minute is not valid.

```
Enter time (hh:mm) in 12-hour notation (for example, 12:56 AM): 5:61 AM
invalid format
```

   (b) When input format is 12:mm AM. The highlights are input and key part of output.

```
Enter time (hh:mm) in 12-hour notation (for example, 12:56): 12:21 AM
12-hour notation 12:21 AM in 24-hour notation is 00:21.
```

   (c) When input format is hh:mm AM, where hh is in [1, 11] and mm is in [0, 59]. The highlights are input and key part of output.

```
Enter time (hh:mm) in 12-hour notation (for example, 12:56): 01:36 AM
12-hour notation 01:36 AM in 24-hour notation is 01:36.
```

   (d) When input format is 12:mm PM, where mm is in [0, 59]. The highlights are input and key part of output.

```
Enter time (hh:mm) in 12-hour notation (for example, 12:56): 12:00 PM
12-hour notation 12:00 PM in 24-hour notation is 12:00.
```

   (e) When input hour is in hh:mm PM, where hh is in [1, 11] and mm is in [0, 59]. The highlights are input and key part of output.

```
Enter time (hh:mm) in 12-hour notation (for example, 12:56): 01:26 PM
12-hour notation 1:26 PM in 24-hour notation is 13:26.
```

4 Task C: calculate duration of two time 24-hour time notations

Enter start- and end- time in 24-hour notation. If start time is later than end time, print "invalid input" and return -1. Otherwise, calculate its duration in whole-number hours and minutes, where minutes is between 0 and 59.

For example, if start time is 12:45 and end time is 11:05, print "invalid input". If start time is 12:45 and end time is 15:30, the duration is 2 hours and 45 minutes.

1. Name the file `get_duration.cpp`.

2. Input start- and end time in 24-hour notation.

3. Extract hours and minutes from start- and end- time.

4. There are two ways to calculate duration between start- and end- time and display the result in whole-number hours and minutes, where minutes is an integer between 0 and 59.

   (a) Approach 1:

      i. Calculate the number of minutes elapsed from midnight 00:00 to end time.
      ii. Calculate the number of minutes elapsed from midnight 00:00 to start time.
      iii. Calculate duration between start- and end- time, in minutes.
      iv. Calculate whole hours and remainder minutes from the previous result using modular operator % and integer division /.

   (b) Approach 2:

      i. If minutes in end time is smaller than minutes in start time, subtract a hour from current hours of end time and add ?? (you find the correct value) minutes to the current minutes in end time.
      ii. Calculate the number of hours and number of minutes for the duration.
      iii. Similar example: to calculate 82 - 17, since ones digit (the rightmost digit) of the minuend (left operand), which is 2, is smaller than the ones digit of the subtrahend (right operand), which is 7, borrow 1 from tens digit from the minuend 82, then the tens digit is changed from 8 to 7.
         A. Perform subtraction for ones digit. In this example, 12 - 7 and get 5.
         B. Perform subtraction for tens digit. In this example, 7 - 1 and get 6.
         C. So the result of 82 - 17 is 65.

5 Task D: select tasks of a given priority and calculate total duration

In the next, we analyze a `csv` (Comma-Separated Values) file, which lists time interval, activity and its priority. Since data for time interval and activities may contain spaces, tsv (Tab-Separated Values) format is not appropriate. We use comma to separate items.

1. Name the file `analyze_12h_planner.cpp`.

2. In the file, besides `main` function, we define two more functions:

   `void hours_minutes_in_24h(string str, int& hours, int& minutes)` and

   `int duration(string str)`

   (a) In function `hours_minutes_in_24h`, given a string in 12-hour time notation, that is, the string is in the format `h:mm am` or `h:mm pm` or `hh:mm am` or `hh:mm pm`, find the number of hours and number of minutes in 24-hour time notation.

   Since there is at most one return in each C++ function, we use pass-by-reference approach to carry two or more results back to caller function – in our example – main function. That is, the original copy of an actual parameter of a function is passed to the corresponding formal parameter of the function. As a result, any change made on the formal parameter inside the callee function is carried back to the actual parameter in the caller function.

   For *pass by reference* approach, please read link in text book and link to slides.

   Hints to define function `hours_minutes_in_24h`, you can use most of codes in Project 1 Task B, pay attention to the following key parts.

   - No need to initialize parameter `str` inside function `hours_minutes_in_24h`, the parameter is initialized by caller function – say, function main – when `hours_minutes_in_24h` is called.

- No print out of intermediate results to the screen.
- For simplicity, **no** need to check whether `str` is valid or not.

(b) For function `int duration(string str)`, do the following:

- No need to initialize parameter `str` inside function `duration`.
- Use string operation to extract substrings representing start time and end time, connected by `'-'`.
- Call function `hours_minutes_in_24h` to find out hours and minutes of start time and end time in 24 hour time notation.
- Use approaches listed in Task C to calculate the duration, represented by minutes.
- Return the number of minutes calculated in the above step.
- For simplicity, **no** need to check whether `str` is valid or not.

