Project “Penguins”

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 further in this text).

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. Teamwork supporting tools will be utilized, including GitLab. It is group project, so consider choosing a project manager among team members to ease the workflow. 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 as well as clearly visible at GitLab.

Git repository

To ease the team work, the use of 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. Use faculty gitlab server: https://gitlab-stud.elka.pw.edu.pl/

Interactive game

The program should 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).

Autonomous game

Gameplay in autonomous mode

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 students 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.

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
- `name` – request to display player's name 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 name
```

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 `name` is used, then other parameters are ignored. Program should display only the player's name. This option enables the game master to identify players.

Result returned by the program

Program should end in a controlled way, returning to the operating 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 identifying the cause and the position of the error in the input file
3 - internal program error, possibly with additional message

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

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 endof 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.

Be aware that each line of the file may end with any number of spaces, try to prepare as much robust program as you can.

Assessment (max 30 points)

1. Assessment of your progress made during T1-T5 (10 points, detailed scores provided in the schedule below)
2. Final assessment and tournament during T6 (20 points)
   a) Project structuring (division of the source code into files, function breakdown) (4 points)
   b) Code quality (clarity, simplicity, comments, good naming convention, formatting, appropriate data types and data structures) (5 points)
   c) Testing (automated tests, documented manual tests) (4 points)
   d) General impression (5 points)
   e) Tournament result (2 points)

Extremely important remark: assessment is made individually (if the program is perfect, but your contribution is low, you will get very few points; if quality of the code is high, but your part is weak, you will get much less points than others; if your program is not fully functional, since one team member failed in his tasks, you still can get maximum).

To trace your activity and see your contribution we will use GIT statistics. Activities not registered at GIT or software not added to repo will be not taken into account.

Schedules (with points)

T0. Kick-off meeting, introduction to the challenge.
Start working on flowcharts to sketch the general concept (start with an interactive mode and then extend it for autonomous mode). Everybody needs to deliver individual flowchart (or flowcharts) one day before T1.

T1. Discussion of submitted flowcharts (max 2 points).
Establishing the teams, preliminary discussion in teams. Agree on communication channels in the team. Organize your work (GitLab required). Put chosen flowcharts to the repository. Creating preliminary code based on the flowcharts.

T2. Present preliminary code for an interactive mode: main loop, interaction with user. Show it and run it. (max 1 point)

T3. Present the structure of the project in terms of division into files and functions. Present declarations of each function with comments describing arguments, results and purpose of the function. Create issues on GitLab and assign team members to issues. (2 points)

T4. Present data structures for board, printing board on the screen. The user should be able to set the penguins, make movement, but still it can be imperfect. (2 points)

T5. Handling the files, runtime arguments processing. Your program should read and write to file

according to the specification. It should interpret properly command line arguments. (3 points)

T6. Tournament, final assessment.

Each presentation cannot be longer than 8 minutes. Everything what is to be presented and to be assessed must be pushed to GIT before the meeting (create folders with meaningful names). If you had team meeting, then make a note and attendance list and put it on GIT. Alternatively, usage of individual group channel on Teams is recommended. Let every team member say few words about their contribution during last period. The main person to present the work you did can be chosen by the lecturer.
The aim of the presentation is to get the feedback and make sure that the project is going well. You are expected to be an active listener - remember, that you can find answers to your concerns or brilliant ideas in other students' projects and use it in your project.

TIPS & TRICKS

1. Always keep the source code ready to be compiled - use commenting for that
2. Assign the tasks to everybody in the team, especially from T3, when you can work in parallel.
3. Don’t start with loading/saving files. You may do a lot of things having some constants in your program and then you can replace them with data read from file.
4. 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 (To be filled in as questions arise)

1.