B3 - C++ Pool

B-CPP-300

Day 09

Kreog's Quest

1.2

Day 09

binary name:	no binary
group size:	1
repository name:	cpp_d09
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,...).

All your exercises will be compiled with `g++` **and the** `-W -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 in a separate directory called **exXX** where XX is the exercise number (for instance `ex01`), unless specified otherwise.

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

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.

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.

Create a directory named `tests`. For each of the functions you turn in, create a file in that directory named `tests-FUNCTION_NAME.c` containing all the tests needed to cover all of the exercise's 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 -1 - C

Compiler: `gcc`
Turn in: `Exo-1.h, Exo-1.c`

This short exercise in C will serve to introduce you to **C++ inheritance**.
This will help you understand what inheritance means, and what it implies when applied to the C language.
It will also give you a sense of how inheritance is implemented under the hood in C++…

I recommend reading the example output very carefully.
You must deduce your own output from it.

This first exercise is coded in C

By Odin, read the exercise TO THE END before you start coding!

Create an `s_cthulhu` structure, and a `cthulhu_t` alias to it.
This type is composed of an `m_power int` and a `m_name char *`.

This type has a set of functions associated with it:#

```
cthulhu_t *NewCthulhu();
```

Creates a new `cthulhu_t` object, initializes it and returns a pointer to it.
When a cthulhu is initialized, its `m_name` field is set to *"Cthulhu"* and its `m_power` field to 42.

```
void PrintPower(cthulhu_t *this);
```

prints the instance's power to the standard output.

```
void Attack(cthulhu_t *this);
```

checks whether the instance has enough energy.
Requires 42 energy points.
Consumes 42 energy points and attacks.

```
void Sleeping(cthulhu_t *this);
```

recharges cthulhu and increases its energy by 42,000 points.

Create an `s_koala` structure and a `koala_t` alias to it.
It is composed of a `cthulhu_t` called `m_parent` and an `m_isALegend` `char`.

Read the previous sentence once again.

Read it one more time.

This type has a set of functions associcated with it:

```
koala_t *NewKoala(char *name, char _isALegend);
```

creates a new `koala_t` object, initializes it and returns a pointer to it.

```
void Eat(koala_t *this);
```

feeds the koala and increases its energy by 42 points.

The `NewCthulhu` and `NewKoala` functions use the following initalization functions:

```
static void KoalaInitializer(koala_t *this, char *_name, char _isALegend);
static void CthulhuInitializer(cthulhu_t *this);
```

Use this sample `main` function to compile your code and display the following output.

Yes, that implies you have to use your brain!

```
int main()
{
    koala_t *_LKoala = NewKoala("Legend", 1);
    koala_t *_NLKoala = NewKoala("NotLegend", 0);
    cthulhu_t *_cthulhu = NewCthulhu();

    printf("----Start----n");
    PrintPower(_cthulhu);
    PrintPower(&_LKoala->m_parent);
    PrintPower(&_NLKoala->m_parent);
    Attack(_cthulhu);
    Attack(&_LKoala->m_parent);
    Attack(&_NLKoala->m_parent);
    Eat(_NLKoala);
    Attack(_cthulhu);
    Sleeping(_cthulhu);
    PrintPower(_cthulhu);
    Attack(&_NLKoala->m_parent);
    return 0;
}
```

Terminal

```
~/B-CPP-300> ./a.out | cat -e
----$
Building Cthulhu$
Building Legend$
----$
Building Cthulhu$
Building NotLegend$
----$
Building Cthulhu$
----Start----$
Power => 42$
Power => 42$
Power => 0$
Cthulhu attacks and destroys the city$
Legend attacks and destroys the city$
NotLegend can't attack, he doesn't have enough power$
NotLegend eats$
Cthulhu can't attack, he doesn't have enough power$
Cthulhu sleeps$
Power => 42000$
NotLegend attacks and destroys the city$
```

It is now time for me to tell you the story of C++ inheritance…

EXERCISE 0 - FAMILY MATTERS

Turn in: `Character.hpp`, `Character.cpp`

Kreog, a human farmer, shows up at the hero academy, with a lust for adventure in his eyes.
A hero's training requires him to succeed at the rite of passage, which is generally a quest.
The trainee then specializes to become a warrior, a magician, etc…
Before leaving for his quest, Kreog is named an apprentice by the Hero Academy general.

