Kalkulator modulo

Dokumentacja

Mateusz Białecki, student Informatyki

na wydziale Matematyki Stosowanej

1. Wstęp, instrukcja
2. Zmienne
3. Obsługa przycisków i pól
    a. Przyciski numeryczne
    b. Przyciski akcji
    c. Walidacja w polach tekstowych
4. Obsługa wyjątków
5. Kontakt

Wstęp, instrukcja

Projekt zakłada stworzenie programu pomocnego w arytmetyce modularnej. Na wejściu użytkownik:

1) określa zbiór modulo, na jakim chce działać (domyślnie 10),
2) podaje liczbę,
3) określa, jaką operację chce wykonać (+, -, *, /),
4) podaje drugą liczbę,
5) Naciska przycisk ‘=’ i otrzymuje wynik.

Na wyjściu użytkownik otrzymuje liczbę całkowitą dodatnią, wpisaną w pole tekstowe eqInput. Następnie użytkownik może wyczyścić środowisko przyciskiem ‘C’ i zadać nowe działanie, lub od razu rozpocząć wpisywanie nowej liczby.

Osoba korzystająca z programu może również korzystać ze swojej fizycznej klawiatury numerycznej (przyciski 1-9, +, -, /, * oraz Enter, który odpowiada przyciskowi ‘=’).

Zbiór modulo może być zmieniany w dowolnym momencie działania programu i następny wynik będzie podany z jego uwzględnieniem.

Zmienne

W programie wykorzystuje się następujące zmienne:

- Dwie zmienne typu long long int (num1 oraz num2) do przechowywania wprowadzonych przez użytkownika liczb,
- zmienna typu String^ (action) do przechowywania operacji, jaką chce wykonać użytkownik (+, -, * lub /),
- Dwie zmienne typu bool (pełnią role pomocnych flag)

```
long long int num1 = 0;
long long int num2 = 0;
String^ action = "none";
bool to_clear = false;
bool equals_just_pressed = true;
```

- Następnie deklarowane są elementy formularza:
- 10 przycisków numerycznych,
- 4 przyciski akcji (+, -, *, /),
- przycisk „równa się”,
- przycisk wyczyść,
- dwa elementy typu *label* (jeden by wyświetlić literę Z oznaczającą zbiór modulo z liczb całkowitych, drugi by wyświetlić aktualnie wykonywaną operację),
- dwa elementy typu Textbox, (jeden do określenia zbioru modulo [zInput], drugi do wprowadzania liczb [eqInput])

Obsługa

Obsługa przycisków numerycznych

Po kliknięciu przycisku numerycznego wywołana zostaje funkcja *button1_Click*. Funkcja ta obsługuje każdy przycisk numeryczny – konwertuje obiekt typu *Object^* na *Button^* co pozwala na pozyskanie wartości Text, czyli cyfry 0-9, która następnie dopisywana jest do eqInput. W razie potrzeby czyszczone jest pole do wpisywania liczby.

```
//Making one handler to 10 buttons 0-9
System::Void button1_Click(System::Object^ sender, System::EventArgs^ e) {
    //casting sender to button so i can access its values easily
    System::Windows::Forms::Button^ button = (System::Windows::Forms::Button^) sender;
    if (to_clear) clear();
    if(eqInput->TextLength < eqInput->MaxLength)
    {
        eqInput->Text += button->Text;
        equals_just_pressed = false;
    }
```

Zaimplementowano również obsługę z klawiatury numerycznej. Jeżeli aktualnie wybrane jest pole eqInput, naciśnięcie cyfry skutkuje wpisaniem jej do pola.

```
System::Void eqInput_KeyPress(System::Object^ sender, System::Windows::Forms::KeyPressEventArgs^ e) {
    char k = e->KeyChar;
    •
    •
    •

    //dont allow any characters expect for numbers (48 is '0' and 57 is '9' in ASCII)
    else if (k < 48 || k > 57)
    {
        e->Handled = true;
    }
```

Obsługa przycisków akcji

Po kliknięciu przycisku akcji wywołana zostaje funkcja *buttonAction_Click*. Funkcja ta obsługuje każdy przycisk akcji – konwertuje obiekt typu *Object^* na *Button^* co pozwala na pozyskanie wartości Text, czyli symbolu, przekazywanego do zmiennej *action*. W razie potrzeby czyszczone jest pole do wpisywania liczby.

Jeżeli label2 jest puste, czyli nie jest obecnie wykonywane jeszcze żadne działanie, a jedynie wpisano pierwszą z liczb, program sczytuje ją do num1, a do label2 wpisuje obecnie wykonywane działanie.

