B3 - C++ Pool

B-PAV-242

Day 06

IOStream, String and objects

KOALA

42.0

Day 06

binary name:	no binary
group size:	1
repository name:	cpp_d06
repository rights:	ramassage-tek
language:	C++

- Your repository must contain the totality of your source files, but no useless files (binary, temp files, obj files,...).

General Setpoints

READ THESE CAREFULLY
You will have no possible excuse if you end up with a 0 because you didn't follow one of these.

If you do half the exercises because you have comprehension problems, it's okay, it happens. But if you do half the exercices because you're lazy, and leave at 2PM, you **WILL** have problems. Do not tempt the devil.

Read the examples CAREFULLY. They might require things that weren't mentioned in the subject...

All output goes to the standard output and must be ended with a newline character, unless specified otherwise.

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

Any use of the `friend` keyword will result in a -42

You are not allowed to use any library other than the C++ standard library.

It must be possible to include each of your header files independently from the others. Headers must include all their dependencies.

All your header files will be included in the correction `main`.

None of your files must contain a `main` function

THINK. Please.

THINK

T.H.I.N.K.! For Pony!

To avoid compilation problems during automated tests, please include all necessary files within your headers.

Please note that none of your files must contain a `main` function, unless specified otherwise. We will use our own `main` functions to compile and test your code.

This subject may be modified up to one hour before turn-in time!

Unit Tests

It is highly recommended to test your functions as you implement them. It is common practice to create and use what are called **unit tests**.

From now on, we expect you to write unit tests for your functions (when possible). To do so, please follow the instructions in the **"How to write Unit Tests"** document on the intranet, available here.

Create a directory named `tests`. For each of the classes you turn in, create a file in that directory named `tests-CLASS-NAME.cpp` containing all the tests needed to cover all of the class' possible cases (regular or irregular).

Here is a sample set of unit tests for the **string** class:

```
#include <criterion/criterion.h>

Test(string, default_value)
{
    std::string s;
    cr_assert_eq(s, "");
}

Test(string, assign)
{
    std::string s;

    s = "test";
    cr_assert_eq(s, "test");
}

Test(string, append)
{
    std::string s("test");

    s += "ing";
    cr_assert_eq(s, "testing");
}
```

Exercise 0 - IOStream

	Exercise: 00	points : 2
IOStream - my_cat		
Turn-in directory: `cpp_d06/ex00`		
Compiler: `g++`	Compilation flags: `-W -Wall -Wextra -Werror -std=c++14`	
Makefile: `Yes`	Rules: `all, clean, fclean, re`	
Files to turn in: `Makefile and your program files`		
Notes: `You must turn in your complete program, as well as your ‘main‘ function`		
Forbidden functions: `*alloc, free, *printf, open, fopen - ‘using namespace‘ keyword`		

Your `Makefile` must generate a `my_cat` executable.

You must write a simplified `cat(1)` command. Your executable will take one or several files as parameters, and does not need to handle the special case of the standard input.
Upon error (file not found, permission denied, etc.), you must write the following message to the error output:

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

`file` must be replaced with the name of the file for which the error was encountered.

If no parameter is passed to your program, you must write the following message to the standard output:

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

Exercise 1 - Temperature Conversion

	Exercise: 01	points : 2
Temperature Conversion		
Turn-in directory: `cpp_d06/ex01`		
Compiler: `g++`	Compilation flags: `-W -Wall -Wextra -Werror -std=c++14`	
Makefile: `Yes`	Rules: `all, clean, fclean, re`	
Files to turn in: `Makefile and your program files`		
Notes: `You must turn in your complete program, as well as your ‘main‘ function`		
Forbidden functions: `*alloc, free, *printf, open, fopen - ‘using namespace‘ keyword`		

Your `Makefile` must generate a `my_convert_temp` executable.

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 to use is the following (we know, it isn't the right one!):

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

Your program wiill read from its standard input (separated by one or more spaces):

- a temperature
- a scale

Example:

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

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

Exercise 2 - The Patient

	Exercise: 02	points : 3
Hospital: the patient		
Turn-in directory: cpp_d06/ex02		
Compiler: g++	Compilation flags: -W -Wall -Wextra -Werror -std=c++14	
Makefile: No	Rules: n/a	
Files to turn in: sickkoala.h, sickkoala.cpp		
Notes: None		
Forbidden functions: *alloc, free, *printf, open, fopen - ‘using namespace‘ keyword		

