Structures

Gabriel Hugh Elkaim
Winter 2014

Definition

Structures are collections of variables grouped together under a common name. The variables within a structure are referred to as the structure’s members, and may be accessed individually as needed.

- Structures:
  - May contain any number of members
  - Members may be of any data type
  - Allow group of related variables to be treated as a single unit, even if different types
  - Ease the organization of complicated data

Structures are collections of variables grouped together under a common name. The variables within a structure are referred to as the structure’s members, and may be accessed individually as needed.

- Structures:
  - May contain any number of members
  - Members may be of any data type
  - Allow group of related variables to be treated as a single unit, even if different types
  - Ease the organization of complicated data

Example Syntax

```c
// Structure to handle complex numbers
struct complex
{
    float re; // Real part
    float im; // Imaginary part
}
```

// How to Create a Structure Definition

```c
struct structName
{
    type1 memberName1;
    ...
    typeN memberNameN;
}
```

Members are declared just like ordinary variables

Example Syntax

```c
// Structure to handle complex numbers
struct complex
{
    float re;
    float im;
}
```

// Declare x and y of type complex

```c
y { q, r }; // Structure to handle complex numbers
struct complex
{
    type1 memberName1;
    ...
    typeN memberNameN;
} varName1, ..., varNameN;
```
Structures

**How to Declare a Structure Variable (Method 2)**

If `structName` has already been defined:

```c
struct structName varName_1, ..., varName_n;
```

Example

```c
struct complex {
    float re;
    float im;
} ...
struct complex x, y; // Declare x and y of type complex
```

**How to Use a Structure Variable**

```c
struct complex {
    float re;
    float im;
} x, y; // Declare x and y of type complex
```

**How to Create a Structure Type with `typedef`**

```c
typedef struct structureTag {
    type_1 memberName_1;
    ... type_n memberName_n;
} typeName;
```

Example

```c
// Structure type to handle complex numbers
typedef struct {
    float re; // Real part
    float im; // Imaginary part
} complex;
```

**How to Declare a Structure Type Variable**

```c
typedef struct {
    float re;
    float im;
} complex;
```

Example

```c
typedef struct {
    float re;
    float im;
} complex;
```

```c
complex x, y; // Declare x and y of type complex
```

If `typeName` has already been defined:

```c
typeName varName_1, ..., varName_n;
```

The keyword `struct` is no longer required!
**Structures**

How to Initialize a Structure Variable at Declaration

If `typeName` or `structName` has already been defined:

```
typeName varName = {const1, ..., constn};
- or -
struct structName varName = {const1, ..., constn};
```

**Example**

```
typedef struct {
    float re;
    float im;
} complex;
...
complex x = (1.25, 2.50); // x.re = 1.25, x.im = 2.50
```

**Structures**

Arrays and Pointers with Strings

- Strings:
  - May be assigned directly to `char` array member only at declaration
  - May be assigned directly to a pointer to `char` member at any time

**Example: Structure**

```
struct strings {
    char a[4];
    char *b;
} str;
```

**Example: Initializing Members**

```
int main(void) {
    str.a[0] = ‘B’;
    str.a[1] = ‘a’;
    str.a[2] = ‘d’;
    str.a[3] = ‘0’;
    str.b = “Good”;
}
```

**Structures**

Nesting Structures

**Example**

```
typedef struct {
    float x;
    float y;
} point;
typedef struct {
    point a;
    point b;
} line;
int main(void) {
    line m;
    m.a.x = 1.2;
    m.a.y = 7.6;
    m.b.x = 38.5;
    m.b.y = 17.8;
    ...
```

**Structures**

How to Declare a Pointer to a Structure

If `typeName` or `structName` has already been defined:

```
typeName *ptrName;
- or -
struct structName *ptrName;
```

**Example 1**

```
typedef struct {
    float re;
    float im;
} complex;
...
complex *p;
```

**Example 2**

```
struct complex {
    float re;
    float im;
} ...
struct complex *p;
```
Structures

How to Use a Pointer to Access Structure Members

If ptrName has already been defined:

\[ p \rightarrow x \]

