Programming and Data Structures I
CS3101

Mid-Term Assignment
on
Flight Reservation System
(Documentation and User Guide)

by

Subham Kumar Singh (17MS020)
Anmol Kumar (17MS028)
Naksatra Kumar Bailung (17MS032)
Harshit Kumar (17MS037)
Souvik Paul (17MS070)

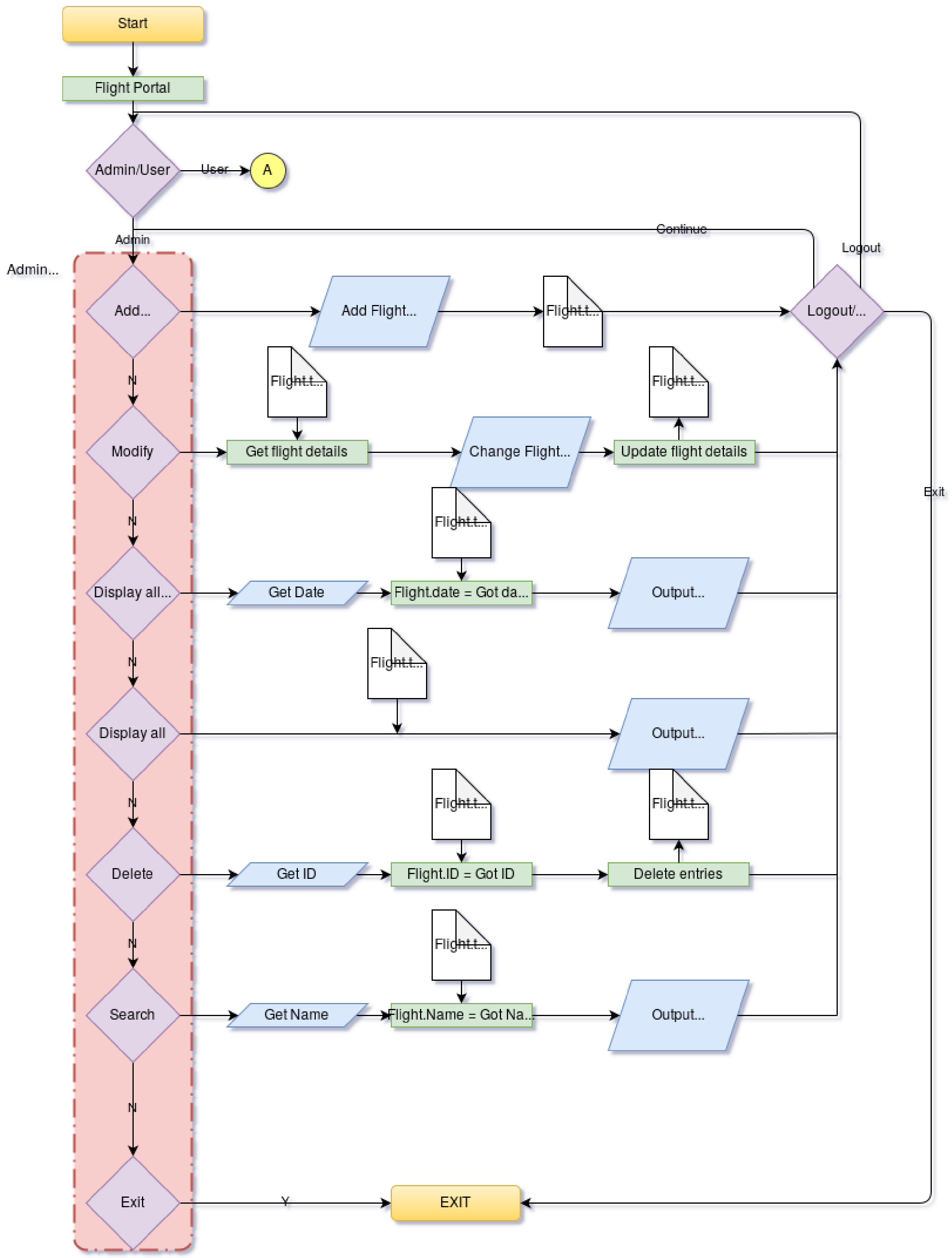
(Group No. 2)

