Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет информационных технологий и управления

ОТЧЕТ

к лабораторной работе №5

**«МНОЖЕСТВЕННОЕ НАСЛЕДОВАНИЕ. ВИРТУАЛЬНОЕ НАСЛЕДОВАНИЕ»**

|  |  |
| --- | --- |
|  | |
|  |
|  |
|  |
|  |  | | |
|  | | |  | |  |
|  | | |  | |  |
| Студент гр. 324402 | | |  | | Цевелюк А. И. |
| Проверил | | |  | | Булыга Д. И. |

Минск 2024

**1 Цель работы**

Цель работы:изучить принципы и получить практические навыки при использовании множественного наследования; рассмотреть случаи, когда необходимо использовать множественное и виртуальное наследование.

**2 Теоретические сведения по лабораторной работе**

Множественное наследование в ООП позволяет производному классу наследовать свойства и методы сразу от нескольких базовых классов. Это обеспечивает повторное использование кода и гибкость программы, разделяя различные аспекты функциональности по разным базовым классам. Однако, множественное наследование может усложнить структуру программы и привести к возможным конфликтам имен, если методы базовых классов имеют одинаковые имена.

Виртуальное наследование позволяет избежать проблем, связанных с множественным наследованием, когда возникают дублирования базовых классов. При использовании виртуального наследования ключевое слово virtual указывает, что базовый класс должен быть унаследован только один раз, избегая дублирования данных и методов. Это упрощает структуру классов и иерархии, хотя может усложнить понимание кода из-за неявности наследования.

**3 Формулировка индивидуального задания**

Построить иерархию классов согласно схеме наследования, приведенной на рисунке ниже по предметной области «Кодеки». Каждый класс должен содержать необходимые конструкторы и методы работы с полями классов. Функция main() должна иллюстрировать работу с массивами объектов всех созданных классов. При необходимости самостоятельно добавить классы для реализации множественного наследования.

**4 Код решения индивидуального задания**

#include <iostream>

#include <string>

#include <vector>

class codec

{

public:

virtual ~codec() = default;

template <typename T>

T encode(const std::string&);

template <typename T>

std::string decode(const T&);

};

class json\_codec : public virtual codec

{

public:

std::string encode(const std::string&);

std::string decode(const std::string&);

};

std::string json\_codec::encode(const std::string& data)

{

std::string result = "{ \"data\": [";

const int size = static\_cast<int>(data.size());

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

{

result += "\"";

result += data[i];

result += "\"";

if (i < size - 1) {

result += ", ";

}

}

result += "] }";

return result;

}

std::string json\_codec::decode(const std::string& data)

{

std::string result;

bool is\_reading\_char = false;

auto it = data.cbegin() + 11;

for (it; it != data.cend(); ++it)

{

const char ch = \*it;

if (ch == '\"' && !is\_reading\_char)

is\_reading\_char = true;

else if (ch == '\"' && is\_reading\_char)

is\_reading\_char = false;

else if (is\_reading\_char)

result += ch;

}

return result;

}

class xml\_codec : public virtual codec

{

public:

std::string encode(const std::string&);

std::string decode(const std::string&);

};

std::string xml\_codec::encode(const std::string& data)

{

std::string result = "<data>\n";

for (const char ch : data)

{

result += "\t<char>";

result += ch;

result += "</char>\n";

}

result += "</data>";

return result;

}

std::string xml\_codec::decode(const std::string& data)

{

std::string result;

std::string::size\_type start;

std::string::size\_type end = 0;

while ((start = data.find("<char>", end)) != std::string::npos) {

start += 6;

end = data.find("</char>", start);

if (end == std::string::npos) {

break;

}

result += data.substr(start, end - start);

}

return result;

}

class protobuf\_codec : public virtual codec

{

public:

std::vector<uint8\_t> encode(const std::string&);

std::string decode(const std::vector<uint8\_t>&);

};

std::vector<uint8\_t> protobuf\_codec::encode(const std::string& data)

{

std::vector<uint8\_t> result;

const size\_t data\_size = data.size();

result.push\_back(static\_cast<uint8\_t>(data\_size));

for (const char ch : data)

result.push\_back(static\_cast<uint8\_t>(ch));

return result;

}

std::string protobuf\_codec::decode(const std::vector<uint8\_t>& data)

