```
=======================================================================
            /local/submit/submit/comp20005/ass1/gmahmood/src/myass1.c
=======================================================================

/*  COMP20005, Assignment 1.
    Gazi Mufti Mahmood. Student ID - 884041
    Moto - Programming is Fun!
    April, 2018.
*/

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


#define MAX_ARRAY_SIZE 10000
#define JAN 1
#define MAR 3
#define MAY 5
#define JUL 7
#define AUG 8
#define OCT 10
#define DEC 12
#define DAYS_YEAR_NO_FEB 337    /* Days in a yaer without february */
#define NGPRS 3
#define WEEK 7                  /* Days in a week */

struct {
    int data[NGPRS];
    int days[WEEK];
} records;


/* function prototypes */
int readfile(int yyyy[], int mm[], int dd[], int day[], int daycount[]);
int mygetchar();
void S1_print(int yyyy[], int mm[], int dd[], int daycount[],
    int line, int key);
void avg_ped_month(int yyyy[], int mm[], int daycount[], int key);
int days_between_dates(int sdd, int smm, int syyyy, int dd, int mm, int yyyy);
int days_year(int dd, int mm, int yyyy);
void trend(int yyyy[], int mm[], int dd[], int day[], int daycount[], int key);
void print_groups(int groups[], int yyyy[], int mm[], int dd[], int grp_key);
void bar_chart(int day[], int groups[], int daycount[], int grp_key);
int count_if(int day[], int daycount[], int d, int range, int j);
void dotter(int num);
int roundoff(double num);

int
main(int argc, char *argv[]) {
    /* Making arrays, their key and the total datalines read. */
    int yyyy[MAX_ARRAY_SIZE], mm[MAX_ARRAY_SIZE], dd[MAX_ARRAY_SIZE];
    int day[MAX_ARRAY_SIZE], daycount[MAX_ARRAY_SIZE];
    int line, key;

    /* Stage 1 */
    line = readfile(&yyyy[0], &mm[0], &dd[0], &day[0], &daycount[0]);

    /* last key of the array is one less than the line */
    key = line - 1;
    S1_print(&yyyy[0], &mm[0], &dd[0], &daycount[0], line, key);

    /* Stage 2 */
    int range;
    double coverage;
    range = days_between_dates(dd[0], mm[0], yyyy[0], dd[key], mm[key],
        yyyy[key]);
    coverage = ((1.0*line)/range)*100;
    printf("S2: range spanned   = %d days\n", range);
    printf("S2: coverage ratio  = %.1f%%\n", coverage);
    printf("\n");

    /* Stage 3 */
    avg_ped_month(&yyyy[0], &mm[0], &daycount[0], key);
```

```c
    printf("\n");

    /* Stage 4 */
    trend(&yyyy[0], &mm[0], &dd[0], &day[0], &daycount[0], key);

    return 0;
}

/* Reads through the file and stores the necessary data into arrays */
int
readfile(int yyyy[], int mm[], int dd[], int day[], int daycount[]){
    char cc;
    int line, a, b, c, d, e;
    line = 0;
    /* Skipping the first two lines */
    while ((cc = mygetchar()) != EOF){
        if (cc == '\n'){
            line += 1;
        }
        if (line > 1){
            line = 0;
            break;
        }
    }
    /* Adding the values to respective arrays */
    while (scanf("%d%d%d%d%d", &a, &b, &c, &d, &e) != EOF){
        yyyy[line] = a;
        mm[line] = b;
        dd[line] = c;
        day[line] = d;
        daycount[line] = e;
        line += 1;
    }

    return line;
}

/* pogram made by Alister Moffat */
int
mygetchar(){
    int c;
    while ((c=getchar())=='\r') {
    }
    return c;
}

/* Prints necessary data for Stage 1 */
void
S1_print(int yyyy[], int mm[], int dd[], int daycount[], int line, int key){
    printf("S1: total data lines = %d\n", line);
    printf("S1: first data line  = %02d/%02d/%04d, %d people counted\n",
        dd[0], mm[0], yyyy[0], daycount[0]);
    printf("S1: last data line   = %02d/%02d/%04d, %d people counted\n",
        dd[key], mm[key], yyyy[key], daycount[key]);
    printf("\n");
}


/* Takes in start date and end date and returns the total days in between */
int
days_between_dates(int sdd, int smm, int syyyy, int dd, int mm, int yyyy){
    int feb, num_days, snum_days, y_days, year;
    num_days = days_year(dd, mm, yyyy);
    snum_days = days_year(sdd, smm, syyyy);
    feb = 28 + (yyyy%4 == 0 && (yyyy%100 != 0 || yyyy%400 == 0));
    if (yyyy != syyyy){
        snum_days = (DAYS_YEAR_NO_FEB + feb) - snum_days;
    } else {
        /* Same year. Adding 1 to include the end date */
        return num_days - snum_days + 1;
    }
    y_days = 0;
    for (year = syyyy + 1; year < yyyy; year++){
        // number of days + 1 (cheaking for leap year)
        y_days += 365 + (year%4 == 0 && (year%100 != 0 || year%400 == 0));
    }
    /* Adding all the days together and adding 1 to include the end date */
    return num_days + snum_days + y_days + 1;
}

/* Calculates the days from the beginnging of a year to the date given */
int
days_year(int dd, int mm, int yyyy){
    int feb, month, num_days;
    num_days = dd;
    feb = 28 + (yyyy%4 == 0 && (yyyy%100 != 0 || yyyy%400 == 0));
    // Cheaking to see if the ending year is a leap year
    for (month = 1; month < mm; month++){
        if (month == 2){
            num_days += feb;
        } else if( month == JAN || month == MAR || month == MAY || month == JUL
                    || month == AUG || month == OCT || month == DEC){
