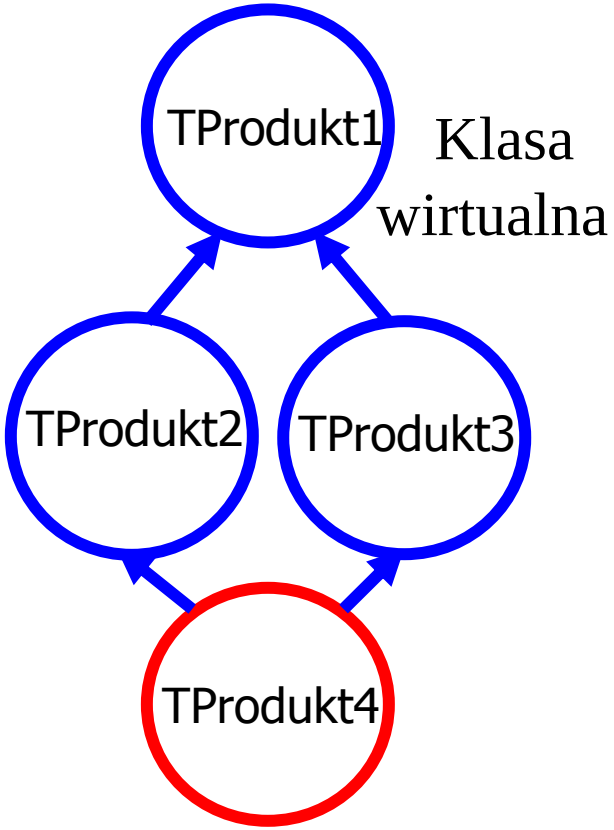
Dziedziczenie wielobazowe- uzupełnienie

1. **Dziedziczenie wielobazowe z powtórzeniami z klasą wirtualną, listy konstruktorów – listy argumentów konstruktorów**
2. **Dziedziczenie wielobazowe z powtórzeniami bez klasy wirtualnej, listy konstruktorów – problemy z polimorfizmem związanym z klasą TProdukt1 dziedziczoną dwukrotnie**
3. **Dziedziczenie wielobazowe bez powtórzeń**
4. **Wzorzec fasady – klasa TAplikacja hermetyzująca dostęp do kolekcji produktów i rachunków**

Dziedziczenie wielobazowe- uzupełnienie

1. **Dziedziczenie wielobazowe z powtórzeniami z klasą wirtualną, listy konstruktorów – listy argumentów konstruktorów**

Dziedziczenie wielobazowe z powtórzeniami



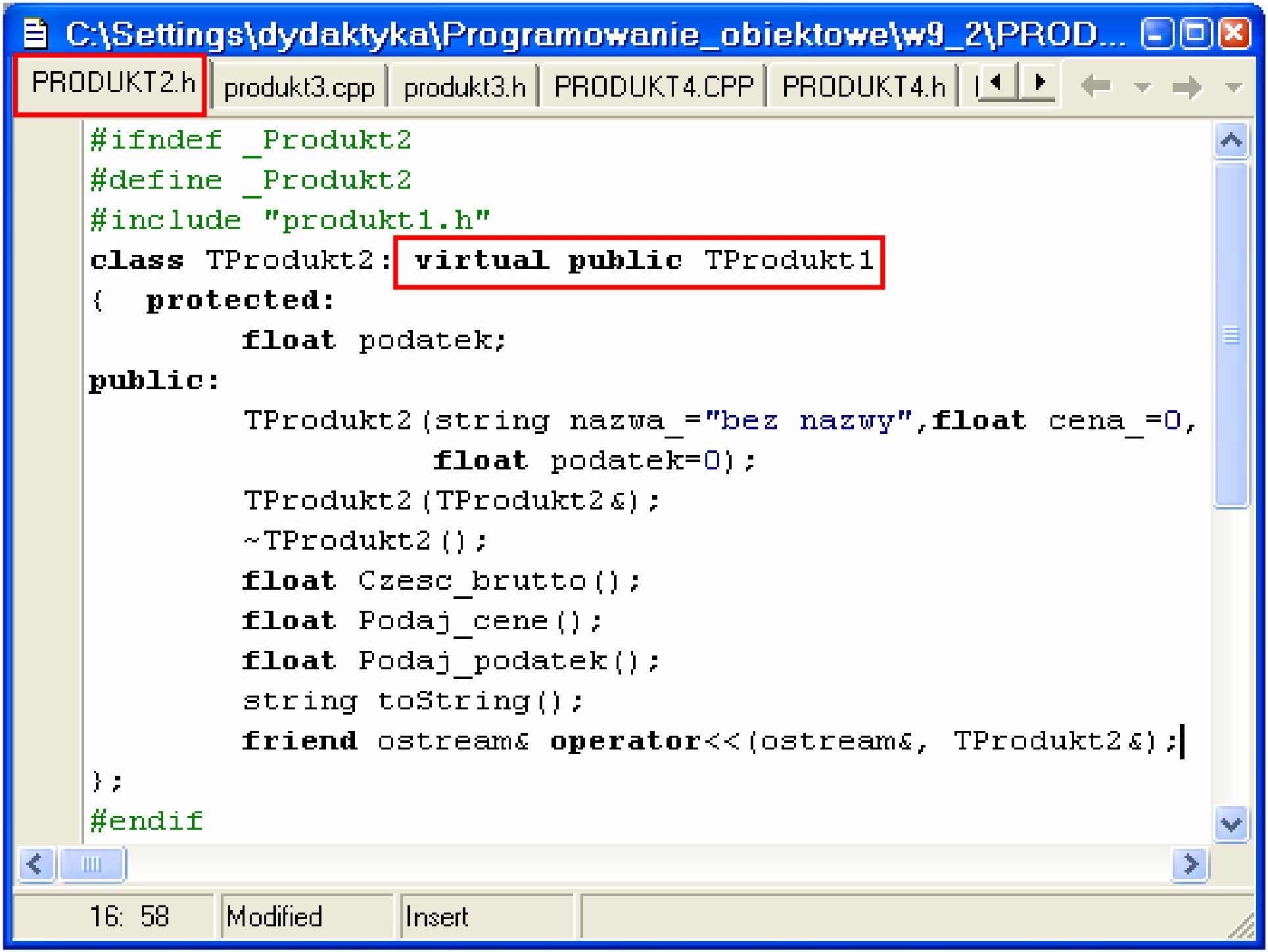
Konstruktory klas dziedziczących wielobazowo z klasą wirtualną

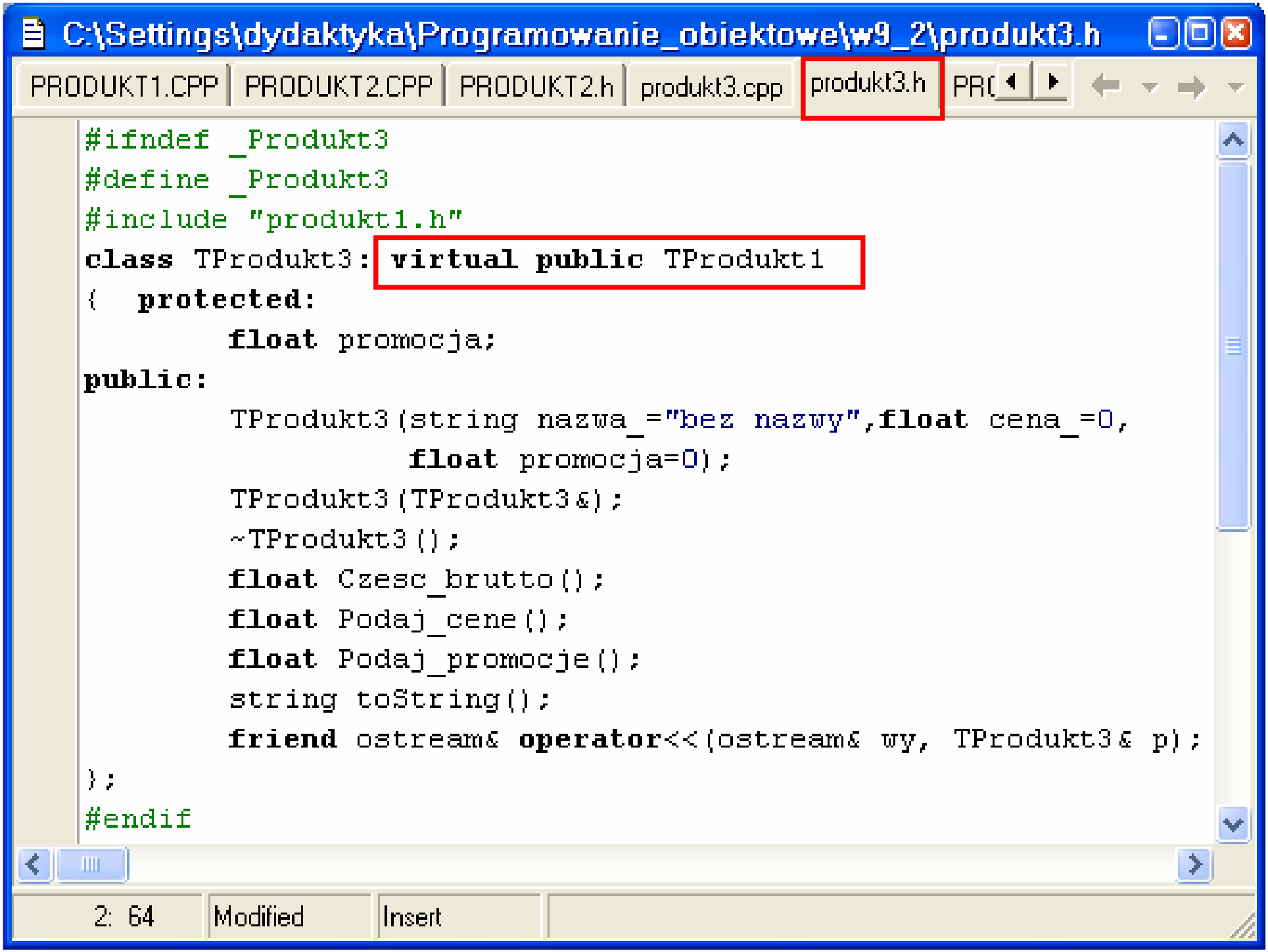
```
TProdukt4::TProdukt4(string nazwa_, float cena_,
                     float podatek_, float promocja_):
   TProdukt1(nazwa_, cena_),
   TProdukt2(nazwa_,cena_,podatek_),
   TProdukt3(nazwa_,cena_,promocja_)
    {  }
TProdukt4::TProdukt4(TProdukt4& p):
   TProdukt1(p),
   TProdukt2(p),
   TProdukt3(p)
    {  }
```

Deklaracje klas dziedziczących wielobazowo z klasą wirtualną

np.

```
class TProdukt1
        {.................};
class TProdukt2 : public virtual TProdukt1
                {.................};
class TProdukt3 : public virtual TProdukt1
                {.................};
class TProdukt4 : public TProdukt2, public TProdukt3
        {.................};
```


C:\Settings\dydaktyka\Programowanie_obiektowe\w9_2\PROD...
PRODUKT2.h
produkt3.cpp
produkt3.h
PRODUKT4.CPP
PRODUKT4.h
#ifndef _Produkt2
#define _Produkt2
#include "produkt1.h"
class TProdukt2: virtual public TProdukt1
{ protected:
float podatek;
public:
TProdukt2(string nazwa_="bez nazwy",float cena_=0,
float podatek=0);
TProdukt2(TProdukt2&);
~TProdukt2();
float Czesc_brutto();
float Podaj_cene();
float Podaj_podatek();
string toString();
friend ostream& operator<<(ostream&, TProdukt2&);
};
#endif
16: 58
Modified
Insert

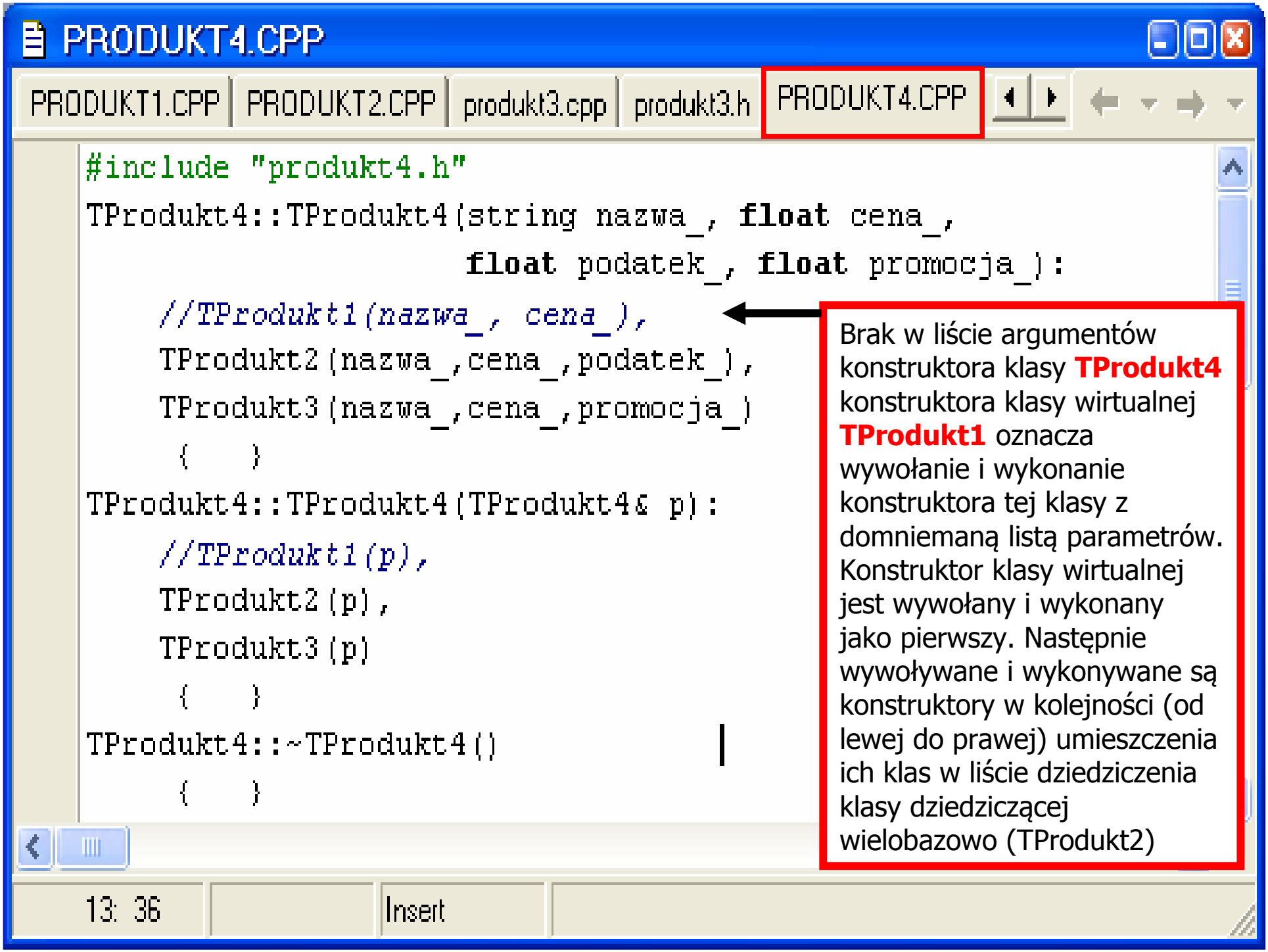

C:\Settings\dydaktyka\Programowanie_obiektowe\w9_2\produkt3.h
PRODUKT1.CPP
PRODUKT2.CPP
PRODUKT2.h
produkt3.cpp
produkt3.h
#ifndef _Produkt3
#define _Produkt3
#include "produkt1.h"
class TProdukt3: virtual public TProdukt1
{ protected:
float promocja;
public:
TProdukt3(string nazwa_="bez nazwy",float cena_=0,
float promocja=0);
TProdukt3(TProdukt3&);
~TProdukt3();
float Czesc_brutto();
float Podaj_cene();
float Podaj_promocje();
string toString();
friend ostream& operator<<(ostream& wy, TProdukt3& p);
};
#endif
2: 64
Modified
Insert

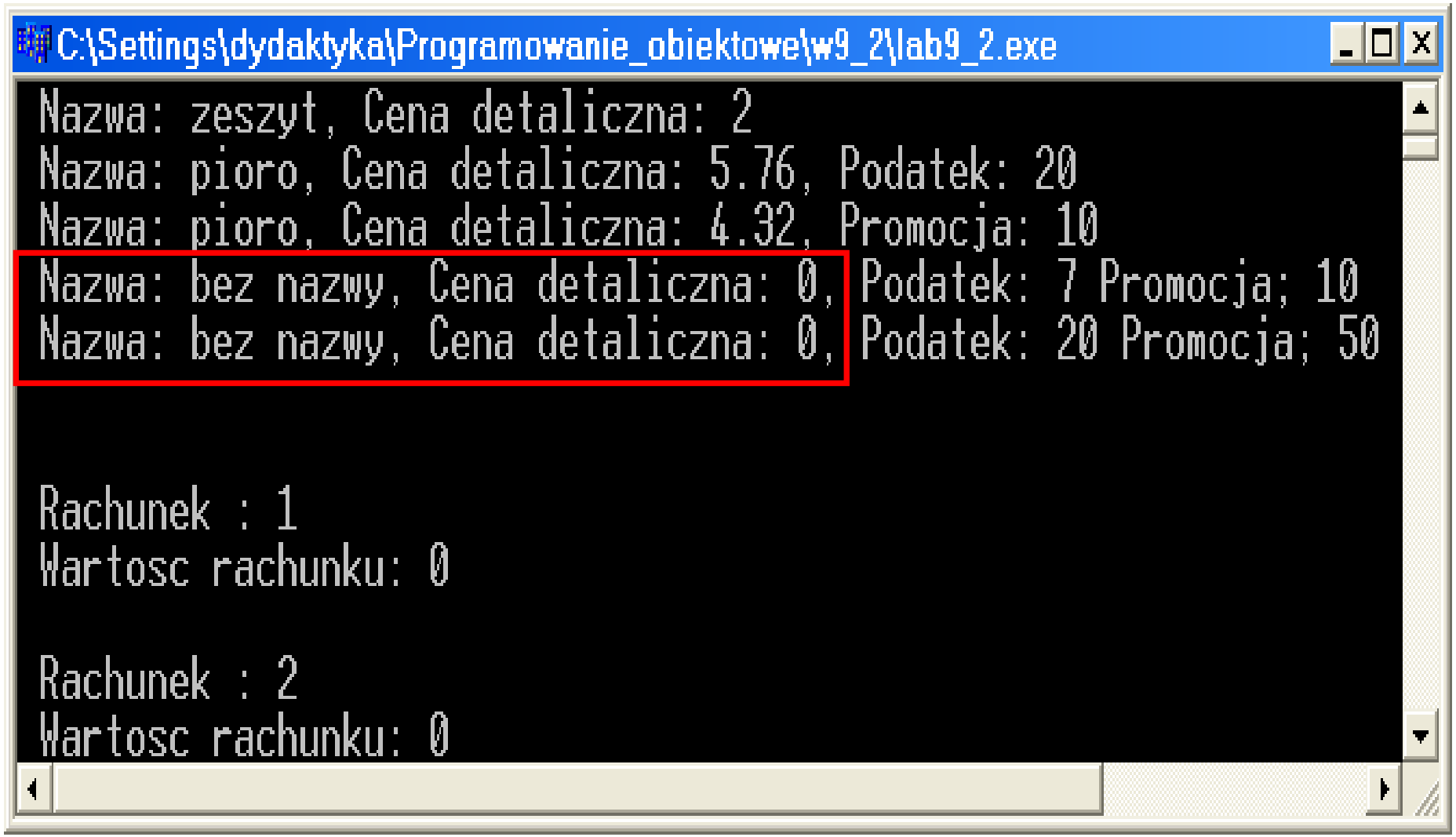
TProdukt4 dziedziczy od dwóch klas TProdukt2, TProdukt2 i pośrednio od TProdukt1

