

实验五、函数实验

计算机学院 熊明 20305055

一、实验目的

1. 了解自顶向下TOP-DOWN结构化程序设计方法（面向过程编程）
2. 掌握函数的定义、声明的方法；
3. 掌握函数的编写要求；
4. 掌握函数的调用方法；
5. 掌握函数参数的传递方法；
6. 掌握多文件编程方法。

二、实验原理

将一些功能封装成函数，在main函数中通过调用函数的方法实现功能

三、实验内容

1. 函数版本的计算器

```
1  #include<iostream>
2  using namespace std;
3  void add(int x,int y){
4      cout<<x<<"+ "<<y<< "="<<x+y<<endl;
5  }
6  void delet(int x,int y){
7      cout<<x<< "- "<<y<< "="<<x-y<<endl;
8  }
9  void mul(int x,int y){
10     cout<<x<< " * "<<y<< "="<<x*y<<endl;
11 }
12 void divide(int x,int y){
13     cout<<x<< " / "<<y<< "="<<x/y<<endl;
14 }
15 int main(){
16     double x,y;
17     int chose,isFlag = 1;
18     do{
19         cout<<"*****计算器*****"<<endl;
20         cout<<"*****1. 加法*****"<<endl;
21         cout<<"*****2. 减法*****"<<endl;
22         cout<<"*****3. 乘法*****"<<endl;
23         cout<<"*****4. 除法*****"<<endl;
24         cout<<"*****5. 退出*****"<<endl;
25         cout<<"*****请选择1-5: *****"<<endl;
26         cin>>chose;
27         if(chose>5 || chose<1){
28             cout<<"非法输入，请重试"<<endl;
29         }else{
30             if(chose==5){
```

```

31         isFlag=0;
32         continue;
33     }
34     cout<<"输入x的值为: "<<endl;
35     cin>>x;
36     cout<<"输入y的值为: "<<endl;
37     cin>>y;
38     switch (chosed)
39     {
40     case 1:
41         add(x,y);
42         break;
43     case 2:
44         delet(x,y);
45         break;
46     case 3:
47         mul(x,y);
48         break;
49     case 4:
50         divide(x,y);
51         break;
52     default:
53         break;
54     }
55 }
56 } while(isFlag);
57 cout<<"程序退出~"<<endl;
58 }

```

2. 研读下面代码，改写成函数实现

```

1  #include<iostream>
2  using namespace std;
3  int numa_to_numb(int a,int b){
4      int ans = 1;
5      for(int i;i<b;i++){
6          ans *= a;
7      }
8      return ans;
9  }
10 int main(){
11     cout<<"3^4 is "<<numa_to_numb(3,4)<<endl;
12     cout<<"6^5 is "<<numa_to_numb(6,5)<<endl;
13     cout<<"12^10 is "<<numa_to_numb(12,10)<<endl;
14 }

```

发现结果错误，经过debug后发现是循环忘记将i赋值为0，改进后代码为：

```

1  #include<iostream>
2  using namespace std;
3  int numa_to_numb(int a,int b){
4      int ans = 1;
5      for(int i=0;i<b;i++){
6          ans *= a;

```

```

7     }
8     return ans;
9 }
10 int main(){
11     cout<<"3^4 is "<<numa_to_num(3,4)<<endl;
12     cout<<"6^5 is "<<numa_to_num(6,5)<<endl;
13     cout<<"12^10 is "<<numa_to_num(12,10)<<endl;
14 }

```

运行结果:

```

PS E:\vscode> & 'c:\Users\15989\.vscode\extensions\ms-vscode.cp
icrosoft-MIEngine-In-ww0fdv3z.stl' '--stdout=Microsoft-MIEngine-
osoft-MIEngine-Pid-ouqvtnc.5bc' '--dbgExe=C:\MinGW\mingw64\bin\
3^4 is 81
6^5 is 7776
12^10 is 1787822080

