1 Задания по C++

1.1 Класс Лоренц-векторов

Реализуйте класс `LorentzVector`, описывающий Лоренц-векторы и содержащий

1. Конструктор по умолчанию
2. Конструктор из четырех переменных `double`
3. Методы чтения компонент вектора
4. Методы модификации компонент вектора
5. Метод вывода компонент в стандартный поток
6. Методы, реализующие операции
   - сложение векторов
   - вычитание векторов
   - скалярное произведение векторов
   - умножение вектора на число
   - нахождение нормы вектора
   - преобразование в систему, движущуюся со скоростью βc вдоль оси z.

Определение класса `LorentzVector` приведено в листинге 1.

Листинг 1: Определение класса LorentzVector

```
1  class LorentzVector {
2   public:
3      LorentzVector();
4      LorentzVector(double, double, double, double);
5
6      double t() const;
7      double x() const;
8      double y() const;
9      double z() const;
10     double norm() const;
11
12     void t(double x);
```

```
13      void x(double x);
14      void y(double x);
15      void z(double x);
16
17      LorentzVector add(const LorentzVector& other) const;
18      LorentzVector sub(const LorentzVector& other) const;
19      LorentzVector mul(double a) const;
20      void zboost(double beta);
21      double dot(const LorentzVector& other) const;
22      void read();
23      void print() const;
24  };
```

Указания:

- Определение классов стандартного вывода находятся в заголовочном файле `<iostream>`.
- Реализуйте инкапсуляцию данных (пользователь не знает, как конкретно хранятся данные в вашем классе, а все манипуляции с компонентами вектора могут производиться только с помощью методов класса).
- Объявляйте константными методы, которые не изменяют состояние объекта.
- Избегайте копирования при передаче вектора в качестве аргумента функции (метода). Используйте константные ссылки `const LorentzVector&`.
- Разделите реализацию класса на два файла: заголовочный файл с объявлением класса `LorentzVector.h` и файл с реализацией методов `LorentzVector.cpp`.

1.2 Перегрузка операторов для `LorentzVector`

Для класса `LorentzVector` перегрузите операторы

- Сложения (`operator+` и `operator+=`)
- Вычитания (`operator-` и `operator-=`)
- Унарного минуса (`operator-`)
- Умножения (`operator*` и `operator*=`)

- Вывода в стандартный поток
  `operator<<(std::ostream& out, const LorentzVector& lv)`

Новое определение класса приведено в листинге 2.

Листинг 2: Определение класса LorentzVector с перегруженными операторами

```
1  class LorentzVector {
2   public:
3      LorentzVector();
4      LorentzVector(double, double, double, double);
5
6      double t() const;
7      double x() const;
8      double y() const;
9      double z() const;
10     double norm() const;
11
12     void t(double x);
13     void x(double x);
14     void y(double x);
15     void z(double x);
16
17     void operator+=(const LorentzVector& other);
18     void operator-=(const LorentzVector& other);
19     void operator*=(double a);
20     LorentzVector operator+(const LorentzVector& other) const;
21     LorentzVector operator-(const LorentzVector& other) const;
22     friend LorentzVector operator*(const LorentzVector& lv,
          double a);
23     friend std::ostream& operator << (std::ostream&, const
          LorentzVector&);
24     void zboost(double beta);
25     double dot(const LorentzVector& other) const;
26 };
27
28 std::ostream& operator << (std::ostream&, const LorentzVector&);
```

Указания

- Создайте для этой задачи новый проект
- Оператор вывода в стандартный поток не следует включать в класс `LorentzVector`.

Если необходим доступ к приватным членам `LorentzVector`, можно объявить этот оператор с помощью ключевого слова `friend`.

1.3 Статистика массива

Часть I. Создайте класс `ArrayStat`, который считывает из файла массив $N \geq 0$ целых чисел $a_1, a_2, \ldots, a_N$. Файл имеет формат

```
N
a1 a2 ... aN
```

В классе должны быть реализованы следующие методы:

1. `int max()` — максимальное значение
2. `int min()` — минимальное значение
3. `double mean()` — среднее значение
4. `double rms()` — среднеквадратичное отклонение
5. `size_t countLarger(int a)` — количество элементов массива со значениями больше порога a
6. `void print()` — вывод элементов массива в стандартный поток вывода в порядке возрастания значений.

Метод `print()` должен работать за время $\mathcal{O}(N)$, а методы `max` и `min` — за время $\mathcal{O}(1)$.

Указания:

- Чтобы выполнить условия задания, необходимо хранить отсортированный массив
- Обрабатывайте с помощью стандартных исключений случай, если входной файл не существует или если формат данных в файле не соответствует условию
- Подумайте над особыми ситуациями, например, если $N = 0$ или $N = 1$
- Не используйте циклы в явном виде, замените их на функции, методы и алгоритмы из STL.

Часть II. Повторите задание для случая массива трехмерных векторов с классом `VecArrayStat`. Вместо значения элементов массива используйте декартову норму трехмерного вектора. Формат входного файла:

```
N
x1 y1 z1
x2 y2 z2
...
xN yN zN
```

Методы `max` и `min` должны возвращать значения `double`, соответствующие длинам векторов с наибольшей и наименьшей длиной, соответственно.

1.4 Баланс скобок

Последовательность скобок может быть сбалансирована или нет. Например, последовательность `[{([][])}[]]` — сбалансированная, а `{[])` — нет. Создайте класс `BraceChecker` со статическим методом `isBalanced`, который определяет, сбалансированы скобки в строке или нет:

```
1 class BraceChecker {
2  public:
3     static bool isBalanced(const std::string&);
4 }
```

Учитывайте три вида скобок: `{}`, `[]` и `()`. Программа должна считывать строчку из файла и выводить в стандартный поток `Balanced`, если скобки сбалансированы, и `Not balanced` в противном случае.

1.5 Иерархия классов

Разработайте иерархию классов с базовым классом и как минимум двумя наследниками. В базовом классе должен быть абстрактный виртуальный метод, реализованный в наследниках. Создайте гетерогенный список, содержащий объекты классов-наследников. Выведите объекты в стандартный вывод. Реализуйте сортировку списка по различным параметрам по возрастанию и по убыванию.

Пример иерархии: геометрические фигуры. Базовый интерфейс: `print()`, `area()`, `perimeter()`.

Указание: алгоритм `std::sort` может использовать произвольную функцию сравнения. Смотрите документацию.