**实 验 报 告**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **课程名称** | 软件测试 | | | | | |
| **实验项目名称** | 软件测试试验报告 | | | | | |
| **实验时间**  **（日期及节次）** | 2024.06.19 (第16周7、8节) | | | | | |
| **专业** | 软件工程 | | **学生所在学院** | | 软件学院 | |
| **年级** | 2022级 | | **学号** | | 20225958 | |
| **姓名** | 李嘉富 | | **指导教师** | | 王楠 | |
| **实验室名称** | 4号楼520 | | | | | |
| **实验成绩** | **预习情况** | **操作技术** | **实验报告** | **附加：综合创新能力** | | **实验**  **综合成绩** |
|  |  |  |  | |  |
| **教师签字** |  | | | | | |

**黑龙江大学教务处**

### 1.1编写目的

此测试分析报告是为了对人民币数字到大写转换系统的设计、编程和使用进行测试考核，以确定其软件本身是否有缺陷和不足以及可能给软件运行带来的影响，改正系统BUG，并对其测试结果做出结论。

### 1.2 程序内容

编写“人民币数字到大写转换”程序。

计算范围：整数部分长度在12位（千亿单位）及以内， 小数部分长度在2位（分单位）

具体需求如下：

1）中文大写金额数字应用壹、贰、叁、肆、伍、陆、柒、捌、玖、拾、佰、仟、万、亿、元、角、分、零、整(正)等字样。

2）中文大写金额数字到"元"为止的，在"元"之后，应写"整"(或"正")字，在"角"之后，可以不写"整"(或"正")字。

3）中文大写金额数字前应标明"人民币"字样，大写金额数字有"分"的，“分"后面不写"整”(或"正")字。

4）大写金额数字应紧接"人民币"字样填写，不得留有空白。

5）阿拉伯数字小写金额数字中有"0"时，中文大写应按照汉语语言规律、金额数字构成和防止涂改的要求进行书写。

举例如下：

1、阿拉伯数字中间有"0"时，中文大写要写"零"字，如￥1409.50，应写成：人民币壹仟肆佰零玖元伍角。

2、阿拉伯数字中间连续有几个"0"时，中文大写金额中间只写一个"零"字，如￥6007.14，应写成：人民币陆仟零柒元壹角肆分。

3、阿拉伯金额数字万位和元位是"0"，或者数字中间连续有几个"0"，万位、元位也是"0"，但千位、角位不是"0"时，中文大写金额中只写一个零字，也可以不写"零"字。如￥1680.32，应写成：人民币壹仟陆佰捌拾元叁角贰分，又如￥107000.53，应写成：人民币壹拾万零柒仟元伍角叁分。

4、阿拉伯金额数字角位是"0"，而分位不是"0"时，中文大写金额"元"后面应写"零"字。如￥16409.02，应写成人民币：壹万陆仟肆佰零玖元零贰分；又如￥325.04，应写成人民币叁佰贰拾伍元零肆分。

### 1.3 运行环境

操作系统:Windows10

软件：CodeBlocks

### 1.4 条件与限制

本次测试采用的主要是黑盒测试，对系统各个功能模块进行测试，设计

合理的测试用例即可，没有其他特殊的要求。

### 1.5 划分等价类

等价表如表1-1所示:

**表1-1:等价表**

|  |  |  |
| --- | --- | --- |
| **输入等价类** | **有效等价类** | **无效等价类** |
| **长度** | 1. 12>=整数长度>0 2. 2>=小数长度>=0 | 1. 整数长度>12 2. 小数长度>2 3. 什么都不输入 4. 整数长度=0 |
| **输入的字符** | 1. 输入有效数字 2. 小数点个数<=1 3. 1<=输入中间连续的0的个数<=10 4. 输入0的位置在整数部分 5. 输入0的位置在小数部分 6. 整数部分为0 7. 小数部分为0 8. 整数部分和小数部分都为0 | 1. 含有非法字符（含有字字母） 2. 非法字符（含有标点）含有 3. 非法字符（含有运算符） 4. 非法字符（含有汉字） 5. 输入负值 6. 小数点个数>1 |
| **输出的字符** | 1. 壹 2. 贰 3. 叁 4. 肆 5. 伍 6. 陆 7. 柒 8. 捌 9. 玖 10. 零 11. 整 12. 角 13. 元 14. 分 15. 拾 16. 佰 17. 仟 18. 万 19. 亿 20. 元后写整 21. 分后不写整 22. 角后不写整 23. 输出紧接人们币字样 |  |

