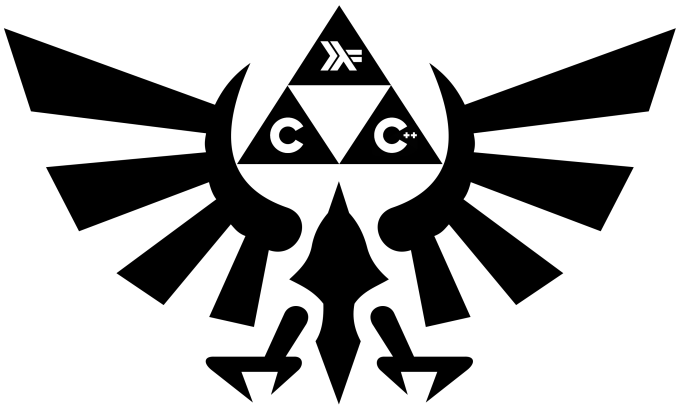
B3 - Paradigms Seminar

B-PDG-300

Day 06

IOStream, String and objects



2.2

Day 06

language: C++

- The totality of your source files, except all useless files (binary, temp files, obj files,...), must be included in your delivery.

All your exercises will be compiled with `g++` **and the** `-std=c++20 -Wall -Wextra -Werror` **flags**, unless specified otherwise.

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

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. It will include your header files.

For each exercise, **the files must be turned-in at the root of your repository unless specified otherwise**.

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

The `*alloc`, `free`, `*printf`, `open` and `fopen` functions, as well as the `using namespace` keyword, are forbidden in C++.
By the way, `friend` is forbidden too, as well as any library except the standard one.

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.

For them to be executed and evaluated, put a `Makefile` at the root of your directory with the `tests_run` rule as mentionned in the documentation linked above.
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 - Z IS (STILL) FOR ZORGLUB

Turn in: `Makefile`, `Z.cpp` in `ex00/`
Makefile rules: `all`, `clean`, `fclean`, `re`
Executable name: `Z`

Our Lord and Genuis Zorglub, master of Zorgland, requires your services.
Your brillant zorgmanizer character selection software must be converted to a much more evoluted programming language.
Rewrite your program in C++.

As we use the CamelCase naming convention in C++, be careful to correctly name your source files and binary.

Terminal

```
~/B-PDG-300> ./Z "0x42242112" | cat -e
z$
~/B-PDG-300> ./Z "invalid_ID" ; echo $?
z
0
```

EXERCISE 1 - MYCAT

Turn in: `Makefile`, `MyCat.cpp` in `ex01/`
Makefile rules: `all`, `clean`, `fclean`, `re`
Executable name: `MyCat`

Write a simplified `cat(1)` command.
Your executable must take zero (standard input), one or several files as parameters.
Upon error (file not found, permission denied, etc.), you must write the following message to the error output:

```
MyCat: file: No such file or directory
```

`file` must be replaced with the name of the file for which the error was encountered.
Your program must return 84 in case of error.

`std::ifstream`

Exercise 2 - Retard Unit Converter

Turn in: `Makefile`, `RetardUnitConverter.cpp` in `ex02/`
Makefile rules: `all`, `clean`, `fclean`, `re`
Executable name: RetardUnitConverter

The Fahrenheit scale was initially defined using the lowest temperature measured in Fahrenheit's home town Danzig in winter 1708 as 0°F and the rectal temperature of a horse as 100°F. On the other hand, the Celsius scale has been based on 0 °C for the freezing point of water and 100 °C for the boiling point of water at 1 atm pressure.

We're gonna create a Retard to Genius Unit Converter.

The purpose of this exercise is to write a program that converts 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 exact right one!):

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

Your program must read lines from its standard input, and converts the temperature from one scale to another. Each line contains two words separated by one or multiple spaces:

- temperature
- scale

Any additional input must be ignored.

`std::stringstream`, `<iomanip>`, `std::getline`

Terminal

```
~/B-PDG-300> ./RetardUnitConverter
-42 Celsius
         -43.600      Fahrenheit
84.21 Fahrenheit
          29.006         Celsius
   0.000     Celsius
          32.000      Fahrenheit
```

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

Exercise 3 - The Student

Turn in: `Student.hpp`, `Student.cpp` in `seminar/`

You are now working on a simulation of a C++ Seminar. To get started, you'll need students on the brink of despair. Therefore, it is time to **create a `Student` class**.

