

**Статические члены класса.
Друзья класса**

- При объявлении класса данные-члены и функции-члены класса можно объявлять *статическими*.
- Существует лишь один экземпляр статических данных-членов класса, разделяемый всеми объектами этого класса в программе.
- Статический член класса является не частью объектов класса, а отдельным объектом.
- Статический член класса будет существовать, даже если не создано ни одного объекта класса.

cL::mem

```
class BitSet
{ private:
    typedef _int64 elem;
    static const int cBits = sizeof(elem) * 8;
    static const int size = 256 / cBits;
    elem set[size];
    ...
};
```

Обычное объявление функции-члена класса гарантирует три логически разные вещи:

- функция имеет право доступа к приватным членам класса;
- функция находится в области видимости класса;
- функция должна вызываться для объекта класса (имеется указатель *this*).

```
class X
{ private:
    int n;
    friend void friend_function(X* p, int i);
public:
    void member_function(int i);
};
```

```
void friend_function(X* p, int i)
{ p->n = i; }
```

```
void X::member_function(int i)
{ n = i; }
```

```
void f()
{ X obj;

    friend_function(&obj, 10);
    obj.member_function(10);
}
```

```
class Y
{ friend void X::f(Y *p);
  ...
};
```

```
class Y
{ friend class X;
  ...
};
```

```
class Complex
{ private:
    double r, m;
public:
    Complex(double r = 0, double m = 0) :
        r(r), m(m) { }
    //operator double () { return r; };
    friend Complex operator ++(Complex& c);
    friend Complex operator ++(Complex& c, int);
    friend Complex operator + (const Complex& c1,
                               const Complex& c2);
    Complex operator +=(const Complex& c);
};
```



```
Complex operator ++(Complex& c)
{ ++c.r; return c; }
```

```
Complex operator ++(Complex& c, int)
{ Complex x = c;
  c.r++;
  return x;
}
```

```
Complex operator +(const Complex& c1,
                  const Complex& c2)
{ return Complex(c1.r + c2.r, c1.m + c2.m); }
```

```
Complex Complex::operator +=(const Complex& c)
{ r += c.r; m += c.m;
  return *this;
}
```

```
void main()
{ Complex a(0, 0), b(2, 2), c;
  double d = 2;

  c = ++a;
  c = a++;
  c = a + ++b;
  c = a + d;
  c = 3 + b;
  c += 4;
  2 += c;
}
```



```
Vector operator + (const Vector& vector1,  
                  const Vector& vector2)  
{ Vector res(vector1.size);  
  
  if (vector1.size != vector2.size) return res;  
  for (int i = 0; i < vector1.size; i++)  
    *(res.v + i) = *(vector1.v + i) + *(vector2.v + i);  
  return res;  
}
```

```
void main()  
{ Vector v1(3), v2;  
  
  v2 = 3.5 + v1;  
}
```



```
class Circle
{ private:
    static int count;
    static long int maxX, maxY;
    int x, y, sx, sy;
    int radius;
    int colour;
public:
    Circle();
    ~Circle();
```

// Дружественные функции

```
friend void SetSize(int x, int y);
```

```
friend int  GetCount();
```

// Статические функции-члены класса

```
static void SetSize(int x, int y);
```

```
static int  GetCount();
```

```
void Paint(HDC hdc);
```

```
void Move();
```

```
int  IsInside(int px, int py);
```

```
int  GetColour();
```

```
void ChangeSpeed();
```

```
};
```

```
int Circle::count = 0;
long int Circle::maxX = 800;
long int Circle::maxY = 600;

void SetSize(int x, int y)
{ Circle::maxX = x; Circle::maxY = y; }

int GetCount()
{ return Circle::count; }

void Circle::SetSize(int x, int y)
{ Circle::maxX = x; Circle::maxY = y; }

int Circle::GetCount()
{ return Circle::count; }
```