

MỤC LỤC

I.PHẦN MỞ ĐẦU	1
1. Lí do chọn đề tài	1
2. Mục tiêu cho đề tài	1
3. Phương pháp nghiên cứu.....	1
4. Giới thiệu đề tài	1
II.PHẦN NỘI DUNG	2
1. Mô tả ứng dụng	2
2. Mô tả quá trình thực hiện	2
2.1.Thiết kế giao diện.....	2
2.2. Ứng dụng kiểu dữ liệu Stack trong Calculator.....	4
2.3. Chạy và kiểm thử.	11
2.4. Các lỗi và cách sửa các lỗi gặp phải	13
2.5. Bản phân công công việc	14
III.KẾT LUẬN	15
IV.TÀI LIỆU THAM KHẢO.....	16

MỤC LỤC HÌNH

<u>Hình 1: Thiết kế giao diện trên máy</u>	2
<u>Hình 2: Thiết kế ứng dụng trên Windows Forms Application</u>	3
<u>Hình 3: Các Stack được sử dụng trong đề tài</u>	4
<u>Hình 4: Hàm chuyển biểu thức trung tố sang biểu thức hậu tố</u>	6
<u>Hình 5: Ví dụ về hàm chuyển biểu thức trung tố sang hậu tố</u>	7
<u>Hình 6: Hàm tính toán biểu thức hậu tố</u>	8
<u>Hình 7: Ví dụ về hàm tính toán biểu thức hậu tố</u>	9
<u>Hình 8: Hàm định dạng dữ liệu vào</u>	10
<u>Hình 9: Hàm kiểm tra độ ưu tiên toán tử</u>	10
<u>Hình 10: Hàm kiểm tra toán tử và kiểm tra toán hạng</u>	10
<u>Hình 11: Tính toán với một số âm</u>	11
<u>Hình 12: Tính toán có dấu ngoặc</u>	11
<u>Hình 13: Tính toán công trừ nhân chia phần trăm</u>	12
<u>Hình 14: Tính toán số thực</u>	12
<u>Hình 15: Lỗi gặp phải</u>	13
<u>Hình 16: Bản phân công công việc</u>	14

I. PHẦN MỞ ĐẦU

1. Lí do chọn đề tài

Với đề tài “Calculator” có thể giúp tác giả vận dụng những kiến thức học được từ môn học, đặc biệt là hiểu rõ hơn về kiểu dữ liệu trừu tượng Stack.

2. Mục tiêu cho đề tài

Hoàn thành sản phẩm, có thể giúp người sử dụng tính toán các phép tính đơn giản. Giúp bản thân tác giả hiểu được, hiểu sâu hơn về kiểu dữ liệu trừu tượng ngăn xếp (Stack). Có thể sử dụng thành thạo các phương thức và thuộc tính của Stack.

3. Phương pháp nghiên cứu

Vận dụng các kiến thức đã học được từ bộ môn “Cấu trúc dữ liệu và giải thuật”, tham khảo các nguồn tài liệu, giáo trình được cung cấp. Ngoài ra còn tham khảo thêm các nguồn tài liệu khác trên mạng internet và vận dụng tất cả để thực hiện đề tài nghiên cứu một cách thành công nhất.

4. Giới thiệu đề tài

Calculator hay máy tính là một ứng dụng cho phép người dùng tính toán các biểu thức từ cơ bản đến nâng cao, giúp việc tính toán trở nên đơn giản hơn và có phần nhanh chóng hơn qua đó cải thiện năng suất làm việc của người dùng.

