Gliwice, 04.07.2019

Computer Programming Laboratories

Topic:
Tetris-game with I/O file operations and dynamic memory allocation

Autor: Paweł Lutostański Kierunek: Informatics, sem. II, grupa I, sekcja II
Prowadzący: dr inż. Anna Gorawska

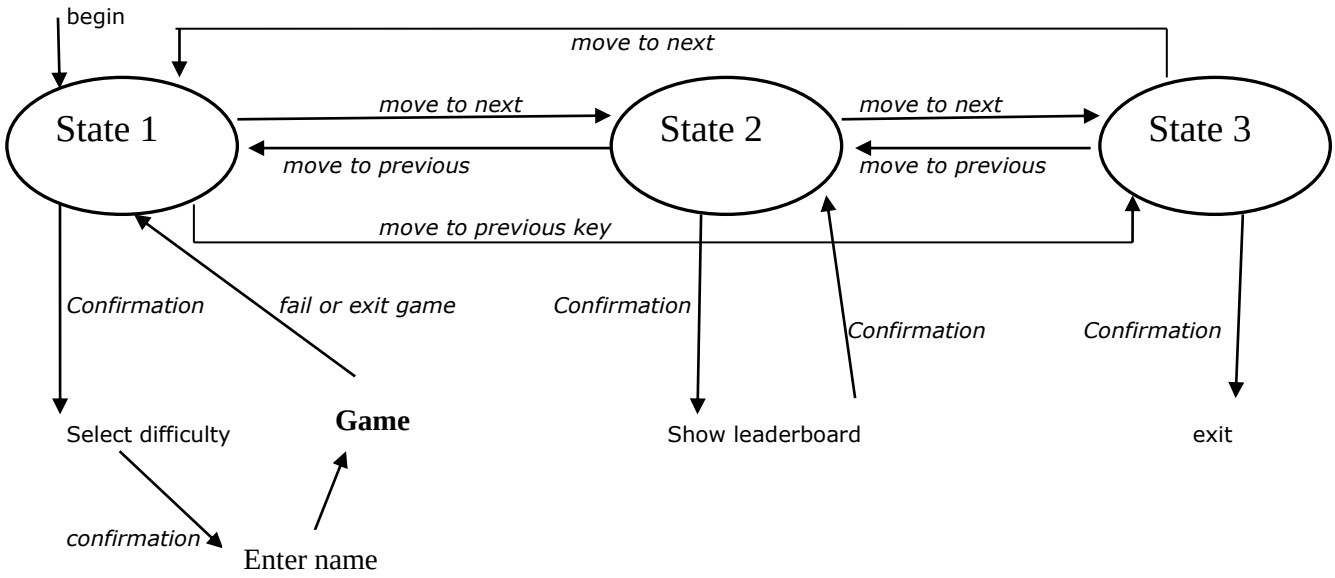
1. Topic

The main aim of project was creating program with I/O file operations and implementation of dynamic data structures. I decided to create computer game inspired Tetris. The program will use dynamic data structures to store position of figure and I/O file operations to create leader board .

2. Analysis, and projecting

• Basic Assumptions

Firstly program should begin with menu. I decided to model menu on state machine where states were user by entering keys can pick and choose states in way presented on diagram:



Where “Select difficulty” works similarly, user can increase and decrease the difficulty (if lower than 0 then it becomes the maximum, if greater than maximum it becomes zero). On the other hand, game would wait in the loop, for player response and then perform operations, till player fails or exit game after that program should save score to file. The proper function will rewrite top five scores with names from file, if a score from previous game will be greater than just rewritten score then instead of that score new score will be written. After that there will be rewritten rest without last in order to avoid useless data in file. The program will work on computers with WINDOWS 10 operating system.

• Structures

There are three structures used in program:

1. **Point** which consists of *x* and *y* integers defining location of point.
2. **Block** which consists of Point *position* and char *symbol*

3. **FigureLList** linked list of FigureLList nodes, each node have pointer to *next* node of list and Block *element*.

