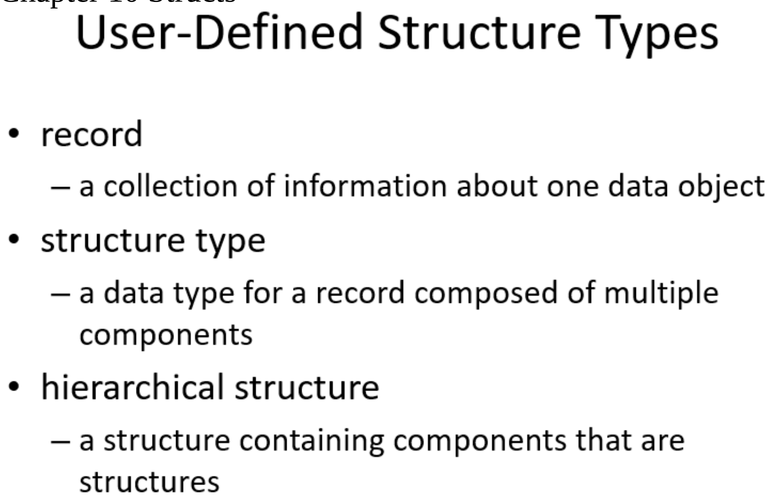
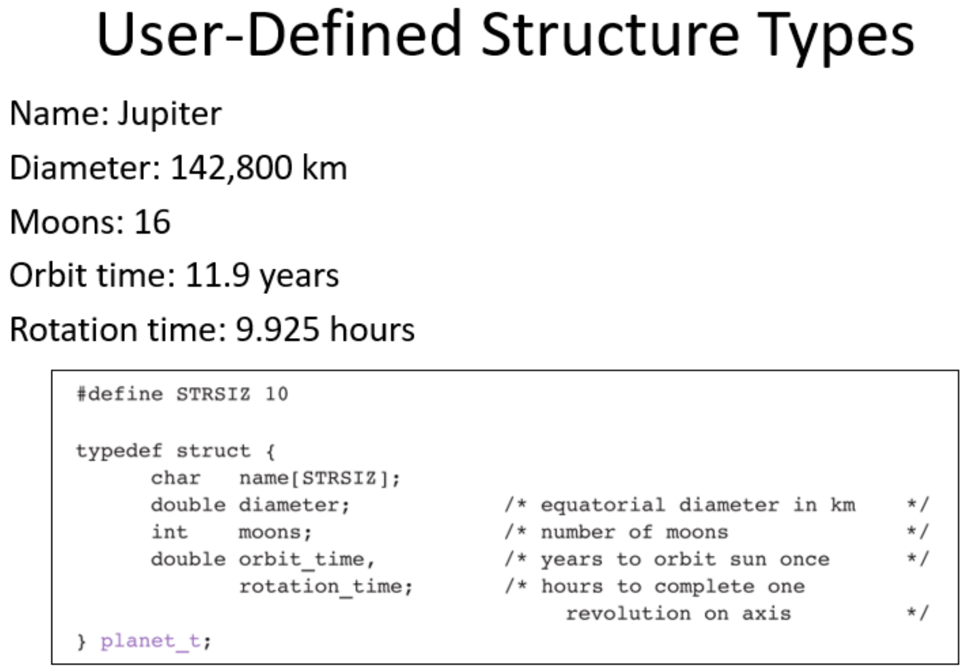
CS 210 **November 1, 2016**
Day 17

Chapter 10 Structs



User-Defined Structure Types

- record
  - a collection of information about one data object
- structure type
  - a data type for a record composed of multiple components
- hierarchical structure
  - a structure containing components that are structures



User-Defined Structure Types

Name: Jupiter

Diameter: 142,800 km

Moons: 16

Orbit time: 11.9 years

Rotation time: 9.925 hours

```
#define STRSIZ 10

typedef struct {
      char   name[STRSIZ];
      double diameter;           /* equatorial diameter in km    */
      int    moons;              /* number of moons              */
      double orbit_time,         /* years to orbit sun once      */
             rotation_time;      /* hours to complete one
                                      revolution on axis         */
} planet_t;
```

This example shows how to declare a struct and use those values in an equation.

```
#include<stdio.h>
#include<math.h>
typedef struct
{
    double x;
    double y;
}point_t;
//Note that since structs are typedefs and easily
//  confused with simple variables, they are
//  typically named with lower case letter followed
//  by _t as in point_t
int main()
{
    point_t p1, p2;
    double distance;
    p1.x = 12.5;
    p1.y = 9.2;
    p2.x = 5.3;
    p2.y = -9.1;
    distance = sqrt(pow(p1.x - p2.x, 2) + pow(p1.y-p2.y,2));
    printf("The distance from %3.2f, %3.2f to %3.2f, %3.2f is %3.2f\n",
                   p1.x, p1.y, p2.x, p2.y, distance);
    //Structs can be copied like variables
    p2 = p1;
    printf("P1 =  %3.2f, %3.2f, P2 =  %3.2f, %3.2f \n",
                 p1.x, p1.y, p2.x, p2.y);
    return 0;
}
/*Prints the following:
The distance from 12.50, 9.20 to 5.30, -9.10 is 19.67
P1 =  12.50, 9.20, P2 =  12.50, 9.20
Press any key to continue . . .
*/
```