II. PHẦN NỘI DUNG

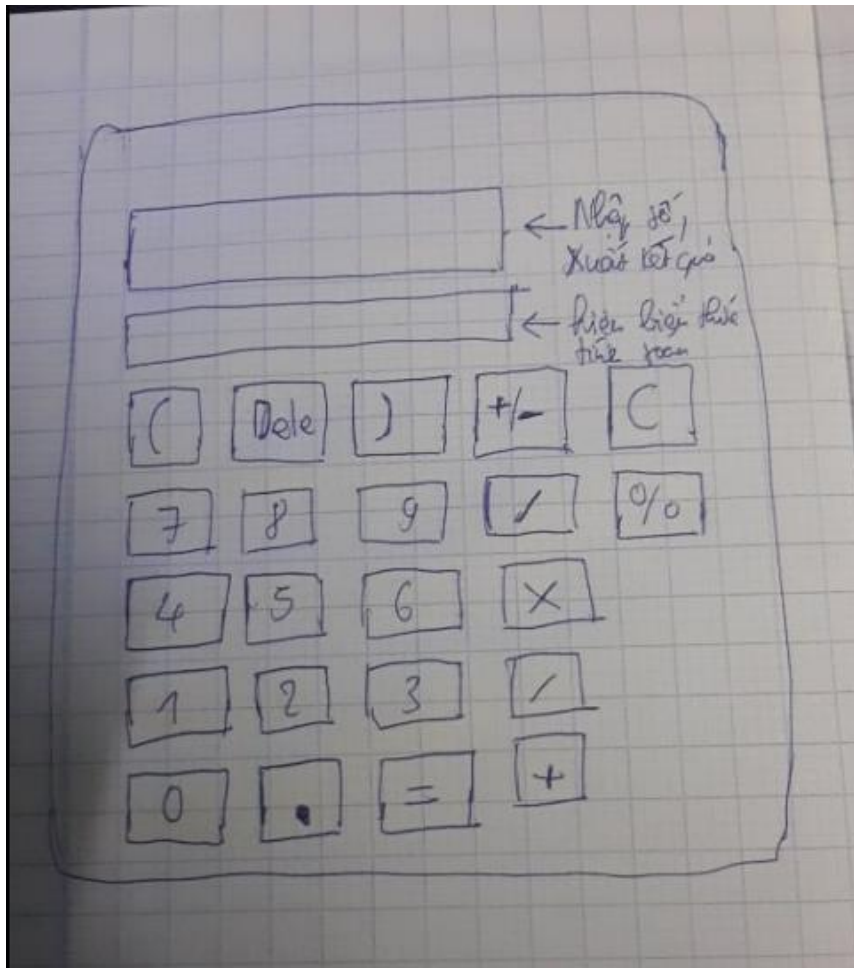
1. Mô tả ứng dụng

Tính toán các biểu thức với các phép toán cơ bản như cộng(+), trừ(-), nhân(*), chia(/), phần trăm(%).

2. Mô tả quá trình thực hiện

2.1. Thiết kế giao diện

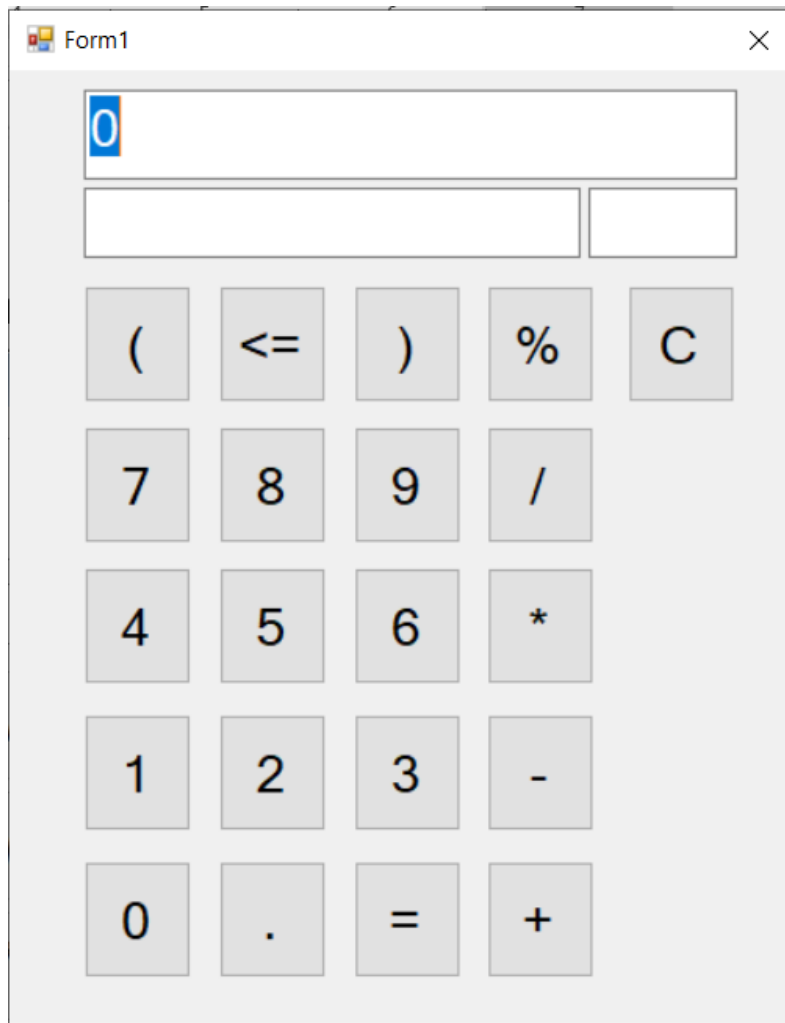
Việc thiết kế giao diện của ứng dụng là một phần rất quan trọng trong việc xây dựng ứng dụng, là bộ mặt của ứng dụng, nơi người dùng trực tiếp thao tác với ứng dụng.



