**Руководство программисту**

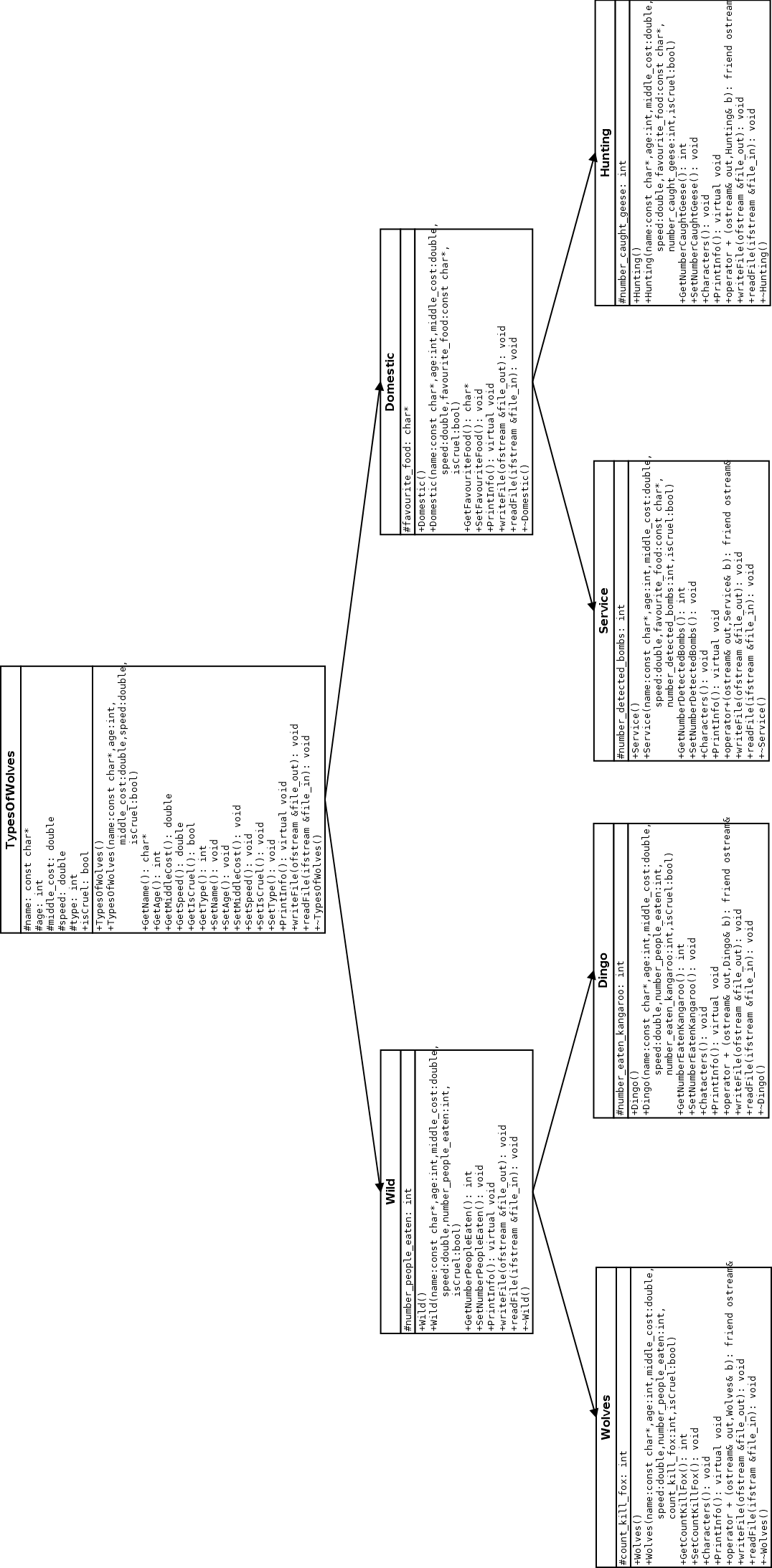
Исходный код программы состоит из 20 файлов, из которых 10 заголовочных файлов с расширением «.h» и 10 файлов с расширением «.cpp»(реализация методов классов и функционала программы). В таблице 1 приведен список файлов с комментариями о назначении того или иного файла.

