```
// Referencja

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

string Rodzaj[4]=   {"TV      ", "wieza    ", "DVD      ",
"kino     "};
string Producent[4]={"Phillips", "Sony     ", "Sanyo    ",
"Samsung  "};

class Sprzet_AV
{
      string rodzaj;
      string firma;

public:
      int numer_fabr;

      void GetFirm(Sprzet_AV & ref);
      void GetSort(Sprzet_AV & ref);
```

```
        void GetFirm()
        {cout<<firma<<"\t";}

        void GetSort()
        {cout<<rodzaj<<"\t";}

        Sprzet_AV(int a1, int a2, int a3)

        :rodzaj(Rodzaj[a1]),firma(Producent[a2]),numer_fabr(a3)
{}
        };

void Sprzet_AV::GetFirm(Sprzet_AV &ref)
{cout<<firma<<"\t";}

void Sprzet_AV::GetSort(Sprzet_AV& ref)
{cout<<rodzaj<<"\t";}

void GetNumber(Sprzet_AV& ref)
{cout<<ref.numer_fabr<<"\n";}

void GetNumber(Sprzet_AV* ptr)
{cout<<ptr->numer_fabr<<"\n";}
```

```
/*void GetNumber()
{cout<<numer_fabr<<"\t";}*/

void main()
{
	Sprzet_AV* sprzet_1=new Sprzet_AV(0,1,1234);
	Sprzet_AV* sprzet_2=new Sprzet_AV(1,3,2345);
	cout<<"Rozmiar wskaznika = "<<sizeof(sprzet_1)<<endl;
	cout<<"Adres wskaznika 1 = "<<&sprzet_1<<endl;
	cout<<"Adres wskaznika 2 = "<<&sprzet_2<<endl;

	Sprzet_AV& referencja=*sprzet_1;
	cout<<"Rozmiar referencji =
"<<sizeof(referencja)<<endl;
	cout<<"Adres referencji = "<<&referencja<<endl;

	cout<<"sprzet_1 :\n";
	sprzet_1->GetFirm(referencja);
	sprzet_1->GetSort(referencja);
```

```
cout<<"\nWywolanie przez referencje\n";
      referencja.GetFirm(referencja);
      referencja.GetSort(referencja);
      cout<<endl;
      GetNumber(sprzet_1);
      GetNumber(referencja);
      cout<<"Funkcje bez parametrow\n";
      referencja.GetFirm();
      referencja.GetSort();
      cout<<endl;
      sprzet_1->GetFirm();
      sprzet_1->GetSort();
      cout<<endl;

      referencja=*sprzet_2;

      cout<<"Adres referencji = "<<&referencja<<endl;

      cout<<"sprzet_2 :\n";
      sprzet_2->GetFirm(referencja);
      sprzet_2->GetSort(referencja);
```

```
cout<<"\nWywolanie przez referencje\n";
	referencja.GetFirm(referencja);
	referencja.GetSort(referencja);
	cout<<endl;
	GetNumber(sprzet_2);
	GetNumber(referencja);
	cout<<"Funkcje bez parametrow\n";
	referencja.GetFirm();
	referencja.GetSort();
	cout<<endl;
	sprzet_2->GetFirm();
	sprzet_2->GetSort();
	cout<<endl;

	system("pause");
}
```

```
Rozmiar wskaznika = 4
Adres wskaznika 1 = 002DFDAC
Adres wskaznika 2 = 002DFDA0
Rozmiar referencji = 68
Adres referencji = 006B4B38
sprzet_1 :
Sony            TV
Wywolanie przez referencje
Sony            TV
1234
1234
Funkcje bez parametrow
Sony            TV
Sony            TV
Adres referencji = 006B4B38
sprzet_2 :
Samsung         wieza
Wywolanie przez referencje
Samsung         wieza
2345
2345
Funkcje bez parametrow
Samsung         wieza
Samsung         wieza
Press any key to continue . . .
