**C++ 语 言 程 序 设 计**

实

验

报

告

实 验 四

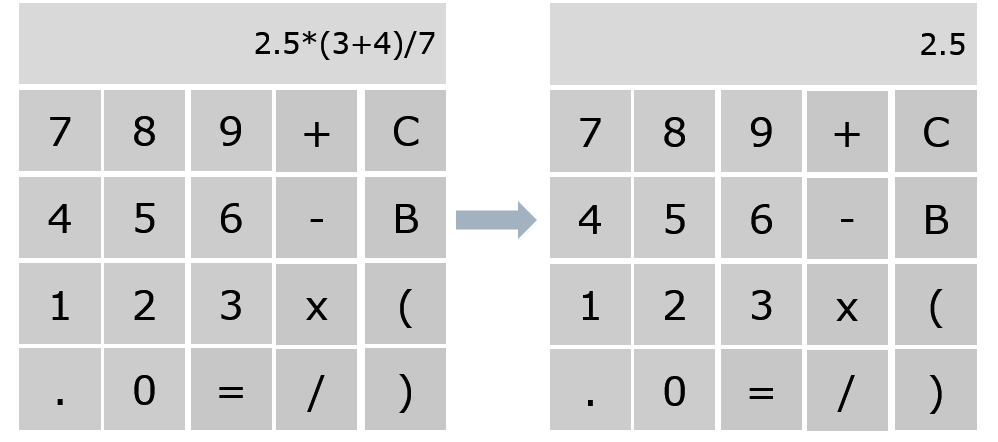
姓名： 葛旭

学号： 190320517

班级： 自动化五班

**一 实验项目**

* 1. 封装实验二的表达式计算功能到一个类中。
  2. 编写基于win32编程的GUI界面，实现计算式的交互式输入与结果显示：

****

3. 实现功能

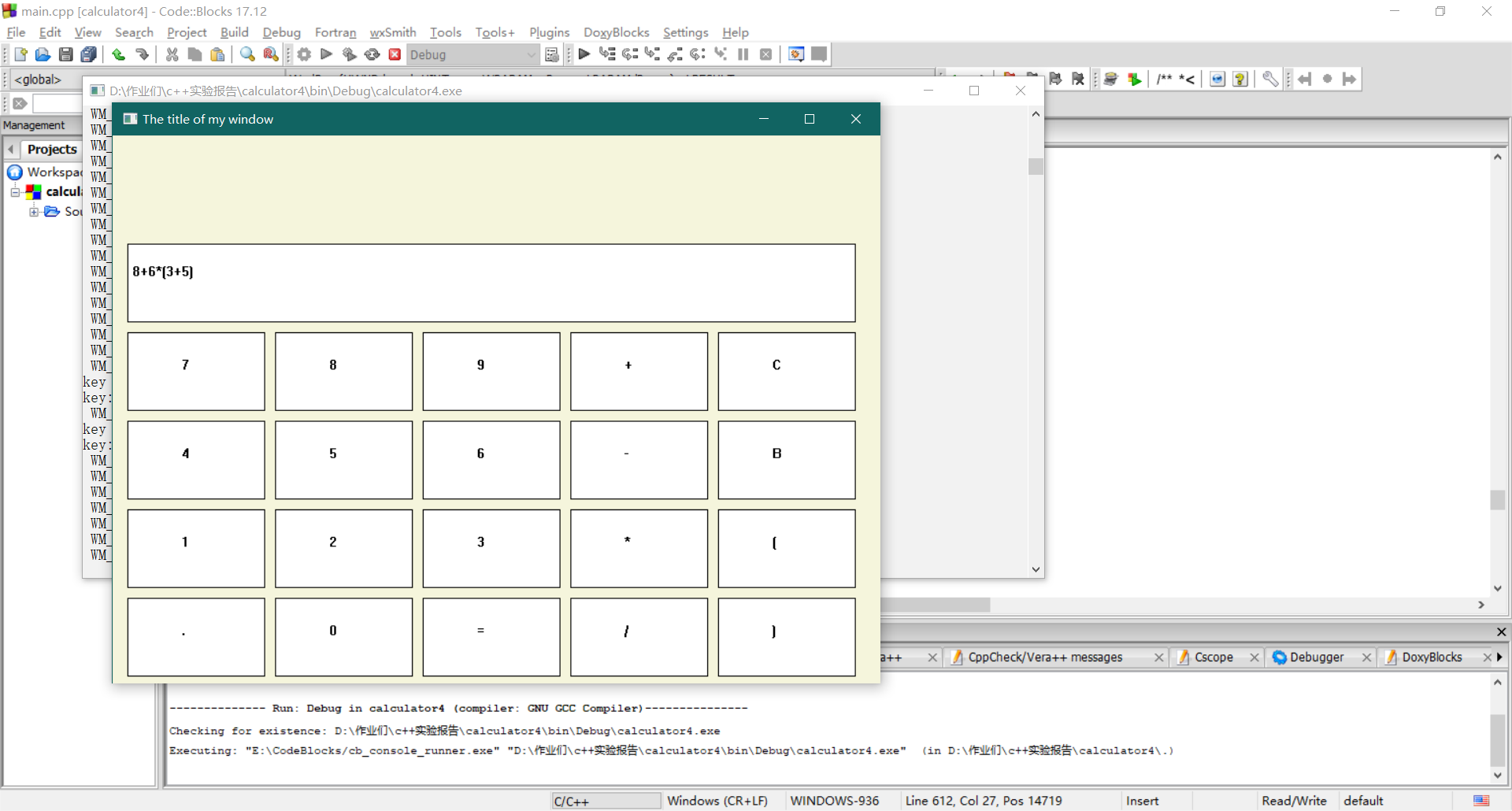
1. 点击数字、运算符号或小数点，上面显示框中即时显示
2. 点击 = ，显示框输出运算结果
3. 点击 C ，清空显示框
4. 点击 B ，实现输入显示的内容退格
5. 鼠标移至数字或符号的按键上时，按键变色，移开后恢复
6. 鼠标点击时，左键按下，按键颜色变深，左键弹起，恢复。
7. 鼠标点击时，不响应右键（按下鼠标右键后，程序界面无反应）
8. 支持键盘输入运算表达式
9. 键盘输入时，非界面上的数字或符号，则不响应
10. 本程序不考虑错误处理，默认输入的表达式为正确的

