C Structure and typedef

Computer Science Department

Eastern Washington University

Yun Tian (Tony) Ph.D.

Recall Last Lecture

- Strings
  - 1D char array with ‘\0’ at the end
  - Or char * points to a piece of memory

Today Lecture

- Structure
- typedef

Structure

- C **arrays** allow you to define type of variables that can hold several data items of the same type,
- C structure is another **user-defined data type** available in C programming,
  - it allows you to combine data items of different kinds.
- C Structure is like a **Java class without** methods(operations)

Structure

- Structures are used to represent a record,
  - Suppose you want to keep track of your books in a library.
  - You might want to track the following attributes about each book:
    - Title
    - Author
    - Subject
    - Book ID

Examples of Structure

```
struct Book   // Book here is structure tag,
{                     // ‘struct Book’ together are used to define a type.
   char  title[50];      //member of structure, static array
   char  *author; //member of structure, char pointer
   char  subject[100];
   int   book_id;
} book2; //define a variable book2 when defining the structure
struct Book myText; //define myText of type struct Book
struct Book mathText; // book2 and mathText have same type
```

Examples of Structure

```
//members can be structures
struct triangle
{
    struct point ptA;
    struct point ptB;
    struct point ptC;
};

struct point
{
    int x;
    int y;
    int z;
};
```

Examples of Structure

```
//members can be self referential
struct chain_element
{
    int data ;
    struct chain_element  *next ;
};
// what does this data structure look like? Looks familiar?
```

Access Structure Members

- To access any member of a structure variable, we use the **member access operator (.)**
- The member access operator is coded as a **period** between the structure variable name and the structure member that we wish to access.
  - Like we learned in java.

  myText.title; //value is supposed to be a char array.

Structure as function argument

- You can pass a structure variable as a function argument in very similar way as you pass any other primitive variables.
- You would access structure variables in the similar way as you have accessed in the example on previous slide.
- Demo are shown later!

Static Array of Structure

- You can define array of structures in very similar way as you define array of other types.

```
char  name[100];//can hold up to 100 characters
struct Book books[10];
```

//defines an array of structures, can hold up to 10 instances of struct Book type.

Both **name** and **books** array are static array, memory are allocated and deallocated automatically.

Pointer to Structure

- You can define pointers to structures in very similar way as you define pointer to any other primitive variable as follows.

```
char * name = (char *) malloc( 100 * sizeof(char) );
struct Book *bookPointer;
bookPointer = (struct Book *)  malloc ( 10 *
             sizeof(struct Book);
//dynamic memory allocation, have to call free() to
deallocate.
```

Pointer to Structure

- E.g.

struct Book *myText;

struct Book mathText; **//memory allocated automatically for mathText;**

//initialize myText, assign address of mathText to myText.

myText = & mathText;

Then, we can use myText to access members in mathText.

(*myText).title //dereference then member access

// retrieve the title of what myText point to

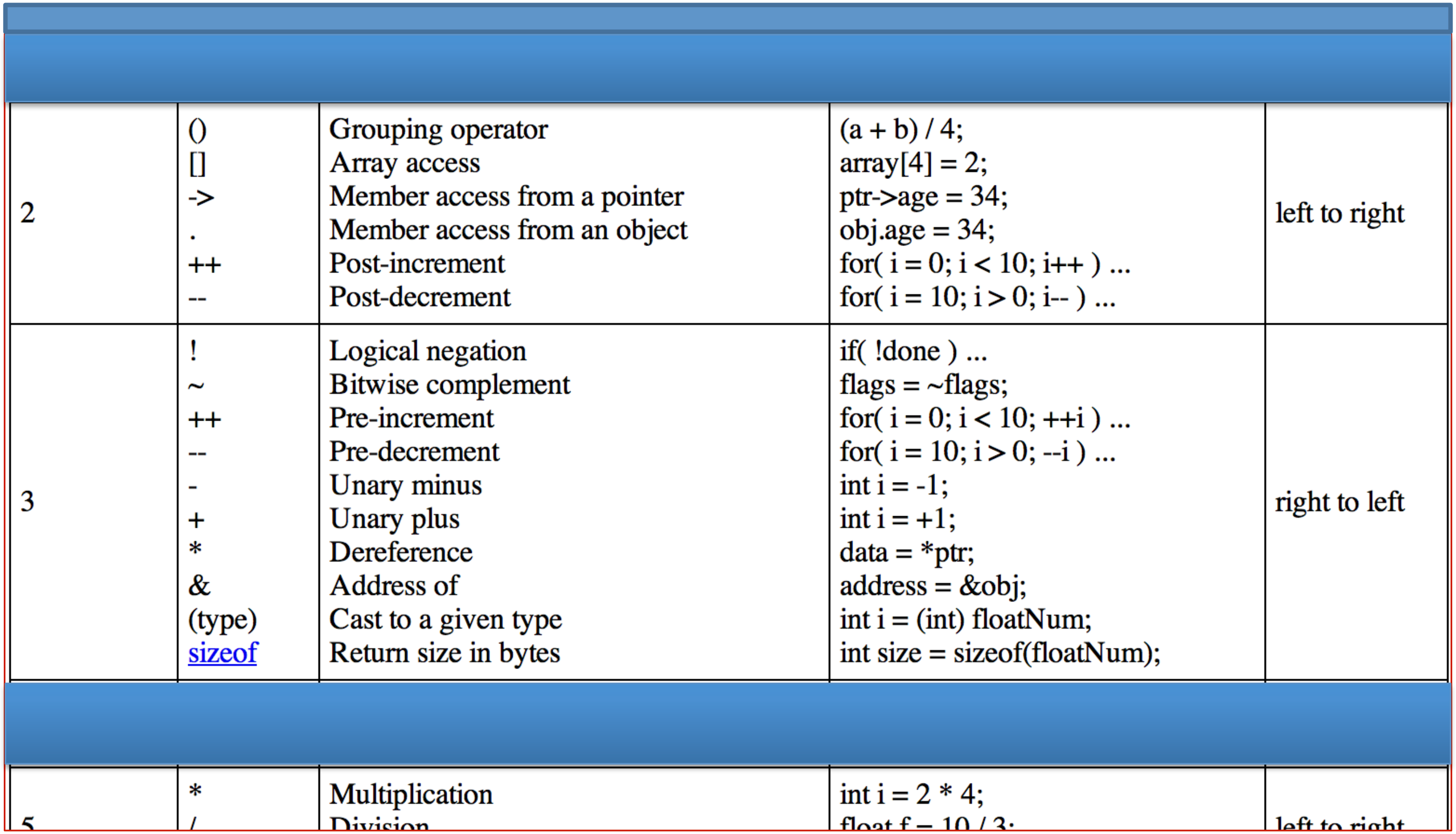
//type of the whole expression is supposed to be a **char array, according to what we have defined on slide 6.**

