Technical University of Košice
Department of Computers and Informatics

Problem Set 6
Parking jam

Ruslan Diakov
2020/2021

Assignment

Using the ncurses library, create a program (game, presentation or other artistic work), with the following minimal requirements:

- Project contains 2D world.
- Project meets at least 3 challenges:
  - Work with colors
  - Keyboard control (no Enter needed)
  - Multiple levels
  - Work in time (in the time the program is changed)
  - Work with command-line arguments
  - Work with files
- Project must be more complicated then the sample programs, with an adequate level of difficulty.

Game logic

The code consists of 43 functions:

- void menuStart(char* fileLogin[], char* filePassword[], int *DoneLvl, int *id) – program element that allows you to select one of the several listed program options "login ", "registration", "exit".
- void menuEnter(int *DoneLvl, int id) – program element that allows you to select one of the several listed program options "Levels", "exit".
- void menuLevels(int *DoneLvl, int id) – program element that allows you to select the game levels "level 1"," Level 2", "Level 3".
- void encoding(char password[20]) – encodes the user's password.
- void deEncoding(char password[20]) – decodes the user's password.
- bool equal(char str1[20], char str2[20]) – compares strings.
- void readLoginFile(char DBLogin[200][20], int *lenth, char* fileLogin[], char* filePassword[]) – read login file.
- void readPasswordFile(char DBPassword[200][20], char* fileLogin[], char* filePassword[]) – read password file.
- void readDoneLvlFile(int *DoneLvl, int id) – read done lvl file.

- void writeLoginFile(char login[20], char* fileLogin[], char* filePassword[]) – write login file.
- void writePasswordFile(char password[20], char* fileLogin[], char* filePassword[]) – write password file.
- bool login(char* fileLogin[], char* filePassword[], int *DoneLvl, int *id2) – log in to account.
- void registration(char* fileLogin[], char* filePassword[]) - create an account.
- void readDoneLvlsFile(char db[]) – read done lvls file.
- void writeDoneLvlFile(int id, int DoneLvl, int lvl) – write done lvl file.
- void game(int id, int *DoneLvl) – start lvl 1.
- void game2(int id, int *DoneLvl) – start lvl 2.
- void game3(int id, int *DoneLvl) – start lvl 3.
- bool FrontIsClear(int lenth1, int height1, int vertical, int gorizontal, char level1[lenth1][height1]) – check if there is a car ahead in lvl 1.
- bool FrontIsClear2(int lenth1, int height1, int vertical, int gorizontal, char level1[lenth1][height1]) – check if there is a car ahead in lvl 2.
- bool FrontIsClear3(int lenth1, int height1, int vertical, int gorizontal, char level1[lenth1][height1]) – check if there is a car ahead in lvl 3.
- void GoGame1(int lenth1, int height1, int *vertical, int *gorizontal, char level1[lenth1][height1], char level1Border[7][5]) – move the car forward in lvl 1.
- void Go2(int lenth1, int height1, int *vertical, int *gorizontal, char level1[lenth1][height1]) – move the car forward in lvl 2.
- void Go3(int lenth1, int height1, int *vertical, int *gorizontal, char level1[lenth1][height1]) – move the car forward in lvl 3.
- bool TheEnd(int lenth1, int height1, char level1[lenth1][height1]) - check if there are cars in the Parking lot.
- void DrawBorder2(int width, int height) – draw game border.
- void drawGame(int lenth1, int height1, int lenth2, int height2, int width, char level1Border[lenth1][height1], char level1[lenth2][height2]) – draw game.