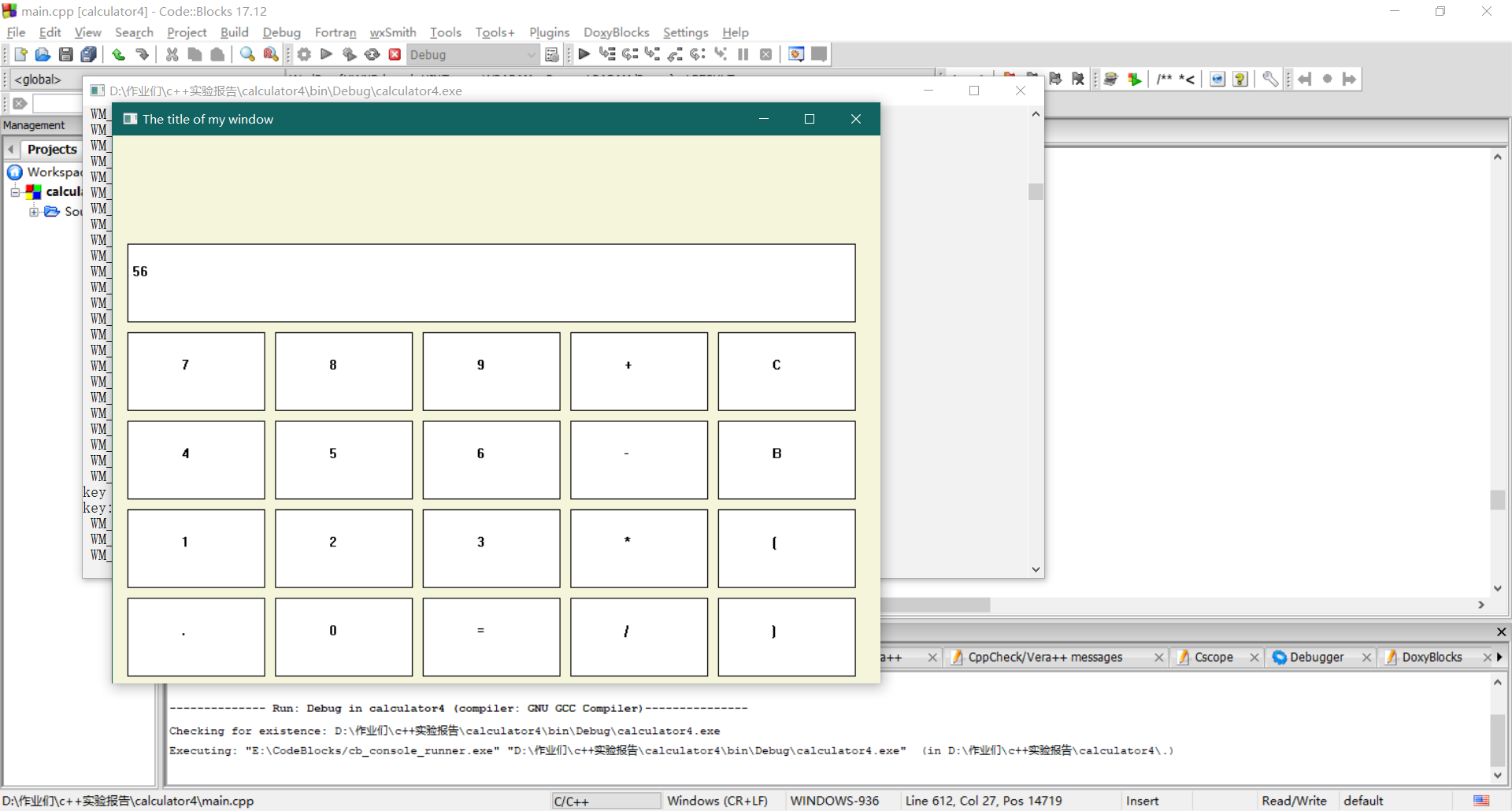
**二 实验原理**

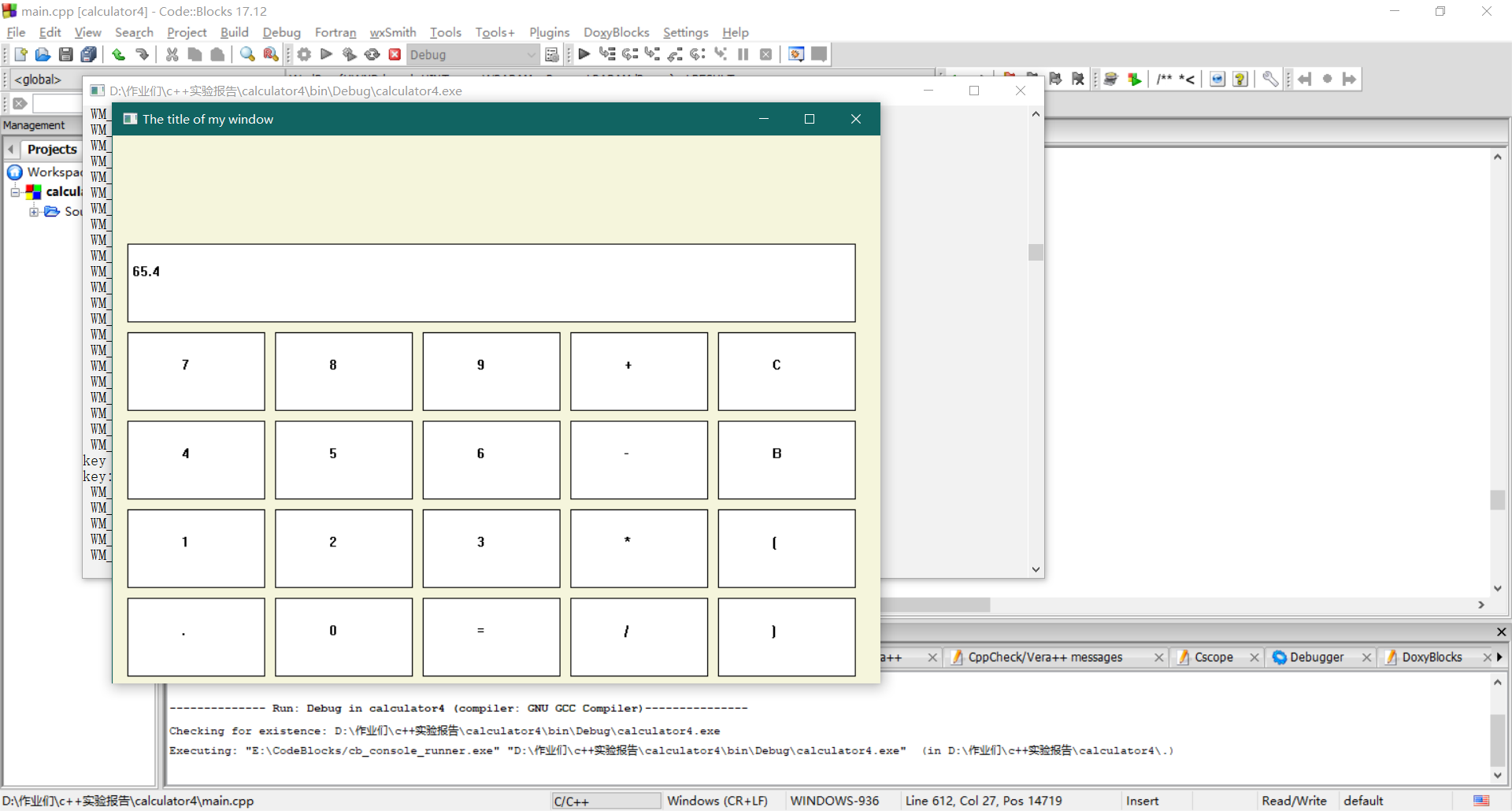
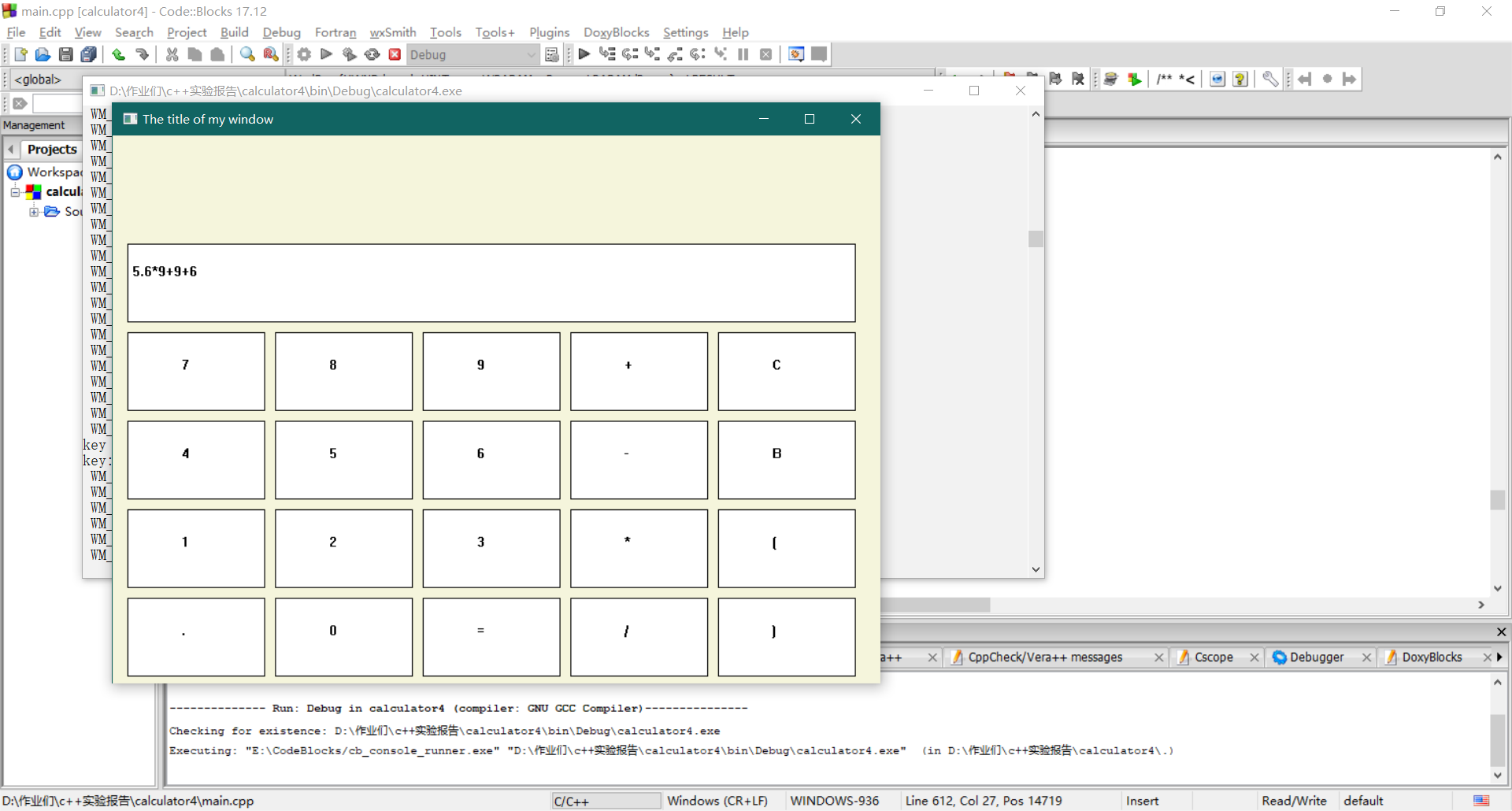
1. 简述程序交互界面功能的实现方法
2. 矩形块的放置：用for循环布置4行5列的矩形块，并建立字符串二维数组，在相应的位置放入字符串。
3. 背景色的设置：在WM\_TIMER里建立一个与窗口一样大小的800\*600的工作区，并用FillRect将工作区涂为自己想要的颜色。
4. 悬浮或点击的颜色响应：在WM\_TIMER最后加入一个对鼠标位置的判断，根据位置对相应的矩形块进行着色，这是悬浮处理。点击时在LBUTTONDOWN里运行click函数，对相应矩形块进行深色着色。
5. 点击输入的功能实现：因为建立了二维数组，所以根据鼠标位置可以得出所在块的位置，也就找到了该位置的字符串，如果是数字、小数点或者运算符和括号，就用一个全局变量字符串expre+=该字符串，如果是C,B就执行相应的字符串置空或者退格。
6. 按键输入的功能实现：在WM\_KEYDOWN里执行keydown函数，根据key的值进行字符串的赋值，其余与（4）相同。
7. 计算功能的实现：根据实验二将计算功能封装到一个类里，当点击=号或者输入=时调用calc函数进行计算并输出结果。
8. #include<windows.h>
9. #include<iostream>
10. #include<stdio.h>
11. #include<string>
12. #include<sstream>
13. #include <math.h>
15. **using** **namespace** std;
17. **const** **char** g\_szClassName[] = "myWindowClass";
18. **int** x,y,key;
19. **int** draw;
20. string expre="";
21. **template**<**typename** T>
22. **struct** stackNode
23. {
24. T data;//存储的具体元素
25. stackNode\* next;//指向下一个节点的指针
26. };
28. **template**<**typename** T>
29. **class** MyStack
30. {
31. **private**:
32. unsigned **int** count;//计数目前stack中存储的元素个数
33. unsigned **int** maxSize;//stack中最大存储的元素个数
34. stackNode<T>\* top;//stack头节点的指针
35. **public**:
36. MyStack();//默认构造函数，创建最多容纳10个元素的stack
37. MyStack(unsigned **int** ms);//自定义容纳个数
38. ~MyStack();//析构函数
39. **void** Push(T data);//压栈
40. T Pop(**void**);//出栈
41. T Top(**void**);//获取栈顶元素
42. **bool** isEmpty(**void**);//检查是否为空
43. **bool** notEmpty(**void**);
44. **bool** isfull(**void**);//经检查是否已满
45. };
47. **int** compare(**char** top,**char** now);
48. **double** calculate(**double** num1,**char** op,**double** num2);
49. **bool** notdod(**char** c);
51. **template**<**typename** T>
52. MyStack<T>::MyStack()
53. {
54. count = 0;
55. maxSize = 30;
56. top = NULL;
57. }
59. **template**<**typename** T>
60. MyStack<T>::MyStack(unsigned **int** ms)
61. {
62. count = 0;
63. maxSize = ms;
64. top = NULL;
65. }
67. **template**<**typename** T>
