Cours MOOC EPFL d'initiation à la programmation illustré en C++

Third assignment
Loops

J.-C. Chappelier & J. Sam

1 Exercise 1 — Number Games

1.1 Description

Assume we play the following "game" with some (positive) integer:

- if can be divided by 3, let's add 4 to it;
- if it can't be divided by 3 but can be divided by 4, let's divide it by 2;
- otherwise, let's substract 1 from it.

We repeat the above operations till we reach 0. Concretely, starting with an integer n_0 and applying the rules once, we get a new number n_1. Then, if n_1 is not null, we apply the above rules to it to get a new integer n_2; and proceed so on till we get a integer n_k which is null.

For instance, if we start from 7, then we get: 6, 10, 9, 13, 12, 16, 8, 4, 2, 1 et 0. The number k of repetitions is thus 11 in this case.

As another example, if we start from 1, we directly get 0, and the number of repetitions is thus $k = 1$.

Now, we ask you to write a program that, for every integer between two integer asked to the user, prints the number of required iteration to get 0 from it.

For instance, if the user asks to see the results between 1 and 7, the program will display:

```
1 -> 1
2 -> 2
```

```
3 -> 12
4 -> 3
5 -> 4
6 -> 10
7 -> 11
```

In the above example, 1 and 7 (in the first column) are the boundaries provided as input by the user. The second columns displays the number of repetitions to get 0 with the game described above starting with the number in the first column; for instance, 11 repetitions starting from 7.

As another example, if the user asks for integers between 99 and 100, the program will display:

```
99 -> 18
100 -> 17
```

Notice: to test if an integer `n` can be divided by `p`, you just need to test whether `n % p` is zero.

1.2 Instructions

Download the source code available at the course webpage and complete it according to the instructions below.

WARNING: you should modify neither the beginning nor the end of the provided file. It's thus mandatory to proceed as follows:

1. save the downloaded file as `suite.cc` or `suite.cpp`;
2. write your code between these two provided comments:

```
/*****************************************************
 * Compléter le code à partir d'ici
 *****************************************************/

/*****************************************
 * Ne rien modifier après cette ligne.
 *****************************************/
```

3. save and test your program to be sure that it works properly; try for instance the values used in the example given below;
4. upload the modified file (still named `suite.cc` or `suite.cpp`) in "OUTPUT submission" (not in "Additional"!).

2 Exercise 2 — Preys and predators

2.1 Introduction

The purpose of this exercise is to simulate the evolution over time of a population of predators (foxes here) and a population of preys (rabbits here), using a quite simple model so-called "Lotka-Volterra model".

Download the source code available at the course webpage and complete it according to the instructions below.

WARNING: you should modify neither the beginning nor the end of the provided file. It's thus mandatory to proceed as follows:

1. save the downloaded file as `predproie.cc` or `predproie.cpp`;
2. write your code between these two provided comments:

```
/*****************************************************
 * Compléter le code à partir d'ici
 *****************************************************/

/*****************************************
 * Ne rien modifier après cette ligne.
 *****************************************/
```

3. save and test your program to be sure that it works properly; try for instance the values used in the example given below;
4. upload the modified file (still named `predproie.cc` or `predproie.cpp`) in "OUTPUT submission" (not in "Additional"!).

2.2 Initial Populations

In the first part of this exercise, you are asked to complete the program below:

```
// ===== PARTIE 1 =====
// Saisie des populations initiales
```

so that it:

1. asks the user to input a inital number of foxes (`renards_i`), not less than 2; the user will be asked to input that number again if it is not superior or equal to 2;

2. then asks the user to input a inital number of rabbits (`lapins_i`), not less than 5; there again the user will be asked to input it as long as it's incorrect.

To ensure correct format for the questions, we already wrote them in the provided code. You shall not change those lines (but use them).

Execution example

```
Combien de renards au départ (>= 2) ? 1
Combien de renards au départ (>= 2) ? 3
Combien de lapins au départ  (>= 5) ? 4
Combien de lapins au départ  (>= 5) ? 20
```

2.3 Evolution of the populations

We here wish to model the evolution of the fox and rabbit populations.
Complete the program below:

```
// ===== PARTIE 2 =====
// Première simulation
cout <<  endl << "***** Le taux d'attaque vaut "<< taux_attaque * 100 << "%" <<  endl;
```

so that it:

1. initialises the populations of foxes and rabbits according to the former user's input;

2. computes (see below) and displays the fox and rabbit populations from the first month to `duree` (so from 1 to 50), incrementing it 1 by 1.

