Mahidol University

ITCS113
Fundamentals of Programming

Lecture 14 - Advanced Struct

Instructor: Asst. Prof. Dr. Akara Supratak

Contact: akara.sup@mahidol.edu

Wisdom of the Land

Agenda

- RECAP: Struct
- Array of structure
- Structure of structure

Recap: Struct

What is Structure?

Structure: a user-defined data type in C/C++.

A structure creates a data type that can be used to group items of possibly different types into a single type.

```
struct struct_name
{
   datatype var_name1;
   datatype *var_name2;
   datatype var_name3[size];
   ...
};
```

```
struct Date
{
   int month;
   int day;
   int year;
};
```

```
struct Point
{
   int x;
   int y;
};
```

Access Members of Structure

To access any variable of a structure, we use the member access operator “.” (i.e., period).

```
struct struct_name
{
  datatype var_name1;
  datatype *var_name2;
  datatype var_name3[size];
  ...
};
```

```
struct struct_name var_name;
```

```
var_name.var_name1 = ...
var_name.var_name2 = ...
var_name.var_name3 = ...
```

```
struct Date
{
  int month;
  int day;
  int year;
};

struct Date birth_day;

birth_day.month = 12;
birth_day.day = 29;
birth_day.year = 2018;

printf("%d", birth_day.month);
printf("%d", birth_day.day);
printf("%d", birth_day.year);
```

Exercise: Design the following `struct`

Create a structure for Book including book id, title, author, and number of pages

Struct Declaration

```
struct Book {
        int book_id;
        char title[50];
        char author[50];
        int num_pages;
};
```

Struct Initialization

```
struct Book book1;
book1.id = 1;
strcpy(book1.title,"Harry Potter and
            the Sorcerer's Stone");
strcpy(book1.author,"J.K. Rowlings");
book1.num_pages = 223;
```

Exercise: `struct Book`

Create a structure for Book including book id, title, author, and number of pages, receives the book information from users and print it out.

```
struct Book{
    int book_id;
    char title[50];
    char author[50];
    int num_pages;
};
```

Exercise: struct Book

```
#include <stdio.h>
#include <string.h>



int main(){
    struct Book b1;
    scanf("%d", &b1.book_id);  // Book ID
    fgets(b1.title, 50, stdin); // Title
    char *pos;
    if ((pos=strchr(b1.title, '\n')) != NULL)
        *pos = '\0';
    fgets(b1.author, 50, stdin); // Author
    if ((pos=strchr(b1.author, '\n')) != NULL)
        *pos = '\0';
    scanf("%d", &b1.num_pages); // Number of Pages
    printf("Book(#%d)named %s authored by %s (%d pages)\n",
    b1.book_id, b1.title, b1.author, b1.num_pages);
    return 0;
}
```

```
struct Book{
    int book_id;
    char title[50];
    char author[50];
    int num_pages;
};
```

Structure Copy

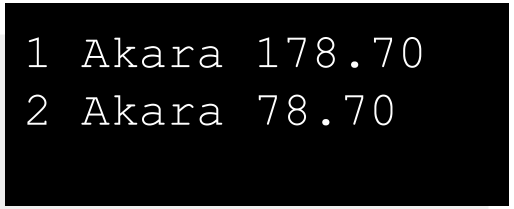
Assignment operator ‘=’ copies all member values to another structure variable.

```
#include <stdio.h>
#include <string.h>
struct Account{
   int id;
   char name[20];
   float amount;
};
int main() {
   struct Account acc1 = {1, "Akara", 178.7};
   struct Account acc2;
   acc2 = acc1; // copy values from 'acc1' to 'acc2'
   acc2.id = 2;
   acc2.amount -= 100;
   printf("%d %s %.2f\n", acc1.id, acc1.name, acc1.amount);
   printf("%d %s %.2f\n", acc2.id, acc2.name, acc2.amount);
   return 0;
}
```

