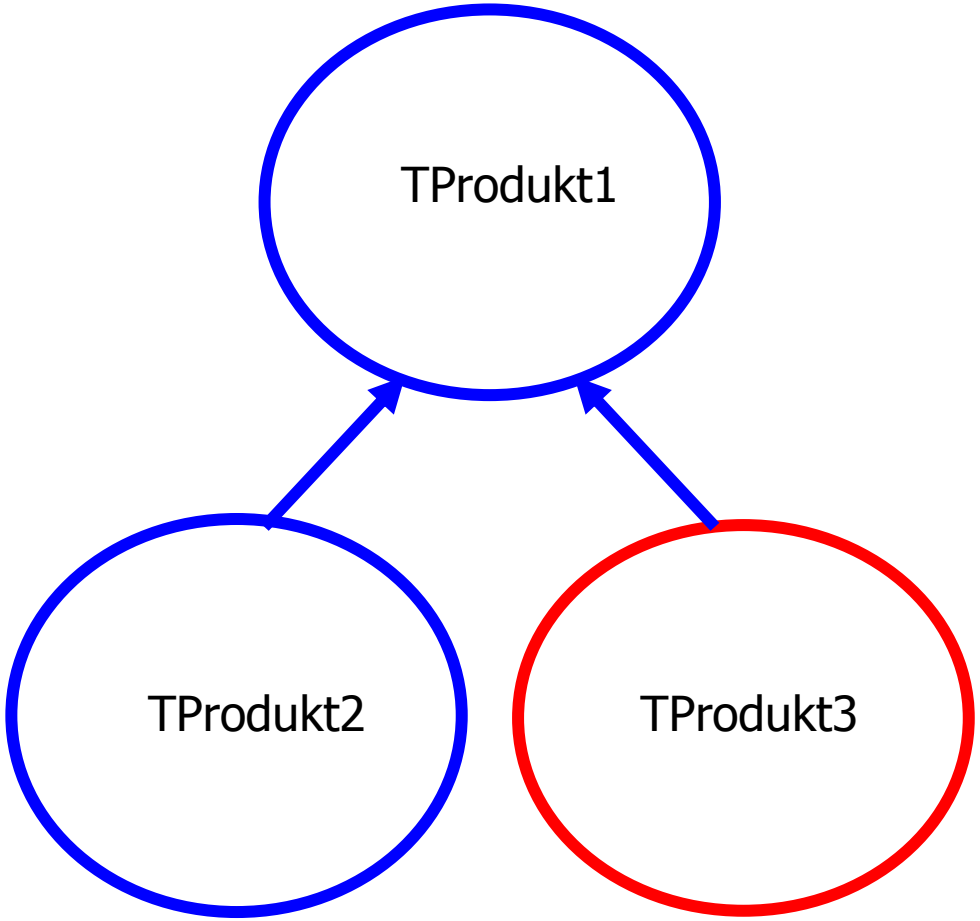
Dziedziczenie wielobazowe- zastosowanie

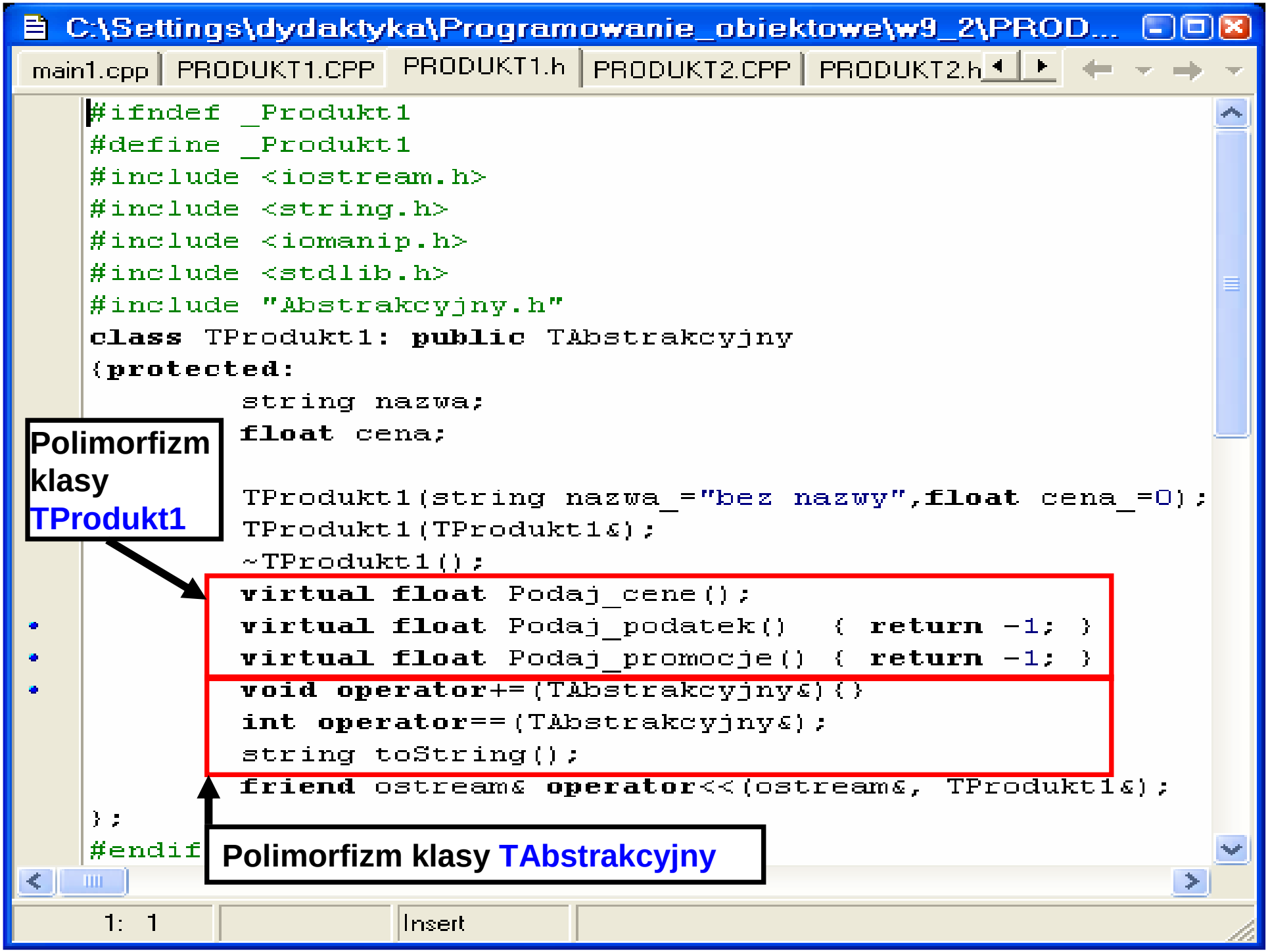
1. **Tworzenie rachunku – nowy typ klasy typu TProdukt3, pochodny klasy typu TProdukt1 – dziedziczenie jednobazowe**
2. **Dziedziczenie wielobazowe z powtórzeniami, listy konstruktorów**
3. **Tworzenie rachunku – nowy typ klasy typu TProdukt4, pochodny klas typu TProdukt2 i TProdukt3 – dziedziczenie dwubazowe**

Dziedziczenie wielobazowe- zastosowanie

1. **Tworzenie rachunku – nowy typ klasy typu TProdukt3, pochodny klasy typu TProdukt1 – dziedziczenie jednobazowe**

Rodzina produktów – uogólnienie cech wszystkich klas w klasie TProdukt1. Kod źródłowy klas tworzących rachunek nie ulegają zmianie



