计算器的设计与实现说明文档

完成人：曾德明 学号：20172131138

完成时间：2019 月 10 月 4 号

一、软件名称

普通四则计算器

二、软件内容简介

这是一个使用 android studio 编写的计算器，整个 app 的界面配色参考了 iphone 自带的计算机的配色，不过在输入框上我选择了保留算式和答案，同时按钮设置成每行五个，能容纳多一点常规算术操作运算符。

计算器的界面我选择使用 LinearLayout 布局来进行排布，用一个 LinearLayout 作为整个计算器的主布局，同时设置背景为黑色，用七个 LinearLayout 子部局把输入输出框、各行按钮独立出来，同时设置对应的权值，使得整个计算器的布局可以适应各种屏幕的手机。

在计算器功能的实现方面，我根据大二数据结构上学习的中缀表达式与后缀表达式的转换和计算，实现了计算器的主体功能，同时根据自己的理解，使得计算器可以支持负数与小数的运算。在有关算法的代码方面，我是在参考了算法思想以及网上的相关算法后后自己改进实现的代码，所以有可能代码并不是最简洁的，但是测试起来是可以完成的。

具体的算法实现方面，我在对计算器进行界面布局之后，使用 Stringbuilder 来存储输入框的算式，在 mainActivity 文件中，主要是对整个计算器的界面进行初始化，设置按钮的响应事件，而在 InfixInToDuffix 文件中，进行中缀表达式转化后缀表达式的操作以及后缀表达式的计算。运算符的优先级我选择定义 Map<Character, Integer>basic 来进行存储，List<String> queue 用于定义存储后缀表达式的队列，List<Character> stack 定义了一个栈，用于存储算术符并根据算法实现中缀转后缀。最后在函数 dealEquation()中进行等式运算。

三、界面设计

（一）主界面：

（二）等式输入：

（三）输入普通计算：

（四）除数为 0 的情况：

（五）阶乘：


9:11
LTE
不科学计算器
5!
120.0
ANS
AC
DEL
+/-
÷
(
7
8
9
×
)
4
5
6
-
^
1
2
3
+
X!
0
.
%
=

（六）乘方：

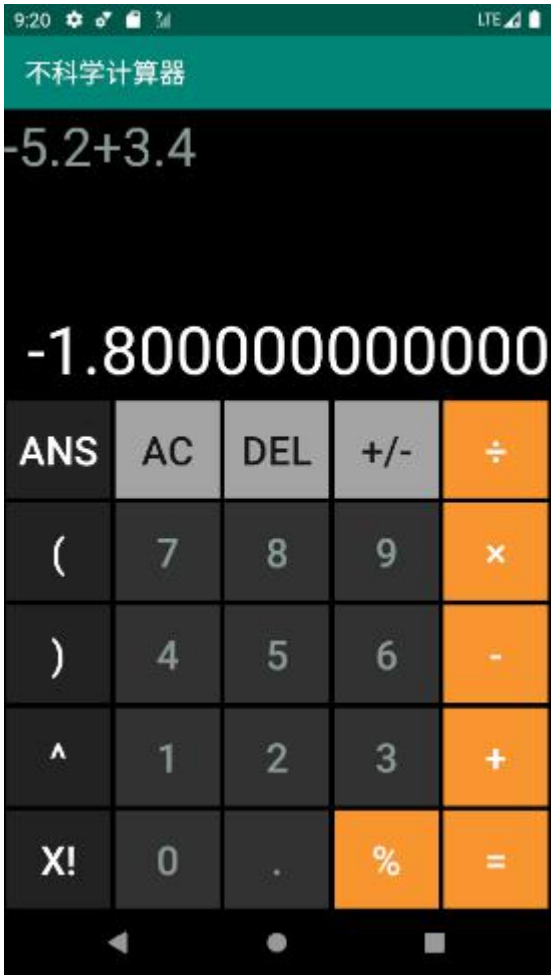

9:11
LTE
不科学计算器
2^9
512.0
ANS
AC
DEL
+/-
÷
(
7
8
9
×
)
4
5
6
-
^
1
2
3
+
X!
0
.
%
=