Create a simple `Character` class representing Kreog.
A character has a `name` and a `level`.
Those two values are required to create a `Character`.
A character with no name or level is like a pony that isn't pink: nonsense.

```
Character(const std::string &name, int level);
const std::string &getName() const;
int getLvl() const;
int getPv() const;
int getPower() const;
```

Anyone can ask a `Character` for their name and level.
No matter what, `Characters`' health points (PV) are capped at 100, as well as their energy points.
When a `Character` is created, its health and energy points are set to their maximum values.

Here is Kreog's record:

```
Name:           Kreog
Lvl:            1
Class:          Character
Race:           Koala
Strength:       5
Stamina:        5
Intelligence:   5
Spirit:         5
Agility:        5
```

A `Character` has characteristics such as `Stamina`, `Spirit` and `Agility`.
These are stored as `ints` within the `Character` class.
These characteristics can be modifiedd by child classes in their constructors.
For example, a `Warrior` has 12 `Strength` points, while a `Magician` only has 6.

Kreog's recocrd shows that a basic `Character` has its characteristics set to 5.

These characteristics can't be modified from outside the class.

During his first day at the Academy, Kreog learned two pieces of information.
The first is that there are two combat modes: close combat and ranged combat.

It is therefore possible to tell a `Character` which combat mode to use during a battle.
The following code shows you how:

```
Character c("poney", 42);

c.Range = Character::CLOSE;
c.Range = Character::RANGE;
```

The default value for `Range` is `CLOSE`.

`Character::Range`'s type is `AttackRange`.

The second piece of information he learned is that using a technique costs energy.
If a `Character` doesn't have enough energy when using a technique, it prints:

```
[name] out of power
```

and the technique's effect is cancelled.

```
int CloseAttack()
```

Cost: 10 energy points
Damage: 10 + `Strength`
Output: *"[name] strikes with a wooden stick"*
Return: the number of damage points dealt by the attack

```
int RangeAttack()
```

Cost: 10 energy points
Damage: 5 + `Strength`
Output: *"[name] tosses a stone"*
Return: the number of damage points dealt by the attack

```
int Heal()
```

Cost: 0 energy point
Cure: adds 50 health points
Output: *"[name] takes a potion"*

```
int RestorePower()
```

Cost: 0 energy point
Cure: restores 100 energy points
Output: *"[name] eats"*

As strong as they are, heroes can also take damage:

```
void TakeDamage(int damage)
```

This functions outputs

```
[name] takes [damage] damage
```

If a `Character` takes too much damage and its life points reach 0 or lower, the character screams

```
[name] out of combat
```

Here is a sample `main` function and its expected output:

```
int main()
{
        Character c("poney", 42);

        c.TakeDamage(50);
        c.TakeDamage(200);
        c.TakeDamage(200);
}
```

Terminal

```
~/B-CPP-300> ./a.out | cat -e
poney Created$
poney takes 50 damage$
poney out of combat$
poney out of combat$
```

Exercise 1 - Existentialism

Turn in: `Character.hpp/cpp`, `Warrior.hpp/cpp`

During their first year at the Academy, apprentices must go through a rite of passage: a quest imposed by the Academy during which they can prove themselves.
Kreog's quest is to recover a particular item in the **Tek Dungeon**.

Our story begins with Kreog walking through the door of the Tek Dungeon.
After barely 10 minutes in the dungeon, Kreog gets crushed by a set of heavy armor.

> ***[Armor]*** *Quountdouce count… By my beard, what on Earth did I stumble upon? Oh, that's a worm!*
> ***[Kreog]*** *I am not a worm, I am a soon to be hero.*
> ***[Armor]*** *Soon to be hero. Ha ha ha. In this case, let me introduce myself.*

```
Name:           Thor
Lvl:            42
Class:          Warrior
Race:           Dwarf
Strength:       12
Stamina:        12
Intelligence:   6
Spirit:         5
Agility:        7
```

```
int CloseAttack()
```

Cost: 30 energy points
Damage: 20 + `Strength`
Output: *"[name] strikes with his [weapon]"*
Weapon: hammer

```
int RangeAttack()
```

Cost: 10 energy points
Damage: 0
Output: *"[name] intercepts"*
Result: following attacks will be close combat only

```
int Heal()
```

Cost: 0 energy point
Cure: adds 50 health points
Output: *"[name] takes a potion"*

```
int RestorePower()
```

Cost: 0 energy point
Cure: restores 100 energy points
Output: *"[name] eats"*

Upon creation, the warrior says:

```
I'm [name] KKKKKKKKKRRRRRRRRRRRRRREEEEEEEEOOOOOOORRRRGGGGGGG
```

The type of the weapon is initialized in the constructor!