Jeżeli label2 nie jest puste, to oznacza, że użytkownik wpisał liczbę, kliknął akcję, wpisał drugą liczbę i znowu kliknął akcję. W tym wypadku liczony jest wynik i gotowa do wykonania jest już druga akcja, wystarczy wpisać drugą liczbę i zatwierdzić.

```
System::Void buttonAction_Click(System::Object^ sender, System::EventArgs^ e) {
    //casting sender to button so i can access its values easily
    System::Windows::Forms::Button^ button = (System::Windows::Forms::Button^) sender;
    try {
        if (to_clear) clear();
        if (label2->Text == "")
        {
            action = Convert::ToString(button->Text); // 'action' is +, -, * or /
            num1 = Convert::ToInt32(eqInput->Text);
            label2->Text = eqInput->Text;
            label2->Text += action;
            to_clear = true;
        }
        else {
            num2 = Convert::ToInt32(eqInput->Text);
            int result = resolve(num1, num2, action);
            action = Convert::ToString(button->Text);
            String^ result_string = Convert::ToString(result);
            eqInput->Text = result_string;
            label2->Text = result_string + action;
            to_clear = true;
            num1 = result;
        }
    }
```

Obsługa przycisku ‘=’

Po kliknięciu znaku równości program sczytuje wartość z pola eqInput i umieszcza ją w zmiennej num2, wykonuje zadaną operację i umieszcza wynik w polu tekstowym.

```
//Clicking "=" results in stashing number from eqInput to num 2 and
//perform an equation with num1 specified by string 'action' (+,-,/ or *)
System::Void buttonEquals_Click(System::Object^ sender, System::EventArgs^ e) {

    if ((action != "none" && equals_just_pressed == false))
    {
        try {
            equals_just_pressed = true;
            num2 = Convert::ToInt32(eqInput->Text);
            int result = resolve(num1, num2, action);
            String^ result_string = Convert::ToString(result);
            eqInput->Text = result_string;
            action = "none";
            label2->Text = "";
            to_clear = true;
        }
```

Walidacja w polach tekstowych

Jeżeli aktywne jest pole eqInput, użytkownik może do niego wpisać cyfry od 0-9. Może też nacisnąć +,-,*,/ lub enter (wywołuje metodę kliknięcia przycisku ‘=’). Może również użyć klawisza backspace. Inne przyciski nie są obsługiwane.

```
System::Void eqInput_KeyPress(System::Object^ sender, System::Windows::Forms::KeyPressEventArgs^ e) {
    char k = e->KeyChar;
    //allow backspace
    if (k == 8)
    {
        e->Handled = false;
    }
    //pressing Enter invokes buttonEquals click
    else if (k == 13) buttonEquals->PerformClick();
    //run action buttons from keyboard
    else if (k == '+' || k == '-' || k == '*' || k == '/' || k == '=')
    {
        switch (k)
        {
        case '+':
            buttonPlus->PerformClick();
            break;
        case '-':
            buttonMinus->PerformClick();
            break;
        case '*':
            buttonTimes->PerformClick();
            break;
        case '/':
            buttonDivided->PerformClick();
            break;
        case '=':
            buttonEquals->PerformClick();
            break;
        }
        e->Handled = true;
    }
    //dont allow any characters expect for numbers (48 is '0' and 57 is '9' in ASCII)
    else if (k < 48 || k > 57)
    {
        e->Handled = true;
    }
    else
    {
        if (to_clear) clear();
    }
        if (k!='=')equals_just_pressed = false;

};
```

Jeżeli aktywne jest pole zInput, użytkownik może jedynie wpisywać cyfry z klawiatury i używać przycisku backspace.

Obsługa wyjątków

Zaimplementowano obsługę wyjątków *DivideByZeroException^* oraz *FormatException^.*

Pierwszy wyjątek pojawia się, gdy użytkownik spróbuje wykonać operację dzielenia przez zero. Wtedy do pola tekstowego wpisywana jest fraza „Divide/0 !”, która uświadamia korzystającemu, jaką operację próbował przeprowadzić.

Drugi wyjątek pojawia się, gdy użytkownik w jakiś sposób wprowadzi niedozwolone znaki do pola tekstowego. W tym wypadku klikany jest przycisk C, by usunąć niedozwolone znaki ze zmiennych.

```
System::Void buttonAction_Click(System::Object^ sender, System::EventArgs^ e) {
    //casting sender to button so i can access its values easily
    System::Windows::Forms::Button^ button = (System::Windows::Forms::Button^) sender;
    try {
        ●
        ●
        ●
    }
    catch (const FormatException^ e)
    {
        //thrown when user somehow entered some illegal characters into eqInput
        buttonC->PerformClick();
    }
    catch (const DivideByZeroException^ zeroerr)
    {
        eqInput->Text = "Divide/0 !";
        to_clear = true;
    }
```

Kontakt: matebia076@student.polsl.pl
GitHub: https://github.com/BialekPL/modulo-calculator