(c) In main function, do the following.

- Enter a file name and priority.
- Read the file and list all the activities with that priority.
- Calculate and print duration of each of those activitivies.
- Calculate and print the total duration.
- For simplicity, assume priority as a string.

To read a csv file, we use getline(*fileObj*, *stringVariable*, ',') to read an item before ',' and put it its in *stringVariable*.

3. Structure of the source code is as follows.

```cpp
//file name: analyze_12h_planner.cpp
#include <iostream>
#include <fstream> //ifstream, ofstream
#include <string>
using namespace std;

//(1) function declaration with only function header, no function body.
//(2) main function needs to know the parameters and return type of
//    a function so that it can call the function.
//(3) function definitation will be put after the definition of main function
void hours_minutes_in_24h(string str, int& hours, int& minutes);

int duration(string str);

#ifndef UNIT_TEST
int main() {
    //TODO: enter file name

    //TODO: enter the priority to search for as a string

    //TODO: instantiate an ifstream object fin to open the file with given file name

    //TODO: skip the first line of the csv file since it is captions, no actual data

    //each row of data has time interval, activity name, and priority
    string time; //variable to save time interval
```

```
    string activity; //variable to save activity name
    string priority; //variable to save priority

    //Note that fin is the ifstream object to read the input file.
    while (getline(fin, time, ',')) {
        getline(fin, activity, ',');
        getline(fin, priority); //priority is the last data in a row, no ',' followed

        //TODO:  process the read data
    }

    //TODO: close file object fin.

    //TODO: print total duration of all activities in that priority

    return 0;
}
#endif

//Purpose: Let str be a string representing time in 12-hour notation.
//Convert 12-hour time notation str to 24-hour notation,
//where hours information is put in variable hours,
//minutes information is put in variable minutes.
//For example, if str is "12:30 am", then hours is 0 and minutes is 30.
//if str is "1:02 pm", then hours is 13 and minutes is 2.
//
//The & following type int in int& hours
//means formal parameter hours is pass by reference.
//Similar for & in int& minutes.

//function definition, with instructions provided in function body.
void hours_minutes_in_24h(string str, int& hours, int& minutes) {
    //TODO: write instructions in function body
}

//Let time be a string using dash symbol '-' to separate two time in 12 hours notation.
//An example of str is "11:30 am-2:15 pm", return the duration between those two time
   notations in minutes.
//Since the number of minutes between 11:30 am and 2:15 pm is 165 minutes,
//the return is 165.
int duration(string str) {
    //TODO: write instructions in function body
}
```

4. Test your codes in a local computer before submission to gradescope.

   (a) Define function `hours_minutes_in_24h` and `duration` in `analyze_12h_planner.cpp`. First unit test these functions work or not.

i. Put `unit_test_analyze_12h_planner.cpp` (provided as follows) and `analyze_12h_planner.cpp` (your job) in the same folder.

```cpp
//file: unit_test_analyze_12h_planner.cpp
#include <iostream>
#include <string>
#include "analyze_12h_planner.cpp" //include the source code of file to be tested

//https://stackoverflow.com/questions/29734231/how-to-conditional-compile-main-in-c
//1. We have two files, analyze_12h_planner.cpp and unit_test_analyze_12h_planner.cpp.
//2. File analyze_12h_planner.cpp reads a csv file and prints the tasks by priority.
//3. Both files have main functions.
//4. Enclose the main function of unit_test_analyze_12h_planner.cpp with #ifdef UNIT_TEST and #endif
//#ifdef UNIT_TEST
//int main(int argc, char** argv) {
//...
//}
//#endif
//5. Enclose the main function of analyze_12h_planner.cpp with #ifnedef UNIT_TEST and #endif
//which unit tests functions //defined in analyze_12h_planner.cpp,
//6. When running the code, option -DUNIT_TEST means to define UNIT_TEST
//       g++ -DUNIT_TEST unit_test_analyze_12h_planner.cpp
//   choose main function of unit_test_analyze_12h_planner.cpp
//
//7. When compiling and linking without -D option,
//     g++ analyze_12h_planner.cpp
//   choose main function of analyze_12h_planner.cpp

//https://stackoverflow.com/questions/17868302/is-it-possible-to-run-tests-on-a-cpp-file-which-already-has-a-main-function
#ifdef UNIT_TEST
int main(int argc, char** argv) {
    if (argc < 2) {
       std::cout << "missing argument in main function" << std::endl;
       exit(0);
    }

    switch (*argv[1]) {
        case '1': {
             int hours = 0;
             int minutes = 0;
             std::string strs[] = {"12:56 am", "11:35 am", "12:30 pm", "3:07 pm"};
             int expected_hours[] = {0, 11, 12, 15};
```

```
            int expected_minutes[] = {56, 35, 30, 7};

            int size = sizeof(strs) / sizeof(strs[0]);

            for (int i = 0; i < size; i++) {
                hours_minutes_in_24h(strs[i], hours, minutes);

                std::cout << "test for " << strs[i] << std::endl;

                if (hours != expected_hours[i] || minutes != expected_minutes[i])
                   std::cout << "fail test" << std::endl
                         << "expected hours: " << expected_hours[i]
                         << " and expected minutes: " << expected_minutes[i]
                         << std::endl
                         << "your code generate: hours: " << hours
                         << " and minutes: " << minutes << endl;
                 else std::cout << "pass test. "
                                << "expected hours: " << expected_hours[i]
                                << " and expected minutes: " << expected_minutes
[i] << std::endl;
          }

          break;
      }

      //for simplicity, no test for invalid input in this example
      //test duration
      case '2': {
          std::string strs[] = {"1:20 am-3:15 am", "11:37 am-2:25 pm",
              "2:15 pm-4:30 pm"};
          int expected_total_minutes[] = {115, 168, 135};
          int size = sizeof(strs) / sizeof(strs[0]);

          for (int i = 0; i < size; i++) {
              if (duration(strs[i]) == expected_total_minutes[i])
                 std::cout << "pass test to calculate duration of " << strs[i]
                           << ". expected return: " << expected_total_minutes[i
] << std::endl;
              else std::cout << "fail test to calculate duration of " << strs[i]
                             << ". expected return: " << expected_total_minutes
[i]
                             << " and your return: " << duration(strs[i])
                             << std::endl;
          }

          break;
      }
  }

