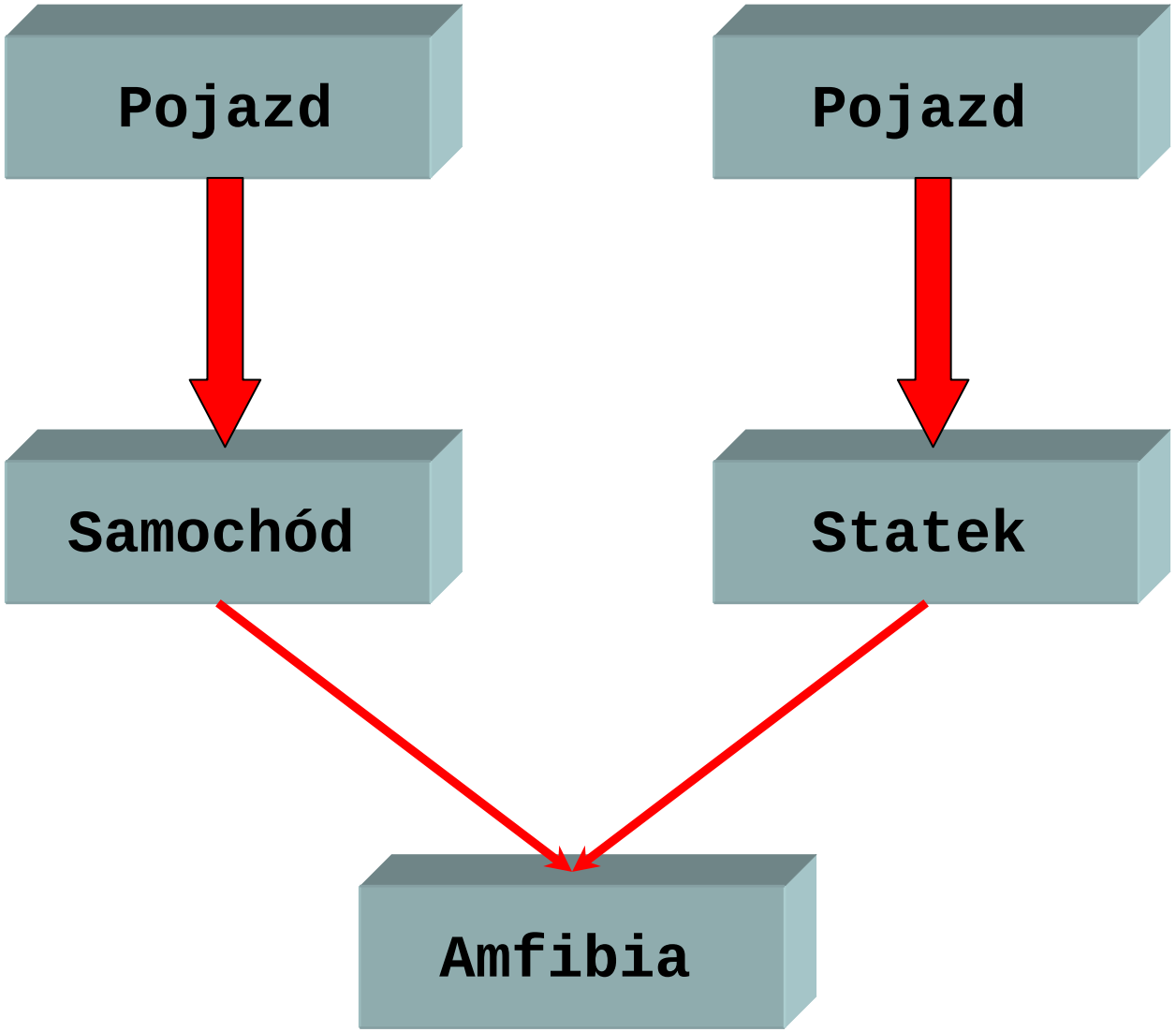
Dziedziczenie ze wspólnej klasy bazowej



```
/*Dziedziczenie wielokrotne
z wspólnej klasy bazowej*/

#include <iostream>
using namespace std;

char* naped_ziemia[]={"benzyna", "diesel" ,"hybryda"};
char* naped_woda[]={"diesel","turbina"};

class Pojazd
{
protected:
	int zaloga;
public:
	Pojazd(int z):zaloga(z)
	{cout<<"Konstruktor pojazdu...\n";}
	virtual void IluLudzi()
	{cout<<"Zaloga pojazdu to "<<zaloga<<" ludzi\n";}
	virtual ~Pojazd()
	{cout<<"Destruktor pojazdu\n";}
};
```

```
class Samochod: public Pojazd
{
public:
char* silnik;
int predkosc;
	Samochod (int,int,int);
	void Jazda(int v)
	{cout<<"Jade "<<v<<" km/h \n";}
	virtual ~Samochod()
	{cout<<"Destruktor samochodu\n";}
};

class Statek : public Pojazd
{
public:
	char* silnik;
	double szybkosc;
	Statek (int,int,int);
	void Plyne(int v)
	{cout<<"Plyne z predkoscia "<<v<<" wezlow\n";}
	virtual ~Statek()
	{cout<<"Statek tonie\n";}
};
```

```
class Amfibia : public Samochod, public Statek
{
	int ladownosc;
public:
	Amfibia(int,int,int,int,int,int);
	Amfibia();
