Project “Fishing”

Game instruction

1. This is a simplified version of board game “Hey, that’s mine fish".
2. Game is played in turns - in each turn each player moves one penguin. The aim is to collect the most fish.
3. The board to move penguins consists of ice tiles, arranged in the shape of a rectangle with dimension m x n (m - rows, n – columns). Each tile has 1, 2 or 3 fish on them, or is empty.
4. There are two phases of the game:
   a) Placing the penguins on the board;
   b) Playing the game.
5. Placing the penguins:
   a) Players receive the board dimension with fish placement and the information on number of penguins for each player.
   b) In each turn each player places one of her penguins on the board. The ice tile chosen for placing the penguin must be unoccupied and must contain exactly one fish.
   c) Placing the penguins is repeated until each player places all her penguins.
6. Playing the game:
   a) In her turn, a player chooses one of her penguins from the board and moves the penguin in a straight line over ice tiles. Penguin can be moved in any direction as far as the player wants but only over unoccupied ice floes. This means that penguin cannot “jump” neither over other penguin, nor over empty tile.
   b) Player collects the tile (and number of fish) from where the penguin started its movement, thereby creating an empty tile which penguins can't cross on future turns.
   c) When a penguin can't move, it's removed from play with its owner claiming the tile on which it stands.
   d) Game ends when there are no more penguins on the board.
7. The player who collects the most fish wins.

Your task

Your aim is to deliver a program that will play the game on your behalf. File format of the game is provided as follows.

File format with the game state

row 1 (board dimension): m n

rows 2 to m+1: n fields separated by single spaces, each field consists of 2 digits: first digit represents number of fish (1-3), second digit represents player’s number (1 to at most 9 players, or 0 if the tile is unoccupied). A combination of numbers 00 is also possible to represent an empty tile. In her turn player modifies two fields: field of

departure (to 00) and field of arrival (only second digit). In the placing phase only field of arrival is modified. In case of no movement only departure field is modified.

row m+2 and consecutive: 3 fields separated by single spaces: first field is player ID (string, without spaces, without quotation marks), second field is player's number (1 to 9), third field shows number of fish collected so far. In her turn player modifies the third field in a row with her ID, adding the first digit from departure field. If there is no row with player's ID (beginning of the game) player must add a corresponding row at the end of file, containing the ID of her choice, consecutive player's number and number 0.

Each row may end with any number of spaces.

Your program should accept command line parameters:

- `phase=phase_mark` – `phase_mark` can take value `placement` lub `movement`
- `penguins=N`, where `N` is a number of penguins that player has; This parameter can
- only be used if `phase=placement`;
- `inputboardfile` – name of file that contains current board
- `outputboardfile` – name of file in which the new board should be stored
- `id` – request to display player's ID end exit the program.

`iputboardfile` and `outputboardfile` can be the same file (can have the same name)

Examples:

```
penguins.exe phase=placement penguins=3 inputboard.txt outpuboard.txt
penguins.exe phase=movement board.txt board.txt
penguins.exe id
```

If `phase=placement`, then the program is supposed to place one penguin on the board and save the board to the output file. The program should only place penguin if they are not all placed on the board, assuming that the number of penguins to be placed is defined by penguins parameter. Therefore, the following sequence of runs should place all penguins for two players and store the final board in file outpuboard.txt:

```
player1.out phase=placement penguins=2 board0.txt board1.txt
player2.out phase=placement penguins=2 board1.txt board2.txt
player1.out phase=placement penguins=2 board2.txt board3.txt
player2.out phase=placement penguins=2 board3.txt board4.txt
```

If `phase=movement`, then the program is supposed to play one turn, i.e., collect fish, empty related tile and move one penguin. The following sequence of runs arranges 2 turns for two players:

```
player1.out phase=movement board4.txt board5.txt
player2.out phase=movement board5.txt board6.txt
player1.out phase=movement board6.txt board7.txt
player2.out phase=movement board7.txt board8.txt
```

If parameter `id` is used, then other parameters are ignored. Program should display only the player's ID hidden in the code. This option enables the game master to identify players.