68. MyStack<T>::~MyStack()
69. {
70. stackNode<T>\* p = **new** stackNode<T>;
71. **for** (unsigned **int** i = count; i >0; i--)
72. {
73. p = top;
74. top = p->next;
75. **delete** p;
76. }
77. }
79. **template**<**typename** T>
80. **void** MyStack<T>::Push(T data)
81. {
82. count++;
83. **if** (isfull())
84. {
85. cout << "Error: This Stack is full" << endl;
86. cout << "File Path =" << \_\_FILE\_\_ << endl;
87. cout << "Function Name =" << \_\_FUNCTION\_\_ << endl;
88. cout << "Line =" << \_\_LINE\_\_ << endl;
89. }
90. stackNode<T>\* node = **new** stackNode<T>;
91. node->data = data;
92. node->next = top;
93. top = node;
94. }
96. **template**<**typename** T>
97. T MyStack<T>::Pop(**void**)
98. {
99. T nowdata;
100. **if** (isEmpty())
101. {
102. cout << "Error: Stack is empty" << endl;
103. cout << "File Path =" << \_\_FILE\_\_ << endl;
104. cout << "Function Name =" << \_\_FUNCTION\_\_ << endl;
105. cout << "Line =" << \_\_LINE\_\_ << endl;
106. **return** 1;
107. }
108. stackNode<T>\* p = top;
109. nowdata=top->data;
110. top = top->next;
111. **delete** p;
112. count--;
113. **return** nowdata;
114. }
116. **template**<**typename** T>
117. T MyStack<T>::Top(**void**)
118. {
119. **return** top->data;
120. }
122. **template**<**typename** T>
123. **bool** MyStack<T>::isEmpty(**void**)
124. {
125. **return** count == 0 ? **true** : **false**;
126. }
128. **template**<**typename** T>
129. **bool** MyStack<T>::notEmpty(**void**)
130. {
131. **return** count == 0 ? **false** : **true**;
132. }
134. **template**<**typename** T>
135. **bool** MyStack<T>::isfull(**void**)
136. {
137. **return** count > maxSize ? **true** : **false**;
138. }
140. **int** compare(**char** top,**char** now)
141. {
142. **int** toplevel=0;
143. **int** nowlevel=0;
144. **if**(top=='+'||top=='-')
145. {
146. toplevel=1;
147. }
148. **else** **if**(top=='\*'||top=='/')
149. {
150. toplevel=2;
151. }
152. **if**(now=='+'||now=='-')
153. {
154. nowlevel=1;
155. }
156. **else** **if**(now=='\*'||now=='/')