```

```
    return 0;
}
#endif
```

ii. You may notice that #ifndef UNIT_TEST and #endif enclose the definition of main function in `analyze_12h_planner.cpp`, while #ifdef UNIT_TEST and #endif enclose the definition of main function in `unit_test_analyze_12h_planner.cpp`.
Recall that main function is the entry point of a C++ project. So any project can have only one main function. To handle this, we use a trick outlined in conditional compilation link. If we define UNIT_TEST, then we choose main function from `unit_test_analyze_12h_planner.cpp`, otherwise, we choose main function from `analyze_12h_planner.cpp`.

iii. Run the following commands, where `-DUNIT_TEST` define `UNIT_TEST`.

```
g++ -DUNIT_TEST unit_test_analyze_12h_planner.cpp
./a.out 1
```

You should see the following output if your defintion of functions is correct.

```
test for 12:56 am
pass test. expected hours: 0 and expected minutes: 56
test for 11:35 am
pass test. expected hours: 11 and expected minutes: 35
test for 12:30 pm
pass test. expected hours: 12 and expected minutes: 30
test for 3:07 pm
pass test. expected hours: 15 and expected minutes: 7
```

Then your definition of `hours_minutes_in_24h` should be fine.

iv. Test `duration` function.

```
g++ -DUNIT_TEST unit_test_analyze_12h_planner.cpp
./a.out 2
```

If you see the following output, your definition of `duration` should be fine.

```
pass test to calculate duration of 1:20 am-3:15 am. expected return: 115
pass test to calculate duration of 11:37 am-2:25 pm. expected return: 168
pass test to calculate duration of 2:15 pm-4:30 pm. expected return: 135
```

(b) After correctly defining functions `hours_minutes_in_24h` and `duration`, finish testing main function in file `analyze_12h_planner.cpp`.

Add data file `plan.csv` and put in the same folder as `analyze_12h_planner.cpp`.

Compile the file as follows. Note that we choose `analyze_12h_planner.cpp`. So `UNIT_TEST` is not defined.

```
g++ analyze_12h_planner.cpp
./a.out
```

If your definition of main function is correct, you should see the following output.

Suppose we have `plan.csv` with the following contents.

```
time,activity,priority
```

```
9:10 am-11:00 am,CS 135 recitation,1
11:30 am-12:15 pm,lunch,2
1:00 pm-2:15 pm,CS 135,1
3:30 pm-5:15 pm,review,1
7:45 pm-9:30 pm,exercise,2
```

After running the program, we should get the following output. The highlighted part is user input.

```
Enter a file name: plan.csv
Enter priority: 1
9:10 am-11:00 am,CS 135 recitation,1 hours and 50 minutes
1:00 pm-2:15 pm,CS 135,1 hours and 15 minutes
3:30 pm-5:15 pm,review,1 hours and 45 minutes

total duration: 4 hours and 50 minutes
```

Here is another run of the program.

```
Enter a file name: plan.csv
Enter priority: 2

11:30 am-12:15 pm,lunch,0 hours and 45 minutes
7:45 pm-9:30 pm,exercise,1 hours and 45 minutes

total duration: 2 hours and 30 minutes
```

(c) If the code runs fine, upload `analyze_12h_planner.cpp` to gradescope.
Good job!