```

```
// Funkcje zaprzyjaznione

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

string Rodzaj[4]=   {"TV      ", "wieza    ", "DVD      ",
"kino     "};
string Producent[4]={"Phillips", "Sony     ", "Sanyo    ",
"Samsung  "};

class Sprzet_AV
{
      string rodzaj;
      string firma;

public:
      int numer_fabr;

      friend void GetFirm(Sprzet_AV & ref);
      friend void GetSort(Sprzet_AV & ref);
      friend void GetFirm(Sprzet_AV * ptr);
      friend void GetSort(Sprzet_AV * ptr);
```

```
        void GetFirm()
        {cout<<firma<<"\t";}

        void GetSort()
        {cout<<rodzaj<<"\t";}

        Sprzet_AV(int a1, int a2, int a3)

        :rodzaj(Rodzaj[a1]),firma(Producent[a2]),numer_fabr(a3)
{}
        };

void GetFirm(Sprzet_AV &ref)
{cout<<ref.firma<<"\t";}

void GetSort(Sprzet_AV& ref)
{cout<<ref.rodzaj<<"\t";}

void GetFirm(Sprzet_AV *ptr)
{cout<<ptr->firma<<"\t";}
```

```
void GetSort(Sprzet_AV *ptr)
{cout<<ptr->rodzaj<<"\t";}


void GetNumber(Sprzet_AV& ref)
{cout<<ref.numer_fabr<<"\n";}

void GetNumber(Sprzet_AV* ptr)
{cout<<ptr->numer_fabr<<"\n";}

void main()
{
      Sprzet_AV* sprzet_1=new Sprzet_AV(0,1,1234);
      Sprzet_AV& referencja=*sprzet_1;

      cout<<"sprzet_1 :\nWywolanie przez referencje\n";
      GetFirm(referencja);
      GetSort(referencja);
      GetNumber(referencja);
      cout<<endl;
```

```
cout<<"sprzet_1 :\nWywolanie przez wskaznik\n";
	GetFirm(sprzet_1);
	GetSort(sprzet_1);
	GetNumber(sprzet_1);
	cout<<"\nFunkcje bez parametrow\n";
	referencja.GetFirm();
	referencja.GetSort();
	cout<<endl;
	sprzet_1->GetFirm();
	sprzet_1->GetSort();
	cout<<endl;

	system("pause");
}
```

```
sprzet_1 :
Wywolanie przez referencje
Sony            TV              1234

sprzet_1 :
Wywolanie przez wskaznik
Sony            TV              1234

Funkcje bez parametrow
Sony            TV
Sony            TV
Press any key to continue . . .
```

```
/* Modyfikator const
nie pozwala na zmianę składowych klasy*/

#include<iostream>
using namespace std;

char* skills[3]={"aportuje","daje lape","daje glos"};

class Pies
{
	char* imie;

public:

	char* umiejetnosc;

	Pies(char* name, int skill);

	void Naucz(int a)
	{umiejetnosc=skills[a];}
```

```
/*void Learn(int a) const
        {umiejetnosc=skills[a];}
        sama deklaracja funkcji sprawia
        że skladowa jest const*/

};
Pies::Pies(char *name, int skill)
{
        imie=name;
        umiejetnosc=skills[skill];
}

void main()
{
        Pies Azor("Azor", 1);
        //Azor.Learn(0);
        Azor.Naucz(2);
        cout<<Azor.umiejetnosc<<endl;

        system("pause");
}
daje glos
Press any key to continue . . .