（七）ANS 获取上一次计算的答案：

（八）正确的优先级：

（九）计算式过长时换行显示：

（十）小数/负数计算：



五、代码设计

（一）主界面布局代码 activity_main.xml：

```
<?xml version="1.0" encoding="utf-8"?>
<!--外层竖线布局-->
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#010101">

    <!--内层线性横向布局-->
    <!--输入框-->
    <!--权重为 2-->
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:orientation="horizontal"
        android:layout_weight="2">

        <!--使用 sp 作为字体大小单位,会随着系统的字体大小改变而 dp 作为单位则不会-->
        <TextView
            android:id="@+id/textInput"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:textSize="40sp"
            android:textColor="#899a94"
            android:hint="请输入算式"
            android:textColorHint="#899a94"/>

    </LinearLayout>

    <!--输出框-->
    <!--权重为 1-->
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:orientation="horizontal"
        android:layout_weight="1">

        <!--使用 sp 作为字体大小单位,会随着系统的字体大小改变而 dp 作为单位则不会-->
        <TextView
            android:id="@+id/textOutput"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:gravity="right"
            android:textSize="50sp"
            android:textColor="#ffffff"
            android:hint="0"
            android:textColorHint="#ffffff"/>

    </LinearLayout>

    <!--第一行按钮;权重为 1;每行四个-->
    <LinearLayout
        android:layout_width="match_parent"
```

```
        android:layout_height="0dp"
        android:orientation="horizontal"
        android:layout_marginBottom="2sp"
        android:layout_weight="1"
        android:weightSum="5">

        <!--ANS 按钮-->
        <Button
            android:id="@+id/ANS"
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_marginLeft="2sp"
            android:layout_marginRight="2sp"
            android:text="@string/ANS"
            android:textColor="#ffffff"
            android:textSize="30sp"
            android:layout_weight="1"
            android:background="#212121" />

        <!--AC 按钮-->
        <Button
            android:id="@+id/AC"
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_marginLeft="2sp"
            android:layout_marginRight="2sp"
            android:text="@string/AC"
            android:textSize="30sp"
            android:layout_weight="1"
            android:background="#a4a4a4"/>

        <!--DEL 按钮-->
        <Button
            android:id="@+id/DEL"
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_marginLeft="2sp"
            android:layout_marginRight="2sp"
            android:text="@string/DEL"
            android:textSize="30sp"
            android:layout_weight="1"
            android:background="#a4a4a4"/>

        <!--正负变化按钮-->
        <Button
            android:id="@+id/status"
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_marginLeft="2sp"
            android:layout_marginRight="2sp"
            android:text="@string/status"
            android:textSize="30sp"
            android:layout_weight="1"
            android:background="#a4a4a4"/>

        <!--除法按钮-->
        <Button
            android:id="@+id/divide"
            android:layout_width="0dp"
```

```
            android:layout_height="match_parent"
            android:layout_marginLeft="2sp"
            android:layout_marginRight="2sp"
            android:text="@string/divide"
            android:textColor="#fefcff"
            android:textSize="30sp"
            android:layout_weight="1"
            android:background="#f8952d"/>

    </LinearLayout>

    <!--第二行按钮-->
    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="0dp"
        android:orientation="horizontal"
        android:layout_marginTop="2sp"
        android:layout_marginBottom="2sp"
        android:layout_weight="1"
        android:weightSum="5">

        <!--(按钮-->
        <Button
            android:id="@+id/leftbracket"
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_marginLeft="2sp"
            android:layout_marginRight="2sp"
            android:text="@string/leftbracket"
            android:textColor="#ffffff"
            android:textSize="30sp"
            android:layout_weight="1"
            android:background="#212121"/>

        <!--7 按钮-->
        <Button
            android:id="@+id/seven"
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_marginLeft="2sp"
            android:layout_marginRight="2sp"
            android:text="@string/seven"
            android:textColor="#899a94"
            android:textSize="30sp"
            android:layout_weight="1"
            android:background="#333333"/>

        <!--8 按钮-->
        <Button
            android:id="@+id/eight"
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_marginLeft="2sp"
            android:layout_marginRight="2sp"
            android:text="@string/eight"
            android:textColor="#899a94"
            android:textSize="30sp"
            android:layout_weight="1"
            android:background="#333333"/>
```

```
        <!--9 按钮-->
        <Button
            android:id="@+id/nine"
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_marginLeft="2sp"
            android:layout_marginRight="2sp"
            android:text="@string/nine"
            android:textColor="#899a94"
            android:textSize="30sp"
            android:layout_weight="1"
            android:background="#333333"/>

        <!--乘法按钮-->
        <Button
            android:id="@+id/multiply"
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_marginLeft="2sp"
            android:layout_marginRight="2sp"
            android:text="@string/multiply"
            android:textColor="#fefcff"
            android:textSize="30sp"
            android:layout_weight="1"
            android:background="#f8952d"/>

    </LinearLayout>

    <!--第三行按钮-->
    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="0dp"
        android:orientation="horizontal"
        android:layout_marginTop="2sp"
        android:layout_marginBottom="2sp"
        android:layout_weight="1"
        android:weightSum="5">

        <!--)按钮-->
        <Button
            android:id="@+id/rightbracket"
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_marginLeft="2sp"
            android:layout_marginRight="2sp"
            android:text="@string/rightbracket"
