***Министерство образования Республики Беларусь***

***Учреждение образования***

***«Брестский государственный технический университет»***

***Кафедра ИИТ***

**Лабораторная работа №4**

**По дисциплине ППвИС за III семестр**

**Тема: «Функции-друзья классов. Перегрузка операторов классов»**

**Выполнил:**

Студент группы ИИ-15 (1)

2-го курса

Волк И. А.

**Проверил:**

Михняев А.Л.

Брест 2018

ЦЕЛЬ РАБОТЫ

1. Изучение использования friend-функций для доступа к классам извне.

2. Изучение особенностей перегрузки операторов в пользовательских классах.

СПИСОК ЗАДАЧ

Индивидуальные задания по разработке классов.

*Выбранная предметная область:* игровой юнит.

*Описание тестового примера.*

В качестве тестовых входных данных будем использовать следующие:

1. Entity::name = “Bob”

Entity::health = 100;

Entity::man = 10;

Entity::attackPower = 20;

Current price = 8.49

2. Entity::name = “Tom”

Entity::health = 80;

Entity::man = 80;

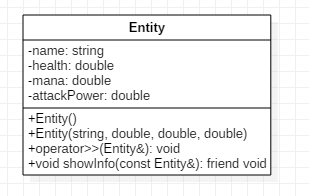
Entity::attackPower = 30;

*Описание классов:*

1. Имя: Entity

Назначение: хранит информацию об юните: его здоровье, ману и мощность атаки, а также методы для взаимодействия юнитов.

UML:



*Листинг кода программы:*

// entity.h

#pragma once

#include <string>

void log(std::string author, std::string message);

class Entity

{

std::string name;

double health;

double mana;

double attackPower;

public:

Entity();

Entity(std::string name, double health, double mana, double attackPower);

void operator>>(Entity& victim);

friend void showInfo(const Entity& entity);

};

void showInfo(const Entity& entity);

// entity.cpp

#include "entity.h"

#include <iostream>

void log(std::string author, std::string message)

{

std::cout << author << ": " << message << std::endl;

}

void showInfo(const Entity& entity)

{

std::cout << "Name: " << entity.name << std::endl;

std::cout << "Health: " << entity.health << std::endl;

std::cout << "Mana: " << entity.mana << std::endl;

std::cout << "Attack power: " << entity.attackPower << std::endl;

}

Entity::Entity()

: name("without\_name"), health(0), mana(0), attackPower(0)

{}

Entity::Entity(std::string name, double health, double mana, double attackPower)

: name(name), health(health), mana(mana), attackPower(attackPower)

{}

void Entity::operator>>(Entity& victim)

{

if (this->health == 0) { return; }

log(this->name, ("attack " + victim.name + "."));

if (victim.health == 0)

{

log(this->name, ("Cannot attack " + victim.name + "."));

log(this->name, (victim.name + " is already dead."));

return;

}

if ((victim.health - this->attackPower) <= 0)

{

std::cout << victim.name << ": health -" << victim.health << std::endl;

victim.health = 0;

std::cout << victim.name << " has been killed by " << this->name << std::endl;

return;

}

victim.health -= this->attackPower;

std::cout << victim.name << ": health -" << this->attackPower << std::endl;

std::cout << victim.name << ": current health: " << victim.health << std::endl;

}

// main.cpp

#include <iostream>

#include "entity.h"

int main()

{

Entity unit1("Bob", 100, 10, 20),

unit2("Tom", 80, 80, 30);

showInfo(unit1);

std::cout << std::endl;

showInfo(unit2);

std::cout << std::endl;

for (int i = 0; i < 5; i++)

{

unit1 >> unit2;

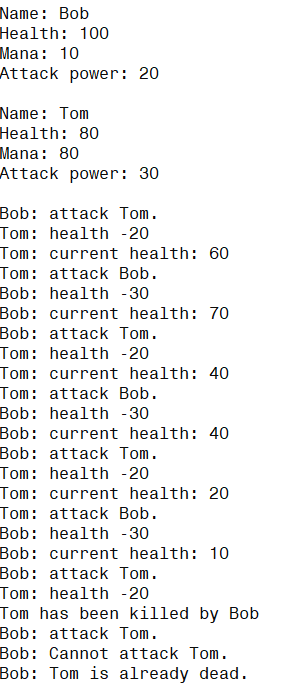
unit2 >> unit1;

}

std::cin.get();

}

*Листинг результата:*



Вывод: изучил использования friend-функций для доступа к классам извне; изучил особенностей перегрузки операторов в пользовательских классах