```

Przeładowanie operatorów

```
// Przeciążenie operatorów

#include<iostream>
using namespace std;

class Pudlo
{
public:

        double dlugosc;
        double szerokosc;
        double wysokosc;

 Pudlo(double l=1.0, double b=1.0, double h=1.0)
        {
                dlugosc=l;
                szerokosc=b;
                wysokosc=h;
        }

        double Pojemnosc()
        {
                return dlugosc*szerokosc*wysokosc;
        }
```

```
//przeładowany operator większości
        bool operator>(Pudlo pudlo);

//przeładowany operator dodawania
        Pudlo operator+(Pudlo* ptr)
                {
                //Nowy obiekt
                return Pudlo(dlugosc > ptr->dlugosc ?
                dlugosc : ptr->dlugosc,
                szerokosc > ptr->szerokosc ?
                szerokosc : ptr->szerokosc,
                wysokosc+ptr->wysokosc);
        }
};

        bool Pudlo::operator>(Pudlo pudlo)
        {
        return this->Pojemnosc() > pudlo.Pojemnosc();
        }
```

```
void main()
{
	Pudlo pudlo_1(2,3,2);
	Pudlo pudlo_2(4,2,3);
	Pudlo* ptr=&pudlo_2;
	bool wynik;
	cout<<"Porownywanie pudelek\n";
	wynik=pudlo_1>pudlo_2;
	cout<<wynik<<endl;
	cout<<"Dodawanie pudelek\n";
	Pudlo pudlo_3=pudlo_1+ptr;
	cout<<"Dlugosc = "<<pudlo_3.dlugosc<<endl;
	cout<<"Szerokosc = "<<pudlo_3.szerokosc<<endl;
	cout<<"Wysokosc = "<<pudlo_3.wysokosc<<endl;

	system("pause");
}
```

```
Porownywanie pudelek
0
Dodawanie pudelek
Dlugosc = 4
Szerokosc = 3
Wysokosc = 5
Press any key to continue . . .
```

Przeciążony operator przypisania

```
// Przeciążony operator przypisania

// Przeciążenie operatorów

#include<iostream>
using namespace std;

class Pudlo
{
public:

       double dlugosc;
       double szerokosc;
       double wysokosc;

 Pudlo(double l=1.0, double b=1.0, double h=1.0)
       {
              dlugosc=l;
              szerokosc=b;
              wysokosc=h;
       }
```

```
//przeładowany operator przypisania
        Pudlo operator=(Pudlo& pudelko)
        {
                //Nowy obiekt
                return Pudlo();
        }
};

void main()
{
        Pudlo pudlo_1(2,3,2);
        Pudlo& ref=pudlo_1;
        Pudlo pudlo_2=ref;
        cout<<"Klonowanie pudelek\n";
        cout<<"Dlugosc = "<<pudlo_2.dlugosc<<endl;
        cout<<"Szerokosc = "<<pudlo_2.szerokosc<<endl;
        cout<<"Wysokosc = "<<pudlo_2.wysokosc<<endl;
        Pudlo pudlo_3;
```

```
cout<<"pudlo_3\n";
	cout<<"Dlugosc =
"<<pudlo_3.dlugosc<<endl;
	cout<<"Szerokosc =
"<<pudlo_3.szerokosc<<endl;
	cout<<"Wysokosc =
"<<pudlo_3.wysokosc<<endl;

	system("pause");
}
```

```
Klonowanie pudelek
Dlugosc = 2
Szerokosc = 3
Wysokosc = 2
pudlo_3
Dlugosc = 1
Szerokosc = 1
Wysokosc = 1
Press any key to continue . . .