            android:textColor="#ffffff"
            android:textSize="30sp"
            android:layout_weight="1"
            android:background="#212121"/>

        <!--4 按钮-->
        <Button
            android:id="@+id/four"
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_marginLeft="2sp"
            android:layout_marginRight="2sp"
```

```
            android:text="@string/four"
            android:textColor="#899a94"
            android:textSize="30sp"
            android:layout_weight="1"
            android:background="#333333"/>

        <!--5 按钮-->
        <Button
            android:id="@+id/five"
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_marginLeft="2sp"
            android:layout_marginRight="2sp"
            android:text="@string/five"
            android:textColor="#899a94"
            android:textSize="30sp"
            android:layout_weight="1"
            android:background="#333333"/>

        <!--6 按钮-->
        <Button
            android:id="@+id/six"
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_marginLeft="2sp"
            android:layout_marginRight="2sp"
            android:text="@string/six"
            android:textColor="#899a94"
            android:textSize="30sp"
            android:layout_weight="1"
            android:background="#333333"/>

        <!--减法按钮-->
        <Button
            android:id="@+id/subtract"
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_marginLeft="2sp"
            android:layout_marginRight="2sp"
            android:text="@string/subtract"
            android:textColor="#fefcff"
            android:textSize="30sp"
            android:layout_weight="1"
            android:background="#f8952d"/>

    </LinearLayout>

    <!--第四行按钮-->
    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="0dp"
        android:orientation="horizontal"
        android:layout_marginTop="2sp"
        android:layout_marginBottom="2sp"
        android:layout_weight="1"
        android:weightSum="5">

        <!--乘方按钮-->
        <Button
```

```
        android:id="@+id/power"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_marginLeft="2sp"
        android:layout_marginRight="2sp"
        android:text="@string/power"
        android:textColor="#ffffff"
        android:textSize="30sp"
        android:layout_weight="1"
        android:background="#212121"/>

    <!--1 按钮-->
    <Button
        android:id="@+id/one"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_marginLeft="2sp"
        android:layout_marginRight="2sp"
        android:text="@string/one"
        android:textColor="#899a94"
        android:textSize="30sp"
        android:layout_weight="1"
        android:background="#333333"/>

    <!--2 按钮-->
    <Button
        android:id="@+id/two"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_marginLeft="2sp"
        android:layout_marginRight="2sp"
        android:text="@string/two"
        android:textColor="#899a94"
        android:textSize="30sp"
        android:layout_weight="1"
        android:background="#333333"/>

    <!--3 按钮-->
    <Button
        android:id="@+id/three"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_marginLeft="2sp"
        android:layout_marginRight="2sp"
        android:text="@string/three"
        android:textColor="#899a94"
        android:textSize="30sp"
        android:layout_weight="1"
        android:background="#333333"/>

    <!--加法按钮-->
    <Button
        android:id="@+id/add"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_marginLeft="2sp"
        android:layout_marginRight="2sp"
        android:text="@string/add"
        android:textColor="#fefcff"
```

```
            android:textSize="30sp"
            android:layout_weight="1"
            android:background="#f8952d"/>

    </LinearLayout>

    <!--第五行按钮-->
    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="0dp"
        android:orientation="horizontal"
        android:layout_marginTop="2sp"
        android:layout_marginBottom="2sp"
        android:layout_weight="1"
        android:weightSum="5">

        <!--阶乘按钮-->
        <Button
            android:id="@+id/factorial"
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_marginLeft="2sp"
            android:layout_marginRight="2sp"
            android:text="@string/factorial"
            android:textColor="#ffffff"
            android:textSize="30sp"
            android:layout_weight="1"
            android:background="#212121"/>

        <!--0 按钮-->
        <Button
            android:id="@+id/zero"
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_marginLeft="2sp"
            android:layout_marginRight="2sp"
            android:text="@string/zero"
            android:textColor="#899a94"
            android:textSize="30sp"
            android:layout_weight="1"
            android:background="#333333"/>

        <!--小数点按钮-->
        <Button
            android:id="@+id/point"
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_marginLeft="2sp"
            android:layout_marginRight="2sp"
            android:text="@string/point"
            android:textColor="#899a94"
            android:textSize="30sp"
            android:layout_weight="1"
            android:background="#333333"/>

        <!--百分号按钮-->
        <Button
            android:id="@+id/mod"
            android:layout_width="0dp"
```

```
            android:layout_height="match_parent"
            android:layout_marginLeft="2sp"
            android:layout_marginRight="2sp"
            android:text="@string/mod"
            android:textColor="#fefcff"
            android:textSize="30sp"
            android:layout_weight="1"
            android:background="#f8952d"/>

        <!--等号按钮-->
        <Button
            android:id="@+id/equal"
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_marginLeft="2sp"
            android:layout_marginRight="2sp"
            android:text="@string/equal"
            android:textColor="#fefcff"
            android:textSize="30sp"
            android:layout_weight="1"
            android:background="#f8952d"/>

    </LinearLayout>

</LinearLayout>
```

