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 game board is arranged in a square grid with m rows and n columns. Each field of the grid can be empty or can have an ice floe with 1, 2 or 3 fish on it.
4. Each game is played on a new, possibly randomly generated, board. Each player has a certain (known) number of penguins.
5. There are two phases of the game:
   a) Placing the penguins on the board;
   b) Playing the game.
6. Placing the penguins:
   a) In a single turn the player places one of his penguins on an unoccupied ice floe with exactly one fish on it.
   b) When placing the penguin, the fish on the target ice floe is collected by the player and removed from the ice floe. This way the ice flow with a penguin does not have any fish.
   c) The move is compulsory.
   d) The placing phase ends when all the penguins are placed on the game board.
7. Playing the game:
   a) In his turn, the player chooses one of his penguins and moves it in a straight line (only along the grid lines) to another unoccupied ice floe. The previous ice floe is removed from the board. The move over the field without ice floe or over other penguin is forbidden. When placing the penguin, the fish on the target ice floe is collected by the player and removed from the ice floe, immediately.
   b) If possible, the move is compulsory.
   c) Game ends when no penguin can make a move.
8. The player who collects the most fish wins.

Your task

Your aim is to deliver a program that will play the game, either autonomously or interactively, within the proposed procedure (see later).

The text file format describing the state of the game

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 (0-3), second digit represents player’s number (1 to at most 9 players, or 0 if the tile is unoccupied). A combination of numbers 00 represents the grid field without the ice floe. In his turn the player modifies the field of departure and field of arrival. The field of departure is modified only in the second phase.

row m+2 and consecutive: 3 fields separated by single spaces: first field is player ID (string of any length, without spaces, without quotation marks), second field is player’s number (1 to 9), third field shows the number of fish collected so far (the score). If there is no row with player’s ID (at the beginning of the game) player must add a corresponding row at the end

of the file, containing the ID of his choice, consecutive player's number and the number of collected fish (which will be 1 in the case of the first placement action). Otherwise, the player should alter his current score by the number of the collected fish, and leave other fields unchanged.

Each line of the file may end with any number of spaces.

Your program should accept command line parameters:

- `phase=phase_mark` – `phase_mark` can take value `placement` or `movement`
- `penguins=N`, where `N` is a number of penguins that player has; this parameter is only used if `phase=placement`;
- `inputboardfile` – name of the file that contains current game state
- `outputboardfile` – name of the file in which the new game state 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 read the game state from the file, place one penguin on the board, save the game state to the output file and exit. The program cannot place more than `N` penguins on the board (as defined by the `penguins` parameter). In such a case the program exits immediately returning appropriate error code (see later).

If `phase=movement`, then the program is supposed to read the game state, move one of his penguins, alter the score, save the game state and exit. If the move is not possible, the program should exit immediately returning appropriate error code (see later).

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 allow to play interactive game between two players sitting in front of the computer. The game state does not have to be stored in the output file. The game state can be presented in a text mode (graphics mode is not compulsory).

Result returned by the program

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

0 - program made the correct move
1 - program can not make any move (in the placement phase - all penguins have been placed; in the movement phase – all penguins are blocked)
2 – error of the game state in the input file; in such a case additional error message should be displayed
3 - internal program error, possibly with additional message

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 (players do not create initial 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;
}
```

Programs compiled for an interactive mode should allow a human player (instead of the computer) to make a decision about next placement/move.

Teamwork

Project is made in groups consisting of 3-4 persons. Group members are selected by Instructor, you are not supposed to change the 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. Tournament 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!

Q & A

1. Player ID is some string (sequence of characters including letters or digits, but without spaces) hard coded in the program and unique for each program (player, team). It could be some fancy name, or just specification of the class and the team number like this: c103t07. The ID is used to recognize player's penguins and the game score. In the placement phase, the missing player's ID denotes no penguins have been placed on the board, so far.

2. .....