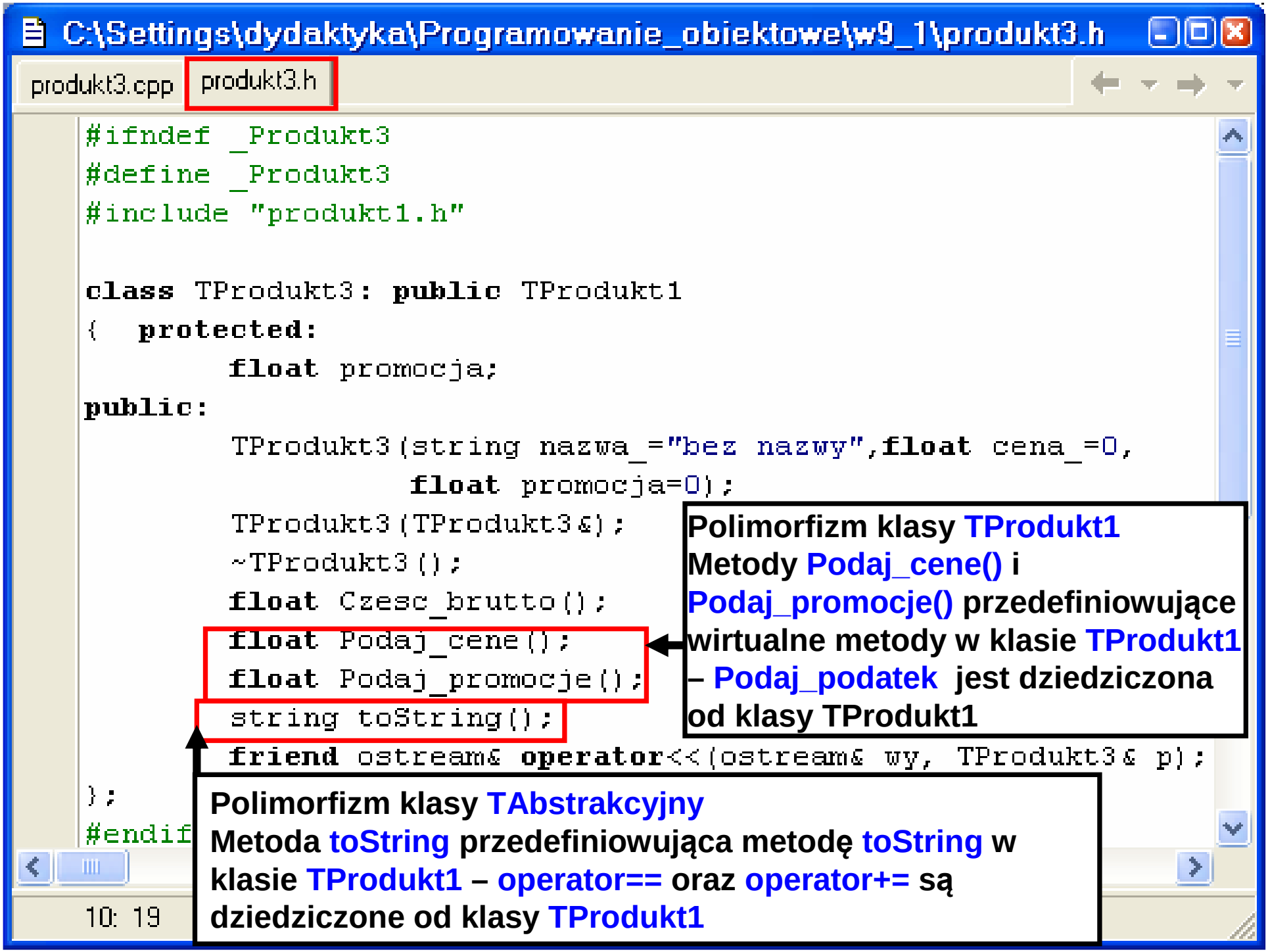

C:\Settings\dydaktyka\Programowanie_obiektowe\w9_2\PROD...
main1.cpp
PRODUKT1.CPP
PRODUKT1.h
PRODUKT2.CPP
PRODUKT2.h
#ifndef _Produkt1
#define _Produkt1
#include <iostream.h>
#include <string.h>
#include <iomanip.h>
#include <stdlib.h>
#include "Abstrakcyjny.h"
class TProdukt1: public TAbstrakcyjny
{protected:
string nazwa;
float cena;
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
Polimorfizm klasy TProdukt1
Polimorfizm klasy TAbstrakcyjny
1: 1
Insert

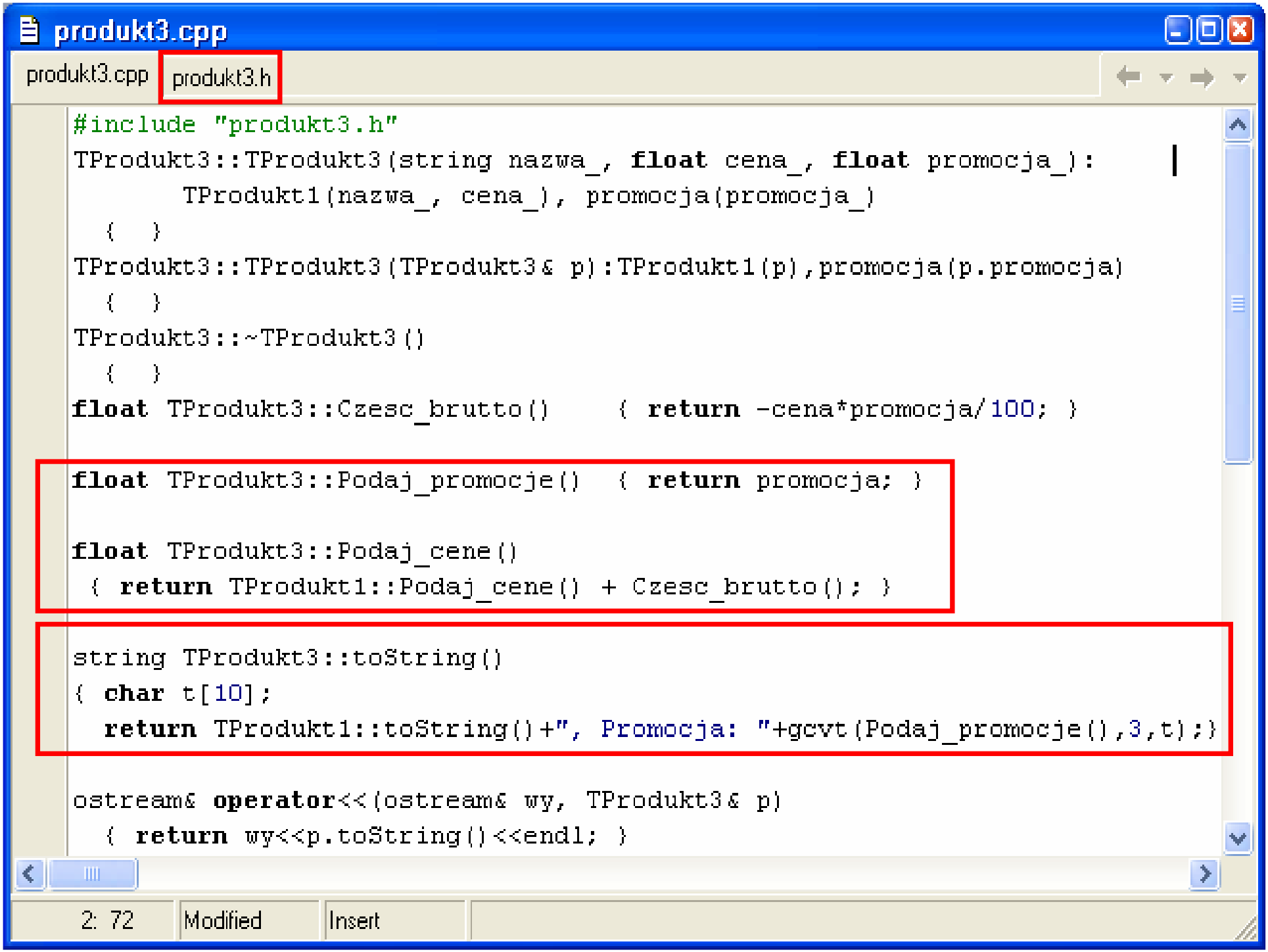
PRODUKT1.CPP

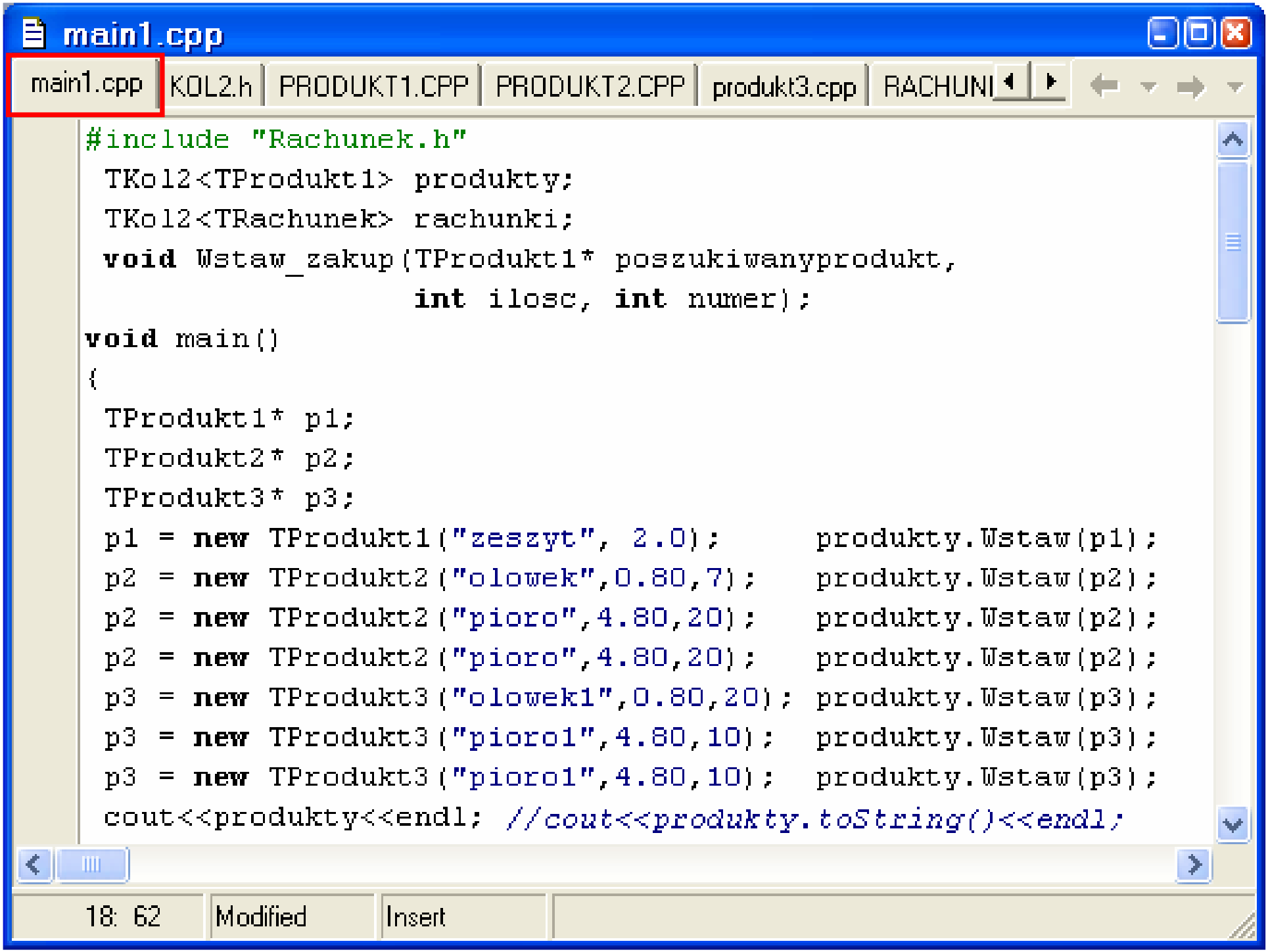
PRODUKT1.CPP | PRODUKT1.h | PRODUKT2.CPP | PRODUKT2.h | main1.cpp | produkt3.cpp | produkt3.

