C++ Pool - d06

IOStream, String and objects

Koalab koala@epitech.eu

Abstract: This document is the subject for d06

Contents

Chapter I

Basic rules

- If you do half the exercises because you have comprehension problems, it's okay, it happens. But it you do half the exercises because you're lazy, and leave at 2PM, you WILL have problems. Do NOT tempt the devil.
- Every function implemented in a header, or unprotected header, means 0 to the exercise.
- Every output goes to the standard output, and will be ended by a newline, unless specified otherwise.
- The imposed filenames must be followed TO THE LETTER, as well as class, functions and method names.
- Remember : You're coding in C++ now, and not in C. Therefore, the following functions are FORBIDDEN, and their use will be punished by a -42, no questions asked :
    - *alloc
    - *printf
    - free
- Generally, Files associated to a class will always be `CLASS_NAME.h` and `CLASS_NAME.cpp` (class name can be in lower case if applicable).
- Turn-in directories are ex00, ex01, ..., exN
- Any use of `"friend"` will result in the grade -42, `no questions asked` .
- You must read the examples thoroughly. They can contain requirements that are not obvious in the exercise's description.

- As much as you are allowed to the C++ tools you are using since the beginning of the swimming pool, you are not authorized to use any external library.
- You are asked to turn in an important number of classes. However, these classes are pretty short. Slackers not accepted!
- Read each exercise FULLY before starting it!
- USE YOUR BRAIN, Please!

EXERCISES COMPILATION :

- The Koalinette compiles your code with the following flags : `-W -Wall -Werror` .
- Each of your includes must be able to be included independently from others. Includes must contains every other includes they are depending on.
- Each include file will be included in the main of the correction.
- Please note that none of your files must contain the `main` function except if it is explicitly asked. We will use our own `main` function to compile and test your code.
- The exercise description can be modified until 4h before the final turn in time!
- The compilator to use is g++ !
- You must use file streams to solve some exercises. The video lecture on streams details how IOStream is working. There are numerous ways to implement it, within which file streams. This particular type of stream substitutes the file handling using the `C` mode based on `*open` and `close` .

I.1 Google+ Group

In order to facilitate communication between us (teachers, assistants and all of you students) a g+ group has been created.

You will find it here :

Group.

I invite you to use it for your questions, your reactions and everything else about the pool.

Chapter II

Exercise 0

	Exercise : 00	points : 2
IOStream : my_cat		
Turn-in directory: `(piscine_cpp_d06)/ex00`		
Compiler: `g++`	Compilation flags: `-Wall -Wextra -Werror`	
Makefile: `Yes`	Rules: `all, clean, fclean, re`	
Files to turn in : `Makefile, and your program files`		
Remarks : `For this exercise, please turn in your complete program, as well as your main`		
Forbidden functions : `*alloc, free, *printf, open, fopen - key word using`		

Your Makefile must generate a binary file named : `my_cat`

You must write a simplified `CAT(1)` command. This command must accept one or several files as parameters, and does not have to handle the standard input.

On error, (file not found, `permission denied` , etc.), you must print on the error output :

```
 my_cat: <file>: No such file or directory
```

With `file` being replaced with the name of the file that can't be read.

If no parameter is passed to your `my_cat` , you must print on the standard output :

```
my_cat: Usage : ./my_cat file [...]
```

Chapter III

Exercise 1

	Exercise : 01 points : 2
Temperature Conversion	
Turn-in directory: `(piscine_cpp_d06)/ex01`	
Compiler: `g++`	Compilation flags: `-W -Wall -Werror`
Makefile: `Yes`	Rules: `all, clean, fclean, re`
Files to turn in : `Makefile, and your program files`	
Remarks : `For this exercise, please turn in your complete program, as well as your main`	
Forbidden functions : `*alloc, free, *printf, open, fopen, *scanf - key word using`	

Your Makefile must generate a binary file named : `my_convert_temp`

The purpose of this exercise is to write a program that will convert temperatures from the `Celsius` scale to the `Fahrenheit` scale , as well as the other way around.
The conversion formula is : (We do agree, this formula is not the right one!) :

```
Celsius = 5.0 / 9.0 * ( Fahrenheit - 32 )
```

The standard input of your program must be : (separated by one or several spaces) :

- a temperature
- a scale

Example :

```
>./my_convert_temp
-10 Celsius
          14.000      Fahrenheit
>./my_convert_temp
46.400 Fahrenheit
           8.000         Celsius
```

Results must be displayed within two columns, aligned on the right with a 16 padding and a 1000th precision.

