KOALA

B3- C++ Pool

B-PAV-242

Day 06

IOStream, String and Objects

v1.6.5

Day 06

IOStream, String and Objects

repository name:	cpp_d06
repository rights:	ramassage-tek
language:	C++
group size:	1

- Your repository must contain the totality of your source files, but no useless files (binary, temp files, obj files,...).
- All the bonus files (including a potential specific Makefile) should be in a directory named *bonus*.
- Error messages have to be written on the error output, and the program should then exit with the 84 error code (0 if there is no error).

If you do half the exercises because you have comprehension problems, it's okay, it happens. But if 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, leads to 0 to the exercise.
Every class must possess a constructor and a destructor.

Every output goes to the standard output, and will be ended by a newline, unless specified otherwise.

Any use of "friend" will result in the grade -42 no questions asked.

To avoid compilation problems, please include necessary files within your headers.

Please note that none of your files must contain the main function except when explicitly asked. We will use our own main function to compile and test your code.

Exercise 00

my_cat

repository subdir:	/ex00
compilation:	g++ -Wall -Wextra -Werror -std=c++03
files to turn in:	Makefile with 'all', 'clean', 'fclean' and 're' rules and program files
forbidden functions:	*alloc, free, *printf, open, fopen - using key word
points:	2

Please turn in your complete program, as well as your main.

Your Makefile must generate a binary file named **my_cat**.

You must write a simplified **CAT(1)** command. This command must take one, or several, files as parameters, and will not handle the standard input.

In the event of an error (file not found, permission denied etc.), you must print the following on the error output:

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

file is replaced with the name of the file that can't be read.

If no parameters are given to your **my_cat**, you must print the following on the standard output:

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

Exercise 01

Temperature Conversion

repository subdir:	/ex01
compilation:	g++ -Wall -Wextra -Werror -std=c++03
files to turn in:	Makefile with 'all', 'clean', 'fclean' and 're' rules, and program files
forbidden functions:	*alloc, free, *printf, open, fopen, *scanf - using key word
points:	2

Please turn in your complete program, as well as your main.

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, and vice versa.
The conversion formula is as follows (yes, we agree, it's not the most accurate one!):

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

Your program will read from the standard input, separated by one or more spaces, a temperature and a scale.

For example:

```
Terminal
~/B-PAV-242> ./my_convert_temp
-10 Celsius
14.000 Fahrenheit
```

```
Terminal
~/B-PAV-242> ./my_convert_temp
46.400 Fahrenheit
8.000 Celsius
```

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

Exercise 02

Hospital - The Patient

repository subdir:	/ex02
compilation:	g++ -Wall -Wextra -Werror -std=c++03
files to turn in:	sickkoala.[h,cpp]
forbidden functions:	*alloc, free, *printf, open, fopen - using key word
points:	3

You will now be working on a simulation of your dear Koalas' health. In order to begin, you need patients. Therefore you need to create the **SickKoala** class.
Here is the information that will let you code the class:

- They cannot be instantiated without a **string** name.
- Following their destruction, the standard output must display the following:

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

- They have a **poke** member function, which takes no parameters and returns nothing. When called, it displays:

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

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

  - **mars** (not case-sensitive), displays:

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

  - **Buronzand** (case-sensitive), displays:

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

- In any other case, the function returns **false** and displays:

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

- Sometimes SickKoalas can get delirious when their fevers are too high. In order to simulate this, SickKoalas have the **overDrive** member function, which returns nothing and takes a **string** as parameter. This member function displays the string passed as parameter preceded by **"Mr. [name]:"**, and with all occurrences of **"Kreog!"** are replaced with **"1337!"**.

  For example, the string **"Kreog! Ca boume ?** will become **Mr. [name]: 1337! Ca boume ?**.

For each of this exercise's outputs, all **[name]**s must be replaced with the SickKoala's name.

Every output must be followed by a newline, unless otherwise specified.

Exercise 03

Hospital - The Nurse

repository subdir:	/ex03
compilation:	g++ -Wall -Wextra -Werror -std=c++03
files to turn in:	koalanurse.[h,cpp], sickkoala.[h,cpp]
forbidden functions:	*alloc, free, *printf, open, fopen - using key word
points:	3

Now that we have patients, we need a nurse to take care of them.
Therefore, you will now be coding the koalas' nurse: The KoalaNurse.
Here is the necessary information to create the KoalaNurse:

- Each KoalaNurse has a numerical identifier (ID), which must be provided when the object is created. (It is not possible to create a Nurse without specifying its ID.)
- When the Nurse is destroyed, it'll express its relief as follows:

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

- The nurse can give drugs to the patient, thanks to the **giveDrug** member function, which has the following parameters: a string (Drug) and a pointer to the patient.
  This member function does not return anything, and then the nurse gives the patient their medication.

