


## C++/CLI LAB2 – Referencia osztály, operátorok, ToString

Új projekt:  WindowsForm  
Windows Form MOGI GP tárgyhoz -> LAB2

Felület:

2 db TextBox

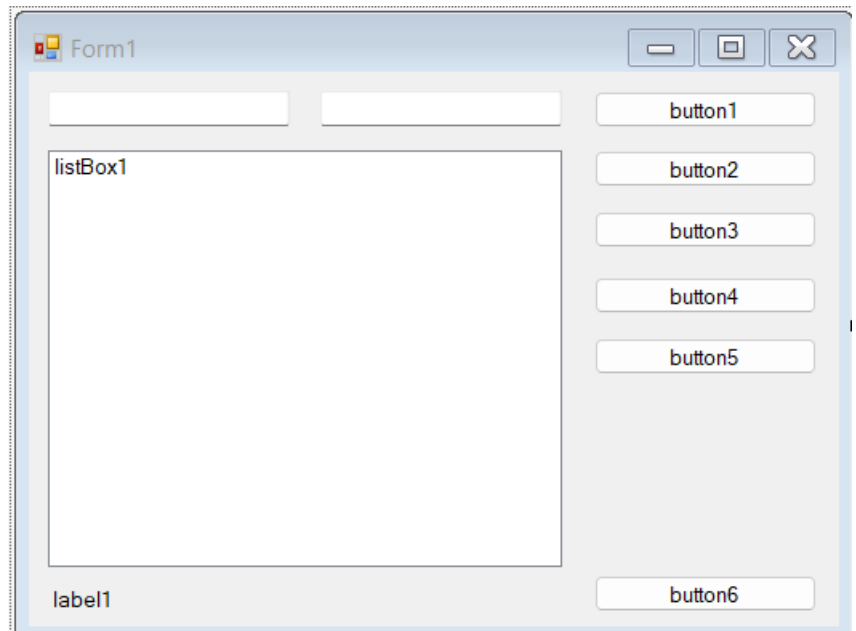
1 db ListBox

6 db Button

1 db Label

Ismeretek:

- referencia osztály sablon alapján
- property-k
- operátorok túlterhelése
- ToString túlterhelése
- try, catch hibakezelés



 Form1.h

```
using namespace System::Collections::Generic; // lista - List<> osztály
```

```
// sablonnal létrehozott referencia osztály
// sablon típusa T, létrehozáskor töltődik ki
template <class T>
ref class Complex {
private:
    // valós és képzetes rész private adattagja T típussal
    T re, im;
public:
    //konstruktorok
    Complex() : im(0), re(0) {};
    Complex(T re, T im) : im(im), re(re) {};
    // írható/olvasható property
    // valós rész
    property T Re {
        T get() { return re; }
        void set(T re) { this->re = re; }
    }
}
```

```

    // írható/olvasható property
    // képzetes rész
    property T Im {
        T get() { return im; }
        void set(T im) { this->im = im; }
    }
    // az ősosztályból származó ToString() metódus felüldefiniálása
    // az override kulcsszót ki kell tenni
    String^ ToString() override {
        String^ s = gcnew String("");
        // ha nulla
        if (re == 0 && im == 0) s = "0";
        // ha van valós rész
        if (re != 0) s = Convert::ToString(re);
        // ha van mindkettő és im +
        if (re != 0 && im > 0) s += "+";
        // ha im -
        if (im < 0) s += "-";
        // im abszolút értéke
        if (im != 0) s += Convert::ToString(Math::Abs(im)) + "i";
        return s;
    }
    // + operátor, komplex számok összege
    Complex<T>^ operator + (Complex<T>^ cm) {
        return gcnew Complex<T>(re + cm->Re, im + cm->Im);
    }
    // * operátor, komplex számok szorzata
    Complex<T>^ operator * (Complex<T>^ cm) {
        T a = re, b = im, c = cm->Re, d = cm->Im;
        return gcnew Complex<T>(a * c - b * d, b * c + a * d);
    }
};

// globális változók
// addig nem fordul le, amíg nem lesz példányosítva
Complex<double>^ c; // = gcnew Complex<double>();
List<Complex<double>^>^ lista = gcnew List<Complex<double>^>(); //üres lista

// ablak betöltődése
private: System::Void Form1_Load(System::Object^ sender, System::EventArgs^ e) {
    Text = "Komplex példa";
    button1->Text = "Hozzáad";
    button2->Text = "Összeadás";
    button3->Text = "Szorzás";
    button4->Text = "Lista törlése";
    button5->Text = "Elem törlése";
    button6->Text = "Kilépés";
    textBox1->Text = "0";
    textBox2->Text = "1";
    listBox1->Items->Clear();
    label1->Text = "";
    ShowButtons(); // gombok állapotainak frissítése
}

// Komplex szám létrehozás és hozzáadása a listákhoz
private: System::Void button1_Click(System::Object^ sender, System::EventArgs^ e) {
    // hiba kezelő blokk - szöveg szám konverzió miatt
    try {
        c = gcnew Complex<double>(); // komplex szám példányosítása
        c->Re = Convert::ToDouble(textBox1->Text); // adattagok felöltése porperty-n
        // keresztül
    }
}