{

if (data.empty()) return "";

const size\_t data\_size = data[0];

std::string result;

for (size\_t i = 1; i <= data\_size; ++i) {

result += static\_cast<char>(data[i]);

}

return result;

}

class codecs final : public json\_codec, public xml\_codec, public protobuf\_codec

{

};

int main()

{

int choice = 0;

codecs codec;

do

{

choice = 0;

std::cout << "Enter your choice:\n1 - continue\nany else - exit: ";

std::cin >> choice;

if (choice == 1)

{

system("cls");

std::string line;

std::cout << "Enter line to encode: ";

std::cin.ignore();

std::getline(std::cin, line);

std::string buffer = codec.json\_codec::encode(line);

std::cout << "Line in JSON:\n";

std::cout << buffer << '\n';

std::cout << "Encoded from JSON:\n";

std::cout << codec.json\_codec::decode(buffer) << '\n';

buffer = codec.xml\_codec::encode(line);

std::cout << "Line in XML:\n";

std::cout << buffer << '\n';

std::cout << "Encoded from XML:\n";

std::cout << codec.xml\_codec::decode(buffer) << '\n';

std::vector<uint8\_t> buffer\_vector = codec.protobuf\_codec::encode(line);

std::cout << "Line in Protobuf:\n";

for (const unsigned char ch : buffer\_vector)

{

std::cout << +ch << ' ';

}

std::cout << "\nEncoded from Protobuf:\n";

std::cout << codec.protobuf\_codec::decode(buffer\_vector) << '\n';

buffer\_vector.clear();

system("pause");

system("cls");

}

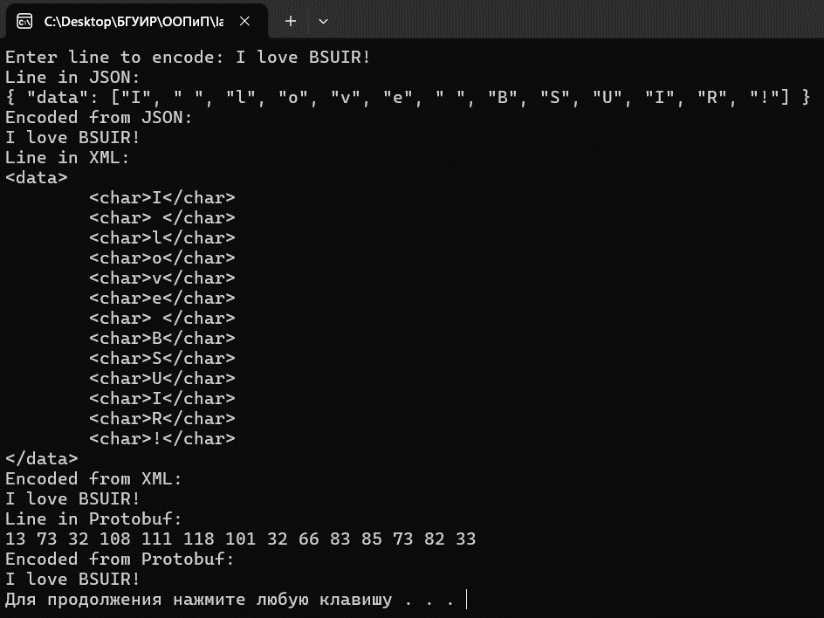
} while (choice == 1 );

std::cout << "Have a nice day!\n";

return 0;

}

**5 Скриншоты выполнения индивидуального задания**



**6 Ответы на контрольные вопросы**

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

Множественное наследование используется для создания класса, который наследует свойства и методы более чем одного родительского класса. Оно отличается от одиночного наследования тем, что позволяет классу комбинировать функциональность нескольких базовых классов.

1. Каков порядок вызова конструкторов и деструкторов при множественном наследовании?

При множественном наследовании конструкторы базовых классов вызываются в порядке их объявления в списке наследования производного класса. Деструкторы вызываются в обратном порядке после выполнения тела деструктора производного класса.

1. Каков порядок вызова конструкторов и деструкторов при множественном виртуальном наследовании?

При множественном виртуальном наследовании конструкторы виртуальных базовых классов вызываются перед конструкторами не виртуальных базовых классов, независимо от их порядка в списке наследования. Деструкторы вызываются в обратном порядке их конструкторов.

