12

Case Study of Structure

Case 1: Complex Number Arithmetic

- Problem definition
  - **Please write a program which can compute complex number. Your work must include :**
    - **Add**
    - **Subtraction**
    - **Multiplication**
    - **division**

Source Code (1/4)

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

struct complex{
    int re;//定義複數的實部
    int im;//定義複數的虛部
};

typedef struct complex COMP;//自定義一個複數的資料形態

COMP add(COMP number1, COMP number2);//加法函式
COMP subtraction(COMP number1, COMP number2);//減法函式
COMP multiplication(COMP number1, COMP number2);//乘法函式
void division(COMP number1, COMP number2);//除法函式

int main()
{
    COMP number[3];

    printf("Enter the first real part.\n");
    scanf("%d",&number[0].re);
    printf("Enter the first  imaginary part.\n");
    scanf("%d",&number[0].im);

    printf("Enter the second real part.\n");
    scanf("%d",&number[1].re);
    printf("Enter the second imaginary part.\n");
    scanf("%d",&number[1].im);
```

Source Code (2/4)

```
    number[2]=add(number[0],number[1]);
    printf("(%d + %di) + (%d + %di)=(%d + %di)\n",number[0].re,number[0].im
                                                  ,number[1].re,number[1].im
                                                  ,number[2].re,number[2].im);

    number[2]=subtraction(number[0],number[1]);
    printf("(%d + %di) - (%d + %di)=(%d + %di)\n",number[0].re,number[0].im
                                                  ,number[1].re,number[1].im
                                                  ,number[2].re,number[2].im);

    number[2]=multiplication(number[0],number[1]);
    printf("(%d + %di) * (%d + %di)=(%d + %di)\n",number[0].re,number[0].im
                                                  ,number[1].re,number[1].im
                                                  ,number[2].re,number[2].im);

    division(number[0],number[1]);


    return 0;
}
```

Source Code (3/4)

```
COMP add(COMP number1, COMP number2)
{
    COMP temp;

    temp.re=number1.re+number2.re;
    temp.im=number1.im+number2.im;

    return temp;
}

COMP subtraction(COMP number1, COMP number2)
{
    COMP temp;

    temp.re=number1.re-number2.re;
    temp.im=number1.im-number2.im;

    return temp;
}
```

Source Code (4/4)

```
COMP multiplication(COMP number1, COMP number2)
{
    COMP temp;

    temp.re=number1.re * number2.re - number1.im * number2.im;
    temp.im=number1.re * number2.im + number1.im * number2.re;

    return temp;
}

void division(COMP number1, COMP number2)
{
    COMP temp;
    int x;


    x=pow(number2.re,2) + pow(number2.im,2);
    temp.re=number1.re * number2.re + number1.im * number2.im;
    temp.im=number1.re * (number2.im * -1) + number1.im * number2.re;

    printf("(%d + %di) / (%d + %di) = (%d + %di) / %d\n",number1.re,number1.im
                                                        ,number2.re,number2.im
                                                        ,temp.re,temp.im,x);
}
```

Case 2: Fractional Number Arithmetic

- **Problem definition**
  - **Please write a program which can compute fractional number. Your work must include :**
    - **Add**
    - **Subtraction**
    - **Multiplication**
    - **division**

Source Code (1/10)

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

struct fraction {
    int integer;//整數部分
    int numerator;//分子
    int denominator;//分母
};

typedef struct fraction FRAC;

FRAC normalize (FRAC number);//假分數變換函式
FRAC add (FRAC number1, FRAC number2);//加法函式
FRAC subtraction (FRAC number1, FRAC number2);//減法函式
FRAC multiplication (FRAC number1, FRAC number2);//乘法函式
FRAC division (FRAC number1, FRAC number2);//除法函式
int reduction (int number1 ,int number2);//化減分數函式
```

