Jacek Matulewski
http://www.fizyka.umk.pl/~jacek/

Szablony 2
Inteligentne wskaźniki

Wersja α

Toruń, 3 listopada 2006

Najnowsza wersja dokumentu dostępna pod adresem
http://www.fizyka.umk.pl/~jacek/dydaktyka/cpp/cpp-szablony2.pdf

Źródła programów z tego skryptu (C++Builder 6 i Visual C++ 2005):
http://www.fizyka.umk.pl/~jacek/dydaktyka/cpp/cpp-szablony2.zip

Inteligentne wskaźniki

1. Zbudujmy prawie pustą klasę informująca o wywoływanych metodach (w szczególności konstruktorach i destruktorze). Dzięki niej będziemy mogli śledzić działanie inteligentnego wskaźnika.

```
class TKlasa
{
    private:
        AnsiString stan;
    public:
        TKlasa(AnsiString stan="Domyslny")
            :stan(stan)
        {
            ShowMessage("TKlasa::TKlasa, stan="+stan);
        }
        TKlasa(const TKlasa& Oryginal)
        {
            stan=Oryginal.stan;
            ShowMessage("TKlasa::TKlasa, konstruktor copy");
        }
        ~TKlasa(){ShowMessage("TKlasa::~TKlasa, stan="+stan);}
        void UstawStan(AnsiString stan)
        {
            ShowMessage("TKlasa::UstawStan, stary stan="+this->stan+", nowy
stan="+stan);
            this->stan=stan;
        }
        void PokazStan(){ShowMessage("TKlasa::PokazStan, stan="+stan);}
};
```

Możemy przetestować jej działanie:

```
void __fastcall TForm1::Button0Click(TObject *Sender)
{
    TKlasa domyslny;
    TKlasa a("a");
    TKlasa b(a);
    TKlasa c=a;

    TKlasa* d=new TKlasa("dynamicznie");
    TKlasa* e=d;
    delete d;
}
```

2. Stwórzmy szablon inteligentnego wskaźnika

```
template<typename T> class InteligentnyWskaznik
{
    private:
        T* wskaznik; //opakowywany wskaznik
        T* get() //ze sprawdzaniem, czy nie mamy wskaznika pustego
        {
            if(wskaznik==NULL) throw Exception("Pusty wskaźnik!");
            return wskaznik;
        };
    public:
        explicit InteligentnyWskaznik(T*
wskaznik):wskaznik(wskaznik){ShowMessage("InteligentnyWskaznik::InteligentnyWskaznik");}
//explicit nie pozwala na niejawne konwersje argumentow konstruktora
        ~InteligentnyWskaznik()
        {
            if (wskaznik==NULL)
            {
                ShowMessage("InteligentnyWskaznik::~InteligentnyWskaznik, wskaznik jest
rowny NULL");

            }
            else
            {
                ShowMessage("InteligentnyWskaznik::~InteligentnyWskaznik, usuwam
obiekt");
                delete wskaznik; //czas zycia jak na stosie
                wskaznik=NULL; //o to zwykle sie niestety nie dba
            }
        }
        T* operator ->() {return get();}
        T& operator *() {return *get();}
};
```

Uwagi:

1) W momencie usunięcia wskaźnika (nawet jeżeli jego wartość zostanie on skopiowana) obiekt zostanie usunięty. Oznacza to, że pomimo utworzenia obiektu na stercie jego zakres jest taki, jak przy tworzeniu na stosie.

2) Inteligentny wskaźnik sprawdza, czy wskaźnik ma wartość `NULL` przed udostępnieniem go na zewnątrz – to pozwoli uniknąć odwołania do elementów składowych pustych wskaźników, ale należy pamiętać, że wskaźniki nie są czyszczone automatycznie.