```

```
// Przeładowane operatory

#include<iostream>
using namespace std;

class Pudlo
{
public:

        double dlugosc;
        double szerokosc;
        double wysokosc;

 Pudlo(double l=1.0, double b=1.0, double h=1.0)
        {
                dlugosc=l;
                szerokosc=b;
                wysokosc=h;
        }
        double Pojemnosc()
        {
                return dlugosc*szerokosc*wysokosc;
        }
```

```
//przeładowane operatory większości
	bool operator>(Pudlo pudlo);
	//bool operator>(Walec walec);

//przeładowany operator dodawania
	Pudlo operator+(Pudlo pudlo)
		{
		return Pudlo(dlugosc > pudlo.dlugosc ? dlugosc :
pudlo.dlugosc,
				szerokosc > pudlo.szerokosc ? szerokosc :
pudlo.szerokosc,
				wysokosc+ pudlo.wysokosc);
	}
};

class Walec
{
public:

	double promien;
	double wysokosc;
```

```
Walec(double r=1.0, double h=1.0)
	{
		promien=r;
		wysokosc=h;
	}
	double Pojemnosc()
	{
		return 3.1415927*promien*promien*wysokosc;
	}
//przeładowane operatory większości
	bool operator>(Pudlo pudlo);
	bool operator>(Walec walec);

//przeładowany operator dodawania
	Walec operator+(Walec walec)
		{
		return Walec(promien > walec.promien ? promien :
walec.promien,
				wysokosc+ walec.wysokosc);
	}
};
```

```
bool Pudlo::operator>(Pudlo pudlo)
{
return this->Pojemnosc() > pudlo.Pojemnosc();
}
/*bool Pudlo::operator>(Walec walec)
{
return this->Pojemnosc() > walec.Pojemnosc();
}*/
bool Walec::operator >(Pudlo pudlo)
{
return this->Pojemnosc() > pudlo.Pojemnosc();
}
bool Walec::operator >(Walec walec)
{
return this->Pojemnosc() > walec.Pojemnosc();
}

void main()
{
	Pudlo pudlo_1(2,3,2);
	Pudlo* pudlo_2=new Pudlo;
	Pudlo pudlo_3=pudlo_1+*pudlo_2;
```

```
cout<<"Klonowanie pudelek\n";
	cout<<"Dlugosc = "<<pudlo_3.dlugosc<<endl;
	cout<<"Szerokosc = "<<pudlo_3.szerokosc<<endl;
	cout<<"Wysokosc = "<<pudlo_3.wysokosc<<endl;
	Walec walec_1(2.0,2.0);
	Walec walec_2;
	Walec walec_3=walec_2+walec_1;
	cout<<"\nKlonowanie walcow\n";
	cout<<"Promien = "<<walec_3.promien<<endl;
	cout<<"Wysokosc = "<<walec_3.wysokosc<<endl;

	cout<<"\nPorownywanie objetosci\n";
	bool wynik;
	//wynik=pudlo_2>walec_2;
	wynik=walec_2>*pudlo_2;
	cout<<wynik<<endl;
	wynik=walec_1>walec_2;
