Лабораторная 1. Реализация классов

1. Цель работы

Изучить структуру класса, механизм создания и использования, описание членов-данных класса и методов доступа к ним.

2. Краткие теоретические сведения

Компоненты класса

Класс - это определяемый пользователем тип. Описание класса очень похоже на описание структуры в Си. В этом смысле класс является расширением понятия структуры. В простейшем случае класс можно определить с помощью конструкции:

тип_класса имя_класса {список_членов_класса};

где

тип_класса – одно из служебных слов **class, struct, union**;

имя_класса – идентификатор;

список_членов_класса – определения и описания типизированных данных и принадлежащих классу функций.

Функции – это методы класса, определяющие операции над объектом.

Данные – это поля объекта, образующие его структуру. Значения полей определяет состояние объекта.

Рассмотрим реализацию понятия даты с использованием struct для того, чтобы определить представление даты и множества функций для работы с переменными этого типа:

```
struct date {
    int month, day, year;    // дата: месяц, день, год
    void set(int, int, int);
    void get(int*, int*, int*};
    void next();
    // ...
};
```

Функции, описанные таким образом, называются функциями-членами и могут вызываться только для специальной переменной соответствующего типа с использованием стандартного синтаксиса для доступа к членам структуры.

Например:

```
date today;              // сегодня
void f()
{
        today.set(18,1,1985);
        today.next();
}
```

Поскольку разные структуры могут иметь функции члены с одинаковыми именами, при определении функции члена необходимо указывать имя структуры:

```
void date::next()
{
        if ( ++day > 28 ) {
        // делает сложную часть работы
        …
```

```
        }
}
```

В функции-члене имена членов структуры могут использоваться без явной ссылки на объект. В этом случае имя относится к члену того объекта, для которого функция была вызвана.

Описание date в предыдущем подразделе дает множество функций для работы с date, но не указывает, что эти функции должны быть единственными для доступа к объектам типа date. Это ограничение можно наложить, используя вместо struct class:

```
class date {    int month, day, year;
        public:
                void set(int, int, int);
                void next();
};
```

Методы доступа к полям (геттеры и сеттеры)

```
class TPen
{       private:
                string FColor;
        public:
                string getColor ();
                void setColor ( string newColor );
};
```

Получить значение:

```
string TPen::getColor () {   return FColor;        }
```

Записать значение:

```
void TPen::setColor ( string newColor )
{       if ( newColor.length()!= 6 )
                FColor = "000000";
        else
                FColor = newColor;
}
```

Для описания объекта класса (экземпляра класса) используется конструкция

имя_класса имя_объекта;

Для изменения видимости компонент в определении класса можно использовать спецификаторы доступа: **public, private, protected**.

Общедоступные (public) компоненты класса доступны в любой части программы. Они могут использоваться любой функцией как внутри данного класса, так и вне его. Доступ извне осуществляется через имя объекта:

имя_объекта.имя_члена_класса
ссылка_на_объект.имя_члена_класса
указатель_на_объект->имя_члена_класса

Собственные (private) компоненты класса локализованы в классе и не доступны извне. Они могут использоваться функциями – членами данного класса и функциями – “друзьями” того класса, в котором они описаны.

Защищенные (protected) компоненты доступны внутри класса и в производных классах.

В том, что доступ к структуре данных ограничен явно описанным списком функций, есть несколько преимуществ. Любая ошибка, которая приводит к тому, что дата принимает недопустимое значение (например, Декабрь 36, 1985), должна быть вызвана кодом функции-члена, поэтому первая стадия отладки, локализация, выполняется еще до того, как программа будет запущена.

Защита закрытых данных связана с ограничением использования имен членов класса. В функции-члене на члены объекта, для которого она была вызвана, можно ссылаться непосредственно. Например:

```
class x {
    int m;
    public:
        int readm() { return m; }
};
x aa;
x bb;
void f(){
        int a = aa.readm();
        int b = bb.readm();
        // ...
}
```

В первом вызове члена readm() m относится к aa.m, а во втором - к bb.m.

Указатель на объект, для которого вызвана функция-член, является скрытым параметром функции. В каждой функции класса x указатель this неявно описан как

```
x* this;
```

и инициализирован так, что он указывает на объект, для которого была вызвана функция член. this не может быть описан явно, так как это ключевое слово.

3. Общее задание (2 балла)

1. Создайте консольный проект. В теле функции main будет выполняться демонстрация работы объекта.
2. Добавьте в проект новый класс и назовите этот класс Worker. В класс добавьте два общедоступных поля: имя и возраст и одно скрытое поле: вес:

```
class Worker {
        public:
                int age;
                char* name;
        private:
                float weight;
};
```

3. В теле функции main создайте объект класса Worker:

```
Worker *wrk1 = new Worker ();
wrk1->age = 34;
wrk1->name =  “Иванов”;
```

   Добавьте оператор(функцию) вывода на экран созданного объекта.
4. Запустите программу на выполнение.
5. Попробуйте записать значение в поле weight. Почему данные не записались?
6. Для записи и чтения данных из скрытых полей используют методы. Добавим во внутрь класса Worker новый метод (действие) который будет отвечать за еду, если человек чего-то там съест, то его вес должен будет увеличиться на количество съеденного.