Here are the information you need to implement this class:

- they can't be instantiated without a name `std::string`
- at creation, they scream on the standard output:

```
Student [name]: I'm ready to learn C++.
```

- following their destruction, the standard output must display

```
Student [name]: Wow, much learning today, very smart, such C++.
```

- they have 100 energy points at creation, can't go below 0 and above 100
- a `learn` member function taking a text `std::string` as parameter that consumes 42 energy points and displays:

```
Student [name]: [text]
```

then it returns `true`. If the student doesn't have enough energy, no energy is consummed and every occurence of `"C++"` in the text is replaced by `"shit"`, then it returns `false`. For example: `"I'm learning C++!"` becomes `"I'm learning shit!"`.

- a `drink` member function taking a drink name `std::string` as parameter and depending on the drink name:
  - For `"Red Bull"`: display `"Student [name]: Red Bull gives you wings!"` and restore 32 energy points
  - For `"Monster"`: display `"Student [name]: Unleash The Beast!"` and restore 64 energy points
  - Otherwise: `Student [name]: ah, yes... enslaved moisture.` and restore 1 energy point

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

Exercise 4 - The Teaching Assistant

Turn in: `Assistant.hpp/cpp`, `Student.hpp/cpp` in `seminar/`

Now that we have students, we need a teaching assistant to take care of them. You are now coding **the `Assistant` class**.

Here is the information you need in order to create the `Assistant`:

- each `Assistant` is just a number on the payroll, it must be provided when the object is created. It is not possible to create a teaching assistant without specifying his ID
- when a `Assistant` enters the room, he says:

```
Assistant [ID]: 'morning everyone *sip coffee*
```

- when a `Assistant` leaves, it'll express its relief like so:

```
Assistant [ID]: see you tomorrow at 9.00 *sip coffee*
```

- the teaching assistant can give drinks to students, through a `giveDrink` member function with the following parameters: a pointer to a `Student` and a drink name `std::string`. This member function does not return anything. When it is called, the teaching assistant displays the following text then gives the drink to the student:

```
Assistant [ID]: drink this, [studentName] *sip coffee*
```

- the teaching assistant can read the Dean of Studies' report through a `readDrink` member function that takes a student name `std::string` as parameter:
  - The filename is built from the student's name, followed by the `.drink` extension.
  - The file contains the name of the drink to give to the student.

This member function returns the name of the drink as a `std::string`, removes the `.drink` file and prints the following to the standard output:

```
Assistant [ID]: [studentName] needs a [drink] *sip coffee*
```

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

- the teaching assistant can help a student through a `helpStudent` method that takes a pointer to a `Student` as parameter. The assistant must `readDrink` the report of the Dean of Studies and `giveDrink` the drink to the student. If there is no drink to give, he just prints:

```
Assistant [ID]: [studentName] seems fine *sip coffee*
```

You'll need to add a `getName` method to the `Student`.

- the teaching assistant can clock in thanks to a `timeCheck` member function that takes no parameter and doesn't return anything. When it clocks in at the start of his job, he says:

```
Assistant [ID]: Time to teach some serious business *sip coffee*
```

When it stops working, it says:

```
Assistant [ID]: Enough teaching for today *sip coffee*
```

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 teaching assistant is not working yet. The `Assistant` being very diligent, he will take any job, even outside the seminar. Only a call to the `timeCheck` member function lets the `Assistant` change his working status: if he is not working, he starts to work; if he is working, he stops.

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

Exercise 5 - The Dean of Studies

Turn in: `Dean.hpp/cpp`, `Assistant.hpp/cpp`, `Student.hpp/cpp` in `seminar/`

We now have students and teaching assistants taking care of them. We still need a Dean of Studies to give instructions to the teaching assistant. **Implement a simulation of the Dean of Studies** with the `Dean` class.

Here's what we know about the `Dean`:

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

```
Dean [name]: I'm Dean [name]! How do you do, fellow kids?
```

- when destroyed, the `Dean` must display:

```
Dean [name]: Time to go home.
```

- it can teach to a student using the `teachStudent` member function that takes a pointer to the `Student` to talk to and a lesson as `std::string` as parameter. He will attempt to teach his lesson to the `Student` using his `learn` method. In case of failure, he displays the following text and write a report:

```
Dean [name]: All work and no play makes [studentName] a dull student.
```

The dean then writes a report for the teaching assistant, in a file named `[studentName].drink`. This file contains the name of the drink to give to the student. The name will be picked at random from the following list:

```
* "Cristaline"
* "Monster"
* "Evian"
* "Red Bull"
* "Sierra Springs"
```

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

- the `Dean` clocks in through a `timeCheck` member function, which takes no parameters and does not return anything. When it starts working, it says:

```
Dean [name]: Where is everyone?
```

When it stops working, it says:

```
Dean [name]: Don't forget to close the windows when you leave.
```

The `Dean` being very diligent, it will take any job. Even outside the seminar.

