CS2100 COMPUTER ORGANISATION

http://www.comp.nus.edu.sg/~cs2100/

Lecture #5d

Arrays, Strings and Structures

Questions?

IMPORTANT: DO NOT SCAN THE QR CODE IN THE VIDEO RECORDINGS. THEY NO LONGER WORK

Ask at
https://sets.netlify.app/module/676ca3a07d7f5ffc1741dc65

OR

Scan and ask your questions here!
(May be obscured in some slides)

4.9 Passing Structure to Function (1/2)

- Passing a structure to a parameter in a function is akin to assigning the structure to the parameter.
- The entire structure is **copied**, i.e., members of the actual parameter are copied into the corresponding members of the formal parameter.
  - *Pass-by-value*
- We use PassStructureToFn.c to illustrate this.

```
player1: name = Brusco; age = 23; gender = M
player2: name = July; age = 21; gender = F
```

4.9 Passing Structure to Function (2/2)

PassStructureToFn.c

```
// #include statements and definition
// of player_t are omitted here for brevity
void print_player(char [], player_t);

int main(void) {
    player_t player1 = { "Brusco", 23, 'M' }, player2;

    strcpy(player2.name, "July");
    player2.age = 21;
    player2.gender = 'F';

    print_player("player1", player1);
    print_player("player2", player2);

    return 0;
}

// Print player's information
void print_player(char header[], player_t player) {
    printf("%s: name = %s; age = %d; gender = %c\n", header,
           player.name, player.age, player.gender);
}
```

Passing a structure to a function

Receiving a structure from the caller

(For own reading)

4.10 Array of Structures

- Combining structures and arrays gives us a lot of flexibility in organizing data.
  - For example, we may have a structure comprising 2 members: student's name and an array of 5 test scores he obtained.
  - Or, we may have an array whose elements are structures.
  - Or, even more complex combinations such as an array whose elements are structures which comprises array as one of the members.
- Case study (Program: NearbyStores.c)
  - A startup company decides to provide location-based services. Its customers are a list of stores.
  - Each store has a name, a location given by (x, y) coordinates, a radius that defines a circle of influence.
  - We can define a structure type store_t for the stores, and have a store_t array store_t variables. We call this array storeList and it represents the list of stores.

4.11 Passing Address of Structure to Function (1/5)

- Given this code, what is the output?

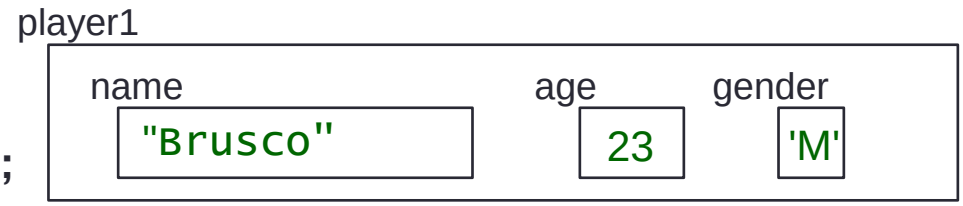
PassStructureToFn2.c

```
// #include statements, definition of player_t,
// and function prototypes are omitted here for brevity
int main(void) {
   player_t player1 = { "Brusco", 23, 'M' };

   change_name_and_age(player1);
   print_player("player1", player1);
   return 0;
}

// To change a player's name and age
void change_name_and_age(player_t player) {
   strcpy(player.name, "Alexandra");
   player.age = 25;
}

// Print player's information
void print_player(char header[], player_t player) {
   printf("%s: name = %s; age = %d; gender = %c\n", header,
          player.name, player.age, player.gender);
}
```

```
player1: name = Brusco; age = 23; gender = M
```

4.11 Passing Address of Structure to Function (2/5)

```
main()

change_name_and_age(player1);
```




```
change_name_and_age(player_t player)
```


```
strcpy(player.name, "Alexandra");
player.age = 25;
```

4.11 Passing Address of Structure to Function (3/5)

- Like an ordinary variable (eg: of type int, char), when a structure variable is passed to a function, a separate copy of it is made in the called function.
- Hence, the original structure variable will not be modified by the function.
- To allow the function to modify the content of the original structure variable, you need to pass in the address (pointer) of the structure variable to the function.
- (Note that passing an array of structures to a function is a different matter. As the array name is a pointer, the function is able to modify the array elements.)