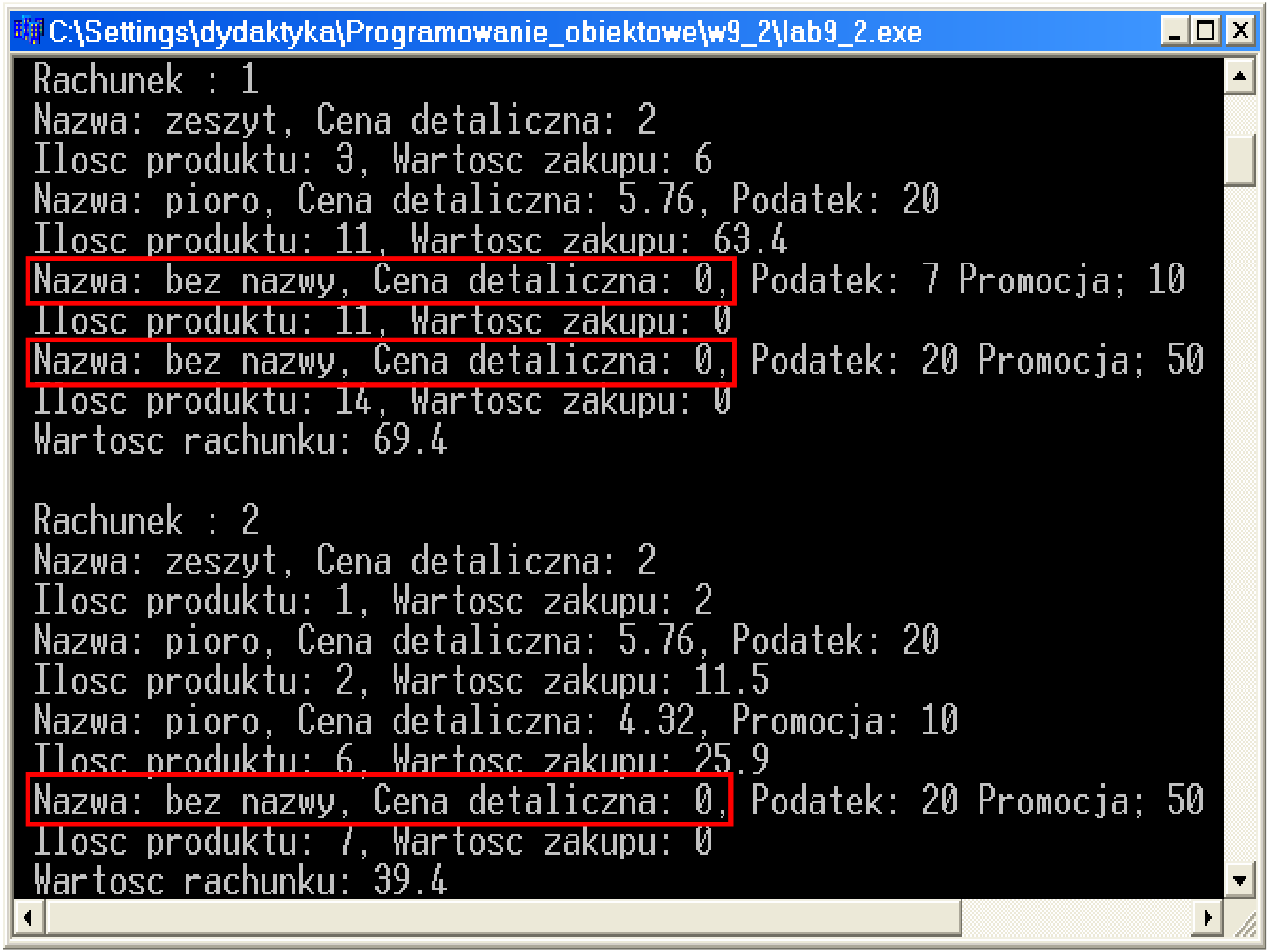
PRODUKT2.h | produkt3.cpp | produkt3.h | PRODUKT4.CPP | PRODUKT4.h

```
#ifndef _Produkt4
#define _Produkt4
#include "produkt2.h"
#include "produkt3.h"
class TProdukt4: public TProdukt3, public TProdukt2
{
    public:
        TProdukt4(string nazwa_="bez nazwy",float cena_=0,
                  float podatek_=0, float promocja_=0);
        TProdukt4(TProdukt4&);
        ~TProdukt4();
        float Podaj_cene();
        float Czesc_brutto();
        string toString();
        friend ostream& operator<<(ostream&, TProdukt4&);
};
#endif
```

17: 48 | Modified | Insert


PRODUKT4.CPP
PRODUKT1.CPP
PRODUKT2.CPP
produkt3.cpp
produkt3.h
PRODUKT4.CPP
#include "produkt4.h"
TProdukt4::TProdukt4(string nazwa_, float cena_,
float podatek_, float promocja_):
//TProdukt1(nazwa_, cena_),
TProdukt2(nazwa_,cena_,podatek_),
TProdukt3(nazwa_,cena_,promocja_)
{ }
TProdukt4::TProdukt4(TProdukt4& p):
//TProdukt1(p),
TProdukt2(p),
TProdukt3(p)
{ }
TProdukt4::~TProdukt4()
{ }
Brak w liście argumentów konstruktora klasy TProdukt4 konstruktora klasy wirtualnej TProdukt1 oznacza wywołanie i wykonanie konstruktora tej klasy z domniemaną listą parametrów. Konstruktor klasy wirtualnej jest wywołany i wykonany jako pierwszy. Następnie wywoływane i wykonywane są konstruktory w kolejności (od lewej do prawej) umieszczenia ich klas w liście dziedziczenia klasy dziedziczącej wielobazowo (TProdukt2)
13: 36
Insert

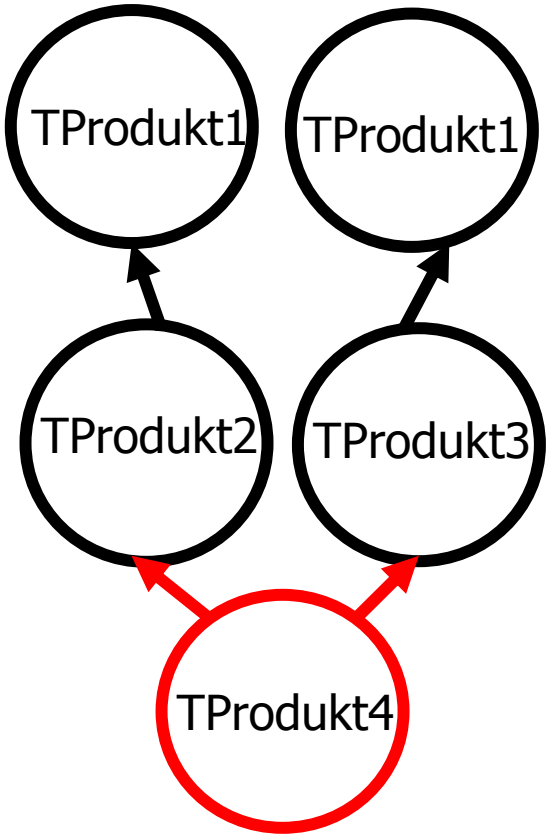

C:\Settings\dydaktyka\Programowanie_obiektowe\w9_2\lab9_2.exe
Nazwa: zeszyt, Cena detaliczna: 2
Nazwa: pioro, Cena detaliczna: 5.76, Podatek: 20
Nazwa: pioro, Cena detaliczna: 4.32, Promocja: 10
Nazwa: bez nazwy, Cena detaliczna: 0, Podatek: 7 Promocja; 10
Nazwa: bez nazwy, Cena detaliczna: 0, Podatek: 20 Promocja; 50
Rachunek : 1
Wartosc rachunku: 0
Rachunek : 2
Wartosc rachunku: 0


C:\Settings\dydaktyka\Programowanie_obiektowe\w9_2\lab9_2.exe
Rachunek : 1
Nazwa: zeszyt, Cena detaliczna: 2
Ilosc produktu: 3, Wartosc zakupu: 6
Nazwa: pioro, Cena detaliczna: 5.76, Podatek: 20
Ilosc produktu: 11, Wartosc zakupu: 63.4
Nazwa: bez nazwy, Cena detaliczna: 0, Podatek: 7 Promocja; 10
Ilosc produktu: 11, Wartosc zakupu: 0
Nazwa: bez nazwy, Cena detaliczna: 0, Podatek: 20 Promocja; 50
Ilosc produktu: 14, Wartosc zakupu: 0
Wartosc rachunku: 69.4
Rachunek : 2
Nazwa: zeszyt, Cena detaliczna: 2
Ilosc produktu: 1, Wartosc zakupu: 2
Nazwa: pioro, Cena detaliczna: 5.76, Podatek: 20
Ilosc produktu: 2, Wartosc zakupu: 11.5
Nazwa: pioro, Cena detaliczna: 4.32, Promocja: 10
Ilosc produktu: 6, Wartosc zakupu: 25.9
Nazwa: bez nazwy, Cena detaliczna: 0, Podatek: 20 Promocja; 50
Ilosc produktu: 7, Wartosc zakupu: 0
Wartosc rachunku: 39.4

Dziedziczenie wielobazowe- uzupełnienie

1. **Dziedziczenie wielobazowe z powtórzeniami z klasą wirtualną, listy konstruktorów – listy argumentów konstruktora**
2. **Dziedziczenie wielobazowe z powtórzeniami bez klasy wirtualnej, listy konstruktorów – problemy z polimorfizmem związanym z klasą TProdukt1 dziedziczoną dwukrotnie**

Dziedziczenie wielobazowe z powtórzeniami



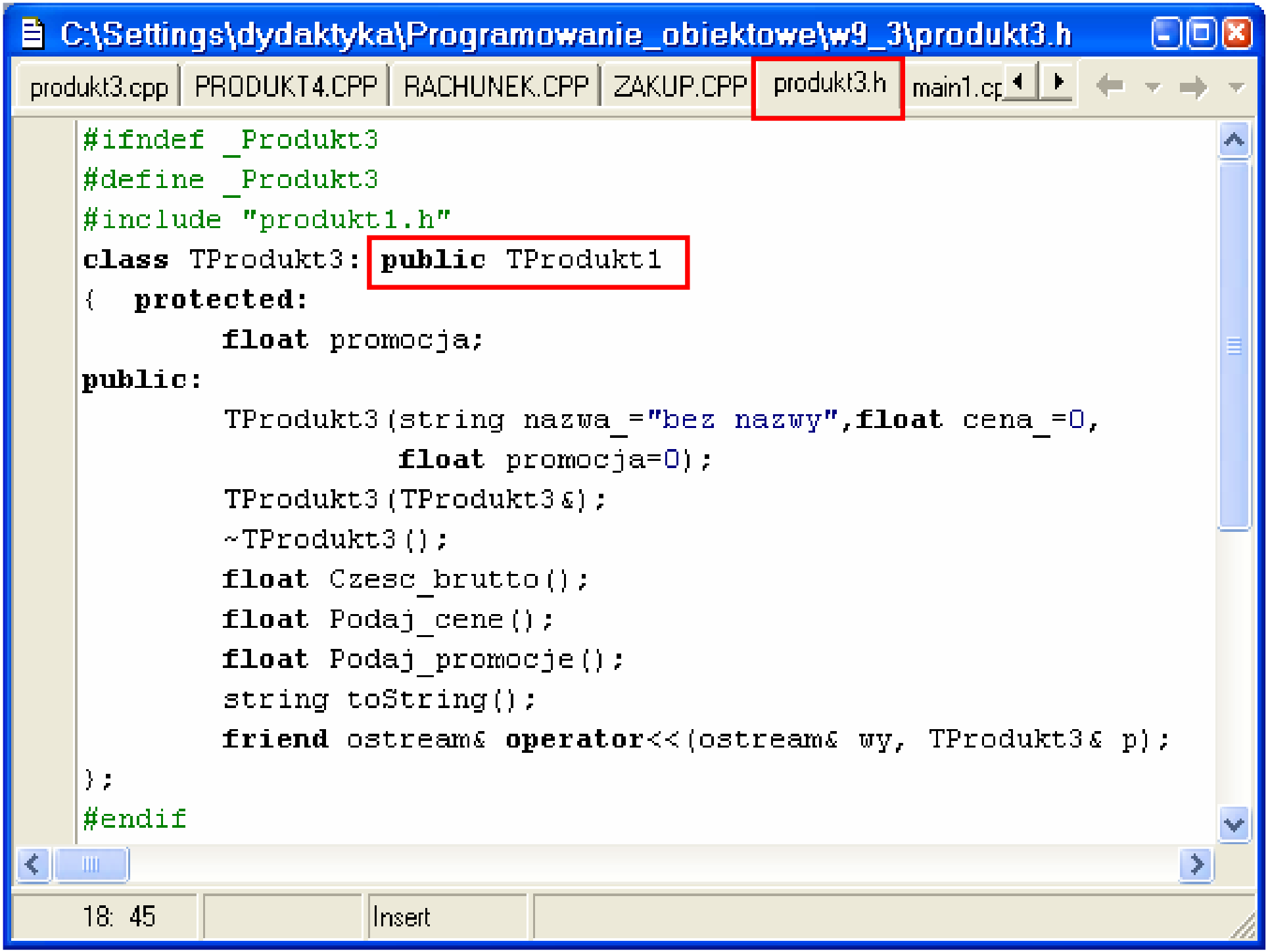
Deklaracje klas dziedziczących wielobazowo bez klasy wirtualnej

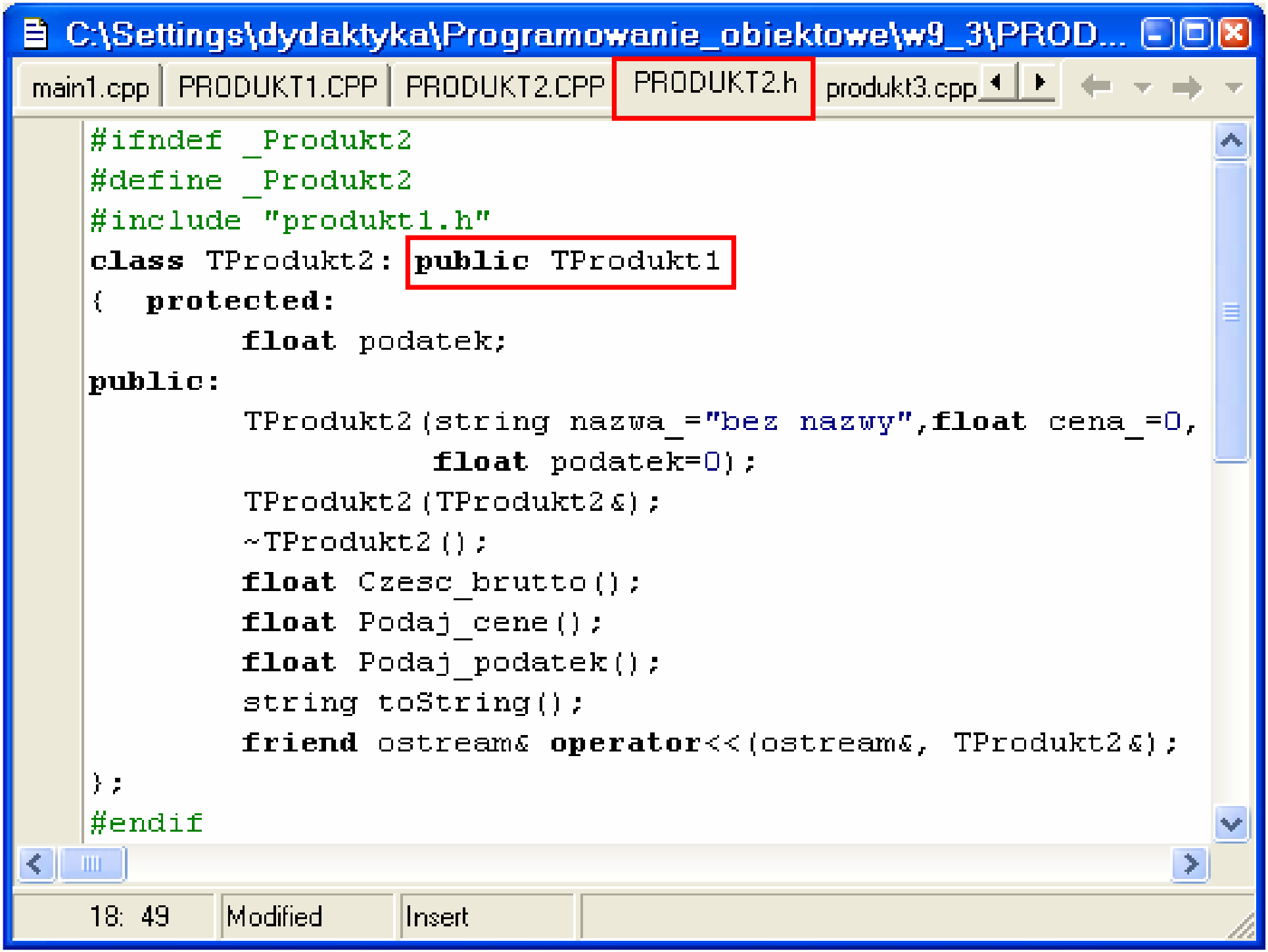
np.

```
class TProdukt1
        {.................};
class TProdukt2 : public TProdukt1
        {.................};
class TProdukt3 : public TProdukt1
        {.................};
class TProdukt4 : public TProdukt2, public TProdukt3
         {.................};
```

Konstruktory klas dziedziczących wielobazowo bez klasy wirtualnej

```
TProdukt4::TProdukt4(string nazwa_, float cena_,
                     float podatek_, float promocja_):
    TProdukt2(nazwa_,cena_,podatek_),
    TProdukt3(nazwa_,cena_,promocja_)
    {  }
TProdukt4::TProdukt4(TProdukt4& p):
    TProdukt2(p),
    TProdukt3(p)
    {  }
```


C:\Settings\dydaktyka\Programowanie_obiektowe\w9_3\produkt3.h
produkt3.cpp
PRODUKT4.CPP
RACHUNEK.CPP
ZAKUP.CPP
produkt3.h
main1.cp
#ifndef _Produkt3
#define _Produkt3
#include "produkt1.h"
class TProdukt3: public TProdukt1
{ protected:
float promocja;
public:
TProdukt3(string nazwa_="bez nazwy",float cena_=0,
float promocja=0);
TProdukt3(TProdukt3&);
~TProdukt3();
float Czesc_brutto();
float Podaj_cene();
float Podaj_promocje();
string toString();
friend ostream& operator<<(ostream& wy, TProdukt3& p);
};
#endif
18: 45
Insert

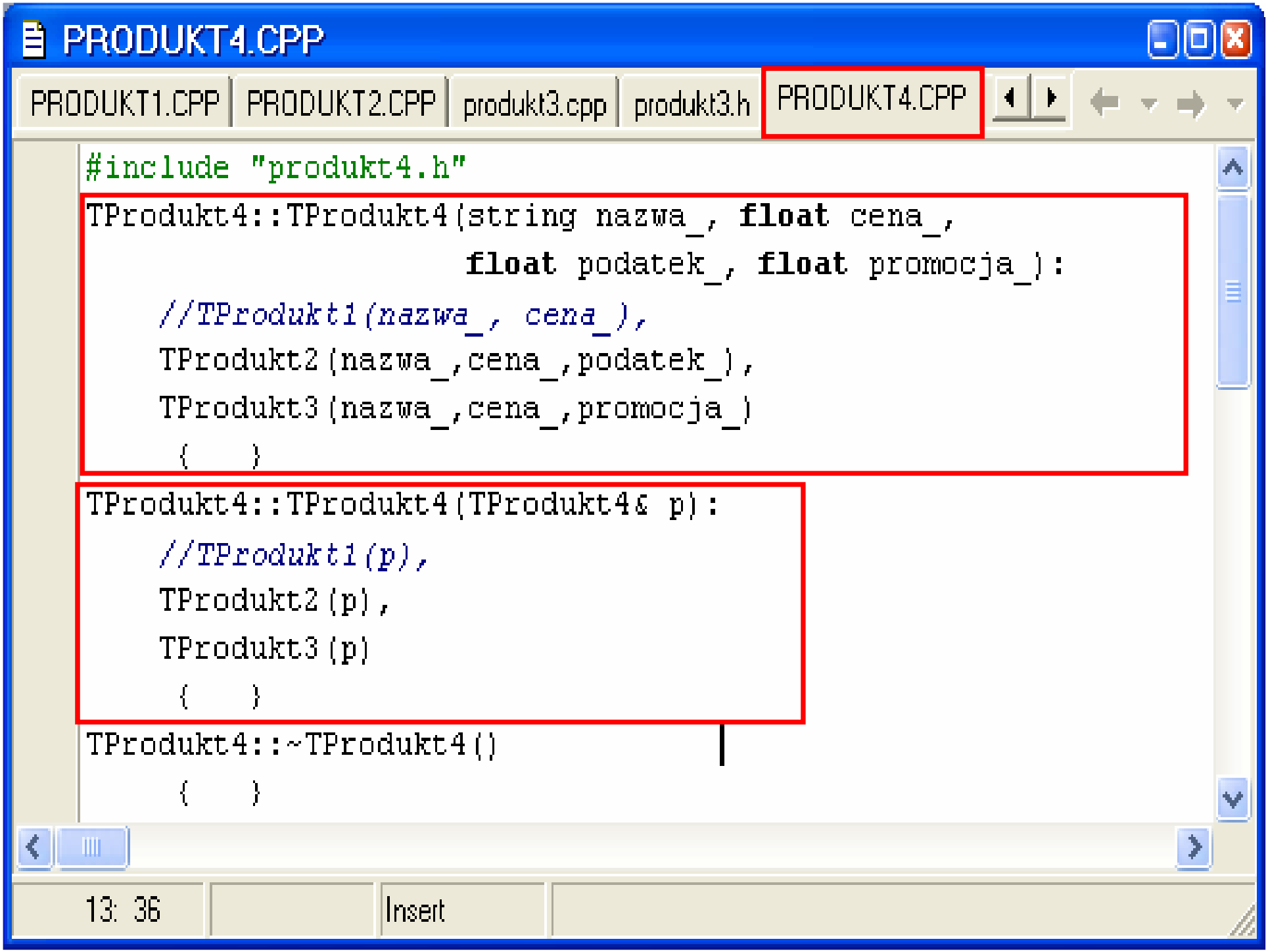

C:\Settings\dydaktyka\Programowanie_obiektowe\w9_3\PROD...
main1.cpp
PRODUKT1.CPP
PRODUKT2.CPP
PRODUKT2.h
produkt3.cpp
#ifndef _Produkt2
#define _Produkt2
#include "produkt1.h"
class TProdukt2: public TProdukt1
{ protected:
float podatek;
public:
TProdukt2(string nazwa_="bez nazwy",float cena_=0,
float podatek=0);
TProdukt2(TProdukt2&);
~TProdukt2();
float Czesc_brutto();
float Podaj_cene();
float Podaj_podatek();
string toString();
friend ostream& operator<<(ostream&, TProdukt2&);
};
#endif
18: 49
Modified
Insert