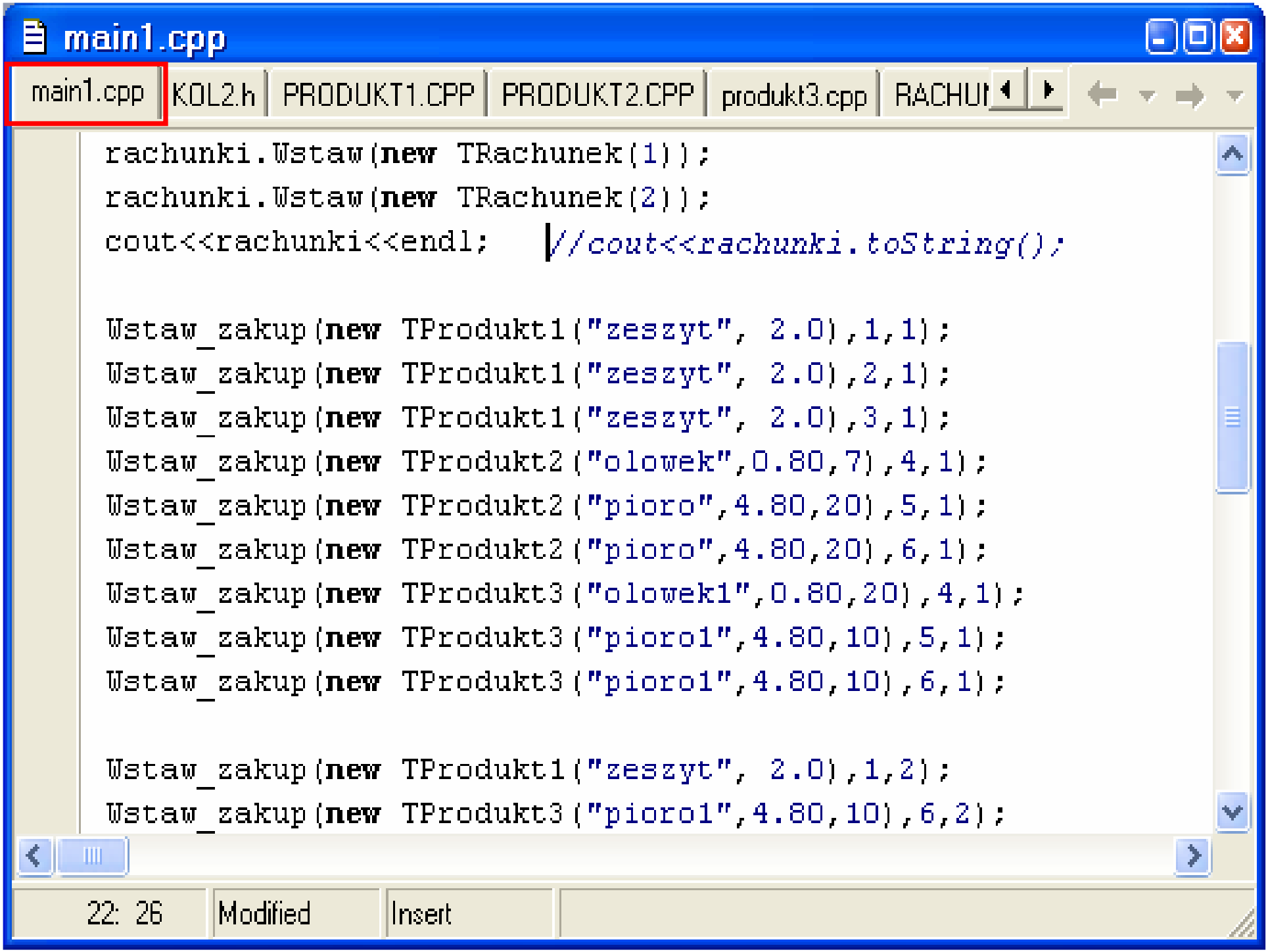
```
#include "produkt1.h"
TProdukt1::TProdukt1(string nazwa_,float cena_)
 { nazwa=nazwa_;
   cena=cena_; }
TProdukt1::TProdukt1(TProdukt1& p)
 { nazwa=p.nazwa;
   cena=p.cena; }
TProdukt1::~TProdukt1()
 { }
float TProdukt1::Podaj_cene()
 { return cena; }
int TProdukt1::operator==(TAbstrakcyjny& p_)
 {  TProdukt1& p = (TProdukt1&)p_;
    float a= Podaj_cene(), b= p.Podaj_cene();
    return nazwa==p.nazwa && a==b && Podaj_podatek()==p.Podaj_podatek()
           && Podaj_promocje()==p.Podaj_promocje();}
 string TProdukt1::toString()
 { char tab[10];
   return " Nazwa: "+nazwa+", Cena detaliczna: "+gcvt(Podaj_cene(),3,tab);}
 ostream& operator<<(ostream& wy, TProdukt1& p)
 { return wy<<p.toString()<<endl; }
```

