***Министерство образования Республики Беларусь***

***Учреждение образования***

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

***Кафедра ИИТ***

**Лабораторная работа №7**

**По дисциплине ППвИС за III семестр**

**Тема: «Шаблоны»**

**Выполнил:**

Студент группы ИИ-15 (1)

2-го курса

Волк И. А.

**Проверил:**

Михняев А.Л.

Брест 2018

ЦЕЛЬ РАБОТЫ

1. Изучение правил написания шаблонов функций.

2. Изучение правил написания шаблонов классов.

3. Реализация шаблонов функций.

4. Реализация шаблонов классов.

Выбранная предметная область: игровой юнит.

Листинг программы:

#include <iostream>

#include <string>

#define UnitDamage double

#define UnitName std::string

#define UnitHealth double

#define UnitType std::string

#define UnitHeal double

class Orc

{

protected:

UnitType type;

UnitHealth health;

UnitDamage damage;

public:

Orc()

: type("Orc"), health(0), damage(0) {}

Orc(UnitHealth health, UnitDamage damage)

: type("Orc"), health(health), damage(damage) {}

};

class Elf

{

protected:

UnitType type;

UnitHealth health;

UnitDamage damage;

public:

Elf()

: type("Elf"), health(0), damage(0) {}

Elf(UnitHealth health, UnitDamage damage)

: type("Elf"), health(health), damage(damage) {}

};

class Healer

{

protected:

UnitHeal heal;

public:

Healer()

: heal(0) {}

Healer(UnitHeal heal)

: heal(heal) {}

};

template<class...Type>

class Unit : Type...

{

UnitName name;

public:

Unit()

: Type()..., name("noname") {}

Unit(std::string name, const Type&... val)

: Type(val)..., name(name) {}

template<class ...Type1, class ...Type2>

friend void heal(Unit<Type1...> u1, Unit<Type2...> u2);

template<class...Type1, class ...Type2>

friend void attack(Unit<Type1...> u1, Unit<Type2...> u2);

};

template<class ...Type1, class ...Type2>

void attack(Unit<Type1...> u1, Unit<Type2...> u2)

{

std::cout << u2.type << " " << u2.name << " has been attacked by " << u1.type << " " << u1.name << std::endl;

u2.health -= u1.damage;

std::cout << u2.name << ": -" << u1.damage << std::endl;

}

template<class ...Type1, class ...Type2>

void heal(Unit<Type1...> u1, Unit<Type2...> u2)

{

std::cout << u2.type << " " << u2.name << " has been healed by " << u1.type << " " << u1.name << std::endl;

u2.health += u1.heal;

std::cout << u2.name << ": +" << u1.heal << std::endl;

}

int main()

{

Unit<Orc> orc("Bob", Orc(100, 10));

Unit<Elf> elf("Tom", Elf(80, 20));

Unit<Elf, Healer> elfH("Fergus", Elf(50, 10), Healer(10));

attack(orc, elf);

attack(elfH, orc);

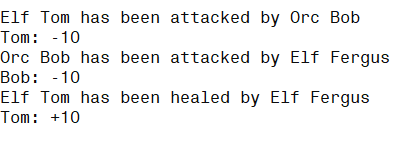
heal(elfH, elf);

std::cin.get();

return 0;

}

Результат выполнения:



Вывод: изучил правила написания шаблонов функций и правила написания шаблонов классов; реализовал шаблоны функций и классов.