	virtual void IluLudzi()
	{cout<<"Zaloga amfibii to "<<Samochod::zaloga<<"
ludzi\n";}
	~Amfibia()
	{cout<<"Amfibie tez tona\n";}

	void Ladownosc()
	{cout<<"Nie zatone przy ciezarze "<<ladownosc<<" ton
!!\n";}
};

Samochod::Samochod(int z,int i, int pred):
Pojazd(z),silnik(naped_ziemia[i]),predkosc(pred)
{cout<<"Konstruktor samochodu\n";}
```

```
Statek::Statek(int z,int a, int v):
Pojazd(z),silnik(naped_woda[a]),szybkosc(v)
{cout<<"Wodowanie statku\n";}

Amfibia::Amfibia(int z,int i1,int v1,int i2, int v2, int
masa)
:Samochod(z,i1,v1),Statek(z,i2,v2),ladownosc(masa)
{cout<<"Konstruktor amfibii\n";}

Amfibia::Amfibia()
:Samochod(5,0,70),Statek(5,0,8),ladownosc(1)
{cout<<"Konstruktor amfibii\n";}

void main()
{
      cout<<"Aligator\n\n";
      Amfibia aligator(10,0,70,0,6,2);
      aligator.Jazda(30);
      aligator.Ladownosc();
      aligator.Plyne(6);
      //cout<<aligator.silnik<<endl;
      cout<<aligator.Samochod::silnik<<endl;
```

```
        cout<<aligator.Statek::silnik<<endl;
        aligator.Pojazd::IluLudzi();
        aligator.IluLudzi();
        aligator.~Amfibia();