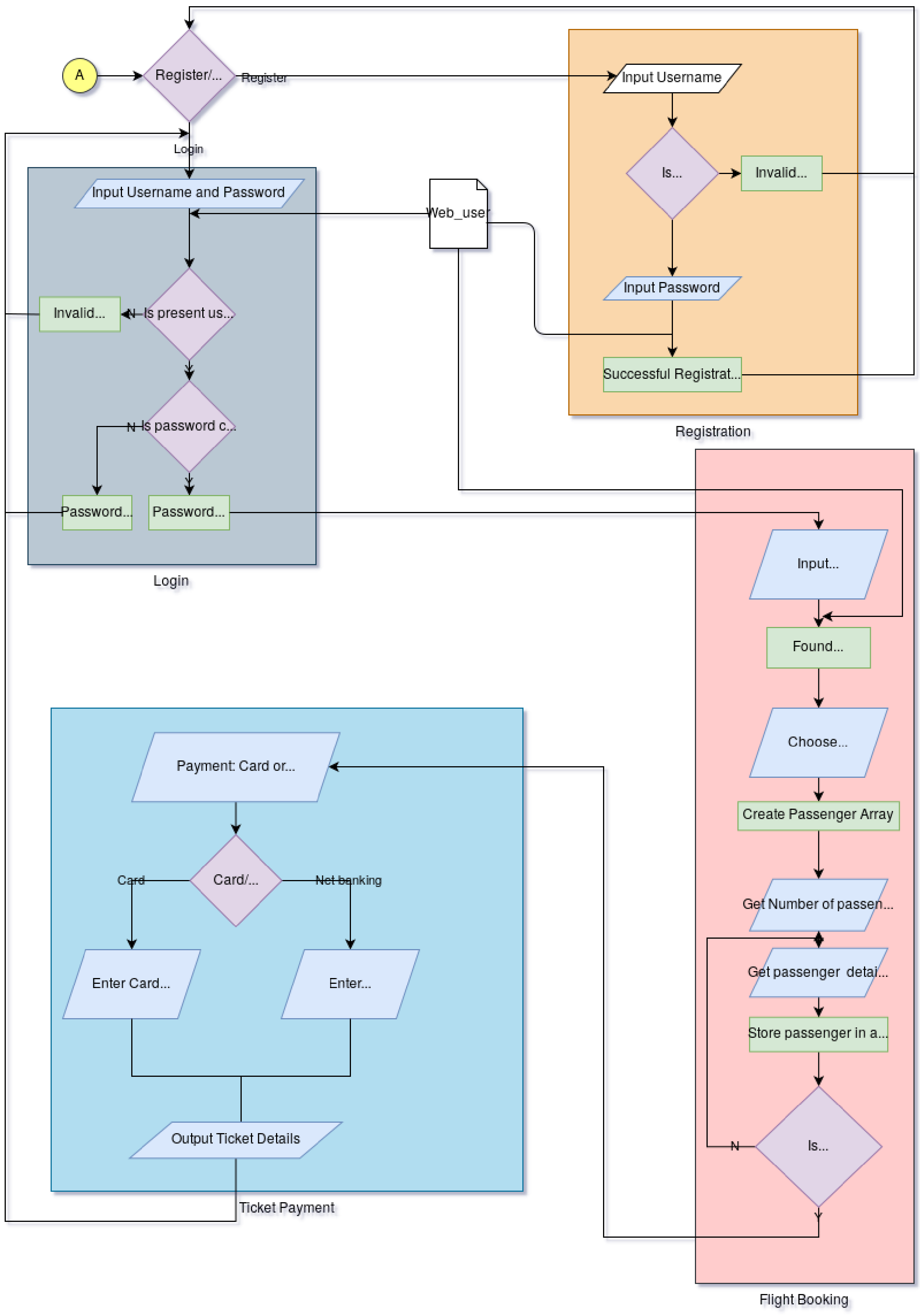
Course Instructor
Dr. Kripabandhu Ghosh
Department of Computer Science and Application

Indian Institute of Science Education and Research Kolkata
November 22, 2020

Contents

1 Basic Workflow of the System: Flowchart




A
Register/...
Register
Login
Input Username
Is...
Invalid...
Input Password
Successful Registrat...
Registration
Input Username and Password
Web_user
Invalid...
N
Is present us...
N
Is password c...
Password...
Password...
Login
Input...
Found...
Choose...
Create Passenger Array
Get Number of passen...
Get passenger detai...
Store passenger in a...
N
Is...
Flight Booking
Payment: Card or...
Card/...
Card
Net banking
Enter Card...
Enter...
Output Ticket Details
Ticket Payment

2 Admin and User Login

This module has been set up to register the user into the website for the Flight Reservation System and also Log in to the portal once the user has registered. The admin can also login to the system to make flight-related edits from this section.

