

Методика выполнения работы

Выполнение задания предполагает решение следующих задач:

- # создание описания класса;
- # создание конструкторов и деструктора класса;
- # создание полей и методов класса и определение областей доступа (видимости) членов класса;
- # реализацию перегрузки методов по заданию;
- # создание перегрузки операций (переопределение операции);
- # реализацию статических полей и методов класса;
- # разработку программы - клиента, где объявить объекты класса и продемонстрировать корректную работу всех полей, методов и операторов класса.

Разберем условие задачи и её решение для разработки простейшего класса **Окружность** (*Circle*).

Условие задачи требуют разработать класс **Окружность** (*Circle*) со следующими спецификациями:

- ✓ скрытые поля окружности - координаты центра X , Y и *Радиус*;

Решение А. описать поля в закрытой области (private) класса – см.

Пример 1;

- ✓ методы, позволяющие получить как доступ для чтения к полям X , Y и *Радиус*, так и доступ для записи с предварительной проверкой на корректность;

Решение В. создать методы с префиксами get<имя Поля> и set<имя Поля> соответственно для вывода и ввода значения полей; разработать конструкторы по умолчанию, с параметрами и конструктор копирования, деструктор класса:

```
class Circle {
    double x, y;
    double r;
public:
    //конструктор по умолчанию
    Circle() { x = 0; y = 0; r = 1; }
    //конструктор с параметрами
    Circle(double Vx, double Vy, double Vr)
    {
        x = Vx; y = Vy; setRadius(Vr);
    }
    //конструктор копирования
    Circle(Circle& C) { x = C.x; y = C.y; r = C.r; }
    // метод - устанавливает значение полей класса
    void setCircle(double Vx, double Vy, double Vr = 1);
    void setRadius(double Vr = 1) { r = abs(Vr); }
    // метод - выводит на экран параметры окружности
    double getRadius() { return r; }
    double getX() { return x; }
    double getY() { return y; }
    // деструктор - пустое тело функции, так как в составе класса нет динамических полей
    ~Circle() {};
};
```

- ✓ методы для вычисления *Площади окружности* по формуле $S=\pi r^2$;

Решение С. добавить в класс метод вычисления формулы площади окружности.

```
class Circle {
    double x, y;
    double r;
public:
    //конструктор по умолчанию
    Circle() { x = 0; y = 0; r = 1; }
    //...
    // метод - вычисляет площадь окружности,
    // PI- объявленная выше константа
    double getArea() { return PI * r * r; }
    //...
    ~Circle() {}
};
```

- ✓ метод вывода, который возвращает строку с информацией об окружности;

Решение D. разрабатываем метод класса, где все поля собираются в одну строку типа string, не забыв подключить необходимую библиотеку:

```
string info() { return "Центр(x,y):" + to_string(x) + "," + to_string(y) + "Радиус:" + to_string(r); }
```

- ✓ метод, который позволит изменить либо только координату X , либо координаты X, Y – в зависимости от количества параметров;

Решение E. добавляем метод перемещения окружности по X ; перегрузка данного метода, когда на входе два параметра X, Y :

```
// метод - перемещает центр оси X, выполняя перемещение окружности
void moveCircle(double dx) { x = x + dx; }
```

```
// метод - перемещает центр, выполняя перемещение окружности
void moveCircle(double dx, double dy) { x = x + dx; y = y + dy; }
```

- ✓ операторы сравнения на равенство и неравенство окружностей по радиусу;

Решение F. добавим в класс перегруженные операции, то есть переопределим операции $==$ и $<$ для объектов класса, где полем для сравнения будет радиус:

```
// перегрузка операций сравнения
bool operator > (Circle& C) { return r > C.r; }
bool operator == (Circle& C) { return r == C.r; }
```

- ✓ статическое поле, в котором будет храниться общее количество объектов окружностей;

Решение G. определить в составе класса статическое поле и статические методы его обработки, поле должно быть также объявлено как глобальная переменная программы:

```

class Circle {
    double x, y;
    double r;
    static int count;
    static void countCircle() { count++; }
    static int getCount() { return count; }
public:
    //...
};

//статическое поле класса, объявленное как глобальная переменная программы
int Circle::count = 0;

```

После описания класса и его решений по составу в программе клиенте класса объявим объекты класса и проверим все методы класса. Хороший стиль программирования предполагает, что процесс отладки идет параллельно добавлению кода. Добавили метод в класс, в программе - клиенте добавляете вызов метода через объект класса и выполняете программу. Затем добавляете следующий метод и проверяете и т.д.

