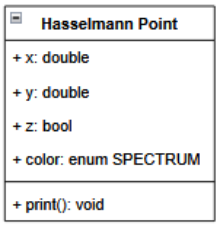
ВАРИАНТ 01: Перегрузка операций (100 баллов)

Для построения некоторой системы оптических точек в C++ определен тип **enum SPECTRUM** с тремя возможными значениями: **RED** и **BLUE**. Определен класс, называемый Hasselmann Point, определенный на диаграмме UML ниже:



[1]. Построить класс **Hasselmann Point**, конструктор по умолчанию, по параметрам, по копированию объекта, методы **setters** и **getters**. Учтите, что все атрибуты имеют привилегию доступа **private**. Определить функцию **print()** для отображения значений каждого атрибута.

[2]. Создайте 2 объекта класса **Hasselmann Point**. Сделайте перегрузку операции **+ и *.** Реализуйте глобальную функцию с привилегированным **friend**, вызываемым по значению (рассмотрите объекты **hp1** и **hp2** класса **Hasselmann Point**):

a) определить операцию + определяется как $result = hp1 + hp2$:

$result.x = \sqrt{(hp1.x)^2 + (hp2.x)^2}$; $result.y = \sqrt{(hp1.y)^2 + (hp2.y)^2}$; $result.z = hp1.z \,||\, hp2.z$; $result.color = hp1.color$

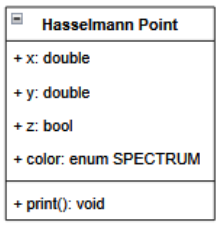
b) определить операцию * определяется как $result = hp1 * hp2$:

$result.x = (hp1.x)(hp2.x)$; $result.y = (h1.y)(h2.y)$ $result.z = hp1.z \,\&\&\, hp2.z$; $result.color = hp2.color$

c) Распечатать атрибуты результата операции + и ***.**

ВАРИАНТ 01: Перегрузка операций (100 баллов)

Для построения некоторой системы оптических точек в C++ определен тип **enum SPECTRUM** с тремя возможными значениями: **RED** и **BLUE**. Определен класс, называемый Hasselmann Point, определенный на диаграмме UML ниже:



[1]. Построить класс **Hasselmann Point**, конструктор по умолчанию, по параметрам, по копированию объекта, методы **setters** и **getters**. Учтите, что все атрибуты имеют привилегию доступа **private**. Определить функцию **print()** для отображения значений каждого атрибута.

[2]. Создайте 2 объекта класса **Hasselmann Point**. Сделайте перегрузку операции **+**. Реализуйте глобальную функцию с привилегированным **friend**, вызываемым по значению (рассмотрите объекты **hp1** и **hp2** класса **Hasselmann Point**):

a) определить операцию + определяется как $result = hp1 + hp2$:

$result.x = \sqrt{(hp1.x)^2 + (hp2.x)^2}$; $result.y = \sqrt{(hp1.y)^2 + (hp2.y)^2}$; $result.z = hp1.z \,||\, hp2.z$; $result.color = hp1.color$

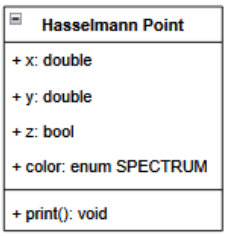
b) определить операцию * определяется как $result = hp1 * hp2$:

$result.x = (hp1.x)(hp2.x)$; $result.y = (h1.y)(h2.y)$ $result.z = hp1.z \,\&\&\, hp2.z$; $result.color = hp2.color$

c) Распечатать атрибуты результата операции **+ и *.**

ВАРИАНТ 01: Перегрузка операций (100 баллов)

Для построения некоторой системы оптических точек в C++ определен тип **enum SPECTRUM** с тремя возможными значениями: **RED** и **BLUE**. Определен класс, называемый Hasselmann Point, определенный на диаграмме UML ниже:



[1]. Построить класс **Hasselmann Point**, конструктор по умолчанию, по параметрам, по копированию объекта, методы **setters** и **getters**. Учтите, что все атрибуты имеют привилегию доступа **private**. Определить функцию **print()** для отображения значений каждого атрибута.

[2]. Создайте 2 объекта класса **Hasselmann Point**. Сделайте перегрузку операции **+**. Реализуйте глобальную функцию с привилегированным **friend**, вызываемым по значению (рассмотрите объекты **hp1** и **hp2** класса **Hasselmann Point**):

a) определить операцию + определяется как $result = hp1 + hp2$:

$result.x = \sqrt{(hp1.x)^2 + (hp2.x)^2}$; $result.y = \sqrt{(hp1.y)^2 + (hp2.y)^2}$; $result.z = hp1.z \,||\, hp2.z$; $result.color = hp1.color$

b) определить операцию * определяется как $result = hp1 * hp2$:

$result.x = (hp1.x)(hp2.x)$; $result.y = (h1.y)(h2.y)$ $result.z = hp1.z \,\&\&\, hp2.z$; $result.color = hp2.color$

b) Распечатать атрибуты результата операции **+ и *.**

ВАРИАНТ 02: Вектор объектов (100 баллов)

В некоторых экспериментах глубокого обучения вычислительно моделируется слой нейронов. Эксперимент состоит в определении 2 основных структур: нейрона и слоя нейронов.

