# Руководство программиста

Программа состоит из 21 файла, среди которых 10 заголовочных файлов с расширением «.h» и 11 файлов с расширением «.cpp» (Таблица 1)

**Таблица 1**

|  |  |
| --- | --- |
| «.h» файл | «.cpp» файл |
| Transport.h | Transport.cpp |
| Ground.h | Ground.cpp |
| Train.h | Train.cpp |
| Bus.h | Bus.cpp |
| Flying.h | Flying.cpp |
| Plane.h | Plane.cpp |
| Airship.h | Airship.cpp |
| Poisk.h | Poisk.cpp |
| PrgList.h | PrgList.cpp |
| PrgMenu.h | PrgMenu.cpp |
|  | main.cpp |
|  |  |

## **Реализация «дерева объектов» в программе.**

Одной из задач курсовой работы было создание так называемого «дерева» объектов. Дерево данной курсовой работы представленно на рисунке 10. Главным классом (классом родителем) является верхний класс, а остальные классы (классы потомки) наследуют все методы и данные классов предков публичным способом наследования. При этом наследование идет сверху вниз (указано с помощью стрелок на рисунке).

«Дерево» объектов:

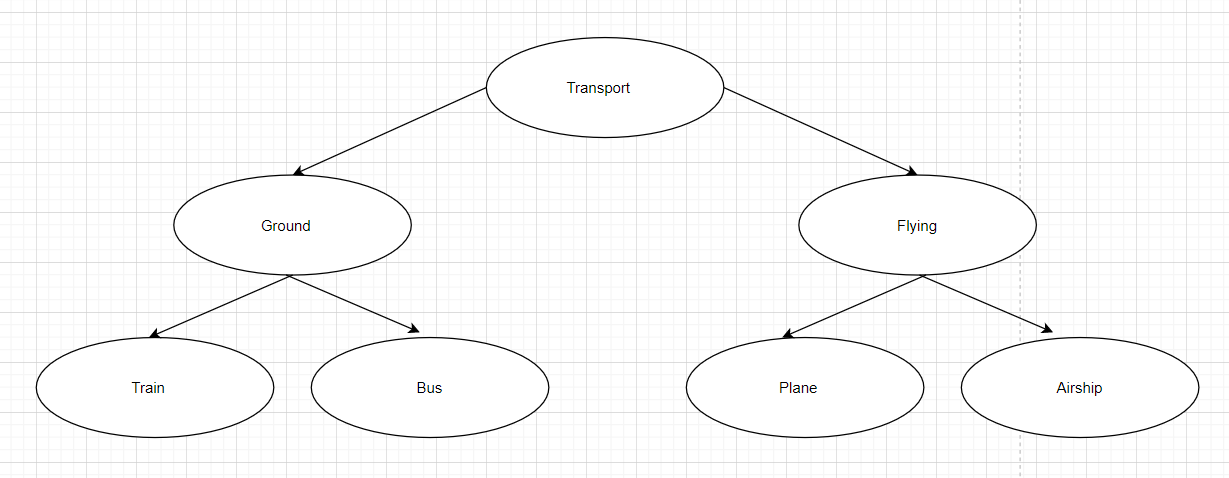


Рисунок № 7

Класс-родитель transport, который описан в одноименном файле Transport.h, содержит основные поля данных, соответствующие каждому двигателю:

private:

float power;//мощность двигателя

int year;//год выпуска

char manufacturer[100];//производитель

char country[100];//страна

При объявлении данных использовалось ключевое слово «private». Это означает, что данные класса transport закрытые и получить доступ к ним можно только при помощи специальных методов, описанных ниже:

float get\_power(){return power;};

int get\_year(){return year;};

const char\* get\_manufacturer(){const char\* s;

s=manufacturer;

return s;};

const char\* get\_country(){const char\* s;

s=country;

retrun s;};

В этом заключается принцип инкапсуляции, одного из свойств объектно-ориентированного программирования.

Кроме методов, представленных выше, класс transport содержит также ряд следующих методов:

Transport();//конструктор по умолчанию

virtual void information()=0;

void set\_power();

void set\_year();

void set\_manufacturer();

void set\_country();

Методы, служащие для обработки данных – виртуальные (определяются ключевым словом «virtual»), они могут быть использованы для обработки информации, содержащейся в объекте определенного класса.

К примеру, функция information, объявленная в базовом классе, для разных видов двигателей выглядит по-разному, то есть метод описывающий действие на поле один и тот же, но результат его выполнения различен для разных объектов.

Так, при вводе информации о поезде запрашивается тип вагонов, а при вводе информации об автобусе — тип двигателя. В этом заключается смысл полиморфизма в объектно-ориентированном программировании.

Также в классах нижнего уровня была реализована такая возможность языка C ++ как перегрузка оператора. Был перегружен оператор <<:

//перегрузка операции << в классе train

friend std::ostream& operator<< (std::ostream &out, Petrol &c)

{

out<<"\nМощность: "<<c.get\_power()<<"\nГод выпуска: "<<c.get\_year()<<"\nПроизводитель: "<<c.get\_manufacturer()

<<"\nСтрана: "<<c.get\_country()<<"\nДлина базы: "<<c.get\_base ()<<"\nОбъем: "<<c.get\_volume()

<<"\nВагоны:"<<c.get\_vags()<<std::endl;

return out;

}

## **Реализация списка в программе**

Еще одной задачей курсового проектирования было создание контейнера типа «список» и ряда функций для его обработки. Прототипы этих функций описаны в файле “PrgList.h”, а их реализация в файле “PrgList.cpp”.

int size();//возвращает длину списка

void clear();//удаление элемента

void addLast(T);//добавление элемента в конец списка

void delLast();//удаление последнего элемента

void delFirst(T);//удаление первого элемента

void Save\_List(PrgList<T> lst);//сохранение списка

int Load\_List(PrgList<T> &lst);//загрузка списка

void Print();//печать списка