Source Code (2/10)

```
int main()
{
    FRAC number[3];
    FRAC temp[2];


    //輸入分數一的值
    printf("Enter the first fraction.(integer part >0)\n");
    scanf("%d",&number[0].integer);
    while(!(number[0].integer >= 0)){
        printf("Enter the first fraction.(integer part >0)\n");
        scanf("%d",&number[0].integer);
    }

    printf("Enter the first fraction.(numerator part >0)\n");
    scanf("%d",&number[0].numerator);
    while(!(number[0].numerator > 0)){
        printf("Enter the first fraction.(numerator part >0)\n");
        scanf("%d",&number[0].numerator);
    }

    printf("Enter the first fraction.(denominator part >0)\n");
    scanf("%d",&number[0].denominator);
    while(!(number[0].denominator > 0)){
        printf("Enter the first fraction.(denominator part >0)\n");
        scanf("%d",&number[0].denominator);
    }
```

Source Code (3/10)

```
//輸入分數二的值
printf("Enter the second fraction.(integer part >0)\n");
scanf("%d",&number[1].integer);
while(!(number[1].integer >= 0)){
    printf("Enter the second fraction.(integer part >0)\n");
    scanf("%d",&number[1].integer);
}

printf("Enter the second fraction.(numerator part >0 )\n");
scanf("%d",&number[1].numerator);
while(!(number[1].numerator > 0)){
    printf("Enter the second fraction.(numerator part >0 )\n");
    scanf("%d",&number[1].numerator);
}

printf("Enter the second fraction.(denominator part >0 )\n");
scanf("%d",&number[1].denominator);
while(!(number[1].denominator > 0)){
    printf("Enter the second fraction.(denominator part >0)\n");
    scanf("%d",&number[1].denominator);
}

temp[0]=normalize(number[0]);
temp[1]=normalize(number[1]);
```

Source Code (4/10)

```
number[2]=add(temp[0],temp[1]); // 呼叫加法函式


if(number[2].numerator!=0){
    printf("%d %d/%d + %d %d/%d = %d %d/%d\n",number[0].integer, number[0].numerator,number[0].denominator,
                                              number[1].integer, number[1].numerator,number[1].denominator,
                                              number[2].integer, number[2].numerator,number[2].denominator);
}
else{
    printf("%d %d/%d + %d %d/%d = %d \n",number[0].integer, number[0].numerator,number[0].denominator,
                                         number[1].integer, number[1].numerator,number[1].denominator,
                                         number[2].integer);
}

number[2]=subtraction(temp[0],temp[1]); // 呼叫減法函式


if(number[2].numerator!=0){
    printf("%d %d/%d - %d %d/%d = %d %d/%d\n",number[0].integer, number[0].numerator,number[0].denominator,
                                              number[1].integer, number[1].numerator,number[1].denominator,
                                              number[2].integer, number[2].numerator,number[2].denominator);
}
else{
    printf("%d %d/%d - %d %d/%d = %d \n",number[0].integer, number[0].numerator,number[0].denominator,
                                         number[1].integer, number[1].numerator,number[1].denominator,
                                         number[2].integer);
}
```

Source Code (5/10)

```
    number[2]=multiplication(temp[0],temp[1]); //呼叫乘法函式

    if(number[2].numerator!=0){
        printf("%d %d/%d * %d %d/%d = %d %d/%d\n",number[0].integer, number[0].numerator,number[0].denominator,
                                                  number[1].integer, number[1].numerator,number[1].denominator,
                                                  number[2].integer, number[2].numerator,number[2].denominator);
    }
    else{
        printf("%d %d/%d * %d %d/%d = %d \n",number[0].integer, number[0].numerator,number[0].denominator,
                                             number[1].integer, number[1].numerator,number[1].denominator,
                                             number[2].integer);
    }
    number[2]=division(temp[0],temp[1]); //呼叫除法函式

    if(number[2].numerator!=0){
        printf("%d %d/%d / %d %d/%d = %d %d/%d\n",number[0].integer, number[0].numerator,number[0].denominator,
                                                  number[1].integer, number[1].numerator,number[1].denominator,
                                                  number[2].integer, number[2].numerator,number[2].denominator);
    }
    else{
        printf("%d %d/%d / %d %d/%d = %d \n",number[0].integer, number[0].numerator,number[0].denominator,
                                             number[1].integer, number[1].numerator,number[1].denominator,
                                             number[2].integer);
    }
    return 0;
}
```