157. {
158. nowlevel=2;
159. }
160. **if**(nowlevel>toplevel)
161. {
162. **return** 2;
163. }
164. **else** **if**(nowlevel==toplevel)
165. {
166. **return** 1;
167. }
168. **else**
169. {
170. **return** 0;
171. }
172. }
174. **bool** notdod(**char** c)
175. {
176. **if**((**int**)c>=48&&(**int**)c<=57)
177. {
178. **return** **false**;
179. }
180. **else** **if**(c=='.')
181. {
182. **return** **false**;
183. }
184. **else**
185. {
186. **return** **true**;
187. }
188. }
190. **double** calculate(**double** num1,**char** op,**double** num2)
191. {
192. **if**(op=='+')
193. **return** num1+num2;
194. **else** **if**(op=='-')
195. **return** num1-num2;
196. **else** **if**(op=='\*')
197. **return** num1\*num2;
198. **else**
199. **return** num1/num2;
200. }
202. **void** drawxy(**int** x,**int** y,**HWND** hwnd)
203. {
204. **for**(**int** i=0; i<4; i++)
205. {
206. **for**(**int** j=0; j<5; j++)
207. {
208. **if**((x>=15+j\*150&&x<=155+j\*150)&&(y>=200+i\*90&&y<=280+i\*90))
209. {
210. string \_c[4][5]= {{"7","8","9","+","C"},{"4","5","6","-","B"},{"1","2","3","\*","("},{".","0","=","/",")"}};
211. **HDC** hdc=GetDC(hwnd);
212. SetBkMode(hdc, TRANSPARENT);
213. RECT \_rect;
214. SetRect(&\_rect, 15+j\*150, 200+i\*90,155+j\*150,280+i\*90);
215. **HBRUSH** \_hbrush;
216. \_hbrush = CreateSolidBrush(RGB(200,200,200));
217. FillRect(hdc, &\_rect, \_hbrush);
218. TextOut(hdc,70+j\*150,225+i\*90,TEXT(\_c[i][j].c\_str()),1);
219. TextOut(hdc,20,130,TEXT(expre.c\_str()),expre.length());
220. ReleaseDC(hwnd,hdc);
221. }
222. }
223. }
224. }
226. **int** calc(string &expre)
227. {
228. MyStack<**char**> C;//字符模板栈
229. MyStack<**double**> D;//数值模板栈
230. stringstream numtemp;
231. **int** flag=0;//标志变量
232. **int** con=0;//除以10的倍数
233. **int** asc=0;//当前字符的asc码
234. **double** num=0;//储存当前数字
235. //string expression="null";
236. **char** s;
237. //cout<<"please enter :"<<endl;
238. //cin>>expression;
239. **for**(unsigned **int** i=0; i<**sizeof**(expre); i++)
240. {
241. s=expre[i];
