Лабораторная работа №1

Классы и объекты. Инкапсуляция.

1. Цель задания:

1) Создание консольного приложения, состоящего из нескольких файлов в системе программирования Visual Studio.
2) Использование классов и объектов в ОО программе.

2. Теоретические сведения

2.1. Классы, объекты и их представление в ОО программе

Класс является абстрактным типом данных, определяемым пользователем, и представляет собой модель реального объекта в виде данных и функций для работы с ними.

Данные класса называются *полями* (по аналогии с полями структуры) или *атрибутами*, а функции класса — *методами.* Поля и методы называются *элементами класса.*

Описание класса в первом приближении выглядит так:

```
class <имя>
{
[ private: ]
<описание скрытых элементов>
public:
<описание доступных элементов>
}; // Описание заканчивается точкой с запятой
```

Спецификаторы доступа `private` и `public` управляют видимостью элементов класса. Элементы, описанные после служебного слова `private`, видимы только внутри класса. Этот вид доступа принят в классе по умолчанию. Интерфейс класса описывается после спецификатора `public`. Действие любого спецификатора распространяется до следующего спецификатора или до конца класса. Можно задавать несколько секций `private` и `public`, порядок их следования значения не имеет.

Получить информацию о содержимом полей, описанных после спецификатора `private` можно только с помощью специальных методов, которые называются селекторами, а изменить – с помощью методов, которые называются модификаторами.

Видимостью элементов класса можно также управлять с помощью ключевых слов `struct` и `class`. Если при описании класса используется слово `struct`, то все поля и методы по умолчанию будут общедоступными (`public`). Если при описании класса используется слово `class`, то по умолчанию все методы и поля класса будут скрытыми `(private)`.

Все методы класса имеют непосредственный доступ к его скрытым полям.

Свойства полей класса:

- могут иметь любой тип, кроме типа этого же класса (но могут быть указателями или ссылками на этот класс);
- могут быть описаны с модификатором `const`, при этом они инициализируются только один раз (с помощью конструктора) и не могут изменяться;
- Инициализация полей при описании не допускается.
- Если тело метода определено внутри класса, он является встроенным (`inline`). Как правило, встроенными делают короткие методы. Если внутри класса записано только объявление (заголовок) метода, сам метод должен быть определен в другом месте

программы с помощью операции доступа к области видимости (::):

```
class Person
{
string name;//имя
int age;//возраст
public:
void Set_Person(string Name,int Age);//модификатор
void Set_Age(int Age);//модификатор
string Get_Person (){return name;}//селектор
int Get_Age(){return age;}//селектор
void Print_Person()//встроенная функция печати
{
cout<<p.name<<" "<<p.age<<"\n";
}
...
};
//определение методов класса
void Person::Set_Person(string Name,int Age);
{
Name=name;
age=Age;
}
```

Переменная класса Person называется экземпляром класса или объектом. Класс объявляется один раз, а переменных такого класса (объектов) может быть сколько угодно. Объявляются такие переменные так же как и переменные встроенных типов:

```
Person p;
Person *pp;//указатель на переменную
Person Arr_p[10];//массив переменных
```

Обращаться к полям и методам класса можно точно также как к полям структуры:

```
Strcpy(p.name,"Ivanov");
p.age=20;
p.Set_Person("Ivanov",20);
pp=new(Person);
pp->Set_Person("Petrov",21);
Arr_p[1].Set_Person("Sidorov",22);
```

3. Постановка задачи

1. Реализовать определение нового класса. Для демонстрации работы с объектами написать главную функцию. Продемонстрировать разные способы создания объектов и массивов объектов.
2. Структура-пара – структура с двумя полями, которые обычно имеют имена first и second. Требуется реализовать тип данных с помощью такой структуры. Во всех заданиях должны присутствовать :
    a. метод инициализации Init (метод должен контролировать значения аргументов на корректность);
    b. ввод с клавиатуры Read;
    c. вывод на экран Show.
3. Реализовать внешнюю функцию make_тип(), где тип – тип реализуемой структуры. Функция должна получать значения для полей структуры как

параметры функции и возвращать структуру как результат. При передаче ошибочных параметров следует выводить сообщение и заканчивать работу.

4. Ход работы

Задача

Поле first – дробное число; поле second – целое число, показатель степени.
Реализовать метод power() – возведение числа first в степень second. Метод должен правильно работать при всех значениях first и second

1. Создать пустой проект. Для этого требуется
   1.1. Запустить MS Visual Studio:
   1.2. Выбрать команду File/New/Project
   1.3. В окне New Project выбрать Win Console 32 Application, в поле Name указать имя проекта (Lab1), в поле Location указать место положения проекта (личную папку), нажать кнопку Ok.
   1.4. В следующем окне выбрать кнопку Next.
   1.5. В следующем окне выбрать кнопку Next.
   1.6. В диалоговом окне Additional Settings установить флажок Empty (Пустой проект) и нажать кнопку Finish.
   1.7. В результате выполненных действий получим пустой проект.
2. Добавим в проект файл fraction.h, содержащий описание класса. Для этого нужно:
   2.1. Вызвать контекстное меню проекта в панели Обозреватель решений (Solution Explorer), выбрать в нем пункт меню Add/ New Item.
   2.2. В диалоговом окне Add New Item – Lab1 выбрать Категорию Code, шаблон – Header File (.h), задать имя файла fraction. В результате выполненных действий получим пустой файл fraction.h.
   2.3. Ввести следующий текст программы:

```
struct fraction
{
        double first;
        int second;
        void Init(double, int);//метод для инициализации полей
        void Read();//метод для чтения значений полей
        void Show();//метод для вывода значений полей
        double Power();//вычисление степени
};
```