Полное программное решение задачи представлено на Листинге 1.

```
#include <iostream>
#include <string>
constexpr auto PI = 3.1415;
using namespace std;

class Circle { // начало области действия класса
public:
    // статические поля и методы класса
    static int count;
    static void countCircle() { count++; }
    static int getCount() { return count; }
private:
    // закрытые члены класса
    double x, y; // абсцисса и ордината центра
    double r;    // радиус
public:
    // открытые члены класса
    //конструктор по умолчанию
    Circle() { x = 0; y = 0; r = 1; countCircle(); }
    //конструктор с параметрами
    Circle(double Vx, double Vy, double Vr){
        setCircle(Vx, Vy, Vr); countCircle();
    }
    //конструктор копирования
    Circle(Circle& C) { x = C.x; y = C.y; r = C.r; countCircle(); }
    // метод -устанавливает значение полей класса
    void setCircle(double, double, double);
    void setRadius(double Vr = 1) { r = abs(Vr); }
    // метод - возвращает параметры окружности
    double getRadius() { return r; }
    double getX() { return x; }
    double getY() { return y; }
    // метод - вычисляет площадь окружности
    double getArea() { return PI * r * r; }

    // метод - возвращает строку информации об окружности
    string info();

    // метод - перемещает центр оси X, выполняя перемещение окружности
    void moveCircle(double dx) { x = x + dx; }

    // метод - перемещает центр, выполняя перемещение окружности
    void moveCircle(double dx, double dy) { x = x + dx; y = y + dy; }

    //метод - масштабирует, преобразование подобия с коэффициентом k
    void zoomCircle(double k) { r = r * k; }

    // перегрузка операций сравнения
    bool operator > (Circle& C) { return r > C.r; }
    bool operator == (Circle& C) { return r == C.r; }
    // деструктор - пустое тело функции, так как в составе класса нет динамических полей
    ~Circle() {};
}; // конец области действия класса
```

```

//описание методов вне класса
void Circle::setCircle(double Vx, double Vy, double Vr = 1) {
    x = Vx; y = Vy; setRadius(Vr);
}

string Circle::info() {
    return "Центр(x,y):" + to_string(x) + "," + to_string(y) + "Радиус:" + to_string(r);
}

//статическое поле класса, объявленное как глобальная переменная программы
int Circle::count = 0;

int main()
{
    // создаем объекты
    Circle H1(-2, 3, 7); //конструктор с параметрами
    Circle H2;           //конструктор по умолчанию

    // выводим информацию об окружностях
    cout << H1.info() << endl; cout << H2.info() << endl;

    // создаем указатель на объект
    Circle* pH = new Circle(H1); //конструктор копирования
    cout << pH->info() << endl;

    //проверяем методы
    cout << "Площадь окружности " << pH->getArea() << endl;
    cout << H1.getRadius() << " ( " << H1.getX() << " , " << H1.getY() << " )" << endl;

    H2.setCircle(-4.5, 0, -5); cout << H2.info() << endl;

    H2.moveCircle(3); cout << H2.info() << endl;
    H1.moveCircle(-2, 2); cout << H1.info() << endl;
    cout << "сравним окружности H1 и H2" << endl;
    if (H1.operator>(H2))
        cout << H1.getRadius() << '>' << H2.getRadius() << endl;
    else cout << H1.getRadius() << '<' << H2.getRadius() << endl;

    if (H1.operator==(H2))
        cout << H1.getRadius() << '=' << H2.getRadius() << endl;
    else cout << H1.getRadius() << '!=' << H2.getRadius() << endl;
    // вызываем статический метод для проверки работы
    cout << "Количество объектов типа Circle " << Circle::getCount() << endl;
}

```

Задание

Создайте класс (в соответствии со своим вариантом), а также консольное приложение, демонстрирующее его корректную работу.

Общая постановка задачи:

- a. объявить скрытые поля;
- b. разработать методы, позволяющие получить как доступ для чтения к полям класса, так и доступ для записи с предварительной проверкой на корректность;
- c. разработать метод для вычисления характеристики по заданной формуле;
- d. разработать метод вывода, который возвращает строку с информацией об объекте класса, то есть значения из всех полей;
- e. разработать метод, который изменяет поля класса по заданному алгоритму и его перегруженный вариант с другим количеством параметров;
- f. разработать перегруженные операции больше и меньше, которые могут сравнивать по заданному полю объекты класса;
- g. объявить статические поля, в которых будет храниться информация общая для объектов классов;

В задании указаны обязательные для реализации поля и методы класса. Необходимо также дополнительно разработать собственные поля и методы, дополняющие функциональность вашего класса. Все поля и методы должны иметь понятные, осмысленные имена.