2.1 Structure data

[1]All user-related data, such as username and password are stored using the structure variable. The following structure variables are used to store the necessary data:

- Username is stored using the variable “name”, which is a char array type variable.
- Password is stored using the variable “pass”, which is a char array type variable.

Two functions have been used in this module:

2.2 reg()

Return type : void Argument type : void

[1]The reg() function is used to register the user into the Flight Reservation System portal and to direct the user to the login section once the registration has been done.

User data is added to a text file “Web_user.txt” using file pointer fp. The file opens in “a+” mode (Open for reading and appending. The file is created if it does not exist). The initial file position for reading is at the beginning of the file, but output/new file information is always appended to the end of the file.
Other variables used are:

- Char array type variable “checker” is used to input username from the user.
- Char array type variable “pw” is used to input password corresponding to a particular username from the user.
- Integer-type variable “valid_username” is used to keep a tab on the conditional segments in username registration.

The user is first required to enter the username using “scanf”. The username is compared with the already available data in the file using “strcmp”. If the username already exists, “valid_username” becomes 0, the user is required to enter a new username, and this action keeps repeating until the user enters a unique username.
Once a unique username is entered, “valid_username” becomes 1, the username is appended to the end of the file and the user is now required to enter the password using “scanf”. The password is appended to the end of the file and the user is directed to the login section.

2.3 login()

Return type : struct flags Argument type : void

[1]The login() function is used both for user and admin login into the Flight Reservation System.

For the admin login, the variables used are:

- flag variable “f.adminf” is used to keep a tab on the admin login permissions
- Char array type variable “admin_pass” and “admin_u” are set at default by the creators of the code
- Char array type variables like “name” and “pass” to input admin username and password

[1]Contributed by Souvik Paul

Once the admin enters the Login zone, they are required to enter the username and password, which is done using "scanf" into char array type variables "name" and "pw" respectively. If the entered data matches with the default admin username and password (compared with "strcmp"), "f.adminf" becomes 1 and admin can access other sections.

User data is accessed from the text file "Web_user.txt" using file pointer fp. The file opens in "r" mode (Opens an existing text file for reading purpose. File pointer is at the first char of the file) For the user login, the variables used are:

- flag variable "f.userf" is used to keep a tab on user login permissions
- Char array type variables like "name" and "pass" to input user's username and password
- Integer-type variables "checku" and "checkp" to keep a tab on username and password on comparison.

The user is first required to enter the username and password using "scanf". The entered username and password are compared with the username and password on the file using "strcmp". If both match, "checku" and "checkp" remain 0, "f.userf" becomes 1 and the user is logged in successfully. If "checku" remains 0 but "checkp" doesn't remain 0, then the user is required to input the login data again. If "checku" doesn't remain 0, then the user is not registered and is directed back to reg(). The do-while loop runs as long as "f.adminf" and "f.userf" remain 0.

3 Admin Actions

This module is responsible for the admin side of the flight reservation system. The functions used are listed below:

3.1 Structure data

[2]All the information regarding flight are store using structure variable. Flight structure contains the following variables:

- integer type variable "id" which stored flight id and array of integer type "date" which stores date of operation of flight in week(like 0-sun, 1-Monday,2-Tuesday and so on....).
- string type variables "name","source","destination" which store name, source and destination of the flight respectively.

3.2 add_data()

Return type : void Argument type : void

[2]The add_data() function is used to add data of new flight.

Data is added to a text file using file pointer fp. The file is open in mode "a" (Opens a text file for writing in append mode. If it does not exist, then a new file is created. Here the program will start appending content in the existing file content.). New flight information is written at the end of the file.

3.3 display_all_date_wise()

Return type : void Argument type : void

[2]File is open in mode "r"(Reading mode). The file pointer will move through the entire file until it reaches the end of file (EOF). at every iteration we read a structure from the file and each structure represents flight information. Then we check that if date provided by the user matches with the date of operation of the flight. Further, we display all such available flight on that specific date.