Create a `Warrior` class that inherits from the `Character` class.
After all, a `Warrior` IS a `Character`.
Just like its parent, a `Warrior` must be constructed with a name and a level.

A `Warrior`'s characteristics are different from those of a basic `Character`.

Moreover, warriors choose their weapon upon construction.
Therefore, the `Warrior` class has an `std::string weaponName` field.
`Warriors` are very proud beings.
They don't let anyone change the weapon they are using.

> ***[Thor]*** *When warriors are born, they make their first scream. For me, it was lordly! I still remember it:*
> `I'm Thor KKKKKKKKKRRRRRRRRRRRRRRREEEEEEEEEOOOOOOORRRRGGGGGGG`
> *So, young novice, did you finish your rite of passage?*
> ***[Kreog]*** *No, I just arrived!*
> ***[Thor]*** *Perfect, I am gonna come with you and help you with your quest!*

Exercise 2 - Children

Turn in: `Character.hpp/cpp`, `Warrior.hpp/cpp`, `Mage.hpp/cpp`, `Priest.hpp/cpp`

Thor is now part of **Team Kreog**.
Traveling through the Dungeon with Thor babbling constantly, a fire ball barely misses the group, burning off some of Thor's beard.
A very angry Thor starts looking for the origin of the fire ball, when a second one razes his feet.
Looking down, he sees a **Gnome** running all over the place.
Suddenly, it froze, stupefied by a magical spell cast by a nearby **Goblin**!
Out of nowhere, a light surrounds the Gnome and frees it.
Growing angrier by the second, Thor charges the Goblin and hits it with his powerful hammer.
The Goblin flies through the room and crashes violently into a pile of rocks.

> ***[Kreog]*** *How are you, little being?*
> ***[Gnome]*** *Little being? Did you take look in the mirror? I am of the tallest among my people! Let me introduce myself.*

```
Name:           Fluffy
Lvl:            40
Class:          Mage
Race:           Gnome
Strength:       6
Stamina:        6
Intelligence:   12
Spirit:         11
Agility:        7
```

```
int CloseAttack()
```

Cost: 10 energy points
Damage: 0
Output: *"[name] blinks"*
Result: after this attack, the `Mage` will use a ranged attack

```
int RangeAttack()
```

Cost: 25 energy points
Damage: 20 + `Spirit`
Output: *"[name] launches a fire ball"*

Upon creation, the mage says:

```
[name] teleported
```

`Mages` are created like so:

```
Mage(const std::string &name, int level);
```

```
int Heal()
```

Cost: 0 energy point
Cure: adds 50 health points
Output: *"[name] takes a potion"*

```
int RestorePower()
```

Cost: 0 energy point
Cure: recharges 50 + `Intelligence` energy points
Output: *"[name] takes a man potion"*

> *[Fluffy] Let me introduce you to my companion.*

```
Name:           Iopi
Lvl:            84
Class:          Priest
Race:           Orc
Strength:       4
Stamina:        4
Intelligence:   42
Spirit:         21
Agility:        2
```

`int CloseAttack()`

Cost: 10 energy points
Damage: 10 + `Spirit`
Output: *"[name] uses a spirit explosion"*
Result: after this attack, the `Mage` will use a ranged attack

`int RangeAttack()`

Cost: 25 energy points
Damage: 20 + `Spirit`
Output: *"[name] launches a fire ball"*

`int Heal()`

Cost: 10 energy points
Cure: adds 70 health points
Output: *"[name] casts a little heal spell"*

Upon creation, the priest says;

```
[name] enters in the order
```

A priest is a magician specialized in sacred magic.

Exercise 3 - Paladins

Turn in: `Character.hpp/cpp`, `Warrior.hpp/cpp`, `Mage.hpp/cpp`, `Priest.hpp/cpp`, `Paladin.hpp/cpp`

After introducing themselves, Fluffy and Iopi decide to escort Thor and Kreog in their quest.
After a few hours of wandering and fighting, the group finds a room in which stands a strange man, covered in sweat, dancing in the middle of the room.