Консольное приложение должно продемонстрировать корректную работу всех полей, методов и операторов класса. При этом должна быть предусмотрена обратная связь с пользователем, т.е. например: у пользователя запрашиваются характеристики объектов, которые необходимо создать, а затем выводится меню доступных операций и запрашивается код операции, которую необходимо выполнить. После ввода кода, а при необходимости – и дополнительных параметров, на экран выводится результат и вновь меню доступных операций. Цикл продолжается, пока пользователь не введет код выхода из программы (например, 0). Для реализации можно использовать в основной программе операторы цикла и выбора.

Класс должен быть реализован в отдельном модуле.

Варианты заданий

Вариант 1. Класс *Прямоугольник* (*CMyRect*):

- a. скрытые поля *Длина*, *Ширина*, *Цвет*;
- b. методы, позволяющие получить как доступ для чтения к полям *Длина*, *Ширина* и *Цвет*, так и доступ для записи с предварительной проверкой на корректность;
- c. методы для вычисления *Площади* и *Периметра* прямоугольника;

- d. метод вывода, который возвращает строку с информацией об экземпляре класса;
- e. метод, который позволит изменить либо только длину и ширину прямоугольника, либо длину, ширину и цвет – в зависимости от количества параметров;
- f. операторы больше и меньше, которые могут сравнивать по площади как два прямоугольника, так и площадь прямоугольника с заданной числовой константой;
- g. статические поля, в котором будут храниться размеры наибольшего прямоугольника среди имеющихся экземпляров классов.

Вариант 2. Класс *Автомобиль* (CAuto):

- a. скрытые поля *Максимальная скорость*, *Расход топлива на 100 км.*, *Ёмкость бензобака* и *Высота*;
- b. методы, позволяющие получить как доступ для чтения к полям *Максимальная скорость*, *Расход топлива на 100 км.*, *Ёмкость бензобака* и *Высота*, так и доступ для записи с предварительной проверкой на корректность;
- c. методы для вычисления *Расстояния, на которое хватает полного бака* и *Минимального времени преодоления заданного расстояния*;
- d. метод вывода, который возвращает строку с информацией об экземпляре класса;
- e. метод, который проверяет, либо сможет ли автомобиль проехать заданное расстояние без дополнительных заправок, либо сможет ли он его проехать без дополнительных заправок и за указанное минимальное время – в зависимости от количества параметров;
- f. операторы больше и меньше, которые могут сравнивать по максимальной скорости как два автомобиля, так и максимальную скорость автомобиля с заданной числовой константой;
- g. статическое поле, в котором будут храниться количество имеющихся на данный момент экземпляров данного класса.

Вариант 3. Класс *Город* (CCity):

- a. скрытые поля *Население*, *Площадь*, *Название* и *Наличие аэропорта* (есть или нет);
- b. методы, позволяющие получить как доступ для чтения к полям *Население*, *Площадь*, *Название* и *Наличие аэропорта*, так и доступ для записи с предварительной проверкой на корректность;
- c. метод для вычисления *Плотности населения на 1 кв.м.* и *Площади на 1*
- d. метод вывода, который возвращает строку с информацией об экземпляре класса;
- e. метод, который позволит изменить либо только население, либо население и площадь, либо население, площадь и статус города – в зависимости от количества параметров;
- f. операторы больше и меньше, которые могут сравнивать по населению как два города, так и население города с заданной числовой константой;
- g. статическое поле, в котором будут храниться количество городов с аэропортом среди имеющихся экземпляров классов.

Вариант 4. Класс *Книга* (CBook):

- a. скрытые поля *Наименование, Автор, Количество страниц, Год издания, Количество экземпляров в библиотеке и Количество экземпляров на руках у читателей*;
- b. методы, позволяющие получить как доступ для чтения к полям *Наименование, Автор, Количество страниц, Год издания, Количество экземпляров в библиотеке и Количество экземпляров на руках у читателей*, так и доступ для записи с предварительной проверкой на корректность;
- c. методы для вычисления *Количества доступных экземпляров на текущий момент времени*;
- d. метод вывода, который возвращает строку с описанием книги (автор, название, количество страниц, год издания);
- e. методы, которые позволяют оформить получение или возврат либо одной книги, либо заданного количества книг – в зависимости от наличия или отсутствия параметров;
- f. операторы больше и меньше, которые могут сравнивать по количеству страниц две книги;
- g. статическое поле, в котором будет храниться общее количество экземпляров различных книг в библиотеке.

Вариант 5. Класс *Человек* (CMan):

- a. скрытые поля *Имя, Пол, Возраст, Рост и Вес*;
- b. методы, позволяющие получить доступ для чтения к полям *Имя, Пол, Возраст, Рост и Вес* и доступ для записи с проверкой на корректность;
- c. метод для вычисления *Коэффициента = Вес/(Рост-100)*;
- d. метод вывода, который возвращает строку с информацией о человеке;
- e. метод, который позволит увеличить возраст либо на один год, либо на заданное количество лет – в зависимости от наличия или отсутствия параметров;
- f. операторы больше и меньше, которые могут сравнивать по возрасту двух людей;
- g. статические поля, в которых будут храниться количество мужчин и количество женщин среди имеющихся объектов классов.

