计算器的设计与实现

1 软件名称

1.1 名称：简易计算器

1.2 完成人：钟昌宏

1.3 学号：20152100127

1.4 完成时间：2017 年 10 月 12 日

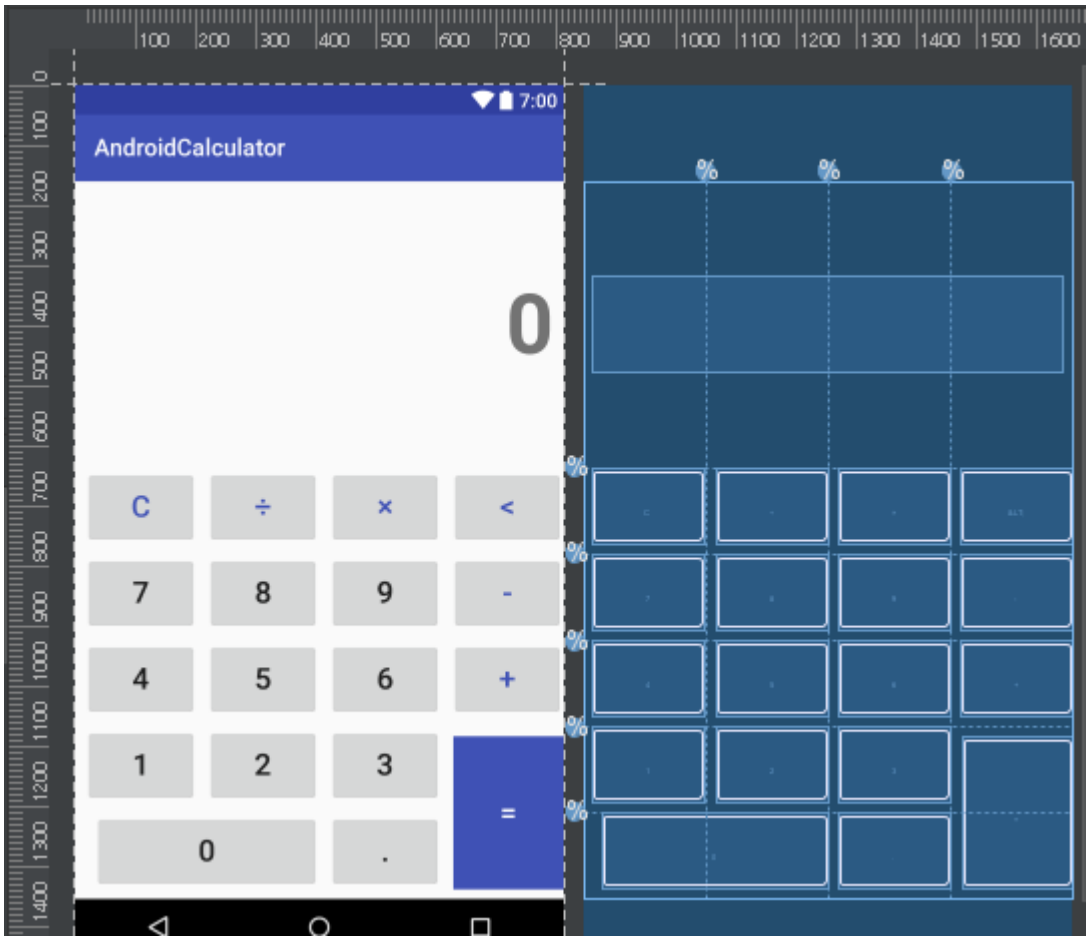
2 软件内容简介

2.1 功能

简单的计算器，实现整数、浮点数的加减乘除，能够实现输入一长条表达式再计算，还能够计算长达 20 位数之间的运算，最终的答案由科学计数法显示。

2.2 界面

有加减乘除按钮，退格按钮，清空按钮等等，计算表达式和结果显示在屏幕上半部分。



3 界面设计

3.1 布局设计

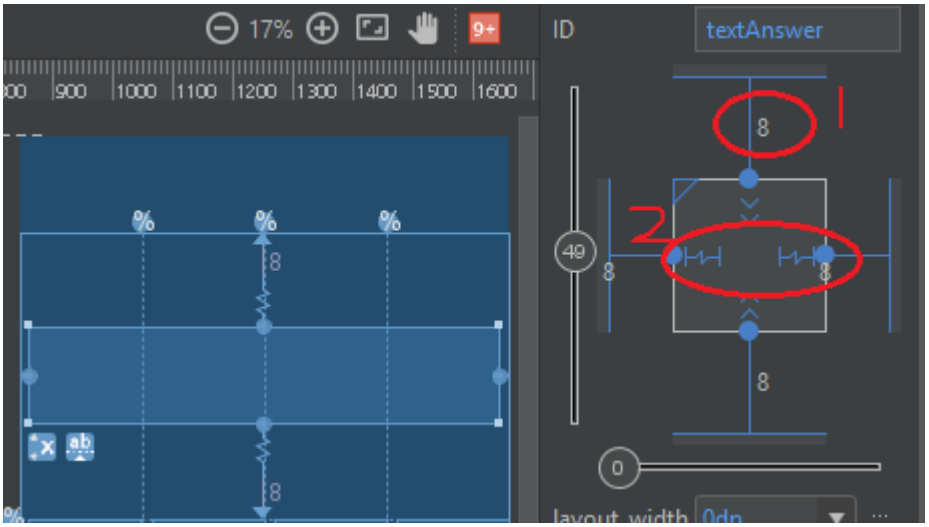
本 APP 用了约束布局（Constraint Layout），一开始先确定面板大致分为 6 行 4 列，所以要放置 5 条横分界线和 3 条竖分界线，并将其设置成百分比模式。

```
<android.support.constraint.Guideline