Structure Copy

Assignment operator '=' copies all member values to another structure variable.

```c
#include <stdio.h>
#include <string.h>
struct Account{
   int id;
   char name[20];
   float amount;
};
int main() {
   struct Account acc1 = {1, "Akara", 178.7};
   struct Account acc2;
   acc2 = acc1; // copy values from 'acc1' to 'acc2'
   acc2.id = 2;
   acc2.amount -= 100;
   printf("%d %s %.2f\n", acc1.id, acc1.name, acc1.amount);
   printf("%d %s %.2f\n", acc2.id, acc2.name, acc2.amount);
   return 0;
}
```



```
1 Akara 178.70
2 Akara 78.70
```

Why Structure?

- Group various data in one place
- Easy to access (e.g., `acc1.amount`, `acc1.name`, etc.)
- Easy to manage (e.g., copy, pass to a function, etc.)
- Array of structures

`struct` Account

```
int id;
char name[20];
float amount;
```

Account: `acc1`

```
acc1.id = 1;
strcpy(acc1.name,"Peter");
acc1.amount = 10.50;
```

Account: `acc2`

```
acc1.id = 2;
strcpy(acc1.name, "Mary");
acc1.amount = 1112.99;
```

Why Structure?

Rewrite with struct

Each item consists of

id, name, quantity, and price:

group in one unit

(With Structure)

```
#include <stdio.h>

struct Item {
    int id;
    char name[20];
    int qty;
    float price;
};
int main() {
    struct Item items[10] = ...;
    for (int i=0 ; i<10 ; i++) {
        printf("ID: %d\n",
              items[i].id);
        printf("Name: %s\n",
              items[i].name);
        printf("Qty: %d\n",
              items[i].qty);
        printf("Price: %.2f\n",
              items[i].price);
    }
    return 0;
}
```

Array of Structures

Array of Structure

As we can use `struct` to define a new data type, we can also create an array of such new type.

```
struct struct_name
{
  datatype var_name1;
  datatype *var_name2;
  datatype var_name3[size];
  ...
};
```

```
struct struct_name var_name[size];
```

Array of Structure

For example, in a vending machine program, each item has a number of attributes associated with it (e.g., ID, name, quantity, price, etc.).

ID	Name	Quantity	Price
1	Lipton	21	20
2	Lay’s	7	30
...	...	...	...
20	Pringles	14	55

Array of Structure

For example, in a vending machine program, each item has a number of attributes associated with it (e.g., ID, name, quantity, price, etc.).

