# B3 - C++ Pool

B-PAV-242

# Day 09

Kreog's Quest

KOALA

42.0

EPITECH.
L'ECOLE DE L'INNOVATION ET DE L'EXPERTISE INFORMATIQUE

# 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,...).

## 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 -1 - C

| | Exercise: -1 | points : 1 |
|---|---|---|
| Simple inheritance in C, it's easy | | |
| Turn-in directory: cpp_d09/ex-1 | | |
| Compiler: gcc | Compilation flags: -W -Wall -Werror -Wextra | |
| Makefile: No | Rules: n/a | |
| Files to turn in: Exo-1.h, Exo-1.c | | |
| Notes: None | | |
| Forbidden functions: None | | |

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, so use the right compiler!

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

Create an `s_cthulhu` structure, and a `t_cthulhu` 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:

```
t_cthulhu *NewCthulhu();
```

Creates a new `t_cthulhu` 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(t_cthulhu *this);
```

Prints the instance's power to the standard output.

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

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

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

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

Create an `s_koala` structure and a `t_koala` alias to it. It is composed of a `t_cthulhu` 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:

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

Creates a new `t_koala` object, initializes it and returns a pointer to it.

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

Feeds the koala and increases its energy by 42 points.

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

```
static void KoalaInitializer(t_koala *this, char *_name, char _isALegend);
static void CthulhuInitializer(t_cthulhu *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()
{
    t_koala *_LKoala = NewKoala("Legend", 1);
    t_koala *_NLKoala = NewKoala("NotLegend", 0);

    t_cthulhu *_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;
}
```

*main.c*

Terminal

```
~/B-PAV-242> ./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

| | | |
|---|---|---|
| KOALA | Exercise: 00 | points : 2 |
| Every child needs a parent | | |
| Turn-in directory: `cpp_d09/ex00` | | |
| Compiler: `g++` | Compilation flags: `-W -Wall -Wextra -Werror -std=c++14` | |
| Makefile: `No` | Rules: `n/a` | |
| Files to turn in: `Character.hh, Character.cpp` | | |
| Notes: `None` | | |
| Forbidden functions: `‘using namespace‘ keyword` | | |

**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
```

- Close combat attack: `int CloseAttack()`
  - Cost: 10 energy points
  - Damage: 10 + `Strength`
  - Output: *"[name] strikes with a wood stick"*
- Ranged attack: `int RangeAttack()`
  - Cost: 10 energy points
  - Damage: 5 + `Strength`
  - Output: *"[name] launches a stone"*

These two functions return the number of damage points dealt by the attack.

- Heal: `void Heal()`
  - Cost: 0 energy points
  - Cure: adds 50 health points
  - Output: *"[name] takes a potion"*
- Restore power: `void RestorePower()`
  - Result: restores 100 energy points
  - Output: *"[name] eats"*

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 `Strength`, `Stamina`, `Intelligence`, `Spirit` and `Agility` 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;
```

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

The default value for `Range` is `CLOSE`

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.

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

- `void TakeDamage(int damage)`
    - Output: *"[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:

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

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

*main.cpp*

Terminal

```
~/B-PAV-242> ./a.out | cat -e
poney Created$
poney takes 50 damage$
poney out of combat$
poney out of combat$
```

## Exercise 1 - Existentialism

| | Exercise: 01 | points : 3 |
|---|---|---|
| Like parent, like child | | |
| Turn-in directory: `cpp_d09/ex01` | | |
| Compiler: `g++` | Compilation flags: `-W -Wall -Wextra -Werror -std=c++14` | |
| Makefile: `No` | Rules: `n/a` | |
| Files to turn in: `Character.hh, Character.cpp, Warrior.hh, Warrior.cpp` | | |
| Notes: `None` | | |
| Forbidden functions: `‘using namespace‘ keyword` | | |

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
```

- Close combat attack: `int CloseAttack()`
  - Cost: 30 energy points
  - Damage: 20 + `Strength`
  - Output: *“[name] strikes with his [weapon]”*
- Weapon: hammer
- Ranged attack: `int RangeAttack()`
  - Cost: 10 energy points
  - Damage: 0
  - Output: *“[name] intercepts”*

    - Result: following attacks will be close combat only

- Heal: `void Heal()`
    - Cost: 0 energy points
    - Cure: adds 50 health points
    - Output: *"[name] takes a potion"*
- Restore power: `void RestorePower()`
    - Result: sets energy points to 100
    - Output: *"[name] eats"*
- Creation
    - Output: *"I'm [name] KKKKKKKKKKRRRRRRRRRRRRRREEEEEEEEOOOOOORRRRGGGGGGG"*

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 KKKKKKKKKKRRRRRRRRRRRRRRREEEEEEEEOOOOOOORRRRGGGGGGG

        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

| | Exercise: 02 | points : 4 |
|---|---|---|
| Children, always more children | | |
| Turn-in directory: `cpp_d09/ex02` | | |
| Compiler: `g++` | Compilation flags: `-W -Wall -Wextra -Werror -std=c++14` | |
| Makefile: `No` | Rules: `n/a` | |
| Files to turn in: `Character.hh, Character.cpp, Warrior.hh, Warrior.cpp, Mage.hh, Mage.cpp, Priest.hh, Priest.cpp` | | |
| Notes: `None` | | |
| Forbidden functions: `‘using namespace‘ keyword` | | |

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
```