### 1.6 测试用例

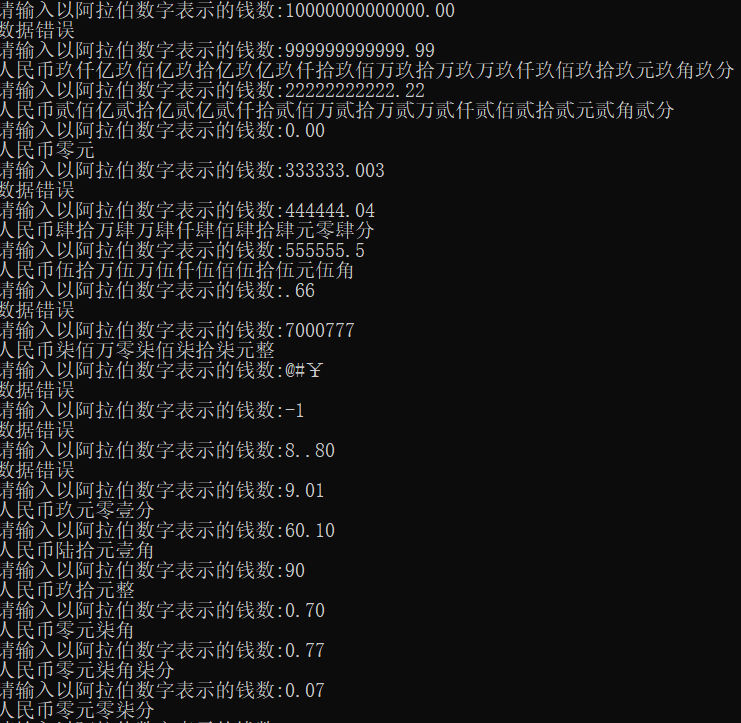
测试用例如表1-2所示:

**表1-2:测试用例**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **测试用例编号** | **输入数据** | **期望结果** | **运行结果** | **覆盖等价类** |
| 1 | 10000000000000.00 | 提示错误 | 数据错误 | 3 |
| 2 | 12345678909.99 | 人民币壹佰贰拾叁亿肆仟伍佰陆拾柒万捌仟玖佰零玖元玖角玖分 | 人民币壹佰亿贰拾亿叁亿肆仟拾伍佰万陆拾万柒万捌仟玖佰零玖元玖角玖分 | 1、2、7、8、38、40 |
| 3 | 22222222222.22 | 人民币壹佰贰拾叁亿肆仟伍佰陆拾玖万陆仟叁佰贰拾壹元伍角肆分 | 人民币贰佰亿贰拾亿贰亿贰仟拾贰佰万贰拾万贰万贰仟贰佰贰拾贰元贰角贰分 | 1、2、7、8、38 |
| 4 | 0.00 | 人民币零元整 | 人民币零元 | 1、2、7、8、14、37 |
| 5 | 333333.003 | 提示错误 | 数据错误 | 4 |
| 6 | 444444.04 | 人民币肆拾万肆万肆仟肆佰肆拾肆元零肆分 | 人民币肆拾万肆万肆仟肆佰肆拾肆元零肆分 | 1、2、7、8、11、38 |
| 7 | 555555.5 | 人民币伍拾万伍万伍仟伍佰伍拾伍元伍角 | 人民币伍拾万伍万伍仟伍佰伍拾伍元伍角 | 1、2、7、8、39 |
| 9 | .66 | 提示错误 | 数据错误 | 6 |
| 10 | 101 | 人民币壹佰零壹元整 | 人民币壹佰零壹元整 | 1、7、9、37 |
| 11 | 1001 | 人民币壹仟零壹元整 | 人民币壹仟零壹元整 | 1、7、9、37 |
| 12 | 1000001 | 人民币壹佰万零壹元整 | 人民币壹佰万零壹元整 | 1、7、9、37 |
| 13 | 100000000001 | 人民币壹仟亿零壹元整 | 人民币壹仟亿零壹元整 | 1、7、9、37 |
| 15 | 10000000001 | 人民币壹佰亿零壹元整 | 人民币壹佰亿零壹元整 | 1、7、9、37 |
| 16 | @#￥ | 提示错误 | 数据错误 | 15 |
| 17 | -1 | 提示错误 | 数据错误 | 16 |
| 18 | 8..80 | 提示错误 | 数据错误 | 17 |
| 19 | 90 | 人民币贰拾元整 | 人民币玖拾元整 | 1、7、10 |
| 20 | 0.70 | 人民币叁角 | 人民币零元柒角 | 1、2、7、8、10、11、39 |
| 21 | 0.77 | 人民币叁角叁分 | 人民币零元柒角柒分 | 1、2、7、8、10、12 |
| 22 | 0.07 | 人民币叁分 | 人民币零元零柒分 | 1、2、7、8、10、12 |
| 23 | 0 | 人民币零元整 | 人民币零元 | 16 |
| 24 | “空” | 数据错误 | “空” | 5 |