（二）MainActivity:

```
package com.example.zdm.calculator;

import android.support.annotation.NonNull;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.support.v7.app.AppCompatDelegate;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

import java.util.Arrays;

public class MainActivity extends AppCompatActivity implements
View.OnClickListener
{
    Button btn_0;
    Button btn_1;
    Button btn_2;
    Button btn_3;
    Button btn_4;
    Button btn_5;
    Button btn_6;
    Button btn_7;
    Button btn_8;
    Button btn_9;
    Button btn_add;          // +
    Button btn_subtract;     // -
    Button btn_multiply;     // ×
    Button btn_divide;       // ÷
    Button btn_mod;          // %
    Button btn_point;        // 小数点
```

```
    Button btn_equal;       // =
    Button btn_clear;       // 清除
    Button btn_del;         // 删除
    Button btn_status;      // 正负状态
    Button btn_leftbracket; // (
    Button btn_rightbracket;// )
    Button btn_power;       // 乘方
    Button btn_factorial;   // 阶乘
    Button btn_ans;         // 上一次的答案
    private TextView textInput;   // 输入框
    private TextView textOutput;  // 输出框
    private StringBuilder pending = new StringBuilder(); // 输入框的
StringBuilder
    private String answer;  // 答案

    // 初始化计算器
    private void initView()
    {
        // 找到对应的按钮 id
        btn_0 = (Button) findViewById(R.id.zero);
        btn_1 = (Button) findViewById(R.id.one);
        btn_2 = (Button) findViewById(R.id.two);
        btn_3 = (Button) findViewById(R.id.three);
        btn_4 = (Button) findViewById(R.id.four);
        btn_5 = (Button) findViewById(R.id.five);
        btn_6 = (Button) findViewById(R.id.six);
        btn_7 = (Button) findViewById(R.id.seven);
        btn_8 = (Button) findViewById(R.id.eight);
        btn_9 = (Button) findViewById(R.id.nine);
        btn_add = (Button) findViewById(R.id.add);
        btn_subtract = (Button) findViewById(R.id.subtract);
        btn_multiply = (Button) findViewById(R.id.multiply);
        btn_divide = (Button) findViewById(R.id.divide);
        btn_mod = (Button) findViewById(R.id.mod);
        btn_point = (Button) findViewById(R.id.point);
        btn_equal = (Button) findViewById(R.id.equal);
        btn_clear = (Button) findViewById(R.id.AC);
        btn_del = (Button) findViewById(R.id.DEL);
        btn_status = (Button) findViewById(R.id.status);
        btn_leftbracket = (Button) findViewById(R.id.leftbracket);
        btn_rightbracket = (Button) findViewById(R.id.rightbracket);
        btn_power = (Button) findViewById(R.id.power);
        btn_factorial = (Button) findViewById(R.id.factorial);
        btn_ans = (Button) findViewById(R.id.ANS);

        textInput = (TextView) findViewById(R.id.textInput);
        textOutput = (TextView) findViewById(R.id.textOutput);

        // 设置点击监听器
       btn_0.setOnClickListener(this);
        btn_1.setOnClickListener(this);
        btn_2.setOnClickListener(this);
        btn_3.setOnClickListener(this);
        btn_4.setOnClickListener(this);
        btn_5.setOnClickListener(this);
        btn_6.setOnClickListener(this);
        btn_7.setOnClickListener(this);
        btn_8.setOnClickListener(this);
        btn_9.setOnClickListener(this);
```

```
        btn_add.setOnClickListener(this);
        btn_subtract.setOnClickListener(this);
        btn_multiply.setOnClickListener(this);
        btn_divide.setOnClickListener(this);
        btn_mod.setOnClickListener(this);
        btn_point.setOnClickListener(this);
        btn_equal.setOnClickListener(this);
        btn_clear.setOnClickListener(this);
        btn_del.setOnClickListener(this);
        btn_status.setOnClickListener(this);
        btn_leftbracket.setOnClickListener(this);
        btn_rightbracket.setOnClickListener(this);
        btn_power.setOnClickListener(this);
        btn_factorial.setOnClickListener(this);
        btn_ans.setOnClickListener(this);

        answer = "0.0"; // 初始化答案
    }

    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        // 创建活动时初始化界面
       initView();
    }

    @Override
    public void onClick(View v)
    {
        int last = 0;
        if(pending.length() != 0)
        {
            // codePointAt()方法返回的是码点的十进制值
           last = pending.codePointAt(pending.length() - 1);
        }

        switch(v.getId())
        {
            case R.id.zero:
            {
                judge3();
                pending = pending.append("0");
                textInput.setText(pending);
                break;
            }
            case R.id.one:
            {
                judge3();
                pending = pending.append("1");
                textInput.setText(pending);
                break;
            }
            case R.id.two:
            {
                judge3();
                pending = pending.append("2");
                textInput.setText(pending);
                break;
```

```
            }
            case R.id.three:
            {
                judge3();
                pending = pending.append("3");
                textInput.setText(pending);
                break;
            }
            case R.id.four:
            {
                judge3();
                pending = pending.append("4");
                textInput.setText(pending);
                break;
            }
            case R.id.five:
            {
                judge3();
                pending = pending.append("5");
                textInput.setText(pending);
                break;
            }
            case R.id.six:
            {
                judge3();
                pending = pending.append("6");
                textInput.setText(pending);
                break;
            }
            case R.id.seven:
            {
                judge3();
                pending = pending.append("7");
                textInput.setText(pending);