Hình 1: Thiết kế giao diện trên giấy.

Kể từ lúc này chúng ta sẽ bắt đầu sử dụng Windows Forms Application để thiết kế và lập trình ứng dụng.

- Sau khi tạo project thành công, chúng ta bắt đầu điều chỉnh form với các thao tác trên thanh properties, chỉnh lại mục *Name* và mục *Text*.
- Tạo *Textbox* để hiển thị kết quả hoặc phép tính đã nhập
- Tạo các *Button* để hiển thị các số, phép tính,...
- Tiến hành tinh chỉnh vị trí, canh khoảng cách giữa các *Button*.



Hình 2: Thiết kế ứng dụng trên Windows Forms Application.

2.2. Ứng dụng kiểu dữ liệu Stack trong Calculator

Ngăn xếp (Stack) là một dạng danh sách được cài đặt nhằm sử dụng cho các ứng dụng cần xử lý theo thứ tự đảo ngược. Trong cấu trúc dữ liệu ngăn xếp, tất cả các thao tác thêm, xóa một phần tử đều phải thực hiện ở đầu một danh sách, đầu này gọi là đỉnh (top) của ngăn xếp. Hoạt động dựa trên quy tắc LIFO (Last In, First Out). Ta có thể ứng dụng ngăn xếp để lưu lại các toán hạng và kết quả trung gian, từ đó phần tử cuối cùng còn sót lại của Stack là kết quả của biểu thức.

2.2.1 Các phương thức được sử dụng với kiểu dữ liệu Ngăn xếp (Stack) trong đề tài:

- *Push()*: thêm vị trí hiện tại vào *Stack*.
- *Pop()*: lấy vị trí trước đó ra khỏi *Stack*.

2.2.2. Các Stack trong đề tài:

```
//chuyển trung tố sang hậu tố
private static string ProcessConvert(string[] tokens)
{
    Stack<string> stack = new Stack<string>(); //tạo 1 stack
    StringBuilder result = new StringBuilder(); //tạo 1 chuỗi

    for (int i = 0; i < tokens.Length; i++) //duyệt chuỗi
    {
        string token = tokens[i];
        if (ExprHelper.IsOperator(token)) //nếu là toán tử
        {
            //...
        }
    }

    #region Evaluate
    //tính toán giá trị hậu tố
    public static double EvaluatePrefix(string prefix)
    {
        return EvaluatePostfix(prefix.Trim().Split(' '));
    }

    private static double EvaluatePostfix(IEnumerable<string> tokens)
    {
        Stack<double> stack = new Stack<double>(); //tạo 1 stack

        foreach (string s in tokens)
        {
            //...
        }
    }
}
```

Hình 3: Các Stack được sử dụng trong đề tài.

- Tôi sử dụng 2 *Stack* trong đề tài dùng để lưu giá trị tạm thời dưới dạng kiểu dữ liệu *string* và *double*.

2.2.3. Ứng dụng các Stack trong đề tài:

- Khái niệm biểu thức trung tố:

Ký pháp trung tố là:

-Phương pháp biểu diễn phép toán 2 ngôi.

-Một phép toán 2 ngôi trên tập X là một ánh xạ $f: X \times X \rightarrow X$ cho $(a,b) \Rightarrow f(a,b)$ thuộc A . Ánh xạ f khi đó thường được kí hiệu bởi $*$, được gọi là toán tử, các phần tử a, b được gọi là các hạng tử (hay toán hạng).

-Khi viết biểu thức biểu diễn phép toán đó tôi đặt kí hiệu toán tử ở giữa các toán hạng. Ví dụ: $a + b$, $a - b$, $a * b \dots$ Khi biểu thức có nhiều phép toán, tôi dùng các cặp dấu ngoặc “(“ và “)” và thứ tự ưu tiên các phép toán để chỉ rõ thứ tự thực hiện các phép toán. Nguyên tắc ưu tiên là $*$, $/$, $\%$, $^$ có độ ưu tiên cao hơn $+$, $-$. (1)