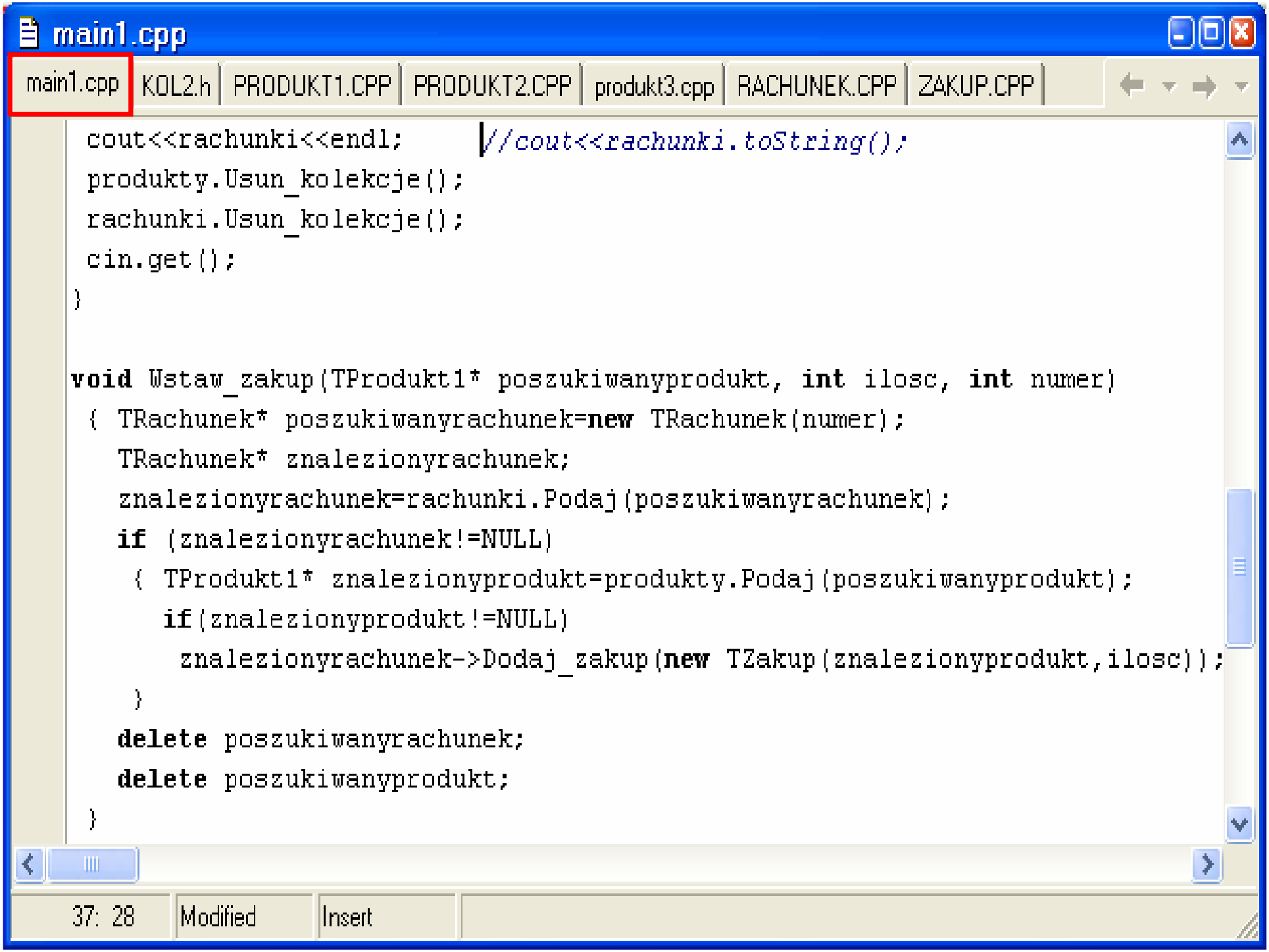
21: 3 | Modified | Insert

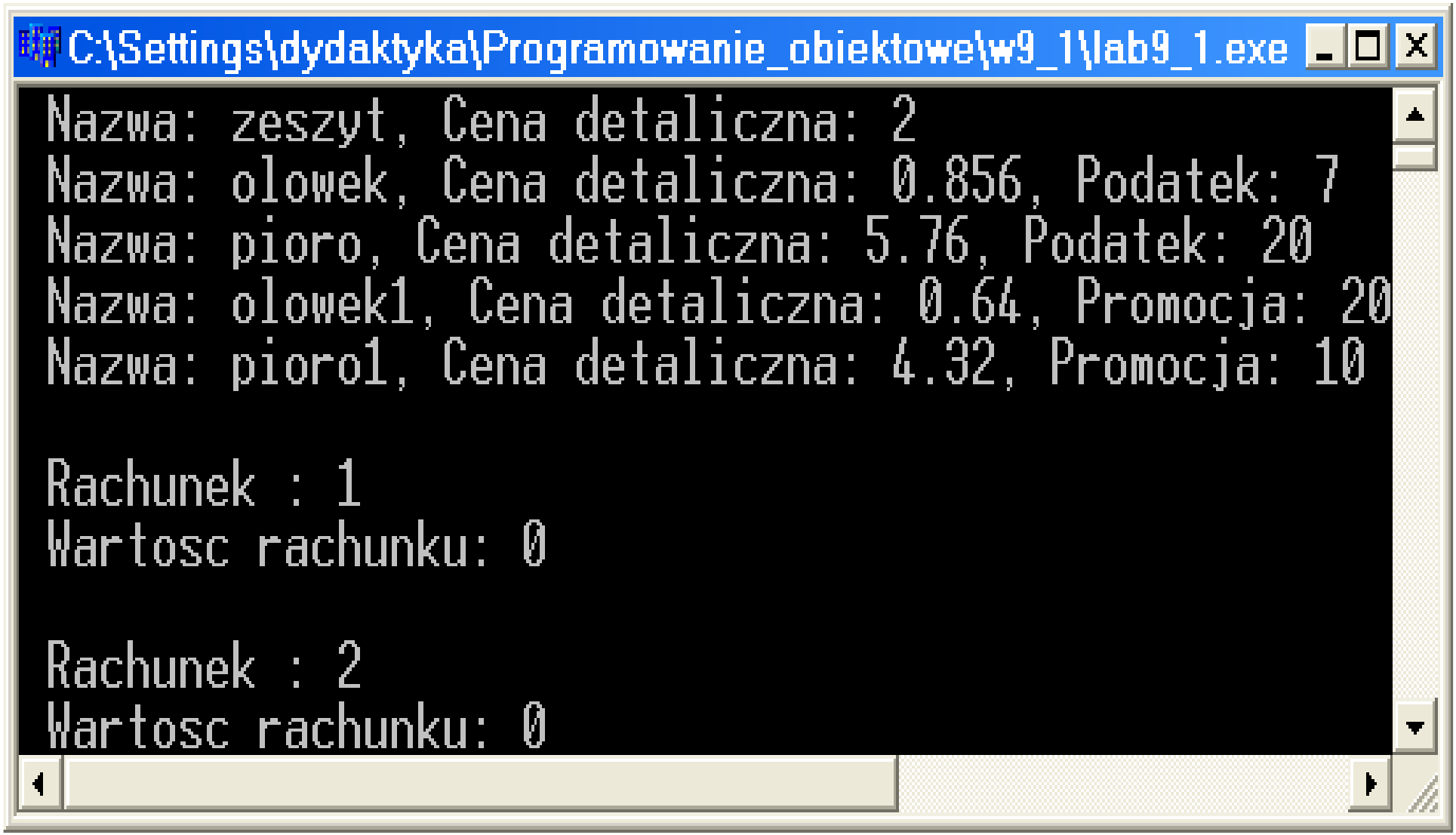

C:\Settings\dydaktyka\Programowanie_obiektowe\w9_1\produkt3.h
produkt3.cpp
produkt3.h
#ifndef _Produkt3
#define _Produkt3
#include "produkt1.h"
class TProdukt3: public TProdukt1
{ protected:
float promocja;
public:
TProdukt3(string nazwa_="bez nazwy",float cena_=0,
float promocja=0);
TProdukt3(TProdukt3&);
~TProdukt3();
float Czesc_brutto();
float Podaj_cene();
float Podaj_promocje();
string toString();
friend ostream& operator<<(ostream& wy, TProdukt3& p);
};
#endif
10: 19
Polimorfizm klasy TProdukt1
Metody Podaj_cene() i Podaj_promocje() przedefiniowujące wirtualne metody w klasie TProdukt1 – Podaj_podatek jest dziedziczona od klasy TProdukt1
Polimorfizm klasy TAbstrakcyjny
Metoda toString przedefiniowująca metodę toString w klasie TProdukt1 – operator== oraz operator+= są dziedziczone od klasy TProdukt1

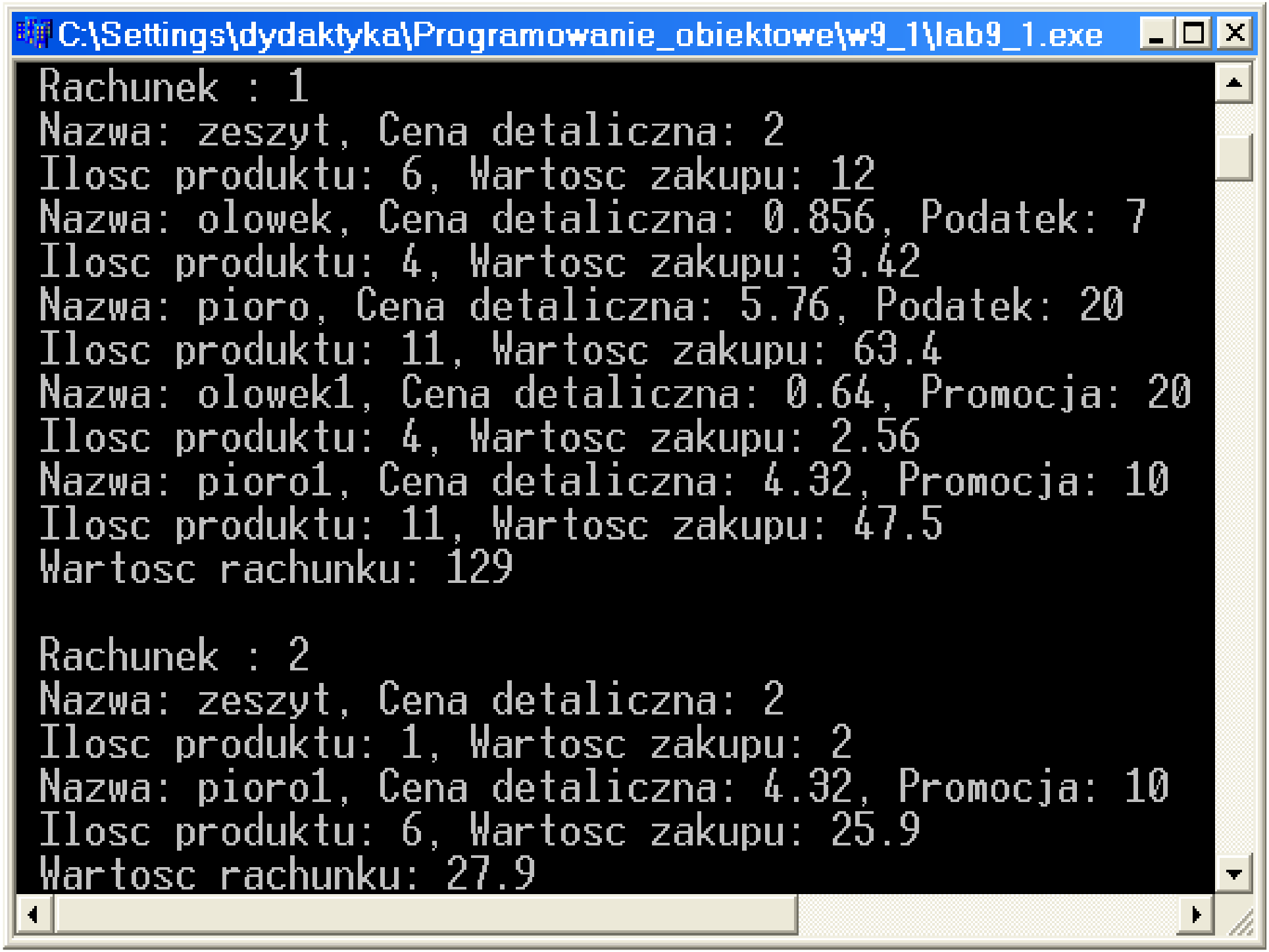

produkt3.cpp
produkt3.cpp
produkt3.h
#include "produkt3.h"
TProdukt3::TProdukt3(string nazwa_, float cena_, float promocja_):
TProdukt1(nazwa_, cena_), promocja(promocja_)
{ }
TProdukt3::TProdukt3(TProdukt3& p):TProdukt1(p),promocja(p.promocja)
{ }
TProdukt3::~TProdukt3()
{ }
float TProdukt3::Czesc_brutto() { return -cena*promocja/100; }
float TProdukt3::Podaj_promocje() { return promocja; }
float TProdukt3::Podaj_cene()
{ return TProdukt1::Podaj_cene() + Czesc_brutto(); }
string TProdukt3::toString()
{ char t[10];
return TProdukt1::toString()+", Promocja: "+gcvt(Podaj_promocje(),3,t);}
ostream& operator<<(ostream& wy, TProdukt3& p)
{ return wy<<p.toString()<<endl; }
2: 72
Modified
Insert


main1.cpp
main1.cpp
KOL2.h
PRODUKT1.CPP
PRODUKT2.CPP
produkt3.cpp
RACHUNI
#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;
p1 = new TProdukt1("zeszyt", 2.0); produkty.Wstaw(p1);
p2 = new TProdukt2("olowek",0.80,7); produkty.Wstaw(p2);
p2 = new TProdukt2("pioro",4.80,20); produkty.Wstaw(p2);
p2 = new TProdukt2("pioro",4.80,20); produkty.Wstaw(p2);
p3 = new TProdukt3("olowek1",0.80,20); produkty.Wstaw(p3);
p3 = new TProdukt3("pioro1",4.80,10); produkty.Wstaw(p3);
p3 = new TProdukt3("pioro1",4.80,10); produkty.Wstaw(p3);
cout<<produkty<<endl; //cout<<produkty.toString()<<endl;
18: 62
Modified
Insert


main1.cpp
main1.cpp
KOL2.h
PRODUKT1.CPP
PRODUKT2.CPP
produkt3.cpp
RACHU
rachunki.Wstaw(new TRachunek(1));
rachunki.Wstaw(new TRachunek(2));
cout<<rachunki<<endl; //cout<<rachunki.toString();
Wstaw_zakup(new TProdukt1("zeszyt", 2.0),1,1);
Wstaw_zakup(new TProdukt1("zeszyt", 2.0),2,1);
Wstaw_zakup(new TProdukt1("zeszyt", 2.0),3,1);
Wstaw_zakup(new TProdukt2("olowek",0.80,7),4,1);
Wstaw_zakup(new TProdukt2("pioro",4.80,20),5,1);
Wstaw_zakup(new TProdukt2("pioro",4.80,20),6,1);
Wstaw_zakup(new TProdukt3("olowek1",0.80,20),4,1);
Wstaw_zakup(new TProdukt3("pioro1",4.80,10),5,1);
Wstaw_zakup(new TProdukt3("pioro1",4.80,10),6,1);
Wstaw_zakup(new TProdukt1("zeszyt", 2.0),1,2);
Wstaw_zakup(new TProdukt3("pioro1",4.80,10),6,2);
22: 26
Modified
Insert


main1.cpp
main1.cpp
KOL2.h
PRODUKT1.CPP
PRODUKT2.CPP
produkt3.cpp
RACHUNEK.CPP
ZAKUP.CPP
cout<<rachunki<<endl; //cout<<rachunki.toString();
produkty.Usun_kolekcje();
rachunki.Usun_kolekcje();
cin.get();
}
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;
}
37: 28
Modified
Insert