            num_days += 31;
        } else {
            num_days += 30;
        }
    }
    return num_days;
}


/* Caculates the Average Pedestrian observed per month */
void
avg_ped_month(int yyyy[], int mm[], int daycount[], int key){
    int i, j, days, tdays, month, cmonth, count, year;
    double average;
    /* Calculating total months */
    month = (12 - mm[0]) + mm[key] + 12*(yyyy[key] - yyyy[0] - 1);
    j = 0;
    /*                       */
    for (i = mm[0] - 1; i < month + mm[0]; i++){
        cmonth = (i%12) + 1;
        days = count = 0;
        year = yyyy[j];
        /* Checking to see the total days in the current months */
        if (cmonth == 1 || cmonth == 3 || cmonth == 5 || cmonth == 7
                    || cmonth == 8 || cmonth == 10 || cmonth == 12){
            tdays = 31;
                    } else if (cmonth == 2){
            tdays = 28 + (year%4 == 0 && (year%100 != 0 || year%400 == 0));
                    } else {
            tdays = 30;
                    }
        /* Counting the days and count per month. **j initialized before** */
        for (; j <= key; j++){
            if (cmonth != mm[j] || year != yyyy[j]){
                break;
            }
            days += 1;
            count += daycount[j];
        }
        /* Skipping months that were not accounted for */
        if (days == 0){
            continue;
        }
        average = ((1.0*count)/days)/1000;
        printf("S3: %02d/%4d %02d/%2d days covered, average count =             %02.1fk\n",
            cmonth, year, days, tdays, average);
    }
}

/* Groups data according to NGPRS and plots a bar chart */
void
trend(int yyyy[], int mm[], int dd[], int day[], int daycount[], int key){
    int i, j, remainder, groups[NGPRS], grp_key, temp_grp_key;

    /* 1 is added to key because the 1st key value for an array is 0 */
    remainder = (key + 1) % NGPRS;
    /* Storing the number of data into equal sized groups */
    for (i = 0; i < NGPRS; i++){
        groups[i] = ((key + 1 - remainder)/NGPRS);
    }
    grp_key = temp_grp_key = i - 1;

    /* Adding the remainder to the gourps by 1 */
    for (j = remainder; j > 0; j--){
        groups[temp_grp_key] += 1;
        temp_grp_key--;
    }
    print_groups(&groups[0], &yyyy[0], &mm[0], &dd[0], grp_key);
    printf("\n");
    bar_chart(&day[0], &groups[0], &daycount[0], grp_key);
}

/* Prints the groups for stage 4 */
void
print_groups(int groups[], int yyyy[], int mm[], int dd[], int grp_key){
    int i, j;
    j = groups[0] - 1;

    /* Printing group 0 */
    printf("S4: group %2d data, %02d/%02d/%4d to %02d/%02d/%4d ", 0, dd[0],
        mm[0], yyyy[0], dd[j], mm[j], yyyy[j]);
    printf("%d data records\n", groups[0]);

    /* Storing the data record */
    records.data[0] = groups[0];
    /* Printing the rest of the groups */
    for (i = 1; i <= grp_key; i++){
        j += 1;
        printf("S4: group %2d data, %02d/%02d/%4d ", i, dd[j], mm[j], yyyy[j]);
        j += groups[i] - 1;
        printf("to %02d/%02d/%4d %d data records\n", dd[j], mm[j], yyyy[j],
            groups[i]);
    /* Storing the data record */
        records.data[i] = groups[i];
    }
}

/* Plots a bar chart for NGPRS, categorized by days of the week*/
void
bar_chart(int day[], int groups[], int daycount[], int grp_key){
    int i, j, k, l, a, b;
    double average;

    /* Making an array to store the short form of the days of the week  */
    char *week[7];
    week[0] = "Sun";
    week[1] = "Mon";
    week[2] = "Tue";
    week[3] = "Wed";
    week[4] = "Thu";
    week[5] = "Fri";
    week[6] = "Sat";

    /* Replacing the data of records with respective aggregate data*/
    for (k = 1; k < NGPRS; k++){
        records.data[k] += records.data[k - 1];
    }
    for (i = 0; i < WEEK; i++){
        l = 0;
        for (j = 0; j <= grp_key; j++){
            a = count_if(&day[0], &daycount[0], i + 1, l, j);
            b = records.days[i];
            l += records.data[0];
            average = ((1.0*a)/b)/1000;
            printf("S4: %s, g%d = %02.1fk |", week[i], j, average);
            dotter(roundoff(average));
            printf("\n");
        }
        printf("\n");
```

```c
        }
    }


    /* Counts the number of days for a given day of the week*/
    int
    count_if(int day[], int daycount[], int d, int range, int j){
        int i, counter, sum_count, a;
        sum_count = counter = 0;
        a = records.data[j];
        for (i = range; i < a; i++){
            if (day[i] == d){
                sum_count+= daycount[i];
                counter += 1;
            }
        }
        records.days[d − 1] = counter;
        return sum_count;
    }

    /* Prints number of ’*’ entered */
    void dotter(int num){
        int i;
        for (i = 0; i < num; i++){
            printf("*");
        }
    }

    /* Rounds up if number has decimal point greatar than 0.5, else rounds down */
    int roundoff(double num){
        int a, b;
        a = (int) num;
        b = (int) (num + 0.444444);
        if (a == b){
            return a;
        } else {
            return b;
        }
    }
```