                break;
            }
            case R.id.eight:
            {
                judge3();
                pending = pending.append("8");
                textInput.setText(pending);
                break;
            }
            case R.id.nine:
            {
                judge3();
                pending = pending.append("9");
                textInput.setText(pending);
                break;
            }
            case R.id.add:
            {
                pending = pending.append("+");
                textInput.setText(pending);
                break;
            }
            case R.id.subtract:
            {
                pending = pending.append("-");
                textInput.setText(pending);
```

```
            break;
        }
        case R.id.multiply:
        {
            pending = pending.append("×");
            textInput.setText(pending);
            break;
        }
        case R.id.divide:
        {
            pending = pending.append("÷");
            textInput.setText(pending);
            break;
        }
        case R.id.mod:
        {
            pending = pending.append("%");
            textInput.setText(pending);
            break;
        }
        case R.id.point:
        {
            judge3();
            if (judge1())
            {
                pending = pending.append(".");
                textInput.setText(pending);
            }
            break;
        }
        case R.id.equal:
        {
            if ((pending.length() > 1))
            {
                InfixInToDuffix inf = new InfixInToDuffix();
                try {
                    String a = inf.toSuffix(pending);
                    answer = inf.dealEquation(a);
                } catch (Exception ex) {
                    answer = "出错";
                }
                textOutput.setText(answer);
                pending = pending.delete(0, pending.length());
                if (Character.isDigit(answer.charAt(0)))
                {
                    pending = pending.append(answer);
                }
            }
            break;
        }
        case R.id.AC:
        {
            pending = pending.delete(0, pending.length());
            textInput.setText(pending);
            textOutput.setText("");
            break;
        }
        case R.id.DEL:
        {
            if (pending.length() != 0)
```

```
                {
                    pending = pending.delete(pending.length() - 1,
pending.length()); // 删掉最后一个
                  textInput.setText(pending);
                }
                break;
            }
            case R.id.status:
            {
                if(textOutput.getText().length() != 0)
                {
                    String ans = textOutput.getText().toString();
                    double temp = Double.parseDouble(ans);
                    double trans = 0.0 - temp;
                    String newPending = String.valueOf(trans);
                    pending = pending.delete(0, pending.length());
                    pending.append(newPending);
                    textInput.setText(pending);
                    textOutput.setText("");
                }
                break;
            }
            case R.id.leftbracket:
            {
                if((last != '(') || (last <='0' && last >= '9'))
                {
                    pending = pending.append("(");
                    textInput.setText(pending);
                }
                break;
            }
            case R.id.rightbracket:
            {
                if((last >= '0' && last <= '9'|| last == ')' || last == '!' ||
last == '%')&&judge2()==1)
                {
                    pending = pending.append(")");
                    textInput.setText(pending);
                }
                break;
            }
            case R.id.power:
            {
                pending = pending.append("^");
                textInput.setText(pending);
                break;
            }
            case R.id.factorial:
            {
                pending = pending.append("!");
                textInput.setText(pending);
                break;
            }
            case R.id.ANS:
            {
                //pending = pending.delete(0, pending.length());
                pending.append(answer);
                textInput.setText(pending);
                break;
            }
```

```
            default:
            {
                break;
            }
        }
    }

    // 判断小数点，防止输入多个小数点
    private boolean judge1()
    {
        String a = "!^+-×÷%.";            // 计算器可输入的所有符号
        int[] b = new int[a.length()]; // 存储索引
        int max; // 最后一个符号的下标
        for (int i = 0; i < a.length(); i++) // 遍历一遍所有符号，找出所有符号出现的最后的一次索引
        {
            String c = "" + a.charAt(i);
            b[i] = pending.lastIndexOf(c);
        }
        Arrays.sort(b); // 升序排列
        if (b[a.length() - 1] == -1)
        {
            max = 0;
        } else {
            max = b[a.length() - 1];
        }
        if (pending.indexOf(".", max) == -1) { // 从 max 位置往后查找"."出现的下标，没有则返回-1
            return true;
        } else {
            return false;
        }
    }

    // 判断左右括号匹配
    private int judge2()
    {
        int a=0,b=0;
        for(int i = 0; i < pending.length(); i++){ // 遍历等式，计算左右括号的个数
           if(pending.charAt(i)=='(' ) {
                a++;
            }
            if(pending.charAt(i)==')' ) {
                b++;
            }
        }
        if(a == b)
            return 0;
        if(a > b) // 左括号大于右括号，此时可以继续输入右括号
           return 1;
        return 2;
    }

    // 如果已经计算过一次答案，那下一次输入数字的时候重新开始输入算式
    private void judge3()
    {
        if(textOutput.getText() != "")
        {
```

```
            textOutput.setText("");
            pending = pending.delete(0, pending.length());
        }
    }
}
```