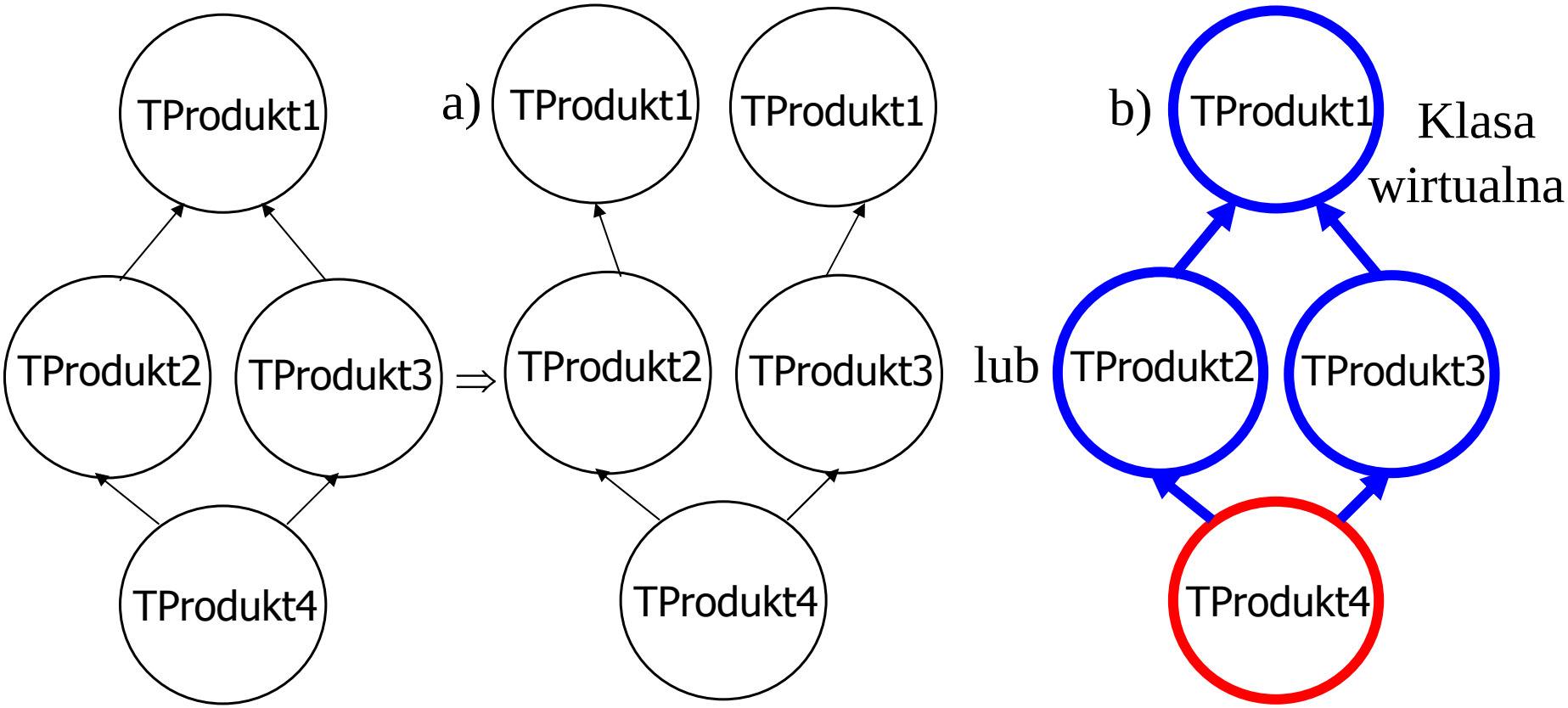

C:\Settings\dydaktyka\Programowanie_obiektowe\w9_1\lab9_1.exe
Nazwa: zeszyt, Cena detaliczna: 2
Nazwa: olowek, Cena detaliczna: 0.856, Podatek: 7
Nazwa: pioro, Cena detaliczna: 5.76, Podatek: 20
Nazwa: olowek1, Cena detaliczna: 0.64, Promocja: 20
Nazwa: pioro1, Cena detaliczna: 4.32, Promocja: 10
Rachunek : 1
Wartosc rachunku: 0
Rachunek : 2
Wartosc rachunku: 0


C:\Settings\dydaktyka\Programowanie_obiektowe\w9_1\lab9_1.exe
Rachunek : 1
Nazwa: zeszyt, Cena detaliczna: 2
Ilosc produktu: 6, Wartosc zakupu: 12
Nazwa: olowek, Cena detaliczna: 0.856, Podatek: 7
Ilosc produktu: 4, Wartosc zakupu: 3.42
Nazwa: pioro, Cena detaliczna: 5.76, Podatek: 20
Ilosc produktu: 11, Wartosc zakupu: 63.4
Nazwa: olowek1, Cena detaliczna: 0.64, Promocja: 20
Ilosc produktu: 4, Wartosc zakupu: 2.56
Nazwa: pioro1, Cena detaliczna: 4.32, Promocja: 10
Ilosc produktu: 11, Wartosc zakupu: 47.5
Wartosc rachunku: 129
Rachunek : 2
Nazwa: zeszyt, Cena detaliczna: 2
Ilosc produktu: 1, Wartosc zakupu: 2
Nazwa: pioro1, Cena detaliczna: 4.32, Promocja: 10
Ilosc produktu: 6, Wartosc zakupu: 25.9
Wartosc rachunku: 27.9

Dziedziczenie wielobazowe- zastosowanie

1. **Tworzenie rachunku – nowy typ klasy typu TProdukt3, pochodny klasy typu TProdukt1 – dziedziczenie jednobazowe**
2. **Dziedziczenie wielobazowe z powtórzeniami, listy konstruktorów**

Dziedziczenie wielobazowe z powtórzeniami



Dziedziczenie z klasami wirtualnymi, listy konstruktorów

Jeśli część z klas bazowych z listy dziedziczenia wielobazowego dziedziczą od wspólnej klasy bazowej występuje dziedziczenie **wielokrotne składowych tej wspólnej klasy.**

Można to zjawisko wyeliminować przez deklarowanie tej klasy w liście dziedziczenia każdej z klas bazowych jako **klasy wirtualnej.**