The reason why linked list was implemented is the fact that number of blocks of each figure will be random. During creation of figure there is a rules:

- Let Figure have a block with point (x,y) then next figure's block should have point with (x±1,y ±1).
- The Figures blocks can overlap with other's blocks

Example of FigureLList illustrates following diagram:

Results in program:

The program operates on the limited Cartesian coordinate system, it moves points by unit vectors depending on keys pressed by user. However user do not need to be familiar with these operations to play game.

3. External specification

User should launch program on WINDOWS 10 operating system. His display should be Full HD, otherwise player should change size of console font.

• Controls

User can move in the menu, after pressing a proper keys he will immediately see result as mark before one of three states. The top one “NEW GAME”, the middle one “LEADERBOARD”, the lowest one “EXIT” He can operate with them with following keys:

- ‘W’, ’w’, ‘D’, ’d’, right and upper arrow - allows user to change a state to “upper” one if state is the top one and user press one of these keys then state will change to the lowest one.
- ‘S’, ’s’, ‘A’, ’a’, left and lower arrow - allows user to change a state to “lower” one if state is the lowest one and user press one of these keys then state will change to the top one.
- SPACE and ENTER – these are confirm keys when user press them then he choose current state and moves to next steps of states described below.
- ESCAPE when player press it program switches off.

When user press confirm key in following state:

1. NEW GAME – program will ask user to enter difficulty of program. User can change them in the same way as in menu. If player press ESCAPE then program returns to menu. If user want to choose displayed difficulty, which is responsible for frequency of bigger figures, he should press a confirm key. Then he will be asked to enter the name. After that he can return to menu by pressing ESCAPE or press any other key to continue and start **GAME**.

2. LEADERBOARD- program will display a leaderboard from the file, if place would not exist in file the program will show ‘-‘ instead of player name and 0 as score. User can press confirm key or ESCAPE to return to menu.

3. EXIT- program will quit.

During playing **GAME** user with:

- ‘W’, ‘w’, ‘S’, ‘s’, down and upper arrow- can move a figure down if there won’t collision.
- ‘D’, ‘d’ and right arrow- can move a figure right if there won’t be collision . Figure will also go down anyway.
- ‘A’, ‘a’ and left arrow - can move a figure left if there won’t be collision. Figure will also go down anyway.
- ESCAPE can return to menu, current score will be saved.

• Game

Figure can move down right or left. If on the right there is figure that have fallen or frame then orders to move right will be ignored. If on the left figure is blocked then situation is similarly as it was with the right side blocking. When figure is blocked and cannot move down it’s position is saved in array and there is created a new figure.
User should operate with figures and fill the rows of an array with them. When he succeed his score will be increased by the square of the rows filled with one figure then multiplied by 100 (100*(rows filled) ^{2}).
If the pile of fallen figure rise above the limit marked by the line player fails game and if his score was greater than

score, of at least one scores from file, then name and score will be saved both in the file “leaderboard.txt” in proper order. If there is such file program creates it and writes data there.
User is not permitted to edit file.

4. Internal specification

Preprocessor:

array.h

`X_MAP 18`	X_MAP is width of array equal to 18
`Y_MAP 47`	Y_MAP is length of array equal to 47
`FALLEN_BLOCK -78`	Symbol of already fallen Block it is equal to -78
`CONTROLED_BLOCKRODUCT -79`	Symbol of Block belonging to figure controlled by player it is equal to -79

structure.h

`_CRT_SECURE_NO_WARNINGS`	Prevents compilation warnings in VS 2017
`_WINSOCK_DEPRECATED_NO_WARNINGS`	Prevents compilation warnings in VS 2017

file.h

`LINE_SIZE 15`	Size of line read from file
`TOP_PLAYERS 5`	Number of top highest

Functions

Program functions operate with console and file I/O and manages list.