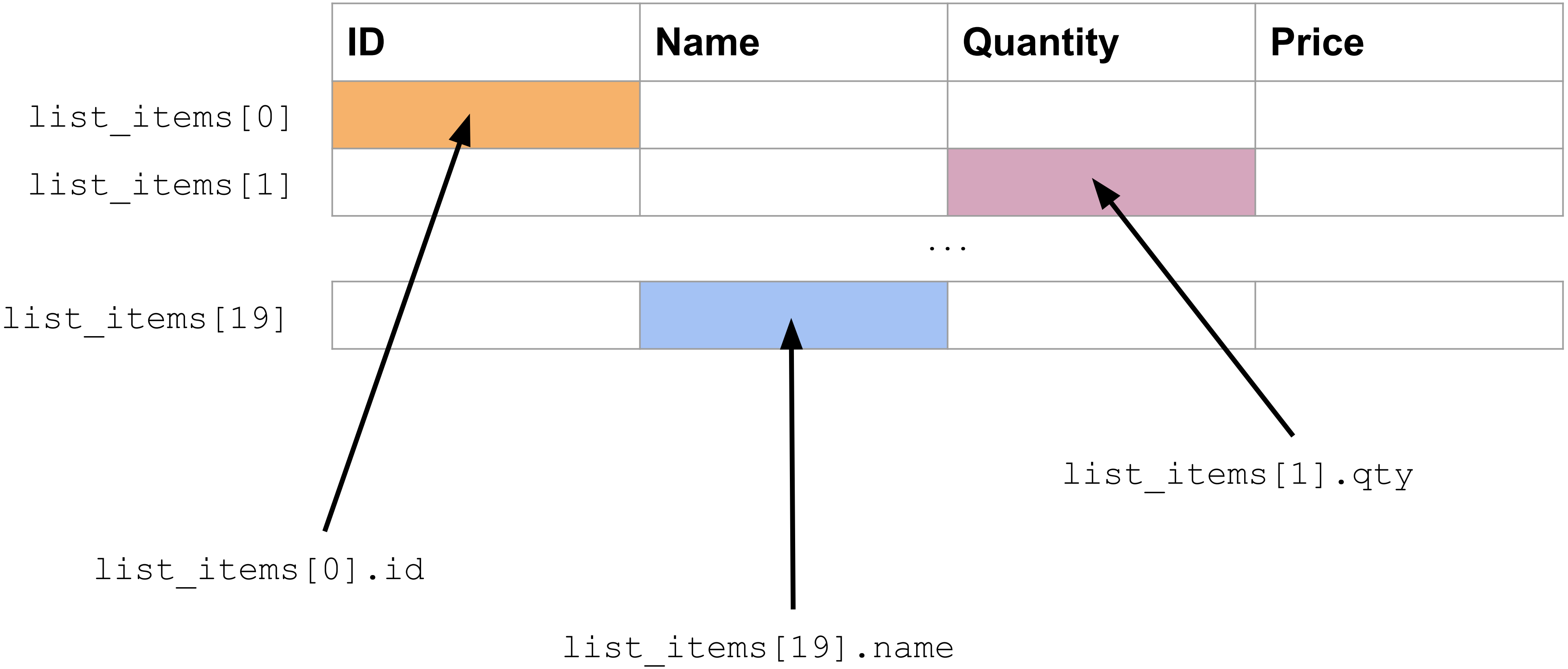
```
struct Item {
    int id;
    char name[16];
    int qty;
    float price;
};

struct Item list_items[20];
```

Array of Structure

	ID	Name	Quantity	Price
`list_items[0]`				
`list_items[1]`				
...				
`list_items[19]`				

Array of Structure



	ID	Name	Quantity	Price
`list_items[0]`				
`list_items[1]`				
		...		
`list_items[19]`				

`list_items[1].qty`

`list_items[0].id`

`list_items[19].name`

Array of Structure

	ID	Name	Quantity	Price
list_items[0]				
list_items[1]				
...				
list_items[19]				

```
list_items[0].id = 1;
strcpy(list_items[0].name, "Lipton");
list_items[0].qty = 21;
list_items[0].price = 20.0;
```

Array of Structure

	ID	Name	Quantity	Price
list_items[0]	1	Lipton	21	20
list_items[1]				
...				
list_items[19]				

```
list_items[0].id = 1;
strcpy(list_items[0].name, "Lipton");
list_items[0].qty = 21;
list_items[0].price = 20.0;
```

```
scanf("%d", &list_items[0].id);
scanf("%s", list_items[0].name);
scanf("%d", &list_items[0].qty);
scanf("%f", &list_items[0].price);
```

Initializing Array of Structure

	ID	Name	Quantity	Price
list_items[0]	1	Lipton	21	20
list_items[1]	2	Lay’s	7	30
	3	Pringles	14	55

```
struct Item {
    int id;
    char name[16];
    int qty;
    float price;
};
```

```
struct Item list_items[3] = {
    {1, "Lipton", 21, 20},
    {2, "Lay's", 7, 30},
    {3, "Pringles", 14, 55},
};
```