Chapter IV

Exercise 2

	Exercise : 02	points : 3
Hospital : the patient		
Turn-in directory: (piscine_cpp_d06)/ex02		
Compiler: g++	Compilation flags: -W -Wall -Werror	
Makefile: No	Rules: n/a	
Files to turn in : sickkoala.h, sickkoala.cpp		
Remarks : n/a		
Forbidden functions : *alloc, free, *printf, open, fopen - key word using		

You are now working on a simulation of your dear Koalas' health. In order to start, you need patients. Therefore you need to create the class SickKoala. Here are the informations that will allow you to code this class :

- They can't be instantiated without a name `string`
- Following their destruction, the standard output must display :

```
Mr.[name]: Kreooogg !! Je suis gueriiii !
```

- They have a member function `poke` which takes no parameter that does not return anything. When called, this member function display :

```
Mr.[name]: Gooeeeeerrk !! :'(
```

- They have a member function `takeDrug` with a `string` as a parameter and returns `true` if the `string` matches one of the following possibilities :

- ◦ `mars` (not case sensitive). The function displays :

```
Mr.[name]: Mars, et ca kreog !
```

- ◦ `Buronzand` (case sensitive). The function displays :

```
Mr.[name]: Et la fatigue a fait son temps !
```

In every other case, the function returns `false` and displays :

```
Mr.[name]: Goerkreog !
```

- Sometimes, Sickkoalas delirium when their fevers are too high. To simulate this, Sickkoalas have the member function `overDrive` that returns nothing, and that takes a `string` as a parameter. This member function displays the string passed in parameter, preceded by `"Mr.[name]:"` , within which all occurences of `"Kreog !"` are replaced by `"1337 !"` .
  For example, the string :

```
Kreog ! Ca boume ?
```

Will become :

```
Mr.[name]: 1337 ! Ca boume ?
```

For each output of this exercise, all [name] must be replaced by the name of the SickKoala.

Each display must be followed by a carriage return unless otherwise mentioned.

Chapter V

Exercise 3

	Exercise : 03	points : 3
Hospital : The nurse		
Turn-in directory: `(piscine_cpp_d06)/ex03`		
Compiler: `g++`	Compilation flags: `-W -Wall -Werror`	
Makefile: `No`	Rules: `n/a`	
Files to turn in : `koalanurse.h, koalanurse.cpp, sickkoala.h, sickkoala.cpp`		
Remarks : `n/a`		
Forbidden functions : `*alloc, free, *printf, open, fopen - key word using`		

We henceforth have patients. We now need a nurse to take care of them. Therefore you are now coding the nurse for the koala : The KoalaNurse.

Here are the informations you have in order to create the KoalaNurse :

- Each KoalaNurse has a numerical identifier (ID) which needs to be indicated when the object is created. (It is not possible to create a Nurse without specifying her ID.)
- For the destruction of the Nurse, she'll express her relief in the following way :

```
Nurse [ID]: Enfin un peu de repos !
```

- The nurse can gives drugs to the patient, thanks to the member function `giveDrug` with the following parameters :
  - a `string` (Drug)
  - pointer to the patient.

This member function does not return anything.
Then, the nurse gives medication to the patient.

- The nurse can read the doctor's report thanks to the member function `readReport` that takes a file name `string` as a parameter.

  - The file name is built from the sick Koala's name followed by the extension `.report` .
  - The file contains the name of the drug to give to the patient.

