26. [Sebesta, 2000] Escreva um tipo de dado abstrato para números complexos, incluindo operações de adição, de subtração, de multiplicação, de divisão, de extração de cada uma das partes de um número complexo e de construção de um número complexo a partir de duas constantes, variáveis ou expressões de ponto flutuante. Use o Modula-2, a Ada, o C ou o Pascal.

Resposta codificada em C, disponível em https://onlinegdb.com/m-YKQaShz

```
// Arquivo Complex.h

typedef struct complex Complex;

Complex* create (double, double);

void destroy (Complex*);

double getRealPart (Complex*);

double getImaginaryPart (Complex*);

Complex* addition (Complex*, Complex*);

Complex* subtraction (Complex*, Complex*);

Complex* multiplication (Complex*, Complex*);

Complex* division (Complex*, Complex*);

void print (Complex*);
```

```
// Arquivo Complex.c

#include <stdio.h>
#include <stdlib.h>
#include "Complex.h"

struct complex {
    double realPart;
    double imaginaryPart;
};

Complex* create (double realPart, double imaginaryPart)
{
    Complex* number = (Complex*) malloc(sizeof(Complex));
    if (number != NULL)
    {
        number->realPart = realPart;
        number->imaginaryPart = imaginaryPart;
    }
    return number;
}

void destroy (Complex* number)
{
    free(number);
}

double getRealPart (Complex* number)
{
    return number->realPart;
}

double getImaginaryPart (Complex* number)
{
    return number->imaginaryPart;
}

Complex* addition (Complex* numberA, Complex* numberB)
{
    double realPart = numberA->realPart + numberB->realPart;

    double imaginaryPart = numberA->imaginaryPart + numberB->imaginaryPart;

    return create(realPart, imaginaryPart);
}

Complex* subtraction (Complex* numberA, Complex* numberB)
{
    double realPart = numberA->realPart - numberB->realPart;

    double imaginaryPart = numberA->imaginaryPart - numberB->imaginaryPart;

    return create(realPart, imaginaryPart);
}
```

```
Complex* multiplication (Complex* numberA, Complex* numberB)
{
    double realPart = (numberA->realPart * numberB->realPart) -
                      (numberA->imaginaryPart * numberB->imaginaryPart);

    double imaginaryPart = (numberA->realPart * numberB->imaginaryPart) +
                           (numberA->imaginaryPart * numberB->realPart);

    return create(realPart, imaginaryPart);
}

Complex* division (Complex* numberA, Complex* numberB)
{
    double realPart = ((numberA->realPart * numberB->realPart) +
                       (numberA->imaginaryPart * numberB->imaginaryPart)) /
                      ((numberB->realPart * numberB->realPart) +
                       (numberB->imaginaryPart * numberB->imaginaryPart));

    double imaginaryPart = ((numberB->realPart * numberA->imaginaryPart) -
                            (numberA->realPart * numberB->imaginaryPart)) /
                           ((numberB->realPart * numberB->realPart) +
                            (numberB->imaginaryPart * numberB->imaginaryPart));

    return create(realPart, imaginaryPart);
}

void print (Complex* number)
{
    printf("(%f, %f)", number->realPart, number->imaginaryPart);
}
```

```
// Arquivo main.c

#include <stdio.h>
#include "Complex.h"

int main()
{
    Complex* complex1 = create(6, -4);

    Complex* complex2 = create(4,  2);

    Complex* complexAddition = addition(complex1, complex2);

    print(complexAddition);

    destroy(complexAddition);

    printf("\n");

    Complex* complexSubtraction = subtraction(complex1, complex2);

    print(complexSubtraction);

    destroy(complexSubtraction);

    printf("\n");

    Complex* complexMultiplication = multiplication(complex1, complex2);

    print(complexMultiplication);

    destroy(complexMultiplication);

    printf("\n");

    Complex* complexDivision = division(complex1, complex2);

    print(complexDivision);

    destroy(complexDivision);

    destroy(complex1);

    destroy(complex2);

    return 0;
}
```