```

numa_to_num函数实现了任意整数a的b次幂运算。

3. 自定义函数尝试使图案旋转起来:

```

1  #include <graphics.h>
2  #include<conio.h>
3  #include<iostream>
4  #define PI 3.14159
5  using namespace std;
6  int tangel = 20;
7  double B = tangel * PI / 180;
8  void rotate1(double &a,double &b) {
9      double x = a, y = b;
10     a = x * cos(B) - y * sin(B) - 160 * cos(B) + 240 * sin(B) + 160;
11     b = x * sin(B) + y * cos(B) - 160 * sin(B) - 240 * cos(B) + 240;
12 }
13 void rotate2(double& a, double& b) {
14     double x = a, y = b;
15     a = x * cos(B) - y * sin(B) - 500 * cos(B) + 240 * sin(B) + 500;
16     b = x * sin(B) + y * cos(B) - 500 * sin(B) - 240 * cos(B) + 240;
17 }
18 int main(){
19     initgraph(640, 480);
20     double a1 = 70, a2 = 160, a3 = 250, a4 = 160, b1 = 150, b2 = 240, b3
= 330, b4 = 240;
21     double aa1 = 600, aa2 = 400, aa3 = 500, aa4 = 500, bb1 = 240, bb2 =
240, bb3 = 340, bb4 = 140;
22     while (1) {
23
24         fillcircle(160, 240, 25); //中心圆
25
26         fillcircle(a1, b4, 50); //
27         rotate1(a1, b4);
28         fillcircle(a3, b2, 50);
29         rotate1(a3, b2);
30         fillcircle(a2, b3, 50);
31         rotate1(a2, b3);
32         fillcircle(a4, b1, 50);

```

```

33     rotate1(a4, b1);
34
35     fillcircle(500, 240, 50); //中心圆
36
37     fillcircle(aa1, bb1, 25);
38     rotate2(aa1, bb1);
39     fillcircle(aa2, bb2, 25);
40     rotate2(aa2, bb2);
41     fillcircle(aa3, bb3, 25);
42     rotate2(aa3, bb3);
43     fillcircle(aa4, bb4, 25);
44     rotate2(aa4, bb4);
45     sleep(150);
46     cleardevice();
47 }
48 closegraph();
49 return 0;
50 }

```

参考[图像\(点\)绕任意中心位置旋转的旋转矩阵推导](#)

设置颜色, 参考[setColor方法: 设置颜色](#)

```

1  #include <graphics.h>
2  #include<conio.h>
3  #include<iostream>
4  #define PI 3.14159
5  using namespace std;
6  int tangel = 20;
7  double B = tangel * PI / 180;
8  void rotate1(double &a, double &b) {
9      double x = a, y = b;
10     a = x * cos(B) - y * sin(B) - 160 * cos(B) + 240 * sin(B) + 160;
11     b = x * sin(B) + y * cos(B) - 160 * sin(B) - 240 * cos(B) + 240;
12 }
13 void rotate2(double& a, double& b) {
14     double x = a, y = b;
15     a = x * cos(B) - y * sin(B) - 500 * cos(B) + 240 * sin(B) + 500;
16     b = x * sin(B) + y * cos(B) - 500 * sin(B) - 240 * cos(B) + 240;
17 }
18 int main(){
19     initgraph(640, 480);
20     double a1 = 70, a2 = 160, a3 = 250, a4 = 160, b1 = 150, b2 = 240, b3
= 330, b4 = 240;
21     double aa1 = 600, aa2 = 400, aa3 = 500, aa4 = 500, bb1 = 240, bb2 =
240, bb3 = 340, bb4 = 140;
22     while (1) {
23         setfillcolor(YELLOW); //设置填充圆的颜色
24         fillcircle(160, 240, 25); //中心圆
25         setfillcolor(RED); //设置填充圆的颜色
26         fillcircle(a1, b4, 50); //
27         rotate1(a1, b4);
28         setfillcolor(BLUE); //设置填充圆的颜色
29         fillcircle(a3, b2, 50);
30         rotate1(a3, b2);

```

```
31     setfillcolor(GREEN); //设置填充圆的颜色
32     fillcircle(a2, b3, 50);
33     rotate1(a2, b3);
34     setfillcolor(MAGENTA); //设置填充圆的颜色
35     fillcircle(a4, b1, 50);
36     rotate1(a4, b1);
37
38     setfillcolor(CYAN); //设置填充圆的颜色
39     fillcircle(500, 240, 50); //中心圆
40
41     setfillcolor(MAGENTA); //设置填充圆的颜色
42     fillcircle(aa1, bb1, 25);
43     rotate2(aa1, bb1);
44     setfillcolor(BLUE); //设置填充圆的颜色
45     fillcircle(aa2, bb2, 25);
46     rotate2(aa2, bb2);
47     setfillcolor(GREEN); //设置填充圆的颜色
48     fillcircle(aa3, bb3, 25);
49     rotate2(aa3, bb3);
50     setfillcolor(YELLOW); //设置填充圆的颜色
51     fillcircle(aa4, bb4, 25);
52     rotate2(aa4, bb4);
53     sleep(100);
54     cleardevice();
55 }
56 closegraph();
57 return 0;
58
```

四、实验心得体会

debug是很好的代码自我检测工具，在编程大工程的时候，往往很难发现自己犯的小错误，这个时候debug工具就非常重要了。