ТЕМА: НАСЛЕДОВАНИЕ и ПОЛИМОРФИЗМ

ЗАДАЧА 1: Реализовать модель «Лисы и Зайцы», которая работает по следующим правилам:

1. Действие происходит на поле размера NxM

2. Действие проходит в течение заданного количества ходов.

3. Начальное количество зверей - количество Лис и количество Зайцев - задано.

В каждый момент времени для любого зверя известны:

- ✓ координаты нахождения на поле;
- ✓ направление движения (следующего шага): вверх, вниз, вправо, влево;
- ✓ стабильность - через сколько ходов зверь меняет направление движения
- ✓ возраст - сколько ходов зверь уже существует. Начальное значение возраста = 0.

4. Каждый ход включает несколько этапов:

1. Движение

Каждая Лиса перемещается на 2 клетки в выбранном направлении.

Каждый заяц перемещается на 1 клетку в выбранном направлении.

Если зверь выходит за границу поля, то он появляется на противоположной его стороне.

После движения зверь меняет направление своего движения на следующее по часовой стрелке, если этого требует его Стабильность.

2. Питание

Если после этапа Движение Лиса и Заяц оказываются на одной клетке, то Лиса съедает Зайца.

Тогда результат: Заяц погибает, Лиса насыщается.

Если Лис и Зайцев в клетке несколько, то первая же Лиса съедает всех Зайцев.

Первыми едят старшие Лисы, что это такое объяснено далее.

3. Старение

Возраст каждого живого зверя увеличивается на 1

4. Размножение

Лиса, съевшая как минимум двух Зайцев, размножается, порождая еще одну лису в той точке, где она находится. После этого Лиса снова голодна (для следующего размножения ей нужно будет съесть еще двух зайцев).

Зайцы размножаются дважды за жизнь, достигнув возраста 5 и 10 соответственно.

При рождении: Новый зверь движется в том же направлении, что и родитель.
Стабильность совпадает с родительской.
Возраст равен нулю.

5. Вымирание

На этом этапе умирают звери, достигшие максимального срока жизни.

Зайцы умирают, когда их возраст достиг 10, Лисы – 15.

Нумерация зверей:

На некоторых этапах важно, какой из двух зверей является старшим (например, при питании лисиц). Старшим считается зверь, родившийся в более ранний ход.

В случае совпадения хода рождения старшим считается тот, кто:

1) Был первее записан во входных данных (для зверей родившихся в нулевой ход).

2) Родился от более старшего зверя (для остальных).

Реализация модели:

При реализации требуется:

1. Создать иерархию зверей с базовым абстрактным классом Animal и наследниками Fox и Rabbit

2. Создать класс Model, позволяющий реализовывать модели с заданным количеством зверей, расположенных в указанных координатах, запускать процесс на заданное количество ходов.

3. Реализовать печать состояния модели после очередного хода следующим образом:

a. Печатается поле размера NxM

b. В каждой клетке стоит либо:

- Символ *, если там сейчас нет зверей
- Число Z, если там Z зайцев
- Число -Z, если там сейчас Z лисиц

Входные данные:

В первой строке входного файла указаны: значения N, M - размеры поля, K – количество ходов.

Во второй строке указаны значения R и F – начальное количество зайцев и лис соответственно.

В следующих R строках описываются зайцы с помощью 4-х значений x, y, d, s ,

где x, y – координаты на поле (точка с координатами {0,0} – верхний левый угол поля),

d – начальное направление (направления кодируются 0 (север), 1 (восток), 2 (юг), 3(запад)),

s – стабильность зверя.

В следующих F строках описываются Лисы (аналогично).

Выходные данные:

В выходные данные записать состояние модели после заданного количества ходов.