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/guideline"
    app:layout_constraintGuide_percent="0.25"
    android:orientation="vertical"
    tools:layout_editor_absoluteY="0dp"
    tools:layout_editor_absoluteX="103dp" />
```

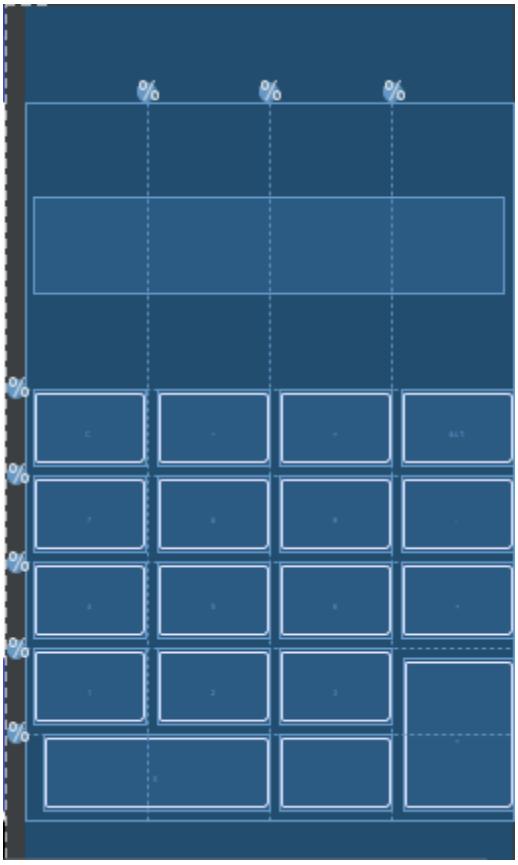
在代码中设置该属性，设置好布局的百分比，控件到时候就由这些分界线来约束。

3.2 控件布置

APP 中只用到两种控件，TextView 和 Button，将控件拖动到界面中，利用约束布局的性质，将控件四边由设置好的分界线来约束。



如 TextView 的设计，1 处是设置与边界的距离，约束布局会以此约束控件的摆放。2 可以设置控件的宽度扩展策略，此时是尽量扩展。按照这样的策略，将所有按钮都紧贴分界线摆放，给所有控件添加约束。



