СУ „Св. Климент Охридски“

Факултет по математика и информатика

Проект „Зоопарк“ по ООП-практикум 2018/2019г.

(специалност Информатика)

Изготвила: Даниела Лозанова

**Глава 1. Увод**

Идеята на проекта е да бъде създаден функционален и оптимален Зоопарк, който да бъде удобен за ползване с изчистени и ясни команди.

**Глава 2. Преглед на предметната област**

Проекта се изправя пред следните проблеми:

1. Дублиране на прекалено много данни при множествено наследяване
2. Нееднозначност на същите тези данни
3. Функции, които трябва да имат една и съща функционалност реализирана по различен начин в зависимост от обектите, през които се извикват
4. Полиморфен контейнер
5. Коректно поведение при добавяне на определено животно

/т.е. да не получаваме разхищение на памет и строеж на ненужно количество клетки- проблемът ще бъде разгледан по- подробно в следващите страници с подходящи примери/

Проблемите биват решени както следва: чрез виртуално множествено наследяване(няма дублиране на данни и те се инициализират по единствен начин- решава се така наречения диамантен проблем). Полиморфните действия се решават, чрез виртуални функции(това важи и за полиморфния контейнер, където от съществено значение е виртуалната функция за клониране/clone/).

Реализиран е потребителски интерфейс, в който обаче е скрита функционалността за строеж на клетка. Защото няма нужда да се създава празна клетка от дадена среда, при положение, че има незапълнена такава от същата среда(ненужно разхищение на ресурси). Ще бъде създадена клетка тогава и само тогава когато няма създадена клетка, която да отговаря на зададените изисквания, когато такава клетка/и вече съществува, но няма налични свободни места.

Скрита е и функционалността за презареждане на склад, в командата за хранене.В нея има допълнително запитване дали склада да бъде зареден или не. Сладът ще бъде зареден с храна с необходимото количество.

**Глава 3. Проектиране**

Споменатата по- горе функционалност е реализирана и скрита по следният начин:

*void Zoo::addAnimal(const Animal \* ani)*

*{*

*for (int i = 0; i < pos; ++i)*

*{*

*if (rightCage(i, ani))*

*{*

*if (!cages[i].addAnimal(ani))*

*{continue;}*

*else return;*

*}*

*}*

*if (pos == cap) resize(cap \* 2);*

*cages[pos].createCage(ani);*

*cages[pos].addAnimal(ani);*

*pos += 1;*

*}*

**Йерархията е следната:**

**Animal**

virtual

-char\* name

virtual

**Predator Group**

-Food f - char\* group

-char\* species

**SpecificAnimal**

**Food Zoo**

-char\* food -FoodStorage fs

-unsigned int kg -Cage\* cages

-int cap

**FoodStorage** -int pos

-unsigned int meat

-unsigned int plants

-unsigned int fish

**Cage**

-Animal\*\* animals

-char\* nameC

-char\* landskape

-int humidity

-int flora

-int latitude

-int pos

**Глава 4. Реализация**

*SpecificAnimal::SpecificAnimal(const char \* name, const char \* group, const Food & f, const char \* species)*

*:Animal(name), Group(name, group), Predator(f, species, name)*

*{*

*}*

*SpecificAnimal::SpecificAnimal(const SpecificAnimal& other)*

*: Animal(other), Group(other), Predator(other)*

*{*

*}*

*SpecificAnimal & SpecificAnimal::operator=(const SpecificAnimal & other)*

*{*

*if (this != &other)*

*{*

*Animal::operator=(other);*

*Group::operator=(other);*

*Predator::operator=(other);*

*}*

*return \*this;*

*}*

*Animal\* SpecificAnimal::clone() const*

*{*

*return new SpecificAnimal(\*this);*

*}*

Оптимизации:

Преоразмеряването става по две.

Изнасяне на повтарящ се или подобен код във помощтни функции, като например функция за копиране или функция за освобождаване на памет. Помощните функции са в private секции.

**Глава 7. Заключение**

Поставените в началото цели са реализирани по възможно най- добрия и оптимален начин. Има място за развитие и добавяне на нова функционалност, като например нови видове клетки с други характеристики или животни.

Литература:

https://en.wikipedia.org/wiki/Main\_Page