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

Exercise 6 - The Seminar

Turn in: `Dean.hpp/cpp`, `Assistant.hpp/cpp`, `Student.hpp/cpp`, `Seminar.hpp/cpp` in `seminar/`

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

The `Seminar` must distribute work between the dean and assistants. 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!

```
int main(void)
{
    std::srand(42);

    Dean        dean_thaynam("Thay-Nam");
    Dean        dean_devoille("Devoille");
    Assistant   ass_42(42);
    Assistant   ass_24(24);
    Assistant   ass_2077(2077);
    Student     stud_jennifer("Jennifer");
    Student     stud_brian("Brian");
    Student     stud_kevin("Kevin");
    Student     stud_dwayne("Dwayne");
    Student     stud_priscilla("Priscilla");
    Student     stud_stewie("Stewie");

    {
        Seminar seminar;

        seminar.run();

        seminar.addDean(&dean_thaynam);
        seminar.addDean(&dean_devoille);
        seminar.addAssistant(&ass_42);
        seminar.addAssistant(&ass_24);
        seminar.addAssistant(&ass_42);
        seminar.addAssistant(&ass_2077);
        seminar.addStudent(&stud_jennifer);
        seminar.addStudent(&stud_brian);
        seminar.addStudent(&stud_kevin);
        seminar.addStudent(&stud_dwayne);
        seminar.addStudent(&stud_brian);
        seminar.addStudent(&stud_priscilla);
        seminar.addStudent(&stud_stewie);

        seminar.run();
    }

    return 0;
}
```

Terminal

```
~/B-PDG-300> ./a.out
Dean Thay-Nam: I'm Dean Thay-Nam! How do you do, fellow kids?
Dean Devoille: I'm Dean Devoille! How do you do, fellow kids?
Assistant 42: 'morning everyone *sip coffee*
Assistant 24: 'morning everyone *sip coffee*
Assistant 2077: 'morning everyone *sip coffee*
Student Jennifer: I'm ready to learn C++.
Student Brian: I'm ready to learn C++.
Student Kevin: I'm ready to learn C++.
Student Dwayne: I'm ready to learn C++.
Student Priscilla: I'm ready to learn C++.
Student Stewie: I'm ready to learn C++.
Seminar: A C++ seminar needs at least one Dean of Studies, one Assistant and one
Student.
Seminar: Dean Thay-Nam is here.
Seminar: There can only be one Dean of Studies.
Seminar: Assistant 42 joined the pedagogical team.
Seminar: Assistant 24 joined the pedagogical team.
Seminar: Assistant 42 is already registered.
Seminar: There is only room for two Teaching Assistants.
Seminar: Student Jennifer joined the seminar.
Seminar: Student Brian joined the seminar.
Seminar: Student Kevin joined the seminar.
Seminar: Student Dwayne joined the seminar.
Seminar: Student Brian is already registered.
Seminar: Student Priscilla joined the seminar.
Seminar: There is only room for five Students.
Seminar: Begining 6th day of seminar.
Dean of Studies: Thay-Nam
Teaching assistants: 42, 24
Students: Jennifer, Brian, Kevin, Dwayne, Priscilla
Dean Thay-Nam: Where is everyone?
Assistant 42: Time to teach some serious business *sip coffee*
Assistant 24: Time to teach some serious business *sip coffee*
Student Jennifer: I'm learning C++!
Assistant 42: Jennifer seems fine *sip coffee*
Student Brian: I'm learning C++!
Assistant 24: Brian seems fine *sip coffee*
Student Kevin: I'm learning C++!
Assistant 42: Kevin seems fine *sip coffee*
Student Dwayne: I'm learning C++!
Assistant 24: Dwayne seems fine *sip coffee*
Student Priscilla: I'm learning C++!
Assistant 42: Priscilla seems fine *sip coffee*
Student Jennifer: I'm learning C++!
```

