(10-1) Structs
H&K Chapter 10

Instructor – Beiyu Lin

CptS 121 (June 3rd, 2019)

Washington State University

Struct: user-defined type

- Data Structure is a way in which data is stored on a computer.

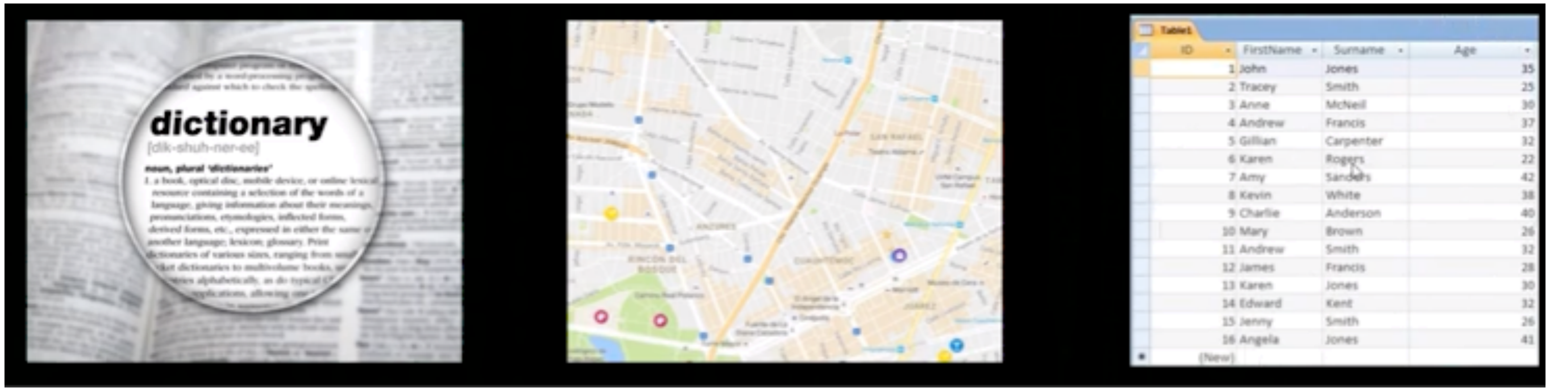

Table1

ID	FirstName	Surname	Age
1	John	Jones	35
2	Tracey	Smith	25
3	Anne	McNeil	30
4	Andrew	Francis	37
5	Gillian	Carpenter	32
6	Karen	Rogers	22
7	Amy	Sanders	42
8	Kevin	White	38
9	Charlie	Anderson	40
10	Mary	Brown	26
11	Andrew	Smith	32
12	James	Francis	28
13	Karen	Jones	30
14	Edward	Kent	32
15	Jenny	Smith	26
16	Angela	Jones	41
(New)			

Graphs are from: https://www.youtube.com/watch?v=NzUVn8Y9deQ

Structs (1)

- Let's first define a struct student

```
typedef struct
{
   details ...
}Type;
```

```
typedef struct
{
   int ID;
   char grade;
   int present;
} Student;
```

Struct: user-defined type

- C supports another kind of user-defined type: the `struct`
- `struct`s are a way to combine multiple variables into a single "package" (this is called "encapsulation")
- Sometimes referred to as an *aggregate*, where all variables are under one name
- Suppose, for example, that we want to create a database of students in a course. We could define a student `struct` as follows:

`struct` Type (2)

```
typedef enum {freshman, sophomore, junior, senior} class_t;

typedef enum {anthropology, biology, chemistry,
                english, compsci, polisci,psychology,
              physics, engineering, sociology} major_t;


typedef struct
{
     int id_number;
     class_t class_standing; /* see above */
     major_t major; /* see above */
     double gpa;
    int credits_taken;
} student_t;
```

`struct` Type (3)

- We can then define some students:

```
student_t student1, student2;
student1.id_num = 123456789;
student1.class_standing = freshman;
student1.major = anthropology;
student1.gpa = 3.5;
student1.credits_taken = 15;
student2.id_num = 321123456;
student2.class_standing = senior;
student2.major = biology;
studnet2.gpa = 3.2;
student2.credits_taken = 100;
```