2 Примера входных и выходных данных:

input.txt	output.txt
4 4 3 2 1 0 1 1 3 2 2 0 2 1 0 1 2	***1 ***1 *-1** ****
2 3 5 1 2	*-1 -12

0 1 1 5 1 2 0 10 0 1 0 2	**

Рекомендации (следовать рекомендациям необязательно):

1. Лучше использовать списки, но можно и вектор Vector (далее приведены фрагменты с применением Vector), например:

```
#include <vector>
…
vector<Fox> masF;
vector<Rabbit> masR;
```

2. Структура классов

class Animal

{

Общие поля для всех зверей

Конструкторы: по умолчанию, с параметрами

void move(int N)

  {

    change(N); - меняет координаты в зависимости от направления

  }

Геттеры, Сеттеры для полей

};

class Rabbit:

конструкторы: по умолчанию, с параметрами

метод change(int N)

метод changeD() - изменяет направление

age1() - изменяет возраст

class Fox:

данные: возраст и еда

Геттеры, Сеттеры для полей

конструкторы: по умолчанию, с параметрами

метод change(int N)

метод changeD() - изменяет направление

age1() - изменяет возраст

class Model

{

данные: размеры поля, количество ходов, количество зайцев, количество лис

Геттеры, Сеттеры для полей

vector<Fox> masF;

vector<Rabbit> masR;

int **mas;

Конструкторы: с параметрами (в теле конструктора: создать двумерный динамический массив по заданным N и M и инициализировать нулями), копирования

Конструктор копирования, пример:

```
Model (Model const & that) : N(that.N), mas(new int*[that.N])
    {
        for (int i = 0; i < this->N; ++i)
        {this->mas[i] = new int[N];
            for (int j = 0; j < this->N; ++j)
             this->mas[i][j] = that.mas[i][j];}
    }
```

Метод добавления зайцев, например:

```
void addR (int x, int y, int s, int dir)
  {
    masR.push_back (Rabbit(x, y, s, dir));
    mas[x][y]+ =1;
  }
```

Метод добавления лис

Метод Шаг, в методе

//Движение зайцев

//Движение лис и питание

Возможный фрагмент питания:

```
            for (int i = 0; i<R; ++i)
               if(masR[i].get_x() == masF[i].get_x() && masR[i].get_y() == masF[i].get_y())
               {
                  auto iter = masR.cbegin();
                  mas[masR[i].get_x()][masR[i].get_y()] = 0;
                  masR.erase(iter + i);
                  --R;
                  masF[i].set_food(masF[i].get_food()+1);
               }

//Смена направления
//Старение
//Размножение
      for(…)
      {
         if (masR[j].get_age() ==5 || masR[j].get_age() == 10)
         {
            masR.push_back(Rabbit(masR[j].get_x(), masR[j].get_y(), masR[j].get_s(), masR[j].get_dir()));
            ++mas[masR[j].get_x()][masR[j].get_y()];
            ++R;
         }
      }
      for(int j = 0; j<masW.size(); ++j)
      {
         if …
         {
            masF.push_back(Fox(masF[j].get_x(), masF[j].get_y(), masF[j].get_s(), masF[j].get_dir(),
masF[j].get_old()));
            --mas[masF[j].get_x()][masF[j].get_y()];
            masW[j].set_food(0);
         }
      }
//Вымирание
```

Пример для зайцев:

```
for(int j = 0; j<masR.size(); ++j)
      {
         if (masR[j].get_age() ==10)
         {
            auto iter = masR.cbegin();
            masR.erase(iter + j);
            --mas[masR[j].get_x()][masR[j].get_y()];
            --j;
            --R;
         }

      }
```

Метод вывода результата (выходной файл)

Деструктор

```
int main()
{
```

Ввод заданных значений для построения модели

Объект класса Model

```
//vector<Rabbit> masR;
```

Ввод данных для зайцев и добавление зайцев (через объект класс Модель)

Ввод данных для лис и добавление лис (через объект класс Модель)

```
//ХОД
For (int i = 0; i<K; ++i)
{
   M.step(i);   // M - объект класса модель
}
M.write();
return 0;
}
```