МИНИСТЕРСТВО ТРАНСПОРТА И КОММУНИКАЦИЙ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования

«Белорусский государственный университет транспорта»

Кафедра «Информационно-управляющие системы и технологии»

Отчет  
по лабораторным работам

по дисциплине «Объектно-ориентированное программирование»

Выполнил Проверил

студент группы ГИ-21 ст. преп. каф. «ИУСиТ»

Зайцев И. С. Голдобина Т. А.

Гомель, 2020

Оглавление

[Лабораторная работа № 6 Тема: «МНОЖЕСТВЕННОЕ НАСЛЕДОВАНИЕ, ВИРТУАЛЬНОЕ НАСЛЕДОВАНИЕ» 3](#_Toc56765293)

1.1 [Задания 3](#_Toc56765294)

[1.2 Контрольные вопросы 7](#_Toc56765295)

Лабораторная работа № 6  
Тема: «МНОЖЕСТВЕННОЕ НАСЛЕДОВАНИЕ, ВИРТУАЛЬНОЕ НАСЛЕДОВАНИЕ»

### **Цель**

Изучить принципы и получить практические навыки при использовании множественного наследования; рассмотреть случаи, когда необходимо использовать виртуальное наследование.

## Задания

### Задание 1.1

### Условие

Первый базовый класс – Млекопитающие; поля – способ питания, вес, среда обитания; производные классы – хищники и травоядные. Во втором базовом классе описываются географические регионы.

Программный код

#include <iostream>

using namespace std;

enum food

{

grazing, predation, parasitism //выедание, хищничество, паразитизм

};

enum habitat

{

terrestrial, underground, aquatic, aerial, woody //наземные, подземные, водные, воздушные, древесные

};

enum continents

{

Eurasia, Africa, North\_America, South\_America, Australia, Antarctica

};

class Mammals

{protected:

int weight;

food a;

habitat b;

public:

Mammals(int weight, food a, habitat b)

{

this->weight = weight;

this->a = a;

this->b = b;

}

Mammals(){}

string GetVivod()

{

string str = "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\nWeight: " + to\_string(weight) + " kg\nFood: ";

switch(a)

{

case grazing: {str += "grazing";} break;

case predation: {str += "predation";} break;

case parasitism: {str += "parasitism";} break;

}

str += "\nHabitat: ";

switch(b)

{

case terrestrial: {str += "terrestrial";} break;

case underground: {str += "underground";} break;

case aquatic: {str += "aquatic";} break;

case aerial: {str += "aerial";} break;

case woody: {str += "woody";} break;

}

str += "\nContinents: ";

return str;

}

};

class Regions

{

protected:

continents c;

public:

Regions(continents c)

{

this->c = c;

}

Regions(){}

void GetC()

{

switch(c)

{

case Eurasia: cout << "Eurasia\n"; break;

case Africa: cout << "Africa\n"; break;

case North\_America: cout << "North America\n"; break;

case South\_America: cout << "South America\n"; break;

case Australia: cout << "Australia\n"; break;

case Antarctica: cout <<"Antarctica\n"; break;

}

}

};

class Predators : public Mammals, public Regions

{

public:

Predators(int weight, habitat b, continents c) : Mammals(), Regions()

{

this->weight = weight;

this->a = predation;

this->b = b;

this->c = c;

}

};

class Herbivores : public Mammals, public Regions

{

public:

Herbivores(int weight, habitat b, continents c) : Mammals(), Regions()

{

this->weight = weight;

this->a = grazing;

this->b = b;

this->c = c;

}

};

food function1(int a)

{

switch(a)

{

case 1: return grazing;

case 2: return predation;

case 3: return parasitism;

}

}

habitat function2(int b)

{

switch(b)

{

case 1: return terrestrial;

case 2: return underground;

case 3: return aquatic;

case 4: return aerial;

case 5: return woody;

}

}

continents function3(int c)

{

switch(c)

{

case 1: return Eurasia;

case 2: return Africa;

case 3: return North\_America;

case 4: return South\_America;

case 5: return Australia;

case 6: return Antarctica;

}

}

int main()

{

int a, b, c, weight;

cout << "Enter weight: ";

cin >> weight;

cout <<"\n1. Grazing\n2. Predation\n3. Parasitism\nEnter number: ";

cin >> a;

cout <<"\n1. Terrestrial\n2. Underground\n3. Aquatic\n4. Aerial\n5. Woody\nEnter number: ";

cin >> b;

cout <<"\n1. Eurasia\n2. Africa\n3. North America\n4. South America\n5. Australia\n6. Antarctica\nEnter number: ";

cin >> c;

Mammals first(weight, function1(a), function2(b));

Regions second(function3(c));

Predators third(weight, function2(b), function3(c));

Herbivores fourth(weight, function2(b), function3(c));

cout <<first.GetVivod();

second.GetC();

return 0;

}

### Результат



Рисунок 1 – Результат задания 1

## 1.2 Контрольные вопросы

1. Для чего используется множественное наследование? Чем оно отличается от простого наследования?

Множественное наследование – это иерархическая структура, в которой производный класс наследует от нескольких базовых классов.

В случае простого наследования производный класс получает доступ к публичным и защищенным элементам базового класса.

В случае множественного наследования производный класс получает доступ к публичным и защищенным элементам всех базовых классов.

Кроме того, в момент создания экземпляра производного класса выполняться все конструкторы базовых классов, а при удалении объекта выполнятся все деструкторы базовых классов.

1. Каков механизм вызова конструкторов при множественном и виртуальном наследовании?

Множественное наследование позволяет порожденному классу наследовать элементы более, чем от одного базового класса.

Базовый класс может быть задан только один раз в списке порождения нового класса.

1. Какие проблемы возможны при множественном наследовании?

Первая – конфликт имен методов или атрибутов нескольких базовых классов.

Вторая возникает при многократном включении некоторого базового класса.

### Вывод

В результате лабораторной работы научились применять множественное наследование.