Example

```
#include <stdio.h>
#define N_ITEMS 3

int main() {
   struct Item list_items[N_ITEMS] = {
       {1, "Lipton", 21, 20},
       {2, "Lay's", 7, 30},
       {3, "Pringles", 14, 55},
   };

   for (int i=0 ; i<N_ITEMS ; i++) {
       printf("%d %s %d %.2f\n",
           list_items[i].id, list_items[i].name,
           list_items[i].qty, list_items[i].price
       );
   }
   return 0;
}
```

```
struct Item {
   int id;
   char name[16];
   int qty;
   float price;
};
```

Example

```c
#include <stdio.h>
#define N_ITEMS 3

int main() {
   struct Item list_items[N_ITEMS] = {
       {1, "Lipton", 21, 20},
       {2, "Lay's", 7, 30},
       {3, "Pringles", 14, 55},
   };

   for (int i=0 ; i<N_ITEMS ; i++) {
       printf("%d %s %d %.2f\n",
           list_items[i].id, list_items[i].name,
           list_items[i].qty, list_items[i].price
       );
   }
   return 0;
}
```

```c
struct Item {
   int id;
   char name[16];
   int qty;
   float price;
};
```

```
1 Lipton 21 20.00
2 Lay's 7 30.00
3 Pringles 14 55.00
```

Exercise

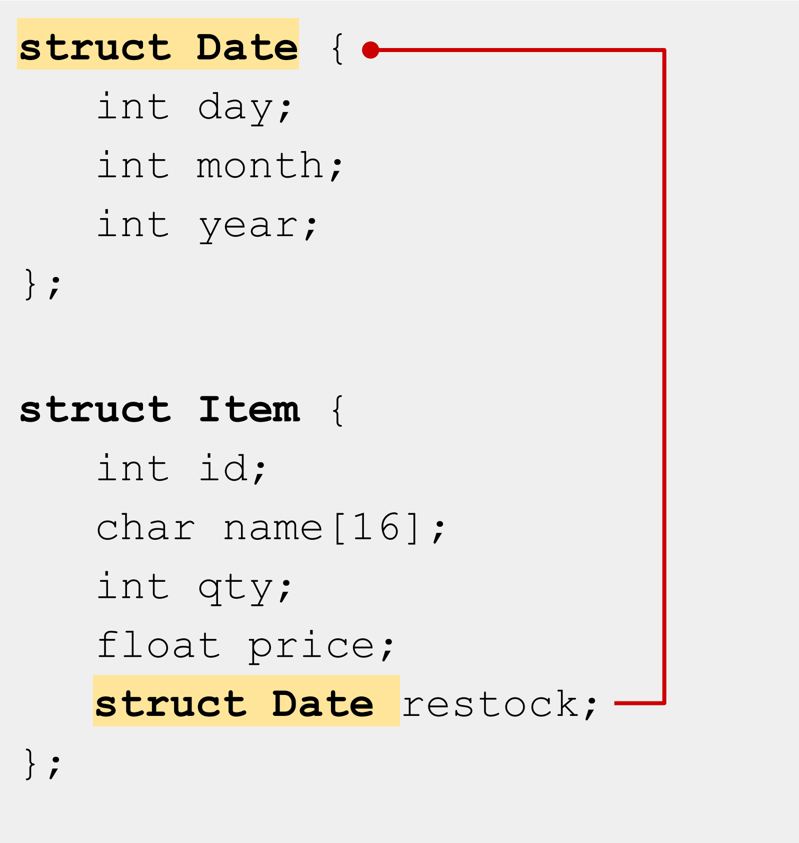
Write a program to create an array of **`HousePrice`** structure containing the following values:

Size of house (square m.)	# of bedrooms	# of bathrooms	Newly renovated	Price (10,000฿)
52.3	1	2	N	115
103.4	3	3	Y	280
99.8	2	2	Y	210

Struct in Struct

Structure in Structure

One structure can be declared inside another structure as we declare structure members inside a structure.



```
struct Date {
    int day;
    int month;
    int year;
};

struct Item {
    int id;
    char name[16];
    int qty;
    float price;
    struct Date restock;
};
```