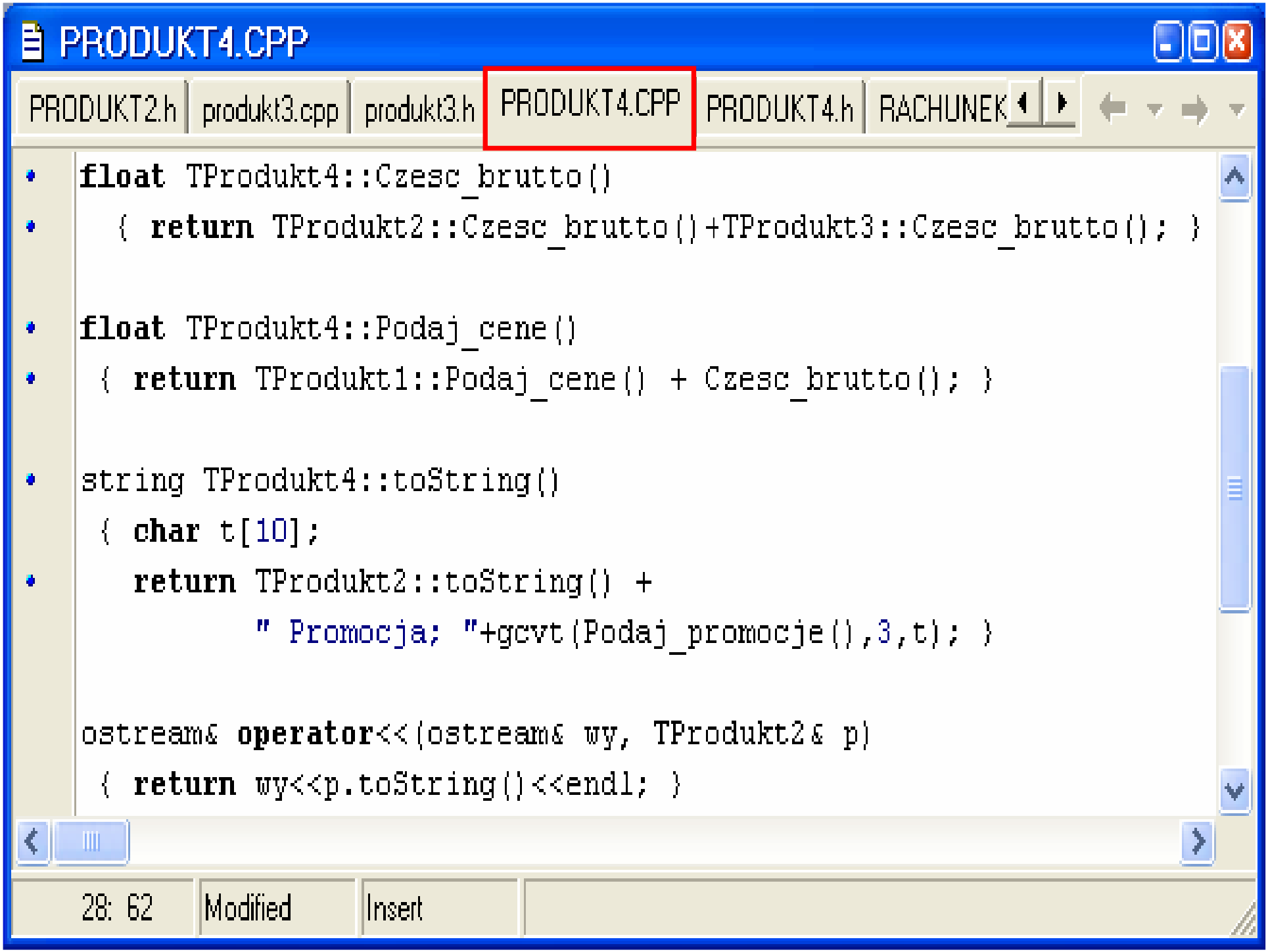
TProdukt4 dziedziczy od dwóch klas TProdukt2, TProdukt2 i podwójnie od TProdukt1

PRODUKT2.h | produkt3.cpp | produkt3.h | PRODUKT4.CPP | PRODUKT4.h

```
#ifndef _Produkt4
#define _Produkt4
#include "produkt2.h"
#include "produkt3.h"
class TProdukt4: public TProdukt3, public TProdukt2
{
    public:
        TProdukt4(string nazwa_="bez nazwy",float cena_=0,
                  float podatek_=0, float promocja_=0);
        TProdukt4(TProdukt4&);
        ~TProdukt4();
        float Podaj_cene();
        float Czesc_brutto();
        string toString();
        friend ostream& operator<<(ostream&, TProdukt4&);
};
#endif
```

17: 48 | Modified | Insert


PRODUKT4.CPP
PRODUKT1.CPP
PRODUKT2.CPP
produkt3.cpp
produkt3.h
PRODUKT4.CPP
#include "produkt4.h"
TProdukt4::TProdukt4(string nazwa_, float cena_,
float podatek_, float promocja_):
//TProdukt1(nazwa_, cena_),
TProdukt2(nazwa_,cena_,podatek_),
TProdukt3(nazwa_,cena_,promocja_)
{ }
TProdukt4::TProdukt4(TProdukt4& p):
//TProdukt1(p),
TProdukt2(p),
TProdukt3(p)
{ }
TProdukt4::~TProdukt4()
{ }
13: 36
Insert

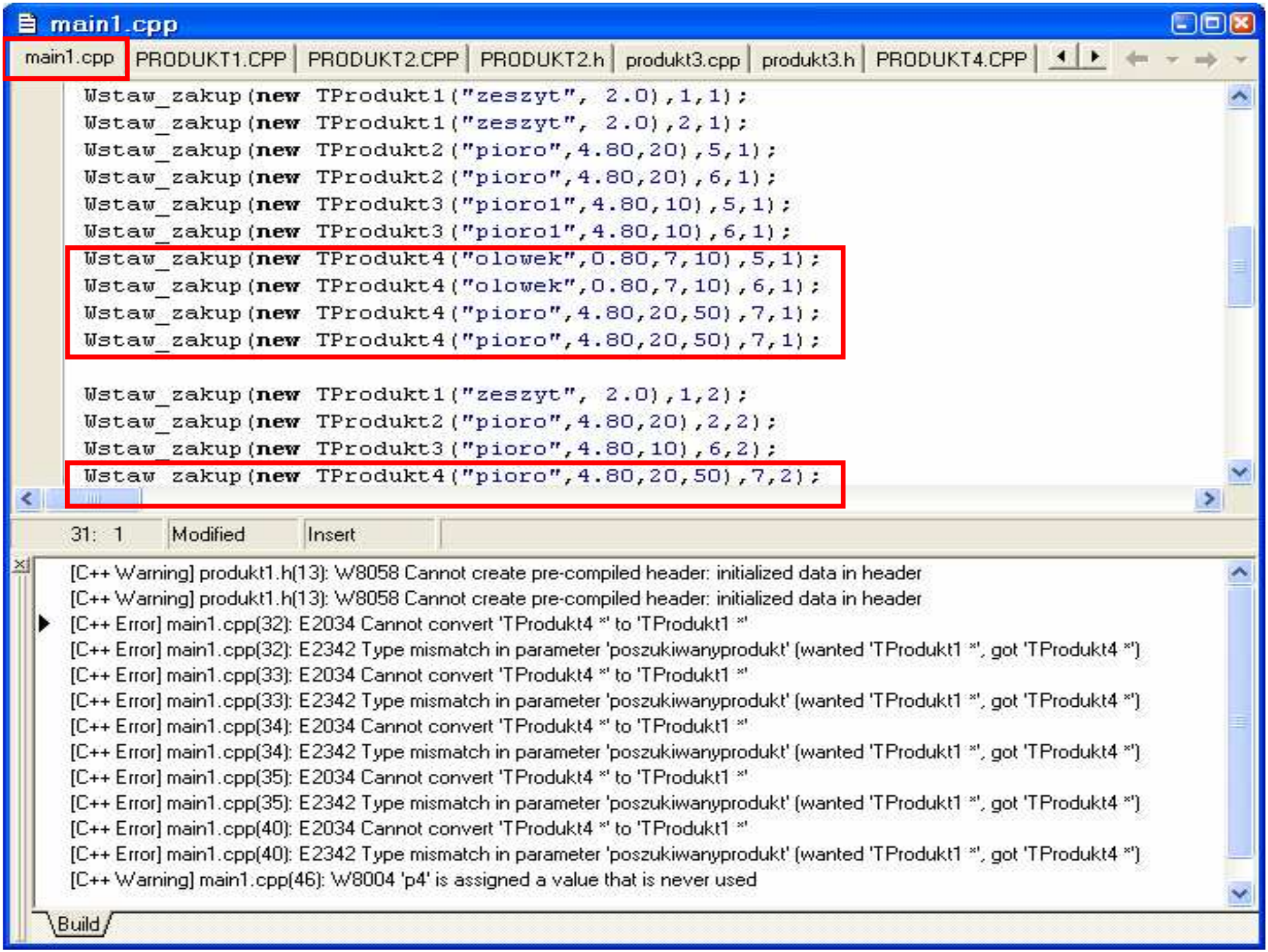

PRODUKT4.CPP
PRODUKT2.h
produkt3.cpp
produkt3.h
PRODUKT4.CPP
PRODUKT4.h
RACHUNEK
float TProdukt4::Czesc_brutto()
{ return TProdukt2::Czesc_brutto()+TProdukt3::Czesc_brutto(); }
float TProdukt4::Podaj_cene()
{ return TProdukt1::Podaj_cene() + Czesc_brutto(); }
string TProdukt4::toString()
{ char t[10];
return TProdukt2::toString() +
" Promocja; "+gcvt(Podaj_promocje(),3,t); }
ostream& operator<<(ostream& wy, TProdukt2& p)
{ return wy<<p.toString()<<endl; }
28: 62
Modified
Insert

Przykład 1

main1.cpp | PRODUKT1.CPP | PRODUKT2.CPP | produkt3.cpp | PRODUKT4.CPP

```
#include "Rachunek.h"
 TKol2<TProdukt1> produkty;
 TKol2<TRachunek> rachunki;
 void Wstaw_zakup(TProdukt1* poszukiwanyprodukt,
                  int ilosc, int numer);
void main()
{
 TProdukt1* p1;
 TProdukt2* p2;
 TProdukt3* p3;
 TProdukt4* p4;
 p1 = new TProdukt1("zeszyt", 2.0);          produkty.Wstaw(p1);
 p2 = new TProdukt2("pioro",4.80,20);        produkty.Wstaw(p2);
 p2 = new TProdukt2("pioro",4.80,20);        produkty.Wstaw(p2);
 p3 = new TProdukt3("pioro",4.80,10);        produkty.Wstaw(p3);
 p3 = new TProdukt3("pioro",4.80,10);        produkty.Wstaw(p3);
 p4 = new TProdukt4("olowek",0.80,7,10);     produkty.Wstaw(p4);
 p4 = new TProdukt4("pioro",4.80,20,50);     produkty.Wstaw(p4);
 p4 = new TProdukt4("pioro",4.80,20,50);     produkty.Wstaw(p4);
 cout<<produkty<<endl;          //cout<<produkty.toString()<<endl;

 rachunki.Wstaw(new TRachunek(1));
 rachunki.Wstaw(new TRachunek(2));
 cout<<rachunki<<endl;   //cout<<rachunki.toString();
```

1: 1 | Insert


main1.cpp
main1.cpp
PRODUKT1.CPP
PRODUKT2.CPP
PRODUKT2.h
produkt3.cpp
produkt3.h
PRODUKT4.CPP
Wstaw_zakup(new TProdukt1("zeszyt", 2.0),1,1);
Wstaw_zakup(new TProdukt1("zeszyt", 2.0),2,1);
Wstaw_zakup(new TProdukt2("pioro",4.80,20),5,1);
Wstaw_zakup(new TProdukt2("pioro",4.80,20),6,1);
Wstaw_zakup(new TProdukt3("pioro1",4.80,10),5,1);
Wstaw_zakup(new TProdukt3("pioro1",4.80,10),6,1);
Wstaw_zakup(new TProdukt4("olowek",0.80,7,10),5,1);
Wstaw_zakup(new TProdukt4("olowek",0.80,7,10),6,1);
Wstaw_zakup(new TProdukt4("pioro",4.80,20,50),7,1);
Wstaw_zakup(new TProdukt4("pioro",4.80,20,50),7,1);
Wstaw_zakup(new TProdukt1("zeszyt", 2.0),1,2);
Wstaw_zakup(new TProdukt2("pioro",4.80,20),2,2);
Wstaw_zakup(new TProdukt3("pioro",4.80,10),6,2);
Wstaw_zakup(new TProdukt4("pioro",4.80,20,50),7,2);
31: 1
Modified
Insert
[C++ Warning] produkt1.h(13): W8058 Cannot create pre-compiled header: initialized data in header
[C++ Warning] produkt1.h(13): W8058 Cannot create pre-compiled header: initialized data in header
[C++ Error] main1.cpp(32): E2034 Cannot convert 'TProdukt4 *' to 'TProdukt1 *'
[C++ Error] main1.cpp(32): E2342 Type mismatch in parameter 'poszukiwanyprodukt' (wanted 'TProdukt1 *', got 'TProdukt4 *')
[C++ Error] main1.cpp(33): E2034 Cannot convert 'TProdukt4 *' to 'TProdukt1 *'
[C++ Error] main1.cpp(33): E2342 Type mismatch in parameter 'poszukiwanyprodukt' (wanted 'TProdukt1 *', got 'TProdukt4 *')
[C++ Error] main1.cpp(34): E2034 Cannot convert 'TProdukt4 *' to 'TProdukt1 *'
[C++ Error] main1.cpp(34): E2342 Type mismatch in parameter 'poszukiwanyprodukt' (wanted 'TProdukt1 *', got 'TProdukt4 *')
[C++ Error] main1.cpp(35): E2034 Cannot convert 'TProdukt4 *' to 'TProdukt1 *'
[C++ Error] main1.cpp(35): E2342 Type mismatch in parameter 'poszukiwanyprodukt' (wanted 'TProdukt1 *', got 'TProdukt4 *')
[C++ Error] main1.cpp(40): E2034 Cannot convert 'TProdukt4 *' to 'TProdukt1 *'
[C++ Error] main1.cpp(40): E2342 Type mismatch in parameter 'poszukiwanyprodukt' (wanted 'TProdukt1 *', got 'TProdukt4 *')
[C++ Warning] main1.cpp(46): W8004 'p4' is assigned a value that is never used
Build