3. Добавим в проект файл fraction.cpp, содержащий описание методов класса fraction. Для этого нужно:
   3.1. Вызвать контекстное меню проекта в панели Обозреватель решений (Solution Explorer), выбрать в нем пункт меню Add/ New Item.
   3.2. В диалоговом окне Add New Item – Lab1 выбрать Категорию Code, шаблон – C++File (.cpp), задать имя файла fraction. В результате выполненных действий получим пустой файл fraction.cpp.
   3.3. Ввести следующий текст программы:

```
#include <iostream>
using namespace std;
//реализация метода для инициализации полей структуры
void fraction::Init(double F, int S)
{
first=F; second=S;
}
//реализация метода для чтения значений полей структуры
void fraction::Read()
```

```
{
cout<<"\nfirst?"; cin>>first;
cout<<"\nsecond?";cin>>second;
}
//реализация метода для вывода значений полей структуры
void fraction::Show()
{
cout<<"\nfirst="<<first;
cout<<"\nsecond="<<second;
cout<<"\n";
}
//метод для возведения в степень
double fraction::Power()
{
return pow (first, second);
}
```

4. Добавим в проект файл Lab1_main.cpp, содержащий основную программу. Для этого нужно:
   - 4.1. Вызвать контекстное меню проекта в панели Обозреватель решений (Solution Explorer), выбрать в нем пункт меню Add/ New Item.
   - 4.2. В диалоговом окне Add New Item – Lab1 выбрать Категорию Code, шаблон – C++File (.cpp), задать имя файла Lab1_main. В результате выполненных действий получим пустой файл Lab1_main.cpp, в котором будет редактироваться текст программы.
   - 4.3. Ввести следующий текст программы:

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

fraction make_fraction(double F, int S)
{
        fraction t;//создали временную переменную
        t.Init(F,S);//инициализировали поля переменной t с помощью параметров функции
        return t;//вернули значение переменной t
}
```

5. Изменить текст функции main()

```
void main()
{
//определение переменных A и B
        fraction A;
        fraction B;
        A.Init(3.0,2);//инициализация переменной A
        B.Read();//ввод полей переменных B
        A.Show();//вывод значений полей переменной A
        B.Show();//вывод значений полей переменной B
//вывод значения степени, вычисленного с помощью функции Power()
        cout<<"A.Power("<<A.first<<","<<A.second<<")="<<A.Power()<<endl;
        cout<<"B.Power("<<B.first<<","<<B.second<<")="<<B.Power()<<endl;
//указатели
        fraction *X=new fraction;//выделение памяти под динамическую переменную
        X->Init(2.0,5);//инициализация
        X->Show();//вывод значений полей
        X->Power();//вычисление степени
```

```
        cout<<"X.Power("<<X->first<<","<<X->second<<")="<<X->Power()<<endl;
//массивы
        fraction mas[3];//определение массива
        for (int i=0;i<3; i++)
                mas[i].Read();//чтение значений полей
        for (i=0;i<3; i++)
                        mas[i].Show(); //вывод значений полей
        for (i=0;i<3; i++)
        {
                mas[i].Power(); //вычисление степени
                cout<<"mas["<<i<<"].Power("<<mas[i].first<<","<<mas[i].second<<")=";
                cout<<mas[i].Power()<<endl;
        }
//динамические масивы
        fraction* p_mas=new fraction[3];//выделение памяти
        for (int i=0;i<3; i++)
                p_mas[i].Read();//чтение значений полей
        for (i=0;i<3; i++)
                p_mas[i].Show();//вывод значений полей