4 代码设计

4.1 按钮属性

```
<Button
    android:id="@+id/button0ne"
    android:layout_height="0dp"
    android:layout_width="0dp"
    android:onClick="click0ne"
    android:text="1"
    android:textSize="25sp"
    app:layout_constraintTop_toTop0f="@+id/guideline8"
    android:layout_marginTop="0dp"
    app:layout_constraintRight_toLeft0f="@+id/guideline"
    android:layout_marginRight="0dp"
    android:layout_marginBottom="8dp"
    app:layout_constraintBottom_toTop0f="@+id/guideline9"
    android:layout_marginLeft="8dp"
    app:layout_constraintVertical_bias="0.0"
    app:layout_constraintLeft_toLeft0f="parent"
    app:layout_constraintHorizontal_bias="1.0" />
```

这是按钮“1”的代码，1 处可以设置与之关联的函数，“clickOne”是已经写好的函数；2 处设置按钮上显示文字的大小，单位一般是 sp。

4.2 数字按钮

```
public void clickNumber(int n)
{
    if(expression.length()>=80)
        return;
    expression.append(n);
    ans.setTextSize(textSize[expression.length()]);
    ans.setText(expression.toString());
}
```

这是按下数字按钮后的代码，判断完表达式长度没问题后，直接添加到表达式后面。

4.3 运算符按钮

```
public void changeSign(char c)
{
    String temp = expression.toString();
    if(temp.equals("")) {
        if(c == '*' || c== '/')
            return;
    }
    if(temp.endsWith("+") || temp.endsWith("-") || temp.endsWith("*") || temp.endsWith("/") || temp.endsWith(".")
        expression.delete(expression.length()-1,expression.length());
    if(expression.length()>=80)
        return;
    expression.append(c);
    ans.setTextSize(textSize[expression.length()]);
    ans.setText(expression.toString());
}
```

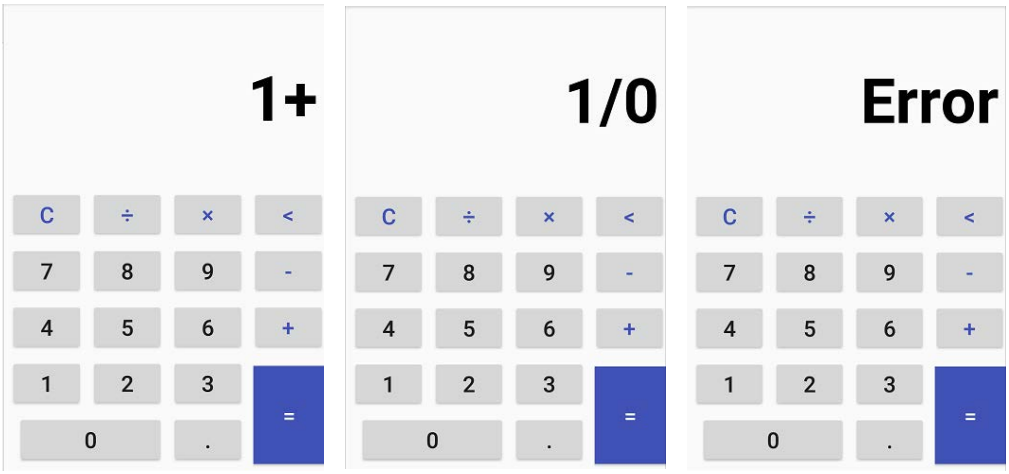
用户的输入是很多种情况的，要考虑周全。首先，用户可能要计算负数间的运算，那么就要允许一开始可以输入负号，不能输入乘号除号；第二，用户可能输错运算符，再按一次别的运算符应该要更改原来的运算符，这样才人性化。

4.4 计算

运算器的运算是最核心的部分，运算出错、运算不准确、数字太大出 bug、结果太长无法在屏幕显示完，都是不允许出现的情况。《编译原理》有一项作业，“中缀表达式转后缀表达式”，我写完后，正好就把这个程序打包起来用作计算器 APP 的运算，中缀转后缀，再直接对后缀运算即可。