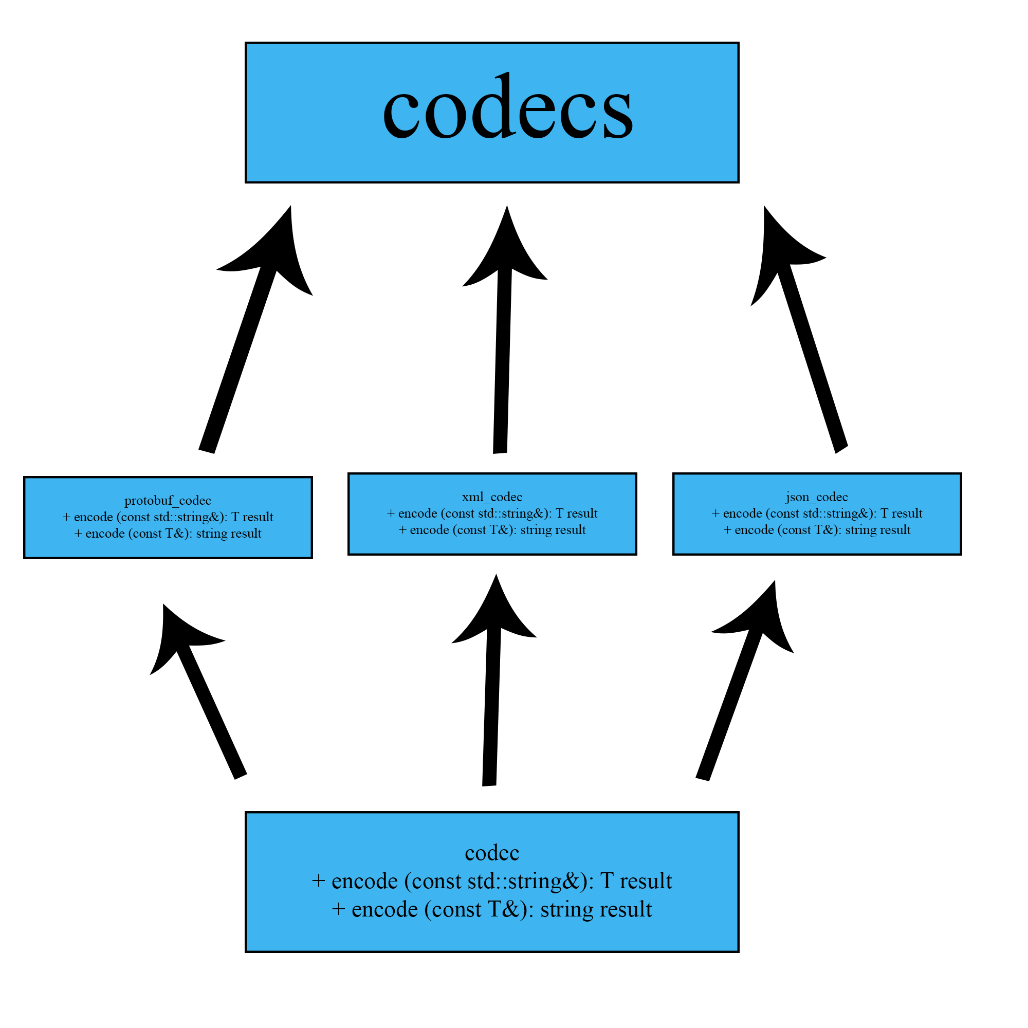
1. Какие проблемы возможны при множественном наследовании и как они разрешаются при использовании виртуального наследования?

Проблемы множественного наследования включают проблему ромбовидного наследования, когда два класса наследуют от одного базового класса, а затем третий класс наследует от обоих этих классов. Это может привести к неоднозначности и дублированию членов. Виртуальное наследование решает эти проблемы, создавая единственный экземпляр базового класса, который разделяется всеми производными классами.

1. Какие языки не поддерживают множественного наследования классов? Почему?

Некоторые языки программирования, такие как Java и C#, не поддерживают множественное наследование классов из-за сложностей, связанных с ромбовидным наследованием и неоднозначностями. Вместо этого они используют интерфейсы для достижения полиморфизма и повторного использования кода.

**7 Диаграмма всех созданных классов**



**8 Выводы по лабораторной работе**

Вывод: я изучил принципы и получил практические навыки при использовании множественного наследования, рассмотрел случаи, когда необходимо использовать множественное и виртуальное наследование.