↔

```
int credit1, credit2;
```

Notice how we use the "." (selection) operator to access the "fields" of the `struct`

`struct` Type (4)

- We can easily make a copy of a whole structure simply by using the assignment operator:

```
/* each field is copied to the corresponding field
   in student3 */
student_t student3 = student1;
```

struct Type (5)

- We can also return a struct as a function result:

```
student_t read_student()
{
    student_t student
    int temp_class, temp_major;
    printf("Please enter ID number of student: ");
    scanf("%d",&student.id_num);
    printf("Please enter class standing (0 = fr,\n");
    printf("1 = so, 2 = ju, 3 = se): ");
    scanf("%d",&temp_class);
    student.class = (class_t)temp_class;
    printf("Please enter major (0 = anthro.,\n");
    printf("1 = biol., 2 = chem., … , 8 = soc.: ");
    scanf("%d",&temp_major);
    student.major = (major_t)temp_major;
    printf("Please enter gpa: ");
    scanf("%lf",&student.gpa);
    printf("Please enter credits taken: ");
    scanf("%d",&student.credits_taken);
    return student;
}
```

```
int read_student()
{
    int int_var
    details….

    return int_var;
}
```

struct Type (6)

- Here's how we could use the previous function:

```
int main(void)
{
   student_t student1, student2;
   student1 = read_student();
   student2 = read_student();
   print_student(student1); /* assume print_student is defined */
   print_student(student2);
   return(1);
}
```

struct Type (7)

- We can rewrite the previous function so that it fills in an output parameter:

```
void read_student(student_t *student)
{
   int temp_class, temp_major;
   printf("Please enter ID number of student: ");
   scanf("%d",&(*student).id_num);
   printf("Please enter class standing (0 = fr,\n");
   printf("1 = so, 2 = ju, 3 = se): ");
   scanf("%d",&temp_class);
   (*student).class = (class_t)temp_class;
   printf("Please enter major (0 = anthro.,\n");
   printf("1 = biol., 2 = chem., … , 8 = soc.: ");
   scanf("%d",&temp_major);
   (*student).major = (major_t)temp_major;
   printf("Please enter gpa: ");
   scanf("%lf",&(*student).gpa);
   printf("Please enter credits taken: ");
   scanf("%d",&(*student).credits_taken);
}
```

`struct` Type (8)

- Here's how we could use the previous function:

```
int main(void)
{
   student_t student1, student2;
   read_student(&student1);
   read_student(&student2);
   print_student(student1); /* assume print_student is defined */
   print_student(student2);
   return(1);
}
```

Similar as read in data from file

struct Type (9)

- C provides the -> (component selection) operator as a means of accessing struct fields. This provides a nice alternative to the * operator:

```
void read_student(student_t *student)
{
    int temp_class, temp_major;
    printf("Please enter ID number of student: ");
    scanf("%d",&(student->id_num));
    printf("Please enter class standing (0 = fr,\n");
    printf("1 = so, 2 = ju, 3 = se): ");
    scanf("%d",&temp_class);
    student->class = (class_t)temp_class;
    printf("Please enter major (0 = anthro.,\n");
    printf("1 = biol., 2 = chem., … , 8 = soc.: ");
    scanf("%d",&temp_major);
    student->.major = (major_t)temp_major;
    printf("Please enter gpa: ");
    scanf("%lf",&(student->gpa));
    printf("Please enter credits taken: ");
    scanf("%d",&(student->credits_taken));
}
```

`struct` Type (10)

- Notes
  - struct types are most often used in applications that work with databases
    - student records
    - employee records
    - planet records
  - Often, we define databases as *arrays* of structs
  - For now, just understand that a `struct` is a way to encapsulate multiple variables in a single "package"

References

- J.R. Hanly & E.B. Koffman, *Problem Solving and Program Design in C (8th Ed.)*, Addison-Wesley, 2016
- P.J. Deitel & H.M. Deitel, *C How to Program (7th Ed.)*, Pearson Education , Inc., 2013.

Collaborators

- Chris Hundhausen
- Andrew O’Fallon