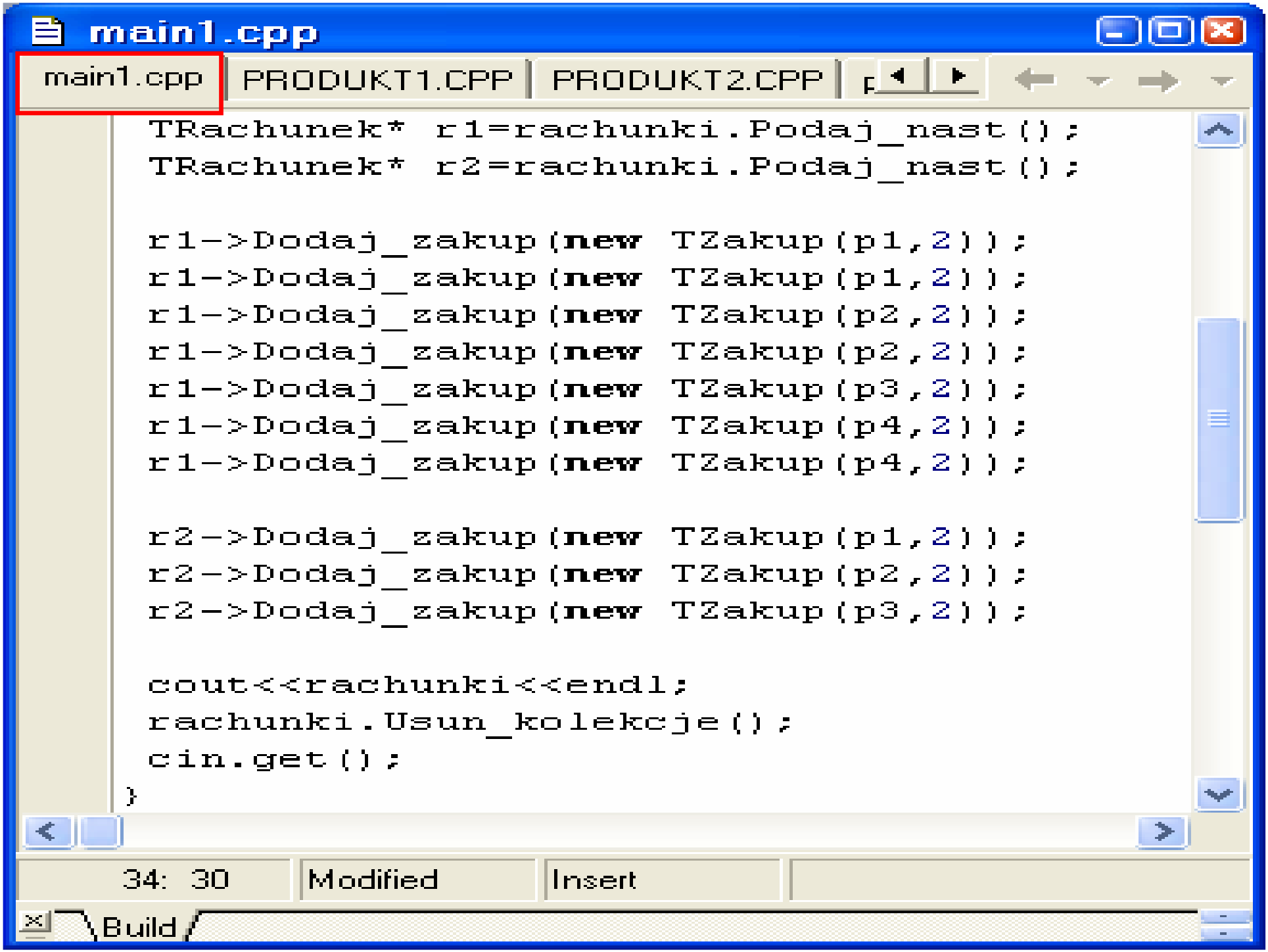
Przykład 2

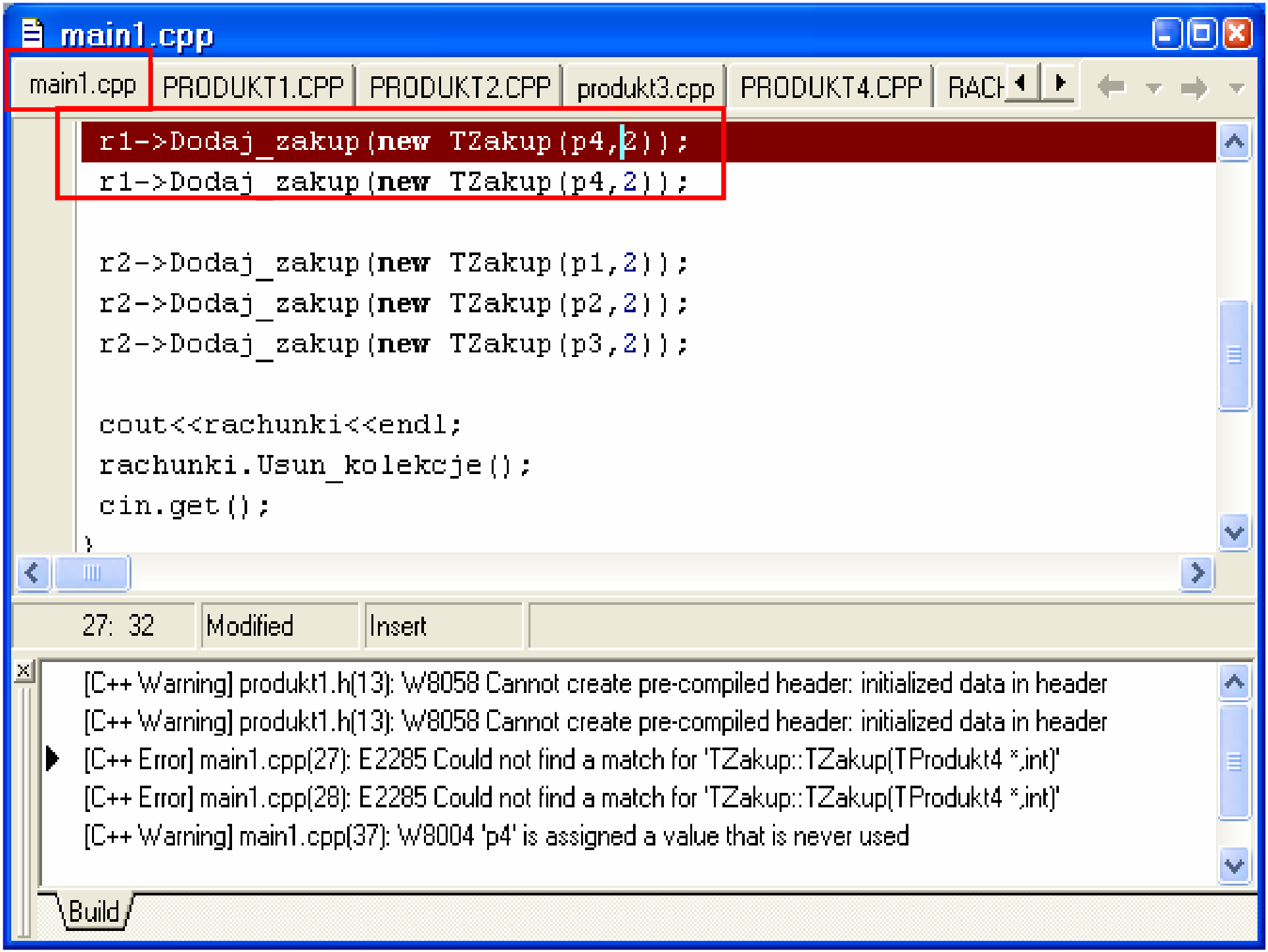
main1.cpp | PRODUKT1.CPP | PRODUKT2.CPP | produkt

```
#include "Rachunek.h"
 TKol2<TRachunek> rachunki;

void main()
{
 TProdukt1* p1;
 TProdukt2* p2;
 TProdukt3* p3;
 TProdukt4* p4;
 p1 = new TProdukt1("zeszyt", 2.0);
 p2 = new TProdukt2("pioro",4.80,20);
 p3 = new TProdukt3("pioro",4.80,10);
 p4 = new TProdukt4("olowek",0.80,7,10);

 rachunki.Wstaw(new TRachunek(1));
 rachunki.Wstaw(new TRachunek(2));
 cout<<rachunki<<endl;
```

17: 26 | Modified | Insert

Build

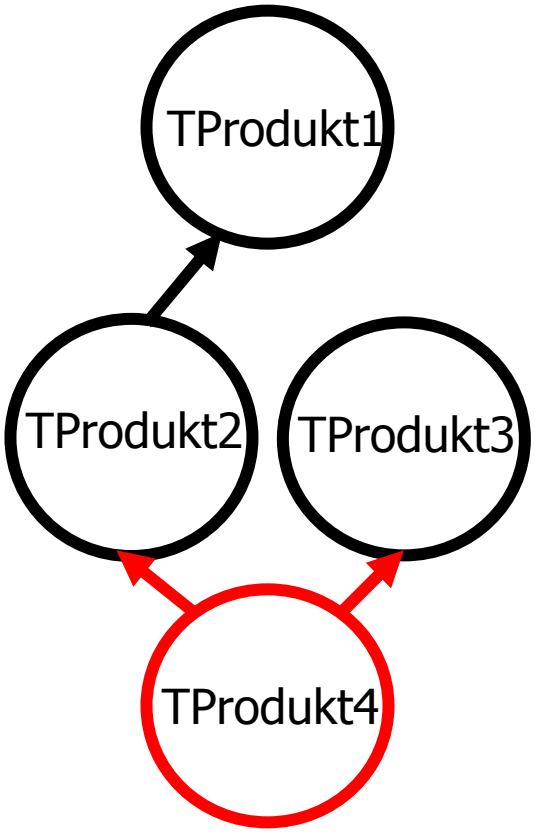

main1.cpp
main1.cpp
PRODUKT1.CPP
PRODUKT2.CPP
TRachunek* r1=rachunki.Podaj_nast();
TRachunek* r2=rachunki.Podaj_nast();
r1->Dodaj_zakup(new TZakup(p1,2));
r1->Dodaj_zakup(new TZakup(p1,2));
r1->Dodaj_zakup(new TZakup(p2,2));
r1->Dodaj_zakup(new TZakup(p2,2));
r1->Dodaj_zakup(new TZakup(p3,2));
r1->Dodaj_zakup(new TZakup(p4,2));
r1->Dodaj_zakup(new TZakup(p4,2));
r2->Dodaj_zakup(new TZakup(p1,2));
r2->Dodaj_zakup(new TZakup(p2,2));
r2->Dodaj_zakup(new TZakup(p3,2));
cout<<rachunki<<endl;
rachunki.Usun_kolekcje();
cin.get();
}
34: 30
Modified
Insert
Build

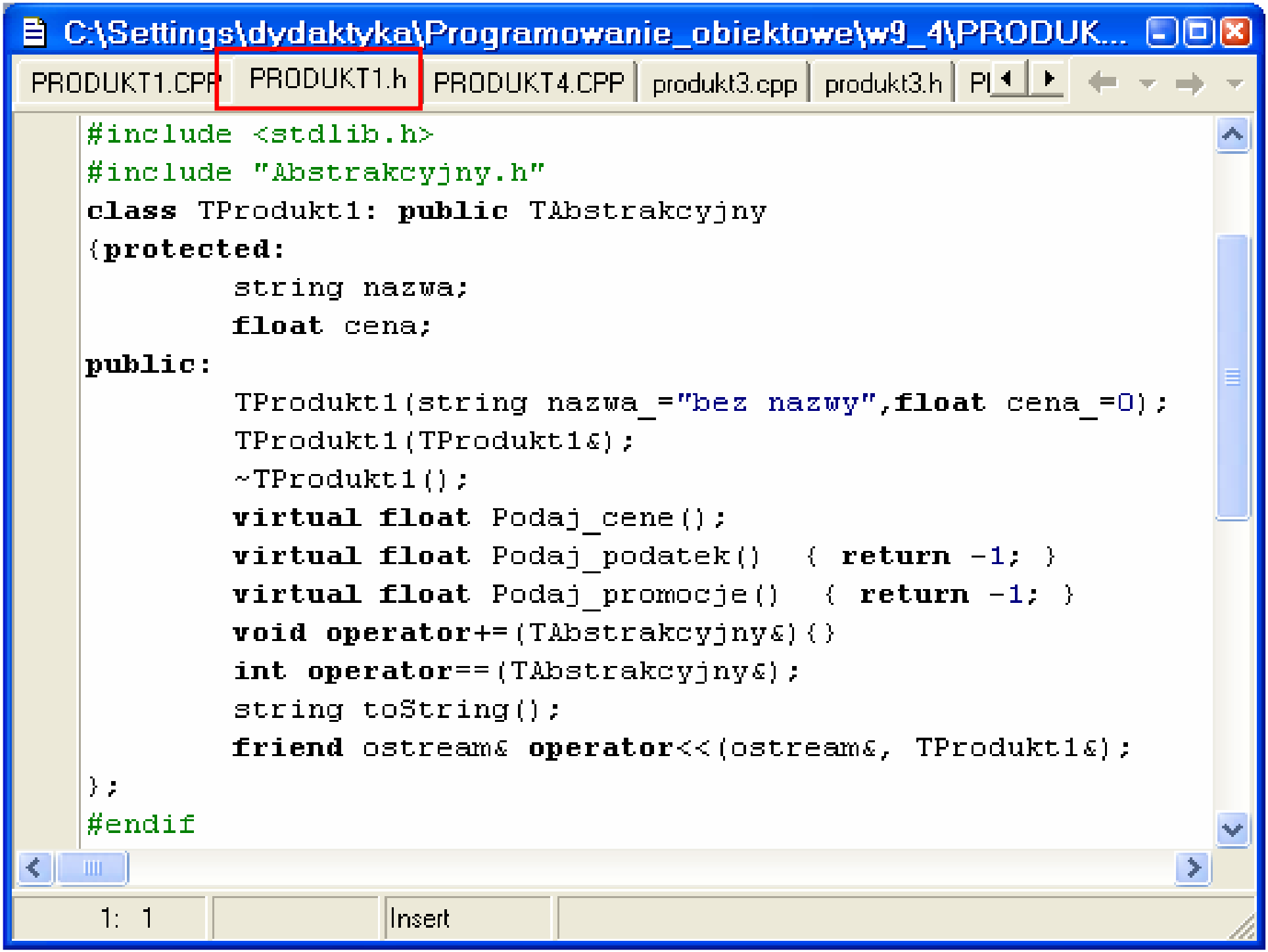

main1.cpp
main1.cpp
PRODUKT1.CPP
PRODUKT2.CPP
produkt3.cpp
PRODUKT4.CPP
RACH
r1->Dodaj_zakup(new TZakup(p4,2));
r1->Dodaj_zakup(new TZakup(p4,2));
r2->Dodaj_zakup(new TZakup(p1,2));
r2->Dodaj_zakup(new TZakup(p2,2));
r2->Dodaj_zakup(new TZakup(p3,2));
cout<<rachunki<<endl;
rachunki.Usun_kolekcje();
cin.get();
}
27: 32
Modified
Insert
[C++ Warning] produkt1.h(13): W8058 Cannot create pre-compiled header: initialized data in header
[C++ Warning] produkt1.h(13): W8058 Cannot create pre-compiled header: initialized data in header
[C++ Error] main1.cpp(27): E2285 Could not find a match for 'TZakup::TZakup(TProdukt4 *,int)'
[C++ Error] main1.cpp(28): E2285 Could not find a match for 'TZakup::TZakup(TProdukt4 *,int)'
[C++ Warning] main1.cpp(37): W8004 'p4' is assigned a value that is never used
Build

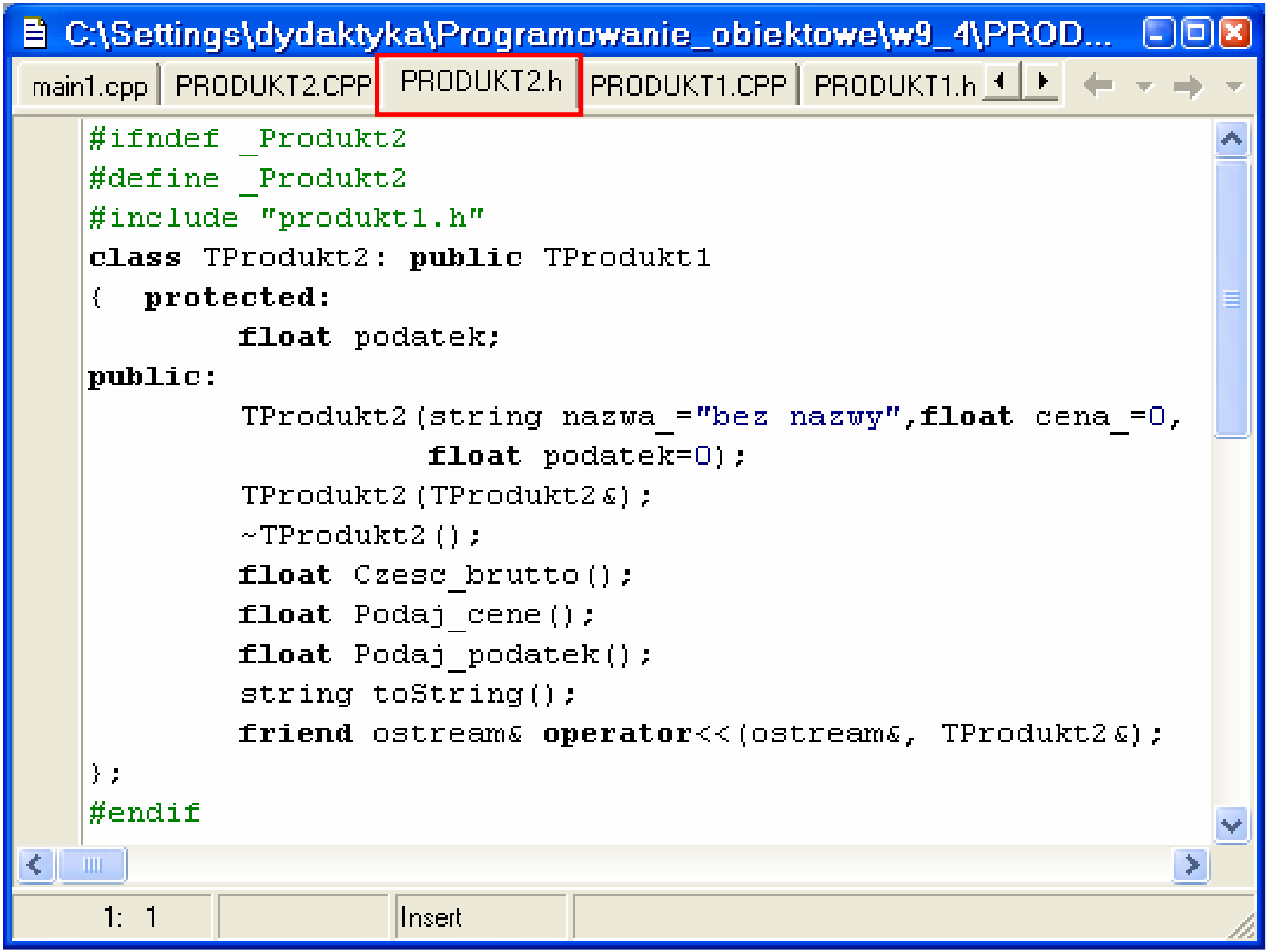
Dziedziczenie wielobazowe- uzupełnienie

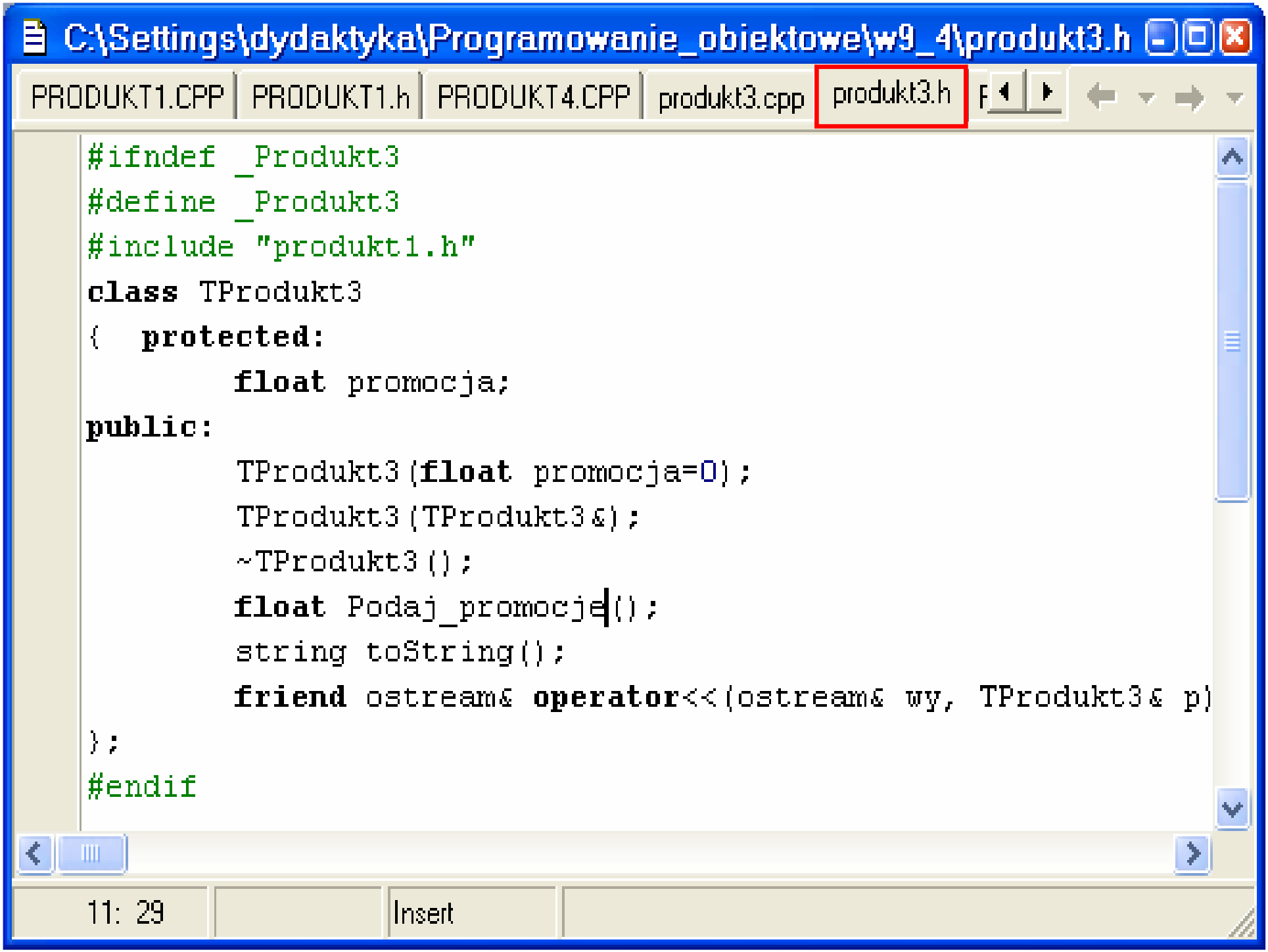
1. **Dziedziczenie wielobazowe z powtórzeniami z klasą wirtualną, listy konstruktorów – listy argumentów konstruktorów**
2. **Dziedziczenie wielobazowe z powtórzeniami bez klasy wirtualnej, listy konstruktorów – problemy z polimorfizmem związanym z klasą TProdukt1 dziedziczoną dwukrotnie**
3. **Dziedziczenie wielobazowe bez powtórzeń**

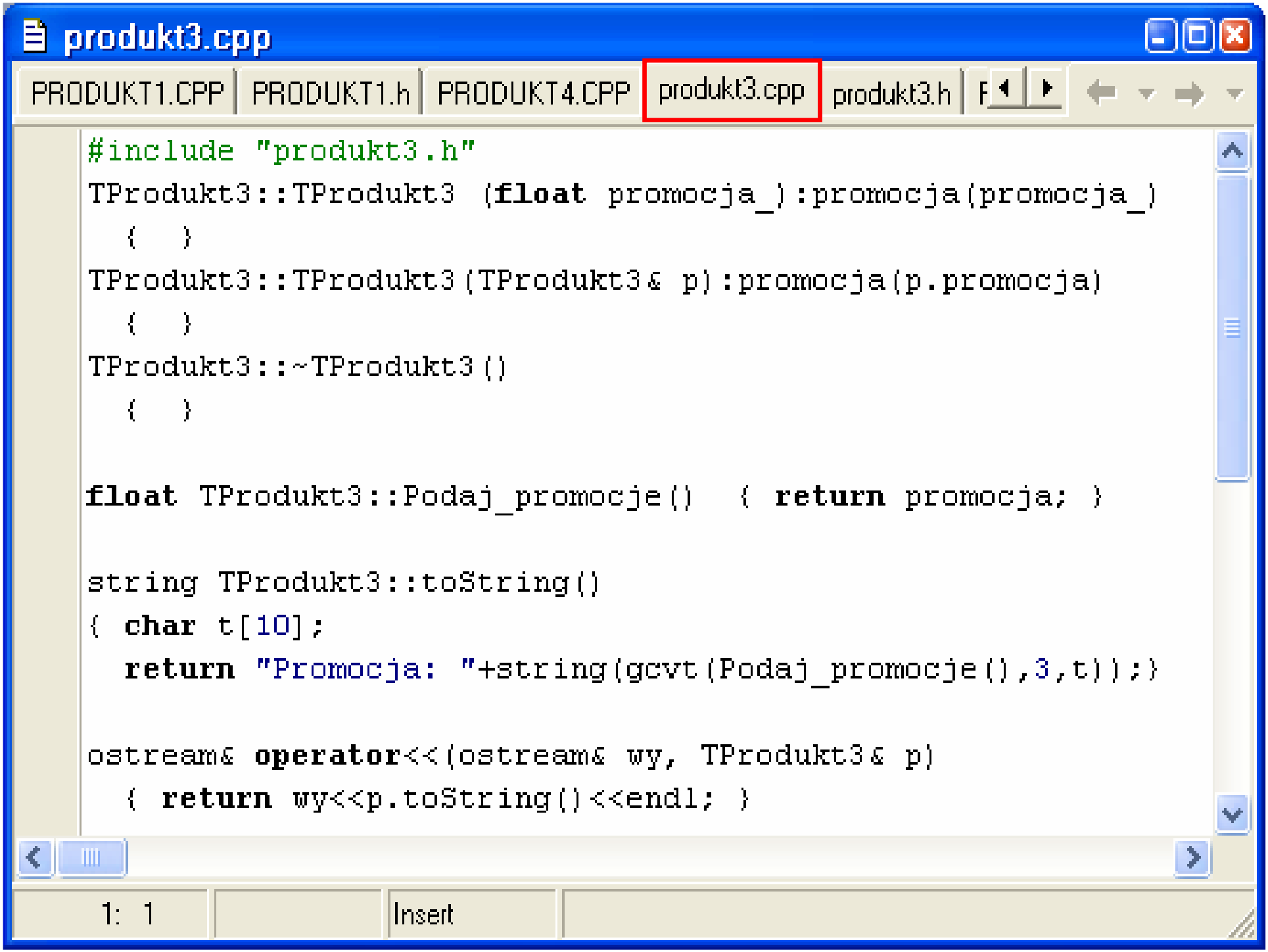
Dziedziczenie wielobazowe bez powtórzeń – klasa TProdukt1 bez zmian, natomiast TProdukt2 w liście dziedziczenia ma klasę TProdukt1, która nie jest już klasą wirtualną. Klasa TProdukt3 dostarcza jedynie promocję i nie jest stosowana do tworzenia samodzielnych obiektów

