

## Практическое занятие №5

«Статические, константные и дружественные члены класса»

(Продолжительность работы 2 часа)

### Цели:

изучить работу со статическими и константными членами класса.

### 1. Краткие теоретические сведения

#### **Статические члены класса**

Каждый объект одного и того же класса имеет собственную копию данных класса. Но существуют задачи, когда данные должны быть компонентами класса, и иметь их нужно только в единственном числе. Такие компоненты должны быть определены в классе как *статические (static)*. Статические данные классов не дублируются при создании объектов, т.е. каждый статический компонент существует в единственном экземпляре.

```
class Point{
int x, y;
static int counter;
public:
    Point (void) { x =0 ; y = 0; counter++;}
    Point(int _x, int _y){ x = _x; y = _y; counter++;}
    ~Point(){ counter--;}
    static int getNumber(){return counter;}
};

int Point::counter = 0;
void main() {
    Point pp = new Point[20];
    Point p1(3,33);
    cout << Point::getNumber()<<'\n';
    delete [] pp;
    cout << p1::getNumber()<<'\n';
}
```

Доступ к статическому компоненту возможен только после его инициализации. Для инициализации используется конструкция

***тип имя\_класса : : имя\_данного\_инициализатор;***

Например, `int Point::counter = 0;`

Это предложение должно быть размещено в глобальной области после определения класса. Только при инициализации статическое данное класса получает память и становится доступным. Обращаться к статическому данному класса можно обычным образом через имя объекта

***имя\_объекта.имя\_компонента***

или через имя класса

***имя\_класса : : имя\_компонента***

Однако так можно обращаться только к *public* компонентам.

К *private* статической компоненте извне можно обратиться с помощью компонента-функции этого класса. Эту проблему решают *статические компоненты-функции*. Эти функции можно вызвать через имя класса.

**имя\_класса :: имя\_статической\_функции**

### **Константные методы**

Отличаются тем, что не изменяют значений полей своего класса. Для того, чтобы сделать функцию константной, необходимо указать ключевое слово *const* после прототипа функции, но до начала тела функции (как в примере) – как при объявлении функции, так и при определении, если они разделены. Те методы, которые лишь считывают данные из поля класса, имеет смысл делать константными, поскольку у них нет необходимости изменять значения полей объектов класса. Использование константных функций помогает компилятору обнаруживать ошибки и указывает читающему листинг, что функция не изменяет значений полей объектов. Пример объявления константного метода приведен ниже.

```
// constfu.cpp
// применение константных методов
class aClass
{
private:
    int alpha;
public:
    void nonFunc()           // неконстантный метод
    { alpha = 99; }         // корректно
    void conFunc()const      // константный метод
    { alpha = 99; }         // ошибка: нельзя изменить значение поля
};
```

Если объект создается как типизированная константа (то есть объявляется с ключевым словом *const*), то он становится недоступным для изменения и для такого объекта можно вызвать только константные методы, поскольку только они гарантируют, что объект не будет изменён. Если же необходимо разрешить константной функции изменять некоторые элементы-данные у константных объектов, то их необходимо объявить с ключевым словом *mutable*.

```
Class AnyClass {
private:
    int value;
    mutable char msg;
public:
    jnt GetValue( ) const;
};
```

```
Jnt AnyClass::GetValue( ) const{
    msg = 10; // Допускается, поскольку msg - mutable.
    //
    // value изменять нельзя:
    //
    // value = 111;
    //
    return value;
}
```

## **2. Практическое задание (100%)**

Для задания практической работы 6 «Перегрузка операций» выполнить доработку программы в соответствии с заданиями:

### **ЗАДАНИЕ 1. Статические члены-данные класса**

1. Добавить статическое поле `int count`, выступающее в роли счетчика объектов класса.
2. Деструктор класса должен уменьшать на единицу значение счетчика.
3. Добавить статический метод `int getCount()` возвращающий значение счетчика.
4. Продемонстрировать изменение значения статического поля.

### **ЗАДАНИЕ 2. Константные методы и объекты**

1. Определить какие методы являются константными, определить константные параметры и константные возвращаемые значения методов.
2. Добавить константное поле, хранящее идентификатор объекта (номер созданного объекта), предусмотреть методы вывода информации о идентификаторе.
3. Описать и инициализировать обычные и константные объекты.
4. Выполнить вызовы обычных и константных методов для каждого вида объектов.
5. Провести тестирование программы: Откомпилировать программу. Имеются ли ошибки компиляции и какие? Если имеются, то закомментировать соответствующие строки кода и вновь провести компиляцию. Какие предупреждения выдает компилятор и в чем их смысл? Как их можно объяснить с позиции обеспечения надежности программы?

## **3. Список рекомендуемой литературы**

1. Павловская Т. А. C/C++. Программирование на языке высокого уровня : для магистров и бакалавров : учеб. для вузов / Т. А. Павловская. - Гриф МО. - Санкт-Петербург: Питер, 2013. - 460 с. : ил.
2. Professional C++, 3rd Edition. Marc Gregoire. ISBN: 978-1-118-85805-9. Paperback 984 pages. September 2014

## **4. Контрольные вопросы**

1. Константный метод, вызываемый для объекта класса
  - а) может изменить как константные, так и неконстантные поля
  - б) может изменить только неконстантные поля
  - в) может изменять только константные поля
  - г) не может изменять никакие поля
2. Для чего нужно объявление поля класса со словом `mutable`
3. Какие свойства приобретает поле данных класса, объявленное как `static`
4. В чем состоит преимущество определения конструктора со списком инициализации элементов?
5. Какие свойства приобретает элемент-функция класса, если она объявлена как статическая
6. Что такое указатель `this`?