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
        {.................};
```

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 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

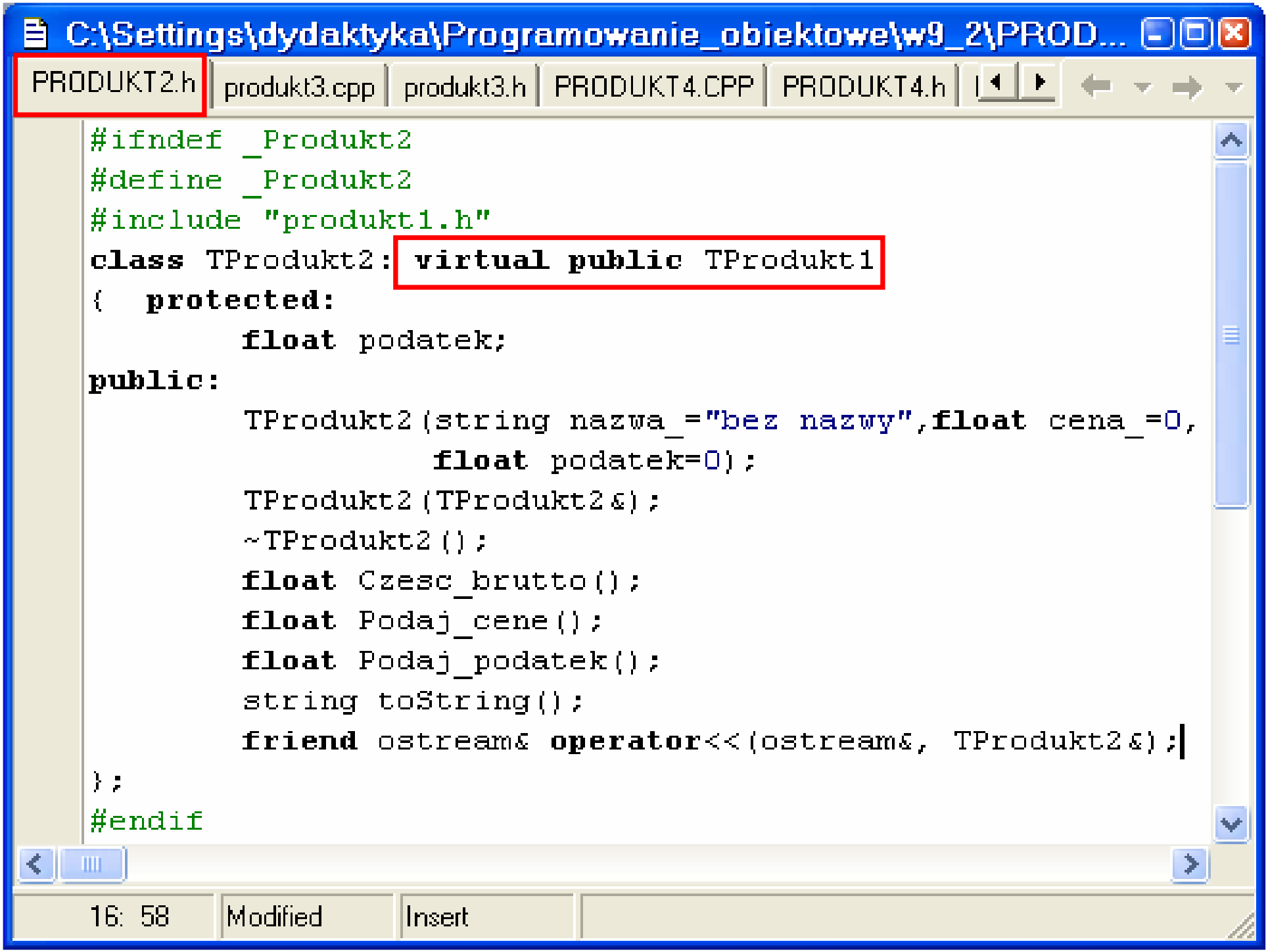
```
TProdukt4::TProdukt4(string nazwa_, float cena_,
                     float podatek_, float promocja_):
    TProdukt2(nazwa_,cena_,podatek_),
    TProdukt3(nazwa_,cena_,promocja_)
    {  }
TProdukt4::TProdukt4(TProdukt4& p):
    TProdukt2(p),
    TProdukt3(p)
    {  }
```

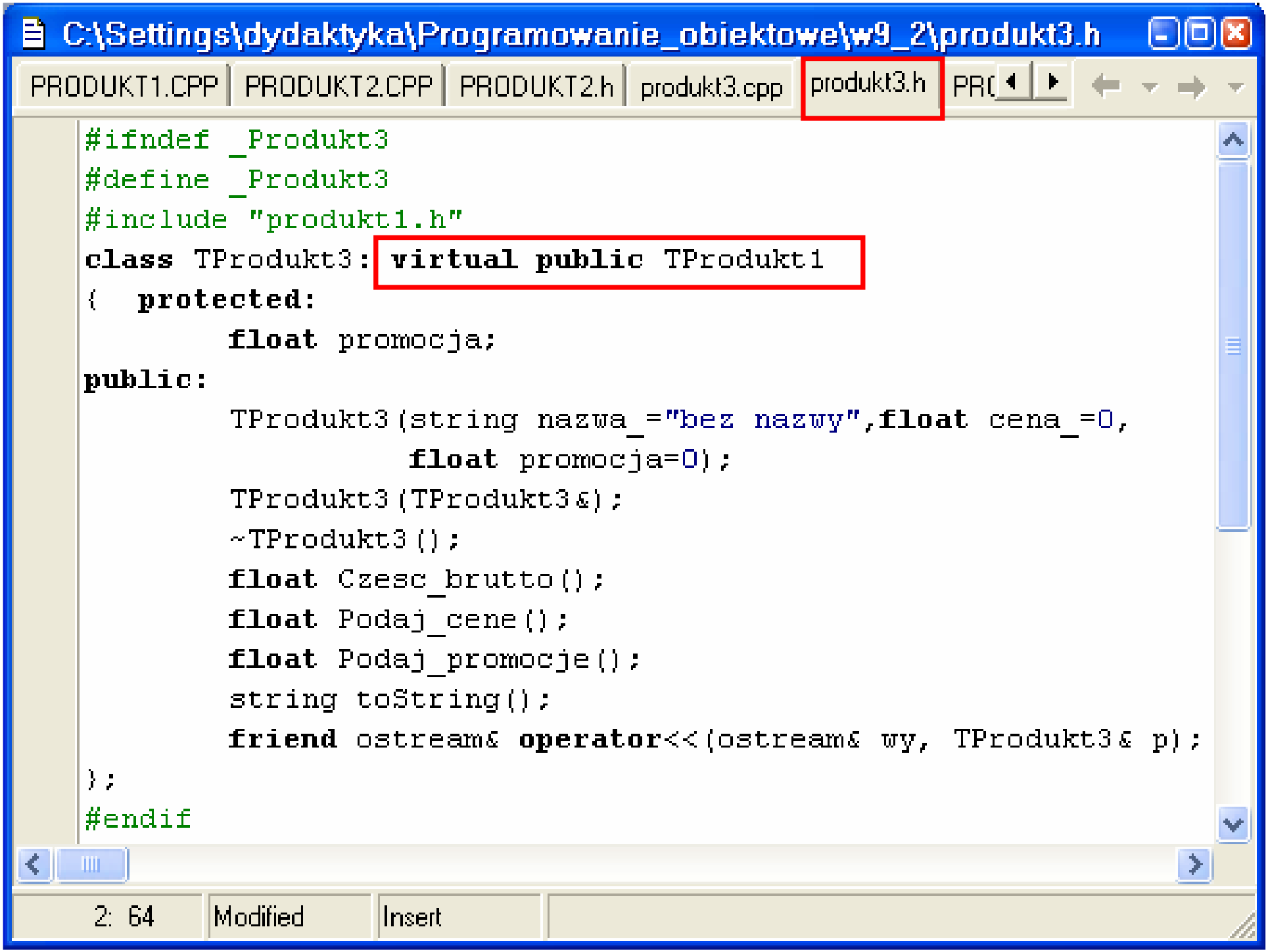
Dziedziczenie z klasami wirtualnymi, listy konstruktorów

1. Słowo **virtual** może być umieszczone przed lub za słowem **public** (lub **private**).
2. W zwykłym wielobazowym dziedziczeniu konstruktor klasy dziedziczącej wielobazowo przekazuje dane do konstruktorów klas z listy dziedziczenia, które następnie przekazują dane do konstruktora swojej klasy bazowej niezależnie - nawet jeśli jest to ta sama klasa.
3. Kolejność wywoływania konstruktorów klas jest taka sama, jak kolejność umieszczenia tych klas w liście dziedziczenia (zazwyczaj od lewej do prawej strony).
4. W przypadku klasy wirtualnej, w nagłówku konstruktora klasy pochodnej, dziedziczącej wielobazowo trzeba wymienić argumenty przeznaczone dla konstruktora klasy wirtualnej. Stąd w liście konstruktorów oprócz konstruktorów klas bazowych ***musi być wymieniony konstruktor klasy* wirtualnej, który jest wywoływany zawsze jako pierwszy i tylko raz**, gdyż ignorowane są wywołania tego konstruktora przez konstruktory klas bazowych.
