Константные объекты и константные функции. Статические члены класса

Тема 3

Константные объекты

Объект класса может быть создан в программе с использованием спецификатора **const**. Такой объект будет константным, то есть во время выполнения программы объект изменять нельзя. Для доступа к полям такого объекту можно использовать только функции, объявленные с суффиксом **const** – константные функции

Ограничения, накладываемые на константные функции:

- не может быть статической;
- не может менять значения полей того объекта, для которого вызвана;
- может вызывать только другие константные функции класса.

Заголовочный файл Rational.h

```
class Rational
{
public:
    …
    int GetNum() const {return numerator;}
    int GetDenom()const {return denominator;}
    void SetNum(int num){numerator = num; Shorten();}
    void Print() const ;
    void GetRational(int & num, int & denom) const;
    …
private:
    …
};
```

Все функции класса, не изменяющие поля текущего объекта класса, следует делать константными

Файл исходного кода Rational.cpp

```
void Rational::Print() const
{
    std::cout << numerator << "/"
              << denominator << std::endl;
}


void Rational::GetRational(int & num,
                           int & denom) const
{
    num = numerator;
    denom = denominator;
}
```

Файл исходного кода
test.cpp

```
void fun ( const Rational & r )
{
    r.SetNum(5);  //ошибка
    r.Print();
    Rational r1(r);
    r1.SetNum(7); //хорошо
    r1.Print();   //хорошо
}
```

Статические поля класса

- Создаются в одном экземпляре
- Доступны всем объектам класса
- Используются для обмена данными между объектами одного класса, для счетчика созданных экземпляров класса и т.д.
- Существуют, даже если не создано ни одного объекта класса
- Для работы с ними нужны статические функции

Ограничения, накладываемые на статические функции:

- не может быть константной
- не может быть виртуальной
- нет указателя this
- нельзя обратится к нестатическим полям

Разработать класс «Счетчик»

```
class Counter
{
public:
   Counter(){count++;}
   Counter(const Counter& r){count++;}
   ~Counter(){count--;}
   static int GetCount(){return count;}
private:
   static int count;
};

int Counter::count  = 0; //инициализация вне класса
```

Тестовая программа для класса «Счетчик»

```
int main() {
    std::cout << Counter::GetCount() << std::endl; //0
    Counter c1;
    std::cout << c1.GetCount() << std::endl;       //1
    Counter c2(c1);
    std::cout << c2.GetCount() << std::endl;       //2
    std::cout << c1.GetCount() << std::endl;       //2
    Counter * p;
    std::cout << Counter::GetCount() << std::endl; //2
    p = new Counter();
    std::cout << p->GetCount() << std::endl;       //3
    std::cout << Counter::GetCount() << std::endl; //3
    std::cout << c1.GetCount() << std::endl;       //3
    delete p;
    std::cout << Counter::GetCount() << std::endl; //2
    std::cout << c1.GetCount() << std::endl;       //2
    return 0;
}
```

Конец