C Programming Language

Lecture 12 (12/15/2000)

Instructor: Wen-Hung Liao
E-mail: whliao@cs.nccu.edu.tw
Extension: 62028
Outline

- User-Defined Structure Types
- Structure Type Data as Input and Output Parameters
- Functions Whose Result Values are Structured
- Case Study I: Complex Numbers
- Case Study II: Parallel Arrays and Arrays of Structures
- Union Types (optional)
- Common Programming Errors
User-Defined Structure Types

- Build your own data type using the basic ones.
- A structure can have individual components that contain data of different types.
- Use the typedef reserved word
  ```c
  typedef struct { ...} newtype_t
  ```
Example

- A planet with name, diameter, moons, orbit time and rotation time can be declared as a `structure` type `planet_t` defined as follows:

```c
#define STRSIZ 10
typedef struct {
    char name[STRSIZ];
    double diameter;
    int moons;
    double orbit_time,
        rotation_time;
} planet_t;
```
Initialization and Referencing

- planet_t planet1,
  planet2={" ", 0,0,0,0};
- planet2.name holds the value ""
- planet2.diameter holds the value 0
- planet2.moons holds the value 0
- planet2.orbit_time holds 0.0
- planet2.rotation_time holds 0.0
Hierarchical Structure

- A structure type variable can be a component of another data structure.
- Example:
  ```c
  typedef struct {
    double diameter;
    planet_t planets[9];
    char galaxy[STRSIZ];
  } solar_sys_t;
  ```
Structure Type Definition

- Syntax:

```c
typedef struct {
    type1 id_list1;
    type2 id_list2;
    .
    .
    typen id_listn;
} struct_type;
```
Component Selection

- We can reference a component of a structure by using the **direct component selection operator**, which is a period.

- For example, the following operations are valid:
  ```c
  strcpy(planet1.name,"Jupiter");
  planet1.diameter = 142800;
  planet1.moons = 16;
  planet1.orbit_time = 11.9;
  planet1.rotation_time = 9.925;
  ```
Manipulating Whole Structure

- The name of a structure type variable used with no component selection operator refers to the entire structure.
- Assignment is allowed: (comparison isn’t !!)
  \[ \text{planet2} = \text{planet1}; \]
- This is equivalent to:
  \[
  \text{strcpy(planet2.name,planet1.name);}
  \text{planet2.diameter= planet1.diameter;}
  \ldots
  \text{planet2.rotation_time= planet1.rotation_time;}
  \]
Structure Type Data as I/O Parameter

- 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 parameter. ==> Call by value
- When used as an output argument, the address-of operator must be applied.
Function with Structured Input

```c
void
print_planet(planet_t pl) /* input - one planet structure */
{
    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);
}
```
int planet_equal(planet_t planet_1,
    planet_t planet_2)
{
    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);
}
Function with Structured Output

```c
Int scan_planet(planet_t *plnp) {
    /* output - address of planet_t structure to fill */
    int result;


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

    return (result);
}
```
Indirect Component Selection

- In the previous example we see statements such as:
  
  ```c
  &(*plnp).diameter,
  &(*plnp).moons,
  &(*plnp).orbit_time,
  ```

- When the variable is a pointer to a structured data, we use the indirect component selection operator `->` for convenience.

- That is,
  
  ```c
  (*structp).component is equivalent to structp->component
  ```
## Analysis of &(*plnp).diameter

<table>
<thead>
<tr>
<th>Reference</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>plnp</td>
<td>planet_t *</td>
<td>Address of structure</td>
</tr>
<tr>
<td>*plnp</td>
<td>planet_t</td>
<td>structure</td>
</tr>
<tr>
<td>(*plnp).diameter</td>
<td>double</td>
<td>12713.5</td>
</tr>
<tr>
<td>&amp;(*plnp).diameter</td>
<td>double *</td>
<td>Address of component</td>
</tr>
</tbody>
</table>
Functions that Return Structure

Example: (return values, not address)

```c
planet_t get_planet(void)
{
    planet_t planet;

    scanf("%s%lf%d%lf%lf", planet.name,
                   &planet.diameter,
                   &planet.moons,
                   &planet.orbit_time,
                   &planet.rotation_time);

    return (planet);
}
```
Case Study: Complex Numbers

- We will define the data structure as well as the functions associated with the data.
- This is often referred to as an abstract data type, a fundamental concept in object-oriented programming languages such as C++.
- During the design process we focus on what are needed, but not how things can be done.
Specification

- Structure: an object of type `complex_t` that consists of a pair of type `double` values.
- Operators: input, output, add, subtract, multiply, divide, magnitude.
- Page 556-563 of text shows the design as well as the source code.
- Your assignment is to provide the multiply and divide operation. (Due on 12/22/2000)
Parallel Array vs Structure Arrays

- Using arrays of structure is more natural and convenient than parallel arrays.
- Example:
  ```c
  typedef struct {
    int id;
    double gpa;
  } student_t;
  student_t student_list[100];
  ```
- Reference: `studentlist[20].id`
Common Programming Errors

- Syntax
- Direct vs indirect selection
- Input/Output
- Comparison
Your Homework

- Read Chapter 11 of textbook
- Complete the complex number case study.
- Programming assignment 10.6: Recursive binary search (page 537 of text)
- Due on 12/22/2000. Please send the source code to your TA via e-mail.