Практическое занятие №2.
Тема: Разработка дружественных функций и декомпозиция проекта программы на модули

Цель: получение практических навыков разработки дружественных функций и формирования проекта программы из нескольких частей.

Дружественные функции

При программировании возникают ситуации, когда требуется функция, которая имела бы доступ к закрытым членам класса, но сама не являлась бы членом этого класса. Для этого введены дружественные функции.

Дружественная функция определяется в программе как обычная функция. Чтобы показать, к какому классу она дружественна, её объявляют в этом классе с ключевым словом **friend**. Дружественная функция – это обыкновенная функция, у которой в параметрах указываются переменные различных типов. Поэтому в качестве **параметра** у дружественной функции должен указываться **объект** (переменная) со спецификатором дружественного класса.

Пример использования дружественной функции.

```
#include<iostream>
using namespace std;

class data
{ int a,b;
public:
     data(int x,int y)// конструктор с параметрами x и y
     { a=x; b=y;}
     friend int diff(data ob);  // дружественная функция
};

int diff(data ob)     // описание дружественной функции
{ return ob.a-ob.b;}

int main()
{ int d;
  data obj(8,5);
  d=diff(obj);    // вызов дружественной функции
  cout<<"Difference = "<<d;
return 0;
}
```

В классе **data** объявлены две переменные **a** и **b,** а также конструктор с параметрами. Кроме того, объявлена дружественная функция **diff(data ob)**.

Необходимо обратить внимание на порядок её объявления: сначала записывается ключевое слово **friend**, затем указывается тип возвращаемого значения функции, после чего – имя функции и в скобках – класс и имя объекта. В главной функции создается объект; при этом запускается конструктор и инициализируются переменные **a** и **b**. Созданный объект передается в функцию diff( ), которая вычисляет разность **a** и **b**. Разность возвращается переменной **d** и затем выводится на экран.

Обычно дружественная функция полезна тогда, когда у двух разных классов имеется нечто общее, что необходимо сравнить. Например, в следующей программе создаются классы **car** (легковая машина) и **truck** (грузовик), причем оба содержат в закрытой переменной скорость соответствующего транспортного средства.

В этой программе имеется функция **sp_greater(),** которая дружественна для классов **car** и **truck**. Эта функция возвращает положительное число, если объект **car** движется быстрее объекта **truck**, нуль, если их скорости одинаковы, и отрицательное число, если скорость объекта **truck** больше, чем скорость объекта **car.**

Эта программа иллюстрирует один важный элемент синтаксиса C++ — предварительное объявление, которое еще называют ссылкой вперед. Поскольку функция **sp_greater()** получает параметры обоих классов **car** и **truck**, то логически невозможно объявить и тот и другой класс перед включением функции **sp_greater()** в каждый из них. Поэтому необходим иной способ сообщить компилятору имя класса без его фактического объявления. Этот способ и называется предварительным объявлением.

В C++, чтобы проинформировать компилятор о том, что данный идентификатор является именем класса, перед первым использованием имени класса делают предварительное объявление. В нижеследующем примере это -

```
class truck;
```

после которого класс **truck** можно использовать в объявлении дружественной функции **sp_greater()** без опасения вызвать ошибку компилятора.

```
#include <iostream>
using namespace std;

class truck; // предварительное объявление
class car {
int passengers;
int speed;
public:
car(int p, int s) { passengers = p; speed = s; }
friend int sp_greater (car c, truck t);
};

class truck {
```

```
int weight;
int speed;
public:
truck(int w, int s) { weight = w; speed = s; }
friend int sp_greater (car c, truck t) ;
};

// описание дружественной функции
int sp_greater (car c, truck t)
{
return c.speed – t.speed;
}

int main()
{
int t;
car c1(6, 55), c2(2, 120) ;
truck t1(10000, 55), t2(20000, 72);
cout << "Сравнение значений c1 и t1:\n";
t = sp_greater (c1, t1);
if(t<0)     cout << "Грузовик быстрее. \n";
else if (t==0)  cout<<"Скорости машин одинаковы. \n";
else cout << "Легковая машина быстрее. \n";
cout << "\n Сравнение значений c2 и t2:\n";
t = sp_greater(c2, t2);
if(t<0)     cout << "Грузовик быстрее. \n";
else if(t==0) cout<<"Скорости машин одинаковы. \n";
else cout << "Легковая машина быстрее. \n";
return 0;
}
```

Сборка проекта из нескольких файлов

Для удобства работы с программой её разбивают на несколько частей (файлов). Файлы различают:

- заголовочные, которые обозначаются с расширением «**.h** »;
- исходного кода, которые обозначаются с расширением «**.cpp**».