	cout<<wynik<<endl;
	wynik=*pudlo_2>pudlo_1;
	cout<<wynik<<endl;

	system("pause");
}
```

```
Klonowanie pudelek
Dlugosc = 2
Szerokosc = 3
Wysokosc = 3

Klonowanie walcow
Promien = 2
Wysokosc = 3

Porownywanie objetosci
1
1
0
Press any key to continue . . .
```

```
/* Przeładowane operatory - wersja z każdą klasą w oddzielnym
pliku*/

#include "Pudlo.h"
#include "Walec.h"

#include<iostream>
using namespace std;


void main()
{
	Pudlo pudlo_1(2,3,2);
	Pudlo* pudlo_2=new Pudlo;
	Pudlo pudlo_3=pudlo_1+*pudlo_2;
	cout<<"Klonowanie pudelek\n";
	cout<<"Dlugosc = "<<pudlo_3.dlugosc<<endl;
	cout<<"Szerokosc = "<<pudlo_3.szerokosc<<endl;
	cout<<"Wysokosc = "<<pudlo_3.wysokosc<<endl;
	Walec walec_1(2.0,2.0);
	Walec walec_2;
	Walec walec_3=walec_2+walec_1;
```

```
cout<<"\nKlonowanie walcow\n";
	cout<<"Promien = "<<walec_3.promien<<endl;
	cout<<"Wysokosc = "<<walec_3.wysokosc<<endl;

	cout<<"\nPorownywanie objetosci\n";
	bool wynik;
	//wynik=pudlo_2>walec_2;
	//wynik=walec_2>*pudlo_2;

	wynik=walec_1>walec_2;
	cout<<wynik<<endl;
	wynik=*pudlo_2>pudlo_1;
	cout<<wynik<<endl;

	system("pause");
}
```

```
//Walec.h
#pragma once
#include "Pudlo.h"

class Walec
{
public:
double promien;
double wysokosc;

Walec(double r=1.0, double h=1.0);

double Pojemnosc();

//bool operator>(Pudlo pudlo);
bool operator>(Walec walec);

Walec operator+(Walec walec);
};
```

```
#include "Walec.h"
#include "Pudlo.h"

Walec::Walec(double r, double h)
	{
		promien=r;		wysokosc=h;
	}
double Walec::Pojemnosc()
	{return 3.1415927*promien*promien*wysokosc;	}

Walec Walec::operator+(Walec walec)
{	return Walec(promien > walec.promien ? promien :
	walec.promien,	wysokosc+ walec.wysokosc); }

/*bool Walec::operator >(Pudlo pudlo)
{
	return this->Pojemnosc() > pudlo.Pojemnosc();
}*/

bool Walec::operator >(Walec walec)
{
	return this->Pojemnosc() > walec.Pojemnosc();
}
```

```
//Pudlo.h
#pragma once
#include "Walec.h"

class Pudlo
{
public:

	double dlugosc;
	double szerokosc;
	double wysokosc;

Pudlo(double l=1.0, double b=1.0, double h=1.0);

double Pojemnosc();

bool operator>(Pudlo pudlo);
//bool operator>(Walec walec);

Pudlo operator+(Pudlo pudlo);
};
```

```
#include "Pudlo.h"
#include "Walec.h"