PtrName->memberName

Pointer must first be initialized to point to the address of the structure itself: ptrName = &structVariable;

Syntax

Example: Definitions

```c
typedef struct
{
    float re;
    float im;
} complex; //complex type
... complex x; //complex var
complex *p; //ptr to complex
```

Example: Usage

```c
int main(void)
{
    p = &x; //Set x.re = 1.25 via p
    p->re = 1.25;
    //Set x.im = 2.50 via p
    p->im = 2.50;
}
```

Structures

Creating Arrays of Structures

If typeName or structName has already been defined:

Syntax

```c
typeName arrName[n];
```

Example

```c
typedef struct
{
    float re;
    float im;
} complex;
...
complex a[3];
```

Structures

Using Arrays of Structures

If arrName has already been defined:

Syntax

```c
arrName[n].memberName
```

Example: Definitions

```c
typedef struct
{
    float re;
    float im;
} complex;
... complex a[3] = {{1.2, 2.5}, {3.9, 6.5}, {7.1, 8.4}};
```

Example: Usage

```c
int main(void)
{
    a[0].re = 1.25;
    a[0].im = 2.50;
    ... complex a[3];
```
Example Structures

How to Pass Structures to Functions

```c
typedef struct {
    float re;
    float im;
} complex;

void display(complex x) {
    printf("(%f + j%f)\n", x.re, x.im);
}

int main(void) {
    complex a = {1.2, 2.5};
    complex b = {3.7, 4.0};
    display(a);
    display(b);
}
```

Exercise 14 Structures

- Open the project’s workspace:
  /Examples/Lab14.zip -> Load “Lab14.mcw”

Lab Exercise 14 Structures

Solution: Steps 1 and 2

```c
powerDiff1 = (PMax1.v * PMax1.i) - (PMin1.v * PMin1.i);
powerDiff2 = (PMax2.v * PMax2.i) - (PMin2.v * PMin2.i);
powerDiff3 = (PMax3.v * PMax3.i) - (PMin3.v * PMin3.i);
```

/*############################################################################ # STEP 1: Calculate the difference between maximum and minimum power in ... */

/*############################################################################ # STEP 2: Calculate the difference between maximum and minimum power in */
Exercise 14

Conclusions

• Structures make it possible to associate related variables of possibly differing types under the same name.
• Structure members (using the dot notation) may be used anywhere an ordinary variable would be used.
• Pointers to structures make it possible to copy one entire structure to another very easily.

Exercise 15

Arrays of Structures

• Open the project’s workspace:

   On the class website
   /Examples/Lab15.zip -> Load “Lab15.mcw”

   Open MPLAB® and select Open Workspace… from the File menu. Open the file listed above.

   If you already have a project open in MPLAB, close it by selecting Close Workspace from the File menu before opening a new one.

Lab Exercise 15

Arrays of Structures

Exercise 15

Arrays of Structures

Solution: Steps 1 and 2

```c
/*############################################################################ # STEP 1: Multiply the real (re) part of each array element by 10
* HINT: Use *=
*############################################################################/
//Multiply re part of current array element by 10
x[i].re *= 10;

/*############################################################################ # STEP 2: Multiply the imaginary (im) part of each array element by 5
* HINT: Use *=
*############################################################################/
//Multiply im part of current array element by 5
x[i].im *= 5;
```
Exercise 15

Conclusions

- Arrays of structures allow groups of related structures to be referenced by a common name
- Individual structures may be referenced by the array index
- Individual structure members may be referenced by the dot notation, in conjunction with the array name and index

Questions?

1. Lab 5: Rooms extended to tonight at midnight
2. Extra lab section — today 2-4 pm
typedef struct board
{
    char data[16];
} board_t;

typedef struct button
{
    char button[16];
} button_t;

typedef struct
{
    char data[16];
    button_t left;
    button_t right;
} menu_t;