• Нейрон

Нейрон определяется 3 входными значениями **(x, y, z),** связанными с вектором **W** синапса с 4 значениями, случайно сгенерированными между [1, 10]. Для генерации случайных значений рассмотрим включение кода:

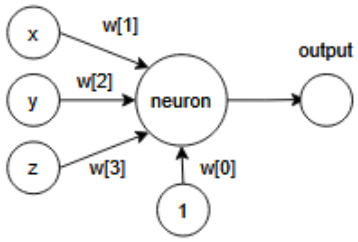
```
#include <ctime>

#include <cstdlib>

srand(static_cast<unsigned int>(time(0)));

generated_value = 1 + rand()%10;
```

Neuron определяется классом C++ с помощью атрибутов **x, y, z**, **W**-вектора и выходных данных (**output**), все с публичным доступом (**public**), следуя схеме ниже:



Атрибут **output** инициализируется нулем. Обновляется только при вычислении через функцию **fit()**, определенную выше:

$$output = x \cdot w[1] + y \cdot w[2] + z \cdot w[3] + w[0]$$

• Нейронный слой

Neuron Layer определяется классом C++, который имеет в качестве атрибутов: вектор объектов **Neuron**, **size** (сколько нейронных единиц имеет, в данном случае: рассмотрим статическое значение 3) и **global output** (глобальный выход). Атрибут **global output** инициализируется нулем и вычисляется просто вызовом функции **process_layer()**, определяемой формулой:

$$global\ output = \frac{1}{size}\sum_{i=1}^{3} output\_neuron_i$$

[1]. Построить классы **Neuron** и **Neuron Layer** и нарисовать диаграмму UML. Определить конструктор по умолчанию, конструктор по параметрам, конструктор по копии объекта. Все атрибуты для обоих классов имеют **public** доступ (не обязательно определять **setters**, **getters**). Атрибуты Neuron **x, y** и **z** определяются пользователем через конструктор. В main.cpp необходимо определить эти 2 объекта класса **Neuron Layer**: **layer1** и **layer2**.

[2]. Определить в классе **Neuron Layer** методы **set_vec_neurons()** и **print_layer()**. Первый метод требует вставки вектора объектов **Neurons** в определенный слой нейронов. Второй метод требует печать списка атрибутов для каждого нейрона слоя. Реализовать глобальную функцию для вычисления разницы между глобальными выходами **layer1** и **layer2**.

ВАРИАНТ 02: Вектор объектов (100 баллов)

В некоторых экспериментах глубокого обучения вычислительно моделируется слой нейронов. Эксперимент состоит в определении 2 основных структур: нейрона и слоя нейронов.

• Нейрон

Нейрон определяется 3 входными значениями **(x, y, z),** связанными с вектором **W** синапса с 4 значениями, случайно сгенерированными между [1, 10]. Для генерации случайных значений рассмотрим включение кода:

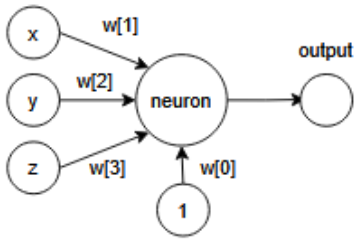
```
#include <ctime>

#include <cstdlib>

srand(static_cast<unsigned int>(time(0)));

generated_value = 1 + rand()%10;
```

Neuron определяется классом C++ с помощью атрибутов **x, y, z**, **W**-вектора и выходных данных (**output**), все с публичным доступом (**public**), следуя схеме ниже:



Атрибут **output** инициализируется нулем. Обновляется только при вычислении через функцию **fit()**, определенную выше:

$$output = x \cdot w[1] + y \cdot w[2] + z \cdot w[3] + w[0]$$

• Нейронный слой

Neuron Layer определяется классом C++, который имеет в качестве атрибутов: вектор объектов **Neuron**, **size** (сколько нейронных единиц имеет, в данном случае: рассмотрим статическое значение 3) и **global output** (глобальный выход). Атрибут **global output** инициализируется нулем и вычисляется просто вызовом функции **process_layer()**, определяемой формулой:

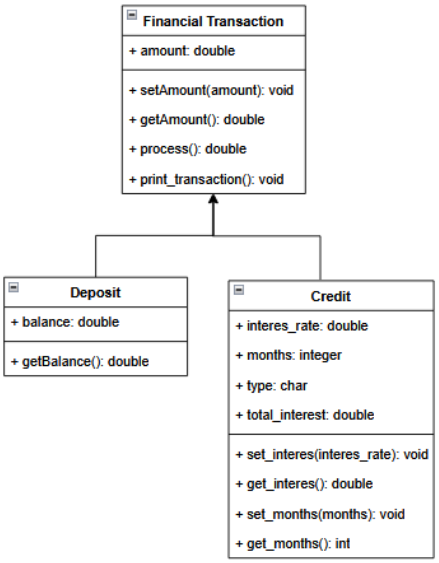
$$global\ output = \frac{1}{size}\sum_{i=1}^{3} output\_neuron_i$$

[1]. Построить классы **Neuron** и **Neuron Layer** и нарисовать диаграмму UML. Определить конструктор по умолчанию, конструктор по параметрам, конструктор по копии объекта. Все атрибуты для обоих классов имеют **public** доступ (не обязательно определять **setters**, **getters**). Атрибуты Neuron **x, y** и **z** определяются пользователем через конструктор. В main.cpp необходимо определить эти 2 объекта класса **Neuron Layer**: **layer1** и **layer2**.

