|  |
| --- |
| 윈도우즈 프로그래밍 |

-과제-

9주차

윈도우 폼 : 계산기 기능 추가

20144638

이한범

# Window Form : 계산기 구현

이전의 코드에서는 계산기에서 기본 사칙연산만 구현하였다. 이번 과제에서는 기존 윈도우에서 제공하는 계산기를 모티브로하여 기능을 추가하였다.

추가한 기능은 다음과 같다.

[ 괄호, ±, %, √, CE, 메모리(MC,MR...), x2, xy, 삼각함수, Mod, log10, 계산식 창(가장 위의 텍스트 박스. 연산이 기록된다.) ]

1. 윈도우 폼 구성도

|  |
| --- |
|  |

1. 작성한 코드내용

# 기존 입력 값 초기화

|  |
| --- |
| private bool newButton;  //textBox1에 0만 입력되어있거나, 연산자가 눌러진 뒤 숫자 입력시  //기존에 입력되있는 값을 지움.  private void btn\_event(string n)  {  if (textBox1.Text == "0" || newButton == true)  {  textBox1.Text = n;  newButton = false;  }  else  textBox1.Text += n;  //textBox2.Text += n;  } |

▲ 수식 입력란에 값을 입력할 때의 조건문. 사칙연산이 눌러졌거나 값이 0만 입력되어있으면 새로운 값을 연산창(2번째 줄 텍스트박스)에 추가한다.

# 특정 계산식 (xy)

|  |
| --- |
| private string otherOper; //별개 연산자  private double front; // front 연산자 back  private double back;  private void Oper\_other()  {  switch (otherOper)  {  case "^":  back = double.Parse(textBox1.Text.Split('^')[1]);  textBox1.Text = Math.Pow(front, back).ToString();  front = 0.0;  back = 0.0;  otherOper = "";  break;  default:  break;  }  } |

▲ xy 버튼을 눌렀을 때 호출하는 함수. split으로 연산창에 입력된 수식의 x부분과 y부분을 분리해 Math.Pow() 메소드를 사용하여 제곱연산수행.

# 연산자 입력 시 실행되는 이벤트

|  |
| --- |
| private double saved; //사칙연산 누르면 값 저장  private bool newButton; //새로 숫자시작  private char myOper; //계산할 연산  //연산자 입력에 대한 조건문  private void Oper(char m)  {  Oper\_other();  //첫번째 if문 : 연산자 입력시 피연산자 여부조사  if (textBox1.Text != "")  {  saved = double.Parse(textBox1.Text);  myOper = m;  newButton = true;  //두번째 if문 : 음수, 양수 판별입력  if ((saved.ToString()[0]).Equals('-'))  save\_math("(" + saved + ")" + myOper.ToString());  else  save\_math(saved + myOper.ToString());  }  else if (textBox1.Text == "" && textBox2.Text == "") { }  else textBox2.Text += m;  } |

▲ 연산자 버튼들을 눌렀을 때 호출하는 함수. 연산자를 입력했을 때, 연산창에 피연산자가 있는지 판별하고, 이 후에 값이 음수인지 양수인지를 판별하여 계산식 창에 입력한다. 즉, 음수일 경우 (-1) 와 같이, 괄호가 같이 입력된다.

# 저장되는 계산식 textBox

|  |
| --- |
| private void save\_math(string n)  {  textBox2.Text += n;  } |

▲ 계산식 창에 인수로 넘긴 값이 입력된다.

# 숫자 입력 이벤트

|  |
| --- |
| private void button4\_Click(object sender, EventArgs e)  {  btn\_event("0");  }  private void button3\_Click(object sender, EventArgs e)  {  btn\_event("1");  }  ...  ...  private void button12\_Click(object sender, EventArgs e)  {  btn\_event("9");  } |

▲ 위의 btn\_event() 함수를 사용한다.

