```c
/******************************************************************************
 *   ghostBuster.h
 *
 *      This header file declares functions that related to program Ghost Buster
 *      database. And also define number, file name and event structure.
 *
 *    Created by Nathaphop Sundarabhogin(KLA) ID : 3420
 *        29 NOVEMBER 2017
 *
 ******************************************************************************
 */

#define LENGTH 100
#define SHORTLEN 30
#define YEARSCOPE 10
#define DATAFILE "dataEvent.dat"
#define DUMPFILE "dataEvent.txt"

/* Declare struct of date. */
typedef struct
    {
    int day;    /* Day of the date from 1-31. */
    int month;  /* Month of the date from 1-12. */
    int year;   /* Year in western year. */
    int hour;   /* Hour in 24 hour format from 0-23. */
    int minute; /* Minute from 0-59. */
    } DATE_T;

/* Declare struct of event data. */
typedef struct
    {
    DATE_T dateEvent;           /* Date and time of event happen. */
    DATE_T dateInvest;          /* Date and time that investigation. */
    char phoneReport[SHORTLEN]; /* Phone number person reporting event. */
    char typeEvent[SHORTLEN];   /* Type of event. */
    char nameReport[LENGTH];    /* Name of person reporting event. */
    char nameInvest[LENGTH];    /* Name of person investigating. */
    float latitude;             /* Latitude of event location in form nn.nnnn. */
    float longitude;            /* Longitude of event location in form nn.nnnn
                                   or nnn.nnnn. */
    int eventCode[2];           /* Event code, keep string in form yyyy-nnnn.
                                 * [0] - yyyy, [1] - nnnn. */
    int result;                 /* Result of event
                                 * 0 - Data Deleted      1 - Success
                                 * 2 - Failure           3 - Unknow */
    } EVENT_T;

/* Define a set of possible error return, that can occur in all validate */
typedef enum
    {
    CORRECT,     /* For correct case. */
    ERR_LEN,     /* Have wrong length. */
    ERR_CHAR,    /* Have wrong character. */
    /* The Other are bad format specific validation that different in each case. */
    ERR_FORM1,
    ERR_FORM2,
    ERR_FORM3,
    ERR_FORM4
    } CHECK_STATUS;
```

```c
/******************** Declare function for ghostBuster.c ********************/

/* PUBLIC FUNCTION. This function gets menu from user
 * and then call function that the user wants to do.
 */
void getMenu();

/* PUBLIC FUNCTION. This function is called when user wants to add information.
 * Get information from the user. During add information,
 * If the user hits to return, program will go to main function to get a menu.
 */
void controlAdd();

/* PUBLIC FUNCTION. This is main search function. It is called when user wants
 * to use search function. Ask the user how to search until the user hits to
 * return to main function to main function.
 */
void controlSearch();

/* PUBLIC FUNCTION. This function is called when user search by event code.
 * Get event code from the user, print that information if it has.
 * and then ask the user to modify or delete information.
 */
void controlSearchCode();

/* PUBLIC FUNCTION.
 * This function is called when user search by others such as
 * event year, event type and result.
 * Ask the user to get event year, type of event or result.
 * Then get event code from the user, print that information if it has.
 * and then ask the to modify or delete that information.
 */
void controlSearchOthers();

/* PUBLIC FUNCTION. This function is called when users want to dump file. */
void controlDump();

/* PUBLIC FUNCTION. This function is called when user get a choice
 * that exit the program. Ask the user again to exit the program.
 *     If the user gets Y, the program will exit.
 *     If the user gets N, the program still works.
 */
void controlExit();
```

```c
/******************* Declare function for validateEvent.c *******************/

/* PUBLIC FUNCTION. Get the date to day and current time.
 * ARGUMENT:
 *     pDate - Address of struct date that want to get current time.
 */
void dateToday(DATE_T *pDate);

/* PUBLIC FUNCTION. Check string of date it is correct or not. If correct,
 * return correct. If not return error in each case.
 * ARGUMENT:
 *     dateStr - String of date that want to check.
 */
CHECK_STATUS checkDateStr(char dateStr[]);

/* PUBLIC FUNCION. Check that date is not in the future. If 'pDateEvent' is NULL,
 * check that date is within past 10 years or not. But if 'pDateEvent' is not
 * NULL, check that date is ealier than event date or not. After validate, If it's
 * correct, return correct. But if not, return error in each case.
 * ARGUMENTS:
 *     date       - Struct of date that want to check.
 *     pDateEvent - Struct of event date if want to check earlier then event
 *                  date or not. If it's NULL, it will check within 10 years instead.
 */
CHECK_STATUS checkDate(DATE_T date, DATE_T *pDateEvent);

/* PUBLIC FUNCTION. Validate string of name. If name is valid, return correct.
 * But if invalid, return error in each case.