        cout<<"Kubelwagen\n\n";
        Amfibia* kubelwagen= new Amfibia;
        kubelwagen->Jazda(25);
        kubelwagen->Plyne(8);
        kubelwagen->Ladownosc();
        cout<<kubelwagen->predkosc<<endl;
        kubelwagen->IluLudzi();
        cout<<kubelwagen->szybkosc<<endl;

        delete kubelwagen;
        system("pause");
}
```

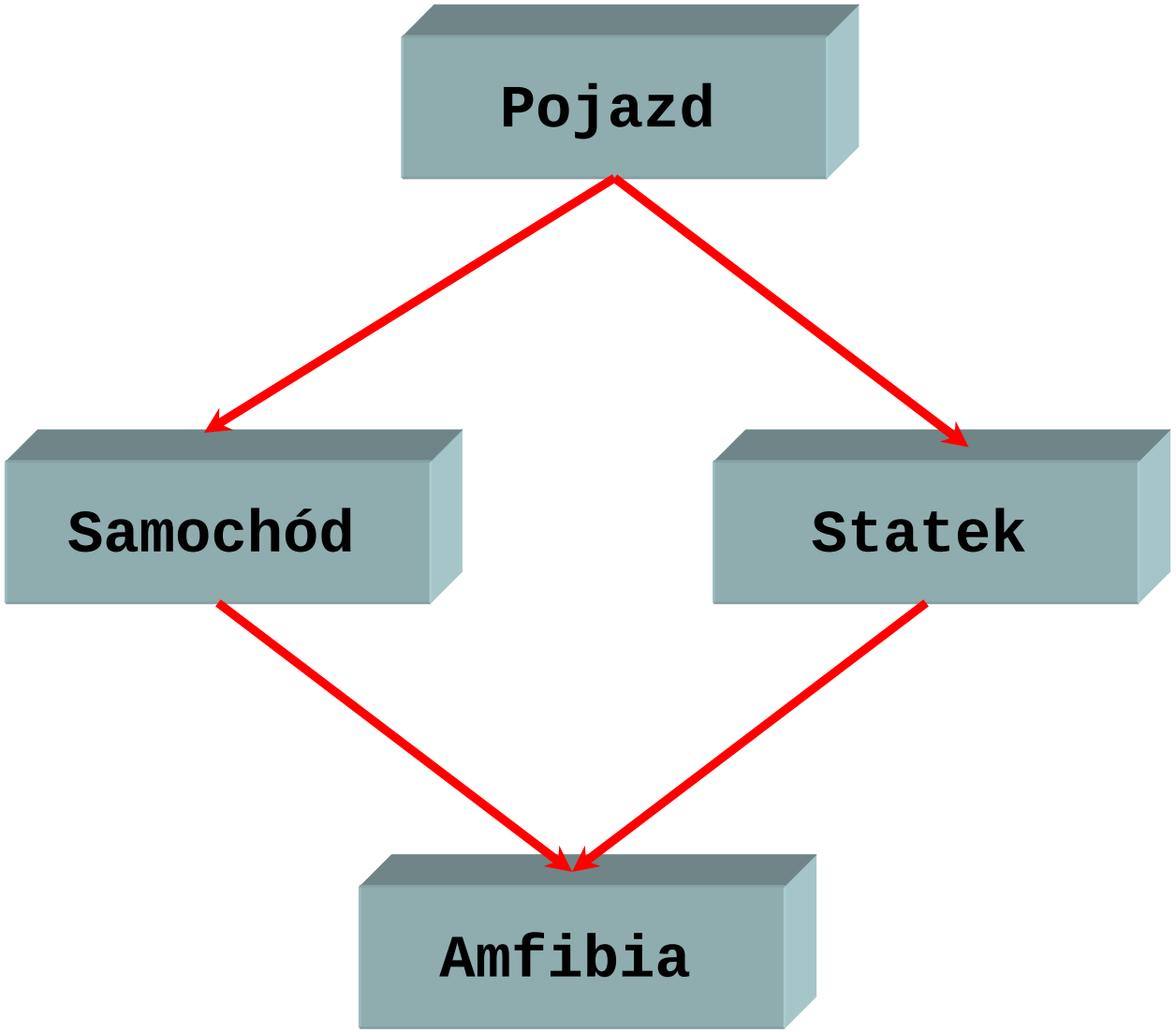
Aligator

Konstruktor pojazdu...
Konstruktor samochodu
Konstruktor pojazdu...
Wodowanie statku
Konstruktor amfibii
Jade 30 km/h
Nie zatone przy ciezarze 2 ton !!
Plyne z predkoscia 6 wezlow
benzyna
diesel
Zaloga pojazdu to 10 ludzi
Zaloga amfibii to 10 ludzi
Amfibie tez tona
Statek tonie
Destruktor pojazdu
Destruktor samochodu
Destruktor pojazdu

```
Kubelwagen

Konstruktor pojazdu...
Konstruktor samochodu
Konstruktor pojazdu...
Wodowanie statku
Konstruktor amfibii
Jade 25 km/h
Plyne z predkoscia 8 wezlow
Nie zatone przy ciezarze 1 ton !!
70
Zaloga amfibii to 5 ludzi
8
Amfibie tez tona
Statek tonie
Destruktor pojazdu
Destruktor samochodu
Destruktor pojazdu
Press any key to continue . . .