（三）InfixInToDuffix：

```
package com.example.zdm.calculator;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class InfixInToDuffix
{
    // 使用集合定义好优先级
    private static final Map<Character, Integer>basic = new HashMap<>();
    static
    {
        basic.put('-', 1);
        basic.put('+', 1);
        basic.put('×', 2);
        basic.put('÷', 2);
        basic.put('%', 2);
        basic.put('^', 3);
        basic.put('!', 4);
        basic.put('(', 0);//在运算中（）的优先级最高，但是此处因程序中需要 故设置为0
    }

    // 中缀转后缀
    public String toSuffix(StringBuilder infix)
    {
        List<String> queue = new ArrayList<String>();        // 定义队列，用于存储数字以及最后的后缀表达式
        List<Character> stack = new ArrayList<Character>(); // 定义栈，用于存储转换过程的运算符，最后会弹空

        if(infix.charAt(0) == '-' || infix.charAt(0) == '.') // 如果算式开头是个负数或者是小数点，则在前面插入 0
        {
            infix.insert(0, '0');
        }

        char[] charArr = infix.substring(0, infix.length()).trim().toCharArray();   // 字符数组，用于拆分数字或符号，trim()用于清空字符串头尾空白符
        String standard = "!^×÷%+-()";    // 判定标准，将表达式中会出现的运算符写出来
        char ch = '&';                    // 在循环中用于保存字符数组的当前循环变量的，这里仅仅是初始化一个值
        int len = 0;                      // 用于记录字符长度 【例如 100*2,则记录的 len 为 3 到时候截取字符串的前三位就是数字】
        for(int i = 0; i < charArr.length; i++) // 迭代转换
        {
```

```
            ch = charArr[i]; //保存当前迭代变量
           if(Character.isDigit(ch)) // 当前读到的是数字
           {
                len++;
            }
            else if(ch == '.') // 当前读到的是小数点
           {
                len++;
            }
            else if(standard.indexOf(ch) != -1) // 当前读到的是符号并且在判定标
准中
           {
                if(len > 0) // 存在长度，则说明符号之前的部分为一个数字
               {
                    queue.add(String.valueOf(Arrays.copyOfRange(charArr, i -
len, i))); // 从数组中截取数字转化为整个字符串存储进队列
                   len = 0; // 长度清零
               }
                if(ch == '(') // 读到左括号
               {
                    stack.add(ch); // 左括号直接进栈
                   continue;       // 跳过本次循环找下一位
               }
                if(!stack.isEmpty()) // 栈不空，则判断当前运算符与栈顶运算符优先
级
               {
                    int size = stack.size() - 1; // 栈顶，栈最后一个元素的下标
                   boolean flag = false;         // 标志位
                   while(size >= 0 && ch == ')' && stack.get(size) != '(') //
当前为)且栈顶不为(
                    {
                        queue.add(String.valueOf(stack.remove(size))); // 循环
弹出栈顶符号直到找到(，ch 并未入栈，所以并未插入队列中
                       size--;
                        flag = true; // 设置标志位为 true，表明一直在取（）中的元
素
                    }
                    if(ch == ')' && stack.get(size) == '(') // 左右括号匹配
                   {
                        flag = true;
                    }
                    while (size >= 0 && !flag && basic.get(stack.get(size)) >=
basic.get(ch)) //若取得不是（）内的元素，并且当前栈顶元素的优先级>=当前对比元素 那就
出栈插入队列
                   {
                        queue.add(String.valueOf(stack.remove(size)));
//同样  此处也是 remove()方法，既能得到要获取的元素，也能将栈中元素移除掉
                       size--;
                    }
                }
                if(ch != ')') // 当前元素不是)且栈空
               {
                    stack.add(ch); // 符号入栈
                }
                else
                {
                    stack.remove(stack.size() - 1);
```