- Khái niệm biểu thức hậu tố: Biểu thức hậu tố tức là các toán tử sẽ được đặt sau các toán hạng. Cách biểu diễn này được gọi là “ký pháp nghịch đảo Ba Lan” hoặc được viết tắt là RPN(Reverse Polish notation), được phát minh vào khoảng giữa thập kỷ 1950 bởi một triết học gia và nhà khoa học máy tính Charles Hamblin người Úc. Thay vì viết $x + y$ như dạng trung tố, ta sẽ viết $x y +$. Tùy theo độ ưu tiên của toán tử mà chúng sẽ được sắp xếp khác nhau, bạn có thể xem một số ví dụ ở phía sau phần giới thiệu này. (2)

```

//chuyển trung tố sang hậu tố
private static string ProcessConvert(string[] tokens)
{
    Stack<string> stack = new Stack<string>(); //tạo 1 stack
    StringBuilder result = new StringBuilder(); //tạo 1 chuỗi
    for (int i = 0; i < tokens.Length; i++) //duyet chuỗi
    {
        string token = tokens[i];
        if (ExprHelper.IsOperator(token)) //nếu là toán tử
        {
            if ((i == 0) || (i > 0 && (ExprHelper.IsOperator(tokens[i - 1]) || tokens[i - 1] == "(")))
            {
                if (token == "--") //công dụng dùng để lấy số âm
                {
                    result.Append(token + tokens[i + 1]).Append(" ");
                    i++;
                }
            }
            else
            {
                while (stack.Count > 0 && ExprHelper.GetPriority(token) <= ExprHelper.GetPriority(stack.Peek()))
                {
                    result.Append(stack.Pop()).Append(" "); //pop toán tử từ stack ra cho đến khi k thỏa điều kiện
                    stack.Push(token); //push toán tử vào stack
                }
            }
        }
        else if (token == "(") //nếu là '(' push vào stack
        {
            stack.Push(token);
        }
        else if (token == ")") //nếu là ')' pop toán tử từ stack ra cho đến khi gặp '('
        {
            string x = stack.Pop();
            while (x != "(")
            {
                result.Append(x).Append(" ");
                x = stack.Pop();
            }
        }
        else // (IsOperand(s)) //kiểm tra có phải là toán tử k
        {
            result.Append(token).Append(" "); //phải thì ỏ vào chuỗi resylt và thêm 1 khoảng trắng
        }
    }
    while (stack.Count > 0) result.Append(stack.Pop()).Append(" ");
    return result.ToString();
}

```

Hình 4: Hàm chuyển biểu thức trung tố sang biểu thức hậu tố .

- Stack đầu tiên dùng để chứa các toán tử nhằm giúp hàm giúp hàm chuyển biểu thức trung tố sang dạng hậu tố.

Thuật toán của hàm này là:

-Khởi tạo tạo 1 Stack có kiểu dữ liệu là string có tên là stack, khởi tạo 1 chuỗi StringBuilder có tên là result.

-Ta bắt đầu duyệt chuỗi. Lấy giá trị kí tự tại vị trí i. (string token = tokens[i])

+Nếu token là một toán tử:

*Nếu tra $i = 0$ hoặc $i > 0$ đồng thời $tokens[i - 1]$ là toán tử hoặc $tokens[i - 1] = "("$), ta kiểm tra $token = '-'$ nếu đúng thì xuất token và $tokens[i + 1]$ vào chuỗi result.

*Ngược lại nếu không đúng, ta kiểm tra stack rỗng không và ta kiểm tra mức độ ưu tiên của toán tử top của stack. Nếu ưu tiên $token \leq$ ưu tiên toán tử top của stack thì pop phần tử top của ghi vào chuỗi result và tiến hành lặp lại bước so sánh trên. Ngược lại, ta $push(token)$ vào stack.

+Nếu $token = "("$, push vào stack

+Nếu $token = "("$, pop các toán tử trong stack và xuất vào result cho tới khi gặp $"("$. Lưu ý không xuất ra dấu mở "(";

+Nếu token là 1 toán toán hạng, xuất vào result.

-Khi duyệt hết chuỗi ta pop toàn bộ toán tử từ stack và xuất vào result.

- Ví dụ minh họa:

Bài toán trung tố đầu vào: $7+3*2+(22-2)/2$.

Token	Stack	Output
7	{Empty}	7
+	+	7
3	+	7 3
*	+ *	7 3
2	+ *	7 3 2
+	+	7 3 2 * +
(+ (7 3 2 * +
22	+ (7 3 2 * + 22
-	+ (-	7 3 2 * + 22
2	+ (-	7 3 2 * + 22 2
)	+	7 3 2 * + 22 2 -
/	+ /	7 3 2 * + 22 2 -
2	+ /	7 3 2 * + 22 2 - 2
	{Empty}	7 3 2 * + 22 2 - 2 / +