`Date` is declared first and used as a member in **`Item`**

Structure in Structure

One structure can be declared inside another structure as we declare structure members inside a structure.

```
struct Date {
   int day;
   int month;
   int year;
};

struct Item {
   int id;
   char name[16];
   int qty;
   float price;
   struct Date restock;
};
```

```
an_item.id = 1;
strcpy(an_item.name, "Lipton");
an_item.qty = 21;
an_item.price = 20;
an_item.restock.day = 1;
an_item.restock.month = 1;
an_item.restock.year = 1999;
```

Structure in Structure

One structure can be declared inside another structure as we declare structure members inside a structure.

```
struct Date {
   int day;
   int month;
   int year;
};

struct Item {
   int id;
   char name[16];
   int qty;
   float price;
   struct Date restock;
};
```

```
struct Item list_items[3] = {
{1,"Lipton",21,20,{1,1,1999}},
{2,"Lay's",7,30,{2,2,2009}},
{3,"Pringles",14,55,{3,3,2019}},
};
```

Structure in Structure

```
#include <stdio.h>
#define N_ITEMS 3

int main() {
    struct Item list_items[N_ITEMS] = {
        {1, "Lipton", 21, 20, {1, 1, 1999}},
        {2, "Lay's", 7, 30, {2, 2, 2009}},
        {3, "Pringles", 14, 55, {3, 3, 2019}},
    };

    for (int i=0 ; i<N_ITEMS ; i++) {
        printf("%d %s %d %.2f %d-%d-%d\n",
            list_items[i].id, list_items[i].name,
            list_items[i].qty,
list_items[i].price,
            list_items[i].restock.day,
            list_items[i].restock.month,
            list_items[i].restock.year
        );
    }
    return 0;
}
```

```
struct Date {
    int day;
    int month;
    int year;
};

struct Item {
    int id;
    char name[16];
    int qty;
    float price;
    struct Date restock;
};
```

Structure in Structure

```
#include <stdio.h>
#define N_ITEMS 3

int main() {
    struct Item list_items[N_ITEMS] = {
        {1, "Lipton", 21, 20, {1, 1, 1999}},
        {2, "Lay's", 7, 30, {2, 2, 2009}},
        {3, "Pringles", 14, 55, {3, 3, 2019}},
    };

    for (int i=0 ; i<N_ITEMS ; i++) {
        printf("%d %s %d %.2f %d-%d-%d\n",
            list_items[i].id, list_items[i].name,
            list_items[i].qty,
list_items[i].price,
            list_items[i].restock.day,
            list_items[i].restock.month,
            list_items[i].restock.year
        );
    }
    return 0;
}
```

```
struct Date {
    int day;
    int month;
    int year;
};

struct Item {
    int id;
    char name[16];
    int qty;
    float price;
    struct Date restock;
};
```

```
1 Lipton 21 20.00 1-1-1999
2 Lay's 7 30.00 2-2-2009
3 Pringles 14 55.00 3-3-2019
```

Exercise

Write a program to create a struct Person with the struct Address as one of the member as shown in the table.

Name	Age	Address		
		House Number	District	Zip Code
Zhongli	20	199	Liyue	11002
Barbara	18	300	Mondstadt	15213
Klee	17	773	Mondstadt	15213

typedef

typedef can be used to **give a type a new name**.

We can then use typedef to shorten the code we use to create a structure variable.

```
struct struct_name
{
  datatype var_name1;
  datatype *var_name2;
  datatype var_name3[size];
  ...
};
```

```
typedef struct struct_name short_name;
```

```
short_name var_name1, var_name2;
// struct struct_name var_name1, var_name2;
```

typedef

typedef can be used to **give a type a new name**.