Terminal

```
Assistant 24: Jennifer seems fine *sip coffee*
Student Brian: I'm learning C++!
Assistant 42: Brian seems fine *sip coffee*
Student Kevin: I'm learning C++!
Assistant 24: Kevin seems fine *sip coffee*
Student Dwayne: I'm learning C++!
Assistant 42: Dwayne seems fine *sip coffee*
Student Priscilla: I'm learning C++!
Assistant 24: Priscilla seems fine *sip coffee*
Student Jennifer: I'm learning shit!
Dean Thay-Nam: All work and no play makes Jennifer a dull student.
Assistant 42: Jennifer needs a Monster *sip coffee*
Assistant 42: drink this, Jennifer *sip coffee*
Student Jennifer: Unleash The Beast!
Student Brian: I'm learning shit!
Dean Thay-Nam: All work and no play makes Brian a dull student.
Assistant 24: Brian needs a Cristaline *sip coffee*
Assistant 24: drink this, Brian *sip coffee*
Student Brian: ah, yes... enslaved moisture.
Student Kevin: I'm learning shit!
Dean Thay-Nam: All work and no play makes Kevin a dull student.
Assistant 42: Kevin needs a Monster *sip coffee*
Assistant 42: drink this, Kevin *sip coffee*
Student Kevin: Unleash The Beast!
Student Dwayne: I'm learning shit!
Dean Thay-Nam: All work and no play makes Dwayne a dull student.
Assistant 24: Dwayne needs a Monster *sip coffee*
Assistant 24: drink this, Dwayne *sip coffee*
Student Dwayne: Unleash The Beast!
Student Priscilla: I'm learning shit!
Dean Thay-Nam: All work and no play makes Priscilla a dull student.
Assistant 42: Priscilla needs a Evian *sip coffee*
Assistant 42: drink this, Priscilla *sip coffee*
Student Priscilla: ah, yes... enslaved moisture.
Student Jennifer: I'm learning C++!
Assistant 24: Jennifer seems fine *sip coffee*
Student Brian: I'm learning shit!
Dean Thay-Nam: All work and no play makes Brian a dull student.
Assistant 42: Brian needs a Red Bull *sip coffee*
Assistant 42: drink this, Brian *sip coffee*
Student Brian: Red Bull gives you wings!
Student Kevin: I'm learning C++!
Assistant 24: Kevin seems fine *sip coffee*
Student Dwayne: I'm learning C++!
Assistant 42: Dwayne seems fine *sip coffee*
```

Terminal

```
Student Priscilla: I'm learning shit!
Dean Thay-Nam: All work and no play makes Priscilla a dull student.
Assistant 24: Priscilla needs a Monster *sip coffee*
Assistant 24: drink this, Priscilla *sip coffee*
Student Priscilla: Unleash The Beast!
Student Jennifer: I'm learning shit!
Dean Thay-Nam: All work and no play makes Jennifer a dull student.
Assistant 42: Jennifer needs a Cristaline *sip coffee*
Assistant 42: drink this, Jennifer *sip coffee*
Student Jennifer: ah, yes... enslaved moisture.
Student Brian: I'm learning C++!
Assistant 24: Brian seems fine *sip coffee*
Student Kevin: I'm learning shit!
Dean Thay-Nam: All work and no play makes Kevin a dull student.
Assistant 42: Kevin needs a Cristaline *sip coffee*
Assistant 42: drink this, Kevin *sip coffee*
Student Kevin: ah, yes... enslaved moisture.
Student Dwayne: I'm learning shit!
Dean Thay-Nam: All work and no play makes Dwayne a dull student.
Assistant 24: Dwayne needs a Red Bull *sip coffee*
Assistant 24: drink this, Dwayne *sip coffee*
Student Dwayne: Red Bull gives you wings!
Student Priscilla: I'm learning C++!
Assistant 42: Priscilla seems fine *sip coffee*
Dean Thay-Nam: Don't forget to close the windows when you leave.
Assistant 42: Enough teaching for today *sip coffee*
Assistant 24: Enough teaching for today *sip coffee*
Student Stewie: Wow, much learning today, very smart, such C++.
Student Priscilla: Wow, much learning today, very smart, such C++.
Student Dwayne: Wow, much learning today, very smart, such C++.
Student Kevin: Wow, much learning today, very smart, such C++.
Student Brian: Wow, much learning today, very smart, such C++.
Student Jennifer: Wow, much learning today, very smart, such C++.
Assistant 2077: see you tomorrow at 9.00 *sip coffee*
Assistant 24: see you tomorrow at 9.00 *sip coffee*
Assistant 42: see you tomorrow at 9.00 *sip coffee*
Dean Devoille: Time to go home.
Dean Thay-Nam: Time to go home.
```

The `main` function is provided in `ex06.cpp`, and the expected output in `ex06.txt`.