- Close combat attack:
  - Cost: 10 energy points
  - Damage: 0
  - Output: *"[name] blinks"*
  - Result: after this attack, the `Mage` will use a ranged attack
- Ranged attack:
  - Cost: 25 energy points
  - Damage: 20 + `Spirit`
  - Output: *"[name] launches a fire ball"*
- Heal: `void Heal()`
  - Cost: 0 energy points
  - Cure: adds 50 health points
  - Output: *"[name] takes a potion"*
- Restore power: `void RestorePower()`
  - Result: Recharges 50 + `Intelligence` energy points
  - Output: *"[name] takes a mana potion"*
- Creation
  - Output: *"[name] teleported"*

`Mages` are created like so:

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

```
[Sully] 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
```

- Close combat attack:
    - Cost: 10 energy points
    - Damage: 10 + `Spirit`
    - Output: *"[name] uses a spirit explosion"*
    - Result: after this attack, the `Mage` will use a ranged attack
- Ranged attack:
    - Cost: 25 energy points
    - Damage: 20 + `Spirit`
    - Output: *"[name] launches a fire ball"*
- Heal: `void Heal()`
    - Cost: 10 energy points
    - Cure: adds 70 health points
    - Output: *"[name] casts a little heal spell"*
- Creation
    - Output: *"[name] enters in the order"*

A priest is a magician specialized in sacred magic.

## Exercise 3 - Paladins

| | Exercise: 03 | points : 4 |
|---|---|---|
| A paladin is a being mixing priest and warrior characteristics | | |
| Turn-in directory: `cpp_d09/ex03` | | |
| Compiler: `g++` | Compilation flags: `-W -Wall -Wextra -Werror -std=c++14` | |
| Makefile: `No` | Rules: `n/a` | |
| Files to turn in: `Character.hh, Character.cpp, Warrior.hh, Warrior.cpp, Mage.hh, Mage.cpp, Priest.hh, Priest.cpp, Paladin.hh, Paladin.cpp` | | |
| Notes: `None` | | |
| Forbidden functions: `‘using namespace‘ keyword` | | |

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.

```
[Flff]          Phiste! Is that you???
[Sweating man]  Flff! Long time no see, my friend.
```

```
Name:           Phiste
Lvl:            42
Class:          Paladin
Race:           Human

Strength:       9
Stamina:        10
Intelligence:   10
Spirit:         10
Agility:        2
```

- Close combat attack:
  - Cost: 30 energy points
  - Damage: 20 + `Strength`
  - Output: *“[name] strikes with his [weapon]”*
- Weapon: hammer
- Ranged attack:
  - Cost: 25 energy points
  - Damage: 20 + `Spirit`
  - Output: *“[name] launches a fire ball”*
- Heal: `void Heal()`
  - Cost: 10 energy points
  - Cure: adds 70 health points
  - Output: *“[name] casts a little heal spell”*

- Restore power: `void RestorePower()`
  - Result: Sets energy points to 100
  - Output: *"[name] eats"*
- Creation
  - Output: *"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?

| | | |
|---|---|---|
| | Exercise: 04 | points : 6 |
| From leaf to tree | | |
| Turn-in directory: `cpp_d09/ex04` | | |
| Compiler: `g++` | Compilation flags: `-W -Wall -Wextra -Werror -std=c++14` | |
| Makefile: `No` | Rules: `n/a` | |
| Files to turn in: `Character.hh, Character.cpp, Warrior.hh, Warrior.cpp, Mage.hh, Mage.cpp, Priest.hh, Priest.cpp, Paladin.hh, Paladin.cpp, Hunter.hh, Hunter.cpp` | | |
| Notes: `None` | | |
| Forbidden functions: `‘using namespace‘ keyword` | | |

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
```

- Close combat attack:
  - Cost: 30 energy points
  - Damage: 20 + `Strength`
  - Output: *"[name] strikes with his [weapon]"*
- Weapon: sword
- Ranged attack:
  - Cost: 25 energy points
  - Damage: 20 + `Agility`
  - Output: *"[name] uses his bow"*
- Heal: `void Heal()`

    - Cost: 0 energy points
    - Cure: adds 50 health points
    - Output: *“[name] takes a potion”*

- Restore power: `void RestorePower()`
    - Result: Sets energy points to 100
    - Output: *“[name] meditates”*

- Creation
    - Output: *“[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 Flff 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!