        for (i=0;i<3; i++)
        {
                p_mas[i].Power();//вычисление степени
                cout<<"p_mas["<<i<<"].Power("<<p_mas[i].first<<","<<p_mas[i].second;
                cout<<")="<<p_mas[i].Power()<<endl;
        }
//вызов функции make_fraction()
        double y; int z;
        cout<<"first?";cin>>y;
        cout<<"second?";cin>>z;
//переменная F формируется с помощью функции make_fraction()
        fraction F=make_fraction(y,z);
        F.Show();
        return 0;
}
```

6. Выполнить компиляцию программы, используя команду Build / Build Solution или функциональную клавишу F7. Исправить имеющиеся синтаксические ошибки и снова запустить программу на компиляцию.
7. После появления в окне вывода сообщения
   1>lab1 - 0 error(s), 0 warning(s)
   ========== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========
   запустить программу на выполнение, используя команду Debug/ Start Without Debugging или комбинацию функциональных клавиш Ctrl+F5.
8. Изучить полученные результаты, сделать выводы.

9. Варианты

№	Задание
1	Поле first – положительное целое число, числитель, поле second – положительное целое число, знаменатель. Реализовать метод ipart() – выделение целой части дроби first/second, метод должен проверять неравенство знаменателя нулю.

2	Поле first – положительное целое число, номинал купюры; номинал может принимать значения 1, 2, 5. 10, 50, 100, 500, 1000, 5000, поле second – положительное целое число, количество купюр данного достоинства. Реализовать метод summa() – вычисление денежной суммы.
3	Поле first – положительное целое число, цена товара, поле second – положительное целое число, количество единиц данного товара. Реализовать метод cost() –вычисление стоимости данного товара.
4	Поле first – дробное число, левая граница диапазона, поле second – дробное число, правая граница диапазона. Реализовать метод rangecheck(double x) – проверку принадлежности заданного числа x на принадлежность диапазону [first;second].
5	Поле first – положительное целое число, часы, поле second – положительное целое число, минуты. Реализовать метод minutes() – приведение времени в минуты.
6	Линейное уравнение y=Ax+B. Поле first – дробное число, коэффициент A, поле second – дробное число, коэффициент B. Реализовать метод root() –вычисление корня линейного уравнения, метод должен проверять неравенство коэффициента A нулю.
7	Линейное уравнение y=Ax+B. Поле first – дробное число, коэффициент A, поле second – дробное число, коэффициент B. Реализовать метод function(double x) –вычисление значения y для заданного x.
8	Поле first – дробное число x, координата точки , поле second – дробное число y, координата точки. Реализовать метод distance() – вычисление расстояния от точки с координатами (first, second) до начала координат.
9	Поле first – дробное число x, координата точки , поле second – дробное число y, координата точки. Реализовать метод distance(double x1, doudle y1) – вычисление расстояния от точки с координатами (first, second) до точки с координатами (x1,y1).
10	Поле first – дробное положительное число, катет a прямоугольного треугольника, поле second – дробное положительное число, катет b прямоугольного треугольника. Реализовать метод hipotenuse() –вычисление гипотенузы.
11	Поле first – дробное положительное число, оклад, поле second – целое положительное число, количество отработанных дней. Реализовать метод summa() –вычисление начисленной суммы за данное количество дней по формуле: оклад/количество_дней_месяца*количество_отработанных _дней
12	Поле first – целое положительное число, продолжительность телефонного разговора в минутах, поле second – дробное положительное число, стоимость одной минуты разговора в рублях. Реализовать метод cost() –вычисление общей стоимости разговора
13	Поле first – положительное целое число, целая часть числа, поле second – положительное дробное число, дробная часть числа. Реализовать метод multiply(double k) – умножение на вещественное число k.
14	Поле first – положительное целое число, целая часть числа, поле second – положительное дробное число, дробная часть числа. Реализовать метод multiply(int k) – умножение на целое число k.
15	Элемент арифметической прогрессии aj вычисляется по формуле: $a_j=a_0 \cdot r^j$.Поле first – дробное число, первый элемент прогрессии a_0, поле second – положительное целое число, постоянное отношение r. Реализовать метод element(int j) –вычисление j-го элемента прогрессии.

7. Контрольные вопросы

1. Что такое класс?
2. Что такое объект (экземпляр) класса?

3. Как называются поля класса?
4. Как называются функции класса?
5. Для чего используются спецификаторы доступа?
6. Для чего используется спецификатор public?
7. Для чего используется спецификатор private?
8. Если описание класса начинается со спецификатора class, то какой спецификатор доступа будет использоваться по умолчанию?
9. Если описание класса начинается со спецификатора struct, то какой спецификатор доступа будет использоваться по умолчанию?
10. Какой спецификатор доступа должен использоваться при описании интерфейса класса? Почему?
11. Каким образом можно изменить значения атрибутов экземпляра класса?
12. Каким образом можно получить значения атрибутов экземпляра класса?
13. Класс описан следующим образом

```
struct Student
{
string  name;
int group;
……
};
```

Объект класса определен следующим образом

```
Student *s=new Student;
```

Как можно обратиться к полю name объекта s?

14. Класс описан следующим образом

```
struct Student
{
string name;
int group;
……..
};
```

Объект класса определен следующим образом

```
Student s;
```

Как можно обратиться к полю name объекта s?

15. Класс описан следующим образом

```
class Student
{
string name;
int group;
…….
};
```

Объект класса определен следующим образом

```
Student *s=new Student;
```

Как можно обратиться к полю name объекта s?

16. Класс описан следующим образом

```
class Student
{
string name;
int group;
public:
```

…..
};
Объект класса определен следующим образом
Student s;
Как можно обратиться к полю name объекта s?

17. Класс описан следующим образом
```
class Student
{
public:
char* name;
int group;
……
};
```
Объект класса определен следующим образом
Student *s=new Student;
Как можно обратиться к полю name объекта s?

6. Содержание отчета

1) Постановка задачи (общая и конкретного варианта).
2) Описание класса.
3) Определение компонентных функций.
4) Определение функции make_().
5) Объяснение результатов работы программы.
6) Ответы на контрольные вопросы