[2]Contributed by Subham Kumar Singh

3.4 display_all()

Return type : void Argument type : void

[2]File is opened in mode "r"(Reading mode) The file pointer will move through the entire file until it reaches the end of file (EOF). at every iteration we read a structure from the file and display all information of the flight.

3.5 update_data()

Return type : void Argument type : void

[2]Main file is opened in reading mode and a temporary file is opened in writing mode. The admin is asked which Flight data he/she want to modify with respect to the flight ID and that particular ID is taken as input. The file pointer will move through the entire file until it reaches the end of file (EOF). We maintain a variable "found"

- At every iteration a condition is check i.e. that flight ID is matched with the ID provided by the admin. If the condition is true then new information is asked from the user and then we write the structure in the temporary file and found is equal to 1 ,otherwise if the condition is false we simply write the structure to the temporary file.
- Ff found is equal to zero then display the message that record with that ID is not found, else again open main file in writing mode and temporary file in reading mode. Now copy everything from temporary and write it in main file.

Hence at the end we have updated main file.

3.6 del()

Return type : void Argument type : void

[2]This function works in a similar fashion like update_date function. It asks the admin which flight information he/she wants to delete from the file and takes the corresponding ID. Main file is opened in reading mode and a temporary file is opened in writing mode. At every iteration, write everything to the temporary file until we found the record. If the matching ID is found, i.e., found==1 (or do nothing) then close both the files.

- If found is equal to zero then display the message that record with that id is not found.
- Else again open main file in writing mode and temporary file in reading mode.
- now copy everything from temporary and write it in main file.

Hence, in the end we have updated main file,

3.7 search()

Return type : void Argument type : void

[2]User is asked to input the flight name he/she want to search. Then file is open in reading mode and a variable found is maintain equal to zero. Then at every iteration if flight name and input flight name matches then all information regarding the flight is displayed. And if found is equal to zero then a message is displayed that no records were found.

[2]Contributed by Subham Kumar Singh

3.8 mygetch()

Return type : void Argument type : void

[2]This function is basically a replacement of getch(). Its purpose is to hold up the program until user press enter.

3.9 day_of_week()

Return type : int Argument type : int

[2]In 1990, Michael Keith and Tom Craver published the foregoing expression that seeks to minimise the number of keystrokes needed to enter a self-contained function for converting a Gregorian date into a numerical day of the week. It preserves neither y nor d, and returns 0 = Sunday, 1 = Monday, etc.

4 Flight Booking

This section of the flight reservation system is responsible for the collection of passenger information, storing that information to a file, and displaying the final ticket. We have also created a file system which contains the information of seat availability for a given flight on a given date. The file name corresponds to these two parameters. Following are the structures and functions for these operations.

4.1 Structure passenger_info

[3] This is a structure, which has field for storing passenger information, including name, gender, age, student ID (if he/she is a student), age category (infant, child or adult) and the corresponding fare. A passenger's age category is automatically decided based on his/her age. Further we go on to declare a global variable of this type to store/display all the information.

4.2 seat_info()

Return type : int Argument type : char

[3]This function is used to extract the number of available of seats on a flight on a given date. The file name corresponding to this is "FlightIDDate.txt", which is passed as a char type pointer. An integer type variable "n" is declared to store the number of seats. The following processes take place:

- Open the file in reading mode ("r").
- If the file fails to open, open the file in append mode ("a"), and assign a value of 100 to n (which we assume to be the total capacity of the flight), and write this value to the file. (Inability to open a file means that flight on that date has no bookings and all seats are available.)
- If the file opens (*else* part), we seek the cursor towards the end of file using "fseek", and read the last integer from the file, and assign this value to "n".

4.3 disp_ticket()

Return type : int Argument type : int n, float total_fare, int seat

[3]This function displays the ticket for the journey using "printf" commands. PNR number and total fares are displayed along with passenger name, age, age category, gender, individual fares and student ID in a tabular format using *for* loop.

[2]Contributed by Subham Kumar Singh
[3]Contributed by Anmol Kumar

4.4 ticket_info()

Return type : void Argument type : char *flight_id, int n, int pnr, float total_fare, int seat

[3]This function is similar to disp_ticket, but the difference is that it prints the ticket information to the file which we created (passed as a char type pointer "flight_id") to store information corresponding to flight ID on a particular date. Instead of "printf", "fprintf" is used here, after the file has been opened in append mode "a". Further, we update the number of available seats as the difference between available seats ("seat") and number of passengers ("n").