  The member function returns the name of the drug (`string`) and displays on the standard output :

```
 Nurse [ID]: Kreog ! Il faut donner un [drugName] a Mr.[patientName] !
```

  If the `.report` file doesn't exist or is not valid, nothing must be displayed and the return is an empty `string` .

- The nurse can clock in thanks to the member function `timeCheck` that takes no parameter and doesn't return anything.
  The nurse calls this member function when she starts working and when she stops working (as she is a very diligent worker.)
  When she clocks in at the start of her job, she displays :

```
   Nurse [ID]: Je commence le travail !
```

  When she stops working :

```
   Nurse [ID]: Je rentre dans ma foret d'eucalyptus !
```

  It is up to you to figure out a way to find out when she starts and when she stops working. By default, when the program starts, the nurse is not yet working. The KoalaNurse being very diligent, she accepts any job. Even outside the hospital.
  Only a call to the member function `timeCheck` allows to switch the KoalaNurse's working status: if she is not working, she starts to work; if she is working, she stops.

`In this exercise, you will replace [ID] by the KoalaNurse ID for the display.`

Chapter VI

Exercise 4

	Exercise : 04	points : 3
Hospital : The Doctor		
Turn-in directory: `(piscine_cpp_d06)/ex04`		
Compiler: `g++`	Compilation flags: `-W -Wall -Werror`	
Makefile: `No`	Rules: `n/a`	
Files to turn in : `koalanurse.h, koalanurse.cpp, sickkoala.h, sickkoala.cpp, koaladoctor.h, koaladoctor.cpp`		
Remarks : `n/a`		
Forbidden functions : `*alloc, free, *printf, open, fopen, srand, srandom - key word using`		

You first need to modify your previous classes.

- Add a member function `getName` to the class `SickKoala` with no parameters and returning the name of the patient (`string`)

We now have patients and nurses taking care of them. We need the doctor to give instructions to nurses. You will code a simulation of the doctor with the class `KoalaDoctor` .

This is what we know about `KoalaDoctor` :

- He must be instantiated with a name (`string`). During the construction he must display on the standard output :

```
  Dr.[name]: Je suis le Dr.[name] ! Comment Kreoggez-vous ?
```

- He can diagnose a patient thanks to the member function `diagnose` that takes a pointer to the patient to diagnose as parameter.
  This member function displays on the standard output :

```
Dr.[name]: Alors qu'est-ce qui vous goerk Mr.[patientName] ?
```

Then it calls the member function `poke` of the SickKoala.
The doctor then writes a report for nurses, in a file named `[patientName].report`
This file contains the name of the drug to give to the patient. The name will be randomly picked from the following list :

- `mars`
- `Buronzand`
- `Viagra`
- `Extasy`
- `Feuille d'eucalyptus`

On this purpose you must use `random() % 5` , on the previous list following the given order. The `srandom` function will be called in the main of the correction.

- The KoalaDoctor clocks in with the member function `timeCheck` (which neither accept parameters nor return anything) when he starts working and when he stops working, because he is a diligent worker.
  When he starts working, he displays :

```
Dr.[name]: Je commence le travail !
```

When he stops working, he displays :

```
Dr.[name]: Je rentre dans ma foret d'eucalyptus !
```

The KoalaDoctor being very diligent, he accepts any job. Even outside the hospital.

`For this exercise, for outputs to create, all` `[name]` need to be replaced by the name of the `KoalaDoctor` that creates the output (name with which you initialized your `KoalaDoctor` instance.) Regarding the `[patientName]` , it is necessary to replace those with the name of the `SickKoala` that the `KoalaDoctor` instance is currently handling.

