

Structures I

CS2263 – Systems Software Development

Little Bones, The Tragically Hip (Road Apples, 1991) <https://www.youtube.com/watch?v=-3bwDb2piZs>

1

Learning Outcomes

At the conclusion of this lecture students should be able to:

- Describe what a structure is
- Define and use a simple structure in C
- Explain how a structure is “laid out” in memory
 - On the stack
 - On the heap
- Explain the purpose of a type definition and use it.

2

References

- Lu, Yung-Hsiang. 2015. Intermediate C Programming. CRC Press. New York. (Chapter 16)
- Kernighan, Brian, and Dennis Ritchie. 1988. The C Programming Language, Second Edition. Prentice Hall. Upper Saddle River, NJ. (Chapter 6)

3

What do we Already Know?

- From Java:
 - What is a class?
 - What is an object?
- From C:
 - What is a module?
 - `String.h/String.o`
- What is an array?
 - Homogenous data type collection
 - Identifier/name is an address
 - Can be indexed

4

What If I Told You...

structv1.c

```
struct person {
    char firstName[20];
    char lastName[20];
    int age;
};

struct person in1, in2;
person totallyWrong;
```

- What can you guess about this code?
- Array: a homogeneous collection of data
- Structure: a heterogeneous collection of data
- How do we declare a struct?

5

Structures on the Stack

structv1.c

```
13 struct person p1 = {"Bob", "Smith", 59};
17 printf("struct person memory:\n");
18 printf("thing1: %p\n", &(p1) );
19 printf("thing1.firstName: %p\n", p1.firstName);
20 printf("thing1.lastName: %p\n", p1.lastName);
21 printf("thing1.age: %p\n", &(p1.age) );

32 * struct person memory:
33 * thing1: 0x7ffee979f988
34 * thing1.firstName: 0x7ffee979f988
35 * thing1.lastName: 0x7ffee979f99c
36 * thing1.age: 0x7ffee979f9b0
```

- We can initialize structs like arrays
- Structs lay out on the stack just like arrays
- We access structure members just like in Java

6

Structure Errors

```
struct example { ... }; //declaration  
example e; // still wrong  
struct example e;  
struct example { ... }; // no, already defined
```

7

Structure Operations

- The structures only hold values
- Legal operations are
 - Copying
 - Assigning
 - Member access

8

Structures Are Awkward

- Declaring instances
- Using as function parameters/arguments
- Using as function return values

9

typedef

- What does it do?
- How could it help? (you know it could!)
- A simple technique that reduces cognitive load.

```
typedef char* String;  
String sName;
```

10

typedef II

structV2.c

- What does it do?
- How could it help? (you know it could!)
- A simple technique that reduces cognitive load.

```
typedef struct person
{
    char firstName[20];
    char lastName[20];
    int age;
} Person;

Person p1;
```

11

Recall Structure Errors

```
struct example { ... };
example e; // still wrong
struct example e;
struct example { ... }; // no, already defined
```

```
typedef struct example { ... } Example;
Example e; // correct!
```

12

typedefs, structs and the Stack

- Using `structV2.c`, model the stack
 - Before the call to `printfPerson()`
 - Once inside `printfPerson()`
- Unlike arrays, structs pass all structure values

```
25     printfPerson(p1);
26     printf("\n");
27
28     return EXIT_SUCCESS;
29 }
```

```
41  * In main
42  * p1: 0x7ffeea72b988
43  * p1.firstName: 0x7ffeea72b988
44  * p1.lastName: 0x7ffeea72b99c
45  * p1.age: 0x7ffeea72b9b0
46  * Inside printfPerson
47  * thing1: 0x7ffeea72b920
48  * thing1.firstName: 0x7ffeea72b920
49  * thing1.lastName: 0x7ffeea72b934
50  * thing1.age: 0x7ffeea72b948
```

13

Nested Structures

`structV3.c/
structMemory.c`

- How should this layout in memory?

```
strcpy(p1.info.firstName,"Bob");
strcpy(p1.info.lastName,"Smith");
p1.info.age = 59;
```

```
62  /*
63  * EmployeeT memory:
64  * p1: 0x7ffec890980
65  * p1.info: 0x7ffec890980
66  * p1.info.firstName: 0x7ffec890980
67  * p1.info.lastName: 0x7ffec890994
68  * p1.info.age: 0x7ffec8909a8
69  * p1.salary: 0x7ffec8909b0
70  * Inside printfEmployeeT:
71  * emp: 0x7ffec890900
72  * emp.info: 0x7ffec890900
73  * emp.salary: 0x7ffec890930
74  * Inside printfPerson:
75  * thing1: 0x7ffec8908a0
76  * thing1.firstName: 0x7ffec8908a0
77  * thing1.lastName: 0x7ffec8908b4
78  * thing1.age: 0x7ffec8908c8
79  */
```

14

Exercise

- Create a type, `Point2D`, that represents a two-dimensional cartesian coordinate with double precision.
- Write code to declare one and assign values to it.
- Now write a function to “pretty-print” the coordinate in the form (x-value, y-value)