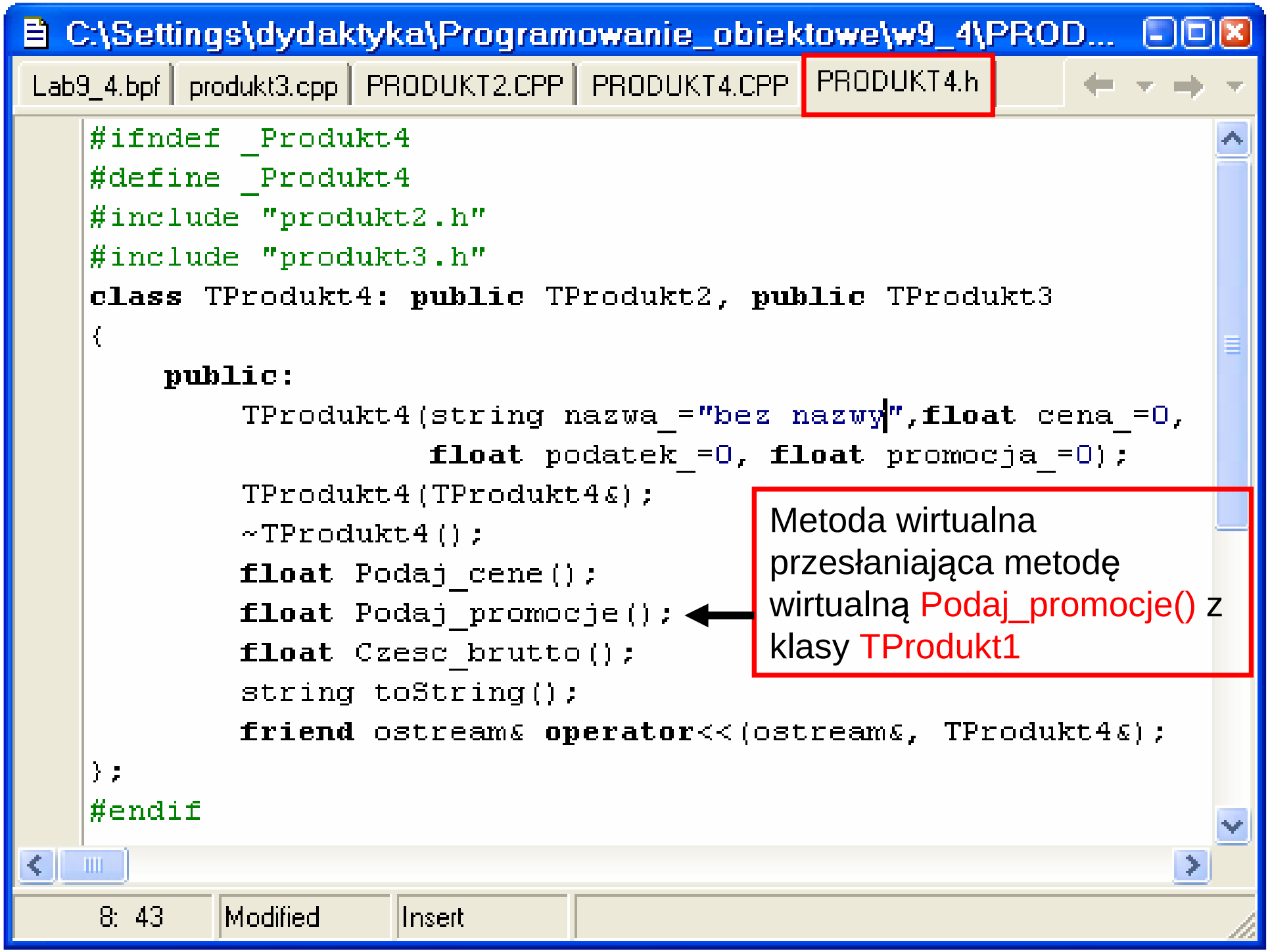


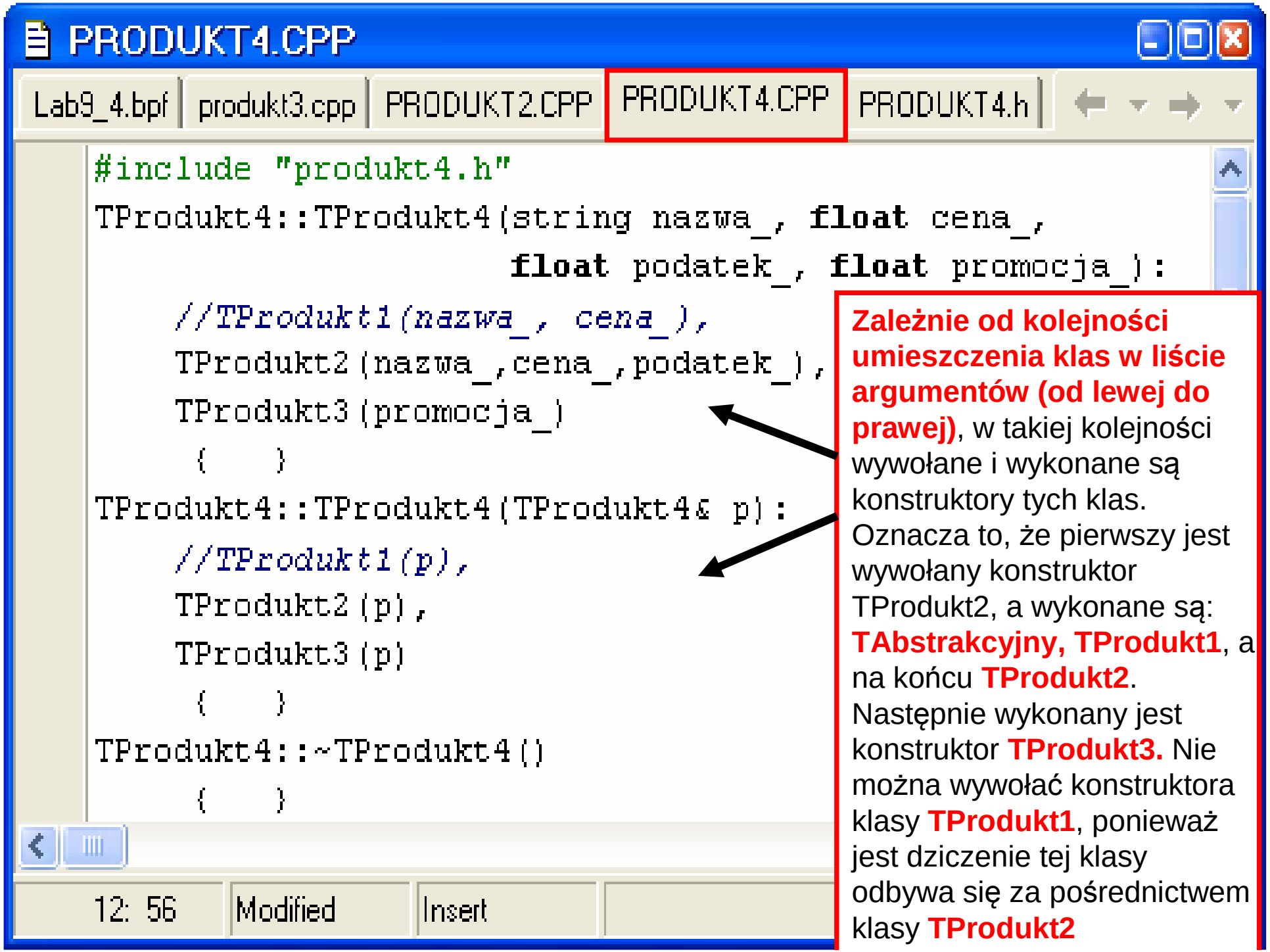

C:\Settings\dydaktyka\Programowanie_obiektowe\w9_4\PRODUK...
PRODUKT1.CPP
PRODUKT1.h
PRODUKT4.CPP
produkt3.cpp
produkt3.h
#include <stdlib.h>
#include "Abstrakcyjny.h"
class TProdukt1: public TAbstrakcyjny
{protected:
string nazwa;
float cena;
public:
TProdukt1(string nazwa_="bez nazwy",float cena_=0);
TProdukt1(TProdukt1&);
~TProdukt1();
virtual float Podaj_cene();
virtual float Podaj_podatek() { return -1; }
virtual float Podaj_promocje() { return -1; }
void operator+=(TAbstrakcyjny&){}
int operator==(TAbstrakcyjny&);
string toString();
friend ostream& operator<<(ostream&, TProdukt1&);
};
#endif
1: 1
Insert

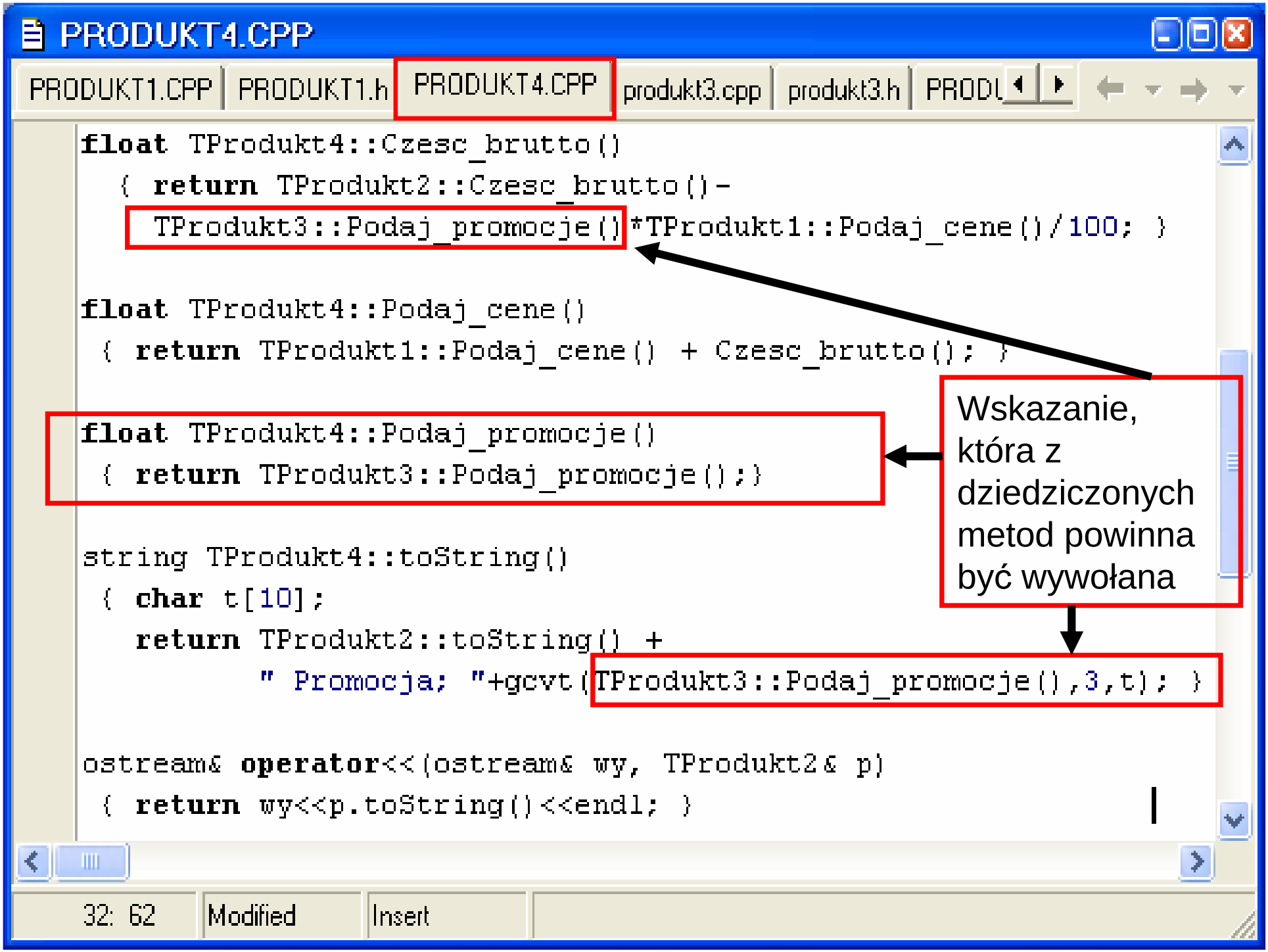

C:\Settings\dydaktyka\Programowanie_obiektowe\w9_4\PROD...
main1.cpp
PRODUKT2.CPP
PRODUKT2.h
PRODUKT1.CPP
PRODUKT1.h
#ifndef _Produkt2
#define _Produkt2
#include "produkt1.h"
class TProdukt2: public TProdukt1
{ protected:
float podatek;
public:
TProdukt2(string nazwa_="bez nazwy",float cena_=0,
float podatek=0);
TProdukt2(TProdukt2&);
~TProdukt2();
float Czesc_brutto();
float Podaj_cene();
float Podaj_podatek();
string toString();
friend ostream& operator<<(ostream&, TProdukt2&);
};
#endif
1: 1
Insert

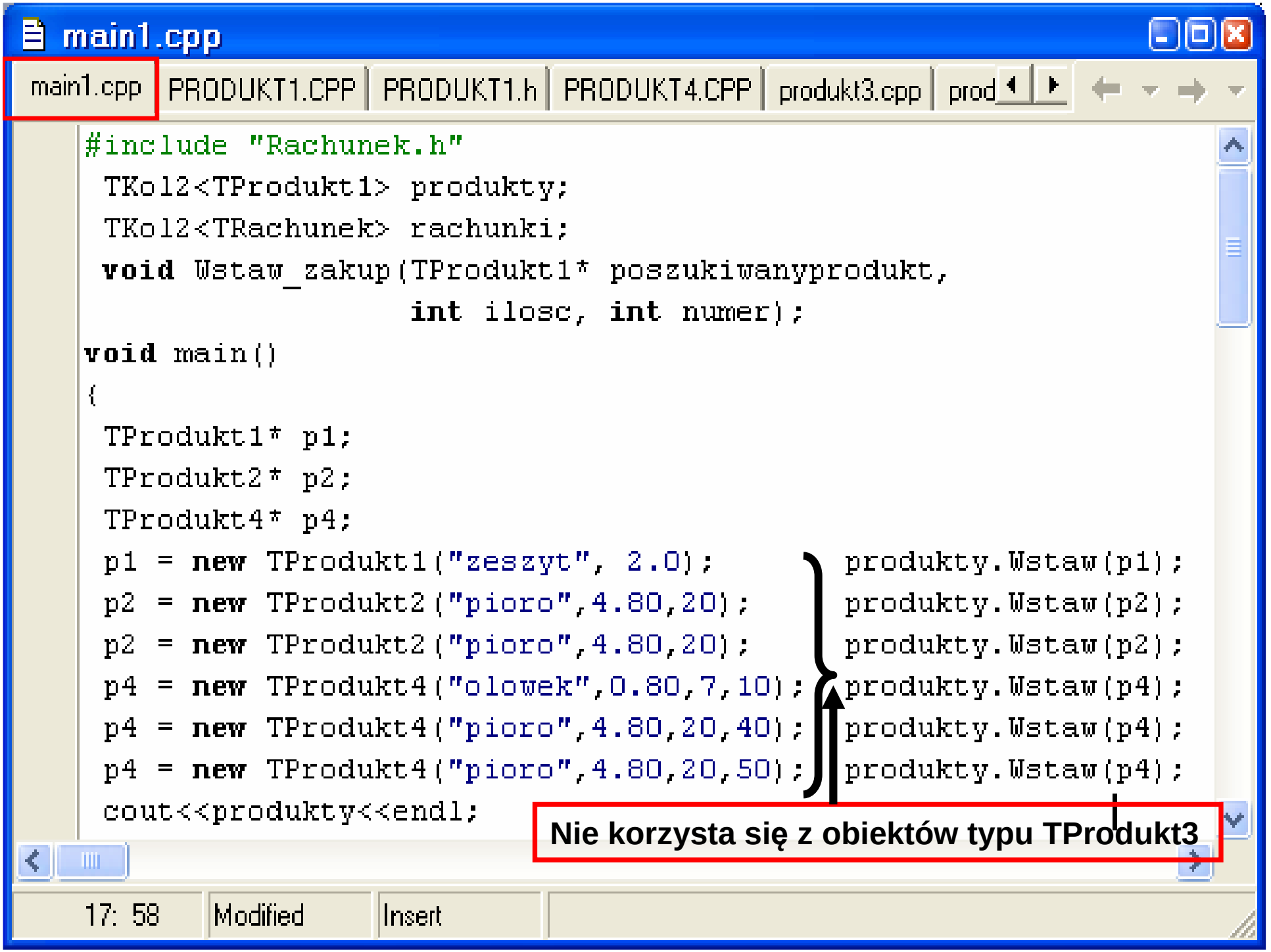

C:\Settings\dydaktyka\Programowanie_obiektowe\w9_4\produkt3.h
PRODUKT1.CPP
PRODUKT1.h
PRODUKT4.CPP
produkt3.cpp
produkt3.h
#ifndef _Produkt3
#define _Produkt3
#include "produkt1.h"
class TProdukt3
{ protected:
float promocja;
public:
TProdukt3(float promocja=0);
TProdukt3(TProdukt3&);
~TProdukt3();
float Podaj_promocje();
string toString();
friend ostream& operator<<(ostream& wy, TProdukt3& p)
};
#endif
11: 29
Insert