Interactive game

The program should also be ready to be compiled for an interactive mode (use preprocessor directives to choose the mode). In the interactive mode the program should ask a user for decisions about penguins placement and movements. The program should work in a loop, each loop should be responsible for one action of a user. In each loop (turn) the program should read input file and write to the output file.

Returned result of the program

Program should end in a controlled way, returning to the operation system one of the values (specified as an argument of `return` in `main`):

- 0 - program made the correct move
- 1 - program can not make any move (in placement phase - all penguins have been placed; in movement phase - there are no more penguins of the player)
- 2 - game state error; in such a case additional error message should be displayed
- 3 - internal program error

Additional issues: think how you can prevent cheating by your opponent.

Gameplay

Game is managed by game master, i.e. operating system script provided by Instructor. Script generates the board, then players' programs are run in a loop. For example, let `program1.exe, program2.exe and program3.exe` be the programs provided by players 1, 2 and 3, then the script works according to the following pseudo-code:

```
generate_board.exe
while(true)
      kod1 = program1.exe  phase=placement penguins=3 board.txt board.txt
      kod2 = program2.exe  phase=placement penguins=3 board.txt board.txt
      kod3 = program3.exe  phase=placement penguins=3 board.txt board.txt

      if (kod1 == 1 && kod2 == 1 && kod3 == 1) break;
}

while(true)
      kod1 = program1.exe  phase=movement board.txt board.txt
      kod2 = program2.exe  phase=movement board.txt board.txt
      kod3 = program3.exe  phase=movement board.txt board.txt

      if (kod1 == 1 && kod2 == 1 && kod3 == 1) break;
}
```

Teamwork

Project is made in groups consisting of 3-4 persons. Group members are selected by Instructor, you are not suppose to change the follows membership.
It is group project, however, each student is expected to be able to explain any part of his/her code. Contribution of each student must be clear and reflected in the source files. Each group is expected to deliver several reports containing information about work progress and each group member contribution. Teamwork supporting tools will be utilized, including GitLab.

Git repository

To ease the team work, the use of some version control system is required. Git is one of the most popular ones. It is required that each group creates its own git repository (git project). It allows both web based and command line based access to projects. See: https://gitlab.com

Assessment [30]

1. Reports from each stage, submitted before the next meeting to GitLab. (16 ptcs., detailed scores provided in the schedule below)
2. Quality of the final code (appropriate data types and data structures, function breakdown, formatting, comments, etc.). (4 ptcs.)
3. The division of the source code into files. (2 ptcs.)
4. Fulfilling the requirements and functionality of the program. (3 ptcs.)
5. Teamwork quality (3 ptcs.)
6. Game result (2 ptcs.)

Schedules (7 meetings)

The implementation of the project should consist of the following parts:
1. Introduction, preliminary discussion, kick-off. Establish your team. Use flowcharts to sketch the general concept. Organize your work (GitLab required).
    - Report: team, flowchart, work organization (1 pts.)
2. Preliminary code for interactive mode: main loop, interaction with user.
    - Report: main loop and user interaction idea, decision on data structure (1.5 pts.)
3. Structure of the project - files, main functions.
    - Report: description of the program structure - files, main functions, dependencies between them. (2.5 pts.)
4. Data structures for board, printing board on the screen.
    - Report: description of internal data structures representing the board and other elements of the game. (3 pts.)
5. Handling the files, runtime arguments processing.
    - Report: implementation of downloading and saving the state of the game, processing of input arguments. Description of recognized errors in the input file. (4 pts.)
6. Algorithm, final data structures.

- Report: implementation of interactive mode game; autonomous algorithm description and implementation. (4 pts.)

7. Tournament, final assessment.

Reports are presented to the audience during the next meeting (meetings 2-6).

Remember - it's better to have simple solution that will work than the most sophisticated algorithm but not working. Try to work incrementally, be agile!