Дружественные классы и функции

Дружественные функции

- **Дружественные функции** – это функции, объявленные вне класса, но имеющие доступ к закрытым и защищенным полям данного класса.
  - Дружественная функция объявляется внутри класса с модификатором **`friend`**.
  - Дружественные функции не являются членами класса, поэтому им не передается указатель **`this`**.

Правила описания и особенности дружественных функций:

- 1) Дружественная функция объявляется внутри класса, к элементам которого ей нужен доступ, с ключевым словом **`friend`**.
- 2) В качестве параметра ей должен передаваться объект или ссылка на объект класса, поскольку указатель **`this`** данной функции не передается.
- 3) дружественная функция может быть обычной функцией или методом другого ранее определенного класса.
- 4) На дружественную функцию не распространяется действие спецификатора доступа, место размещения ее объявления в классе безразлично.
- 5) Одна функция может быть дружественной сразу несколькими классами.
- 6) Дружественная функция не наследуется.

Пример

```
class Foo
{
void friend Bar(Foo & foo);
private:
   int data;
};

void Bar(Foo & foo)
{
   foo.data = 1;
}
```

Дружественные операции

- Дружественные операции, как и дружественные функции, могут иметь доступ к закрытым и защищенным методам класса.

```
class CMyString
{
public:
    …
    CMyString const friend operator+(const char* left, CMyString const& right);
private:
};

CMyString const operator+(const char* left, CMyString const& right)
{
    …
}
```

Дружественные классы

- Некоторым классам может понадобиться доступ к закрытым данным друг друга
  - Например, классу «дерево» может понадобиться доступ к закрытым полям его узлов
  - В этом случае необходимо объявить дружественный класс внутри определения класса
- Дружественная связь между классами является самой сильной
  - Реализации классов оказываются связанными, что противоречит принципу инкапсуляции
  - **Не используйте дружественные классы до тех пор, пока их использование не окажется единственным способом решения задачи**

Пример

```
class Bar;

class Foo
{
friend class Bar;
private:
    int data;
};

class Bar
{
    void Do(Foo & foo)
    {
        foo.data = 1;
    }
};
```

Вложенные классы

Вложенное объявление классов и других типов данных

- Язык C++ позволяет разместить объявление одного класса (или другого типа данных) внутри объявления другого
  - Это полезно, когда вложенный тип данных в основном используется лишь внешним классом, или совместно с ним
    - Пример - итераторы стандартных контейнеров STL
- Использование вложенного класса
  - Из методов внешнего класса – по имени вложенного класса
  - Снаружи – при помощи указания имени внешнего класса:
    - ExternalClass::Internal

Пример 1

```
class External
{
public:
     class Internal
     {
     public:
          void Foo(){}
     };
private:
     void Bar()
     {
           // из методов внешнего класса можем обращаться по имени
          Internal internal;
          internal.Foo();
     }
};

int main(int argc, char* argv[])
{
     External::Internal internal;
     internal.Foo();
     return 0;
}
```