3. Sprawdźmy czy działa:

```
//DOBRZE!!!
void __fastcall TForm1::Button1Click(TObject *Sender)
{
    try
    {
        InteligentnyWskaznik<TKlasa> wsk(new TKlasa("A"));
```

```
        wsk->UstawStan("B");
        (*wsk).PokazStan();
    }
    catch(Exception& exc)
    {
        ShowMessage("Wyjątek: "+exc.Message);
    }
}

//BŁĄD - pojawi się wyjątek!!!
void __fastcall TForm1::Button2Click(TObject *Sender)
{
    try
    {
        TKlasa* pustyWskaznik=NULL;
        InteligentnyWskaznik<TKlasa> wsk(pustyWskaznik);
        wsk->UstawStan("B");
        (*wsk).PokazStan();
    }
    catch(Exception& exc)
    {
        ShowMessage("Wyjątek: "+exc.Message);
    }
}
```

4. Analogiczne działania na klasie `TButton`:

```
//DOBRZE!!!
void __fastcall TForm1::Button5Click(TObject *Sender)
{
    try
    {
        InteligentnyWskaznik<TButton> wsk(new TButton(this));
        wsk->Parent=this;
        wsk->Width=100;
        (*wsk).Caption="Wskaznik z IQ";
    }
    catch(Exception& exc)
    {
        ShowMessage("Wyjątek: "+exc.Message);
    }
}

//BŁĄD - pojawi się wyjątek!!!
void __fastcall TForm1::Button6Click(TObject *Sender)
{
    try
    {
```

```
        TButton* pustyWskaznik=new TButton(this);
        delete pustyWskaznik; //!!! samo delete nie czysci wskaznika -> Inteligentny wskaznik nie zadziala
        pustyWskaznik=NULL;
        InteligentnyWskaznik<TButton> wsk(pustyWskaznik);
        wsk->Parent=this;
        wsk->Width=100;
        (*wsk).Caption="Wskaznik z IQ";
    }
    catch(Exception& exc)
    {
        ShowMessage("Wyjątek: "+exc.Message);
    }
}
```

5. Działa także dla typów wbudowanych:

```
void __fastcall TForm1::Button3Click(TObject *Sender)
{
    InteligentnyWskaznik<double> wsk(new double(1.0));
    *wsk=2.0;
}
```

6. W tej chwili kopiowanie inteligentnego wskaźnika odbywa się za pomocą domyślnego konstruktora copy – nie jest tworzony nowy obiekt, a jedynie nowy wskaźnik do istniejącego. Sprawdźmy to w następujących testach:

```
void __fastcall TForm1::Button4Click(TObject *Sender)
{
    //Sprawdzic dzialanie ze zdefiniowanym konstruktorem copy i bez niego
    InteligentnyWskaznik<TKlasa> wsk(new TKlasa("A"));
    InteligentnyWskaznik<TKlasa> kopia_wsk(wsk);
    wsk->UstawStan("A-oryginal");
    kopia_wsk->UstawStan("A-kopia");
    (*wsk).PokazStan();
    (*kopia_wsk).PokazStan();
}
```

Po usunięciu pierwszego obiektu (nawet po wyczyszczeniu jego wskaźnika do stanu NULL) drugi wskaźnik będzie i tak pokazywał adres nieistniejącego już obiektu. Ta wersja inteligentnego wskaźnika nie zapobiega więc w zasadzie dzikim wskaźnikom.

7. Moglibyśmy to zmienić definiując własny konstruktor copy:

```
InteligentnyWskaznik(const InteligentnyWskaznik& Oryginal)
{
    ShowMessage("InteligentnyWskaznik::InteligentnyWskaznik, konstruktor copy");
    wskaznik=new T(*Oryginal.operator->()); //tworzy nowy obiekt
}
```

8. Inteligentny wskaźnik czyści (przypisuje makro NULL) przechowywany wskaźnik do usuwanych obiektów:

```
void __fastcall TForm1::Button7Click(TObject *Sender)
{
    InteligentnyWskaznik<TKlasa> wsk(new TKlasa("A"));
```

```
    delete &wsk; //tu wskaznik jest czyszczony, ale operator wymaga adresu
inteligentnego wskaznika (tu widac, ze inteligentny wsk. nie jest prawdziwym
wskaznikiem)

    InteligentnyWskaznik<TKlasa> kopia_wsk(wsk);

    wsk->UstawStan("A-oryginal");

    kopia_wsk->UstawStan("A-kopia");

    (*wsk).PokazStan();

    (*kopia_wsk).PokazStan();

}
```