5. W zwykłym i wirtualnym dziedziczeniu w przypadku zdefiniowania konstruktorów domniemanych lub w przypadku braku jawnych konstruktorów stosowanie list parametrów nie jest obowiązkowe.
6. Wywołania metod klasy wirtualnej przez obiekty klasy dziedziczącej wielobazowo jest jednoznaczne.

Dziedziczenie wielobazowe- zastosowanie

1. **Tworzenie rachunku – nowy typ klasy typu TProdukt3, pochodny klasy typu TProdukt1 – dziedziczenie jednobazowe**
2. **Dziedziczenie wielobazowe z powtórzeniami, listy konstruktorów**
3. **Tworzenie rachunku – nowy typ klasy typu TProdukt4, pochodny klas typu TProdukt2 i TProdukt3 – dziedziczenie dwubazowe**


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

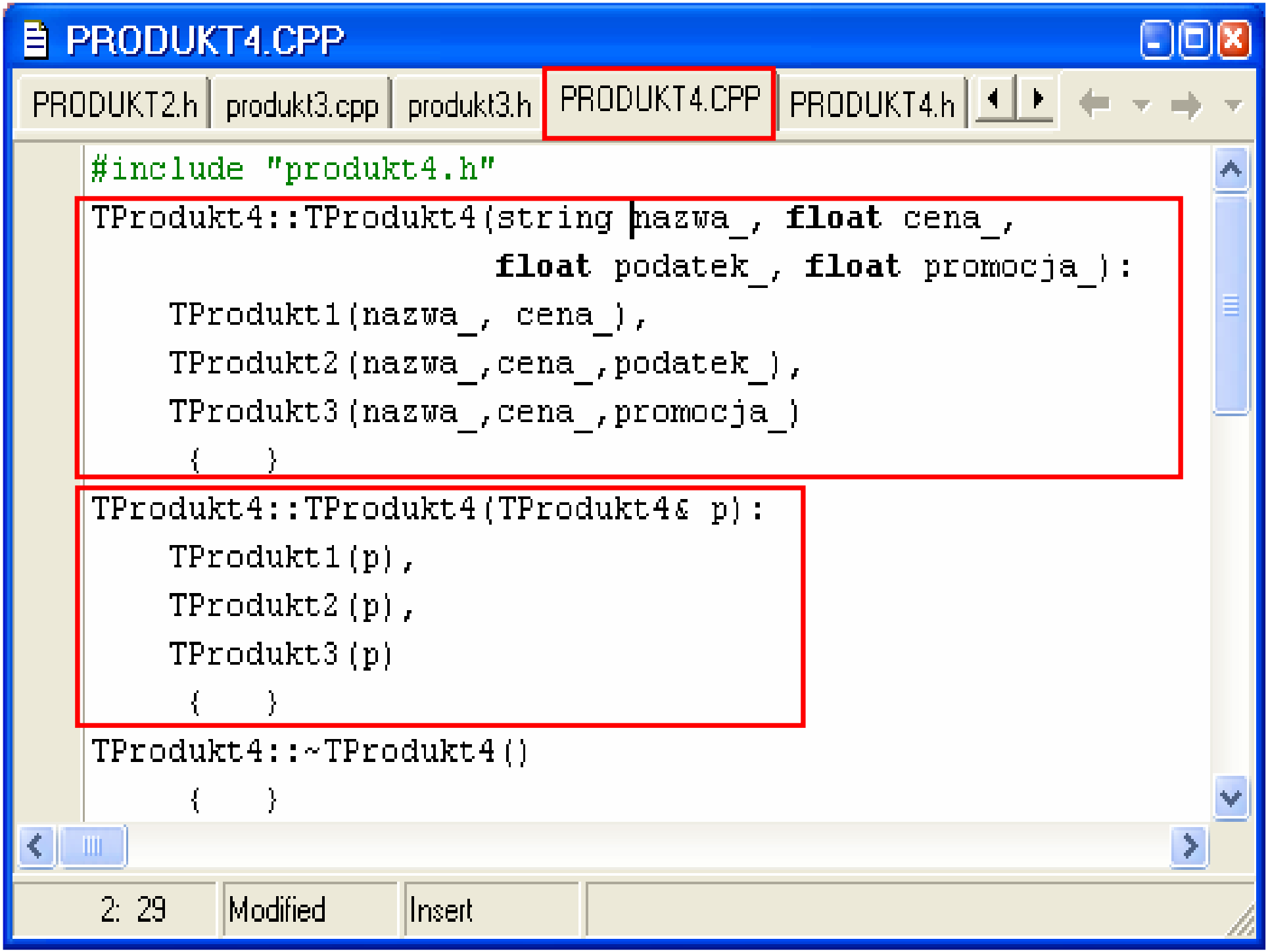

C:\Settings\dydaktyka\Programowanie_obiektowe\w9_2\produkt3.h
PRODUKT1.CPP
PRODUKT2.CPP
PRODUKT2.h
produkt3.cpp
produkt3.h
#ifndef _Produkt3
#define _Produkt3
#include "produkt1.h"
class TProdukt3: virtual public TProdukt1
{ protected:
float promocja;
public:
TProdukt3(string nazwa_="bez nazwy",float 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

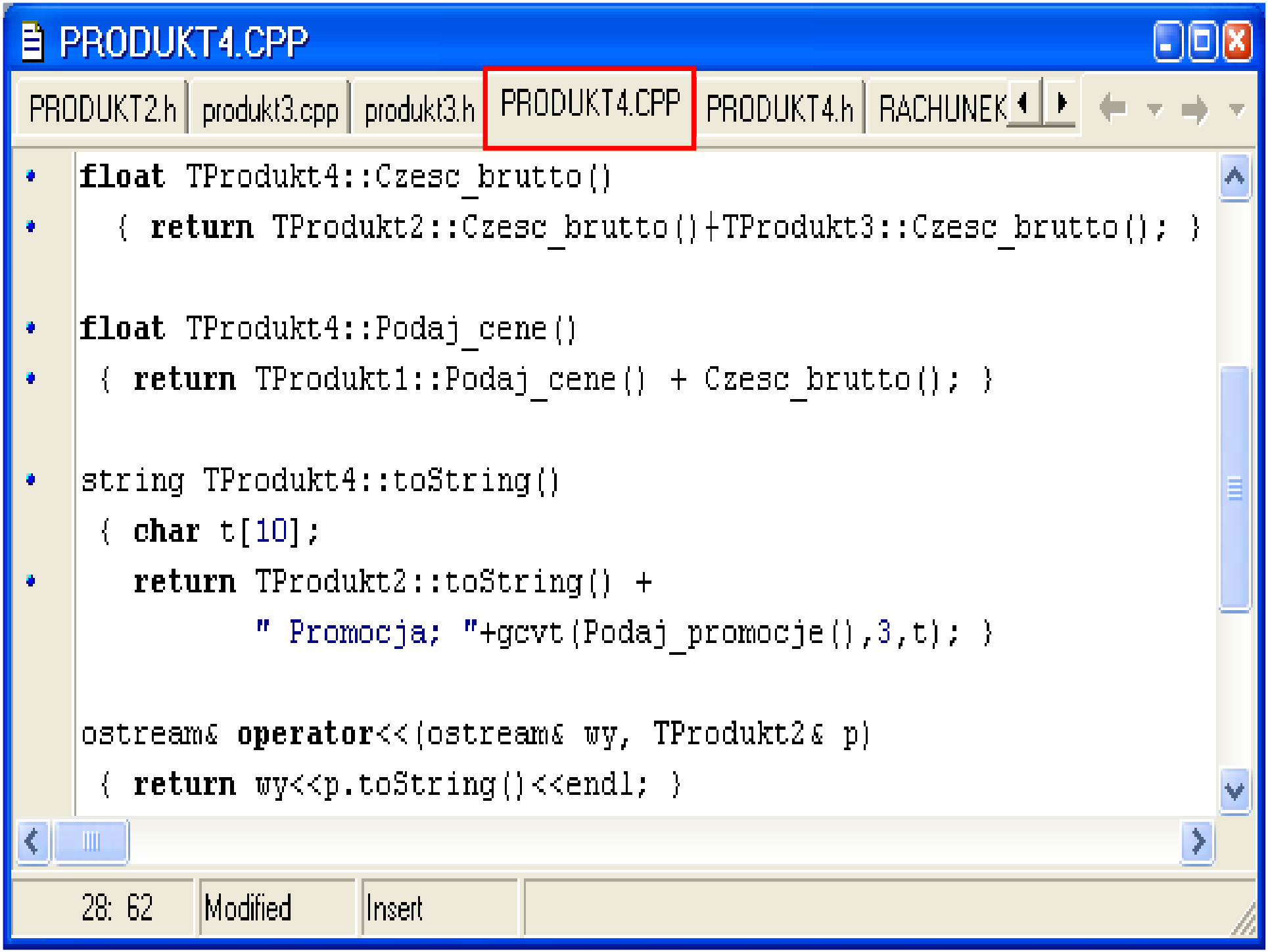
TProdukt4 dziedziczy od dwóch klas TProdukt2, TProdukt2 i pośrednio od TProdukt1

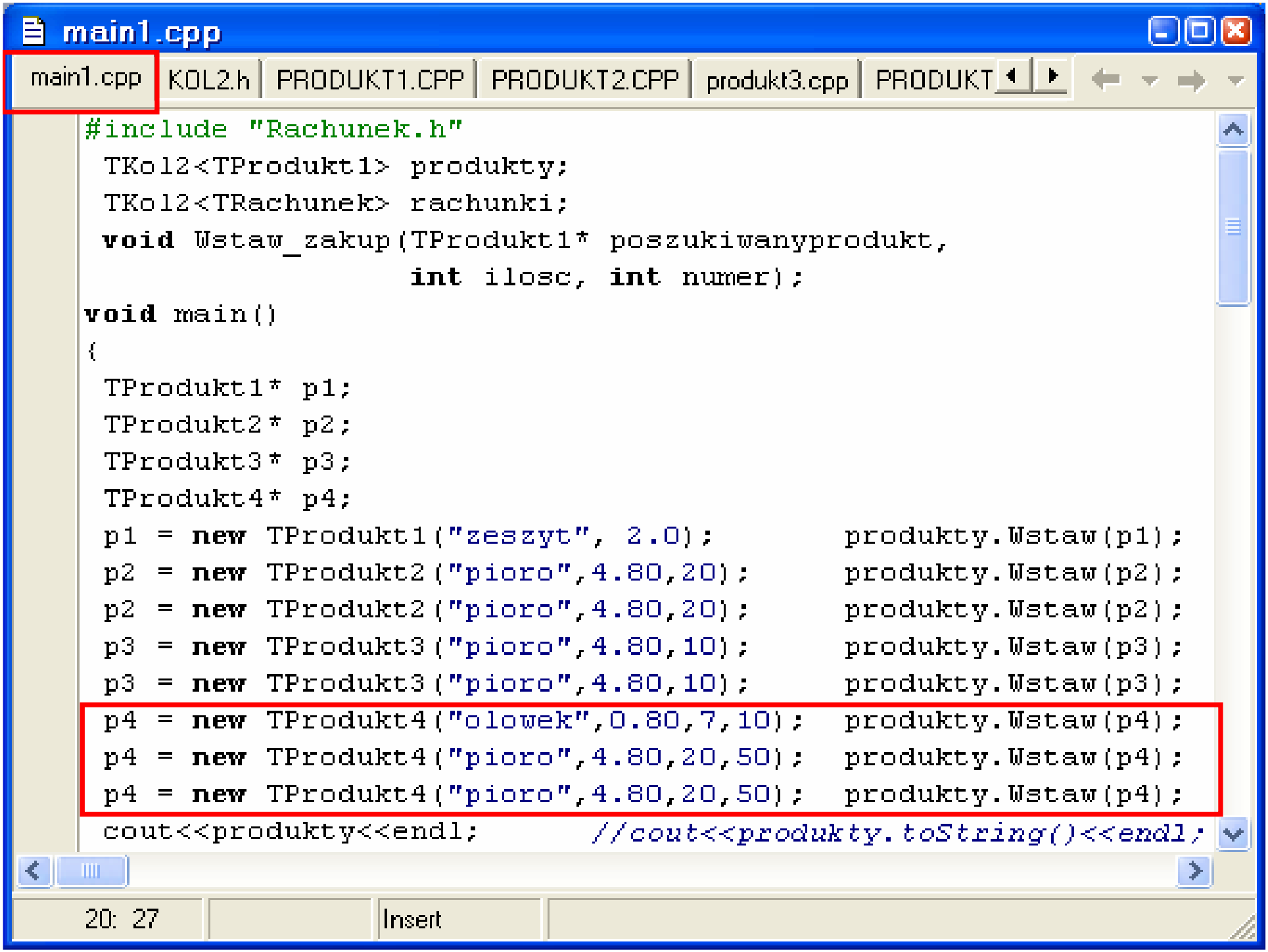
PRODUKT2.h | produkt3.cpp | produkt3.h | PRODUKT4.CPP | PRODUKT4.h