```

```

        c->Im = Convert::ToDouble(textBox2->Text);
        lista->Add(c); // adat hozzáadása a listához
        listBox1->Items->Add(c->ToString()); // komplex szám kiíratása a ToString
        // felüldefiniálásával
    }
    catch (Exception^ e) {
        MessageBox::Show("Message: " + e->Message, "Hiba"); // hiba lépett fel
    }
    label1->Text = "";
    ShowButtons(); // gombok állapotainak frissítése
}

// komplex számok összege
private: System::Void button2_Click(System::Object^ sender, System::EventArgs^ e) {
    Complex<double>^ szum = gcnew Complex<double>();
    for (int i = 0; i < lista->Count; i++)
        szum = szum + lista[i];
    label1->Text = "A komplex számok összege: " + szum->ToString();
}

// komplex számok szorzata
private: System::Void button3_Click(System::Object^ sender, System::EventArgs^ e) {
    Complex<double>^ mult = lista[0]; // az első elemmel kezdjük
    for (int i = 1; i < lista->Count; i++) // a másodiktól folytatjuk
        mult = mult * lista[i];
    label1->Text = "A komplex számok szorzata: " + mult->ToString();
}

// listák törlése
private: System::Void button4_Click(System::Object^ sender, System::EventArgs^ e) {
    lista->Clear();
    listBox1->Items->Clear();
    ShowButtons(); // gombok állapotainak frissítése
}

// adott elem törlése
private: System::Void button5_Click(System::Object^ sender, System::EventArgs^ e) {
    int index = listBox1->SelectedIndex; // kijelölt indexű elem, ha nincs kijelölve semmi,
    // akkor - 1
    if (index != -1) { // ha volt kijelölve elem
        listBox1->Items->RemoveAt(index); // adott indexű elem törlése
        lista->RemoveAt(index); // adott indexű elem törlése
    }
    ShowButtons(); // gombok állapotainak frissítése
}

// ablak bezárása
private: System::Void button6_Click(System::Object^ sender, System::EventArgs^ e) {
    Close();
}

// bezárás előtt még meggondolhatjuk magunkat
private: System::Void Form1_FormClosing(System::Object^ sender,
System::Windows::Forms::FormClosingEventArgs^ e) {
    // dialógus ablak válasza
    System::Windows::Forms::DialogResult valasz;
    // MessageBox elem megjelenítése (Show) Yes és No gombokkal
    valasz = MessageBox::Show("Már kész is haza?", "Ajajjj", MessageBoxButtons::YesNo);
    // ha Nem-et nyomtunk, akkor nem engedi bezárni az ablakot
    if (valasz == System::Windows::Forms::DialogResult::No) e->Cancel = true;
}

```

```
// gombok láthatóságának vizsgálata
void ShowButtons() {
    if (lista->Count == 0) {
        button2->Enabled = false;
        button3->Enabled = false;
        button4->Enabled = false;
        button5->Enabled = false;
        label1->Text = "";
    } if (lista->Count > 0) {
        button4->Enabled = true;
        button5->Enabled = true;
    } if (lista->Count == 1) {
        button2->Enabled = false;
        button3->Enabled = false;
    } if (lista->Count > 1) {
        button2->Enabled = true;
        button3->Enabled = true;
    }
}
```