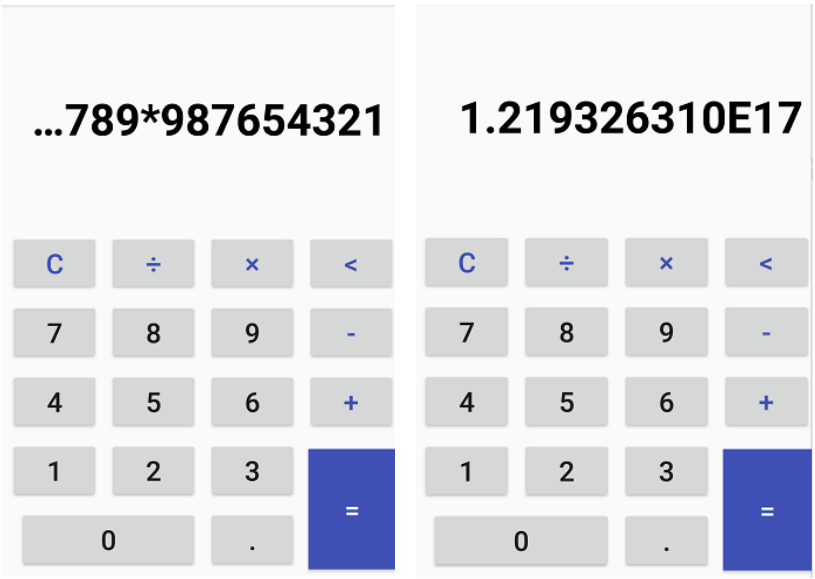
```
import Main.PostfixExp;
```

写这个作业的时候考虑了很多情况，思考较为严密，比如用户恶意输入，除数为 0 等，甚至还包括有括号的处理，所以还没遇见计算出 bug 的情况。



如果输入无法计算的表达式，会显示 Error。

计算器还有很重要的一点是，计算的结果要科学地显示出来，如果得到的数字太大无法在 TextView 里显示，需要用科学计数法。



以上是 123456789×987654321 的显示，保留 9 位小数以保证能够在屏幕里显示。

5 软件操作流程

软件操作没有什么难度，就和普通的计算器一样，输错了可以按退格或清空，按等号计算，得到的结果可以继续运算。

6 难点和解决方案

6.1 布局管理

第一次做 Android 的布局设计，用 Android Studio 的布局管理，实在是太难上手了。一开始尝试了约束布局（Constraint Layout）失败，因为不懂得使用百分比分隔线，做出来的界面在手机里会出现超出屏幕的问题。后来使用表格布局（Table Layout），使用感觉良好，但是并不能实现控件竖着占两个格子，不能够满足我的需要，于是放弃。接着使用了网格布局（Grid Layout），布局一切完美，但是到手机上运行会出现超出屏幕问题，无法解决，据说是由于 Grid Layout 在 API 21 以下等级下都会出现这种问题，于是放弃。最后又用回了约束布局，使用了百分比分隔线把问题解决，基本了解约束布局的用法。

6.2 计算

上面说到了，计算部分代码是用了《编译原理》的作业加以完善得到的。原本的程序里可以解决有括号的情况，处理恶意输入的情况，所以用到计算器 APP 里难度降低了，应该问题不大。这个作业我真的是思考了很久怎么去解决这些问题。整个计算器 APP 都是完全独立完成，每一部分都是亲自设计。

7 优点和不足之处

7.1 优点

界面简洁优雅，表达式的输入和计算结果的显示都十分人性化，超过 20 位数也能计算。

7.2 不足之处

颜色不够丰富；数字的显示太直接。

8 今后的设想

8.1 横屏

横屏后会自动变成功能更强大的科学计算器。

8.2 数字显示效果

计算出结果后，数字以滑动的效果出现在 TextView 里。