You are now working on a simulation of your dear Koalas' health. To get started, you'll need patients to treat. Therefore, it is time to create a `SickKoala` class. Here are the information you need to implement 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 !
```

- A `poke` member function taking no parameters or return value will display the following when called:

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

- A `takeDrug` function taking a `string` as parameter will return `true` if the `string` matches one of the following:
  - `mars` (not case sensitive). The function will then display:
    Mr.[name]: Mars, et ca kreog !
  - `Buronzand` (case sensitive). The function will then display:
    Mr.[name]: Et la fatigue a fait son temps !

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

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

- Sometimes, `SickKoalas` go crazy when their fever is too high. To simulate this, `SickKoalas` have an `overDrive` member function that returns nothing and takes a `string` as parameter. It displays the string passed as parameter, preceded by `"Mr.[name]:"`, within which all occurences of `"Kreog !"` are replaced by `"1337 !"`.
  For instance:

```
Kreog ! Ca boume ?
```

Will become:

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

For all outputs in this exercise, `[name]` must be replaced by the name of the `SickKoala`

EXERCISE 3 - THE NURSE

KOALA	Exercise: 03	points : 3
Hospital: the nurse		
Turn-in directory: `cpp_d06/ex03`		
Compiler: `g++`	Compilation flags: `-W -Wall -Wextra -Werror -std=c++14`	
Makefile: `No`	Rules: `n/a`	
Files to turn in: `koalanurse.h, koalanurse.cpp, sickkoala.h, sickkoala.cpp`		
Notes: `None`		
Forbidden functions: `*alloc, free, *printf, open, fopen - 'using namespace' keyword`		

Now that we have patients, we need a nurse to take care of them. You are now coding the nurse for the koala: The `KoalaNurse`.
Here is the information you need in order 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 her ID)
- When a `KoalaNurse` is destroyed, it'll express its relief like so:

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

- The nurse can give drugs to patients, through a `giveDrug` member function with the following parameters:
  - a `string` (Drug)
  - a pointer to the patient

This member function does not return anything. When it is called, the nurse gives medication to the patient.

- The nurse can read the doctor's report through a `readReport` member function that takes a filename `string` as parameter.
  - The filename is built from the sick Koala's name, followed by the `.report` extension.
  - The file contains the name of the drug to give to the patient.

This member function returns the name of the drug as a `string` and prints the following to 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 value must be an empty `string`.

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

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

When it stops working, it says:

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

It is up to you to figure out a way to find out when it starts and stops working. By default, when the program starts, the nurse is not working yet. The `KoalaNurse` being very diligent, it will take any job. Even outsided the hospital. Only a call the `timeCheck` member function lets the `KoalaNurse` change her working status: if it is not working, it starts to work; if it is working, it stops.

In this exercise, `[ID]` must be replaced with the `KoalaNurse`'s ID in any output

EXERCISE 4 - THE DOCTOR

KOALA	Exercise: 04	points : 3
Hospital: The Doctor		
Turn-in directory: cpp_d06/ex04		
Compiler: g++	Compilation flags: -W -Wall -Wextra -Werror -std=c++14	
Makefile: No	Rules: n/a	
Files to turn in: koaladoctor.h, koaladoctor.cpp, koalanurse.h, koalanurse.cpp, sickkoala.h, sickkoala.cpp		
Notes: None		
Forbidden functions: *alloc, free, *printf, open, fopen, srand, srandom - ‘using namespace‘ keyword		

Before we get started, let's modify your existing classes.

- Add a getName member function to the SickcKoala class, taking no parameters and returning the name of the patient as a string.

We now have patients and nurses taking care of them. We still need a doctor to give instructions to the nurses. You must implement a simulation of the doctor with the KoalaDoctor class.

Here's what we know about the KoalaDoctor:

- It must be instantiated with a name string. During construction, it will print the following to the standard output:

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

- It can diagnose patients using the diagnose member function that takes a pointer to the patient to diagnose as parameter. This member function prints the following to the standard output:

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

It then calls the poke member function 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 picked at random from the following list:

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

To do this, you must use `random()% 5` on the previous list, in the given order. The `srandom` function will be called by the correction `main`.

- The `KoalaDoctor` clocks in through a `timeCheck` member function, which takes no parameters and does not return anything, when it starts or stops working, as it is a diligent worker.
  When it starts working, it says:

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

When it stops working, it says:

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

The `KoalaDoctor` being very diligent, it will take any job. Even outside the hospital.

In this exercise, any occurence of `[name]` must be replaced with the name of the `KoalaDoctor`, and occurences of `[patientName]` must be replaced with the name of the `SickKoala` that is currently being treated.

Exercise 5 - Lists

	Exercise: 05	points : 3
Hospital: A way to manage all of that		
Turn-in directory: `cpp_d06/ex05`		