Note that even though you can copy one struct to another as in p1 = p2 you cannot use the comparison operators on structs: **`if(p1 < p2)`** ***is illegal.***

This example show how to use a struct as a parameter passed to a function

```c
#include<stdio.h>
#include<math.h>
typedef struct
{
    double x;
    double y;
}point_t;
double FindDistance(point_t p1, point_t p2);
int main()
{
    double distance;
    point_t p1, p2;
    p1.x = 0; p1.y = 0;
    p2.x = 5; p2.y = 5;
    distance = FindDistance(p1, p2);
    printf("The distance between (%6.2f, %6.2f) and (%6.2f, %6.2f) is %6.2f\n",
                p1.x, p1.y, p2.x, p2.y, distance);
    return 0;
}
double FindDistance(point_t p1, point_t p2)
{
   double d;
   d = sqrt(pow(p1.x-p2.x, 2)+pow(p1.y-p2.y, 2));
   return d;
}
/*The distance between (  0.00,   0.00) and (  5.00,   5.00) is   7.07
Press any key to continue . . .
*/
```

Structs can also be passed by reference using * and & just like variables.

```c
#define PI 3.14159265359
#include<stdio.h>
#include<math.h>
typedef struct
{
    double x;
    double y;
}point_t;
//This function returns a point p2 a distance d and
//  angle theta from p1.
void FindPoint(point_t p1, point_t *p2, double d, double theta);
int main()
{
    double d, theta;
    point_t p1, p2;
    p1.x = 0; p1.y = 0;
    d = 10; theta = 45;
    FindPoint(p1, &p2, d, theta);
    printf("The new point is at (%6.2f, %6.2f)\n",
                p2.x, p2.y);
    return 0;
}
void FindPoint(point_t p1, point_t *p2, double d, double theta)
{
    (*p2).x = d * cos(theta*PI/180) + p1.x;
    (*p2).y = d * sin(theta*PI/180) + p1.y;
}
/*Note that writing *p2.x because, according to table
10.1 p. 576 the dot operator is done before * operator.

The notation (*p2).x is awkward so there is a new symbol
that replaces it.  We can write p2 -> x instead.
*/
//void FindPoint(point_t p1, point_t *p2, double d, double theta)
//{
//    p2 -> x = d * cos(theta*PI/180) + p1.x;
//    p2 -> y = d * sin(theta*PI/180) + p1.y;
//}
```

The arrow -> operator has the awkward name of *indirect component selection operator*.

It is also possible to create an array of structs. This is the shortest distance problem from Asn 06

```c
#ifdef _MSC_VER
#define _CRT_SECURE_NO_WARNINGS
#endif
#include<stdio.h>
#include<math.h>
typedef struct
{
    double x;
    double y;
}point_t;
double FindLine(point_t p1[], int numPoints);
double DistanceBetweenPoints(point_t pt1, point_t pt2);

int main()
{
    double d;
    point_t p1[20];
    int i, totalPts, status;
    FILE *inFilep;
    inFilep = fopen("Asn06.txt", "r");
    i = 0;
    while ((status = fscanf(inFilep, " %lf,%lf", &p1[i].x, &p1[i].y)) != EOF && i < 20)
    {
        printf("%lf, %lf\n", p1[i].x, p1[i].y);
        i++;
    }
    printf("Number of items = %d\n", i);
    totalPts = i;
    fclose(inFilep);
    d = FindLine(p1, totalPts);
    printf("Shortest distance is %6.2f\n", d);
    return 0;
}
double FindLine(point_t p1[], int numPoints)
{
    int i, j;
    double shortest;
    double a;
    shortest = DistanceBetweenPoints(p1[0], p1[1]);

    for (i = 0; i<numPoints - 1; i++)
    {
        for (j = i + 1; j<numPoints; j++)
        {
            a = DistanceBetweenPoints(p1[i], p1[j]);
            if(shortest > a)
               shortest = a;
        }
    }
    return shortest;
}
double DistanceBetweenPoints(point_t pt1, point_t pt2)
{
    double d1, d2;
    d1 = pt1.x - pt2.x;
    d2 = pt1.y - pt2.y;
    return sqrt(d1*d1 + d2*d2);
}
```

CS 210 **November 1, 2016**
Structs

A complex number has a real part and an imaginary part. Define a struct as shown in the main program outline below. The program defines a complex number struct and three complex variable c1, c2, and c3. It calls a function called MultiplyComplex which accepts two complex arguments and returns a complex result. The function mulitplies the two complex arguments to get the result. Complex multiplication is defined as follows:
If `x = a + ib` and `y = c + id` then
`z = x * y = (a*c – b*d) + i(ad + cb)`
where `i` is $\sqrt{-1}$

```
typedef struct
{
    double real;
    double imag;
} complex_t;
//Put your function prototype here.
int main()
{
   complex_t c1, c2, c3;
   c1.real = 4; c1.imag = 3;
   c2.real = 1; c2.imag = 2;
   c3 = MultiplyComplex(c1, c2);
   printf("(%4.1f + i%4.1f) * (%4.1f + i %4.1f) = (%4.1f + i%4.1f)\n",
       c1.real, c1.imag, c2.real, c2.imag, c3.real, c3.imag);
   return 0;
}
//Put your function definition here.
```

Turn in a hard copy of your source code.