Pudlo::Pudlo(double l, double b, double h)
{
	dlugosc=l;
	szerokosc=b;
	wysokosc=h;
}
double Pudlo::Pojemnosc()
{
	return dlugosc*szerokosc*wysokosc;
}

Pudlo Pudlo::operator+(Pudlo pudlo)
{
	return Pudlo(dlugosc > pudlo.dlugosc ? dlugosc :
pudlo.dlugosc,
	szerokosc > pudlo.szerokosc ? szerokosc :
pudlo.szerokosc,
	wysokosc+ pudlo.wysokosc);
}
```

```
bool Pudlo::operator>(Pudlo pudlo)
{
	return this->Pojemnosc() > pudlo.Pojemnosc();
}
/*bool Pudlo::operator>(Walec walec)
{
	return this->Pojemnosc() > walec.Pojemnosc();
}*/


Klonowanie pudelek
Dlugosc = 2
Szerokosc = 3
Wysokosc = 3

Klonowanie walcow
Promien = 2
Wysokosc = 3

Porownywanie objetosci
1
0
Aby kontynuować, naciśnij dowolny klawisz . . .
```

```
//Szablon funkcji

#include "Pudlo.h"
#include "Walec.h"
#include<iostream>
using namespace std;

template <class T1,class T2> bool Porownaj(T1 obiekt_1, T2
obiekt_2)
{return obiekt_1.Pojemnosc()>obiekt_2.Pojemnosc();}

void main()
{
	Walec walec;
	Walec walec_1(2,2);
	Pudlo pudlo;
	Pudlo pudlo_1(2,2,2);
	bool wynik;
	wynik=Porownaj(walec, pudlo);
	cout<<wynik<<endl;
	wynik=Porownaj(walec, walec_1);
	cout<<wynik<<endl;
```

```
wynik=Porownaj(pudlo_1,walec_1);
	cout<<wynik<<endl;
	wynik=Porownaj(pudlo_1, pudlo);
	cout<<wynik<<endl;
	system("pause");
}


#pragma once

class Pudlo
{
public:

	double dlugosc;
	double szerokosc;
	double wysokosc;

Pudlo(double l=1.0, double b=1.0, double h=1.0);

double Pojemnosc();

};
```

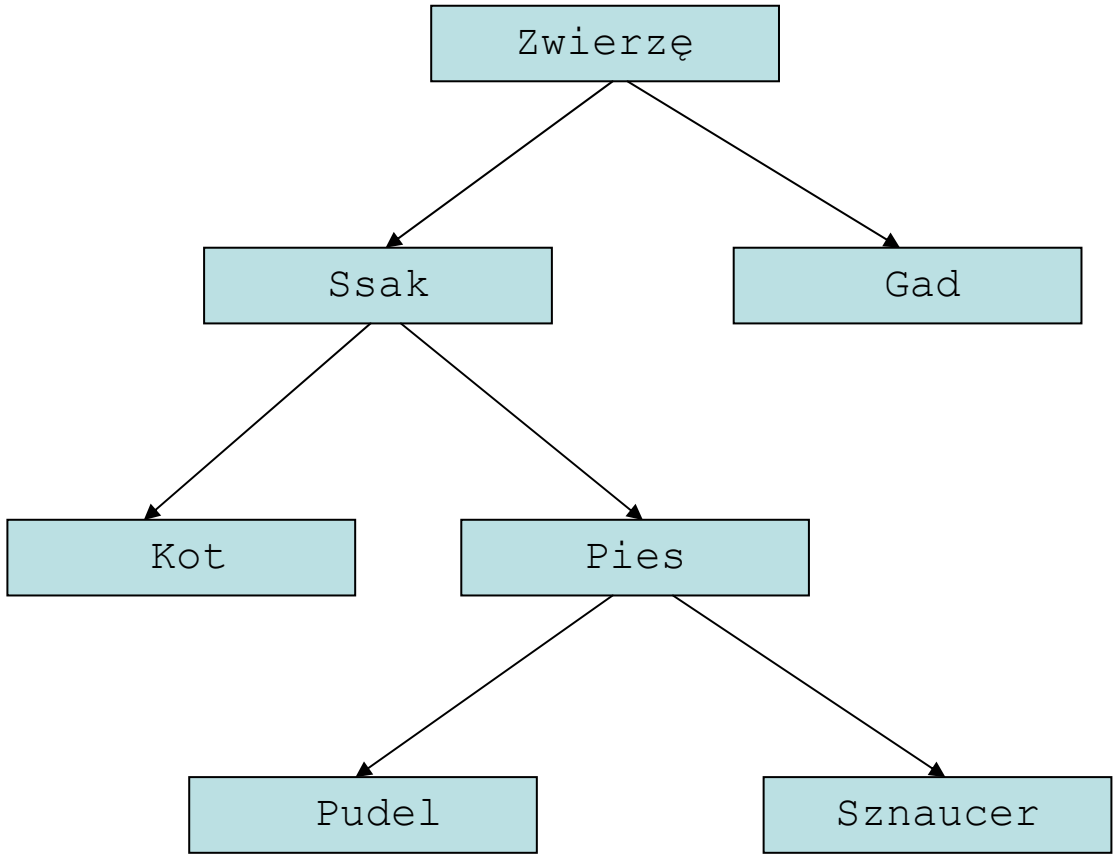
```
#include "Pudlo.h"

Pudlo::Pudlo(double l, double b, double h)
{
	dlugosc=l;
	szerokosc=b;
	wysokosc=h;
}
double Pudlo::Pojemnosc()
{
	return dlugosc*szerokosc*wysokosc;
}
```

```
#pragma once

class Walec
{
public:
double promien;
double wysokosc;

Walec(double r=1.0, double h=1.0);

double Pojemnosc();

};
```

```
include "Walec.h"

Walec::Walec(double r, double h)
	{
		promien=r;
		wysokosc=h;
	}
double Walec::Pojemnosc()
	{
		return 3.1415927*promien*promien*wysokosc;
	}
```

```
1
0
0
1
Aby kontynuować, naciśnij dowolny klawisz . . .
```

Dziedziczenie



Mechanizm pozwalający na wyprowadzanie obiektów klas pochodnych z klas bazowych

Klasa bazowa – klasa na podstawie które tworzymy nową klasę.

