```c
head = NULL;

while (listA) {
    list = listA;
    listA = listA->next;
    listB = list;

    if (listB == NULL) {
        listB = list;
        listA = listB;
    }
    else if (listB->next == list) {
        listB = list;
        listA = listB;
    }
}
```
```c
struct Student {
    char *firstName;
    char *lastName;
    unsigned char year;
    unsigned int IDnumber;
};

struct Student class[50];

for (i = 0; i < numStudents; i++) {
    class[i].firstName = "Gabriel";
    class[i].lastName = "Elkaim";
    class[i].year = 2013;
    class[i].IDnumber = 0;
}
for (i = 0; i < numStudents; i++) {
    class[i].firstName[i] = 'C';
}
```
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
How to Create a Structure Definition

Syntax

```c
struct structName
{
    type1 memberName1;
    ...
    typen memberName_n;
}
```

Members are declared just like ordinary variables

Example

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

Structures
How to Declare a Structure Variable (Method 1)

Syntax

```c
struct structName
{
    type1 memberName1;
    ...
    typen memberName_n;
} varName1,...,varName_n;
```

Example

```c
// Structure to handle complex numbers
struct complex
{
    float re;
    float im;
} x, y;  // Declare x and y of type complex
```
Structures

How to Declare a Structure Variable (Method 2)

Syntax

If *structName* has already been defined:

```plaintext
struct structName varName1, ..., varName_n;
```

Example

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

Structures

How to Use a Structure Variable

Syntax

```c
structVariableName.memberName
```

Example

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

int main(void)
{
    x.re = 1.25;    // Initialize real part of x
    x.im = 2.50;    // Initialize imaginary part of x
    y = x;          // Set struct y equal to struct x
    ...
```
**Structures**

How to Create a Structure Type with `typedef`

**Syntax**

```c
typedef struct structTag_{optional}
{
  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;
```

---

**Structures**

How to Declare a Structure Type Variable

**Syntax**

If `typeName` has already been defined:

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

The keyword `struct` is no longer required!

**Example**

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

How to Initialize a Structure Variable at Declaration

Syntax

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

```c
typeName varName = {const_1, ..., const_n};
- or -
struct structName varName = {const_1, ..., const_n};
```

Example

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

Structures

Nesting Structures

Example

```c
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
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

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

Example: Initializing Members

```c
int main(void)
{
    str.a[0] = 'B';
    str.a[1] = 'a';
    str.a[2] = 'd';
    str.a[3] = '\0';
    str.b = "Good";
}
```

Example 1

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

Example 2

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

How to Use a Pointer to Access Structure Members

If \( \text{ptrName} \) has already been defined:

Syntax

\[
\text{ptrName} \rightarrow \text{memberName}
\]

Pointer must first be initialized to point to the address of the structure itself: \( \text{ptrName} = \&\text{structVariable}; \)

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;
}
```

Pointer must first be initialized to point to the address of the structure itself: \( \text{ptrName} = \&\text{structVariable}; \)

Structures

Creating Arrays of Structures

If \( \text{typeName} \) or \( \text{structName} \) has already been defined:

Syntax

```c
\text{typeName} \ arrName[n];
```

- or -

```c
\text{struct} \ structName \ arrName[n];
```

Example

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

Initializing Arrays of Structures at Declaration

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

**Syntax**

```
typeName arrName[n] = {{list_1}, ..., {list_n}};
- or -
struct structName arrName[n] = {{list_1}, ..., {list_n}};
```

**Example**

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

---

### Lab Exercise 14

Structures
Exercise 14
Structures

• Open the project’s workspace:

On the class website
/Examples/Lab14.zip -> Load “Lab14.mcw”

1 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.

/*############################################################################
# STEP 1: Calculate the difference between maximum and minimum power in #
#         circuit 1 using the individual power structures (i.e. variables #
#         PMax1 & PMin1). Algebraic Notation: #
#         Pdiff = (Vmax * Imax) - (Vmin * Imin) #
############################################################################*/

powerDiff1 = (PMax1.v * PMax1.i) - (PMin1.v * PMin1.i);

/*############################################################################
# STEP 2: Calculate the difference between maximum and minimum power in #
#         circuit 1 using the structure of structures (i.e. variable PRange1). #
#         Algebraic Notation: Pdiff = (Vmax * Imax) - (Vmin * Imin) #
############################################################################*/

powerDiff1 = (PRange1.max.v * PRange1.max.i) - (PRange1.min.v * PRange1.min.i);

Solution: Steps 1 and 2

Gabriel Hugh Elkaim – Spring 2013
CMPE-013/L: “C” Programming
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

Lab Exercise 15
Arrays of Structures
Exercise 15
Arrays of Structures

• Open the project’s workspace:

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

1. 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.

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?