242. asc=(**int**)s;
243. **if**(asc>=48&&asc<=57)//对数字进行处理
244. {
245. **if**(flag==0)
246. {
247. num=num\*10+asc-48;
248. }
249. **else**
250. {
251. num=num+(**float**)(asc-48)/pow(10,con);
252. con++;
253. }
254. **if**(notdod(expre[i+1]))
255. {
256. D.Push(num);
257. num=0;
258. con=0;
259. flag=0;
260. }
261. }
262. **else** **if**(asc==46)//遇到小数点进入除法模式
263. {
264. flag=1;
265. con++;
266. }
267. **else** **if**(s=='(')
268. {
269. C.Push(s);
270. }
271. **else** **if**(s==')')
272. {
273. **if**(C.Top()=='(')//遇到左括号弹出
274. C.Pop();
275. **else**
276. {
277. **do**
278. {
279. D.Push(calculate(D.Pop(),C.Pop(),D.Pop()));//将数字弹出
280. **if**(C.isEmpty())//若空则弹出
281. {
282. cout<<"匹配错误"<<endl;
283. **break**;
284. }
285. }
286. **while**(C.Top()!='(');
287. **if**(C.Top()=='(')//弹出括号
288. C.Pop();
289. }
290. }
291. **else** **if**(s=='+')
292. {
293. **if**(C.isEmpty())
294. C.Push(s);
295. **else** **if**(C.Top()=='(')
296. C.Push(s);
297. **else**
298. {
299. **do**
300. {
301. D.Push(calculate(D.Pop(),C.Pop(),D.Pop()));
302. **if**(C.isEmpty())
303. {
304. **break**;
305. }
306. }
307. **while**(C.Top()!='(');
308. C.Push(s);
309. }
311. }
312. **else** **if**(s=='-')
313. {
314. **if**(C.isEmpty())
315. C.Push(s);
316. **else** **if**(C.Top()=='(')
317. C.Push(s);
318. **else**
319. {
320. **do**
321. {
322. D.Push(calculate(D.Pop(),C.Pop(),D.Pop()));
323. **if**(C.isEmpty())
324. {
325. **break**;
326. }
327. }
328. **while**(C.Top()!='(');
329. C.Push(s);
330. }
331. }
332. **else** **if**(s=='\*')
333. {
334. **if**(C.isEmpty())
335. C.Push(s);
336. **else** **if**(C.Top()=='(')
337. C.Push(s);
338. **else** **if**(compare(C.Top(),s)==1)
339. {
340. D.Push(calculate(D.Pop(),C.Pop(),D.Pop()));
341. C.Push(s);
342. }
343. **else** **if**(compare(C.Top(),s)==2)
344. C.Push(s);
345. }
346. **else** **if**(s=='/')
347. {
348. **if**(C.isEmpty())
349. C.Push(s);
350. **else** **if**(C.Top()=='(')
351. C.Push(s);
352. **else** **if**(compare(C.Top(),s)==1)
353. {