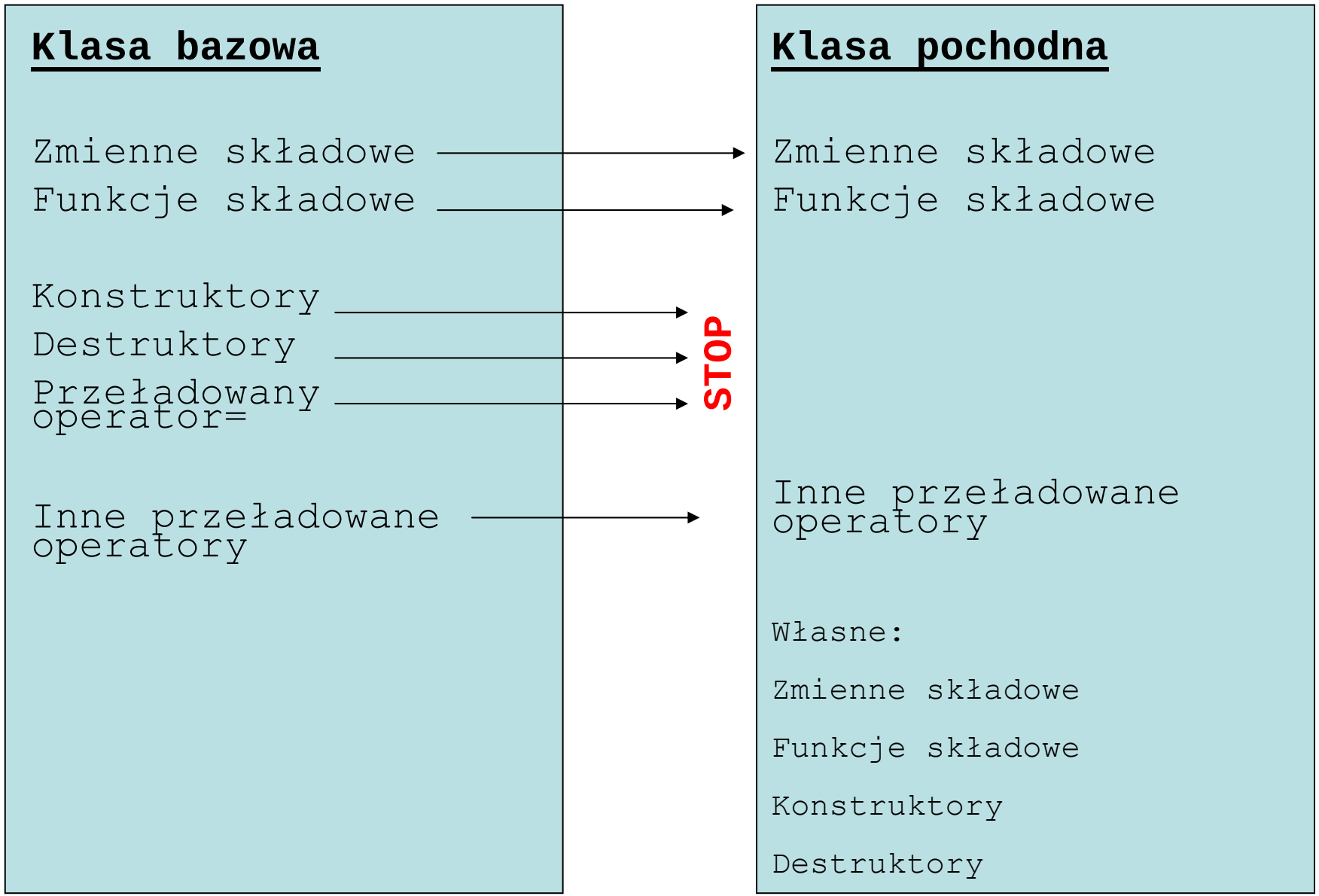
Klasa A → Klasa B →Klasa C

A – bezpośrednia kasa bazowa dla B

B – bezpośrednia kasa bazowa dla C

A – pośrednia kasa bazowa dla C

Dziedziczenie w klasach



Poziom dostępu do dziedziczonych składowych klasy