# 소수점

|  |
| --- |
| //소수점  private void button9\_Click(object sender, EventArgs e)  {  //Contains : 결과창에서 "."이 중복되지 않도록 검사  if (textBox1.Text.Contains(".") == false)  {  textBox1.Text += ".";  }  } |

▲ 소수점이 한번 이상 찍히면 더 이상 찍지 않는다. 실제로 소수점 값을 입력하여 계산하면 타입이 double이므로 정상적으로 실수계산이 된다.

# 기초 사칙연산 및 moduler

|  |
| --- |
| private void button5\_Click(object sender, EventArgs e)  {  Oper('+');  }  ...  ...  private void mod(object sender, EventArgs e)  {  Oper('%');  } |

▲ moduler의 경우 나머지 계산이므로 % 기호를 사용하여 계산한다. 위에서 설명한 Oper() 함수를 사용하여 구현하였다.

# Enter key (=)

|  |
| --- |
| // enter  private void button17\_Click(object sender, EventArgs e)  {  Oper\_other();  if (textBox1.Text == "") { }  else if ((textBox1.Text[0]).Equals('-'))  save\_math("(" + textBox1.Text + ")");  else  save\_math(textBox1.Text);  //사칙연산을 인식하여 계산값을 도출해내어줌.  DataTable dt = new DataTable();  var v = dt.Compute(textBox2.Text,"");  textBox1.Text = v.ToString();  } |

▲ Enter키를 누르게 되면 맨 처음 xy 연산인지 확인 후, 조건식에 따라 사칙연산을 수행한다.

# BackSpace, C(Clear), CE

|  |
| --- |
| private void button16\_Click(object sender, EventArgs e)  {  //backspace  //Substring : 0 ~ 문자열길이-1 만큼으로 자른다  if(textBox1.Text != "")  textBox1.Text = textBox1.Text.Substring(0, textBox1.Text.Length - 1);  }  //C  private void button22\_Click(object sender, EventArgs e)  {  textBox1.Text = "";  textBox2.Text = "";  saved = 0.0;  myOper = ' ';  }  //CE  private void button21\_Click(object sender, EventArgs e)  {  textBox1.Text = "";  } |

▲ BackSpace(<-) 버튼을 누르게 되면 연산창에 입력된 값을 하나 씩 지운다. 만일 연산창에 아무런 값도 없으면 아무일도 수행하지 않는다.

▲ C키를 누르면 연산창, 계산식 창 모든 내용을 지운다.

▲ CE키를 누르면 연산창에 있는 내용만 지운다.

# M(Memory)

|  |
| --- |
| private double memory; //메모리에 저장된 값    //MC  private void button18\_Click(object sender, EventArgs e)  {  memory = 0.0;  }  //MR : Memory Read  private void button19\_Click(object sender, EventArgs e)  {  textBox1.Text = memory.ToString();  }  //MS : Memory save  private void button20\_Click(object sender, EventArgs e)  {  memory = double.Parse(textBox1.Text);  }  //M+  private void button23\_Click(object sender, EventArgs e)  {  if (textBox1.Text == "") textBox1.Text = "";  else memory += double.Parse(textBox1.Text);  }  //M-  private void button26\_Click(object sender, EventArgs e)  {  if (textBox1.Text == "") textBox1.Text = "";  else memory -= double.Parse(textBox1.Text);  } |

▲ Memory 변수를 사용한다. memory 변수에 값을 저장하고, 불러오며, 값을 더하고 뺄 수 있도록 구현하였다. MC 버튼을 누르게되면 memory 변수에 있는 값을 삭제한다.

# 괄호

|  |
| --- |
| private int count;    private void left(object sender, EventArgs e)  {  textBox2.Text += "(";  count++;  }  // )  private void right(object sender, EventArgs e)  {  if (count > 0)  {  if (textBox1.Text != "" && (textBox1.Text[0]).Equals('-'))  save\_math("(" + textBox1.Text + ")" + ")");  else  save\_math(textBox1.Text + ")");  textBox1.Text = "";  count--;  }  } |

▲ 괄호를 입력한다. count 변수를 사용하여 왼쪽 괄호가 입력되면 늘려주고 오른쪽 괄호가 입력되면 감소시켜준다. 최종적으로 왼쪽 괄호에 따라 오른쪽 괄호를 입력할 수 있는 수를 제한한다.