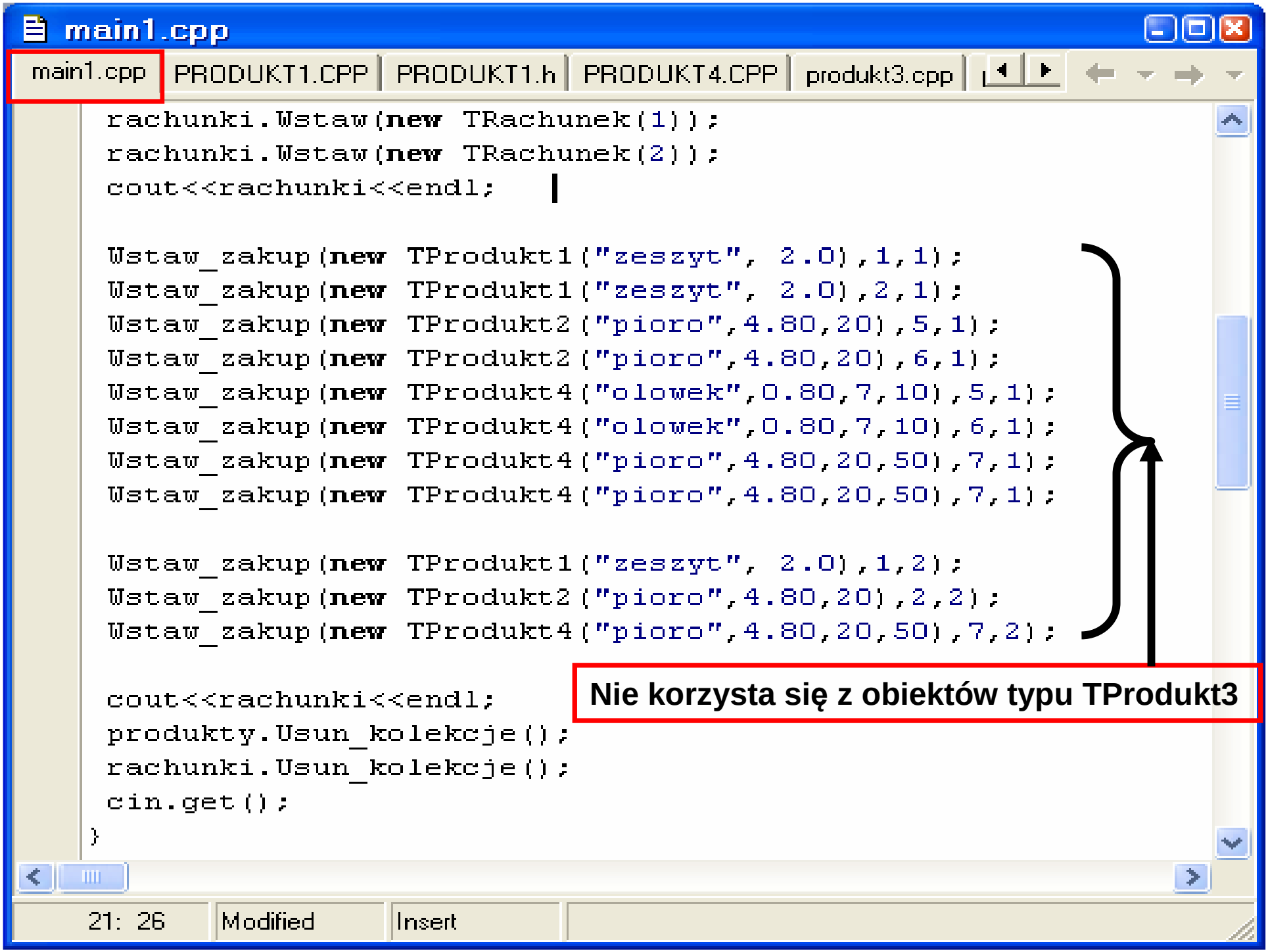

produkt3.cpp
PRODUKT1.CPP
PRODUKT1.h
PRODUKT4.CPP
produkt3.cpp
produkt3.h
#include "produkt3.h"
TProdukt3::TProdukt3 (float promocja_):promocja(promocja_)
{ }
TProdukt3::TProdukt3(TProdukt3& p):promocja(p.promocja)
{ }
TProdukt3::~TProdukt3()
{ }
float TProdukt3::Podaj_promocje() { return promocja; }
string TProdukt3::toString()
{ char t[10];
return "Promocja: "+string(gcvt(Podaj_promocje(),3,t));}
ostream& operator<<(ostream& wy, TProdukt3& p)
{ return wy<<p.toString()<<endl; }
1: 1
Insert

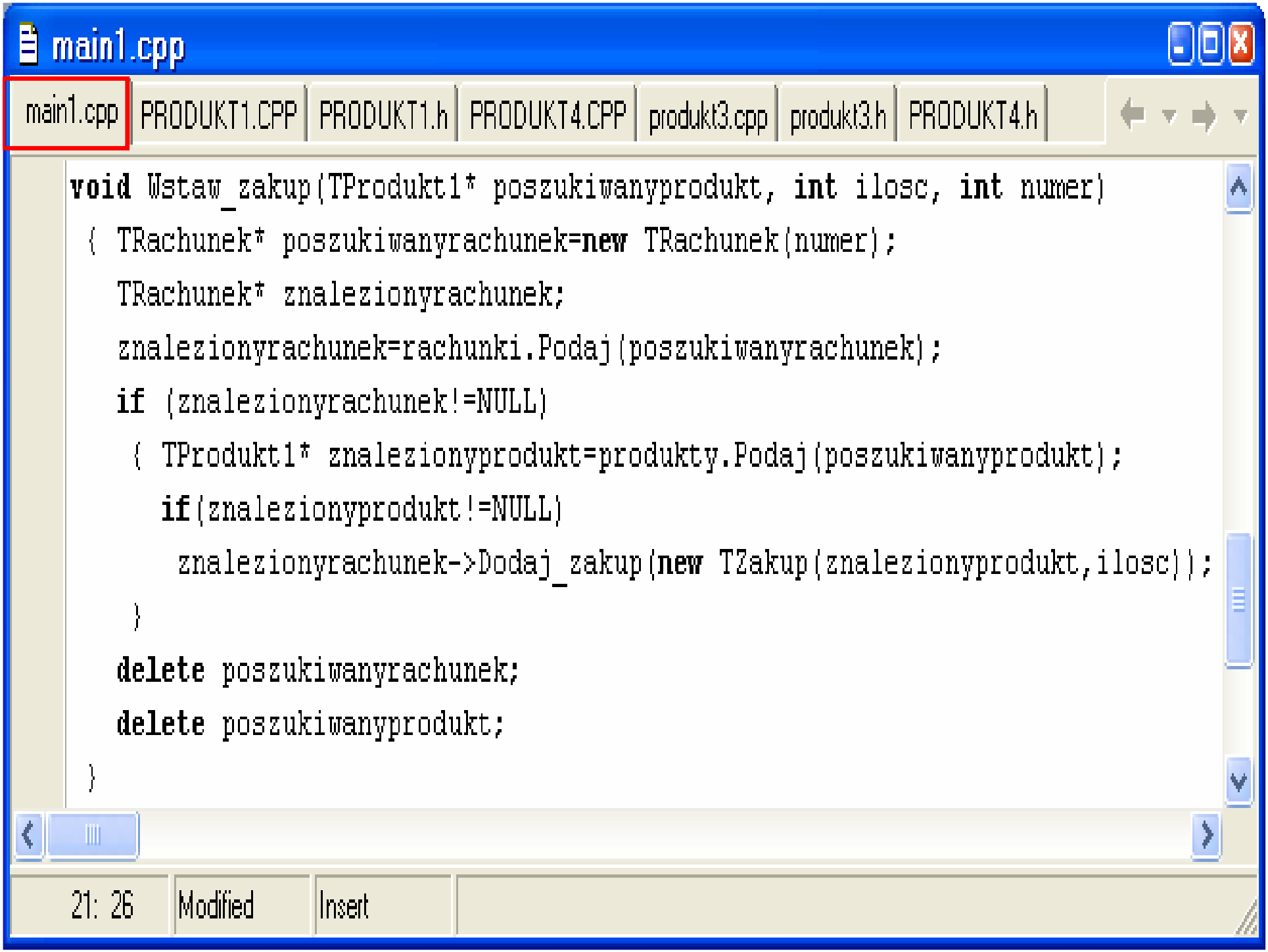

C:\Settings\dydaktyka\Programowanie_obiektowe\w9_4\PROD...
Lab9_4.bpf
produkt3.cpp
PRODUKT2.CPP
PRODUKT4.CPP
PRODUKT4.h
#ifndef _Produkt4
#define _Produkt4
#include "produkt2.h"
#include "produkt3.h"
class TProdukt4: public TProdukt2, public TProdukt3
{
public:
TProdukt4(string nazwa_="bez nazwy",float cena_=0,
float podatek_=0, float promocja_=0);
TProdukt4(TProdukt4&);
~TProdukt4();
float Podaj_cene();
float Podaj_promocje();
float Czesc_brutto();
string toString();
friend ostream& operator<<(ostream&, TProdukt4&);
};
#endif
Metoda wirtualna przesłaniająca metodę wirtualną Podaj_promocje() z klasy TProdukt1
8: 43
Modified
Insert

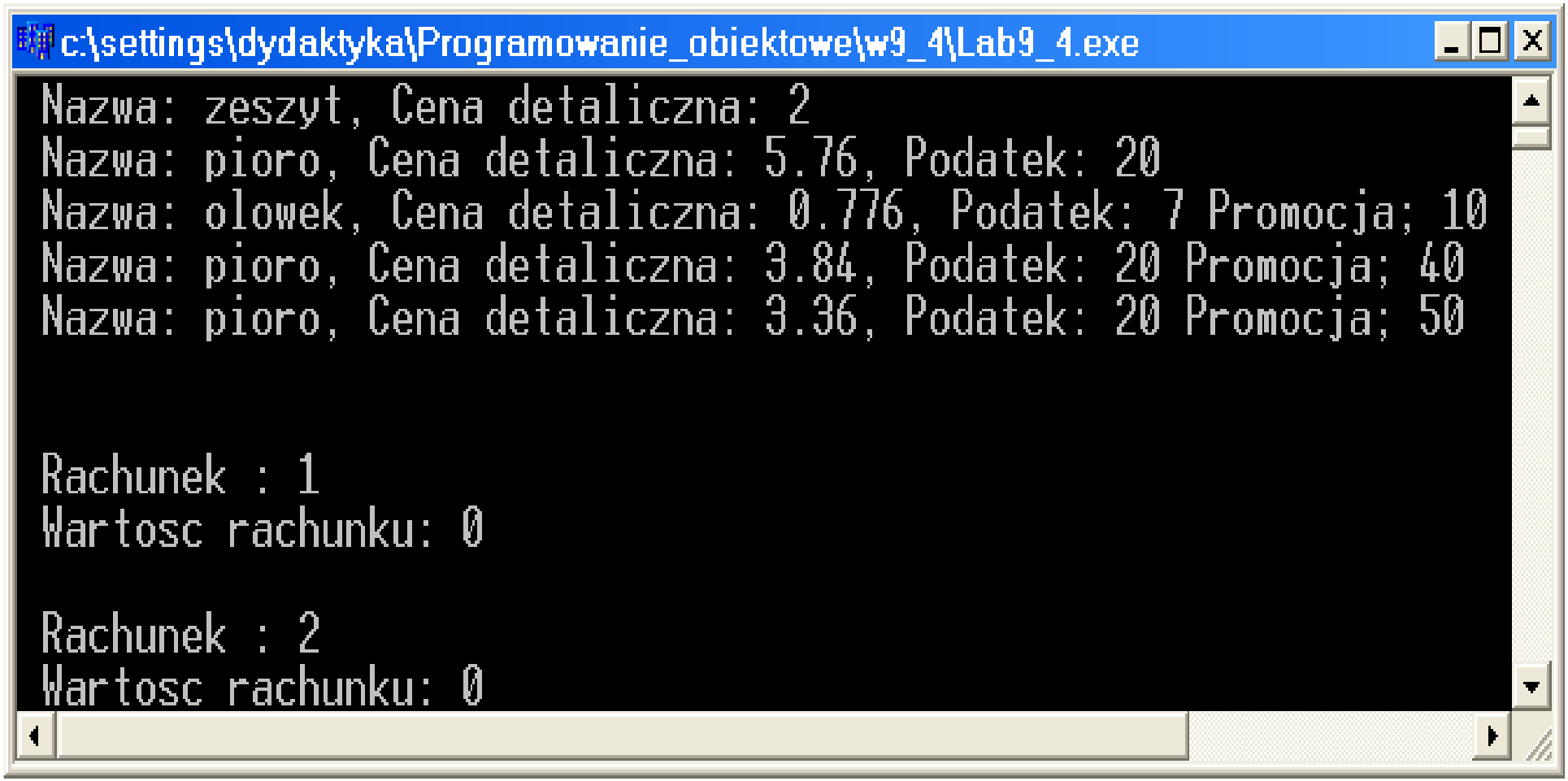

PRODUKT4.CPP
Lab9_4.bpf
produkt3.cpp
PRODUKT2.CPP
PRODUKT4.CPP
PRODUKT4.h
#include "produkt4.h"
TProdukt4::TProdukt4(string nazwa_, float cena_,
float podatek_, float promocja_):
//TProdukt1(nazwa_, cena_),
TProdukt2(nazwa_,cena_,podatek_),
TProdukt3(promocja_)
{ }
TProdukt4::TProdukt4(TProdukt4& p):
//TProdukt1(p),
TProdukt2(p),
TProdukt3(p)
{ }
TProdukt4::~TProdukt4()
{ }
12: 56
Modified
Insert
Zależnie od kolejności umieszczenia klas w liście argumentów (od lewej do prawej), w takiej kolejności wywołane i wykonane są konstruktory tych klas. Oznacza to, że pierwszy jest wywołany konstruktor TProdukt2, a wykonane są: TAbstrakcyjny, TProdukt1, a na końcu TProdukt2. Następnie wykonany jest konstruktor TProdukt3. Nie można wywołać konstruktora klasy TProdukt1, ponieważ jest dziczenie tej klasy odbywa się za pośrednictwem klasy TProdukt2

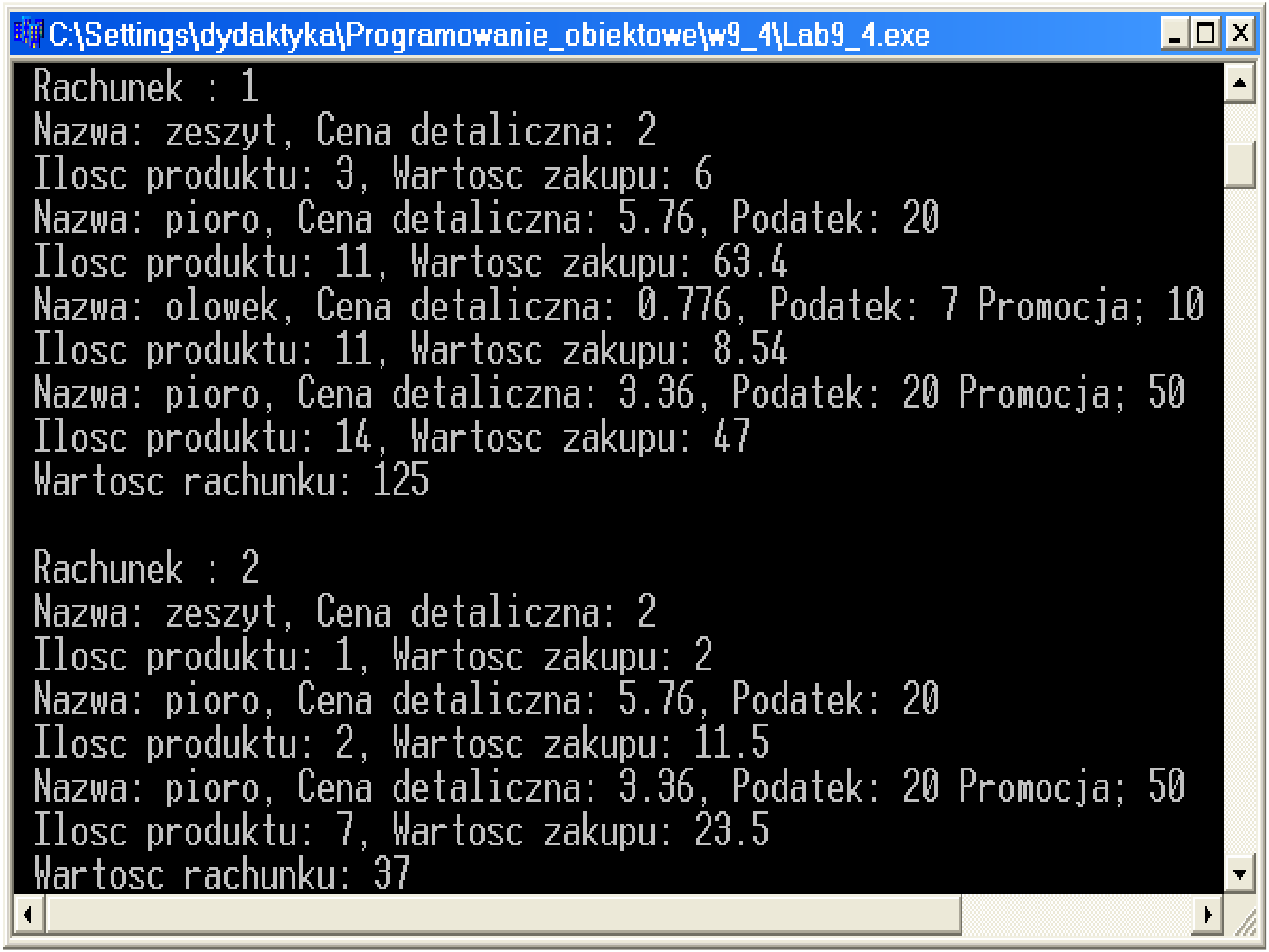

PRODUKT4.CPP
PRODUKT1.CPP
PRODUKT1.h
PRODUKT4.CPP
produkt3.cpp
produkt3.h
float TProdukt4::Czesc_brutto()
{ return TProdukt2::Czesc_brutto()-
TProdukt3::Podaj_promocje()*TProdukt1::Podaj_cene()/100; }
float TProdukt4::Podaj_cene()
{ return TProdukt1::Podaj_cene() + Czesc_brutto(); }
float TProdukt4::Podaj_promocje()
{ return TProdukt3::Podaj_promocje();}
string TProdukt4::toString()
{ char t[10];
return TProdukt2::toString() +
" Promocja; "+gcvt(TProdukt3::Podaj_promocje(),3,t); }
ostream& operator<<(ostream& wy, TProdukt2& p)
{ return wy<<p.toString()<<endl; }
Wskazanie, która z dziedziczonych metod powinna być wywołana
32: 62
Modified
Insert


main1.cpp
main1.cpp
PRODUKT1.CPP
PRODUKT1.h
PRODUKT4.CPP
produkt3.cpp
prod
#include "Rachunek.h"
TKol2<TProdukt1> produkty;
TKol2<TRachunek> rachunki;
void Wstaw_zakup(TProdukt1* poszukiwanyprodukt,
int ilosc, int numer);
void main()
{
TProdukt1* p1;
TProdukt2* p2;
TProdukt4* p4;
p1 = new TProdukt1("zeszyt", 2.0);
p2 = new TProdukt2("pioro",4.80,20);
p2 = new TProdukt2("pioro",4.80,20);
p4 = new TProdukt4("olowek",0.80,7,10);
p4 = new TProdukt4("pioro",4.80,20,40);
p4 = new TProdukt4("pioro",4.80,20,50);
cout<<produkty<<endl;
produkty.Wstaw(p1);
produkty.Wstaw(p2);
produkty.Wstaw(p2);
produkty.Wstaw(p4);
produkty.Wstaw(p4);
produkty.Wstaw(p4);
Nie korzysta się z obiektów typu TProdukt3
17: 58
Modified
Insert


main1.cpp
main1.cpp
PRODUKT1.CPP
PRODUKT1.h
PRODUKT4.CPP
produkt3.cpp
rachunki.Wstaw(new TRachunek(1));
rachunki.Wstaw(new TRachunek(2));
cout<<rachunki<<endl;
Wstaw_zakup(new TProdukt1("zeszyt", 2.0),1,1);
Wstaw_zakup(new TProdukt1("zeszyt", 2.0),2,1);
Wstaw_zakup(new TProdukt2("pioro",4.80,20),5,1);
Wstaw_zakup(new TProdukt2("pioro",4.80,20),6,1);
Wstaw_zakup(new TProdukt4("olowek",0.80,7,10),5,1);
Wstaw_zakup(new TProdukt4("olowek",0.80,7,10),6,1);
Wstaw_zakup(new TProdukt4("pioro",4.80,20,50),7,1);
Wstaw_zakup(new TProdukt4("pioro",4.80,20,50),7,1);
Wstaw_zakup(new TProdukt1("zeszyt", 2.0),1,2);
Wstaw_zakup(new TProdukt2("pioro",4.80,20),2,2);
Wstaw_zakup(new TProdukt4("pioro",4.80,20,50),7,2);
Nie korzysta się z obiektów typu TProdukt3
cout<<rachunki<<endl;
produkty.Usun_kolekcje();
rachunki.Usun_kolekcje();
cin.get();
}
21: 26
Modified
Insert


main1.cpp
main1.cpp
PRODUKT1.CPP
PRODUKT1.h
PRODUKT4.CPP
produkt3.cpp
produkt3.h
PRODUKT4.h
void Wstaw_zakup(TProdukt1* poszukiwanyprodukt, int ilosc, int numer)
{ TRachunek* poszukiwanyrachunek=new TRachunek(numer);
TRachunek* znalezionyrachunek;
znalezionyrachunek=rachunki.Podaj(poszukiwanyrachunek);
if (znalezionyrachunek!=NULL)
{ TProdukt1* znalezionyprodukt=produkty.Podaj(poszukiwanyprodukt);
if(znalezionyprodukt!=NULL)
znalezionyrachunek->Dodaj_zakup(new TZakup(znalezionyprodukt,ilosc));
}
delete poszukiwanyrachunek;
delete poszukiwanyprodukt;
}
21: 26
Modified
Insert

