

Tutorial

Programming & Data Structure: CS 11001

Section - 4/D

Department of Computer Science and
Engineering

I.I.T. Kharagpur

Spring Semester: 2013 - 2014 (13.03.2014)

Download

**Download the file tut130314.pdf from
Programming & Data Structures ... of**

<http://cse.iitkgp.ac.in/~goutam>

**View the file using the command `acroread` & or
`xpdf` &**

A complex Example

A complex number (or its approximation) is not a built-in datatype of C language^a.

^aNot really - C99 has a data type `_Complex`, `complex`. See the man page of `complex.h`.

C99 complex in GCC

```
#include <stdio.h>
#include <complex.h>
int main() // gComplex1.c
{
    complex double x, y ;
    x = 1.0 + 2.0i; // real + imaginary part
                    // Operator overloading
    y = ~x;        // complex conjugate
    printf("x: %lf+j%lf\n",creal(x),cimag(x));
    printf("y: %lf - j%lf\n",creal(y),-cimag(y));
    return 0;
}
```

Note

But we shall assume that the data type **complex** is not supported in C language. We shall define and support as a user defined data type.

Data in a Complex Number

A complex number z is written as $z = a + jb$, where a is the real part and b is the imaginary part. The value of $j = \sqrt{-1}$.

Data in a Complex Number

But the actual data in z may be viewed as an ordered pair (vector) of real numbers i.e.

$z = (a, b)$, where the first component is the real part and the second component is the imaginary part. The collection of complex numbers \mathbb{C} is identified with \mathbb{R}^2 .

Approximation for Representations

The first question is how to represent a complex number in a C program. We have already identified a complex number with a pair of reals. But a real number cannot have an exact representation in a computer. It is approximated as a floating-point number (`float` or `double`). So a complex number may be approximated as an **ordered pair of floating-point numbers**.

Product Constructor in C

C language provides a type constructor for product called a **structure**. A structure may have data of different types with a **tag/name** for each component. A structure corresponding to the type *complex* is

```
struct complexType {  
    double real, imag ;  
};
```

struct complexType is a new data type

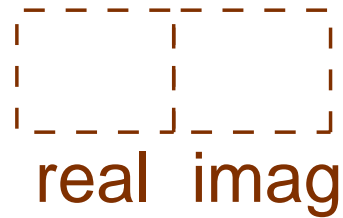
Note

A data type is a **plan** for a data object. The defined data type **struct^a complexType** has two components or members, each of type **double**.

^a'**struct**' is a **reserve word** like **if, while, int, return** etc. They have specific meaning in a language and cannot be used as name of an object.

```
struct complexType {  
    double real, imag  
}
```

data type



A Good Name to Data Type

A new name can be given to a data type using `typedef` e.g. `typedef int integer` creates the another name `integer` for the data type `int`. We use `typedef` to give a better and shorter name to `struct complexType`.

Type: complex

```
struct complexType {  
    double real, imag ;  
};  
typedef struct complexType complex;
```

or we may have

```
typedef struct {  
    double real, imag ;  
} complex;
```

Language Support

We can declare variable, array, pointer variable of type **complex**.

```
complex a, b, c[5], *p ;
```

We can use **sizeof**, **&**, *****, **=** etc. on a data of type **complex**.

A *structure* can be passed as a **parameter** to a function and can be returned as a **value** from a function.

There are operators specific to structure e.g.

. **->**

Tutorial IX.1

1. What are the sizes of **a**, **c** and **p**?
2. How do we initialize **a**?
3. How to access the **real** and **imag** fields of the variable **a**?
4. How to access the **real** fields of **c[2]**?
5. Initialize the elements of **c[]** with $1 + i2$, $2.5 + i3.5$, $i7.5$, 10.25 and 0 .
6. How can we assign the address of **a** to the

pointer **p**?

7. How can we access the **real** field of **a** through the pointer **p** (after the address of **a** is assigned to **p**).

Tutorial IX.2

1. Write a function `complex readComplex()` to read a pair of floating point numbers from the keyboard, construct a complex number and return it.
2. Can you read from keyboard an input in the form $a + ib$ ($2.5 + i3.5$), where `a, b` are two floating point numbers?
3. Write a function `void`

`writeComplex(complex)` to write a complex number in the form $a \pm ib$.

4. Write a function

`complex addComplex(complex, complex)`.

Tutorial IX.3

Consider the structure

```
struct abc {  
    char x; int y; double z; } a, *p;
```

1. What is the size of **a**?
2. How do you print the addresses of different components of **a**?
3. Consider the statement **p = &a;**. Give an equivalent expression for **p -> y**.

Tutorial IX.4

What is the output of the following code.

```
#include <stdio.h>
struct abc {char *p;};
struct xyz {struct abc *p;};
int main() // tutIX.4.c
{
    char iitKgp[] = "IIT Kharagpur";
    struct abc s = {iitKgp} ;
    struct xyz t = {&s}, *p = &t;
    printf("%s\n", p -> p -> p) ;
    return 0;
}
```

Tutorial IX.5

Give C definitions of the following three data types.

1. A **circle** on a 2-D plane.
2. A **straight line** on a 2-D plane in $y = mx + c$ form.
3. An **n -dimensional vector**.

Tutorial IX.6

List a few essential functions for each of these data types.