- void DrawCursorVertical(int lenth1, int height1, int vertical, int gorizontal, int width, int space, char level1[lenth1][height1]) – draw cursor vertical in menu.
- void DrawCursorGorizontal(int lenth1, int height1, int vertical, int gorizontal, int width, int space, char level1[lenth1][height1]) – draw cursor gorizontalin menu.
- void DrawWin() – draw window when win.
- void DrawTimer() – draw timer in game.
- void DrawSteps(int steps) – draw count steps.
- void DrawInputLogin(int width, int height, int n) – draw window for input login.
- void DrawInputPassword(int width) – draw window for input password.
- void DrawLogPswd(int width, int y, int i, char x) - draw input login or password.
- void DrawBorder(int width, int height) – draw menu border.
- void DrawMenuStart(int width, int height, int vertical, int n) – draw window menu start.
- void DrawMenuEnter(int width, int height, int vertical, int n) – draw window menu enter.
- void DrawCursor(int vertical, int width, int space) – draw cursor in game.
- void DrawIncorrect(int width) – draw text when username or password is incorrect.
- void DrawCorrectProg(int width) – draw text when username or password is correct.
- void DrawMenuLevels(int width, int height, int vertical, int n, int DoneLvl) – draw window menu start.

Game data is stored in two two-dimensional arrays, where the first array represents the boundaries of the game world, and the second represents the Parking of cars . The array contains:

- ’ ’ – empty position
- ’^’ – car direction north
- ’v’ – car direction south
- ’<’ – car direction west
- ’>’ – car direction east

- ’|’ – vertical wall
- ’-’ – horizontal wall

Snake has two pairs of helper variables:

- int vertical, gorizontal
- char type;

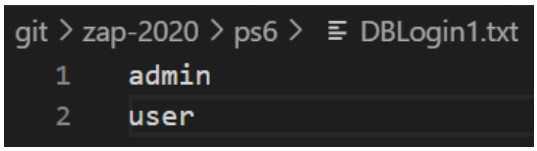
The first two determine the position of the car selection cursor, and the third determines the direction of the car.

The game loop is responsible for moving the car selection cursor. If the cursor is on the car and the “space” key is pressed, the car will move in its own direction until it hits another car or leaves the Parking lot. Rewriting both the contents of the game array and the output to the screen. If the player does not have time to get the cars out of the Parking lot, the Parking lot will start to close. The game loop also updates and rewrites the game status bar.

Game play

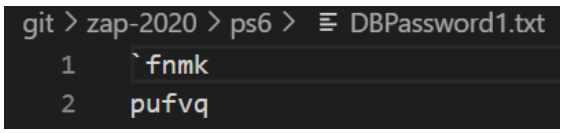
The game is encoded in the main program.c file but also contains additional files (menu. h, machineController.h, logReg.h, game.h, drawMenu.h, drawLogReg.h, drawGame.h) where all functions are described. The program also requires 3 or three additional files

DBLogin.txt - This file contains a list of logins of Registered users.



Obr. 1 file DBLogin.txt

DBPassword.txt - This file contains a list of encrypted user passwords.



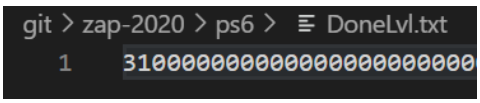
Obr. 2 file DBPassword.txt

password / encoded password

1 admin / `fnmk

2 qwerty / pufvq

DoneLvl.txt - This file contains a list of levels that users have completed.



Obr. 3 file DoneLvl.txt

3 means that the user “admin” has passed 3 levels

1 means that the user “user” has passed 1 level

The program is launched using the following command ./program “file with login” “file with password”

```
./program DBLogin1.txt DBPassword1.txt
```

Obr. 4 lunch program

When start the program, the user sees a menu in which can choose three items:

“Login” - enter to account.

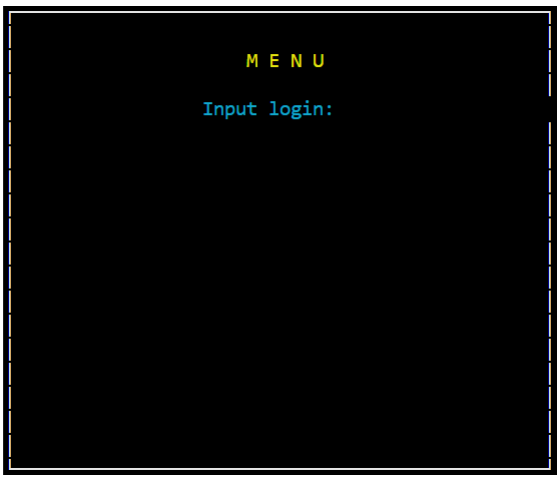
“Registration” - create a new account.