[2]. Определить в классе **Neuron Layer** методы **set_vec_neurons()** и **print_layer()**. Первый метод требует вставки вектора объектов **Neurons** в определенный слой нейронов. Второй метод требует печать списка атрибутов для каждого нейрона слоя. Реализовать глобальную функцию для вычисления разницы между глобальными выходами **layer1** и **layer2**.

ВАРИАНТ 03: Наследование классов (100 баллов)

В Банке обрабатываются финансовые транзакции (**Financial Transaction**) на некоторую сумму денег в долларах. Существуют 2 типа транзакций: Депозит (**Deposit**) и Кредит (**Credit**). Существует связь простого наследования между классом **Финансовая транзакция** (базовый класс) и классами **Депозит** и **Кредит** (производные классы). Диаграмма UML выше определяет атрибуты и методы, необходимые для реализации в этих классах:



Следуйте следующим инструкциям:

[1]. Создайте классы, разработайте конструкторы, сеттеры и геттеры, следуя отношениям наследования. Учтите, что процентная ставка (**interest rate**) — это значение в интервале [0, 100]. Атрибуты класса **Financial Transaction** имеют **protected** привилегию доступа.

[2]. Реализуйте виртуальные функции **process()** — чисто виртуальную функцию и **print_transaction()** — простой виртуальный метод, учитывая следующие условия:

- метод **process()**

- для класса **Deposit**: обновить баланс, добавив сумму

$$balance += amount$$

- для класса **Credit**: рассчитать общую сумму ставку к оплате (**total interest**). Зависит от типа. Имеют 2 типа: A и B.

a) типа A:

$$total\ interest = amount + (amount * interest_{rate} * month)/100$$

b) типа B:

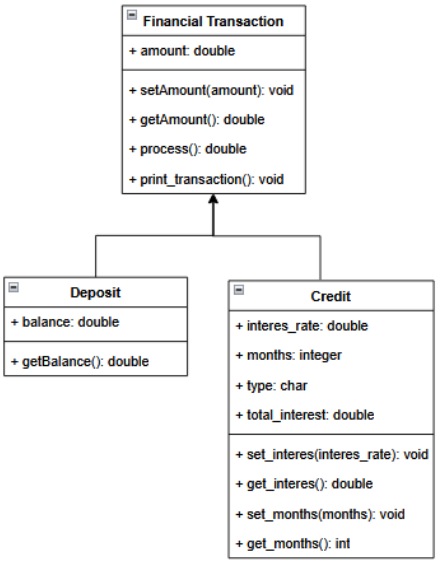
$$total\ interest = amount * (1 + interest_{rate}/100)^{month}$$

- метод **print_transaction()**

Распечатать в обоих производных классах все атрибуты, связанные с созданным объектом. В распечатку транзакции включить результаты, полученные в функции **process()** в зависимости от производного класса.

ВАРИАНТ 03: Наследование классов (100 баллов)

В Банке обрабатываются финансовые транзакции (**Financial Transaction**) на некоторую сумму денег в долларах. Существуют 2 типа транзакций: Депозит (**Deposit**) и Кредит (**Credit**). Существует связь простого наследования между классом **Финансовая транзакция** (базовый класс) и классами **Депозит** и **Кредит** (производные классы). Диаграмма UML выше определяет атрибуты и методы, необходимые для реализации в этих классах:



Следуйте следующим инструкциям:

[1]. Создайте классы, разработайте конструкторы, сеттеры и геттеры, следуя отношениям наследования. Учтите, что процентная ставка (**interest rate**) — это значение в интервале [0, 100]. Атрибуты класса **Financial Transaction** имеют **protected** привилегию доступа.

[2]. Реализуйте виртуальные функции **process()** — чисто виртуальную функцию и **print_transaction()** — простой виртуальный метод, учитывая следующие условия:

- метод **process()**

- для класса **Deposit**: обновить баланс, добавив сумму

$$balance += amount$$

- для класса **Credit**: рассчитать общую сумму ставку к оплате (**total interest**). Зависит от типа. Имеют 2 типа: A и B.

a) типа A:

$$total\ interest = amount + (amount * interest_{rate} * month)/100$$

b) типа B:

$$total\ interest = amount * (1 + interest_{rate}/100)^{month}$$

- метод **print_transaction()**

Распечатать в обоих производных классах все атрибуты, связанные с созданным объектом. В распечатку транзакции включить результаты, полученные в функции **process()** в зависимости от производного класса.