Programming with C I

Fangtian Zhong
CSCI 112

Gianforte School of Computing
Norm Asbjornson College of Engineering
E-mail: fangtian.zhong@montana.edu

2024.03.18

Objectives

- To learn how to declare a **struct** data type which consists of several data fields, each with its own name and data type
- To understand how to use a **struct** to store data for a structured object or record
- To learn how to use dot notation to process individual fields of a structured object
- To learn how to use **structs** as function parameters and to return function results
- To understand the relationship between parallel arrays and arrays of structured objects

User-Defined Structure Types

- Name: Jupiter
- Diameter: 142,800 km
- Moons: 16
- Orbit time: 11.9 years
- Rotation time: 9.925 hours

I will always use this syntax

```
#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;
```

Individual Components of a Structured Data Object

- **direct component selection operator**
  - a period placed between a structure type variable and a component name to create a reference to the component

```
planet_t p1;
p1.moons = 10;
printf("p1 has %d moons\n", p1.moons);
```

Individual Components of a Structured Data Object

```
strcpy(current_planet.name, “Jupiter”);
current_planet.diameter = 142800;
current_planet.moons = 16;
current_planet.orbit_time = 11.9;
current_planet.rotation_time = 9.925;
```

Variable current_planet, a structure of type planet_t

.name	J u p i t e r \ 0 ? ?
.diameter	142800.0
.moons	16
.orbit_time	11.9
.rotation_time	9.925

User-Defined Structure Types

- **Another syntax:**

```
struct Planet {
        char name[70];
        double diameter;
        int moons;
        double orbit_time,
                rotation_time;
};
// in a function
struct Planet p1, p2;
```

Structure Data Type as Input and Output Parameters

- When a structured variable is passed as an input argument to a function, all of its component values are copied into the components of the function's corresponding formal parameter.

Structure Data Type as Input and Output Parameters

- When such a variable is used as an output argument, the address-of operator must be applied in the same way that we would pass output arguments of the standard types char, int, and double.

Figure Function with a Structured Input Parameter

```
/*
 * Display with labels all components of a planet_t structure
 */
void
print_planet(planet_t pl)  /*input - one planet strcuture  */
{
        printf("%s\n", pl.name);
        printf("    Equatorial diameter: %.0f  Km\n", pl.diameter);
        printf("    Number of moons: %d\n", pl.moons);
        printf("    Time to complete one orbit of the sun: %.2f years\n", pl.orbit_time);
        printf("    Time to complete one rotation on axis: %.4f hours\n", pl. rotation_time);
}
```

Figure Function Comparing Two Structured Values for Equality

```
# include <string.h>

 */
 * Determines whether or not the components of planet_1 and planet_2 match
 */
int
planet_equal(planet_t planet_1,  /* input - planets to                    */
             planet_t planet_2)  /*         compare                       */
{
        return (strcmp(planet_1.name, planet_2.name) == 0     &&
                planet_1.diameter == planet_2.diameter        &&
                planet_1.moons == planet_2.moons              &&
                planet_1.orbit_time == planet_2.orbit_time    &&
                planet-1.rotation_time == planet_2.rotation_time);
}
```

Structure Data Type as Input and Output Parameters

indirect component selection operator

- the character sequence -> placed between a pointer variable and a component name creates a reference that follows the pointer to a structure and selects the component

Figure Function with a Structured Input Argument

```
/*
 * Fills a type planet_t structure with input data. Integer returned as
 * function result is success/failure/EOF indicator.
 *        1 => successful input of one planet
 *        0 => error encountered
 *        EOF => insufficient data before end of file
 * In case of error or EOF, value of type planet_t output argument is underfined.
 */
 int
scan_planet(planet_t *plnp)  /* output -address of planet_t structure to fill        */
```

continued

Figure Function with a Structured Input Argument

```
{
        int result;

        result = scanf("%s%1f%d%1f%1f", (*plnp).name,
                                        &(*plnp).diameter,
                                        &(*plnp).moons,
                                        &(*plnp).orbit_time,
                                        &(*plnp).rotation_time);

        if (result == 5)
                result = 1;
        else if (result != EOF)
                result = 0;

        return (result);
}
```

continued

Figure Data Areas of main and scan_planet During Execution of status = scan_planet(&current_planet);