Hình 5: Ví dụ về hàm chuyển biểu thức trung tố sang hậu tố.

```

private static double EvaluatePostfix(IEnumerable<string> tokens)
{
    Stack<double> stack = new Stack<double>(); //tạo 1 stack

    foreach (string s in tokens)
    {
        if (ExprHelper.IsOperator(s)) //nếu gặp toán tử pop 2 phần tử
        {
            double x = stack.Pop();
            double y = stack.Pop();
            switch (s)
            {
                case "+": y += x; break;
                case "-": y -= x; break;
                case "*": y *= x; break;
                case "/": y /= x; break;
            }
            stack.Push(y);
        }
        else // IsOperand(toán hạng) push vào stack
        {
            stack.Push(double.Parse(s));
        }
    }
    return stack.Pop(); //lấy kết quả từ stack
}

```

Hình 6: Hàm tính toán biểu thức hậu tố.

- Stack thứ hai dùng để lưu các toán hạng và kết quả trung gian, sau quá trình tính toán, phần tử cuối cùng còn lại của stack chính là giá trị của biểu thức hậu tố.

Cụ thể thuật toán của hàm tính toán biểu thức hậu tố:

-Khởi tạo 1 Stack rỗng có tên stack. (vì là số nên ta lấy kiểu dữ liệu là double)

-Ta bắt đầu duyệt chuỗi:

+Nếu là toán tử thì pop 2 phần tử đầu stack ra tính toán và push kết quả trở lại stack. Nếu stack không có đủ 2 toán hạng để thao tác thì chương trình sẽ gặp lỗi.

+Nếu giá trị hiện thời là toán hạng, push vào stack.

-Khi duyệt hết chuỗi, giá trị top của stack chính là giá trị kết quả, ta pop sử dụng.

- Ví dụ minh họa:

Hậu tố đầu vào: $7\ 3\ 2\ *\ +\ 22\ 2\ -\ 2\ /\ +\ .$

token	stack	x	y	
7	7			
3	7 3			
2	7 3 2			
*	7	2	3	$3*2=6$
	7 6			
+	empty	6	7	$7+6=13$
	13			
22	13 22			
2	13 22 2			
-	13	2	22	$22-2=20$
	13 20			
2	13 20 2			
/	13	2	20	$20/2=10$
	13 10			
+		10	13	$13+10=23$

Hình 7: Ví dụ về hàm tính toán biểu thức hậu tố.

- Các hàm phụ khi tính toán:

```
public static string FormatExpression(string expression) //định dạng infix
{
    expression = expression.Replace(" ", ""); //xóa hết các khoảng trắng
    expression = Regex.Replace(expression, @"\+|\-|\*|\/|\(|\)", match =>
        String.Format(" {0} ", match.Value) //gấp các dấu các toán tử thêm khoản trắng ở 2 bên toán tử
    );
    expression = expression.Replace("|", " "); //có trường hợp toán tử liền nhau nên cần phải có
    expression = expression.Trim(); //xóa khoản trắng giữa đầu và cuối chuỗi

    return expression;
}
```

Hình 8: Hàm định dạng dữ liệu vào.

```
public static int GetPriority(string op)
{
    if (op == "*" || op == "/" )
        return 2;
    if (op == "+" || op == "-")
        return 1;
    return 0;
}
```

Hình 9: Hàm kiểm tra độ ưu tiên của toán tử.

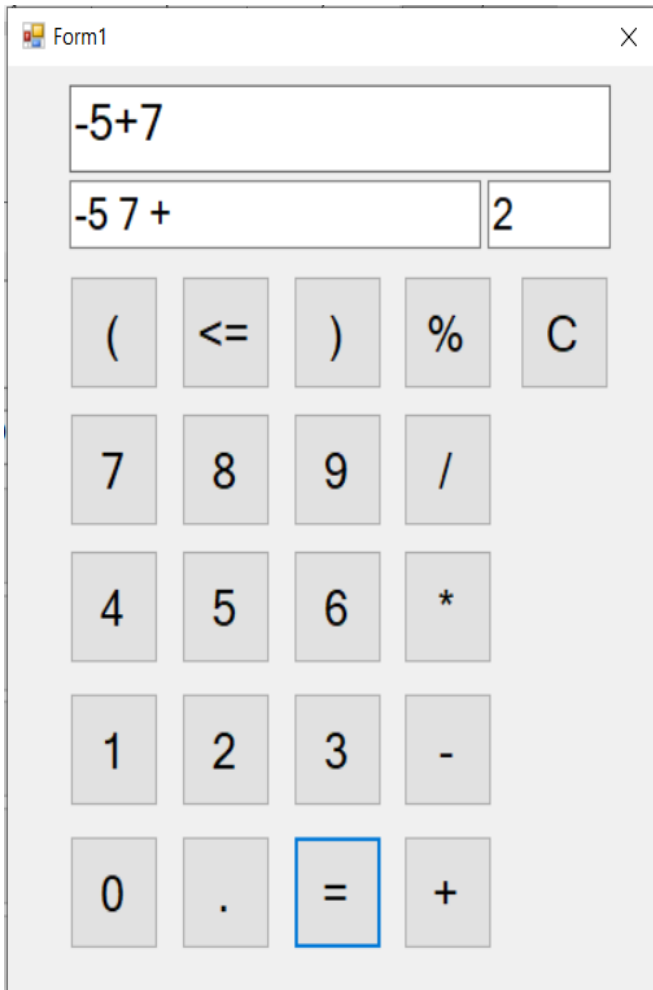
```
public static bool IsOperator(string str) //kiểm tra toán tử
{
    return Regex.Match(str, @"^(\+|\-|\*|\/)$").Success;
}

public static bool IsOperand(string str) //kiểm tra toán hạng
{
    return Regex.Match(str, @"^\d+$|^([a-z]|[A-Z])$").Success;
}
```