```

Dziedziczenie wirtualne



```
/*Dziedziczenie wirtualne
z wspólnej klasy bazowej*/

#include <iostream>
using namespace std;

char* naped_ziemia[]={"benzyna", "diesel" ,"hybryda"};
char* naped_woda[]={"diesel","turbina"};

class Pojazd
{
protected:
	int zaloga;
public:
	Pojazd():zaloga(3)
	{cout<<"Konstruktor domyslny pojazdu...\n";}
	/*Niezbędny konstruktor domyślny
	klasy Pojazd - bez niego błędy
	podczas konstrukcji obiektu klasy Amfibia*/
	Pojazd(int z):zaloga(z)
	{cout<<"Konstruktor pojazdu...\n";}
```

```
virtual void IluLudzi()
	{cout<<"Zaloga pojazdu to "<<zaloga<<" ludzi\n";}
	virtual ~Pojazd()
	{cout<<"Destruktor pojazdu\n";}
};

class Samochod: virtual public Pojazd
{
public:
char* silnik;
int predkosc;
	Samochod (int,int,int);
	void Jazda(int v)
	{cout<<"Jade "<<v<<" km/h \n";}
	virtual ~Samochod()
	{cout<<"Destruktor samochodu\n";}
};

class Statek : virtual public Pojazd
```

```
{
public:
	char* silnik;
	double szybkosc;
	Statek (int,int,int);
	void Plyne(int v)
	{cout<<"Plyne z predkoscia "<<v<<" wezlow\n";}
	virtual ~Statek()
	{cout<<"Statek tonie\n";}
};

class Amfibia : public Samochod, public Statek
{
	int ladownosc;
public:
	Amfibia(int,int,int,int,int,int);
	Amfibia();
	virtual void IluLudzi()
	{cout<<"Zaloga amfibii to "<<Samochod::zaloga<<"
ludzi\n";}
	~Amfibia()
	{cout<<"Amfibie tez tona\n";}
```

```
        void Ladownosc()
        {cout<<"Nie zatone przy ciezarze "<<ladownosc<<" ton
!!\n";}
};

Samochod::Samochod(int z,int i, int pred):
Pojazd(z),silnik(naped_ziemia[i]),predkosc(pred)
{cout<<"Konstruktor samochodu\n";}

Statek::Statek(int z,int a, int v):
Pojazd(z),silnik(naped_woda[a]),szybkosc(v)
{cout<<"Wodowanie statku\n";}

Amfibia::Amfibia(int z,int i1,int v1,int i2, int v2, int masa)
:Samochod(z,i1,v1),Statek(z,i2,v2),ladownosc(masa)
{cout<<"Konstruktor amfibii\n";}

Amfibia::Amfibia()
:Samochod(5,0,70),Statek(5,0,8),ladownosc(1)
{cout<<"Konstruktor amfibii\n";}
```

```
void main()
{
	cout<<"Aligator\n\n";
	Amfibia aligator(10,0,70,0,6,2);
	aligator.Jazda(30);
	aligator.Ladownosc();
	aligator.Plyne(6);
	cout<<aligator.Samochod::silnik<<endl;
	cout<<aligator.Statek::silnik<<endl;
	aligator.Pojazd::IluLudzi();
	aligator.IluLudzi();
	aligator.~Amfibia();
	cout<<"Kubelwagen\n\n";
	Amfibia* kubelwagen= new Amfibia;
	kubelwagen->Jazda(25);
	kubelwagen->Plyne(8);
	kubelwagen->Ladownosc();
	cout<<kubelwagen->predkosc<<endl;
	kubelwagen->IluLudzi();
	cout<<kubelwagen->szybkosc<<endl;
	delete kubelwagen;
	system("pause");
}
```

Aligator

Konstruktor domyslny pojazdu...
Konstruktor samochodu
Wodowanie statku
Konstruktor amfibii
Jade 30 km/h
Nie zatone przy ciezarze 2 ton !!
Plyne z predkoscia 6 wezlow
benzyna
diesel
Zaloga pojazdu to 3 ludzi
Zaloga amfibii to 3 ludzi
Amfibie tez tona
Statek tonie
Destruktor samochodu
Destruktor pojazdu

```
Kubelwagen

Konstruktor domyslny pojazdu...
Konstruktor samochodu
Wodowanie statku
Konstruktor amfibii
Jade 25 km/h
Plyne z predkoscia 8 wezlow
Nie zatone przy ciezarze 1 ton !!
70
Zaloga amfibii to 3 ludzi
8
Amfibie tez tona
Statek tonie
Destruktor samochodu
Destruktor pojazdu
Press any key to continue . . .
