Лабораторная работа №2
по теме:"ДРУЗЬЯ КЛАССА"

Друг класса - это функция, которая не является членом этого класса, но которой доступны члены класса из закрытого (private) и защищенного (protected) разделов класса.

Друг класса, как и функции-члены класса, является частью интерфейса класса. Поэтому объявление друга класса должно быть включено в описание этого класса, и более того, оно может появиться только в описании класса.
Таким образом, привилегии функций-друзей определяются разработчиком класса, а не пользователем класса. Это исключает возможность появления "непрошенных друзей" и не приводит к разрушению механизма защиты.

Для объявления функции-друга нужно в описание класса поместить ее прототип, перед которым расположить ключевое слово friend. Например:

friend int foo(int, char*);

Являясь частью интерфейса класса, функция-друг класса тем не менее не является членом класса. Из этого вытекают несколько следствий.

Во-первых, неважно, в каком из разделов описания класса (private, protected или public) расположить объявления друзей. Обычно принято описания друзей группировать сразу после заголовка класса.

Во-вторых, в отличие от функций-членов, функции-другу при вызове не передается указатель this. Это значит, что при вызове функции-друга нужно явно указывать ей объект, для которого она вызывается, а обычные механизмы доступа к полям класса (через точку и "стрелочку") теряют смысл.

ПРИМЕР:

Рассмотрим две функции: member_set (функция-член) и friend_set (функция- друг). Они используются для установки значения защищенного поля данных 'a' некоторого класса Test.

```
class Test{
        int a;
        friend void friend_set(Test*,int);
    public:
        void member_set(int);
    };

void friend_set(Test* ptr,int Value)
{ ptr->a=Value;}     // нужно явно указать адреса объекта;
                     // this здесь не передается!

void Test::member_set(int Value)
{ a=Value;}          // фактически выполняется как  this->a=Value;

void foo(void)
{                    // применение
 Test x;
 friend_set(&x,10);
 x.member_set(10);
}
```

Функция может быть одновременно другом нескольких классов. Это может повысить эффективность и исключить нужду в специальных функциях-членах, выполняющих ту же роль и создаваемых в каждом из классов.

Если в дружественной объявляется функция, которая используется в программе как перегружаемая, то другом становится лишь функция с конкретно заданными типами параметров. Т.е., нужно описывать как друга каждый вариант перегружаемой функции, который нужно сделать дружественным.

Чтобы сделать друзьями функции-члены некоторого класса, достаточно объявить дружественным это класс:

```
class B{
        friend class Test;   // служебное слово class можно опускать
        ... };
```

Объявление целого класса дружественным предполагает, что все закрытые и защищенные имена (private и protected) класса, предоставляющего дружбу, могут использоваться функциями, получающими эту привилегию. Например,

```
class Test{
        friend class B;
        int a;
       };
 class B {
        void foo(Test& t)
        { cout << t.a ; }
       };
```

Дружественным можно объявлять класс, который еще не был описан или даже не был объявлен. Это же относится и к функциям, которые могут объявляться друзьями еще до появления их прототипа или описания.

Для того чтобы определить взаимную дружбу двух классов, нужно в каждом из них объявить дружественным другой.

Дружественость не является транзитивным свойством: если класс Y является другом класса X, а класс Z является другом класса Y, то класс Z не является другом X, если только это не объявлено явно.

ЗАДАНИЕ 1

1. Описать класс Test с защищенными числовым полем W и функцией Z, которая выводит сообщение "Это закрытая функция класса Test".

Написать конструктор для инициализации объектов класса Test с одним параметром, принимающим по умолчанию значение 1.

Объявить другом класса функцию fun, которая не возвращает значений и получает указатель на объект типа Test.

2. Описать на внешнем уровне функцию fun, которая выводит на экран значение параметра W и вызывает из класса Test функцию Z.

3. В функции main описать переменную класса Test (без явной инициализации) и применить к ней функцию fun.

ЗАДАНИЕ 2

1. Реализовать класс Abonent, который содержит следующую информацию о владельце телефона: индивидуальный (уникальный) номер, фамилию, номер телефона. Для инициализации полей класса использовать конструктор с параметрами. Определить деструктор. Предусмотреть вывод информации о времени вызове конструктора и деструктора.
2. Реализовать класс Notebook (Записная книжка) для работы со списком абонентов. Объявить созданный класс дружественным классу Abonent.

В классе Notebook определить открытые функции ***change()*** для редактирования номера телефона абонента и ***show()*** для отображения информации об абоненте.

3. В функции ***main()***создать массив объектов класса Abonent (минимум 5 объектов).

Продемонстрировать работу функции ***change()***класса Notebook**.** Найдите нужного абонента по индивидуальному индексу и измените его телефонный номер.

С помощью функции ***show()*** класса Notebook вывести информацию обо всех абонентах в виде таблицы.