“Exit” - exit the program.

Obr. 5 Menu Start

Control in the program is implemented using the arrows on the keyboard and when you press the space bar, the command will be Executed.

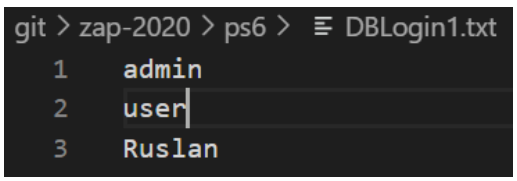
When users have chosen the option “Registration” it opens a window in which the user can enter a username and password.



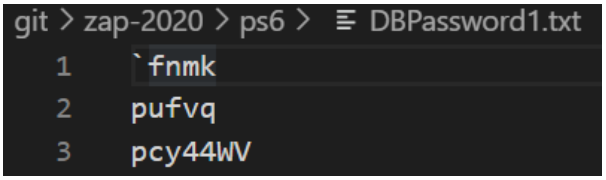
Obr. 6 Menu Login Reg

Obr. 7 Menu Login Reg

The username will be written to files with usernames, and the password will be written in Encrypted form to a file with passwords.



Obr. 8 DBLogin.txt



Obr. 9 DBPassword.txt

When the user has selected “Login”, a window opens where they can enter their username and password. At this point, the logins File opens And all data is Written to the dblogin[200][20]. The program also opens a file with passwords, decrypts them and writes the DBPassword[200][20]. After that, the username is compared with the password, if such a user exists, the program is logged in.

If you entered an incorrect username or password, the program will notify you .

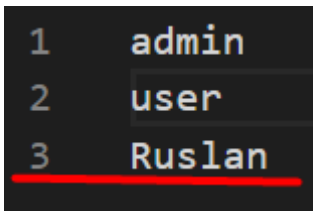
Obr. 10 incorrect username or password

Obr. 11 correct username or password

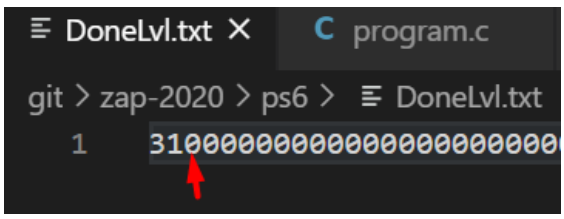
After successful authorization a second menu opens where you can select two items Levels or Exit.

Obr. 11 Menu Enter

When the user selects Levels, a new menu opens with the available levels. If the user Did not Pass the first level, then the next level will be closed and highlighted in red. In the file DoneLvl.txt Contains information About how many levels the user has completed.