```

1. Dziedziczenia wielokrotnego należy używać, gdy nowa klasa wymaga funkcji i cech pochodzących z więcej niż jednej klasy bazowej
2. Dziedziczenia wirtualnego używa się, gdy najbardziej wyprowadzona klasa wymaga tylko jednego egzemplarza wspólnej klasy bazowej
3. Wirtualna klasa bazowa powinna być inicjalizowana w najbardziej wyprowadzonej klasie
4. Dziedziczenia wielokrotnego nie używa się tam, gzie wystarczy dziedziczenie jednokrotne

```
/*Czyste funkcje wirtualne
  Klasa abstrakcyjna*/

#include <iostream>
using namespace std;

class Zwierze
{
public:
       virtual void Glos() = 0;
       Zwierze()
       {cout<<"Konstruktor klasy Zwierze\n";}
       ~Zwierze()
       {cout<<"Destruktor klasy Zwierze\n";}
};
```

```
class Pies : public Zwierze
{
public:
	void Aktywnosc(){cout<<"Pies : Macham ogonem
!!"<<endl;}
	void Glos(){cout<<"Pies : Hau hau !!"<<endl;}
Pies()
	{cout<<"Konstruktor klasy Pies\n";}
	~Pies()
	{cout<<"Destruktor klasy Pies\n";}
};

class Kot : public Zwierze
{
public:
	void Aktywnosc(){cout<<"Kot : Zlapalem mysz !!"<<endl;}
	void Glos(){cout<<"Kot : Miau miau !!"<<endl;}
	Kot()
	{cout<<"Konstruktor klasy Kot\n";}
	~Kot()
	{cout<<"Destruktor klasy Kot\n";}
};
```

```
        void main()
{
        Zwierze* pZwierze1 = new Pies;
        pZwierze1->Glos();
        delete pZwierze1;
        Zwierze* pZwierze2 = new Kot;
        pZwierze2->Glos();
        delete pZwierze2;
        Pies Azor;
        Azor.Glos();
        Azor.Aktywnosc();
        Kot Mruczek;
        Mruczek.Glos();
        Mruczek.Aktywnosc();

        //Zwierze zwierz;
        //Zwierze* pZwierz= new Zwierze;
        /*Nie można stworzyć obiektu
        klasy abstrakcyjnej*/
        system("pause");
}
```

```
Konstruktor klasy Zwierze
Konstruktor klasy Pies
Pies : Hau hau !!
Destruktor klasy Zwierze
Konstruktor klasy Zwierze
Konstruktor klasy Kot
Kot : Miau miau !!
Destruktor klasy Zwierze
Konstruktor klasy Zwierze
Konstruktor klasy Pies
Pies : Hau hau !!
Pies : Macham ogonem !!
Konstruktor klasy Zwierze
Konstruktor klasy Kot
Kot : Miau miau !!
Kot : Zlapalem mysz !!
Press any key to continue . . .
```

```
// Wyprowadzenie klas abstrakcyjnych z innych klas
abstrakcyjnych

#include <iostream>
using namespace std;

class Zwierze
{
public:
	Zwierze();
	virtual ~Zwierze()
	{cout<<"Destruktor klasy Zwierze\n";}
	virtual void Sen() const = 0;
	virtual void Jedz() const = 0;
	virtual void Rozmnazanie() const = 0;
	virtual void Ruch() const = 0;
	virtual void Glos() const = 0;
};

Zwierze::Zwierze()
{cout<<"Konstruktor klasy Zwierze\n";}
```

```
class Ssak : public Zwierze
{
public:
       Ssak ()
       {cout<<"Konstruktor klasy Ssak...\n";}
       virtual ~Ssak()
       {cout<<"Destruktor klasy Ssak\n";}
       virtual void Rozmnazanie()const
       {cout<<"Rozmnazanie klasy Ssak...\n";}
};
class Ryba : public Zwierze
{
public:
       Ryba()
       {cout<<"Konstruktor klasy Ryba...\n";}
       virtual ~Ryba()
       {cout<<"Destruktor klasy Ryba\n";}
       virtual void Sen() const   {cout<<"Ryba spi...\n";}
       virtual void Jedz() const  {cout<<"Ryba je...\n";}
       virtual void Rozmnazanie() const  {cout<<"Ryba sklada
jaja...\n";}
       virtual void Ruch() const  {cout<<"Ryba plywa...\n";}
       virtual void Glos() const  {cout<<"Ryba milczy...\n";}
};
```

```
class Kon: public Ssak
{
public:
	Kon()
	{cout<<"Konstruktor klasy Kon...\n";}
	virtual ~Kon()
	{cout<<"Destruktor klasy Kon...\n";}
