

Project2 实验报告

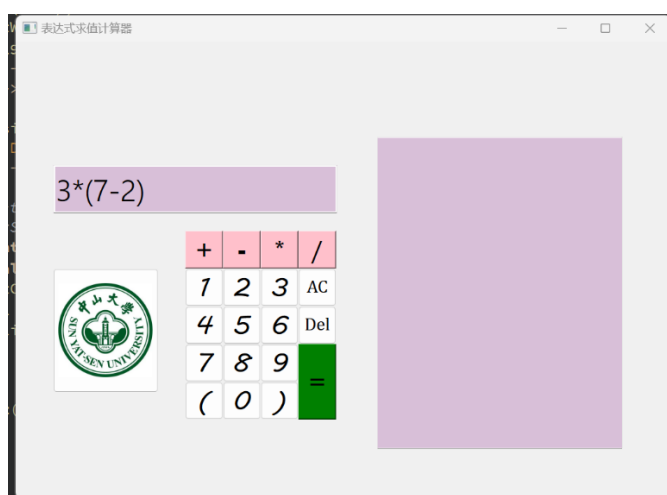
学号 22336313 姓名 郑鸿鑫

1、 程序功能简要说明：

这是一个用 QtCreator 编写的四则运算表达式计算器，有交互式的个性化界面，而且可以正常计算语法正确的不含变量的整数表达式，实现对算数混合表达式的求值。程序允许计算过程和结果出现浮点数，而且能对分母为 0 的情况进行识别报错。

2、 程序运行截图，包括计算功能演示、部分实际运行结果展示、命令行或交互式界面效果等。

输入样例：



输出样例：



说明：界面中左边的方框显示用户输入的表达式，右侧的方框用于演示该表达式在程序中计算的过程。数字与运算符都与日常计算器功能相同。Del 按键用于删除一个字符，可以在输入出错时进行修改，AC 作为 clear 的功能用于清空两个紫色文本框内的数据，当表达式输入完成后，按下“=”即可得到结果。需要注意的是：其中的“-”只能做为减法使用，即需要两个操作数，如果将其用于代表负号，程序将无法正常使用。

3、部分关键代码及其说明。

关键代码 1 ()：

```
Widget::Widget(QWidget *parent)
    : QWidget(parent)
    , ui(new Ui::Widget)
{
    ui->setupUi(this);
    this -> setWindowTitle("表达式求值计算器");
    ui -> one_19 -> setStyleSheet("background:green");
    ui -> plus -> setStyleSheet("background:pink");
    ui -> sub -> setStyleSheet("background:pink");
    ui -> multi -> setStyleSheet("background:pink");
    ui -> division -> setStyleSheet("background:pink");
    QIcon con("D:\\26618\\Downloads\\Browser_downloads\\SYSU.jpg");
    ui -> sysu -> setIcon(con);

    // QString styleSheet = "QWidget { background-color: lightblue; }";
    // this->setStyleSheet(styleSheet);
    QColor lightpink(216,191,216); // 定义浅紫色
    QPalette palette = ui -> show1 -> palette();
    palette.setColor(QPalette::Base, lightpink); // 设置背景色为浅紫色
    ui -> show1 -> setPalette(palette);
    ui -> mainlineEdit -> setPalette(palette);
}
```

说明：这部分的代码用于控制界面的名称，颜色，还有按键的颜色。

```
void Widget::on_one_clicked()
{
    expression += "1";
    ui -> mainlineEdit -> setText(expression);
}
```

说明：这是按键“1”对应的槽函数，它的作用是让表达式后面附上“1”这个字符，同时显示在程序左侧的文本框上。（对于其他几个输入按键，都是相似的代码，此处不再给出）

```
void Widget::on_clear_clicked()
{
    expression.clear();
    ui -> mainlineEdit -> clear();
    ui -> show1 -> clear();
}

void Widget::on_delete_2_clicked()
{
    expression .chop(1);
    ui -> mainlineEdit -> setText(expression);
}
```