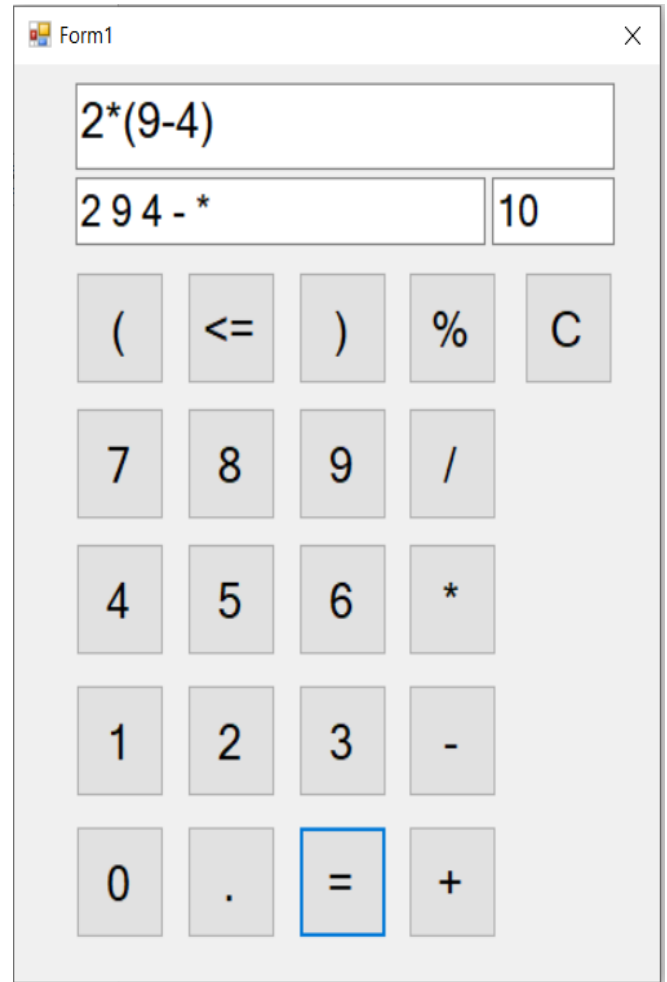
Hình 10: Hàm kiểm tra toán tử và kiểm tra toán hạng.

2.3. Chạy và kiểm thử.

Một số các chương hợp test thử chương trình:



Hình 11: Tính toán với một số âm.



Hình 22: Tính toán có dấu ngoặc.

