Experiment5-董皓彧

环境：

```
gcc.exe (x86_64-win32-seh-rev0, Built by MinGW-W64 project) 8.1.0
Visual Stdio Code 1.83.1
```

作业仓库地址：
https://github.com/FHYQ-Dong/Tsinghua-Program-Design-Assignments/tree/main/Experiment5

必做题

Experiment5-1

题目：

```
输入三个整数，判断它们能否构成三角形（等边，等腰，一般，无法）
```

输入格式：

```
一行，三个整数，用空格隔开
```

输出格式：

```
一行，三角形类型，或者无法构成三角形
```

代码：

```c
#include<stdio.h>
#define true 1
#define false 0
typedef int bool;

bool check_tri(int x, int y, int z) {
    if (x + y > z && x + z > y && y + z > x) {
        return true;
    }
    return false;
}

int main() {
    int a, b, c;
    scanf("%d%d%d", &a, &b, &c);
    if(check_tri(a, b, c)) {
        if (a==b && a==c) printf("Equilateral triangle\n");
        else if (a==b || a==c || b==c) printf("Isosceles triangle\n");
        else printf("Triangle\n");
    }
    else printf("Not triangle\n");
    return 0;
}
```

输入1:

```
1 1 1
```

输出1:

```
Equilateral triangle
```

输入2:

```
1 2 2
```

输出2:

```
Isosceles triangle
```

输入3:

```
2 3 4
```

输出3:

```
Triangle
```

输入4:

```
1 2 3
```

输出4:

```
Not triangle
```

Experiment5-2

题目:

```
输入运算符@和四个整数a, b, c, d, 计算a@b@c@d的值
```

输入格式:

```
一行，@和四个整数，用空格隔开
```

输出格式:

```
一行，计算结果，或者错误信息
```

代码：

```
#include<stdio.h>
#define true 1
#define false 0
typedef int bool;
typedef struct Result Result;

struct Result {
    double res;
    bool err;
};

Result One_Result(double r, bool e) {
    Result tmp;
    tmp.res = r;
    tmp.err = e;
    return tmp;
}

Result operate(Result x, Result y, char op) {
    if (x.err || y.err) return One_Result(0, true);
    double a = x.res, b = y.res;
    if (op == '/' && b == 0) return One_Result(0, true);
    switch (op) {
        case '+': return One_Result(a + b, false);
        case '-': return One_Result(a - b, false);
        case '*': return One_Result(a * b, false);
        case '/': return One_Result(a / b, false);
    }
}

Result a, b, c, d;
char op;

int main() {
    scanf("%c %lf %lf %lf %lf", &op, &a.res, &b.res, &c.res, &d.res);
    Result res = operate(operate(operate(a, b, op), c, op), d, op);
    if(res.err) printf("input error\n");
    else printf("%lf\n", res.res);
    return 0;
}
```

输入1：

```
+ 1 2 3 4
```

输出1：

```
10.000000
```

输入2：

```
- 5 6 7 8
```

输出2：

```
-16.000000
```

输入3：

```
* 9 10 11 12
```

输出3：

```
11880.000000
```

输入4：

```
/ 13 14 15 16
```

输出4：

```
0.003869
```

输入5：

```
/ 1 2 3 0
```

输出5：

```
input error
```

Experiment5-3

题目：

```
给出一个int型正整数x：
（1）求出x的位数；
（2）打印出每一位数字；
（3）逆向打印各位数字
```

输入格式：

```
一行，一个正整数
```

输出格式：

```
三行，分别是位数，每一位数字，逆向每一位数字
```

代码：

```c
#include<stdio.h>

int main() {
    int a, digit[20] = {0}, len = 0;
    scanf("%d", &a);
    while(a) {
        ++len;
        digit[len] = a % 10;
        a /= 10;
    }
    printf("length: %d\n", len);
    printf("digits: ");
    for(int i = len; i >= 1; --i) printf("%d ", digit[i]);
    printf("\n");
    printf("reverse: ");
    for(int i = 1; i <= len; ++i) printf("%d ", digit[i]);
    return 0;
}
```

输入1：

```
12345678
```

输出1：

```
length: 8
digits: 1 2 3 4 5 6 7 8
reverse: 8 7 6 5 4 3 2 1
```

输入2：

```
1
```

输出2：

```
length: 1
digits: 1
reverse: 1
```

输入3：

```
100000
```

输出3：

```
length: 6
digits: 1 0 0 0 0 0
reverse: 0 0 0 0 0 1
```

输入4：

```
1000000000
```

输出4：

```
length: 10
digits: 1 0 0 0 0 0 0 0 0 0
reverse: 0 0 0 0 0 0 0 0 0 1
```