Главная функция помещается в файл с расширением «**.cpp**». К нему в тексте программы с помощью директивы **#include** подключаются остальные файлы, например,

#include ”Classes.h”.

Имя файла указывается в кавычках (а не в угловых скобках), чтобы не путать с библиотечными файлами.

Практикум

Задание 1. Спроектировать два класса в соответствии с индивидуальным заданием (см. Приложение), содержащие каждый одну-две закрытые переменные, функции доступа к ним, конструкторы и деструктор. Создать массив объектов для каждого класса, задав его размер с клавиатуры. В главной функции определить *Вычисляемый показатель*.

Задание 2. Доработать проект *Задания 1,* добавив дружественную функцию к классам, определяющую *Вычисляемый показатель*. Создать массив объектов для каждого класса и получить значение *Вычисляемого показателя*.

Задание 3. Разработанный проект представить в виде трёх файлов: два заголовочных – для классов и один – с функцией **`main()`**.

Отчет оформляется по общеустановленным правилам в ***электронном виде*** со следующим содержанием:

1) титульный лист,
2) тема и цель практического занятия,
3) задание на практическое занятие,
4) текст программы с комментариями,
5) результаты работы программы и
6) выводы по разработанным элементам программы.

Приложение

№ п/п	Классы	Вычисляемый показатель
1.	Настольный компьютер, ноутбук	Экземпляр с наибольшей оперативной памятью
2.	Одноранговая компьютерная сеть, сеть типа клиент-сервер	Минимальная стоимость монтажа
3.	Легковой автомобиль, грузовой автомобиль	Максимальный пробег на полном бензобаке
4.	Операционная система, текстовый редактор	Последняя версия
5.	Паспорт, студенческий билет	Количество документов, выданных в прошлом году
6.	Диета для здоровья, диета для похудения	Максимальное дневное количество белков
7.	Клавиатура, мышь	Минимальная цена устройства
8.	Сыпучие материалы, инструмент	Суммарная стоимость
9.	Преподаватель, студент	Количество студентов, обучающихся у конкретного преподавателя
10.	Магазин, парикмахерская	Название предприятия с максимальным числом сотрудников
11.	СУБД : реляционная, иерархическая	Количество СУБД заданного производителя
12.	Полис обязательного медицинского страхования, полис страхования жилища	Количество полисов на заданную фамилию
13.	Проводные наушники, беспроводные наушники	Тип с максимальным частотным диапазоном
14.	Таунхаус, квартира в многоквартирном доме	Максимальная жилая площадь
15.	Электронные часы, механические часы	Самый дорогой экземпляр
16.	Процедурный язык программирования, объектно-ориентированный язык программирования	Последний по году разработки язык программирования
17.	Телевизор, холодильник	Количество экземпляров заданной фирмы
18.	Карта общего назначения для проезда в метро, льготная транспортная карта учащегося	Количество карт без поездок
19.	Стрекоза, бабочка	Максимальный размер крыльев
20.	Бумага, авторучка	Количество товаров заданной фирмы
21.	Посуда, продукты питания	Суммарная стоимость
22.	Школьник, студент	Учащийся с максимальным IQ (коэффициентом интеллекта)
23.	Морской круиз, отдых на море	Самый дешевый тур на 7 и более дней

24.	Кровать, стол	Количество предметов из дерева
25.	Системное ПО, прикладное ПО	Представители с максимальным и минимальным объемом занимаемой памяти
26.	Спектакль (театр), сеанс (кино)	Минимальное число зрителей в зале
27.	Флэш-карта, карта памяти	Экземпляры с максимальным и минимальным размером
28.	Струйный, лазерный	Представитель с наибольшей производительностью
29.	Книга: электронная, на бумаге	Количество книг одного автора
30.	Телефон: смартфон, кнопочный	Самая новая модель