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

  - The file is built from the sick Koala's name, followed by the **.report** extension.
  - The file contains the name of the drug to be given to the patient.
  - The member function returns the name of the drug (**string**) and displays the following on the standard output:

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

  If the **.report** file doesn't exist, or if it isn't valid, nothing should be displayed and the return value will be an empty **string**.

- The nurse can clock-in thanks to the **timeCheck** member function, which takes no parameters and doesn't return anything. The nurse calls this member function when it starts and stops working (it is very diligent).

  When it clocks-in at the start of its job, it displays:

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

  When it stops working:

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

  It is up to you to figure out when it starts and stops working. By default, when the program starts, the nurse isn't working yet. As the KoalaNurse is very diligent, it will accept any job, even outside the hospital. Only a call to the **timeCheck** member function can switch the KoalaNurse's working status: if it is not working, it will start, and vice versa.

You will replace **[ID]** with the KoalaNurse ID for the display.

Exercise 04

Hospital - The Doctor

repository subdir:	/ex04
compilation:	g++ -Wall -Wextra -Werror -std=c++03
files to turn in:	koalanurse.[h,cpp], sickkoala.[h,cpp], koaladoctor.[h,cpp]
forbidden functions:	*alloc, free, *printf, open, fopen, srand, srandom - using key word
points:	3

First, modify your previous classes by adding a **getName** member function to the **SickKoala** class, that takes no parameter and returns the patient's name (**string**).

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

Here's what we know about **KoalaDoctor**:

- He must be instantiated with a **string** name. During the construction, he must display the following on the standard output:

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

- He can diagnose a patient thanks to the **diagnose** member function, which takes a pointer to the patient to be diagnosed as parameter.
  This member function displays the following on the standard output:

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

  It then calls the SickKoala's **poke** member function. The doctor writes a report for the nurses, in a file named **[patientName].report**. This file contains the name of the drug to be given to the patient. The name will be randomly chosen from the following list:
  - mars
  - Buronzand
  - Viagra
  - Extasy
  - Feuille d'eucalyptus

  For this purpose, you must use **random() %5**, on the previous list and follow the given order. The **srandom** function will be called in the correction's main.

- The KoalaDoctor clocks-in with the **timeCheck** member function, which neither accepts parameters nor returns anything, when it starts working and vice versa.
  When it starts working, it displays:

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

  When it stops working, it displays:

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

  As the KoalaDoctor is very diligent, it accepts any job, even outside the hospital.

All **[name]**s must be replaced with the name of the KoalaDoctor that is creating the output (the name with which you intitialized your KoalaDoctor, for instance). As for the **[patientName]**, you must replace those with the name of the SickKoala that the KoalaDoctor instance is currently handling.

Exercise 05

Hospital - How to handle it all

repository subdir:	/ex05
compilation:	g++ -Wall -Wextra -Werror -std=c++03
files to turn in:	koalanurse.[h,cpp], sickkoala.[h,cpp], koaladoctor.[h,cpp], koalanurselist.[h,cpp], sickkoalalist.[h,cpp], koaladoctorlist.[h,cpp]
forbidden functions:	*alloc, free, *printf, open, fopen, srand, srandom - using key word
points:	3

At this point, recursive programming can save you a lot of development time...

You must modify the following classes for this exercise: **KoalaNurse** and **KoalaDoctor**.

- Add a **getID** member function to the **KoalaNurse** class. The function returns an int and takes no parameters.
- Add a **getName** member function to the **KoalaDoctor** class. The function returns a string and takes no parameters.

Now we need to take a look at 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. In order to do be able to do this, you must code a list for each category.

A list's node is a List* object.

First we need to build lists.

The SickKoalaList class:

- When constructed, takes a pointer to **SickKoala**. The pointer can be **NULL**.
- Has an **isEnd** member function, which does not take any parameter and returns a boolean that is true if the **SickKoalaList** is the last element of its list.
- Has an **append** member function, which takes a pointer to **SickKoalaList** as parameter and does not return anything. The node passed as parameter is added to the end of the linked list.
- Has a **getFromName** member function, which takes a string as parameter and returns the pointer to the first **SickKoala** whose name matches the string passed as parameter. If no such **SickKoala** is in the list, the function will return a null pointer.
- Has a **remove** member function, which takes a pointer to **SickKoalaList** as parameter and removes the **SickKoalaList** that matches the pointer to the list. This member function returns a pointer to the first node of the linked list.
- Has a **removeFromName** member function, which takes a string as parameter and removes the first **SickKoalaList** whose name matches the string passed as parameter from the list. This member function returns a pointer to the first node of the list.
- Has a **dump** member function, with no parameters, that does not return anything. This member function displays the name of all the **SickKoalas** in the list, complying with the list's sorting order (begin -> end):

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

If an element is missing, the name to be displayed is **[NULL]**.