输入5：

```
64648513
```

输出5：

```
length: 8
digits: 6 4 6 4 8 5 1 3
reverse: 3 1 5 8 4 6 4 6
```

输入6：

```
5631355
```

输出6：

```
length: 7
digits: 5 6 3 1 3 5 5
reverse: 5 5 3 1 3 6 5
```

输入7：

```
789645
```

输出7：

```
length: 6
digits: 7 8 9 6 4 5
reverse: 5 4 6 9 8 7
```

输入8：

```
89645348
```

输出8：

```
length: 8
digits: 8 9 6 4 5 3 4 8
reverse: 8 4 3 5 4 6 9 8
```

输入9：

```
789534861
```

输出9：

```
length: 9
digits: 7 8 9 5 3 4 8 6 1
reverse: 1 6 8 4 3 5 9 8 7
```

输入10：

```
654321
```

输出10：

```
length: 6
digits: 6 5 4 3 2 1
reverse: 1 2 3 4 5 6
```

选做题

Optional-Experiment5-1

题目：

```
将数字时间用英文表述
```

输入格式：

```
一行，一个数字时间，格式为HH MM（用Tab隔开）
```

输出格式：

```
一行，英文表述
```

代码：

```
#include<stdio.h>

int ihour, iminute;
char shour, sminute;
const char spell_20[21][15] = {"zero", "one", "two", "three", "four", "five",
"six", "seven", "eight", "nine",
                "ten", "eleven", "twelve", "thirteen", "fourteen", "fifteen",
"sixteen",
                "seventeen", "eighteen", "nineteen", "twenty"};
const char spell_50[6][10] = {"0", "0", "twenty", "thirty", "forty", "fifty"};

void print_hour() {
    if(ihour <= 20) {
        printf("%s", spell_20[ihour]);
    }
    else {
        printf("%s %s", spell_50[ihour / 10], spell_20[ihour % 10]);
    }
    return;
}

void print_minute() {
    if(iminute <= 20) {
        printf("%s", spell_20[iminute]);
    }
    else {
        switch (iminute % 10) {
            case 0: printf("%s", spell_50[iminute / 10]); break;
            default: printf("%s %s", spell_50[iminute / 10], spell_20[iminute %
10]);
        }
    }
    return;
}

void print_time() {
    if(iminute == 0) {
        print_hour();
        printf(" o'clock\n");
    }
    else {
        print_hour();
        printf(" ");
        print_minute();
        printf("\n");
    }
}

int main() {
    scanf("%d\t%d", &ihour, &iminute);
    print_time();
    return 0;
}
```

输入1：

```
00  00
```

输出1：

```
zero o'clock
```

输入2：

```
01  10
```

输出2：

```
one ten
```

输入3:

```
10  19
```

输出3:

```
ten nineteen
```

输入4:

```
21  30
```

输出4:

```
twenty one thirty
```

输入5:

```
23  59
```

输出5:

```
twenty three fifty nine
```

Optional-Experiment5-2

题目:

```
存在4个高塔，具体信息见课件，给出一个坐标，判断其高度
```

输入格式:

```
一行，两个整数，用空格隔开，表示某点的坐标
```

输出格式:

```
一行，高度
```

代码:

```
#include<stdio.h>
#include<math.h>
#define true 1
#define false 0
typedef int bool;
typedef struct Tower Tower;
```

```
struct Tower {
    double x, y;
    double radius;
    int height;
};
Tower tower[5];

Tower One_Tower(double x, double y, double r, int h) {
    Tower tmp;
    tmp.x = x;
    tmp.y = y;
    tmp.radius = r;
    tmp.height = h;
    return tmp;
}

bool on_tower(Tower t, double x, double y) {
    if (pow(x - t.x, 2) + pow(y - t.y, 2) <= pow(t.radius, 2)) return true;
    return false;
}

int main() {
    tower[1] = One_Tower(2.0, 2.0, 1.0, 10); tower[2] = One_Tower(-2.0, 2.0, 1.0, 9);
    tower[3] = One_Tower(-2.0, -2.0, 1.0, 8); tower[4] = One_Tower(2.0, -2.0, 1.0, 7);
    double x, y;
    scanf("%lf%lf", &x, &y);
    for(int i=1; i<=4; ++i) {
        if(on_tower(tower[i], x, y)) {
            printf("height: %d\n", tower[i].height);
            return 0;
        }
    }
    printf("height: 0\n");
    return 0;
}
```

输入1：

```
0 0
```

输出1：

```
height: 0
```

输入2：

```
1 1
```

输出2：

```
height: 0
```

输入3：

```
2 2
```

输出3：

```
height: 10
```

输入4：

```
1.5 1.5
```

输出4：

```
height: 10
```

输入5：

```
-2 2
```

输出5：

```
height: 9
```

输入6：

```
-1.5 1.5
```

输出6：

```
height: 9
```

Optional-Experiment5-3

题目：

```
统计一个int型正整数x中数字5、6、7出现的次数
```

输入格式：

```
一行，一个正整数
```

输出格式：

```
一行，三个数字，分别是5、6、7出现的次数，用空格隔开
```

代码：

```
#include<stdio.h>
```

```
int cnt[10] = {0};

int main() {
    int a;
    scanf("%d", &a);
    while(a) {
        ++cnt[a % 10];
        a /= 10;
    }
    for(int i = 5; i <= 7; ++i) printf("%d ",cnt[i]);
    return 0;
}
```

输入1：

```
0
```

输出1：

```
0 0 0
```

输入2：

```
11111111
```

输出2：

```
0 0 0
```

输入3：

```
123456789
```

输出3：

```
1 1 1
```

输入4：

```
77777
```

输出4：

```
0 0 5
```

输入5：

```
5555555
```

输出5：

```
7 0 0
```

输入6：

```
666666666
```

输出6：

```
0 9 0
```

输入7：

```
1234890
```

输出7：

```
0 0 0
```