```
#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
PRODUKT2.h
produkt3.cpp
produkt3.h
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(nazwa_,cena_,promocja_)
{ }
TProdukt4::TProdukt4(TProdukt4& p):
TProdukt1(p),
TProdukt2(p),
TProdukt3(p)
{ }
TProdukt4::~TProdukt4()
{ }
2: 29
Modified
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

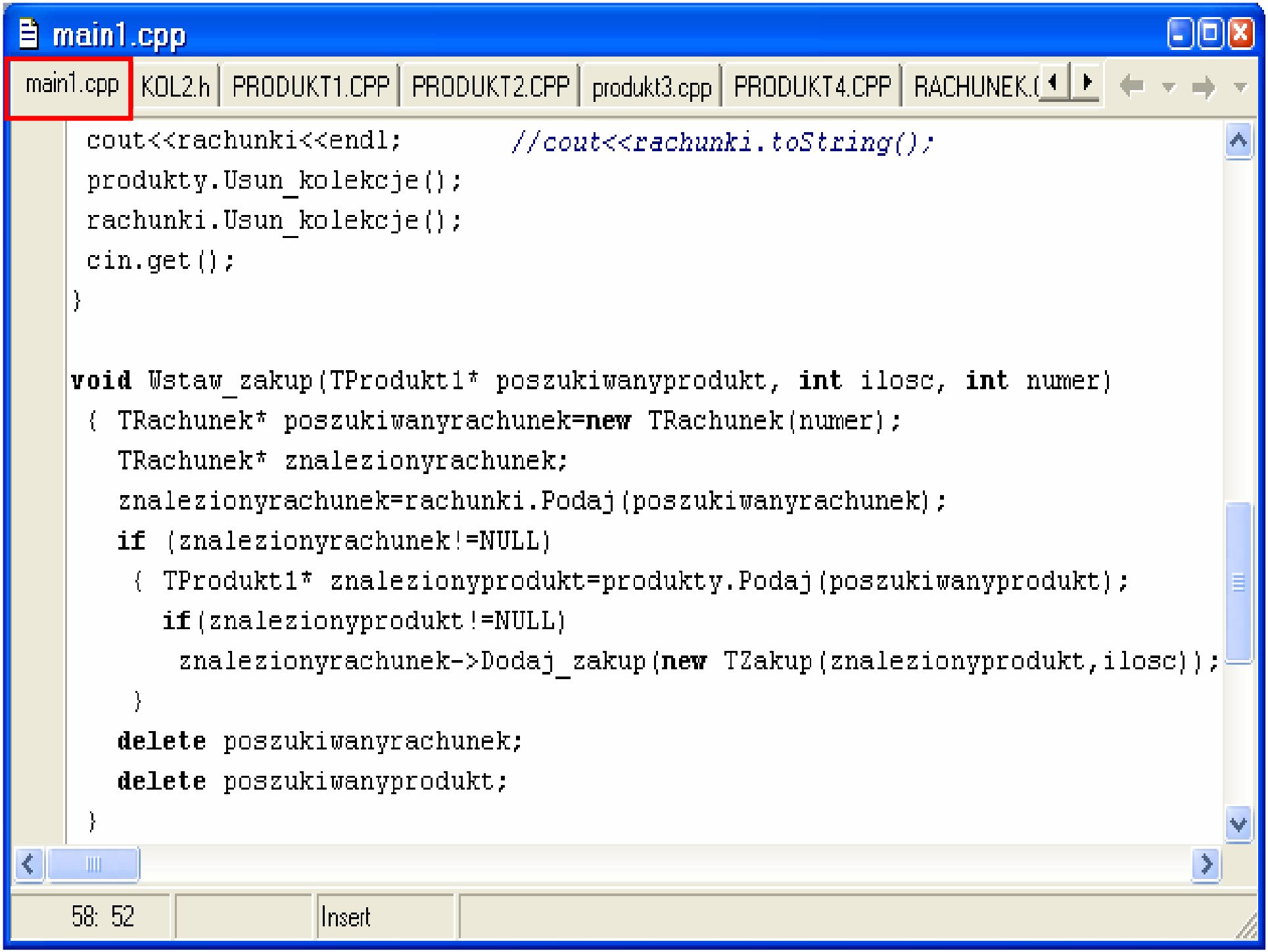

main1.cpp
main1.cpp
KOL2.h
PRODUKT1.CPP
PRODUKT2.CPP
produkt3.cpp
PRODUKT
#include "Rachunek.h"
TKol2<TProdukt1> produkty;
TKol2<TRachunek> rachunki;
void Wstaw_zakup(TProdukt1* poszukiwanyprodukt,
int ilosc, int numer);
void main()
{
TProdukt1* p1;
TProdukt2* p2;
TProdukt3* p3;
TProdukt4* p4;
p1 = new 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;
20: 27
Insert

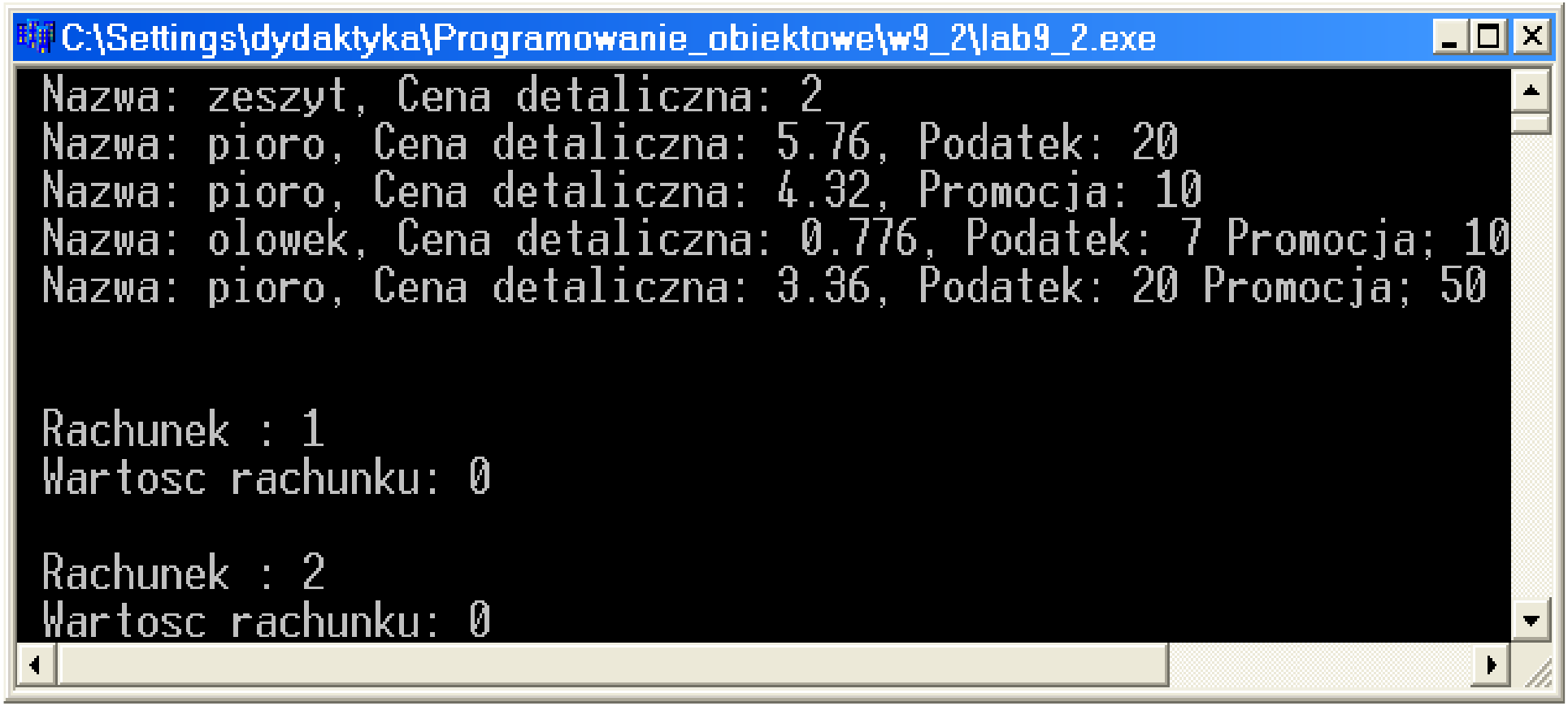
main1.cpp

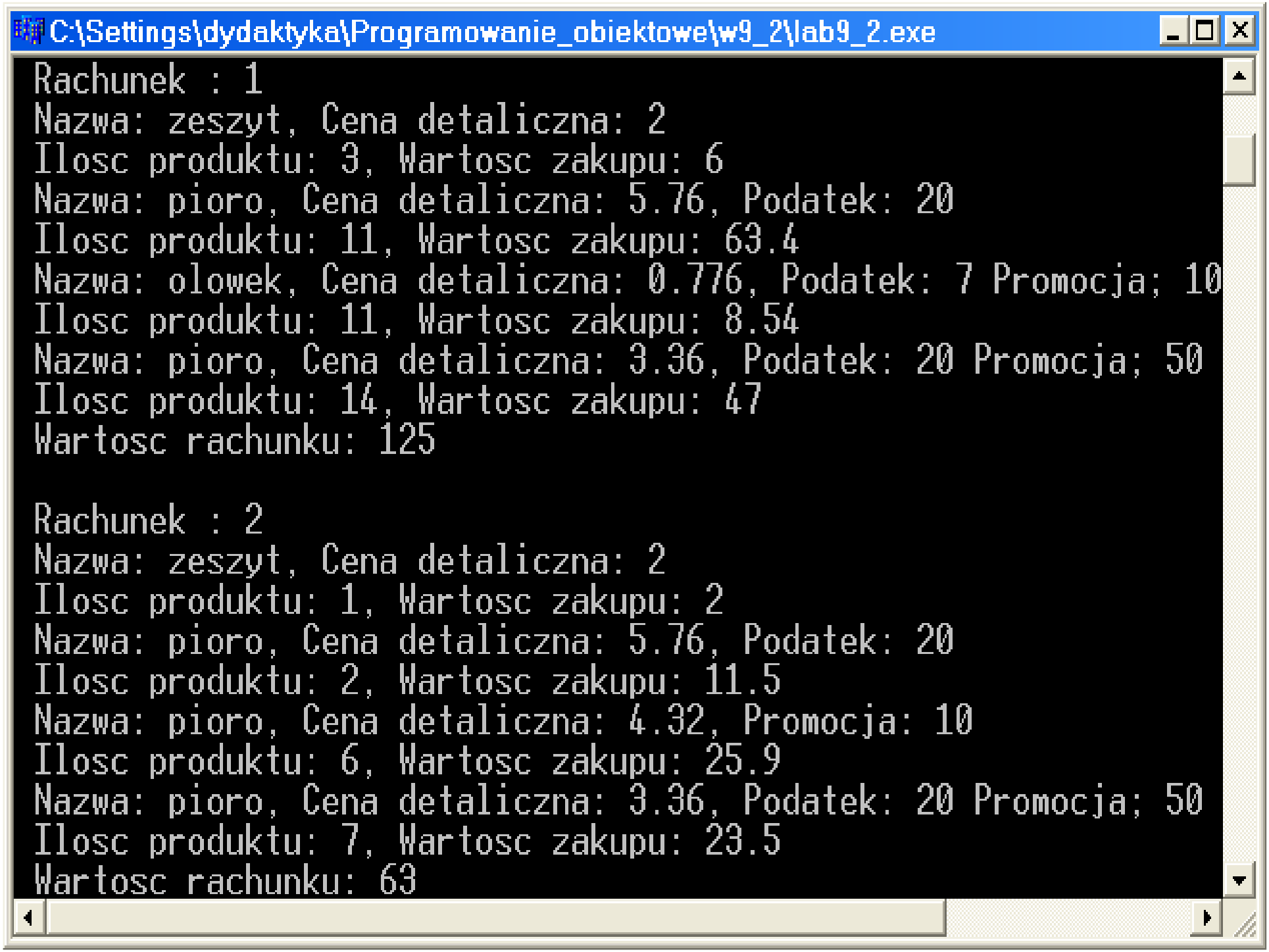
main1.cpp | PRODUKT1.CPP | PRODUKT2.CPP | produkt3.cpp | PRODUKT4.CPP | RACHUNEK.CPP | ZAKUP.CPP

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

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);
```

39: 34 | Insert


main1.cpp
main1.cpp
KOL2.h
PRODUKT1.CPP
PRODUKT2.CPP
produkt3.cpp
PRODUKT4.CPP
RACHUNEK.(
cout<<rachunki<<endl; //cout<<rachunki.toString();
produkty.Usun_kolekcje();
rachunki.Usun_kolekcje();
cin.get();
}
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;
}
58: 52
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: 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\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: 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: 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