- Has a **getContent** member function, with no parameters, that returns a pointer to the element within the current node of the list.
- Has a **getNext** member function, with no parameters, that returns a pointer to the list's next node. If there is no next node, then it returns **(NULL)**.

The KoalaNurseList Class:

- When constructed, takes a pointer to **KoalaNurse**. This pointer can be **NULL**.
- Has an **isEnd** member function, which does not take any parameters and returns a boolean that is true if the **KoalaNurseList** is the last element in the list one.
- Has an **append** member function, which takes a pointer to **KoalaNurseList** as parameter and does not return anything. The node passed as parameter is added to the end of the list.
- Has a **getFromID** member function, which takes an **int** as parameter and returns a pointer to the first **KoalaNurse** whose ID matches the **int** passed as parameter. If no such **KoalaNurse** is in the list, the function will return a null pointer.
- Has a **remove** member function, which takes a pointer to **KoalaNurseList** as parameter and removes the **KoalaNurseList** that matches the pointer from the list. This member function returns a pointer to the first node of the list.
- Has a **removeFromID** member function, which takes an **integer** as parameter. This member function removes the first **KoalaNurseList** whose ID matches the **integer** passed as parameter from the list. This member function returns a pointer to the first node of the list.
- Has a **dump** member function, with no parameters, that does not return anything. This member function displays the name of all the **KoalaNurses** in the list, complying with the list's sorting order (begin -> end):

```
Liste des infirmieres: [id1], [id2], ..., [idX].
```

If an element is missing, the name to be displayed is **[NULL]**.

The KoalaDoctorList Class:

- When constructed, takes a pointer to **KoalaDoctor**. This pointer can be **NULL**.
- Has an **isEnd** member function, which does not take any parameters and returns a boolean which is true if the **KoalaDoctorList** is the last element of its list.
- Has an **append** member function, which takes a pointer to **KoalaDoctorList** as parameter and does not return anything. The node passed as parameter is added to the end of the list.
- Has a **getFromName** member function, which takes a string as parameter and returns the first **KoalaDoctor** whose name matches the string passed as parameter from the list. If no such **KoalaDoctor** is in the list, the function will return a null pointer.
- Has a **remove** member function, which takes a pointer to **KoalaDoctorList** as parameter and removes the **KoalaDoctorList** that matches the pointer from the list.
  This member function returns a pointer to the first node of the list.
- Has a **removeFromName** member function, which takes a string as parameter. This member function removes the first **KoalaDoctorList** whose name matches the string passed as parameter from the list. This member function returns a pointer to the first node of the list.
- Has a **dump** member function, with no parameters, that does not return anything. This member function displays the name of all the **KoalaDoctors** in the list, complying with the list's sorting order (begin -> end):

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

If an element is missing, the name to be displayed is **[NULL]**.

Exercise 06

The Hospital

repository subdir:	/ex06
compilation:	g++ -Wall -Wextra -Werror -std=c++03
files to turn in:	koalanurse.[h,cpp], sickkoala.[h,cpp], koaladoctor.[h,cpp], koalanurselist.[h,cpp], sickkoalalist.[h,cpp], koaladoctorlist.[h,cpp], hospital.[h,cpp]
forbidden functions:	*alloc, free, *printf, open, fopen, srand, srandom - using key word
points:	4

Now you can manage several patients, nurses and doctors. So we can move on and manage the entire hospital!

Now you're going to code without any help.

You must deduce the hospital's member functions from the test main, as described below.

The hospital must distribute the work between the doctors and the nurses.

For this exercise, you might have to modify existing classes. You are responsible for these modifications, as long as they comply with the 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 "hospital.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 measles ("RedFace");
   SickKoala smallpox ("Varia");
   SickKoala fracture ("Falter");
   SickKoalaList sick1(&cancer);
```

```
    SickKoalaList sick2(&gangrene);
    SickKoalaList sick3(&measles);
    SickKoalaList sick4(&smallpox);
    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);
}
```

Next page is the expected output:

Terminal

```
~/B-PAV-242> ./day06
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: Kreoooogg!! Je suis gueriiii!
Mr. Varia: Kreoooogg!! Je suis gueriiii!
Mr. RedFace: Kreoooogg!! Je suis gueriiii!
Mr. Scarface: Kreoooogg!! Je suis gueriiii!
Mr. Ganepar: Kreoooogg!! Je suis gueriiii!
Nurse 2: Enfin un peu de repos!
Nurse 1: Enfin un peu de repos!
```