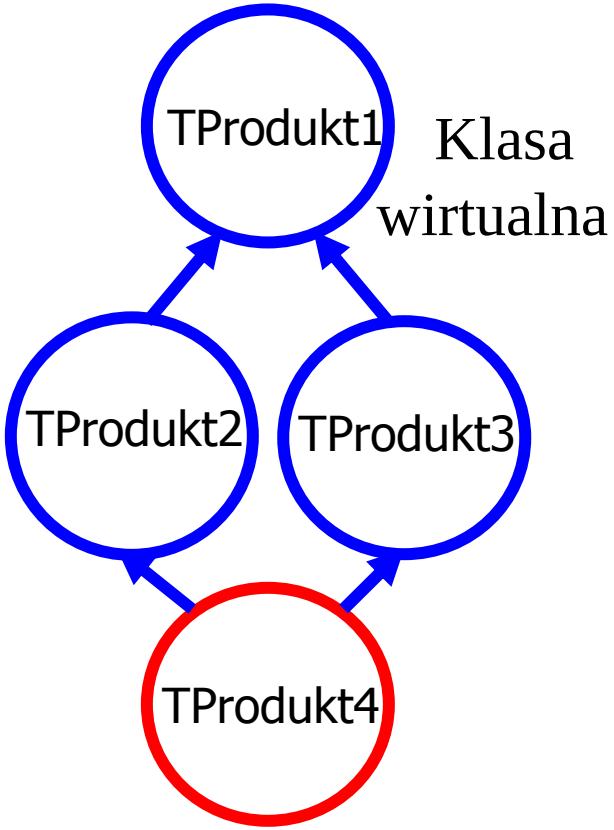

c:\settings\dydaktyka\Programowanie_obiektowe\w9_4\Lab9_4.exe
Nazwa: zeszyt, Cena detaliczna: 2
Nazwa: pioro, Cena detaliczna: 5.76, Podatek: 20
Nazwa: olowek, Cena detaliczna: 0.776, Podatek: 7 Promocja; 10
Nazwa: pioro, Cena detaliczna: 3.84, Podatek: 20 Promocja; 40
Nazwa: pioro, Cena detaliczna: 3.36, Podatek: 20 Promocja; 50
Rachunek : 1
Wartosc rachunku: 0
Rachunek : 2
Wartosc rachunku: 0

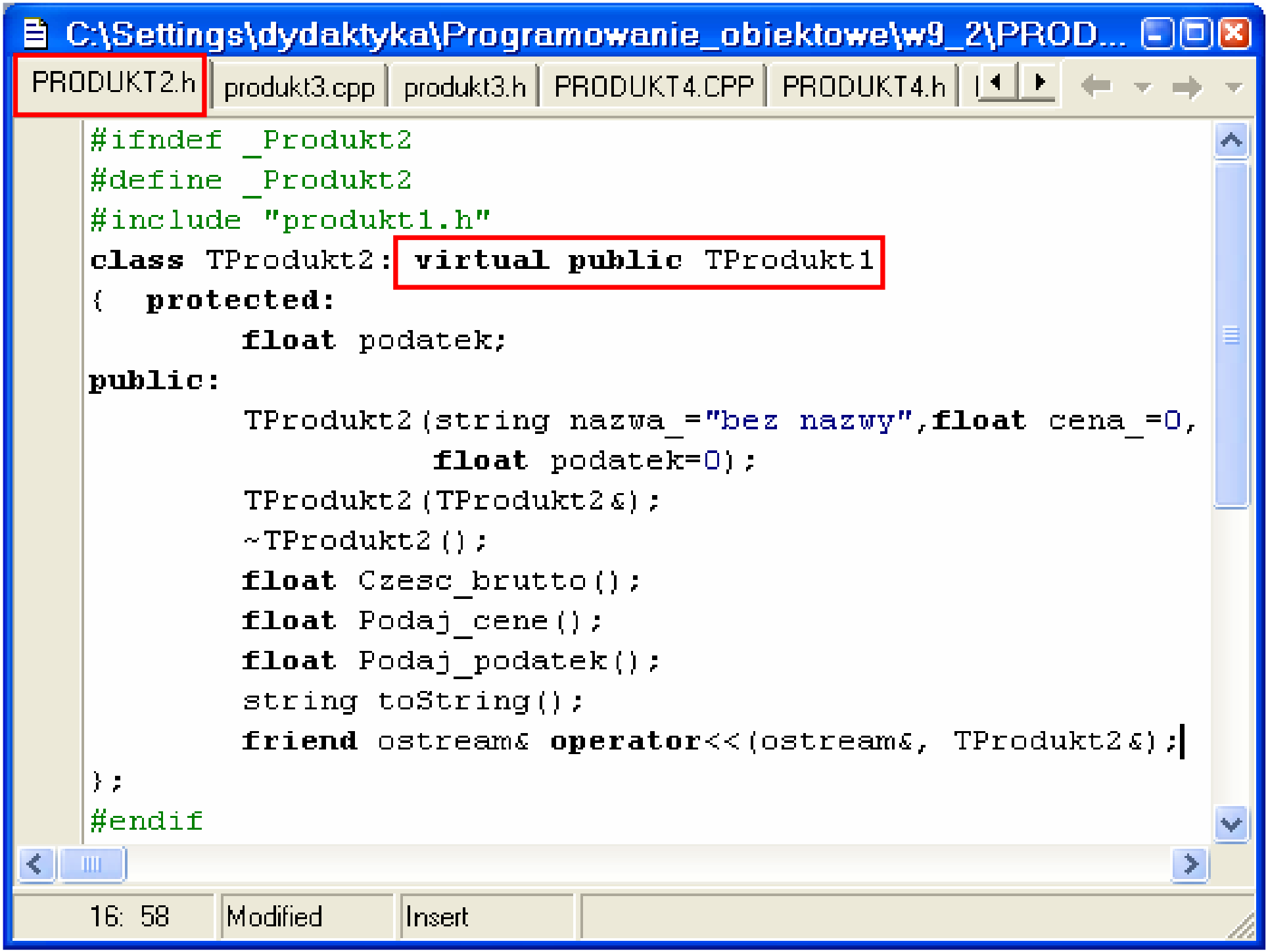

C:\Settings\dydaktyka\Programowanie_obiektowe\w9_4\Lab9_4.exe
Rachunek : 1
Nazwa: zeszyt, Cena detaliczna: 2
Ilosc produktu: 3, Wartosc zakupu: 6
Nazwa: pioro, Cena detaliczna: 5.76, Podatek: 20
Ilosc produktu: 11, Wartosc zakupu: 63.4
Nazwa: olowek, Cena detaliczna: 0.776, Podatek: 7 Promocja; 10
Ilosc produktu: 11, Wartosc zakupu: 8.54
Nazwa: pioro, Cena detaliczna: 3.36, Podatek: 20 Promocja; 50
Ilosc produktu: 14, Wartosc zakupu: 47
Wartosc rachunku: 125
Rachunek : 2
Nazwa: zeszyt, Cena detaliczna: 2
Ilosc produktu: 1, Wartosc zakupu: 2
Nazwa: pioro, Cena detaliczna: 5.76, Podatek: 20
Ilosc produktu: 2, Wartosc zakupu: 11.5
Nazwa: pioro, Cena detaliczna: 3.36, Podatek: 20 Promocja; 50
Ilosc produktu: 7, Wartosc zakupu: 23.5
Wartosc rachunku: 37

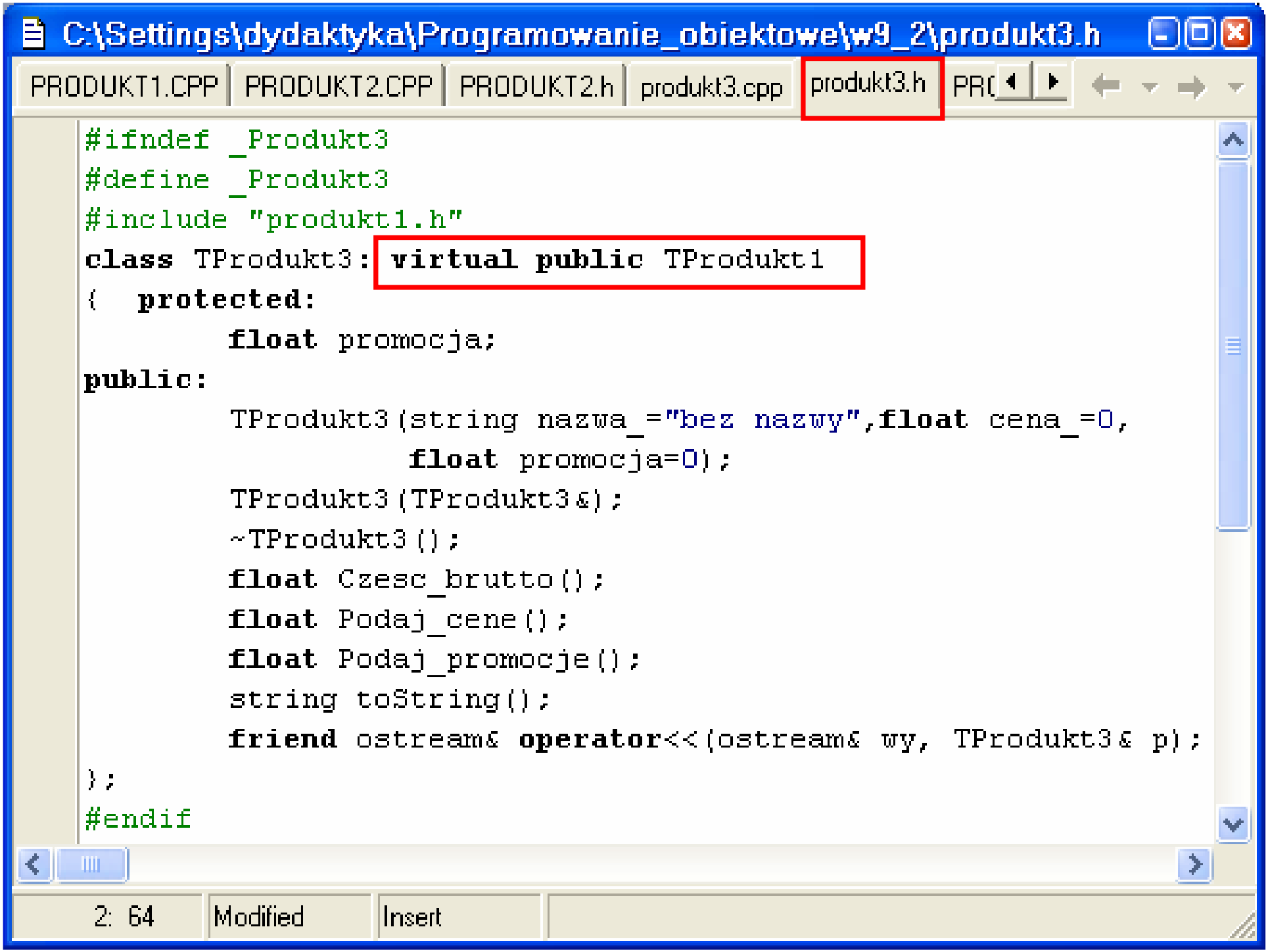
Dziedziczenie wielobazowe-uzupełnienie

1. **Dziedziczenie wielobazowe z powtórzeniami z klasą wirtualną, listy konstruktorów – listy argumentów konstruktorów**
2. **Dziedziczenie wielobazowe z powtórzeniami bez klasy wirtualnej, listy konstruktorów – problemy z polimorfizmem związanym z klasą TProdukt1 dziedziczoną dwukrotnie**
3. **Dziedziczenie wielobazowe bez powtórzeń**
4. **Wzorzec fasady – klasa TAplikacja hermetyzująca dostęp do kolekcji produktów i rachunków**

Dziedziczenie wielobazowe z powtórzeniami




C:\Settings\dydaktyka\Programowanie_obiektowe\w9_2\PROD...
PRODUKT2.h
produkt3.cpp
produkt3.h
PRODUKT4.CPP
PRODUKT4.h
#ifndef _Produkt2
#define _Produkt2
#include "produkt1.h"
class TProdukt2: virtual public TProdukt1
{ protected:
float podatek;
public:
TProdukt2(string nazwa_="bez nazwy",float cena_=0,
float podatek=0);
TProdukt2(TProdukt2&);
~TProdukt2();
float Czesc_brutto();
float Podaj_cene();
float Podaj_podatek();
string toString();
friend ostream& operator<<(ostream&, TProdukt2&);
};
#endif
16: 58
Modified
Insert

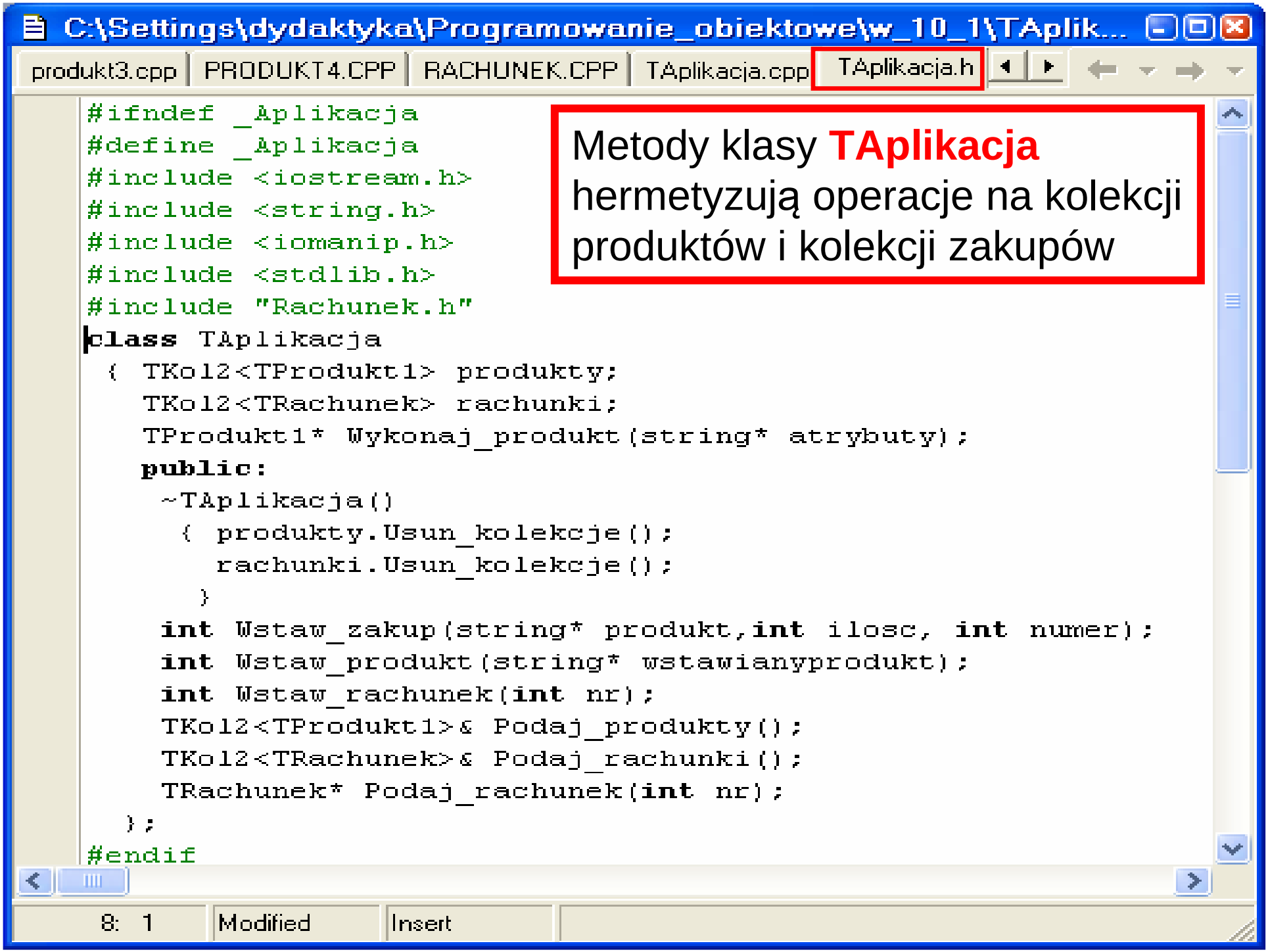

C:\Settings\dydaktyka\Programowanie_obiektowe\w9_2\produkt3.h
PRODUKT1.CPP
PRODUKT2.CPP
PRODUKT2.h
produkt3.cpp
produkt3.h
#ifndef _Produkt3
#define _Produkt3
#include "produkt1.h"
class TProdukt3: virtual public TProdukt1
{ protected:
float promocja;
public:
TProdukt3(string nazwa_="bez nazwy",float cena_=0,
float promocja=0);
TProdukt3(TProdukt3&);
~TProdukt3();
float Czesc_brutto();
float Podaj_cene();
float Podaj_promocje();
string toString();
friend ostream& operator<<(ostream& wy, TProdukt3& p);
};
#endif
2: 64
Modified
Insert

TProdukt4 dziedziczy od dwóch klas TProdukt2, TProdukt2 i pośrednio od TProdukt1

PRODUKT2.h | produkt3.cpp | produkt3.h | PRODUKT4.CPP | PRODUKT4.h

```
#ifndef _Produkt4
#define _Produkt4
#include "produkt2.h"
#include "produkt3.h"
class TProdukt4: public TProdukt3, public TProdukt2
{
    public:
        TProdukt4(string nazwa_="bez nazwy",float cena_=0,
                  float podatek_=0, float promocja_=0);
        TProdukt4(TProdukt4&);
        ~TProdukt4();
        float Podaj_cene();
        float Czesc_brutto();
        string toString();
        friend ostream& operator<<(ostream&, TProdukt4&);
};
#endif
```

17: 48 | Modified | Insert

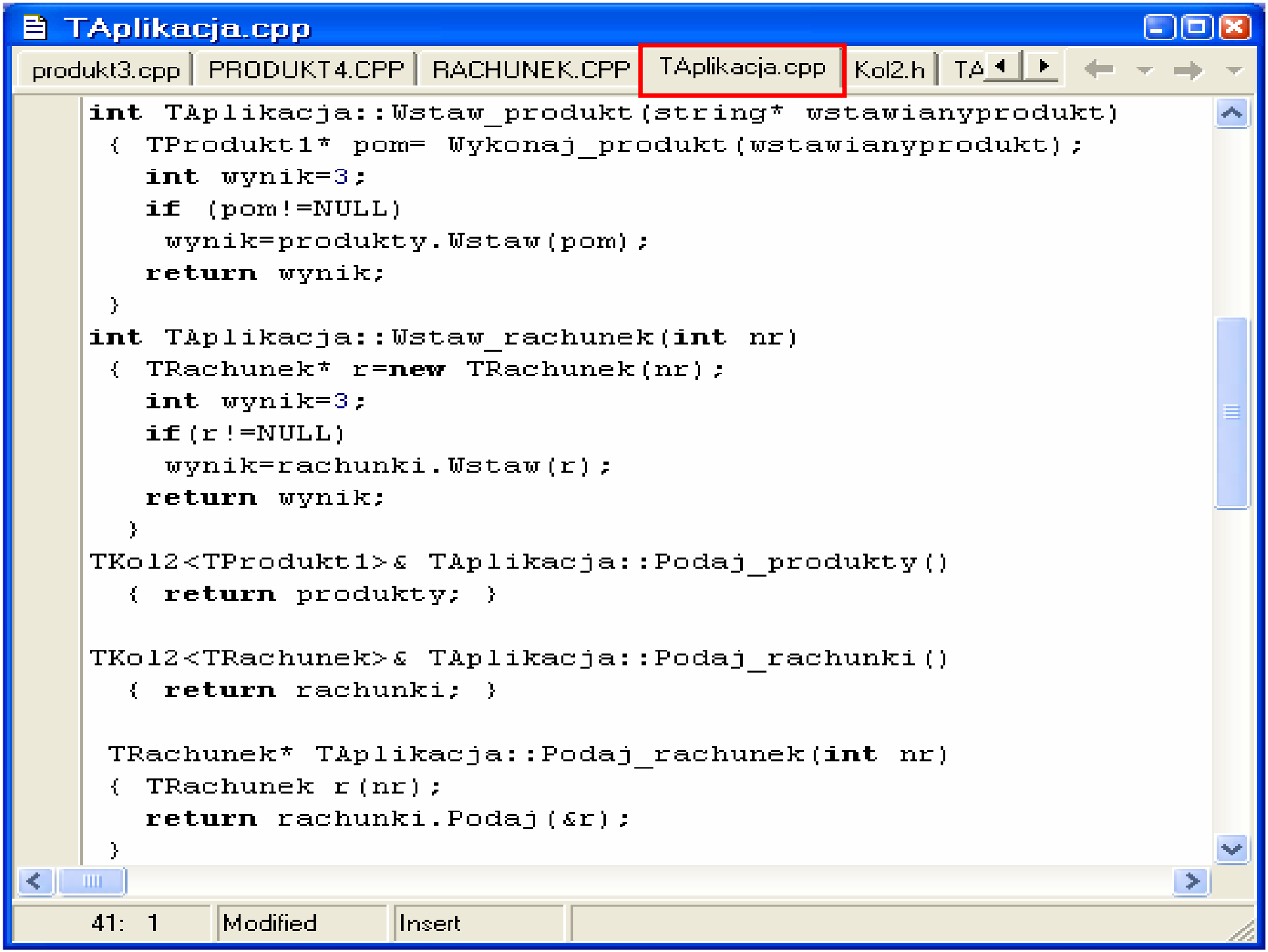

C:\Settings\dydaktyka\Programowanie_obiektowe\w_10_1\TAplik...
produkt3.cpp
PRODUKT4.CPP
RACHUNEK.CPP
TAplikacja.cpp
TAplikacja.h
#ifndef _Aplikacja
#define _Aplikacja
#include <iostream.h>
#include <string.h>
#include <iomanip.h>
#include <stdlib.h>
#include "Rachunek.h"
class TAplikacja
{ TKol2<TProdukt1> produkty;
TKol2<TRachunek> rachunki;
TProdukt1* Wykonaj_produkt(string* atrybuty);
public:
~TAplikacja()
{ produkty.Usun_kolekcje();
rachunki.Usun_kolekcje();
}
int Wstaw_zakup(string* produkt,int ilosc, int numer);
int Wstaw_produkt(string* wstawianyprodukt);
int Wstaw_rachunek(int nr);
TKol2<TProdukt1>& Podaj_produkty();
TKol2<TRachunek>& Podaj_rachunki();
TRachunek* Podaj_rachunek(int nr);
};
#endif
Metody klasy TAplikacja hermetyzują operacje na kolekcji produktów i kolekcji zakupów
8: 1
Modified
Insert