说明：AC 和 Del 按键对应的槽函数，功能在上文已经解释。

```

143 //void string::parse_33_ahbashed()
144 {
145     QThread::connect(this, &start);
146     QString char_n = opt1;
147     char opt1[] = {' '};
148     int i = 1;
149     while (true)
150     {
151         double temp = x_num_read_error * i;
152         QString error = QString::fromStdString(temp);
153         //Call for function "strcpy" is insecure as it does not provide bounds
154         while (opt1[i] != '\0') { x_opt_empty[i] = true;
155             {
156                 if (opt1[i] == '0' && opt1[i] == '0')
157                     temp = temp * 10 + (opt1[i] - '0');
158                 else if (opt1[i] < '0') { opt1[i] = '0';
159                     x_num_posopt(temp);
160                     show1 = "0";
161                     show2 = QString::number(temp);
162                     show3 = "  正负数符号错误";
163                     ui->show1->setText(show1);
164                     ui->show2->setText(show2);
165                     temp = 1;
166                 }
167             }
168             else if (x_opt_empty[i] == true) { Priority(opt1[i]) > Priority(x_opt_top[i]) { x_opt_top[i] = '0' && opt1[i] == '0'; }
169                 {
170                     x_opt_posopt(opt1[i]);
171                     show1 = "0";
172                     show2 = opt1[i];
173                     show3 = " 正负数符号错误";
174                     ui->show1->setText(show1);
175                     ui->show2->setText(show2);
176                     continue;
177                 }
178                 if (x_opt_top[i] == '0' && opt1[i] == '0')
179                 {
180                     x_opt_posopt('0');
181                     show1 = "0"; // 输出正数符号错误
182                     ui->show1->setText(show1);
183                     temp = 1;
184                     continue;
185                 }
186                 if (Priority(opt1[i]) < Priority(x_opt_top[i]) || (opt1[i] == '0' && x_opt_top[i] != '0') || (opt1[i] != '0' && x_opt_empty[i] == true))
187                 {
188                     char ch = x_opt_top[i];
189                     show1 = "0";
190                     show2 = ch;
191                     show3 = " 正负数符号错误";
192                     ui->show1->setText(show1);
193                     ui->show2->setText(show2);
194                     x_opt_posopt(ch);
195                     continue;
196                 }
197                 else { i++;
198                     num1 = x_num_top[i];
199                     show1 = "0";
200                     show2 = QString::number(num1);
201                     show3 = "  输出数符错误";
202                     ui->show1->setText(show1);
203                     ui->show2->setText(show2);
204                     x_num_posopt(i);
205                     num2 = x_num_top[i];
206                     show1 = "0";
207                     show2 = QString::number(num2);
208                     show3 = "  输出数符错误";
209                     ui->show1->setText(show1);
210                     ui->show2->setText(show2);
211                     x_num_posopt(i);
212                     num3 = num1 + num2;
213                     show1 = "0";
214                     show2 = QString::number(num1+num2);
215                     show3 = "  正负数符号错误";
216                     ui->show1->setText(show1);
217                     ui->show2->setText(show2);
218                     continue;
219                 }
220                 else { i++;
221                     num1 = x_num_top[i];
222                     show1 = "0";
223                     show2 = QString::number(num1);
224                     show3 = "  输出数符错误";
225                     ui->show1->setText(show1);
226                     ui->show2->setText(show2);
227                     x_num_posopt(i);
228                     num2 = num1 + num2;
229                     show1 = "0";
230                     show2 = QString::number(num1+num2);
231                     show3 = "  正负数符号错误";
232                     ui->show1->setText(show1);
233                     ui->show2->setText(show2);
234                     continue;
235                 }
236             }
237         }
238     }
239     if (error) ui->mainTextEdit->setText("Error");
240     while (ui->mainTextEdit->toPlainText() != QString::number(x_num_top[i]))
241         expression.clear();
242     ui->show1->setText(char_ch);
243     while (true)
244     {
245         case '0':
246             return 0;
247         case '+':
248             return 1;
249         case '-':
250             return 1;
251         case '*':
252             return 1;
253         case '/':
254             return 1;
255         case '^':
256             return 1;
257         case '%':
258             return 1;
259         case '(':
260             return 1;
261         case ')':
262             return 1;
263         case ' ':
264             return 1;
265         case '=':
266             return 1;
267         case '+':
268             return 1;
269         case '-':
270             return 1;
271         case '*':
272             return 1;
273         case '/':
274             return 1;
275         case '^':
276             return 1;
277         case '%':
278             return 1;
279         case '(':
280             return 1;
281         case ')':
282             return 1;
283         case ' ':
284             return 1;
285         case '=':
286             return 1;
287         case '+':
288             return 1;
289         case '-':
290             return 1;
291         case '*':
292             return 1;
293         case '/':
294             return 1;
295         case '^':
296             return 1;
297         case '%':
298             return 1;
299         case '(':
300             return 1;
301         case ')':
302             return 1;
303         case ' ':
304             return 1;
305         case '=':
306             return 1;
307         case '+':
308             return 1;
309         case '-':
310             return 1;
311         case '*':
312             return 1;
313         case '/':
314             return 1;
315         case '^':