4.5 collect_info()

Return type : float Argument type : int n, float distance_fare, int seat

[3]This function is responsible for the collection of passenger details from the user. The information is stored in a passenger_info structure type variable. The user is one by one asked his/her details including name, gender, whether he/she is a student and age. If the passenger is a student, he/she is further asked to provide a student ID. Passenger's age is used to categorize into infant, child and adult. Age, student parameter and "distance_fare" (fare corresponding to a particular flight) are then passed to "fare_calculator" to calculate individual fares and sum them up to calculate total fare, which is returned. The input/output operations are performed using "printf" and "scanf" functions.

4.6 fare_calculator()

Return type : float Argument type : int age, float distance_fare, int c

[4]The total fare is calculated as a sum of base fare, a flight's distance fare, student concession (if eligible) and taxes (10 %). The base fare is calculated based on the age as follows:

- If age is less than 3, then base fare is 200.
- If age is between 3 and 5, then base fare is 600
- Else, the base fare is 1000

The above age groups are also used while deciding the age category of a passenger.
The integer c takes two values 1 and 0 (depending on whether the passenger is a student or not), and is multiplied to a variable "student_concession" which stores the value of -400 (because it is a concession), and finally summed to the total fare. The total fare is returned.

4.7 ticket_pnr()

Return type : int Argument type : void

[4]This function generates a PNR number for the ticket using a built-in "srand" function. This function takes a seed value, which we give as the time of booking, and then map it to a 6-digit number using simple algebra.

4.8 book_tickets()

Return type : int Argument type : char *f1, float distance_fare

[4]This function is like the main function responsible for the complete booking procedure. All the above functions are called in a logical order, where the only external parameters are flight information and date (stored in a char type pointer "f1") and its fare (distance_fare). The input/output operations are performed using "printf" and "scanf" functions. We have restricted the maximum number of passengers that a user can add to 5, and implemented it using *if-else* statements.

[3]Contributed by Anmol Kumar
[4]Contributed by Harshit Kumar

5 Payment

This module is used to make the payment for the ticket booked.

5.1 booking()

Return type : void   Argument type : float

[5]This function asks the user to choose the type of payment preferred. There are two possibilities: (i) Card Payment (ii) Net Banking. Depending on the user's choice, "switch" directs the user to that particular kind of payment option. The function has 3 associated structs in "structs.h":

- Card: Used to store the card info

Sl. No.	Variables	Type	Length
1	card_type	char array	6
2	card_no	unsigned long int	-
3	cvv	unsigned int	-
4	expiry_date	unsigned int	-
5	expiry_month	unsigned int	-

- net_banking: Used to store net banking information

Sl. No	Variables	Type	Length
1	bank_name	char array	50
2	bank_id	char array	50
3	bank_pass	char array	50

- payment_details: Used to store complete information on the payment

Sl. No	Variables	Type	Length
1	payment_type	char array	10
2	payment_card	card	-
3	net_banking	net_banking	-

Additional inline variables, char array variables are included to be called when the user interacts the portal to make it user friendly. After completion of the process the user is given a final acknowledgement message about his ticket being booked. Changes that can be applied:

- A validation of the literals may be added to ensure proper input at the proper entries in this function.
- A card saver might be added so that user's card/net banking details may be stored in the database to facilitate quick transactions.

6 Color Changer

This module facilitates color change and is used to clear screen in two different distributions (Windows/Linux). A _COLOR_H macro is defined to ensure the functions are not formed multiple times.

[5]Contributed by Naksatra Kumar Bailung

6.1 color_change()

Return type : void Argument type : int

[5]This function changes the color of the next text being written. (Refer list in COLOR_CHANGER.C for k values)
This should work in Linux/Windows environment:

Variable	Type	Purpose
hConsole	HANDLE	To manipulate the color scheming in Windows terminal
k	int	To manipulate color scheme

7 Compiling Instructions

Dev C++ is not recommended to compile this code as it still uses tdm 4.9.2 equivalent version of gcc.

- Windows: (Recommended) gcc 9.0+ / clang 9.0+ (tdm compiler does not support cross platform code and escape sequences)
- Linux: gcc 9.0+ / clang 9.0+

[5]Contributed by Naksatra Kumar Bailung