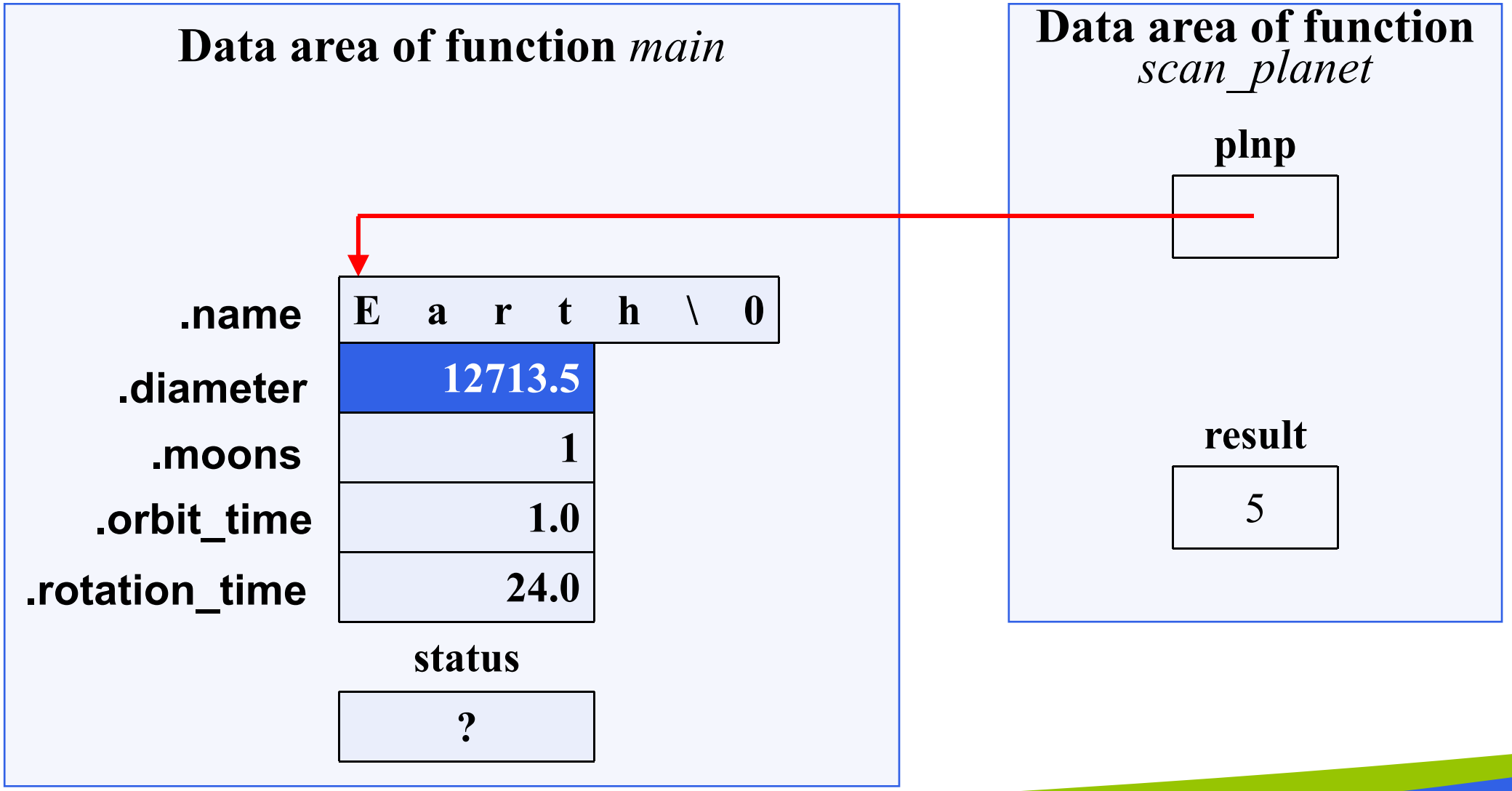

Table Step-by-Step Analysis of Reference &(*plnp).diameter

Reference	Type	Value
plnp	planet_t *	address of structure that main refers to as current_planet
*plnp	planet_t	structure that main refers to as current_planet
(*plnp).diameter	double	12713.5
&(*plnp).diameter	double *	address of colored component of structure that main refers to as current_planet

THE END

Fangtian Zhong
CSCI 112

Gianforte School of Computing
Norm Asbjornson College of Engineering
E-mail: fangtian.zhong@montana.edu

2024.03.18