316             return 1;
317         case '%':
318             return 1;
319         case '(':
320             return 1;
321         case ')':
322             return 1;
323         case ' ':
324             return 1;
325         case '=':
326             return 1;
327         case '+':
328             return 1;
329         case '-':
330             return 1;
331         case '*':
332             return 1;
333         case '/':
334             return 1;
335         case '^':
336             return 1;
337         case '%':
338             return 1;
339         case '(':
340             return 1;
341         case ')':
342             return 1;
343         case ' ':
344             return 1;
345         case '=':
346             return 1;
347         case '+':
348             return 1;
349         case '-':
350             return 1;
351         case '*':
352             return 1;
353         case '/':
354             return 1;
355         case '^':
356             return 1;
357         case '%':
358             return 1;
359         case '(':
360             return 1;
361         case ')':
362             return 1;
363         case ' ':
364             return 1;
365         case '=':
366             return 1;
367         case '+':
368             return 1;
369         case '-':
370             return 1;
371         case '*':
372             return 1;
373         case '/':
374             return 1;
375         case '^':
376             return 1;
377         case '%':
378             return 1;
379         case '(':
380             return 1;
381         case ')':
382             return 1;
383         case ' ':
384             return 1;
385         case '=':
386             return 1;
387         case '+':
388             return 1;
389         case '-':
390             return 1;
391         case '*':
392             return 1;
393         case '/':
394             return 1;
395         case '^':
396             return 1;
397         case '%':
398             return 1;
399         case '(':
400             return 1;
401         case ')':
402             return 1;
403         case ' ':
404             return 1;
405         case '=':
406             return 1;
407         case '+':
408             return 1;
409         case '-':
410             return 1;
411         case '*':
412             return 1;
413         case '/':
414             return 1;
415         case '^':
416             return 1;
417         case '%':
418             return 1;
419         case '(':
420             return 1;
421         case ')':
422             return 1;
423         case ' ':
424             return 1;
425         case '=':
426             return 1;
427         case '+':
428             return 1;
429         case '-':
430             return 1;
431         case '*':
432             return 1;
433         case '/':
434             return 1;
435         case '^':
436             return 1;
437         case '%':
438             return 1;
439         case '(':
440             return 1;
441         case ')':
442             return 1;
443         case ' ':
444             return 1;
445         case '=':
446             return 1;
447         case '+':
448             return 1;
449         case '-':
450             return 1;
451         case '*':
452             return 1;
453         case '/':
454             return 1;
455         case '^':
456             return 1;
457         case '%':
458             return 1;
459         case '(':
460             return 1;
461         case ')':
462             return 1;
463         case ' ':
464             return 1;
465         case '=':
466             return 1;
467         case '+':
468             return 1;
469         case '-':
470             return 1;
471         case '*':
472             return 1;
473         case '/':
474             return 1;
475         case '^':
476             return 1;
477         case '%':
478             return 1;
479         case '(':
480             return 1;
481         case ')':
482             return 1;
483         case ' ':
484             return 1;
485         case '=':
486             return 1;
487         case '+':
488             return 1;
489         case '-':
490             return 1;
491         case '*':
492             return 1;
493         case '/':
494             return 1;
495         case '^':
496             return 1;
497         case '%':
498             return 1;
499         case '(':
500             return 1;
501         case ')':
502             return 1;
503         case ' ':
504             return 1;
505         case '=':
506             return 1;
507         case '+':
508             return 1;
509         case '-':
510             return 1;
511         case '*':
512             return 1;
513         case '/':
514             return 1;
515         case '^':
516             return 1;
517         case '%':
518             return 1;
519         case '(':
520             return 1;
521         case ')':
522             return 1;
523         case ' ':
524             return 1;
525         case '=':
526             return 1;
527         case '+':
528             return 1;
529         case '-':
530             return 1;
531         case '*':
532             return 1;
533         case '/':
534             return 1;
535         case '^':
536             return 1;
537         case '%':
538             return 1;
539         case '(':
540             return 1;
541         case ')':
542             return 1;
543         case ' ':
544             return 1;
545         case '=':
546             return 1;
547         case '+':
548             return 1;
549         case '-':
550             return 1;
551         case '*':
552             return 1;
553         case '/':
554             return 1;
555        
```

说明：这是“=”对饮的槽函数，对于表达式求值的计算就在这个函数中实现。其中用到两个栈来实现，当读到的字符为数字时，继续扫描直到解析出整个数字后将其压入操作数栈，当读到的不是数字且运算符栈非空时，让其与栈顶的操作符比较优先级，优先级比栈顶高则入栈，相等则脱括号，优先级比栈顶低则从操作数栈弹出两个操作数进行计算，并将结果压入操作数栈，最后将操作数栈栈顶的元素展示出来。每次出栈入栈的操作都会演示在右侧的方框中。

4、 程序运行方式简要说明。

用户在运行程序时，通过按键进行输入表达式，表达式必须是正确规范的式子，最后键入“=”得到结果。

注意：用键盘进行输入时，左侧的方框也会显示出来，但其实没有真正的输入到程序中，所以只能用界面中的按键进行输入！