354. D.Push(calculate(D.Pop(),C.Pop(),D.Pop()));
355. C.Push(s);
356. }
357. **else** **if**(compare(C.Top(),s)==2)
358. C.Push(s);
359. }
360. }
362. **if**(C.notEmpty())
363. {
364. **while**(C.Top()!='(')
365. {
366. D.Push(calculate(D.Pop(),C.Pop(),D.Pop()));
367. **if**(C.isEmpty())
368. {
369. **break**;
370. }
371. }
372. **if**(C.notEmpty())
373. {
374. cout<<"匹配错误"<<endl;
375. **return** 0;
376. }
378. }
379. num=D.Pop();
380. //cout<<"answer:"<<num<<endl;
381. numtemp<<num;
382. numtemp>>expre;
383. **return** 0;
384. }
386. **void** judgestr(string str,string &expre,**HDC** hdc)
387. {
388. **for**(**int** i=0; i<10; i++)
389. {
390. stringstream Temp;
391. Temp<<i;
392. string j=Temp.str();
393. **if**(str==j)
394. {
395. expre+=j;
396. TextOut(hdc,20,130,TEXT(expre.c\_str()),expre.length());
397. }
398. }
399. **if**(str=="+")
400. {
401. expre+="+";
402. TextOut(hdc,20,130,TEXT(expre.c\_str()),expre.length());
403. }
404. **if**(str=="-")
405. {
406. expre+="-";
407. TextOut(hdc,20,130,TEXT(expre.c\_str()),expre.length());
408. }
409. **if**(str=="\*")
410. {
411. expre+="\*";
412. TextOut(hdc,20,130,TEXT(expre.c\_str()),expre.length());
413. }
414. **if**(str=="/")
415. {
416. expre+="/";
417. TextOut(hdc,20,130,TEXT(expre.c\_str()),expre.length());
418. }
419. **if**(str==".")
420. {
421. expre+=".";
422. TextOut(hdc,20,130,TEXT(expre.c\_str()),expre.length());
423. }
424. **if**(str=="(")
425. {
426. expre+="(";
427. TextOut(hdc,20,130,TEXT(expre.c\_str()),expre.length());
428. }
429. **if**(str==")")
430. {
431. expre+=")";
432. TextOut(hdc,20,130,TEXT(expre.c\_str()),expre.length());
433. }
434. **if**(str=="C")
435. {
436. expre="";
437. TextOut(hdc,20,130,TEXT(expre.c\_str()),expre.length());
438. }
439. **if**(str=="B")
440. {
441. expre.erase(expre.end() - 1);
442. TextOut(hdc,20,130,TEXT(expre.c\_str()),expre.length());
443. }
444. **if**(str=="=")
445. {
446. calc(expre);
447. TextOut(hdc,20,130,TEXT(expre.c\_str()),expre.length());
448. }
449. }


