**Exercitiul 5**

**var 1**

#include <iostream>

using namespace std;

class B {

protected:

static int x;

int offset;

public:

B()

{x++; offset = 100;}

~B() { x--; }

static int get\_x() { return x; }

int get\_offset() { return offset; }

int f() { return (x + offset) / 2; }

};

int B::x = 0;

class D : public B {

public:

D() { x++; }

~D() { x--; }

int f() { return ( (x + offset) / 2 + 1); }

};

void func(B\* q, int n) {cout << q->get\_x() << " ";

for(int i = 0; i < n; i++) cout << q[i].f() << " ";

cout<<"\n";

}

int main()

{

B\* p = new B[2]; func(p, 2); delete[] p;

p = new D; func(p, 1); delete p;

cout << D::get\_x(); return 0;

}

//Compileaza 0.1p

//Se afiseaza

//2 51 51 0.2

//2 51 0.1

//1 0.1

------------------------------------------------------

**var 2**

#include <iostream>

using namespace std;

class B {protected:

static int x;

int offset;

public:

B() { x++;offset = 100; }

~B() { x--; }

static int get\_x() { return x; }

int get\_offset() { return offset; }

int f() { return (x + offset) / 2; }

};

int B::x = 0;

class D : public B {public:

D() { x++; }

~D() { x--; }

int f() { return (x + offset) / 2 + 1; }

};

void func(B\* q, int n){cout << q->get\_x() << " ";

for(int i = 0; i < n; i++) cout << q[i].f() << " ";

cout<<"\n";}

int main()

{

B\* p = new B[2];func(p, 2);delete[] p;

p = new D;func(p, 1); delete p;

cout << D::get\_x();

}

//Nu compileaza 0.1

//deoarece incercam sa apelam variabila offset in

//cadrul unei functii statice f() din clasa B 0.2

//Pentru a face programul sa compileze trebuie sa scoatem

//identificatorul static pentru aceasta functie 0.2

-------------------------------------------------------

**var 3**

#include <iostream>

using namespace std;

class B {

protected:

static int x;

int offset;

public:

B() {x++; offset = 100; }

~B() { x--; }

virtual static int get\_x() { return x; }

int get\_offset() { return offset; }

virtual int f() { return (x + offset) / 2; }

};

int B::x = 0;

class D : public B {public:

D() { x++; }

~D() { x--; }

int f() { return (x + offset) / 2 + 1; }

};

void func(B\* q, int n){cout << q->get\_x() << " ";

for(int i = 0; i < n; i++)cout << q[i].f() << " ";

cout<<"\n";}

int main()

{

B\* p = new B[2];func(p, 2);delete[] p;

p = new D;func(p, 1);delete p;

cout << D::get\_x();

}

//Nu compileaza 0.1

//functia get\_x din B nu poate fi declarata si virtual si static (nu exista functii virtuale statice)

//0.2

//Pentru ca programul a compileze trebuie sa eliminam identificatorul virtual pentru get\_x

//0.2

-------------------------------------------------------

**var 4**

#include <iostream>

using namespace std;

class B {

protected:

static int x;

int offset;

public:

B(){ x++;offset = 100;}

~B() { x--; }

static int f() { return x; }

int get\_offset() { return offset; }

virtual int f() { return (x + offset) % 2; }

};

int B::x = 0;

class D : public B {public:

D() { x++; }

~D() { x--; }

int f() { return (x + offset) % 2 + 1; }

};

void func(B\* q, int n){

for(int i = 0; i < n; i++) cout << q[i].f() << " ";

cout<<"\n";

}

int main()

{

B\* p = new B[2];func(p, 2);delete[] p;

p = new D;func(p, 1);delete p;

}

//Nu compileaza 0.1

//Nu putem avea in aceeasi clasa o functie cu aceeasi semnatura

//care sa fie si statica si non-statica in acelasi timp 0.2

//Pentru a compila trebuia de exemplu sa dam un argument x functiei f()

//de la linia 24 pentru a le diferentia semnatura (x va fi aici o variabila locala considerata in loc de x-ul static)

//Sau comentam linia ce contine varianta de functie statica

//Sau comentam linia ce contine varianta virtual int f() 0.2

-------------------------------------------------------

**var 5**

#include <iostream>

using namespace std;

class B {

protected:

static int x;

int offset;

public:

B() { x++; offset = 100; }

~B() { x--; }

static int get\_x() { return x; }

int get\_offset() { return offset; }

static int f() { return (x + get\_offset()) % 2; }

};

int B::x = 0;

class D : public B {public:

D() { x++; }

~D() { x--; }

int f() { return (x + offset) % 2 + 1; }

};

void func(B\* q, int n){

for(int i = 0; i < n; i++)cout << q[i].f() << " ";

cout<<"\n";

}

int main()

{

B\* p = new B[2];func(p, 2); delete[] p;

p = new D; func(p, 1); delete p;

cout << D::get\_x();

}

//Nu compileaza 0.1

//Nu putem apela o metoda non-statica intr-o metoda statica

//(nu exista \*this) 0.2

//Pentru a compila trebuie sa scoatem identificatorul static pentru

//metoda f() din clasa B 0.2