 * ARGUMENT:
 *     nameStr[] - String of name that want to validate.
 */
CHECK_STATUS checkName(char nameStr[]);

/* PUBLIC FUNCTION. Validate string of phone. If phone is valid, return correct.
 * But if invalid, return error in each case.
 * ARGUMENT:
 *     phoneStr[] - String of phone that want to check.
 */
CHECK_STATUS checkPhone(char phoneStr[]);

/* PUBLIC FUNCTION. Validate string of type. If type is valid, return correct.
 * But if invalid, return error in each case.
 * ARGUMENT:
 *     typeStr[] - String of type that want to check.
 */
CHECK_STATUS checkType(char typeStr[]);

/* PUBLIC FUNCTION. Get string of latitude or longitude and validate string
 * correct or not. If string of latitude/longitude is correct return correct,
 * But if not return error in each case.
 * ARGUMENTS:
 *     locationStr[] - String that want to check in latitude/longitude form.
 *     select      - Select to check latitude or longitude.
 *                    (1=latitude, 2=longitude)
 */
CHECK_STATUS checkStrLocation(char locationStr[], int select);
```

```
/* PUBLIC FUNCTION. Check latitude/longitude is correct or not, which know by
 * 'select'. Select equal 1, check latitude. Select equal 2, check longitude.
 * If correct, return correct. But if not, return error in each case.
 * ARGUMENTS:
 *     location - Latitude/longitude that want to check.
 *     select   - Select to check latitude or longitude.
 *                (1=latitude, 2=longitude)
 */
CHECK_STATUS checkLocation(float location,int select);

/* PUBLIC FUNCTION. Check event code correct or not. If event code is correct,
 * return 1. But if not, return 0.
 * ARGUMENT:
 *     codeStr[] - Keep event code that want to check.
 */
CHECK_STATUS checkEventCodeStr(char codeStr[]);

/* PUBLIC FUNCTION. Check all data in event (except event code) correct or not.
 * If all of data in event is correct, return 1. But if not, return 0.
 * ARGUMENT:
 *     event - Struct of event that want to check data.
 */
int checkEvent(EVENT_T *event);

/* PUBLIC FUNCTION. Check event code it's correct or not. If correct, return 1
 * If not correct, return 0.
 * ARGUMENT:
 *     code[] - Event code that want to check.
*/
CHECK_STATUS checkEventCode(int code[]);
```

```c
/******************* Declare function for askEvent.c *******************/

/* PUBLIC FUNCTION. Print event detail to the terminal line.
 * ARGUMENT:
 *     event - Struct event that want to print
 */
void printEvent(EVENT_T event);

/* PUBLIC FUNCTION. Ask user to input "date of event" or "date of investigate" and
 * validate it. If user input <CR>, return '0' back (That's mean user cancel
 * to input data). But if date is valid, return date and '1' back.
 * ARGUMENTS:
 *     pDate      - Address of date structure that want to return it back.
 *     pDateEvent - Get address date of event. If it's NULL, means ask date Event.
 */
int askDate(DATE_T *pDate, DATE_T *pDateEvent);

/* PUBLIC FUNCTION. Ask user to input name of report or investigate. Then validate
 * input. If user inputs <CR>, return '0' back (That's mean user cancel to input
 * data). But if name is valid, return name and '1' back.
 * ARGUMENTS:
 *     pOutput - Address of string that want to return it back.
 *     select  - To know what user wants to ask.
 *                   If select is '1', will ask name report.
 *                   If select is '2', will ask name investigate.
 */
int askName(char *pOutput, int select);

/* PUBLIC FUNCTION. Ask user to input phone number and validate it. If user input
 * <CR>, return '0' back (That's mean user cancel to input data).
 * But if number phone is valid, return phone number and '1' back.
 * ARGUMENT:
 *     pOutput - Address of string that want to return it back.
 */
int askPhone(char *pOutput);

/* PUBLIC FUNCTION. Ask user to input type and validate it. If the user hits <CR>,
 * return '0' back (That's mean user cancel to input data). But if type is valid,
 * return type of event and '1' back.
 * ARGUMENT:
 *     pOutput - Address of string that want to return it back.
 */
int askType(char *pOutput);

/* PUBLIC FUNCTION. Ask user to input latitude or longitude and validate it.
 * If the user hits <CR>, return '0' back (That's mean user cancel to input data).
 * But if latitude or longitude is valid, return it and '1' back.
 * ARGUMENTS:
 *     pOutput - Address of string that want to return it back.
 *     select  - To know what user wants to ask
 *                   If select is '1', ask latitude.
 *                   If select is '2', ask longitude.
 */
int askLocation(float *pOutput,int select);

/* PUBLIC FUNCTION. Ask user to input event code and validate it.
 * If the user hits <CR>, return 0 back.