   в структуру класса:

```
public:
…
void eat (float how_much);
```

после описания класса, но до функции main:

```
void Worker::eat (float how_much){
    weight = weight + how_much;
}
```

7. Если поле вес скрытое, то мы в него не только писать не можем, но и читать тоже не можем. Для чтения данных из скрытого поля необходимо использовать еще один метод:

   в структуру класса:

   ```
   public:
   …
   float get_weight();
   ```

   после описания класса, но до функции main:

   ```
   float Worker::get_weight(){
       return weight;
   }
   ```

8. Почему в последних двух функциях после слова public идут различные слова? Что они обозначают и на что влияют?
9. Теперь эти два метода надо использовать в нашей программе. Заставьте рабочего съесть 2, а затем 3 кг пищи. Проверьте его вес.

   ```
   wrk1->eat(2);
   wrk1->eat(3);
   float ves;
   ves = wrk1->get_weight();
   ```

   Отобразите результат на экран.
10. Запустите программу на выполнение. Проверьте работоспособность. Добавьте комментарии.
11. Усовершенствуйте метод eat таким образом, что если рабочий за раз съедает более чем 10 кг, то его возраст увеличивается на год, а вес увеличивается только на половину от съеденного.
12. Попросите рабочего съесть 15 кг и посмотрите на результат работы программы.
13. Измените программу так, что бы имя рабочего и его первоначальный возраст вводились с клавиатуры и вносились в соответствующие переменные.
14. Запустите программу. Проверьте ее работоспособность.
15. Добавьте рабочему еще одно скрытое поле, которое будет отвечать за настроение и будет иметь первоначальное значение равное 10.
16. Добавьте три метода: гулять (метод должен увеличивать настроение на 1), танцевать (метод должен увеличивать настроение на 2) и работать (метод должен уменьшать настроение на 2).
17. Дополните основную программу так, что бы рабочий после еды два раза погулял и три раза потанцевал.
18. Добавьте в класс функцию, которая будет возвращать текущее настроение пользователя.

19. Добавьте в основную программу метод работать 9 раз (можно в цикле) и выведите настроение пользователя на экран.
20. Настроение получилось отрицательным? – ужасно. Измените метод работать таким образом, что бы настроение никогда не было меньше нуля (т.е. если настроение было 1 и человек поработал, то оно должно стать не меньше 0).
21. Проверьте заново работоспособность программы.

4. Индивидуальное задание (3 балла)

- Реализовать пользовательский класс в соответствии с вариантом задания.
- При реализации классов поля должны быть скрытыми.
- Определить метод установки свойств (при недопустимых аргументах функции возвращать «false» и выдавать текст ошибки на экран).
- Определить метод чтения свойств.
- Написать демонстрационную программу, в которой показать использование объектов созданного класса.

Вариант 1
Класс Треугольник
Свойства: три стороны
Операции:
 увеличение/уменьшение размера сторон в заданное количество раз;
 вычисление периметра;
 вычисление площади;
 определение значений углов.

Вариант 2.
Класс Треугольник
Свойства: три стороны
Операции:
 увеличение/уменьшение размера сторон на заданное количество процентов;
 вычисление средней линии для любой из сторон;
 определение вида треугольника по величине углов (Остроугольный, Тупоугольный, Прямоугольный);
 определение значений углов.

Вариант 3.
Класс Треугольник
Свойства: две стороны и угол между ними
Операции:
 увеличение/уменьшение размера угла на заданное количество процентов;
 определение вида треугольника по числу равных сторон (Разносторонний, Равнобедренный, Равносторонний);
 определение расстояния между центрами вписанной и описанной окружностей.
 определение значений углов.

Вариант 4.
Класс Треугольник
Свойства: две стороны и угол между ними
Операции:
 уменьшение/увеличение размера угла (из свойств) в заданное количество раз;

 вычисление длины биссектрисы принадлежащей любому углу;
 вычисление длин отрезков, на которые биссектриса делит любую сторону;
 определение значений углов.

Вариант 5.
Класс Треугольник
Свойства: сторона и два прилежащих к ней угла
Операции:
 уменьшение/увеличение размера стороны (из свойств) в заданное количество раз;
 вычисление длины медианы, принадлежащей любой стороне;
 определение подобен ли другой треугольник данному (указанному по индексу массива);
 определение значений сторон.

