Задача:

У далекій-далекій галактиці … поштова служба “Нова Галактика” повинна розвести вантажі по відповідним пунктам призначення. У розпорядженні поштової служби є тільки один вантажний космічний корабель. Корабель має обмежений простір для вантажу і певну вантажопідйомність. Під час подорожі в космосі, корабель і екіпаж витрачають ресурси (їжа, вода, паливо). Поповнити корабель ресурсами можна тільки на базі поштової служби.

Потрібно знайти такий маршрут перевезень вантажу, при якому корабель пролетить найменший шлях і доставить все вантажі у відповідні пункти призначення.

Вхідні дані:

Программа повинна бути виконана на C++. Клас, який проводить розрахунок шляху, повинен успадковуватися від IGalaxyPathFinder, який знаходиться в IGalaxyPathFinder.h.

Реалізована функція повинна приймати на вхід JSON-файл з наступною інформацією:

- Інформація про кораблі: габарити вантажного відсіку (половина по кожному виміру відповідно), максимально підйомна вага (вантаж + ресурси), максимальна вага ресурсів, витрати ресурсів на відстань
- Інформація про точки призначення: координати в просторі
- Інформація про кожний вантаж: габарити (половина по кожному виміру відповідно), вага, точка призначення

Форма вантажу і вантажного відсіку - прямокутний паралелепіпед. Вантаж і вантажний відсік не можна повертати у просторі.

Одиниці виміру: вага в тоннах, відстані в астрономічних одиницях (ao), розмір вантажу і вантажного відсіку в метрах, витрата палива в т/ao.

Приклади вхідного JSON в файлах: inputDataX.json

Типи даних та одиниці виміру:

```
{
   "ship":{
      "maxCarryingCapacity":{
         "half_x":0,                // int (м)
         "half_y":0,                // int (м)
         "half_z":0                 // int (м)
      },
      "maxResourcesWeight":0.0,     // float (т)
      "maxCarryingWeight":0.0,      // float (т)
      "resourcesConsumption":0.0    // float (т/ao)
   },
   "targetPoints":[
      {
         "y":0.0,                   // float (ao)
         "x":0.0,                   // float (ao)
         "z":0.0,                   // float (ao)
         "pointId":0                // int
      }
   ],
```

```
   "boxes":[
      {
         "half_x":0,                // int (м)
         "half_y":0,                // int (м)
         "half_z":0,                // int (м)
         "weight": 0.0,             // float (т)
         "targetPointId":0,         // int
         "boxId":0                  // int
      }
   ]
}
```

База має pointId == 0. Корабель стартує з цієї точки. Заправлятися і завантажувати вантаж можна тільки на базі. Сумарна вага вантажу і ресурсів не повинна перевищувати максимально підйомну вагу корабля.

Вихідні дані:

Реалізована функція повинна генерувати JSON-файл із записами переміщення корабля з однієї точки в іншу.

Вихідний JSON містить масив кроків, кожен крок містить наступну інформацію:

- Точка призначення;
- Скільки ресурсів завантажено (якщо виліт з бази); нуль, якщо поточний крок був зроблений не з бази;
- Який вантаж зараз доставляє корабель і як вантаж розташовується в вантажному відсіку. Розташування кожної коробки відносно центру вантажного відсіку корабля.

При поверненні на базу, ресурси, що залишилися, обнуляються. У кінці, коли весь вантаж було доставлено, корабель завжди повинен повертатися на базу.

Приклад вихідного JSON в файлі: outData.json

```
{
  "steps":[
    {
      "shippedBoxes":[
        {
          "boxId":0,                // int
          "x":0,                    // int (м)
          "y":0,                    // int (м)
          "z":0                     // int (м)
        }
      ],
      "shippedResources":0.0,       // float (т)
      "destinationPointId":0        // int
    }
  ]
}
```

Перевірка результатів:

Результат буде оцінюватися за кількома критеріями:

1. Розв’язок задачі - мінімальна сумарна відстань подорожі.
2. Швидкість виконання розрахунків.
3. Використання пам'яті - кількість виділень/звільнень, максимальне значення виділеної пам'яті.

Кожне рішення буде запускатися на великій кількості різних вхідних даних, у тому числі крайових/некоректних значеннях (наприклад, один з вантажів більше вантажного відсіку корабля). Програма повинна правильно обробляти відповідні вхідні дані, а саме: після доставки всіх вантажів і повернення на базу, останнім фіктивним кроком - "перевезення" всіх вантажів, які неможливо доставити, з бази на базу (destinationPointId == 0) без пакування (усі коробки з координатами (0,0,0)).

Для перевірки роботи програми буде використовуватися тільки cpp-файл з класом спадкоємцем IGalaxyPathFinder, який буде додано до загального проекту з усіма рішеннями. Дивись ExampleSolution.cpp.

Програма буде компілюватися за допомогою Visual Studio 2019 у схемі Release з оптимізацією “/O2”

З будь-якими питаннями звертайтеся до контактних осіб.
Контактні особи: Володимир (v.ivanov@dragonlk.com), Євгенія (e.beirak@dragonlk.com), Іван (i.kozmyk@dragonlk.com), Ольга (o.strukova@dragonlk.com)