Pointer to Structure

(*myText).title //dereference then member access

- This format or syntax looks ugly.
- NOTE: the () around *myText is required in this context.
  - Which means (*myText).title differs from *myText.title.
  - Let us look at the C operator precedence table.
  - .(member access operator) has higher precedence than the * (dereference operator).
    - *myText.title is actually equivalent to *(myText.title)
    - What does the syntax of *(myText.title) look like ? //we have demo.

Pointer To Structures



2	()
	[]
	->
	.
	++
	--
		Grouping operator
		Array access
		Member access from a pointer
		Member access from an object
		Post-increment
		Post-decrement
			(a + b) / 4;
			array[4] = 2;
			ptr->age = 34;
			obj.age = 34;
			for(i = 0; i < 10; i++) ...
			for(i = 10; i > 0; i--) ...
				left to right

3	!
	~
	++
	--
	-
	+
	*
	&
	(type)
	sizeof
		Logical negation
		Bitwise complement
		Pre-increment
		Pre-decrement
		Unary minus
		Unary plus
		Dereference
		Address of
		Cast to a given type
		Return size in bytes
			if(!done) ...
			flags = ~flags;
			for(i = 0; i < 10; ++i) ...
			for(i = 10; i > 0; --i) ...
			int i = -1;
			int i = +1;
			data = *ptr;
			address = &obj;
			int i = (int) floatNum;
			int size = sizeof(floatNum);
				right to left

5	*
	/
		Multiplication
		Division
			int i = 2 * 4;
			float f = 10 / 3;
				left to right

Pointer to Structure

(*myText).title //dereference then member access

- This format or syntax looks ugly.
- In c, we can use -> operator for structure pointers when accessing members.
  - access the members of a structure using a pointer to that structure.

  (*myText).title is the same thing as

  myText->title

  Both of two expressions return a char array.

  -> **operator combines two operations**: deference first, then access the member to the right of the operator.

Review pointer

```
int a[] = {2, 3, 4 ,5 };
int *p = a;
*p ++;  //what is value of this expression here?
// *p ++ is equivalent to *(p++) according to precedence table
// return what p points to, then move p to next integer. the whole expression returns 2, then after this expression is executed, p points to integer 3 in the array a;
```

- Two more questions
  - (*p) ++; //what is value of this expression?
  - *(p++);   //what is value of this expression?

Review pointer

```
int a[] = {2, 3, 4 ,5 };
int *p = a;

(*p) ++;  //what is value of this expression?
```

- dereference p first,
- Meaning the value at the memory location that p points to(p holds) will be incremented by one.
- The whole expression returns 2;
- After this statement is executed, p stays the same place.
  - Pointing to the first element in array a;
  - But this array element has been changed in this statement.

Demo of Structures

Summary

- Structures
  - How to define structure type?
    - Like a java object without methods in it.
  - Access members using .
  - → operator only used with structure pointers
- **Very careful when dealing with pointer members in a structure instance.**
- **Inside a structure, we could have another structure variable or pointer.**