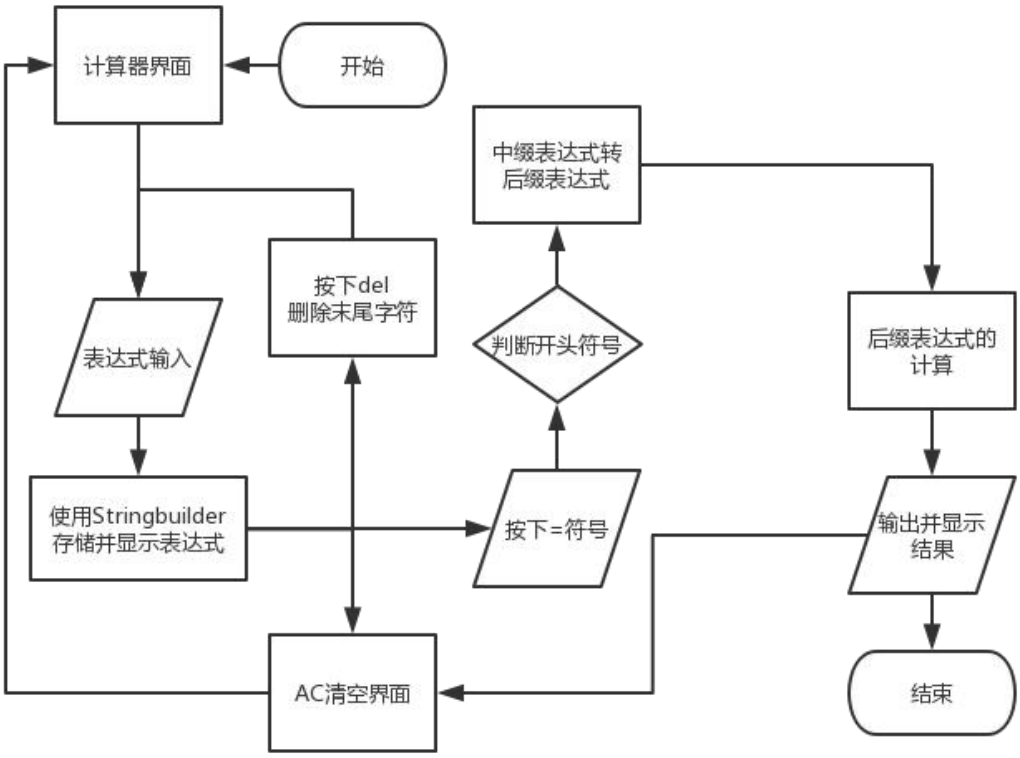
```
                }
            }

            if(i == charArr.length - 1) //如果已经走到了中缀表达式的最后一位
            {
                if(len > 0) //如果 len>0  就截取数字
                {
                    queue.add(String.valueOf(Arrays.copyOfRange(charArr, i - len+1, i+1)));
                }
                int size = stack.size() - 1; //size 表示栈内最后一个元素下标
                while (size >= 0) //一直将栈内符号全部出栈 并且加入队列中  【最终的后缀表达式是存放在队列中的，而栈内最后会被弹空】
                {
                    queue.add(String.valueOf(stack.remove(size)));
                    size--;
                }
            }
        }

        String a = queue.toString(); // 用 toString()函数之后，List 转化成用,分隔的数组
        return a.substring(1, a.length()-1);
    }

    // 处理等号运算(后缀表达式的计算)
    public String dealEquation(String equation)
    {
        String [] arr = equation.split(", "); //根据，拆分后缀表达式字符串
        List<String> list = new ArrayList<String>();

        for(int i = 0; i < arr.length; i++)
        {   //此处就是上面说的运算过程，因为 list.remove 的缘故导致 remove 元素后面的下标变化，所以取出最后一个数个最后两个数都是 size-2
            int size = list.size(); // 指向找到的第一个符号
            switch (arr[i]) {
                case "+": double a = Double.parseDouble(list.remove(size-2)) + Double.parseDouble(list.remove(size-2)); list.add(String.valueOf(a)); break;
                case "-": double b = Double.parseDouble(list.remove(size-2)) - Double.parseDouble(list.remove(size-2)); list.add(String.valueOf(b)); break;
                case "×": double c = Double.parseDouble(list.remove(size-2)) * Double.parseDouble(list.remove(size-2)); list.add(String.valueOf(c)); break;
                case "÷": double d = Double.parseDouble(list.remove(size-2)) / Double.parseDouble(list.remove(size-2)); list.add(String.valueOf(d)); break;
                case "%": double e = Double.parseDouble(list.remove(size-1)); list.add(String.valueOf(e/100.0)); break; // %运算为百分号运算
                case "^": double f = Math.pow(Double.parseDouble(list.remove(size-2)), Double.parseDouble(list.remove(size-2))); list.add(String.valueOf(f)); break;
                case "!": double g = Double.parseDouble(list.remove(size-1)); list.add(String.valueOf(factorial(g))); break; // 阶乘为独立计算，所以前面只有一个数字
                default: list.add(arr[i]); break; //如果是数字，直接放进 list 中
            }
        }

        return list.size() == 1 ? list.get(0) : "运算失败" ; // 最后剩下答案
    }
```

```
    // 计算阶乘
    private double factorial(double x)
    {
        double ans = 1.0;
        if(x == 0.0 || x == 1.0) return ans;
        else {
            for(int i = 1; i <= x; i++)
            {
                ans = ans * i;
            }
            return ans;
        }
    }
}
```