Source Code (6/10)

```
FRAC add (FRAC number1, FRAC number2)
{
    FRAC temp[2];
    int reduction_number;
    FRAC result;

    temp[0]=number1;
    temp[1]=number2;

    temp[0].denominator=temp[0].denominator * number2.denominator;
    temp[0].numerator=temp[0].numerator * number2.denominator;
    temp[1].numerator=temp[1].numerator * number1.denominator;

    result.denominator=temp[0].denominator;
    temp[0].numerator=temp[0].numerator+temp[1].numerator;
    result.integer=0 + (temp[0].numerator / result.denominator);
    result.numerator=temp[0].numerator- result.integer*result.denominator;

    reduction_number=reduction(result.numerator,result.denominator);
    result.denominator =result.denominator / reduction_number;
    result.numerator=result.numerator / reduction_number;

    return result;
}
```

Source Code (7/10)

```
FRAC subtraction (FRAC number1, FRAC number2)
{
    FRAC temp[2];
    int reduction_number;
    FRAC result;

    temp[0]=number1;
    temp[1]=number2;

    temp[0].denominator=temp[0].denominator * number2.denominator;
    temp[0].numerator=temp[0].numerator * number2.denominator;
    temp[1].numerator=temp[1].numerator * number1.denominator;

    result.denominator=temp[0].denominator;
    temp[0].numerator=temp[0].numerator - temp[1].numerator;
    result.integer=0 + (temp[0].numerator / result.denominator);
    result.numerator=temp[0].numerator- result.integer*result.denominator;

    reduction_number=reduction(result.numerator,result.denominator);
    result.denominator =result.denominator / reduction_number;
    result.numerator=result.numerator / reduction_number;

    return result;
}
```

Source Code (8/10)

```
FRAC multiplication  (FRAC number1, FRAC number2)
{
    FRAC temp[2];
    int reduction_number;
    FRAC result;

    temp[0]=number1;
    temp[1]=number2;

    result.integer=0;

    result.denominator=temp[0].denominator * temp[1].denominator;
    result.numerator=temp[0].numerator * temp[1].numerator;
    result.integer=0 + (result.numerator / result.denominator);
    result.numerator=result.numerator % result.denominator;

    reduction_number=reduction(result.numerator,result.denominator);
    result.denominator =result.denominator / reduction_number;
    result.numerator=result.numerator / reduction_number;

    return result;
}
```

Source Code (9/10)

```
FRAC division (FRAC number1, FRAC number2)
{
    FRAC temp[2];
    int reduction_number;
    FRAC result;

    temp[0]=number1;
    temp[1]=number2;

    result.integer=0;

    result.denominator=temp[0].denominator * temp[1].numerator;
    result.numerator=temp[0].numerator * temp[1].denominator;
    result.integer=0 + (result.numerator / result.denominator);
    result.numerator=result.numerator % result.denominator;

    reduction_number=reduction(result.numerator,result.denominator);
    result.denominator =result.denominator / reduction_number;
    result.numerator=result.numerator / reduction_number;

    return result;
}
```

Source Code (10/10)

```
FRAC normalize (FRAC number)
{
    FRAC temp;

    temp.denominator=number.denominator;
    temp.integer=number.integer;
    temp.numerator=number.numerator;

    temp.numerator=temp.numerator+temp.denominator*temp.integer;
    temp.integer=0;

    return temp;
}

int reduction (int number1 ,int number2)
{
    int temp;

    if(abs(number1)>abs(number2))
        temp=abs(number2);
    else
        temp=abs(number1);

    while(temp>=1){
        if(number1%temp==0 && number2%temp==0)
            return temp;
        else
            temp--;
    }

    return 1;
}
```