Compiler: `g++`	Compilation flags: `-W -Wall -Wextra -Werror -std=c++14`	
Makefile: `No`	Rules: `n/a`	
Files to turn in: `koaladoctor.h, koaladoctor.cpp, koalanurse.h, koalanurse.cpp, sickkoala.h, sickkoala.cpp, sickkoalalist.h, sickkoalalist.cpp, koalanurselist.h, koalanurselist.cpp, koaladoctorlist.h, koaladoctorlist.cpp`		
Notes: `Recursive programming can save you a lot of development time...`		
Forbidden functions: `*alloc, free, *printf, open, fopen, srand, srandom - ‘using namespace‘ keyword`		

Before we get started, modify your `KoalaNurse` and `KoalaDoctor` classes:

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

We now need to watch 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. To do so, it is time to code a list for each of these categories.

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

Implement the following classes:

- `SickKoalaList`:
  - Takes a pointer to a `SickKoala` as a constructor parameter. This pointer can be `NULL`.
  - Has an `isEnd` member function which takes no parameter and returns a boolean set to `true` if the `SickKoalaList` is the last node of its list.
  - Has an `append` member function which takes a pointer to a `SickKoalaList` as a 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 a parameter and returns a pointer to the first `SickKoala` in the list whose name matches that `string`.
  - Has a `remove` member function which takes a pointer to a `SickKoalaList` as a parameter and removes the `SickKoalaList` matching this pointer from the list. It returns a pointer to the first node of the list.

    - Has a `removeFromName` member function which takes a `string` as a parameter and removes the first `SickKoala` whose name matches that `string` from the list. It returns a pointer to the first node of the list.
    - Has a `getContent` member function which takes no parameter and returns a pointer to the element held in the current instance.
    - Has a `getNext` member function which takes no parameter and returns a pointer to the next node of the list. If there is no such node, the function returns `NULL`.
    - Has a `dump` member function which takes no parameter and does not return anything. It displays the name of all the `SickKoalas` in the list in order (begin -> end):

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

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

- `KoalaNurseList`:
    - Takes a pointer to a `KoalaNurse` as a constructor parameter. This pointer can be `NULL`.
    - Has an `isEnd` member function which takes no parameter and returns a boolean set to `true` if the `KoalaNurseList` is the last node of its list.
    - Has an `append` member function which takes a pointer to a `KoalaNurseList` as a parameter and does not return anything. The node passed as parameter is added to the end of the linked list.
    - Has a `getFromID` member function which takes an `int` as a parameter and returns a pointer to the first `KoalaNurse` in the list whose ID matches that `int`.
    - Has a `remove` member function which takes a pointer to a `KoalaNurseList` and removes the `KoalaNurseList` matching this pointer from the list. It returns a pointer to the first node of the list.
    - Has a `removeFromID` member function which takes an `int` as parameter and removes the first `KoalaNurse` whose ID matches that `int` from the list. It returns a pointer to the first node of the list.
    - Has a `dump` member function which takes no parameter and does not return anything. It displays the ID of all the `KoalaNurses` in the list in order (begin -> end):

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

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

- `KoalaDoctorList`:
    - Takes a pointer to a `KoalaDoctor` as a constructor parameter. This pointer can be `NULL`.
    - Has an `isEnd` member function which takes no parameter and returns a boolean set to `true` if the `KoalaDoctorList` is the last node of its list.
    - Has an `append` member function which takes a pointer to a `KoalaDoctorList` as a 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 a parameter and returns the first KoalaDoctor in the list whose name matches that string.
- Has a remove member function which takes a pointer to a KoalaDoctorList as a parameter and removes the KoalaDoctorList matching this pointer from the list. It returns a pointer to the first node of the list.
- Has a removeFromName member function which takes a string as a parameter and removes the first KoalaDoctor whose name matches that string from the list. It returns a pointer to the first node of the list.
- Has a dump member function which takes no parameter and does not return anything. It displays the name of all KoalaDoctors in the list in order (begin -> end):

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

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

Exercise 6 - The Hospital

	Exercise: 06	points : 4
The Hospital		
Turn-in directory: `cpp_d06/ex06`		
Compiler: `g++`	Compilation flags: `-W -Wall -Wextra -Werror -std=c++14`	
Makefile: `No`	Rules: `n/a`	
Files to turn in: `koaladoctor.h, koaladoctor.cpp, koalanurse.h, koalanurse.cpp, sickkoala.h, sickkoala.cpp, sickkoalalist.h, sickkoalalist.cpp, koalanurselist.h, koalanurselist.cpp, koaladoctorlist.h, koaladoctorlist.cpp, hopital.h, hopital.cpp`		
Notes: `None`		
Forbidden functions: `*alloc, free, *printf, open, fopen, srand, srandom - ‘using namespace‘ keyword`		

It is now possible to manage several patients, nurses and doctors. It is time to move on and manage the entire `Hospital`!
You will now code without any help. You must deduce the member functions of the `Hospital` based on the sample `main` function you will find below.

The `Hospital` must distribute 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 comply with the requirements and descriptions of the previous exercises!

```
#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);
}
```

main.cpp

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 !
```