**Таблица 1.**

|  |  |  |
| --- | --- | --- |
| **«.h» файл** | **«.cpp» файл** | **Комментарий** |
| TypesOfWolves.h | TypesOfWolves.cpp | Заголовочный файл и файл реализации класса TypesOfWolves |
| Wild.h | Wild.cpp | Заголовочный файл и файл реализации класса Wild |
| Domestic.h | Domestic.cpp | Заголовочный файл и файл реализации класса Domestic |
| Wolves.h | Wolves.cpp | Заголовочный файл и файл реализации класса Wolves |
| Dingo.h | Dingo.cpp | Заголовочный файл и файл реализации класса Dingo |
| Service.h | Service.cpp | Заголовочный файл и файл реализации класса Service |
| Hunting.h | Hunting.cpp | Заголовочный файл и файл реализации класса Hunting |
| List.h |  | Заголовочный файл и он же файл реализации класса List |
| Interface.h | Interface.cpp | Заголовочный файл и файл реализации меню программы |
| Node.h | Node.cpp | Заголовочный файл и файл реализации класса Node |
|  | Main.cpp | Файл содержащий функцию main() |

## Иерархия классов

При выполнении курсовой работы, согласно техническому заданию, была разработана иерархия классов в виде «дерева», реализующая единственное наследование.  
 Наследование в классах осуществляется с помощью спецификатора :**public**.



Класс TypesOfWolves является базовым классом. Данный класс имеет 2 подкласса – Wild и Domestic («сыновья» базового класса). Объекты этих классов не создаются и не используются, т.к. данные классы являются абстрактными. Классы Wild и Domestic имеют, в свою очередь, по 2 «сына» Wolves, Dingo и Service, Hunting соответственно, которые являются «внуками» базового класса TypesOfWolves. Пользователю в программе доступно создание и операции только с объектами типа: Wolves, Dingo, Service, Hunting

Поля данных, всех вышеперечисленных классов, размещены в секции protected: описания классов для обеспечения инкапсуляции данных. Т.е. доступ к полям данных классов возможен только из самого класса и классов-наследников. Доступ из вне возможен только с помощью функций-членов (интерфейса) класса, чем и обеспечивается принцип инкапсуляции.

Для обеспечения функциональности работы с объектами в каждом классе были созданы методы, обеспечивающие:

* заполнение полей данных

Например, в классе TypesOfWolves:

*void SetName();*

*void SetAge();*

*void SetMiddleCost();*

*void SetSpeed();*

*void SetIsCruel();*

*void SetType();*

* получение значений полей данных (все методы, имеющие приставку get\_).

Например, в классе TypesOfWolves:

*char\* GetName();  
int GetAge();*

*double GetMiddleCost();*

*double GetSpeed();*

*bool GetIsCruel();*

*int GetType();*

* отображение полей данных.

Например, в классе TypesOfWolves:

*virtual void PrintInfo() const;*

Функция ​ virtual void PrintInfo()​ ​ объявлена при помощи ключевого слова virtual, что означает, что она может быть определена где-то в других классах. В этом заключается принцип полиморфизма (в зависимости от вызываемого объекта функция будет иметь разную реализацию). Принципы построения других классов аналогичны.

* загрузка данных из файла.

Например, в классе TypesOfWolves:

*void readFile (std::ifstream &file\_in);*

* сохранение данных в файл.

Например, в классе TypesOfWolves:

*void writeFile (std::ofstream &file\_out);*

Также в данной работе также была реализована такая возможность языка как перегрузка оператора, например перегрузка оператора вывода на стандартное устройство.

*friend std::ostream& operator+(std::ostream& out, Dingo& b)*

# Список

Согласно техническому заданию, для реализации списка объектов было предложено использовать класс list (список). В данной курсовой работе реализован контейнер – односвязный список. Структура односвязного списка представлена на Рис.1.

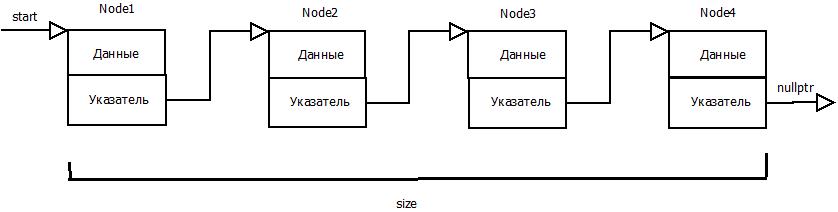


Рис.1 Структура односвязного списка.

Структура списка тривиальна и не требует особых пояснений. Необходимо отметить, что в данной реализации Список List (и узлы списка Node) оформлены в виде шаблонных классов, что делает код более универсальным (позволяет использовать любые типы данных).

Так же необходимо отметить, что в качестве данных, в узлах списка, мы будем хранить не объекты, а указатели на них: каждый элемент списка будет иметь указатель на следующий элемент. Данное решение позволит хранить в одном списке указатели на объекты производных классов (Wolves, Dingo, Service, Hunting), т.к. они имеют общий базовый тип – TypesOfWolves.

В классе List реализованы следующие методы работы со списком:

* загрузка списка из файла: *void download()*
* сохранение списка в файл: *void Output()*
* добавление элемента в список:  *void Add(T \*element)*
* удаление элемента из списка: *void deleteElement(int index)*
* поиск элемента в списке: *T findById(int index)*
* отображение полей данных элементов списка на экране: *void printAll()*
* отображение количества элементов в списке: *void printSize()*
* удаление элемента из начала списка: void *popFront()*