Chapter VII

Exercise 5

	Exercise : 05	points : 3
Hospital : A way to handle all of that		
Turn-in directory: `(piscine_cpp_d06)/ex05`		
Compiler: `g++`	Compilation flags: `-W -Wall -Werror`	
Makefile: `No`	Rules: `n/a`	
Files to turn in : `koalanurse.h, koalanurse.cpp, sickkoala.h, sickkoala.cpp, koaladoctor.h, koaladoctor.cpp, sickkoalalist.h, sickkoalalist.cpp, koalanurselist.h, koalanurselist.cpp, koaladoctorlist.cpp, koaladoctorlist.h`		
Remarks : `At this point, recursive programing can save a good part of developing time...`		
Forbidden functions : `*alloc, free, *printf, open, fopen, srand, srandom - key word using`		

For this exercise, it is necessary to modify the following classes: `KoalaNurse` and `KoalaDoctor` .

- Add a member function `getID` to the class `KoalaNurse` . This function takes no parameter and returns an `int` .
- Add a member function `getName` to the class `KoalaDoctor` . This function takes no parameter and returns a `string` .

We now need to look over all these people working together in harmony. It is necessary to be able to handle several patients, doctors and/or nurses at the same time. For this, it is necessary to code a list for each of these category.

`For this exercise, a node of the list is a List* object.`

First we need to build lists.

- The class `SickKoalaList`

  - When constructed, takes a pointer on `SickKoala` . This pointer can be `NULL`
  - Posesses a member function `isEnd` which does not take any parameter and that returns a boolean to `true` if the `SickKoalaList` is the last one of the list.
  - Posesses a member function `append` which takes a pointer on `SickKoalaList` as a parameter and does not return anything. The node passed as a parameter is added at the end of the chained list.
  - Posesses a member function `getFromName` which takes a `string` as a parameter and returns the pointer on the first `SickKoala` whose name matches the `string` passed as parameter.
  - Posesses a member function `remove` which takes a pointer on `SickKoalaList` as a parameter and removes the `SickKoalaList` matching with this pointer from the list.
    This member function returns the pointer on the first node of the linked list.
  - Posesses a member function `removeFromName` which takes a `string` as a parameter and removes from the list the first SickKoala whose name matches the `string` passed as a parameter.
    This member function returns the pointer on the first node of the list.
  - Posesses a member function `dump` with no parameters and that does not return anything. This member function displays the name of all the SickKoalas belonging to the list, respecting the sorting order of the list (begin → end) :

    ```
        Liste des patients : [name1], [name2], ..., [nameX].
    ```

    If an element is missing, the name to display is `[NULL]` .
  - Posesses a member function `getContent` with no parameters and returns a pointer on the element within the actual node of the linked list (`SickKoalaList`).
  - Posesses a member function `getNext` with no parameters and that returns the pointer on the next node of the linked link. If there's no next node, then it returns 0 (`NULL`)

- KoalaNurseList Class:

- When constructing this class, it takes a pointer on a `KoalaNurse`. This pointer can be `NULL`.
- Posesses a member function `isEnd` which does not take any parameter and returns a boolean equal to `true` if the `KoalaNurseList` is the last node of the list.
- Posesses a member function `append` which takes a pointer on `KoalaNurseList` as a parameter and does not return anything. The node passed as a parameter is added to the end of the chained list.
- Posesses a member function `getFromID` which takes an `int` as a parameter and returns the pointer on the first `KoalaNurse` whose number matches the `int` passed as a parameter.
- Posesses a member function `remove` which takes a pointer on `KoalaNurseList` as a parameter and removes the `KoalaNurseList` matching with this pointer from the list.
  This member function returns the pointer on the first node of the list.
- Posesses a member function `removeFromID` which accepts an `integer` as a parameter. This member function removes the first KoalaNurse whose ID matches the integer passed as a parameter from the list.

  This member function returns a pointer on the first node of the list.
- Posesses a member function `dump` which does not take any parameter and does not return anything. This member function displays the name of all the KoalaNurse belonging to the list respecting its sorting order (begin→ end) :

```
    Liste des infirmieres : [name1], [name2], ..., [nameX].
```

  If an element is missing, the name to display is `[NULL]`.