 * But if event code is valid, return event code and '1' back.
 * ARGUMENT:
 *     pOutput[] - Event code that user wants to get
 */
int askEventCode(int pOutput[]);
```

```
/* PUBLIC FUNCTION. Ask user to input event year and validate it.
 * If the user hits <CR>, return 0 back. But if event year is valid,
 * return event year and '1' back.
 * ARGUMENT:
 *     pEventYear - Event year that user wants to get.
 */
int askEventYear(int *pEventYear);

/* PUBLIC FUNCTION. Ask user to input result and validate it.
 * If the user hits <CR>, return 0 back. But if result is valid,
 * return result and '1' back.
 * ARGUMENT:
 *     pResult   - Result that user wants to get.
 */
int askResult(int * pResult);
```

```
/******************* Declare function for database.c *******************/

/* PUBLIC FUNCTION. This function will read amount of data and allocate dynamic
 * memory with that number. Then read all of event data in database and keep it
 * in dynamic memory. After that validate all of data, Then send memory of
 * event data and amount of event back.
 * ARGUMENT:
 *     eventAmount - Address of event amount.
 */
EVENT_T *readData(int *eventAmount);

/* PUBLIC FUNCTION. Get event data and amount of data then write down all of event
 * data to database.
 * ARGUMENTS:
 *     pEvent      - Dynamic memory all of event data.
 *     eventAmount - Amount of event data.
 */
void saveData(EVENT_T * pEvent, int eventAmount);

/* PUBLIC FUNCTION. Open text file and write all of event data into text file.
 * ARGUMENTS:
 *     pEvent      - Dynamic memory all of event data.
 *     eventAmount - Amount of event data.
 */
void dumpFile(EVENT_T * pEvent, int eventAmount);
```

```c
/******************* Declare function for manageData.c *******************/

/* PUBLIC FUNCTION. Function that manage event data to database. Get the command
 * to decide what to do to database.
 * ARGUMENT:
 *     command - Get number to know that what to do to database.
 *         1 - Start program. Read all data in database.
 *         2 - Save Data. Save all data to database.
 *         3 - Dump File. Dump text file output.
 *         4 - Save data and free data. Used when close program.
 */
void controlDatabase(int command);

/* PUBLIC FUNCTION. Reallocate new one dynamic memory from last data, copy
 * new event into new dynamic memory and sorting it.
 * ARGUMENT:
 *     event - Event that want to add to data.
 */
void addEvent(EVENT_T event);

/* PUBLIC FUNCION. Print data of each event.
 * ARGUMENTS:
 *     pInt   - Position in data that want to print event data.
 *     amount - The amount of position.
 */
void printEachEvent(int *pInt, int amount);

/* PUBLIC FUNCTION. Loop print all of event data to terminal line */
void printAllEvent();

/* PUBLIC FUNCTION. Loop find the lastest code in year that want to know. If found
 * the code, return that count plus 1 (New code for that year). But if not found
 * or there aren't any data, return 1.
 * ARGUMENT:
 *     eventYear - Year that want to run event code.
 */
int runEventCode(int eventYear);

/* PUBLIC FUNCTION. Search event code in data. If there is event code in data,
 * print information and return '1' and position back.
 * If it does not have, print message and return '0' back.
 * ARGUMENTS:
 *     code[]     - Keep event code that want to search.
 *     pPosition  - Keep position of data.
 */
int searchEventCode(int code[], int *pPosition);

/* PUBLIC FUNCTION. Search data by using event year
 * and then return all of the position of data.
 * ARGUMENTS:
 *     eventYear  - Event year that want to search.
 *     pCountData - The amount of position.
 */
int * searchEventYear(int eventYear, int * pCountData);

/* PUBLIC FUNCTION. Search data by using result
 * and then return all of the position of data.
 * ARGUMENTS:
 *     result     - Result that want to search.
 *     pPosition  - Last position of data in database.
 *     pCountData - The amount of data.
 */
int * searchResult(int result, int * pPosition, int * pCountData);
```

```c
/* PUBLIC FUNCTION. Search data by using type of event
 * and then return all of the position of information
 * ARGUMENTS:
 *     type[]     - Type of event that want to search.
 *     pPosition  - Last position of data in database.
 *     pCountData - The amount of data.
 */
int * searchEventType(char type[], int * pPosition, int *pCountData);

/* PUBLIC FUNCTION. Get position of event data and ask user to modify.
 * If the user hits <CR> or get input, program will ask next information until finished.
 * In the end, program asks users to save data.
 * ARGUMENT:
 *     position - Position of data in event data.
 */
void modifyEvent(int position);

/* PUBLIC FUNCTION. Get position of event data and ask user
 * for sure to delete event code.
 * ARGUMENT:
 *     position - Position of data in database.
 */
void deleteEvent(int position);
```