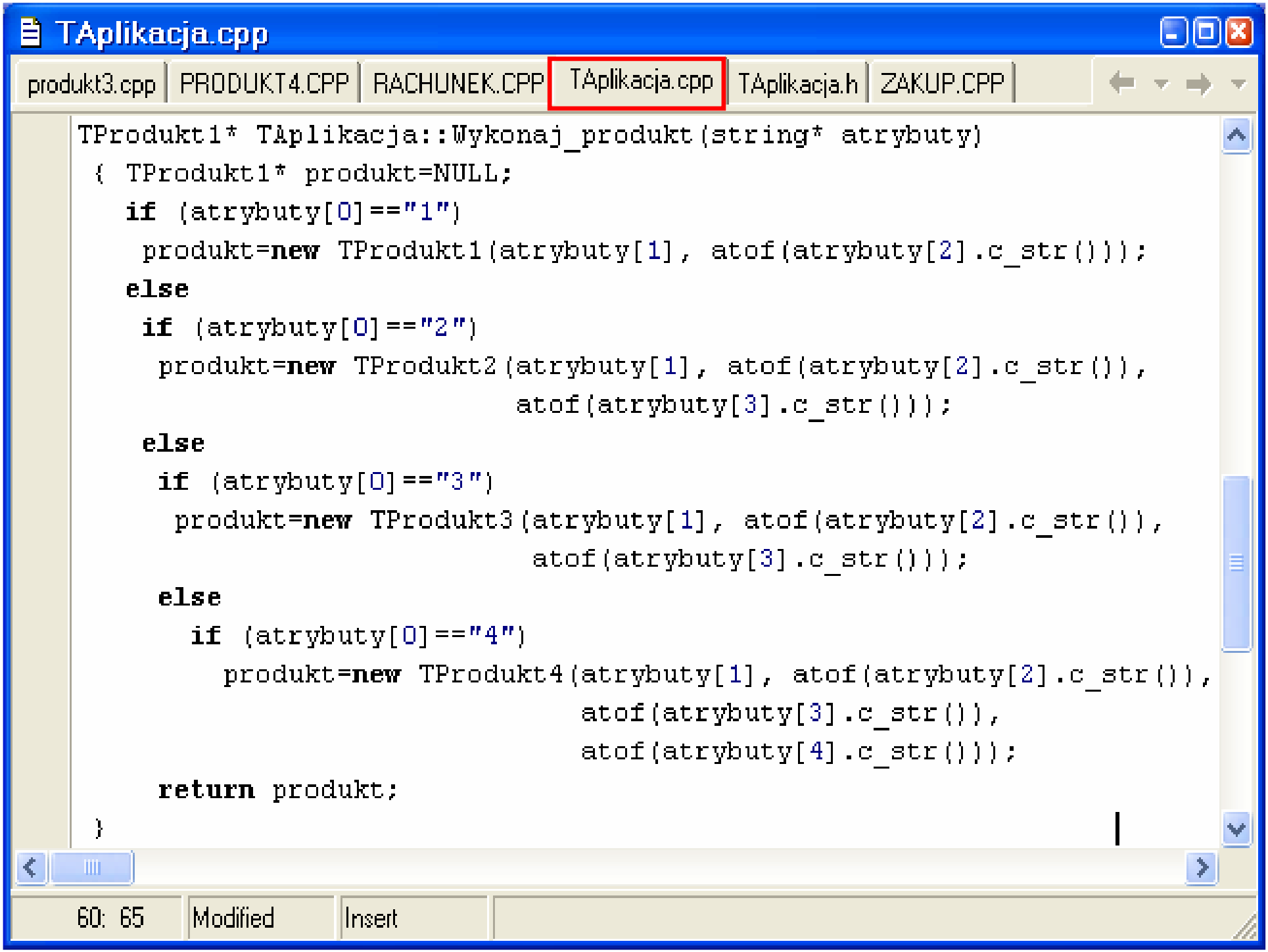
TAplikacja.cpp

produkt3.cpp | PRODUKT4.CPP | RACHUNEK.CPP | TAplikacja.cpp | Kol2.h | TAplikacja.h | ZAKUP.CPP

```
#include "TAplikacja.h"
int TAplikacja::Wstaw_zakup(string* produkt, int ilosc, int numer)
{   int wynik;
    TRachunek* poszukiwanyrachunek=new TRachunek(numer);
    TProdukt1* poszukiwanyprodukt=Wykonaj_produkt(produkt);
    if (poszukiwanyrachunek!=NULL && poszukiwanyprodukt!=NULL)
     { TRachunek* znalezionyrachunek;
       znalezionyrachunek=rachunki.Podaj(poszukiwanyrachunek);
       if (znalezionyrachunek!=NULL)
        { TProdukt1* znalezionyprodukt=
              produkty.Podaj(poszukiwanyprodukt);
          if(znalezionyprodukt!=NULL)
            wynik=
             znalezionyrachunek->Dodaj_zakup(new TZakup(znalezionyprodukt,ilosc));
          else
           wynik=5;
        }
       else
        wynik=4;
     }
    else
       wynik=3;
    delete poszukiwanyrachunek;
    delete poszukiwanyprodukt;
    return wynik;
 }
```

22: 15 | Modified | Insert


TAplikacja.cpp
produkt3.cpp
PRODUKT4.CPP
RACHUNEK.CPP
TAplikacja.cpp
Kol2.h
TA
int TAplikacja::Wstaw_produkt(string* wstawianyprodukt)
{ TProdukt1* pom= Wykonaj_produkt(wstawianyprodukt);
int wynik=3;
if (pom!=NULL)
wynik=produkty.Wstaw(pom);
return wynik;
}
int TAplikacja::Wstaw_rachunek(int nr)
{ TRachunek* r=new TRachunek(nr);
int wynik=3;
if(r!=NULL)
wynik=rachunki.Wstaw(r);
return wynik;
}
TKol2<TProdukt1>& TAplikacja::Podaj_produkty()
{ return produkty; }
TKol2<TRachunek>& TAplikacja::Podaj_rachunki()
{ return rachunki; }
TRachunek* TAplikacja::Podaj_rachunek(int nr)
{ TRachunek r(nr);
return rachunki.Podaj(&r);
}
41: 1
Modified
Insert


TAplikacja.cpp
produkt3.cpp
PRODUKT4.CPP
RACHUNEK.CPP
TAplikacja.cpp
TAplikacja.h
ZAKUP.CPP
TProdukt1* TAplikacja::Wykonaj_produkt(string* atrybuty)
{ TProdukt1* produkt=NULL;
if (atrybuty[0]=="1")
produkt=new TProdukt1(atrybuty[1], atof(atrybuty[2].c_str()));
else
if (atrybuty[0]=="2")
produkt=new TProdukt2(atrybuty[1], atof(atrybuty[2].c_str()),
atof(atrybuty[3].c_str()));
else
if (atrybuty[0]=="3")
produkt=new TProdukt3(atrybuty[1], atof(atrybuty[2].c_str()),
atof(atrybuty[3].c_str()));
else
if (atrybuty[0]=="4")
produkt=new TProdukt4(atrybuty[1], atof(atrybuty[2].c_str()),
atof(atrybuty[3].c_str()),
atof(atrybuty[4].c_str()));
return produkt;
}
60: 65
Modified
Insert

main1.cpp

produkt3.cpp | PRODUKT4.CPP | RACHUNEK.CPP | TAplikacja.cpp

```
#include "TAplikacja.h"
void main()
{
 TAplikacja aplikacja;
 string tablica1[3]= {"1","zeszyt", "2.0"};
 string tablica2[4]= {"2","pioro","4.80","20"};
 string tablica3[4]= {"3","pioro","4.80","10"};
 string tablica4[5]= {"4","olowek","0.80","7","10"};
 string tablica5[5]= {"4","pioro","4.80","20","50"};
 string tablica6[5]= {"4","pioro","4.80","20","60"};
 aplikacja.Wstaw_produkt(tablica1);
 aplikacja.Wstaw_produkt(tablica2);
 aplikacja.Wstaw_produkt(tablica2);
 aplikacja.Wstaw_produkt(tablica3);
 aplikacja.Wstaw_produkt(tablica3);
 aplikacja.Wstaw_produkt(tablica4);
 aplikacja.Wstaw_produkt(tablica5);
 aplikacja.Wstaw_produkt(tablica6);
 cout<<aplikacja.Podaj_produkty()<<endl;
```

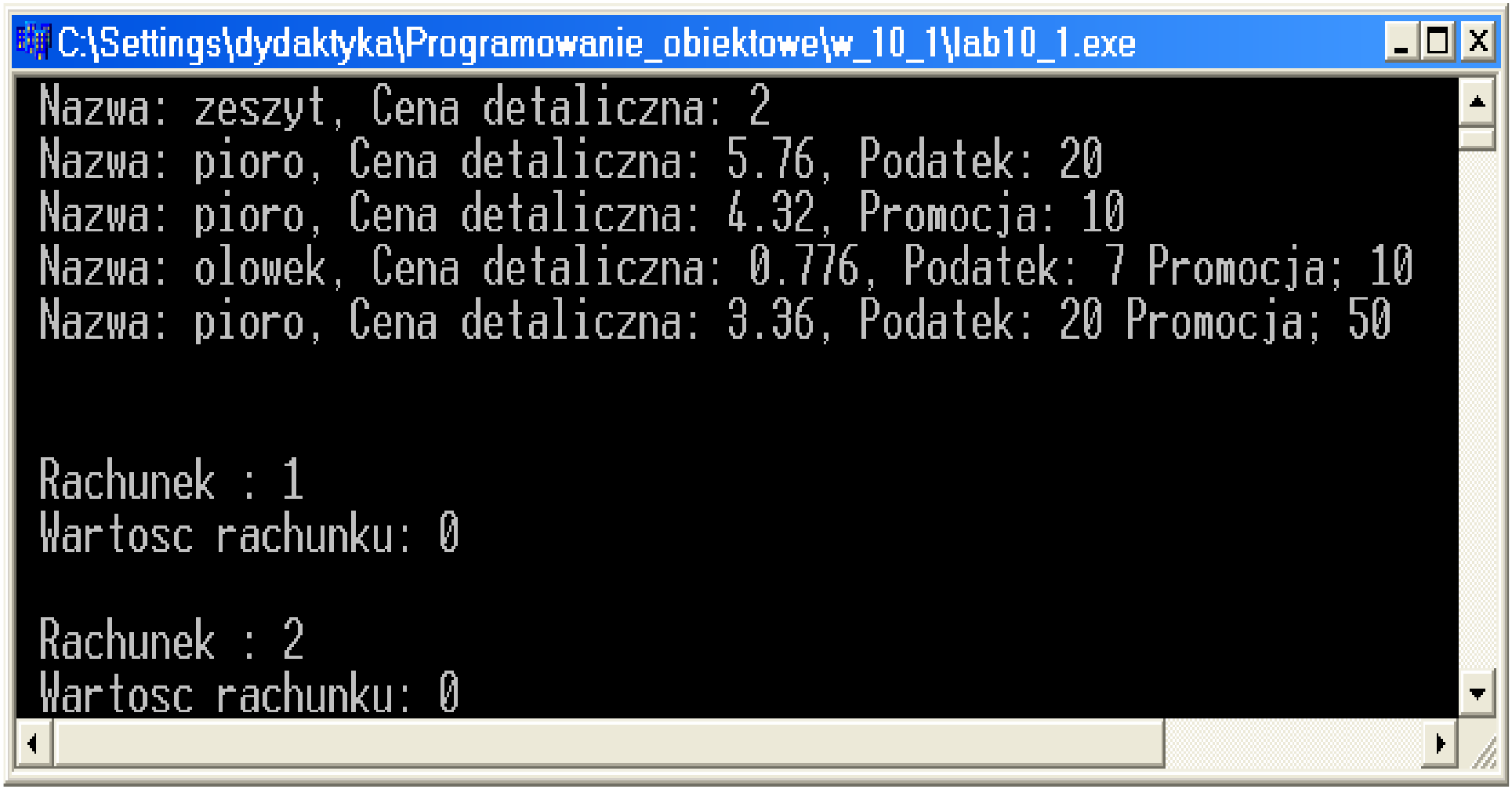
35: 32 | Insert

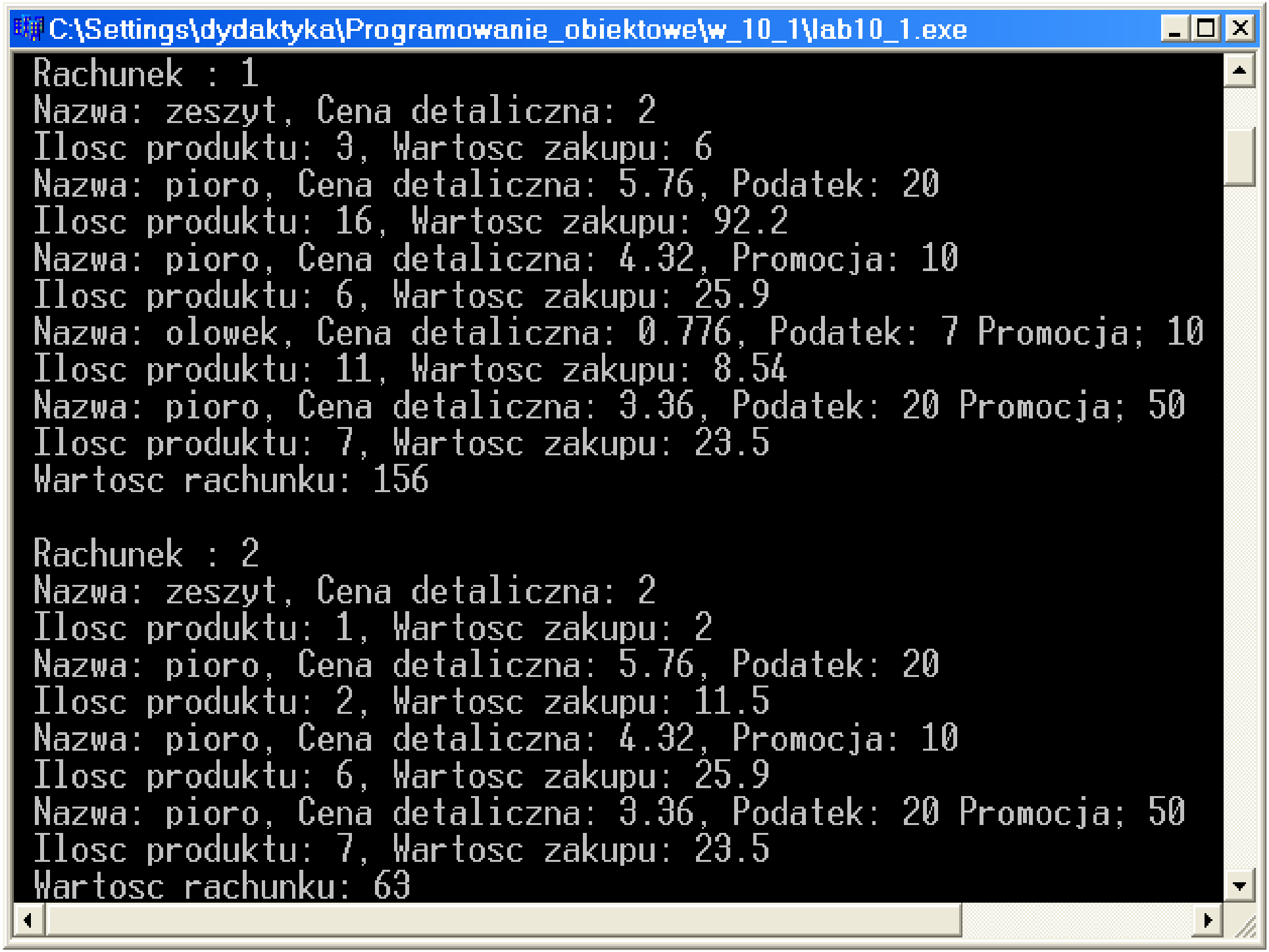
main1.cpp

produkt3.cpp | PRODUKT4.CPP | RACHUNEK.CPP | TAplikac

```
 aplikacja.Wstaw_rachunek(1);
 aplikacja.Wstaw_rachunek(1);
 aplikacja.Wstaw_rachunek(2);
 cout<<aplikacja.Podaj_rachunki()<<endl;

 aplikacja.Wstaw_zakup(tablica1,1,1);
 aplikacja.Wstaw_zakup(tablica1,2,1);
 aplikacja.Wstaw_zakup(tablica2,5,1);
 aplikacja.Wstaw_zakup(tablica2,6,1);
 aplikacja.Wstaw_zakup(tablica2,5,1);
 aplikacja.Wstaw_zakup(tablica3,6,1);
 aplikacja.Wstaw_zakup(tablica4,5,1);
 aplikacja.Wstaw_zakup(tablica4,6,1);
 aplikacja.Wstaw_zakup(tablica5,7,1);
 aplikacja.Wstaw_zakup(tablica6,7,1);

 aplikacja.Wstaw_zakup(tablica1,1,2);
 aplikacja.Wstaw_zakup(tablica2,2,2);
 aplikacja.Wstaw_zakup(tablica3,6,2);
 aplikacja.Wstaw_zakup(tablica5,7,2);
 cout<<aplikacja.Podaj_rachunki()<<endl;
 cin.get();
}
```

35: 32 | Insert


C:\Settings\dydaktyka\Programowanie_obiektowe\w_10_1\lab10_1.exe
Nazwa: zeszyt, Cena detaliczna: 2
Nazwa: pioro, Cena detaliczna: 5.76, Podatek: 20
Nazwa: pioro, Cena detaliczna: 4.32, Promocja: 10
Nazwa: olowek, Cena detaliczna: 0.776, Podatek: 7 Promocja; 10
Nazwa: pioro, Cena detaliczna: 3.36, Podatek: 20 Promocja; 50
Rachunek : 1
Wartosc rachunku: 0
Rachunek : 2
Wartosc rachunku: 0


C:\Settings\dydaktyka\Programowanie_obiektowe\w_10_1\lab10_1.exe
Rachunek : 1
Nazwa: zeszyt, Cena detaliczna: 2
Ilosc produktu: 3, Wartosc zakupu: 6
Nazwa: pioro, Cena detaliczna: 5.76, Podatek: 20
Ilosc produktu: 16, Wartosc zakupu: 92.2
Nazwa: pioro, Cena detaliczna: 4.32, Promocja: 10
Ilosc produktu: 6, Wartosc zakupu: 25.9
Nazwa: olowek, Cena detaliczna: 0.776, Podatek: 7 Promocja; 10
Ilosc produktu: 11, Wartosc zakupu: 8.54
Nazwa: pioro, Cena detaliczna: 3.36, Podatek: 20 Promocja; 50
Ilosc produktu: 7, Wartosc zakupu: 23.5
Wartosc rachunku: 156
Rachunek : 2
Nazwa: zeszyt, Cena detaliczna: 2
Ilosc produktu: 1, Wartosc zakupu: 2
Nazwa: pioro, Cena detaliczna: 5.76, Podatek: 20
Ilosc produktu: 2, Wartosc zakupu: 11.5
Nazwa: pioro, Cena detaliczna: 4.32, Promocja: 10
Ilosc produktu: 6, Wartosc zakupu: 25.9
Nazwa: pioro, Cena detaliczna: 3.36, Podatek: 20 Promocja; 50
Ilosc produktu: 7, Wartosc zakupu: 23.5
Wartosc rachunku: 63