（四）string.xml：

```
<resources>
    <!--做到统一添加字符串-->
    <string name="app_name">Calculator</string>
    <string name="app_names">不科学计算器</string>
    <string name="AC">AC</string>
    <string name="DEL">DEL</string>
    <string name="ANS">ANS</string>
    <string name="status">+/-</string>
    <string name="divide">÷</string>
    <string name="leftbracket">(</string>
    <string name="seven">7</string>
    <string name="eight">8</string>
    <string name="nine">9</string>
    <string name="multiply">×</string>
    <string name="rightbracket">)</string>
    <string name="four">4</string>
    <string name="five">5</string>
    <string name="six">6</string>
    <string name="subtract">-</string>
    <string name="power">^</string>
    <string name="one">1</string>
    <string name="two">2</string>
    <string name="three">3</string>
    <string name="add">+</string>
    <string name="factorial">x!</string>
    <string name="zero">0</string>
    <string name="point">.</string>
    <string name="mod">%</string>
    <string name="equal">=</string>
</resources>
```

六、软件操作流程


计算器界面
开始
中缀表达式转
后缀表达式
按下del
删除末尾字符
表达式输入
判断开头符号
后缀表达式的
计算
使用Stringbuilder
存储并显示表达式
按下=符号
输出并显示
结果
AC清空界面
结束

七、难点（或遇到的问题）和解决方案

这次计算器的设计与实现作业难点包括界面设计以及算法设计上面，由于是初次接触 android studio 设计，所以界面的难度也包括对 android studio 的上手难度。

1.布局：布局上，要让计算器的界面可以适配所有类型的手机
解决方式：使用线性布局设置权重，使控件自适应所有的屏幕

2.算法：最基本的中缀转后缀的算法仅仅适用于都是整数且都是个位数的情况，要使计算器可支持实数。
解决方案：若遇到第一个数字为负数，则直接在等式最前面添加一个 0；在中缀转后缀的时候，当读取完一串数字后，再把它们合并成一整个字符串，形成一个完整的数字

3.运算符的优先级：只有进行正确的优先级设定，才能算出正确的答案

4.括号匹配：左括号与右括号数量不匹配，会算出错误答案
解决方案：编写一个函数进行左右括号的统计，当左右括号相等后，不能再输入右括号

5.运算精度：小数运算容易出现精度问题

八、不足之处

1.计算器仅能支持最基本的四册运算，未能支持更多的科学计算
2.只设计了一个界面，没有进行多界面的设计
3.阶乘的运算容易超出运算范围
4.并不能保存历史纪录，仅仅能查看上一次运算的答案
5.小数精度处理上还有一些 bug，会出现计算上的误差

九、今后的设想

这次的计算器开发，让我学习到如何对 Android app 进行 UI 界面的设计与排布，同时对计算器的算法原理有了更深的理解，补充了曾经对中缀转后缀算法的空白部分。同时这也是我做出来的第一个可以日常使用的 Android app，这也增添了我对安卓的兴趣。

在这次计算器的编写中我也发现了自己编写的计算器尚且有许多不足之处，下一步我的目标将是把我的简单计算器变成为较为复杂的计算器，算是针对上一次的计算器的一次大改进。这种循循渐进的感觉让我在移动应用开发的学习上方向感更加明确。

在这次 android studio 的学习以及代码的编写中我也感觉自己的能力又有了一些提升，不过未来的路也还很长，我还需要更努力地学习！