

Skilaverkefni V (Erfðir og fjölvirkni / Inheritance and polymorphism)

In this assignment, you will create a 2D predator-prey simulation. In this simulation, the prey are **ants** and the predators are **bugs**. These critters live in a world composed of a 20×20 grid of cells. Only one critter may occupy a cell at a time. The grid is enclosed, so a critter is not allowed to move off the edges of the grid. Time is simulated in discrete steps. Each critter performs some action every time step. The behaviour of the world and the critters is further described below.

Ant behaviour:

Move: Every time step, randomly try to make one move UP, DOWN, LEFT, or RIGHT. If the cell in the selected direction is occupied or would move the ant off the grid, then the ant stays in the current cell.

Breed: If an ant survives for three time steps, then at the end of the third time step (i.e., after moving) the ant will breed. This is simulated by creating one new ant in an empty adjacent cell (found in the order UP, DOWN, LEFT, or RIGHT). If all adjacent cells are occupied, breeding cannot occur in the current time step and will be tried in the following time steps, until successful. When an offspring is produced, the ant cannot produce an offspring until three more time steps have elapsed.

Ants move after the bugs.

Bug behaviour:

Move: Every time step, if there is an adjacent cell occupied by an ant then the bug will move to that cell and eat the ant (which is then removed from the grid of cells). Otherwise, the bug moves according to the same rules as the ant. Note that a bug cannot eat other bugs.

Breed: If a bug survives for eight time steps, then at the end of the time step it will spawn off a new bug in the same manner as the ant.

Starve: If a bug has not eaten an ant within the last three time steps, then at the end of the third time step it will starve and die. The bug should then be removed from the grid of cells.

Bugs move before the ants.

World behaviour:

The world is initialized with 5 bugs and 100 ants. After each time step, the user is prompted to press a character to move to the next time step. You should see a cyclical pattern between the population of predators and prey, although random perturbations may lead to the elimination of one or both species.

The provided template World includes code to present this simulation graphically using ASCII characters of “o” for an ant, “X” for a bug, and “.” for an empty cell.

Implementation:

A main program to drive the simulation is given in `main.cpp`. You are also supplied with the interface and partial implementation of the classes `World` and `Organism`.

Your task is to:

- Complete the implementation of the class `World`. This class represents the world in which the organisms live. It contains, for example, operations for carrying out the simulation.
- Complete the implementation of the class `Organism`. This class encapsulates basic data and operations common to both ants and bugs.
- Write the interface and implementation for the class `Ant`. This class is a subclass of `Organism` and encapsulates data and operations for ants.
- Write the interface and implementation for the class `Bug`. This class is a subclass of `Organism` and encapsulates data and operations for bugs.

Notes about the implementation:

- Study the overall design of the given code before you start filling into the missing parts.
- The world has information about which organism occupies which cell (through its two-dimensional grid array) and each organism has information about where it is located in the world (through its (x,y) coordinates).
- Use the constants and enumerations, defined in `World.h` and `Organism.h`, in your solution.
- When developing your program, insert `cout` statements that print out various actions, like when an ant or a bug breeds (and at which coordinates), when a bug eats an ant (at which coordinate), and when a bug (at which coordinates) starves. This will make it easier for you to see if something is wrong, but make sure that you comment these statements out when submitting your solution.
- The `Organism` class is an *abstract* class because it contains so-called *pure* virtual functions. The textbook does not cover this concept, but you can find information on the web, e.g. <http://www.exforsys.com/tutorials/c-plus-plus/c-pure-virtual-function-and-base-class.html>

- You should view the grid in the `World` class as a two-dimensional cartesian coordinates system (http://en.wikipedia.org/wiki/Cartesian_coordinate_system). A point (x,y) in this system then denotes column x and row y. When moving upwards/downwards (changing row), y is increased/decreased, and when moving to the right/left (changing column), x is increased/decreased. When the grid is displayed, the given function (in the `World` class) prints out the grid „downwards“, meaning that that the first row (row 0) is at the top of the screen and the last row (row `WORLD_SIZE-1`) is at the bottom.

Submitting:

To submit your solution in Mooshak, create a .zip file containing:

Organism.h, Organism.cpp, World.h, World.cpp, Ant.h, Ant.cpp, Bug.h, Bug.cpp