Вариант 6. Класс *Компьютер* (CComputer):

- a. скрытые поля *Частота процессора, Количество ядер, Объем оперативной памяти, Объем жесткого диска и Объем занятого пространства на жестком диске*;
- b. методы, позволяющие получить как доступ для чтения к полям *Частота процессора, Количество ядер, Объем оперативной памяти, Объем жесткого диска и Объем занятого пространства на жестком диске*, так и доступ для записи с предварительной проверкой на корректность;
- c. методы для вычисления *Объема свободного пространства на жестком диске* и проверки, *Можно ли скопировать на жесткий диск файл указанного размера*;
- d. метод вывода, который возвращает строку с информацией о компьютере;
- e. метод, который позволит изменить либо только объем занятого места на жестком диске, либо объем занятого места на жестком диске и размер самого жесткого диска – в зависимости от количества параметров;
- f. операторы больше и меньше, которые могут сравнивать по количеству ядер два компьютера;

- g. статическое поле, в котором будет храниться общий объем всех жестких дисков среди имеющихся экземпляров класса *Компьютер*.

Вариант 7. Класс *Банковский Счет*(CAccount):

- a. скрытые поля *Имя владельца, Номер счета, Сумма, Годовой процент*;
- b. методы, позволяющие получить как доступ для чтения к полям *Имя владельца, Номер счета, Сумма*, так и доступ для записи с предварительной проверкой на корректность;
- c. метод для вычисления ежемесячной *Суммы* счета согласно *Годовому проценту*: $Сумма = Сумма + Сумма * Годовой\ процент / 12$;
- d. метод вывода, который возвращает строку с элементами счета;
- e. метод, который позволит либо изменить только сумму на счету (+/-), либо еще и годовой процент – в зависимости от количества параметров;
- f. операторы сравнения на равенство и неравенство (>) два счета по полю *сумма*;
- g. статическое поле, в котором будет храниться общая сумма на всех счетах на текущий момент.

Вариант 8. Класс *Цилиндр* (CCylinder):

- a. скрытые поля *Радиус, Высота*;
- b. методы, позволяющие получить как доступ для чтения к полям класса *Радиус, Высота*, так и доступ для записи с предварительной проверкой на корректность;
- c. методы для вычисления площади основания и объема цилиндра $Объем = Площадь\ основания * Высота$;
- d. метод вывода, который возвращает строку с параметрами цилиндра;
- e. метод, который позволит увеличить/уменьшить радиус цилиндра на заданную величину, либо изменить еще и высоту – в зависимости от числа параметров;
- f. операторы сравнения на равенство и неравенство (>) двух цилиндров, по полю *радиус*;
- g. статическое поле, в котором будет храниться количество имеющихся на данный момент экземпляров данного класса.

Вариант 9. Класс *Кассовый чек*(CCheck):

- a. скрытые поля *Номер, Сумма чека со скидкой, Форма оплаты (наличные/платежная карта), ФИО кассира*;
- b. методы, позволяющие получить как доступ для чтения к полям класса *Номер, Сумма чека со скидкой, Форма оплаты (наличные/платежная карта), ФИО кассира*, так и доступ для записи с предварительной проверкой на корректность;
- c. метод расчета суммы скидки по чеку, процент скидки – параметр метода;
- d. метод вывода, который возвращает строку с информацией о чеке;
- e. метод, который изменит форму оплаты чека, либо изменить еще и ФИО кассира – в зависимости от числа параметров;
- f. операторы сравнения на равенство и неравенство (>) двух чеков по сумме;
- g. два статических поля, в которых будут храниться соответственно суммы по чекам оплаченных наличными и платежными картами.

Вариант 10. Класс *Телефонный звонок* (CCall):

- a. скрытые поля *Абонент, Номер, Важность Звонка (bool), Тема, Время начала звонка, Время окончания звонка*;

- b. методы, позволяющие получить как доступ для чтения к полям класса, так и доступ для записи с предварительной проверкой на корректность;
- c. метод для вычисления длительности звонка;
- d. метод вывода, который возвращает строку с информацией о звонке;
- e. метод, который позволит изменить либо только тему, либо еще и важность телефонного звонка – в зависимости от количества параметров;
- f. операторы сравнения на равенство и неравенство ($>$) звонков по длительности;
- g. статическое поле, в котором будет храниться количество важных звонков.