structure.h	
`typedef struct Point { int x; int y; };`	Cartesian coordinate system's point (x,y) but only integers are legal, x,y will describe location in array
`struct Block{ struct Point position; char sym; };`	Block is defined as point with symbol
`struct FigureLList { struct Block* element; struct FigureLList* next; };`	FigureLList is Linked List of Blocks
`void addToList(struct FigureLList** head, struct Block* data);`	Appends Block to figure (last *next of list is always NULL) * @param head- pointer to head of figure * @param data- pointer to Block

`void clearList(struct FigureLList** head);`	frees memory where nodes and their elements were allocated, then set head to NULL @param head-pointer to head of figure
`struct Block* initBlock(int x, int y, char c);`	Creates new Block and allocate memory for it @param x- x coordinate of Point of new Block @param y- y coordinate of Point of new Block @param c- symbol of new Block @return pointer to new Block
setFigure.h	
`int findMinX(struct FigureLList** head);`	Find minimal x coordinate of figure @param head- pointer to head of figure @return minimal x coordinate
`int findMinY(struct FigureLList** head);`	Find minimal y coordinate of figure @param head- pointer to head of figure @return minimal y coordinate
`void moveBlockaByXY(struct FigureLList** head, int xMore, int yMore);`	Moves by given vector each figure's Block which head was given @param head- pointer to head of figure @param xMore- move each x coordinate by xMore @param yMore- move y coordinate by yMore
`void randomFigure(struct FigureLList** head, int difficulty);`	Creates random figure (list of blocks),where each block of point (x,y) have next block which point (x+/-1,y+/-1) @param head- pointer to head of figure (figure should be cleared previously) @param difficulty- the higher it is the more big figures appear.
figureOper.h	
file.h `int leftCheck(char arrayOfChars[Y_MAP][X_MAP], struct FigureLList *head);`	Checks if there would occur collision with array symbols after moving figure left @param arrayOfChars- array where symbols are @param head- head of figure @return 1 if no collision 0 is collision
`void allLeft(char arrayOfChars[Y_MAP][X_MAP],struct FigureLList **head);`	Moves all blocks cords of figure left if there would not be collision @param arrayOfChars- array for which we leftCheck @param head- pointer to head of figure
`int rightCheck(char arrayOfChars[Y_MAP][X_MAP], struct FigureLList *head);`	Checks if there would occur collision with array symbols after moving figure right @param arrayOfChars - array where symbols are @param head - head of figure @return 1 if no collision 0 is collision
`void allRight(char arrayOfChars[Y_MAP][X_MAP], struct FigureLList **head);`	Moves all blocks cords of figure right if there would not be collision @param arrayOfChars- array for which we rightCheck @param head- head of figure

`int checkBottom(char arrayOfChars[Y_MAP][X_MAP], struct FigureLList* head);`	Checks if there would be collision with array symbols after moving figure down @param arrayOfChars- array of symbols @param head- pointer to head of figure @return returns 1 if there would be collision and 0 if not
`int bottomDown(char arrayOfChars[Y_MAP][X_MAP], struct FigureLList **head);`	Operates with other "moving figure in array" functions and moves figure to one cell down or all to save point (X_MAP/2 , 0) @param arrayOfChars- array of symbols @param head- head of figure @return
`void addAllBlocks (char arrayOfChars[Y_MAP][X_MAP], struct FigureLList **head, char symbol);`	Add to symbol to array cells defined by Points of Blocks of Figure @param arrayOfChars-array @param head- pointer to head of figure @param symbol- symbol added to array
array.h `void fillArray(char arrayOfChars[Y_MAP][X_MAP]);`	The function fills array @param arrayOfChars array (given by pointer) is filled in proper way, frames and spaces
`void display(char arrayOfChars[Y_MAP][X_MAP]);`	The function displays array @param arrayOfChars- array is displayed
`void addPlayer(char arrayOfChars[Y_MAP][X_MAP], struct Block* blo,char sym);`	The function adds character sym to array coordinates given by Block's points @param arrayOfChars- array(given by pointer) to which we put Block @param blo- Block but we use from it only Point's x and y @param sym- a symbol put to the array
`void addPoint(char arrayOfChars[Y_MAP][X_MAP], int x, int y,char symbol);`	Function adds symbol to array coordinates x and y (the main difference between addPlayer is the fact that addPoint does not require Block structure) @param arrayOfChars- array(given by pointer) to which we put symbol @param x- x coordinates @param y- y coordinates @param symbol- symbol put in (x,y) point
`void deleteRow(char arrayOfChars[Y_MAP][X_MAP], int whichRow);`	deleteRow replaces row with spaces and then replace array cell with cell "above", if it occure there '_' it replaces array with space @param arrayOfChars- array(given by pointer) in which we delete row @param whichRow- defines which row function deletes

