// 1

#include <iostream>

using namespace std;

class a1

{

int x;

public:

a1 ()

{x = 0;

}

a1 (int n)

{

x = n;

}

int get()

{

return x;

}

};

class b1 : public a1

{

public:

int get()

{

return (a1 ::get() + 1);

}

b1 (int n) : a1 (n + 1)

{

}

};

int main ()

{

a1 a(10);

b1 b(12);

int x = a.get();

int y = b.get();

cout<< x << endl;

cout<< y << endl;

return 0;

}

//…….. 2

#include <iostream>

using namespace std;

class a2

{

int x;

public:

a2() { x = 0; }

a2(int n)

{

x = n;

}

int inc()

{

return ++x;

}

};

class b2: public a2

{

public:

int inc()

{

int n = a2::inc();

return n-1;

}

b2(int n) : a2(n + 1)

{}

};

int main()

{

a2 a(10);

b2 b(12);

int x = a.inc();

int y = b.inc() + a.inc();

cout<< x << endl;

cout<< y << endl;

return 0;

}

//…….. 3

#include <iostream>

using namespace std;

class a3

{

int x;

public:

a3()

{

x = 0;

}

a3(int n)

{

x = n;

}

int inc()

{

return ++x;

}

};

class b3 : public a3

{

public:

int inc()

{

int n = a3::inc();

return n-1;

}

b3(int n) : a3(n + 1) {}

};

int main()

{

a3 a(10);

b3 b(12);

a3 \*pa = &b;

b3 \*pb = &b;

int x = a.inc();

int y = b.inc() + pa->inc();

int z = pb->inc();

cout<< x << endl;

cout<< y << endl;

return 0;

}

//…………. 4

#include<iostream>

using namespace std;

class a4

{

int x;

public:

virtual int *out*()

{

return x;

}

a4(int n)

{

x = n;

}

};

class b4 : public a4

{

public:

virtual int *out*()

{

return (a4::*out*() + 1);

}

b4(int n) : a4(n) { }

};

class c4 : public a4

{

public:

c4(int n) : a4(n) { }

};

int main()

{

a4 A1 (5);

b4 B1 (6);

c4 C1 (1);

a4 \*p[] = { &A1, &B1, &C1 };

int r1 = p[0]->*out*() + p[1]->*out*() + p[2]->*out*();

int r2 = A1.*out*() + B1.*out*() + C1.*out*();

cout<<r1<<endl<<r2<<endl;

return 0;

}

//…………… 5

#include <iostream>

#include <string.h>

using namespace std;

**class a5**

{

public:

virtual int *put*()=0;

a5() {};

};

class integer : public a5

{

int val;

public:

virtual int *put*()

{

return val;

}

integer(int n)

{

val = n;

}

};

**class String : public a5**

{

char \*str;

public:

virtual int *put*()

{

return strlen(str);

}

String(char \*s)

{

str = s;

}

};

int main()

{

integer a1 (12),a2(24);

String a3("aaaaH");

a5 \*p[4]= { &a1, &a2, &a3, &a1 };

int x=0;

for ( int i = 0; i < 4; i++)

x += p[i]->*put*();

cout <<x<<endl;

return 0;

}

#include <iostream>

using namespace std;

**//абстрактный базовый класс**

**class Animal**

{

public:

char \*Title; //кличка животного

Animal(char \*t) {Title=t;} //простой конструктор

virtual void *speak*(void)=0; //чистая виртуальная функция

};

**//класс лягушка**

**class Frog: public Animal**

{

public:

Frog(char \*Title): Animal(Title) { };

virtual void *speak*(void) { cout<<Title<<" say"<<"'kva-kva'"<<endl; };

};

**//класс собака**

**class Dog: public Animal**

{

public:

Dog(char \*Title): Animal(Title) { };

virtual void *speak*(void) { cout<<Title<<" say "<<"'gav-gav'"<<endl;};

};

**//класс кошка**

**class Cat: public Animal**

{

public:

Cat(char \*Title): Animal(Title) { };

virtual void *speak*(void) { cout<<Title<<" say "<<"'may-may'"<<endl;};

};

**//класс лев**

**class Lion: public Cat**

{

public:

Lion(char \*Title): Cat(Title) { };

virtual void *speak*(void) { cout<<Title<<" say "<<"'rrr-rrr'"<<endl;};

};

int main ()

{

//объявим массив указателей на базовый класс Animal

//и сразу его заполним указателями, создавая объекты

Animal \*animals[4] = { new Dog("Bob"),

new Cat("Murka"),

new Frog("Green"),

new Lion("King")}; // cписок животных

for(int k=0; k<4; k++) animals[k]->*speak*();

return 0;

}