4.11 Passing Address of Structure to Function (4/5)

- Need to pass address of the structure variable

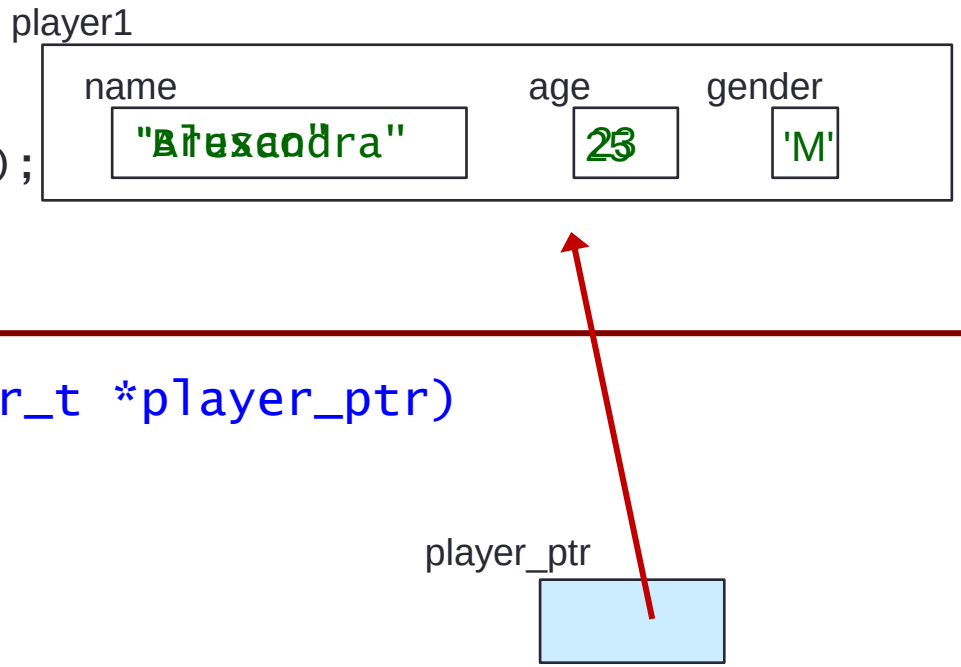
PassAddrStructToFn.c

```
// #include statements, definition of player_t,
// and function prototypes are omitted here for brevity
int main(void) {
   player_t player1 = { "Brusco", 23, 'M' };

   change_name_and_age(&player1);
   print_player("player1", player1);
   return 0;
}

// To change a player's name and age
void change_name_and_age(player_t *player_ptr) {
   strcpy((*player_ptr).name, "Alexandra");
   (*player_ptr).age = 25;
}

// Print player's information
void print_player(char header[], player_t player) {
   printf("%s: name = %s; age = %d; gender = %c\n", header,
          player.name, player.age, player.gender);
}
```

```
player1: name = Alexandra; age = 25; gender = M
```

4.11 Passing Address of Structure to Function (5/5)

```
main()

change_name_and_age(&player1);
```




```
change_name_and_age(player_t *player_ptr)

strcpy((*player_ptr).name, "Alexandra");
(*player_ptr).age = 25;
```

4.12 The Arrow Operator (->) (1/2)

- Expressions like `(*player_ptr).name` appear very often. Hence an alternative "shortcut" syntax is created for it.
- The arrow operator (**->**)

`(*player_ptr).name`	*is equivalent to*	`player_ptr->name`
`(*player_ptr).age`	*is equivalent to*	`player_ptr->age`

- Can we write *player_ptr.name instead of (*player_ptr).name?
- *No*, because **.** (dot) has higher precedence than *****, so *player_ptr.name means *(player_ptr.name)!

4.12 The Arrow Operator (->) (2/2)

- Function change_name_and_age() in PassAddrStructToFn2.c modified to use the **->** operator.

PassAddrStructToFn2.c

```
// To change a player's name and age
void change_name_and_age(player_t *player_ptr) {
   strcpy(player_ptr->name, "Alexandra");
   player_ptr->age = 25;
}
```

Quiz

- Please Arrays, Strings and Structures Quiz 1 before 3 pm on 23 August 2022.

End of File