Obr. 12 DBLogin.txt

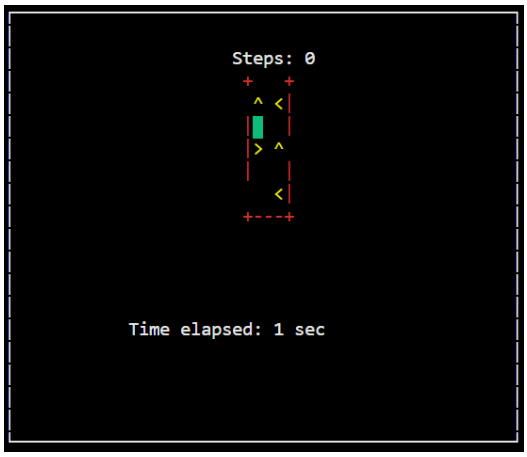


Obr. 13 file DoneLvl.txt

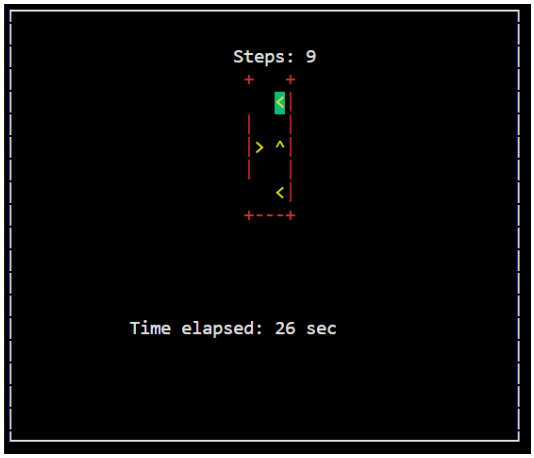
The game has Three different levels of difficulty.

Obr. 14 Menu Levels

Level 1 is the easiest. It contains five cars in the Parking lot. If the user does not have time to withdraw the Central car from the Parking lot in 10 seconds, the Parking lot is closed by a barrier and the user loses. The number of steps Taken by the user is also counted.

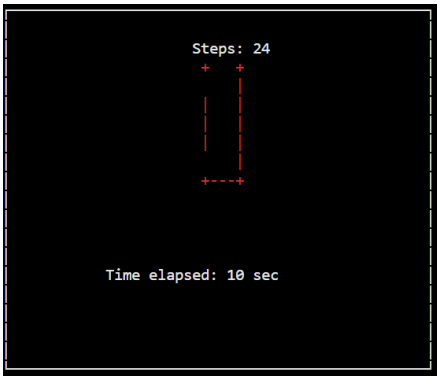


Obr. 15 Start Lvl1



Obr. 16 Barrier closed

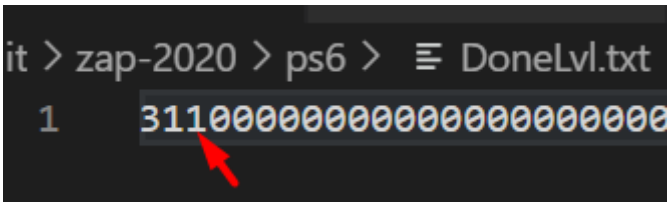
The game is considered completed if there are no cars left in the Parking lot. After the user has successfully completed the level. The task is to select a car press the space bar and take it out of the Parking lot.



Obr. 17 Empty Parking lot.

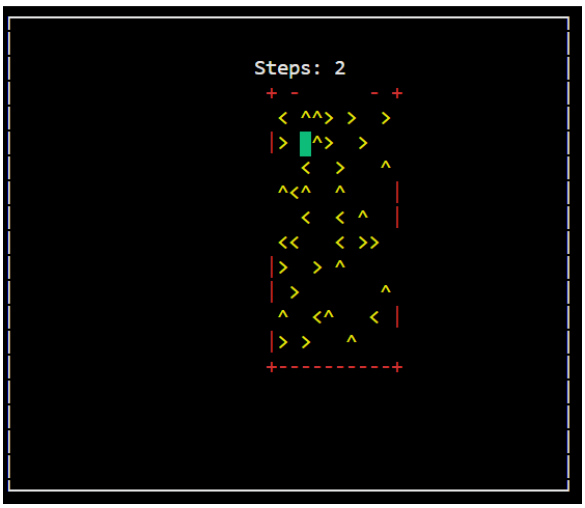
Obr. 18 WIN 1 lvl

The file that contains the user's Completed levels will be overwritten with the new completed level.



Obr. 19 Menu Levels after complete lvl 1. Obr. 20 Overwritten DoneLvl.txt

The second and third levels differ in the Size and number of cars in the Parking lot. and on level 3, a new southbound direction is added.



Obr. 21 Levl 2 Obr. 22 Levl 3

Conclusion

The game has its own database of users and their achievements in the game. 3 levels of varying difficulty, as well as changing the map over time.