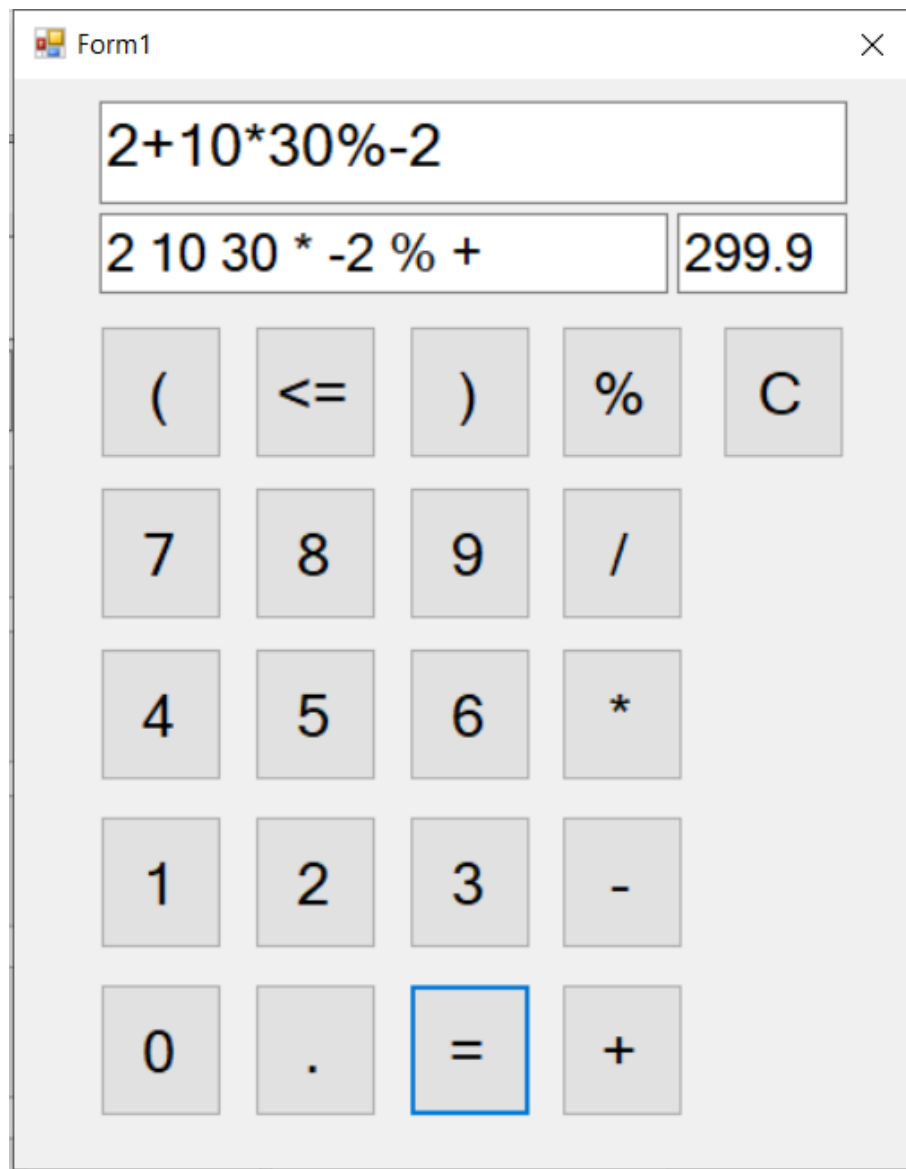
The screenshot shows a Windows calculator application titled "Form1". The display shows the expression $2+10-20/2*30\%$. Below the display, a row of buttons contains the text "2 10 + 20 2 / 30 * % -" followed by the result "9". The calculator interface includes a numeric keypad (0-9), a decimal point, a percentage button, and various mathematical operators. The equals button is highlighted with a blue border.

Hình 33: Tính toán cộng trừ nhân chia phần trăm.

The screenshot shows a Windows calculator application titled "Form1". The display shows the expression $3*1.5-(3+2)$. Below the display, a row of buttons contains the text "3 1.5 * 3 2 + -" followed by the result "-0.5". The calculator interface includes a numeric keypad (0-9), a decimal point, a percentage button, and various mathematical operators. The equals button is highlighted with a blue border.

Hình 14: Tính toán với số thực.

2.4. Các lỗi và cách sửa các lỗi gặp phải



Hình 15: Lỗi gặp phải.

-Trường hợp lỗi: Trong bài toán nếu các phép tính(+ - * /) nếu ở sau dấu % thì khi chạy ra kết quả sẽ sai.

-Giải pháp: Tính luôn kết quả luôn trong khi nhập. Ví dụ: Khi nhập phép toán $2+10*30\%-2$ thì nó sẽ biến thành $2+10*0.3-2$ rồi mới tính toán.