The growth of the rabbits is such that the number of rabbits is each time multiply by
`(1.0 + taux_croissance_lapins - taux_attaque * nb_renards )`
where `nb_renards` is the number of foxes before the new computation below.

It means that the new number of rabbits is equal to the previous number of rabbits plus the rabbits that were born minus the rabbits that were killed by the foxes. The

number of rabbits killed depends on the number of foxes as the more foxes, the more eaten rabbits.

The growth of the foxes is such that their number is each time multiplied by
`(1.0 + taux_attaque * nb_lapins * taux_croissance_renards - taux_mortalite)`
where `nb_lapins` is the number of rabbits obtained by the former computation.

The number of foxes increases by the number of foxes borned minus the number of foxes that died. The number of newborn foxes depends on the number of rabbits because the foxes need to catch rabbits in order to feed themself, reproduce and feed their offspring.

The former formulas allow the number of foxes and rabbits to be negative. You thus have to add code that will set these numbers to zero whenever they become negative.

Notice: Both numbers of rabbits and foxes are represented as `double` so as to keep enough precision on the computation. Non-integer values could be interpreted as the uncertainty we have about the exact values of animals actually living.

Execution example

```
Combien de renards au départ (>= 2) ? 20
Combien de lapins au départ  (>= 5) ? 500

***** Le taux d'attaque vaut 1%
Après 1 mois, il y a 550 lapins et 18.8 renards
Après 2 mois, il y a 611.6 lapins et 17.75 renards
Après 3 mois, il y a 686.5 lapins et 16.84 renards
Après 4 mois, il y a 776.9 lapins et 16.08 renards
Après 5 mois, il y a 885 lapins et 15.47 renards
Après 6 mois, il y a 1014 lapins et 15.02 renards
Après 7 mois, il y a 1165 lapins et 14.74 renards
Après 8 mois, il y a 1343 lapins et 14.64 renards
Après 9 mois, il y a 1550 lapins et 14.75 renards
Après 10 mois, il y a 1786 lapins et 15.1 renards
Après 11 mois, il y a 2052 lapins et 15.75 renards
Après 12 mois, il y a 2345 lapins et 16.76 renards
Après 13 mois, il y a 2655 lapins et 18.23 renards
Après 14 mois, il y a 2968 lapins et 20.27 renards
Après 15 mois, il y a 3256 lapins et 23.06 renards
Après 16 mois, il y a 3482 lapins et 26.76 renards
Après 17 mois, il y a 3595 lapins et 31.54 renards
```

```
Après 18 mois, il y a 3540 lapins et 37.46 renards
Après 19 mois, il y a 3276 lapins et 44.32 renards
Après 20 mois, il y a 2807 lapins et 51.5 renards
Après 21 mois, il y a 2203 lapins et 57.91 renards
Après 22 mois, il y a 1588 lapins et 62.33 renards
Après 23 mois, il y a 1075 lapins et 64.02 renards
Après 24 mois, il y a 709.1 lapins et 63.12 renards
Après 25 mois, il y a 474.3 lapins et 60.39 renards
Après 26 mois, il y a 330.1 lapins et 56.64 renards
Après 27 mois, il y a 242.2 lapins et 52.47 renards
Après 28 mois, il y a 187.8 lapins et 48.24 renards
Après 29 mois, il y a 153.5 lapins et 44.14 renards
Après 30 mois, il y a 131.8 lapins et 40.27 renards
Après 31 mois, il y a 118.3 lapins et 36.67 renards
Après 32 mois, il y a 110.4 lapins et 33.35 renards
Après 33 mois, il y a 106.7 lapins et 30.31 renards
Après 34 mois, il y a 106.4 lapins et 27.54 renards
Après 35 mois, il y a 109 lapins et 25.02 renards
Après 36 mois, il y a 114.4 lapins et 22.73 renards
Après 37 mois, il y a 122.7 lapins et 20.67 renards
Après 38 mois, il y a 134.2 lapins et 18.8 renards
Après 39 mois, il y a 149.2 lapins et 17.13 renards
Après 40 mois, il y a 168.4 lapins et 15.62 renards
Après 41 mois, il y a 192.6 lapins et 14.27 renards
Après 42 mois, il y a 222.9 lapins et 13.06 renards
Après 43 mois, il y a 260.7 lapins et 11.99 renards
Après 44 mois, il y a 307.7 lapins et 11.04 renards
Après 45 mois, il y a 366 lapins et 10.21 renards
Après 46 mois, il y a 438.5 lapins et 9.484 renards
Après 47 mois, il y a 528.4 lapins et 8.868 renards
Après 48 mois, il y a 640.1 lapins et 8.356 renards
Après 49 mois, il y a 778.6 lapins et 7.948 renards
Après 50 mois, il y a 950.3 lapins et 7.648 renards
```

Notice: If you want to observe interesting evolution of the populations, the number of rabbits should be way superior than the number of foxes. Otherwise, the foxes are killing all the rabbits before they had time to reproduce. Furthermore, avoid too small populations at the beginning. Ten foxes and 1000 rabbits or 20 foxes and 500 rabbits are interesting examples of stable populations' dynamics.