- KoalaDoctorList Class :
  - When constructing this class, it takes a pointer on a KoalaDoctor . This pointer can be NULL .
  - Posesses a member function isEnd which does not take any parameter and returns a boolean equal to true if the KoalaDoctorList is the last node of the list.
  - Posesses a member function append which takes a pointer on KoalaDoctorList as a parameter and does not return anything. The node passed as a parameter is added to the end of the chained list.

- Posesses a member function getFromName that takes a string as a parameter and returns the first KoalaDoctor whose name matches the string passed as a parameter.
- Posesses a member function remove that takes a pointer on KoalaDoctorList as a parameter and removes from the list the KoalaDoctorList matching with this pointer. This member function returns a the pointer on the first node of the list.
- Posesses a member function removeFromName which takes a string as a parameter. This member function removes from the list the first KoalaDoctor whose name matches the string passed as a parameter. This member function returns the pointer on the first node of the list.
- Posesses a member function dump that does not take any parameter and that does not return anything. This member function displays the name of all the KoalaDoctors belonging to the list respecting its sorting order (begin -> end) :

```
   Liste des medecins : [name1], [name2], ..., [nameX].
```

If an element is missing the name to display is [NULL] .

Chapter VIII

Exercise 6

	Exercise : 06	points : 4
The Hospital		
Turn-in directory: `(piscine_cpp_d06)/ex06`		
Compiler: `g++`	Compilation flags: `-W -Wall -Werror`	
Makefile: `No`	Rules: `n/a`	
Files to turn in : `koalanurse.h, koalanurse.cpp, sickkoala.h, sickkoala.cpp, koaladoctor.h, koaladoctor.cpp, sickkoalalist.h, sickkoalalist.cpp, koalanurselist.h, koalanurselist.cpp, koaladoctorlist.h, koaladoctorlist.cpp, hopital.h, hopital.cpp`		
Remarks : `n/a`		
Forbidden functions : `*alloc, free, *printf, open, fopen - mot-clé using`		

It is possible to manage several patients, nurses and doctors. We henceforth can move on and manage the whole Hospital !
Now you will code without any help !
You must deduce the member functions of the Hospital from the main of the test as described underneath.

The Hospital must distribute the work between doctors and nurses.
For this exercise you may have to modify existing classes. You are responsible for these modifications as long as they respect previous exercises requirements/descriptions !

This is a main for a test with the expected output :

```
#include <iostream>
#include <string>
#include <cstdlib>

#include "sickkoala.h"
#include "koalanurse.h"
#include "koaladoctor.h"
#include "sickkoalalist.h"
#include "koalanurselist.h"
#include "koaladoctorlist.h"
#include "hopital.h"

int main()
{
    srandom(42);

    KoalaDoctor cox("Cox");
    KoalaDoctor house("House");
    KoalaDoctor tired("Boudur-Oulot");
    KoalaDoctorList doc1(&cox);
    KoalaDoctorList doc2(&house);
    KoalaDoctorList doc3(&tired);

    KoalaNurse a(1);
    KoalaNurse b(2);
    KoalaNurseList nurse1(&a);
    KoalaNurseList nurse2(&b);

    SickKoala cancer("Ganepar");
    SickKoala gangrene("Scarface");
    SickKoala rougeole("RedFace");
    SickKoala variole("Varia");
    SickKoala fracture("Falter");
    SickKoalaList sick1(&cancer);
    SickKoalaList sick2(&gangrene);
    SickKoalaList sick3(&rougeole);
    SickKoalaList sick4(&variole);
    SickKoalaList sick5(&fracture);

    {
        Hospital stAnne;

        stAnne.addDoctor(&doc1);
        stAnne.addDoctor(&doc2);
        stAnne.addDoctor(&doc3);
        stAnne.addSick(&sick1);
        stAnne.addSick(&sick2);
        stAnne.addSick(&sick3);
        stAnne.addSick(&sick4);
        stAnne.addSick(&sick5);
        stAnne.addNurse(&nurse1);
        stAnne.addNurse(&nurse2);

        stAnne.addSick(&sick4);

        stAnne.run();
    }

    if (nurse1.isEnd() && sick1.isEnd() && doc1.isEnd())
        std::cout << "Lists cleaned up." << std::endl;
    else
        std::cerr << "You fail ! Boo !" << std::endl;

    return (0);
}
```