### 1.7 测试结果截图

测试结果如图1-1所示



**图1-1测试结果**

### 1.8 制约条件

根据程序的具体要求,将程序的输入(原因)列如下几条:

1.输入的整数长度最大为12

2.没有小数部分

3.输入至小数点后一位

4.输入至小数点后两位

5.输入的数字中间有零

6.非法输入

7.输入的数是最大数

E关系:①2,3,7

②2,3,4

③6与其他

R关系:4.7

### 1.9 测试结论

通过对程序三进行的黑盒边界值法进行测试,发现代码中间的转换有一定的问题，测试用例并没有全部通过。

### 1.10 代码

#include<stdio.h>

#include<stdlib.h>

int change()

{

char m[20] = { '0' };

printf("请输入以阿拉伯数字表示的钱数:");

scanf("%s", &m);

int d = 0, sum = 0, length = 0, length2 = 0;

int num = 0;

for (int i = 0; m[i] != '\0'; i++)

{

if (m[i] == '.')

{

d = i;

num++;

}

if (m[i] >= '0'&&m[i] <= '9' || m[i] == '.')

length2++;

length = i;

}

if (m[0] == '.' || num > 1|| length + 1 != length2)

{

printf("数据错误\n");

return 0;

}

if (d != 0 && length == d + 1)

{

m[d + 2] = '0';

}

if (d == 0)

d = length + 1;

if (length > d + 2 || d > 12)

{

printf("数据错误\n");

return 0;

}

if (m[0] == '0'&&m[1] != '.')

{

printf("数据错误\n");

return 0;

}

printf("人民币");

for (int i = 0; m[i] != '\0'; i++)

{

switch (m[i])

{

case '0':

{

sum = 0;

for (int j = i; j < d; j++)

{

if (m[j] == '0')

{

sum++;

}

}

if (sum != d - i && i != d + 2 && m[i + 1] != '0' || i == 0 )

{

printf("零");

}

break;

}

case '1':

printf("壹");

break;

case '2':

printf("贰");

break;

case '3':

printf("叁");

break;

case '4':

printf("肆");

break;

case '5':

printf("伍");

break;

case '6':

printf("陆");

break;

case '7':

printf("柒");

break;

case '8':

printf("捌");

break;

case '9':

printf("玖");

break;

}

if (i == d - 1 && d == length + 1)

printf("元整");

if (i == d - 1 && d != length + 1)

printf("元");

if (m[i] != '0')

{

switch (d - i)

{

case -1:

printf("角");

break;

case -2:

printf("分");

break;

case 2:

printf("拾");

break;

case 3:

printf("佰");

break;

case 4:

printf("仟");

break;

case 5:

printf("万");

break;

case 6:

printf("拾万");

break;

case 7:

printf("佰万");

break;

case 8:

printf("仟拾");

break;

case 9:

printf("亿");

break;

case 10:

printf("拾亿");

break;

case 11:

printf("佰亿");

break;

case 12:

printf("仟亿");

break;

}

}

}

printf("\n");

return 0;

}

int main()

{

while(1)

{

change();

}

}