	virtual void Glos() const  {cout<<"Ihahahaha !!!\n";}
	virtual void Sen() const   {cout<<"Kon spi !!!\n";}
	virtual void Jedz() const  {cout<<"Kon sie pasie
!!!\n";}
	virtual void Ruch() const  {cout<<"Kon galopuje
!!!\n";}
};
class Pies: public Ssak
{
public:
	Pies()
	{cout<<"Konstruktor klasy Pies...\n";}
	virtual ~Pies()
	{cout<<"Destruktor klasy Pies...\n";}
```

```
virtual void Glos() const  {cout<<"Hau !!\n";}
	virtual void Sen() const   {cout<<"Pies spi !!!\n";}
	virtual void Jedz() const  {cout<<"Pies je !!!\n";}
	virtual void Ruch() const  {cout<<"Pies biegnie
!!!\n";}
};
void main()
{
	Zwierze *ptr = 0;
	int wybor;
	bool Wyjscie = false;
	while (Wyjscie == false)
	{
		cout<<"(1) Pies (2)Kon (3)Ryba (0)Wyjscie : ";
		cin>>wybor;
		switch(wybor)
		{
			case 1: ptr = new Pies;
				break;
			case 2:	ptr = new Kon;
				break;
			case 3:	ptr = new Ryba;
				break;
```

```
                        default: Wyjscie = true;
                                break;
                }
if (Wyjscie == false)
                {
                        ptr->Glos();
                        ptr->Jedz();
                        ptr->Rozmnazanie();
                        ptr->Ruch();
                        ptr->Sen();
                        delete ptr;
                        cout<<endl;
                }
        }
        system("pause");
}
```

```
(1) Pies (2)Kon (3)Ryba (0)Wyjscie : 1
Konstruktor klasy Zwierze
Konstruktor klasy Ssak...
Konstruktor klasy Pies...
Hau !!
Pies je !!!
Rozmnazanie klasy Ssak...
Pies biegnie !!!
Pies spi !!!
Destruktor klasy Pies...
Destruktor klasy Ssak
Destruktor klasy Zwierze

(1) Pies (2)Kon (3)Ryba (0)Wyjscie : 2
Konstruktor klasy Zwierze
Konstruktor klasy Ssak...
Konstruktor klasy Kon...
Ihahahaha !!!
Kon sie pasie !!!
Rozmnazanie klasy Ssak...
Kon galopuje !!!
Kon spi !!!
Destruktor klasy Kon...
Destruktor klasy Ssak
Destruktor klasy Zwierze

(1) Pies (2)Kon (3)Ryba (0)Wyjscie : 3
Konstruktor klasy Zwierze
Konstruktor klasy Ryba...
Ryba milczy...
Ryba je...
Ryba sklada jaja...
Ryba plywa...
Ryba spi...
Destruktor klasy Ryba
Destruktor klasy Zwierze

(1) Pies (2)Kon (3)Ryba (0)Wyjscie : 0
Press any key to continue . . .
```

```
void main()
{
        //Zwierze zwierz;
        //Ssak krolik;
        cout<<endl;
        Pies Azor;
        cout<<endl;
        Azor.Glos();
        Azor.Jedz();
        Azor.Rozmnazanie();
        Azor.Ruch();
        Azor.Sen();
        Azor.~Pies();
        cout<<endl;
        Kon Lysek;
        Lysek.Glos();
        Lysek.Jedz();
        Lysek.Rozmnazanie();
        Lysek.Ruch ();
        Lysek.Sen();
        Lysek.~Kon();
        cout<<endl;
```

```
Ryba karp;
	karp.Glos();
	karp.Jedz();
	karp.Rozmnazanie();
	karp.Ruch();
	karp.Sen();
	karp.~Ryba();
	cout<<endl;
	system("pause");
}
```

```
Konstruktor klasy Zwierze
Konstruktor klasy Ssak...
Konstruktor klasy Pies...

Hau !!
Pies je !!!
Rozmnazanie klasy Ssak...
Pies biegnie !!!
Pies spi !!!
Destruktor klasy Pies...
Destruktor klasy Ssak
Destruktor klasy Zwierze

Konstruktor klasy Zwierze
Konstruktor klasy Ssak...
Konstruktor klasy Kon...
Ihahahaha !!!
Kon sie pasie !!!
Rozmnazanie klasy Ssak...
Kon galopuje !!!
Kon spi !!!
Destruktor klasy Kon...
Destruktor klasy Ssak
Destruktor klasy Zwierze
```

```
Konstruktor klasy Zwierze
Konstruktor klasy Ryba...
Ryba milczy...
Ryba je...
Ryba sklada jaja...
Ryba plywa...
Ryba spi...
Destruktor klasy Ryba
Destruktor klasy Zwierze

Press any key to continue . . .
```