`int checkRows(char arrayOfChars[Y_MAP][X_MAP]);`	checkRows checks rows and order to deletes rows filled with fallen figures @param arrayOfChars- array(given by pointer) in which we checks rows @return - reters number of deleted rows
`int addLine(int whichRow, char arrayOfChars[Y_MAP][X_MAP]);`	Adds line conisted of '_' to array @param whichRow- defines in which row put line @param arrayOfChars- array(given by pointer) in which we add line @return- returns 1 if '_' will overwrite symbol of figure
game.h	
`void gotoxy(int column, int row);`	The function is used to move the cursor to the desire position in console (ONLY WINDOWS). @param column- to which column move cursor @param row- to which row move cursor
`int points(int addThem,char mode);`	points performs operation on static integer depending on mode @param- addThem nubmer which is added in add mode @param- mode if it is 'a' that means operates in add mode and adds addThem to pool, if 'r' resets static variable @return- returns static integer after mode operations
`void displayScore(int score);`	The function displays "SCORE:" and argument of function @param score- this variable is displayed
`int fail();`	fail displays fail message and waits till player press Enter @return- it is always 0
`int game(int difficulty);`	game operates with other functions responible for game (figure operations, displaying etc.) @param difficulty- defines difficulty, the higher it is the more big figures appear @return- returns number of points right before exit operation
file.h	
`void savingScore(char playerName[15], int score);`	Copies TOP_PLAYERSx2 lines to appropiete arrays, then write parameters in sorted order score and playerName (WARNING! Function does not sort array i put data only to sorted or file, if array is not sorted by scores then after this function work it will still remain unsorted) @param playerName- user's string @param score- user's score

`void displayBlank(int i, int j, int k);`	Displays blank Leaderboard for k place with score=0 and playerName ='-' @param i- collumn where Leaderboard should start displaying @param j- row where Leaderboard should start displaying @param k- which placement is blank
`void showLeaderboard(int i, int j);`	Displays message and then scores and playerNames from filewith such formating: "<Placement> . <playerName> <score>" @param i- collumn where Leaderboard should start displaying @param j- row where Leaderboard should start displaying
menu.h	
`int whichChoice(char c,int max);`	The Function operates with static variable @param c- is inputed charater, depending on it static integer changes @param max- maximal value of static variable @return- returns changed static integer
`int selectDiff(char key);`	selectDiff displays message to user about difficulty after changes @param key- depending on key and whichChoice we obtain difficulty @return difficulty- <0-6> depending on previous value
`void displayMenu(int whereMark);`	The function displays message to user about operation performed after action @param whereMark- defines before which operation displays Mark
gameName.h	
`void symbolXTimes(int X, char symbol);`	The function prints in console symbol X times in console @param X- number of print repetions @param symbol- symbol printed

`void row1Print();`
`void row2Print();`
`void row3Print();`
`void row4Print();`
`void row5Print();`
	Functions prints following rows of "tetris" made of other symbols

`void displayGameName(int col, int row);`	The function operates with other functions and displays word "tetris" made of other symbols @param col- collumn where is displayed first row @param row- row is the space from left console output side