2.5. Bản phân công công việc

KẾ HOẠCH THỰC HIỆN ĐỒ ÁN CUỐI KỲ MÔN CẤU TRÚC DỮ LIỆU & GIẢI THUẬT - 2019							
DANH SÁCH CÔNG VIỆC	NỘI DUNG CHI TIẾT CÔNG VIỆC	Nguyễn Đức Chánh	Vũ Đức Tuấn	Ngày bắt đầu (dự kiến)	Ngày kết thúc (dự kiến)	Ngày bắt đầu (thực tế)	Ngày kết thúc (thực tế)
Tìm hiểu đề tài	Tìm hiểu giao diện calculator		X	5/9/2019	6/9/2019	20/9/2019	21/9/2019
	Vẽ ra dao diện cơ bản bằng giấy		X	5/9/2019	6/9/2019	20/9/2019	21/9/2019
	Tìm hiểu github.tạo link github	X		6/9/2019	15/9/2019	20/9/2019	20/9/2019
	Tìm hiểu code ở trên các trang web	X	X	11/9/2019	20/9/2019	21/9/2019	10/10/2019
	Tìm hiểu thiết kế giao diện	X	X	11/9/2019	20/9/2019	21/9/2019	10/10/2019
	Tìm hiểu cách tạo ứng dụng Calculator	X	X	11/9/2019	20/9/2019	21/9/2019	10/10/2019
	Viết code	X	X	20/9/2019	30/11/2019	1/10/2019	28/10/2019
Triển khai đề tài	1.mục ke giao diện		X	20/9/2019	30/11/2019	1/10/2019	4/10/2019
	Tạo ứng dụng	X		20/9/2019	30/11/2019	1/10/2019	28/10/2019
	Chạy thử. Sửa lỗi	X	X	30/11/2019		28/10/2019	

Hình 16: Bản phân công công việc.

III.KẾT LUẬN

- Mức độ hoàn thành mục tiêu: 100%.
- Các khó khăn gặp phải: Lúc bắt đầu chưa biết cách sử dụng Winform, chưa nắm chắc về các kiến thức chuỗi, chưa nắm cách chuyển biểu thức trung tố sang hậu tố cũng như tính toán biểu thức hậu tố, còn nhiều bất đồng quan điểm khi lên kế hoạch, chưa biết ứng dụng stack vào ứng dụng để tính toán.
- Cách khắc phục: tìm hiểu và học cách sử dụng Winform từ nhiều nguồn tài liệu trên internet, điều chỉnh size ảnh thích hợp cho việc code cũng như dễ nhìn cho giao diện, xin lời khuyên của giảng viên để lên kế hoạch cho đồ án, tìm một số ứng dụng đã có sẵn để đọc và hiểu thêm về ứng dụng stack để tính toán trung tố hậu tố.
- Ưu điểm của đồ án: sử dụng, khai thác hết các thuộc tính của kiểu dữ liệu Ngăn xếp (Stack), có tính ứng dụng cho việc học tập để tính toán các bài toán đơn giản, độ phức tạp thấp dễ sử dụng và lập trình.
- Những khuyết điểm của đồ án: so với các sản phẩm tương tự khác bên ngoài thì sản phẩm hiện tại vẫn còn chưa phát triển hoàn thiện nhất (không tính được sin, cos, tan,), chưa được phổ biến đến mọi người, chưa tự động sửa biểu thức lúc nhập, giao diện chưa thật sự quá bắt mắt, cầu kì bằng các sản phẩm khác, còn có các tính toán sai trường hợp có dấu % nằm giữa của phép toán.
- Hướng phát triển đồ án: có thể thêm các tính năng khác như tính được sin, cos, tan,.... Báo lỗi khi biểu thức đầu vào sai. Tăng tốc độ tính toán các bài toán khó, có độ phức tạp cao. Đầu tư thêm về phần xử lý đồ họa giúp tăng độ thẩm mỹ, bắt mắt thu hút người dùng hơn.

IV. TÀI LIỆU THAM KHẢO

Tài liệu tham khảo:

“GIÁO TRÌNH: CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT” của tác giả: Lê Văn Vinh, NXB Đại học Quốc gia TP. Hồ Chí Minh.

Các trang web tham khảo:

- <https://yinyangit.wordpress.com/2011/01/27/algorithm-tinh-gia-tri-bieu-thuc-tien-to-va-hau-to/>
- <https://www.howkteam.vn/course/khoa-hoc-lap-trinh-c-nang-cao/stack-trong-c-1562>
- <https://o7planning.org/vi/10795/huong-dan-su-dung-bieu-thuc-chinh-quy-trong-csharp>
- <https://o7planning.org/vi/10441/huong-dan-csharp-string-va-stringbuilder#a1341803>
- <https://yinyangit.wordpress.com/2011/01/26/algorithm-chuy%E1%BB%83n-bi%E1%BB%83u-th%E1%BB%A9c-trung-t%E1%BB%91-sang-ti%E1%BB%81n-t%E1%BB%91-va-h%E1%BA%ADu-t%E1%BB%91-b%E1%BA%B1ng-stack/>
- (1) <https://www.stdio.vn/articles/ung-dung-stack-bieu-thuc-trung-to-infix-470>
- (2) <https://text.xemtailieu.com/tai-lieu/bai-tieu-luan-ung-dung-ngan-xep-stack-va-hang-doi-queue-de-viet-chuong-trinh-bien-doi-bieu-thuc-trung-to-thanh-tien-to-va-hau-to-550637.html>