Expected output :

```
Dr.Cox: Je suis le Dr.Cox ! Comment Kreoggez-vous ?
Dr.House: Je suis le Dr.House ! Comment Kreoggez-vous ?
Dr.Boudur-Oulot: Je suis le Dr.Boudur-Oulot ! Comment Kreoggez-vous ?
[HOSPITAL] Doctor Cox just arrived !
Dr.Cox: Je commence le travail !
[HOSPITAL] Doctor House just arrived !
Dr.House: Je commence le travail !
[HOSPITAL] Doctor Boudur-Oulot just arrived !
Dr.Boudur-Oulot: Je commence le travail !
[HOSPITAL] Patient Ganepar just arrived !
[HOSPITAL] Patient Scarface just arrived !
[HOSPITAL] Patient RedFace just arrived !
[HOSPITAL] Patient Varia just arrived !
[HOSPITAL] Patient Falter just arrived !
[HOSPITAL] Nurse 1 just arrived !
Nurse 1: Je commence le travail !
[HOSPITAL] Nurse 2 just arrived !
Nurse 2: Je commence le travail !
[HOSPITAL] Debut du travail avec :
Liste des medecins : Cox, House, Boudur-Oulot.
Liste des infirmieres : 1, 2.
Liste des patients : Ganepar, Scarface, RedFace, Varia, Falter.

Dr.Cox: Alors qu'est-ce qui vous goerk Mr.Ganepar ?
Mr.Ganepar: Gooeeeeerrk !! :'(
Nurse 1: Kreog ! Il faut donner un Buronzand a Mr.Ganepar !
Mr.Ganepar: Et la fatigue a fait son temps !
Dr.House: Alors qu'est-ce qui vous goerk Mr.Scarface ?
Mr.Scarface: Gooeeeeerrk !! :'(
Nurse 2: Kreog ! Il faut donner un mars a Mr.Scarface !
Mr.Scarface: Mars, et ca kreog !
Dr.Boudur-Oulot: Alors qu'est-ce qui vous goerk Mr.RedFace ?
Mr.RedFace: Gooeeeeerrk !! :'(
Nurse 1: Kreog ! Il faut donner un Buronzand a Mr.RedFace !
Mr.RedFace: Et la fatigue a fait son temps !
Dr.Cox: Alors qu'est-ce qui vous goerk Mr.Varia ?
Mr.Varia: Gooeeeeerrk !! :'(
Nurse 2: Kreog ! Il faut donner un Buronzand a Mr.Varia !
Mr.Varia: Et la fatigue a fait son temps !
Dr.House: Alors qu'est-ce qui vous goerk Mr.Falter ?
Mr.Falter: Gooeeeeerrk !! :'(
Nurse 1: Kreog ! Il faut donner un Viagra a Mr.Falter !
Mr.Falter: Goerkreog !
Nurse 1: Je rentre dans ma foret d'eucalyptus !
Nurse 2: Je rentre dans ma foret d'eucalyptus !
Dr.Cox: Je rentre dans ma foret d'eucalyptus !
Dr.House: Je rentre dans ma foret d'eucalyptus !
Dr.Boudur-Oulot: Je rentre dans ma foret d'eucalyptus !
Lists cleaned up.
```

```
Mr.Falter: Kreooogg !! Je suis gueriiii !
Mr.Varia: Kreooogg !! Je suis gueriiii !
Mr.RedFace: Kreooogg !! Je suis gueriiii !
Mr.Scarface: Kreooogg !! Je suis gueriiii !
Mr.Ganepar: Kreooogg !! Je suis gueriiii !
Nurse 2: Enfin un peu de repos !
Nurse 1: Enfin un peu de repos !
```