# Статические переменные класса

//Student.h

class Student

{

int m\_age;

**static** int m\_pupleNum;

public:

Student(int age)

{

m\_age = age;

m\_pupleNum++;

}

~Student() { m\_pupleNum-- };

};

//Student.cpp

**int Student::m\_pupleNum = 0;**

# Статические методы класса

class Student {

**static** int m\_pupleNum;

public:

Student() { m\_pupleNum++; }

~Student() { m\_pupleNum-- };

**static** int getPurpleNum() { return m\_pupleNum; }

};

//Student.cpp

**int Student::m\_pupleNum = 0;**

void main() {

int x = **Student :: getPurpleNum();**

Student st1;

x = **st1. getPurpleNum();**

}

# Массивы, указатели и объекты

class Student {

int m\_age;

public:

Student() { m\_age = 5; }

Student(int age) { m\_age = age; }

};

int main()

{

Student st\_array1[5];

Student st\_array2[2] = { Student(10), Student(15) };

Student st1 (10);

Student st\_arr[2] = {st1, Student(10)};

}

# Динамическая память

int main() {

Student\* stud = **new** Student ();

stud **->** setAge(50);//вызов public-метода

delete stud;

stud=0;

}

# Ключевое слово const

## Методы, возвращающие константное значение

class Student {

int m\_age;

public:

Student(int age = 20) { m\_age = age; }

**const** int& getAge()

{

m\_age++;

return m\_age;

}

};

int main()

{

Student st;

int x = st.getAge(); //Ok

st.getAge() = 10;// Ошибка

}

## Константные методы класса

class Student {

int m\_age;

public:

void setAge (int age) { m\_age = age; }

void readAge () **const** { cout << m\_age; }

}

## Константные объекты

# Конструктор копирования

Animal.h

class Animal

{

…

public:

**Animal (const Animal& other);**//конструктор копирования

};

Animal.cpp

Animal::Animal (const Animal& other)

{

m\_age = other.m\_age;

m\_weight = other. m\_weight;

m\_pName = new char [strlen(other.m\_pName) + 1];

strcpy\_s(m\_pName, strlen(other.m\_pName)+1, other.m\_pName);

}

{

Animal a1(1, 5, “Bobik”);

Animal a2 = a1;

}

# Перегрузка операторов

## Оператор присваивания =

class Animal

{

…

public:

Animal & operator = (const Animal& other) // оператор присваивания

};

Animal.cpp

Animal& Animal::operator = (const Animal& other)

{

if (this != & other)

{

m\_age = other.m\_age;

m\_weight = other. m\_weight;

delete [] m\_pName;

m\_pName = new char [strlen(other.m\_pName) + 1];

strcpy\_s(m\_pName, strlen(other.m\_pName) +1, other.m\_pName);

}

return \*this;

}

## Перегрузка оператора сравнения

Animal.h

class Animal

{

…

public:

bool operator == (const Animal& r); // оператор сравнения

};

Animal.cpp

bool Animal ::operator == (const Animal& r)

{

return ( m\_age == r.m\_age &&

m\_weight == r.m\_weight &&

strcmp (m\_pName,r. m\_pName) == 0 );

}