453. **void** click(**int** x,**int** y,**HWND** hwnd)
454. {
455. **for**(**int** i=0; i<4; i++)
456. {
457. **for**(**int** j=0; j<5; j++)
458. {
459. **if**((x>=15+j\*150&&x<=155+j\*150)&&(y>=200+i\*90&&y<=280+i\*90))
460. {
461. string \_c[4][5]= {{"7","8","9","+","C"},{"4","5","6","-","B"},{"1","2","3","\*","("},{".","0","=","/",")"}};
462. **HDC** hdc=GetDC(hwnd);
463. SetBkMode(hdc, TRANSPARENT);
464. RECT \_rect;
465. SetRect(&\_rect, 15+j\*150, 200+i\*90,155+j\*150,280+i\*90);
466. **HBRUSH** \_hbrush;
467. \_hbrush = CreateSolidBrush(RGB(150,150,150));
468. FillRect(hdc, &\_rect, \_hbrush);
469. TextOut(hdc,70+j\*150,225+i\*90,TEXT(\_c[i][j].c\_str()),1);
470. judgestr(\_c[i][j],expre,hdc);
471. ReleaseDC(hwnd,hdc);
472. }
473. }
474. }
475. }
477. **void** keydown(**int** key,**HWND** hwnd)
478. {
479. **HDC** hdc=GetDC(hwnd);
480. stringstream alltemp;
481. string temp;
482. cout<<"key down????????"<<endl;
483. cout<<"key:::"<<key<<endl;
484. **for**(**int** i=48;i<=57;i++)
485. {
486. **if**(key==i)//用最上面那一横行的键盘输入
487. {
488. alltemp<<(key-48);
489. alltemp>>temp;
490. expre+=temp;
491. TextOut(hdc,20,130,TEXT(expre.c\_str()),expre.length());
492. }
493. }
494. **if**(key==40)//(
495. {
496. expre+="(";
497. TextOut(hdc,20,130,TEXT(expre.c\_str()),expre.length());
498. }
499. **if**(key==41)//)
500. {
501. expre+=")";
502. TextOut(hdc,20,130,TEXT(expre.c\_str()),expre.length());
503. }
504. **if**(key==107)//小键盘上的加
505. {
506. expre+="+";
507. TextOut(hdc,20,130,TEXT(expre.c\_str()),expre.length());
508. }
509. **if**(key==109)//小键盘上的减
510. {
511. expre+="-";
512. TextOut(hdc,20,130,TEXT(expre.c\_str()),expre.length());
513. }
514. **if**(key==106)//小键盘上的乘
515. {
516. expre+="\*";
517. TextOut(hdc,20,130,TEXT(expre.c\_str()),expre.length());
518. }
519. **if**(key==111)//小键盘上的除
520. {
521. expre+="/";
522. TextOut(hdc,20,130,TEXT(expre.c\_str()),expre.length());
523. }
524. **if**(key==67||key==99)// C
525. {
526. expre+="";
527. TextOut(hdc,20,130,TEXT(expre.c\_str()),expre.length());
528. }
529. **if**(key==66||key==98)// B
530. {
531. expre.erase(expre.end() - 1);
532. TextOut(hdc,20,130,TEXT(expre.c\_str()),expre.length());
533. }
534. **if**(key==187)
535. {
536. calc(expre);
537. TextOut(hdc,20,130,TEXT(expre.c\_str()),expre.length());
538. }
539. TextOut(hdc,20,130,TEXT(expre.c\_str()),expre.length());
540. ReleaseDC(hwnd,hdc);
541. }
543. **LRESULT** CALLBACK WndProc(**HWND** hwnd, **UINT** msg, **WPARAM** wParam, **LPARAM** lParam)
544. {
545. **switch**(msg)
546. {
547. **case** WM\_CLOSE:
548. cout<<" WM\_CLOSE"<<endl;
549. DestroyWindow(hwnd);
550. **break**;
552. **case** WM\_DESTROY:
553. cout<<" WM\_DESTROY"<<endl;
554. PostQuitMessage(0);
555. **break**;
557. **case** WM\_CREATE:
558. cout<<" WM\_CREATE"<<endl;
559. SetTimer(hwnd,1,100,NULL);
560. draw=1;
561. **break**;
563. **case** WM\_MOVE:
564. cout<<" WM\_MOVING"<<endl;
565. **break**;
567. **case** WM\_SIZE:
568. cout<<" WM\_SIZE"<<endl;
569. **break**;
571. **case** WM\_LBUTTONDOWN:
572. x=LOWORD(lParam);
573. y=HIWORD(lParam);
574. drawxy(x,y,hwnd);
575. click(x,y,hwnd);
576. cout<<" WM\_LBUTTONDOWN at "<<x<<" "<<y<<endl;
577. **break**;
579. **case** WM\_LBUTTONUP:
580. x=LOWORD(lParam);
581. y=HIWORD(lParam);
582. drawxy(x,y,hwnd);
583. cout<<" WM\_LBUTTONUP at "<<x<<" "<<y<<endl;
584. **break**;
586. **case** WM\_MOUSEMOVE:
587. x=LOWORD(lParam);
588. y=HIWORD(lParam);
589. drawxy(x,y,hwnd);
590. cout<<" WM\_MOUSEMOVE at "<<x<<" "<<y<<endl;
591. **break**;
593. **case** WM\_RBUTTONDOWN:
594. x=LOWORD(lParam);
595. y=HIWORD(lParam);
596. cout<<" WM\_RBUTTONDOWN at "<<x<<" "<<y<<endl;
597. **break**;
599. **case** WM\_KEYDOWN:
600. key=wParam;
601. keydown(key,hwnd);
602. cout<<" WM\_KEYDOWN : "<<key<<endl;
603. //cout<<key<<endl;
604. **break**;
606. **case** WM\_KEYUP:
607. key=wParam;
608. cout<<" WM\_KEYUP : "<<key<<endl;
609. **break**;
611. **case** WM\_TIMER:
612. cout<<" WM\_TIMER"<<endl;
613. **if**(draw)
614. {
615. string c[4][5]= {{"7","8","9","+","C"},{"4","5","6","-","B"},{"1","2","3","\*","("},{".","0","=","/",")"}};
616. //int len=sprintf(c,"%d",ind);
617. **HDC** hdc=GetDC(hwnd);
618. SetBkMode(hdc, TRANSPARENT);
619. **HBRUSH** hBrush;
620. RECT rect;
621. SetRect(&rect, 0, 0,800,600);
622. hBrush = CreateSolidBrush(RGB(245,245,220));
623. FillRect(hdc, &rect, hBrush);
624. hBrush = NULL;
626. **for**(**int** i=0; i<4; i++)
627. {
628. **for**(**int** j=0; j<5; j++)
629. {
630. Rectangle(hdc, 15+j\*150,200+i\*90,155+j\*150,280+i\*90);
631. TextOut(hdc,70+j\*150,225+i\*90,TEXT(c[i][j].c\_str()),1);
632. TextOut(hdc,20,130,TEXT(expre.c\_str()),expre.length());
633. }
634. }
635. Rectangle(hdc, 15,110,755,190);
636. ReleaseDC(hwnd,hdc);
637. }
638. drawxy(x,y,hwnd);
639. TextOut(GetDC(hwnd),20,130,TEXT(expre.c\_str()),expre.length());
640. **break**;
642. **default**:
643. **return** DefWindowProc(hwnd, msg, wParam, lParam);
644. }
645. **return** 0;
646. }