We can then use typedef to shorten the code we use to create a structure variable.

```
struct struct_name
{
  datatype var_name1;
  datatype *var_name2;
  datatype var_name3[size];
  ...
};
```

```
typedef struct struct_name short_name;
```

```
short_name var_name1, var_name2;
// struct struct_name var_name1, var_name2;
```

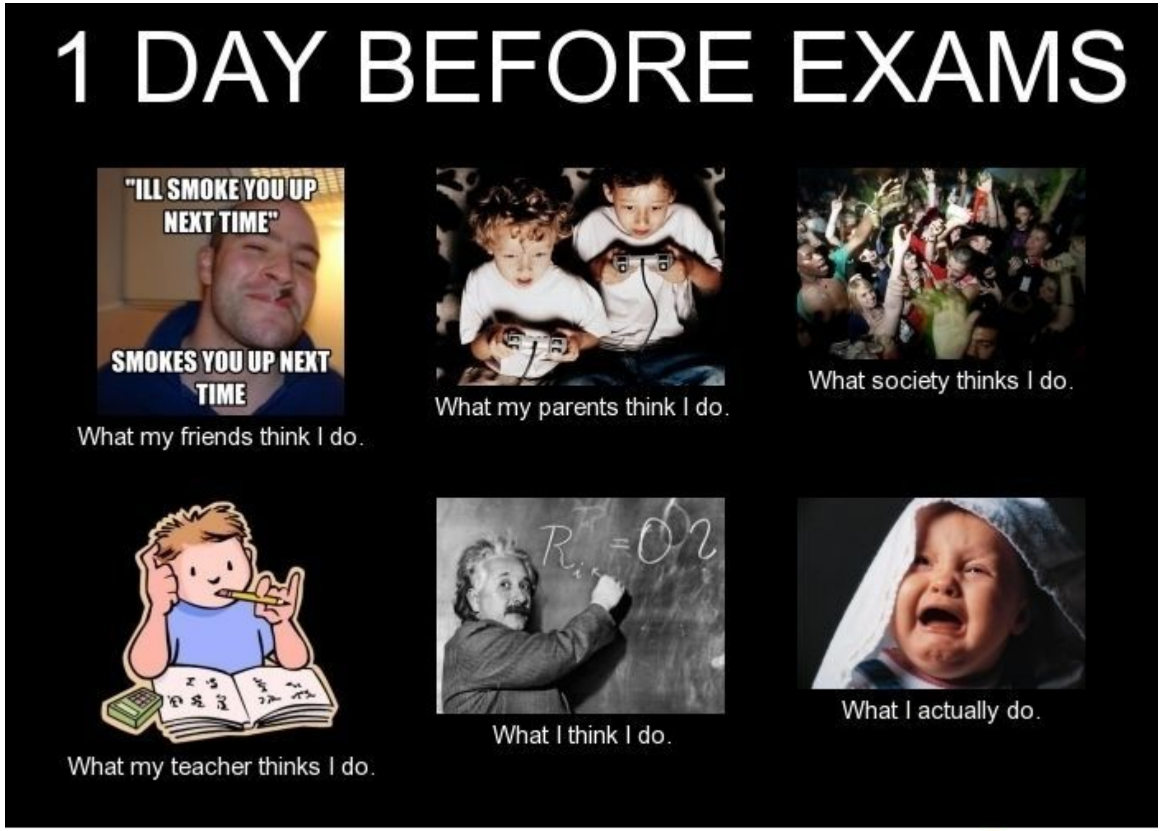
```
struct Date {
    int month;
    int day;
    int year;
};

typedef struct Date DATE;

DATE d1, d2;
```

Mock Final Exam (Quiz 6)

- 3 questions (1.5 hour)
- Closed-book
- Allowed materials
  - A PDF cheat sheet (MyCourse)
  - An empty physical A4 paper


1 DAY BEFORE EXAMS
"ILL SMOKE YOU UP NEXT TIME"
SMOKES YOU UP NEXT TIME
What my friends think I do.
What my parents think I do.
What society thinks I do.
What my teacher thinks I do.
What I think I do.
What I actually do.

MEMES & FUNNY PICS FRABZ.COM

Lab Exercises