第 7 章 枚举与结构

课后练习题参考解答

1. 填空题

(1) 设有如下枚举类型定义：

```
enum language {Basic=3, Assembly, Ada=100, COBOL, Fortran};
```

则枚举量 Fortran 的值为 102 。

(2) 下面程序段有没有错误？如果有，错误在哪里？

```
struct abc{
    int a;
    int b;
    int c;
};

abc.a = 25;
abc.b = 25;
abc.c = 25;
```

答： abc 是结构类型标识符，不能作为结构变量来使用 。

(3) 若要定义一个类型名 D，使得声明语句`D s;` 等价于`double s;` ，则定义类型名 D 的语句为： `typedef double D;`

2. (1) 已知程序的运行结果如下：

```
求和结果直角坐标表示为: 1.00 + 1.00i
求和结果极坐标表示为: 1.41∠45.00°
```

请结合运行结果和程序中的注释填空完善如下程序：

```
#include <stdio.h>
#include <math.h>
#define PI 3.1415926

/*======声明======*/
//RECTANGULAR表示直角坐标, POLAR表示极坐标
enum Coordinate_Type { RECTANGULAR, POLAR };
//当t==RECTANGULAR时，a和b分别表示复数的实部和虚部
//当t==POLAR时，a和b分别表示复数的幅值和相角（弧度）
struct complex_struct {
    enum Coordinate_Type t;
    double a, b;
};
typedef enum Coordinate_Type CT;
typedef struct complex_struct CP;

/*======复数操作======*/
//转换复数的表示形式
//若复数为直角坐标表示，则转换为极坐标表示
//若复数为极坐标表示,则转化为直角坐标表示
CP convert_complex (CP z){
    CP new_z;
    if (z.t == POLAR) {
        new_z.t = RECTANGULAR;
        new_z.a = z.a * cos(z.b);
        new_z.b = z.a * sin(z.b);
    }
    else {
        new_z.t = POLAR;
        new_z.a = sqrt(z.a*z.a + z.b*z.b);
        new_z.b = atan2(z.b, z.a);
    }
    return new_z;
}
//求两个复数的和
//当参数t==RECTANGULAR时，返回的复数用直角坐标形式表示
//当参数t==POLAR时，返回的复数用极坐标形式表示
CP add_complex (CP z1, CP z2, CT t) {
    CP z3;
    if (z1.t == POLAR) z1 = convert_complex(z1);
    if (z2.t == POLAR) z2 = convert_complex(z2);
    z3.a = z1.a + z2.a;
    z3.b = z1.b + z2.b;
    z3.t = RECTANGULAR;
    if (t == POLAR) z3 = convert_complex(z3);
    return z3;
}
```

```
int main(void) {
    //定义两个复数z1和z2
    CP z1 = {RECTANGULAR, 0.7, 0.6};
    CP z2 = {POLAR, 0.5, atan2(0.4,0.3)};
    //求这两个复数的和，结果存入z3
    CP z3 = add_complex (z1, z2, RECTANGULAR);
    //以直角坐标的形式打印求和结果
    printf("求和结果直角坐标表示为: %.2lf + %.2lfi\n",z3.a,z3.b);
    //以极坐标形式打印求和过程和结果
    //要求相角用角度表示，不能不用弧度
    z3 = convert_complex(z3);
    printf("求和结果极坐标表示为: %.2lf∠%.2lf°\n",z3.a,z3.b*180/PI);
}
```

(2) 在(1)的基础上，写一个函数，用于计算两个复数的乘积。函数声明如下：

```
//求两个复数的乘积
//当参数t==RECTANGULAR时，返回的复数用直角坐标形式表示
//当参数t==POLAR时，返回的复数用极坐标形式表示
CP mul_complex (CP z1, CP z2, CT t);
```

参考程序：

```
//求两个复数的乘积
//当参数t==RECTANGULAR时，返回的复数用直角坐标形式表示
//当参数t==POLAR时，返回的复数用极坐标形式表示
CP mul_complex (CP z1, CP z2, CT t) {
    CP z3;
    if (z1.t == RECTANGULAR) z1 = convert_complex(z1);
    if (z2.t == RECTANGULAR) z2 = convert_complex(z2);
    z3.a = z1.a * z2.a;
    z3.b = z1.b + z2.b;
    z3.t = POLAR;
    if (t == RECTANGULAR) z3 = convert_complex(z3);
    return z3;
}
```

3. (1) 声明一个结构用于表示一个长方形，该结构中应包含长方形的左下顶点的坐标（包括横坐标和纵坐标）以及长方形的长（与 x 轴平行的边的长度）和高（与 y 轴平行的边的长度）；（假设长方形的边与坐标轴平行）

(2) 写一个函数，提示用户输入长方形左下顶点坐标，以及长方形的长和高，并以用户输入的信息构造一个长方形结构变量；

(3) 写一个函数用于计算长方形的面积；

(4) 写一个函数，用于在 xy 平面内，以长方形的左下顶点为旋转中心，将一

个长方形逆时针旋转 90°；

(5) 写一个函数，用于判断两个长方形是否存在重叠部分，若存在则返回 1，否则返回-1。

参考程序：

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

//第(1)问：声明长方形结构
struct rectangle_struct {
    double llx;    //左下顶点的横坐标
    double lly;    //左下顶点的纵坐标
    double width;  //长（与x轴平行的边的长度）
    double height; //高（与y轴平行的边的长度）
};
typedef struct rectangle_struct RT;

//第(2)问：创建一个RT类型的变量
RT creat_rectangle(void) {
    RT rect;
    printf("请依次输入长方形的左下顶点横坐标、纵坐标、长和高（以空格隔开）：\n");
    scanf("%lf%lf%lf%lf",&rect.llx,&rect.lly,&rect.width,&rect.height);
    return rect;
}

//第(3)问：计算长方形的面积
double calculate_area(RT rect) {
    return rect.width * rect.height;
}

//第(4)问：将长方形逆时针旋转90°
RT rotate_left(RT rect) {
    RT new_rect;
    new_rect.height = rect.width;
    new_rect.width = rect.height;
    new_rect.llx = rect.llx - rect.height;
    new_rect.lly = rect.lly;
    return new_rect;
}

//第(5)问：判断两个长方形是否重叠，是则返回重叠面积，否则返回-1
int overlap(RT rect1, RT rect2) {
    double x,y;
    if( ( (x=rect1.llx-rect2.llx)<0 && rect1.width>=-x || x>=0 && rect2.width>=x )
        && ( (y=rect1.lly-rect2.lly)<0 && rect1.height>=-y || y>=0 && rect2.height>=y ) )
        return 1;
    return -1;
}
```