649. **int** WINAPI WinMain(**HINSTANCE** hInstance, **HINSTANCE** hPrevInstance,
650. **LPSTR** lpCmdLine, **int** nCmdShow)
651. {
652. WNDCLASSEX wc;
653. **HWND** hwnd;
654. MSG Msg;
655. //Step 1: Registering the Window Class
656. wc.cbSize        = **sizeof**(WNDCLASSEX);
657. wc.style         = 0;
658. wc.lpfnWndProc   = WndProc;
659. wc.cbClsExtra    = 0;
660. wc.cbWndExtra    = 0;
661. wc.hInstance     = hInstance;
662. wc.hIcon         = LoadIcon(NULL, IDI\_APPLICATION);
663. wc.hCursor       = LoadCursor(NULL, IDC\_ARROW);
664. wc.hbrBackground = (**HBRUSH**)(COLOR\_WINDOW+1);
665. wc.lpszMenuName  = NULL;
666. wc.lpszClassName = g\_szClassName;
667. wc.hIconSm       = LoadIcon(NULL, IDI\_APPLICATION);
669. **if**(!RegisterClassEx(&wc))
670. {
671. MessageBox(NULL, "Window Registration Failed!", "Error!",
672. MB\_ICONEXCLAMATION | MB\_OK);
673. **return** 0;
674. }
676. // Step 2: Creating the Window
677. hwnd = CreateWindowEx(
678. WS\_EX\_CLIENTEDGE,
679. g\_szClassName,
680. "The title of my window",
681. WS\_OVERLAPPEDWINDOW,
682. CW\_USEDEFAULT, CW\_USEDEFAULT, 800,600,
683. NULL, NULL, hInstance, NULL);
685. **if**(hwnd == NULL)
686. {
687. MessageBox(NULL, "Window Creation Failed!", "Error!",
688. MB\_ICONEXCLAMATION | MB\_OK);
689. **return** 0;
690. }
692. ShowWindow(hwnd, nCmdShow);
693. UpdateWindow(hwnd);
695. // Step 3: The Message Loop
696. **while**(GetMessage(&Msg, NULL, 0, 0) > 0)
697. {
698. TranslateMessage(&Msg);
699. DispatchMessage(&Msg);
700. }
701. **return** Msg.wParam;
702. }
703. 给出实验测试结果







**三 实验总结与建议**

（总结实验实施过程，说明实验过程中遇到的问题与解决方案；提出实验环节的建议）

在读懂了相应的hdc，hwnd，rect，hbrush，textout操作后，调出窗口并绘制图象就变得不那么难了。

而且在实验二的基础上，经过封装之后可以很容易的将显示的表达式计算出来，可移植性很强。