# ± 전환

|  |
| --- |
| //+- 전환  private void sign\_transformation(object sender, EventArgs e)  {  if (textBox1.Text == "") textBox1.Text = "";  else textBox1.Text = (double.Parse(textBox1.Text) \* (-1)).ToString();  } |

▲ ± 버튼을 누르게되면 연산창에있는 값의 부호가 전환된다.

# 삼각함수

|  |
| --- |
| //sin  private void sin(object sender, EventArgs e)  {  if (textBox1.Text == "") textBox1.Text = "";  else textBox1.Text = Math.Sin(double.Parse(textBox1.Text)).ToString();  }  //cos  private void cos(object sender, EventArgs e)  {  if (textBox1.Text == "") textBox1.Text = "";  else textBox1.Text = Math.Cos(double.Parse(textBox1.Text)).ToString();  }  //tan  private void tan(object sender, EventArgs e)  {  if (textBox1.Text == "") textBox1.Text = "";  else textBox1.Text = Math.Tan(double.Parse(textBox1.Text)).ToString();  } |

▲ Math 라이브러리가 제공하는 삼각함수 메소드를 사용하여 구현하였다. 연산창에서 각 삼각함수를 적용한 값으로 치환된다.

# 제곱, 제곱근(루트) 및 log

|  |
| --- |
| private void pow(object sender, EventArgs e)  {  if (textBox1.Text == "") textBox1.Text = "";  else textBox1.Text = Math.Pow(double.Parse(textBox1.Text),2).ToString();  //√  private void root(object sender, EventArgs e)  {  if (textBox1.Text == "") textBox1.Text = "";  else textBox1.Text = Math.Sqrt(double.Parse(textBox1.Text)).ToString();  }  //pow\_y  private void pow\_y(object sender, EventArgs e)  {  if (textBox1.Text == "") textBox1.Text = "";  else  {  front = double.Parse(textBox1.Text);  textBox1.Text += "^";  otherOper = "^";  }  }  private void pow\_3(object sender, EventArgs e)  {  if (textBox1.Text == "") textBox1.Text = "";  else textBox1.Text = Math.Pow(double.Parse(textBox1.Text), 3).ToString();  }  private void log(object sender, EventArgs e)  {  if (textBox1.Text == "") textBox1.Text = "";  else textBox1.Text = Math.Log(double.Parse(textBox1.Text), 10).ToString();  } |

▲ x2, x3을 누르면 각각 pow, pow\_3 이벤트를 발생시킨다. Math 라이브러리가 제공하는 Pow 메서드를 사용하여 구현하였다.

▲ xy의 경우 위에서 설명하였듯, "^" 문자열로 구분하여 계산한다.

▲ 루트의 경우 삼각함수나 +,- 전환식과 비슷한 구조로 만들었다. Math 라이브러리가 제공하는 Sqrt 메서드를 사용하여 구현하였다. 연산창의 값이 루트를 씌운 값으로 치환된다.

▲ 로그 또한 루트와 유사하다.

# n!

|  |
| --- |
| private void factorial(object sender, EventArgs e)  {  if (textBox1.Text == "") textBox1.Text = "";  else  {  int n=1;  for (int i=2; i<=Int32.Parse(textBox1.Text); i++)  {  n \*= i;  }  textBox1.Text = n.ToString();  } |

▲ 팩토리얼을 for문을 통하여 구현하였다. 입력한 값만큼 팩토리얼 연산을 하는데, 너무 큰값이 입력되게 될 경우 실행되지않는다.

3) 실행결과

|  |
| --- |
|  |