> ***[Fluffy]*** *Phiste! Is that you???*
> ***[Sweating man]*** *Flff! Long time no see*

```
Name:           Phiste
Lvl:            42
Class:          Paladin
Race:           Human
Strength:       9
Stamina:        10
Intelligence:   10
Spirit:         10
Agility:        2
```

```
int CloseAttack()
```

Cost: 30 energy points
Damage: 20 + `Strength`
Output: *"[name] strikes with his [weapon]"*
Weapon: hammer

```
int RangeAttack()
```

Cost: 25 energy points
Damage: 20 + `Spirit`
Output: *"[name] launches a fire ball"*

```
int Heal()
```

Cost: 10 energy points
Cure: adds 70 health points
Output: *"[name] casts a little heal spell"*

```
int RestorePower()
```

Cost: 0 energy point
Cure: restores 100 energy points
Output: *"[name] eats"*

Upon creation, the paladin says:

```
the light falls on [name]
```

A `Paladin` is a mix between a `Warrior` and a `Priest`.

Follow the inheritance order described above.

The `Paladin` uses the `Priest`'s healing spell and fire pall, and the `Warrior`'s close combat attack.
`Paladins` can also charge like `Warriors` using the following function:

```
int Intercept();
```

Refer to the `Warrior` description.

Pay attention to the initialization order of virtually inherited parents.

Phiste, still sweating, gives his lifelong friend Flff a big hug.
After hours of telling old tales and memories, the team decides to accept Phiste as a new group member for the quest.

Exercise 4 - Or is it eleven?

Turn in: `Character.hpp/cpp`, `Warrior.hpp/cpp`, `Mage.hpp/cpp`, `Priest.hpp/cpp`, `Paladin.hpp/cpp`, `Hunter.hpp/cpp`

Upset because of Phiste and Flff's reminiscing, the other team members decide to charge a pack of Goblins to clear their heads.
Suddenly, an arrow strikes a Goblin right in front of Kreog.
Surprised, Kreog turns around and sees a little green something jumping across the room.
The battle is now over, and a cute little she-elf emerges from the shadows and walks up to the group.

> ***[Phiste]*** *Quountdouce count… An elf in the dungeon! You are pretty far from your home-wood, little miss!*
> ***[Elf]*** *LITTLE MISS??? Your eyes betray you, filthy human! I am NO MISS! I am a HE-ELF!*

```
Name:           Fourdr
Lvl:            40
Class:          Hunter
Race:           Elf
Strength:       9
Stamina:        9
Intelligence:   5
Spirit:         6
Agility:        25
```

```
int CloseAttack()
```

Cost: 30 energy points
Damage: 20 + `Strength`
Output: *"[name] strikes with his [weapon]"*
Weapon: sword

```
int RangeAttack()
```

Cost: 25 energy points
Damage: 20 + `Agility`
Output: *"[name] uses his bow"*

```
int Heal()
```

Cost: 0 energy point
Cure: adds 50 health points
Output: *"[name] takes a potion"*

```
int RestorePower()
```

Cost: 0 energy point
Cure: restores 100 energy points
Output: *"[name] meditates"*

Upon creation, an elf sats:

```
[name] is born from a tree
```

A `Hunter` is a character that uses warrior characteristics for close combat.
However, elves don't want to be seen as warrior by other creatures.
It is, however, a familiy tradition to be associated to a warrior within the elven clan.

Conclusion

After hours of arguing with Phiste, Fourdr joins the group and everyone starts moving through the dungeon again.
Unfortunately, Iopi asked Fluffy who the strongest between the Dwarf and the Elf was.
This question started a violent debate, making more noise than a Hobbit in the Moria.

The argument was so noisy that a dragon came in to see what the ruckus was about.

> ***[Dragon]*** *Hey, can you shut up now? Some of us are trying to get some sleep!*
> ***[Thor]*** *Where the hell did this guy come from? Can't he let us have a friendly debate? Give me a minute to knock him out.*
> ***[Kreog]*** *WAIT! My quest description mentioned a sacred dragon! Dragon, do you have something for me?*
> ***[Dragon]*** *Sure. I can give you my treasure, under two conditions: tell the dwarf and elf to shut up, and tell the big guys from the academy to use another dungeon to test their little punks. I am growing tired of these soon to be losers!*

Kreog accepts the deal and asks the Dwarf and Elf to be quiet.
Five minutes later, the dragon comes back with a stuffed Koala and hands it to Kreog.

> ***[Dragon]*** *Here is the glorious object of your quest!*
> ***[Thor]*** *What the…? I was forced to settle with an Elf to get a stuffed Koala? Quountdounce count… This is the last time I team up with newbies…*

The quest now over, the group heads back to the Academy and throw a huge party.

Ok guys, I know the end is a little short, but if you come up with something better, post it on Yammer!
If a suggestion is good enough, it will be included in the subject for next year!