Статические методы и атрибуты классов

```
//.h
class A{
        static const int constant;
        static void f ();
};

//.cpp
const int A::constant = 2;
void A::f (){
        //…
}
```

```
A a;
//обращение через
//объект
a.f ();
//обращение через
//класс
A::f ();
```

Статические методы классов

- Не могут быть виртуальными
- Не имеют дополнительного аргумента (this)
- Не могут быть константными (f () const)

Применение статических методов и атрибутов

Задача 1: необходимо реализовать класс, который вел подсчет своих экземпляров

Задача 2: необходимо реализовать класс, который бы имел не более одного экземпляра (ровно один экземпляр, не более N экземпляров)

Monostate

```
class Monostate{
private:
        static int a;
        static double b;
public:
        static void f1 ();
        static void f2 ();
};
```

Преимущества:

- Простота реализации
- Не требуется явное создание экземпляра

Недостатки:

- Невозможна построение иерархий
- Затруднительно построение сложных агрегаций таких классов

Singleton

```
class Singleton{
private:
        int a;
        double b;
        static Singleton * instance;
        Singleton ();
        Singleton (const Singleton&);
        ~Singleton ();
public:
        void f1 ();
        void f2 ();
        static Singleton * getInstance ();
        static void removeInstance ();
};
Singleton * Singleton::instance = 0;
```

Преимущества:

- Гибкость – возможность построения иерархий таких классов

Недостатки:

- Необходимость явного конструирования и уничтожения такого объекта

Потоки в C++

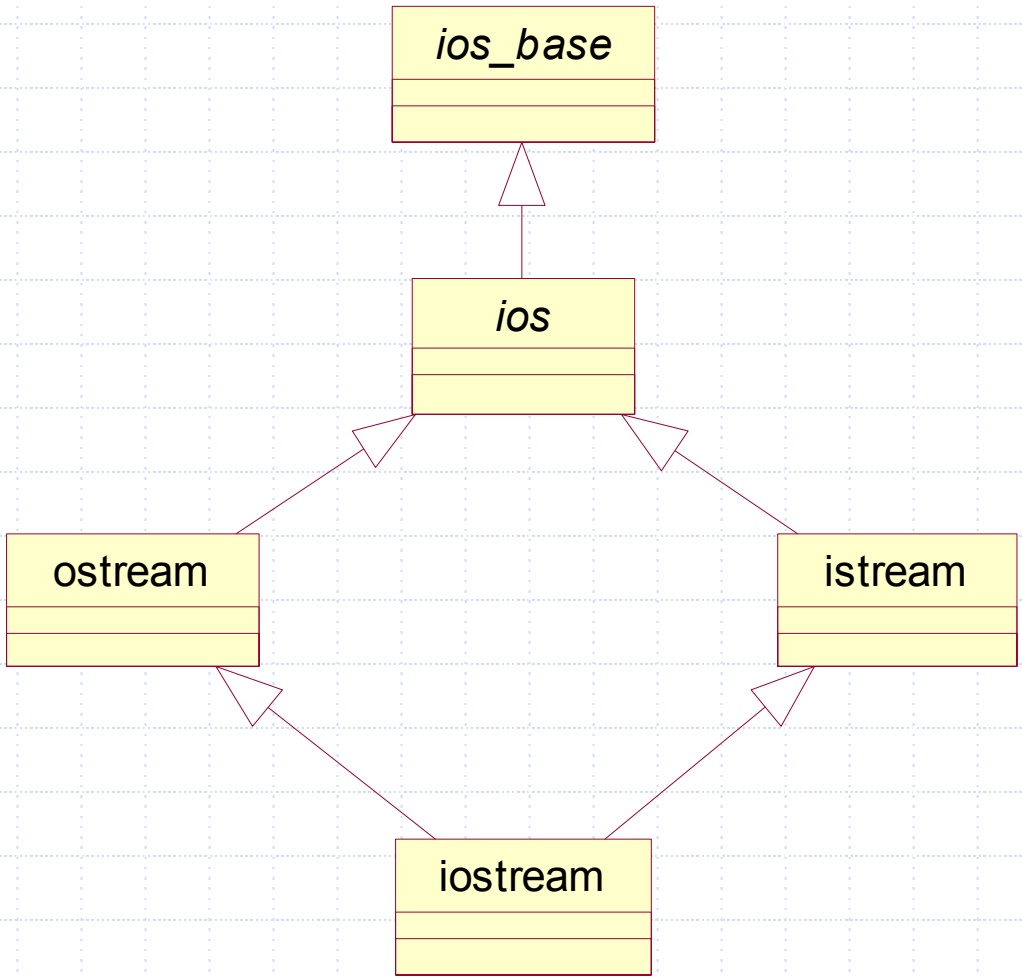
```
#include <stdio.h>

int main (){
        char * name = “Vasya”;
        printf (“Hello, %s!\n”, name);
        return 0
}
```

```
#include <iostream>
using namespace std;

int main (){
        char * name = “Vasya”;
        cout<<“Hello, ”<<name<<“!”<<endl;
        return 0;
}
```

Файловые потоки в C++

```
#include <fstream>
using namespace std;

int main (){
        ifstream file;
        file.open (“temp.txt”);
        while (1){
                int val;
                file >> val;
                if (file.eof ()) break;
                cout << val << endl;
        }
        return 0;
}
```

Иерархия классов в библиотеке потоков



Классы ios_base и ios

Абстрактные классы, базовые для всех классов потоков. В них определяются:

- Понятие состояния потока
- Буфер чтения/записи
- Базовые типы и константы

Методы:

```
bool eof() const;      //достигнут конец файла
bool fail() const;     //ошибка чтения/записи
bool bad() const;      //нарушение целостности потока
bool good() const;     //отсутствие ошибок
```

Класс ostream

Базовый класс для всех потоков вывода.

Содержит следующие методы:

```
ostream& operator << (…);
ostream& put (char);
ostream& write (char *s, streamsize n);
ostream& flush ();
ostream& seekp (pos_type pos);
ostream& seekp (pos_type pos, ios_base::seek_dir way);
//static const seekdir beg, cur, end;
pos_type tellp ();
```

Класс istream

Базовый класс для всех потоков ввода.

Содержит следующие методы:

```
ostream& operator >> (…&);
ostream& get (char&);
ostream& read (char *s, streamsize n);
ostream& seekg (pos_type pos);
ostream& seekg (pos_type pos, ios_base::seek_dir way);
//static const seekdir beg, cur, end;
pos_type tellg ();
```

Класс iostream

```
class iostream: public istream, public ostream{
//...
};
```

Ввод/вывод собственных типов

```
class Complex{
friend ostream& operator << (ostream&, const Complex&);
//…
};

ostream& operator << (ostream& s, const Complex& c){
        return s << c.real << "+I*" << c.image;
}
```

Виртуальные методы ввода/вывода

```
class Base{
public:
        virtual ostream& print (&ostream) const;
};

ostream& operator << (ostream& s, const Base& c){
        return c.print (s);
}
```