2.4 Effect of the attack rate

We want now to simplify the display in order to not display the evolution of the two populations at each time `t` but only the final populations after 50 months. In addition, we want alert messages to indicate whether the specie was close from extinction or if it has disappeared. Then, by using this simpler display, we will evaluate the effect of the attack rate on the two populations dynamic.

Now complete the program below:

```
// ===== PARTIE 3 =====
// Variation du taux d'attaque
cout << endl;
```

2.4.1 Alert message

1. In order to simplify the display, we won't display the population evolution anymore at every time `t`. Copy and paste your entire loop `for` from part 2 with the equations that simulate the population evolution in part 3 and then take the display command out of the loop. The aim is that the program display only *at the end of the simulation* the number of remaining foxes and rabbits. Be careful to indicate the right number of months. Modify your code so that alert messages are displayed *once the simulation is done* (look at the execution example below):

   - if, during the simulation, the number of foxes or of rabbits go below 5, an alert message should indicate:

     `Les ... ont été en voie d'extinction`

     (with `...` being either `renards` or `lapins`).
   - if during the simulation the number of foxes or rabbits go below 5 but increases back above 5 afterword, the previous message should be displayed (as we get below 5) but followed by

     `mais la population est remontée ! Ouf !`

   If the population of foxes or rabbits go below two, we are assuming that the specie can't reproduce aynmore. In this situation, the number of foxes or rabbits should be set to zero (during the simulation) and the following message displayed (at the end of the simulation):

   `et les ... ont disparus :-(`

(with `...` being either `renards` or `lapins`).

2. If during the simulation, **none** of the previously described event happened, display at the end of the simulation

   ```
   Les lapins et les renards ont des populations stables.
   ```

3. Modify also your code so as to stop the simulation when the numbers of both foxes and rabbits are both zero. Look for example at what happened in the example below for attack rate of 5%: the simulation stopped at month 25 rather than the usual 50.

2.4.2 Attack rate modification

Once you have set up the display, modify your program so that it:

1. asks the user to enter the initial attack rate (in percent) between 0.5 and 6; this initial rate should be asked again if the conditions are not satisfied; notice that here what ask this rate in percents; it should thus be divided by 100 to be used in the former formulas;

2. asks the user to enter the final attack rate (in percent) between the previously entered initial rate and 6; this final rate should be asked again if the conditions are not satisfied;

3. run the simulation for all the attack rates from the initial rate to the final rate (not included) by incrementing the rate of 1% each time. For each loop iteration display the corresponding attack rate

   ```
   ***** Le taux d'attaque vaut ...%
   ```

Execution example (20 foxes and 500 rabbits at the beginning)

```
taux d'attaque au départ en % (entre 0.5 et 6) ? 7
taux d'attaque au départ en % (entre 0.5 et 6) ? 1
taux d'attaque à la fin  en % (entre 1 et 6) ? 0.5
taux d'attaque à la fin  en % (entre 1 et 6) ? 6

***** Le taux d'attaque vaut 1%
Après 50 mois, il y a 950.3 lapins et 7.648 renards
Les lapins et les renards ont des populations stables.
```

```
***** Le taux d'attaque vaut 2%
Après 50 mois, il y a 205 lapins et 10.1 renards
Les lapins et les renards ont des populations stables.

***** Le taux d'attaque vaut 3%
Après 50 mois, il y a 0 lapins et 16.9 renards
Les renards ont été en voie d'extinction
mais la population est remontée ! Ouf !
Les lapins ont été en voie d'extinction
et les lapins ont disparu :-(

***** Le taux d'attaque vaut 4%
Après 50 mois, il y a 9774 lapins et 0 renards
Les renards ont été en voie d'extinction
et les renards ont disparu :-(
Les lapins ont été en voie d'extinction
mais la population est remontée ! Ouf !

***** Le taux d'attaque vaut 5%
Après 25 mois, il y a 0 lapins et 0 renards
Les renards ont été en voie d'extinction
et les renards ont disparu :-(
Les lapins ont été en voie d'extinction
et les lapins ont disparu :-(
```