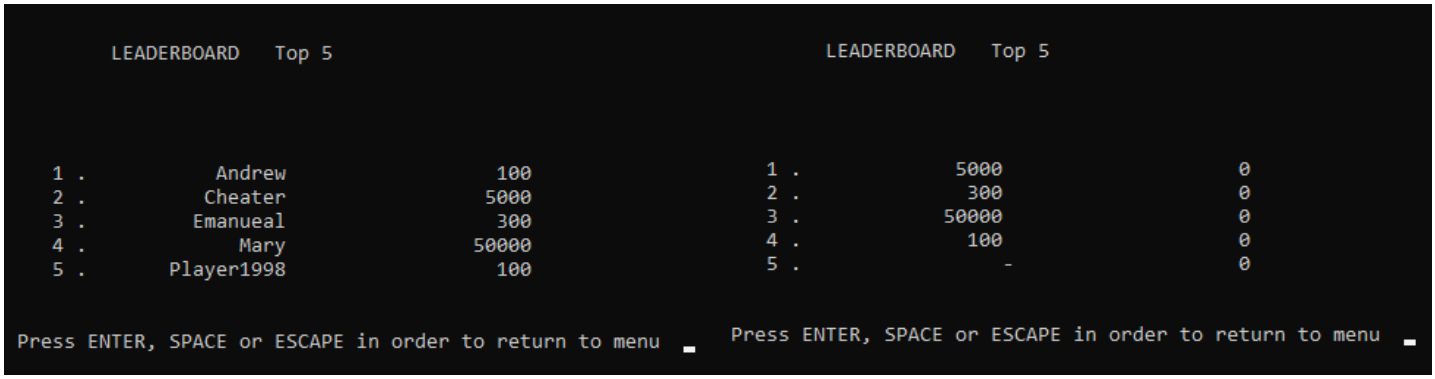
Choosen Variables

Game.c `struct FigureLList* figure;`	Head of list. We send pointer of figure to randomFigure function

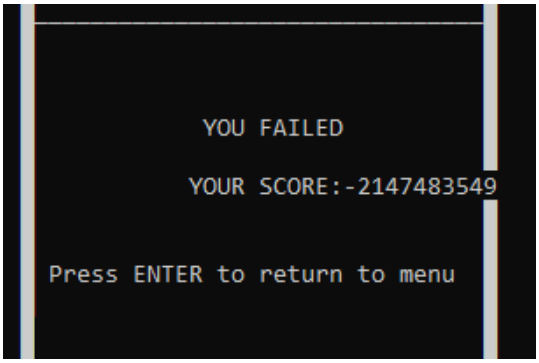
	to obtain new figure. After figure fall we send pointer of figure to clearList.
`static int points;`	Static variable program add 100*(rowsfilled) ^{2} or resets it before and after ending game.
`char  key;`	Key pressed by user, depending on it are performed certain functions.
`char gameMap[Y_MAP][X_MAP];`	Two dimensional array of chars with Y_MAP columns and X_MAP rows.
main.c `char key;`	Key pressed by user, depending on it are performed certain functions.
`int points;`	Number of points obtained by user during game. If this score is higher than previous top 5 scores it will be saved to "leaderboard.txt".
`char name[15];`	Name entered by user it is saved after user score.
menu.c `static int choiceVal;`	Value of choice defined by keys pressed by user.

5. Testing

Program was tested in many ways. No memory leaks reported or crashes caused by file-related operations (lack of file). Program should not crash during work. There were two reported bugs. First one, if user change "leaderboard.txt" it may result in displaying results in wrong way, like not displaying names or displaying placements in wrong older.



The remain reported bug concerns displaying score when player obtain greater score than 2147483600, then his score will become negative.
The reason why it happen is a maximal Value of int which is 2,147,483,647.
After fail the with such score, the frame will be overwritten during displaying message.



6. Conclusions

The program required knowledge about dynamic memory allocation and how to perform I/O file operations. Manipulations of standard output also was required to order program to overwrite data instead of clearing it and then writing it one more.