Вариант 6.
Класс Треугольник
Свойства: сторона и два прилежащих к ней угла
Операции:
 увеличение/уменьшение значения любого угла (из свойств) на заданное количество процентов;
 вычисление длины высот, принадлежащей любой стороне;
 определение значений сторон.

Вариант 7.
Класс Прямоугольный треугольник
Свойства: две стороны
Операции:
 увеличение/уменьшение размера любой стороны (из свойств) на заданное количество процентов;
 вычисление радиуса описанной окружности;
 вычисление полупериметра;
 определение значений углов.

Вариант 8.
Класс Прямоугольный треугольник
Свойства: сторона и угол
Операции:
 уменьшение/увеличение размера любой стороны (из свойств) на заданный процент;
 вычисление радиуса вписанной окружности;
 определение расстояния между центрами вписанной и описанной окружностей;
 вычисление квадратного корня из площади;
 определение значений сторон.

Вариант 9.
Класс Равнобедренный треугольник
Свойства: основание и боковая сторона
Операции: увеличение/уменьшение размера на определенный процент;
 вычисление длины медианы, принадлежащей любой стороне;
 вычисление периметра и площади;
 определение значений углов.

Вариант 10.

Класс Равнобедренный треугольник
Свойства: боковая сторона и угол при основании
Операции:
 увеличение/уменьшение размера в заданное количество раз;
 вычисление длины биссектрисы принадлежащей любому углу;
 вычисление длины высот, принадлежащей любой стороне;
 определение значений сторон.

Вариант 11.
Класс Параллелограмм
Свойства: две стороны и угол между ними
Операции:
 увеличение/уменьшение размера любой из сторон (из свойств) на определенный процент;
 вычисление периметра и площади;
 вычисление диагоналей;
 вычисление высоты.

Вариант 12.
Класс Параллелограмм
Свойства: две стороны и диагональ (прилегающая к ним так, что бы образовать треугольник)
Операции:
 увеличение/уменьшение размера в заданное количество раз;
 вычисление квадратного корня из периметра и площади;
 вычисление диагонали и стороны;
 вычисление высоты.

Вариант 13.
Класс Прямоугольник
Свойства: две стороны
Операции:
 увеличение/уменьшение размера любой из сторон на определенный процент;
 вычисление периметра и площади;
 вычисление диагонали.

Вариант 14.
Класс Квадрат
Свойства: Сторона
Операции:
 увеличение/уменьшение размера на определенный процент;
 вычисление периметра и площади;
 вычисление диагонали.

Вариант 15.
Класс Ромб
Свойства: сторона и диагональ (меньшая)
Операции:
 увеличение/уменьшение размера на определенный процент;
 вычисление периметра и площади;
 вычисление высоты.

Вариант 16.
Класс Трапеция
Свойства: четыре стороны
Операции:
 увеличение/уменьшение размера в заданное количество раз;
 вычисление периметра и площади;
 определение подобна ли другая трапеция данной (указанной по индексу массива);
 определение размера средней линии;
 вычисление высоты.

Вариант 17.
Класс Окружность
Свойства: радиус
Операции:
 увеличение/уменьшение размера на определенный процент;
 вычисление длины окружности и площади круга;
 определение диаметра.

Вариант 18.
Класс Сегмент окружности
Свойства: хорда и высота сегмента
Операции:
 увеличение/уменьшение размера в заданное количество раз;
 вычисление площади;
 определение длины дуги;
 вычисление длины окружности и ее диаметра.

Вариант 19.
Класс Сектор окружности
Свойства: радиус и центральный угол
Операции:
 увеличение/уменьшение размера в заданное количество раз;
 вычисление площади;
 определение длины дуги;
 вычисление длины окружности и ее диаметра.

Вариант 20.
Класс Круговое кольцо
Свойства: внешний и внутренний диаметр
Операции:
 увеличение/уменьшение размера в заданное количество раз;
 вычисление площади;
 вычисление среднего радиуса;
 вычисление толщины кольца.

5. Контрольные вопросы

1. В определении класса члены класса с ключевым словом private доступны:
   а) любой функции программы;
   б) в случае, если известен пароль;
   в) методам этого класса;
   г) только открытым членам класса.

2. Напишите определение класса studentgroup, включающего одно закрытое поле типа int с именем number и одним открытым методом с прототипом voidadd( ).
3. Истинно ли следующее утверждение: поля класса должны быть закрытыми?
4. Для чего при работе с объектами применяется операция «точка»?
5. Для чего при работе с объектами применяется операция «стрелка»?
6. Методу класса всегда доступны данные:
    а) объекта, членом которого он является;
    б) класса, членом которого он является
    в) любого объекта класса, членом которого он является;
7. Что является единственным формальным различием между структурами и классами в C++?
8. Пусть определены три объекта класса. Сколько копий полей класса содержится в памяти? Сколько копий методов класса?
9. Для чего необходимо переопределять операции new и delete?
10. Что такое указатель this?