Lesson 14
Structures

Trịnh Thành Trung

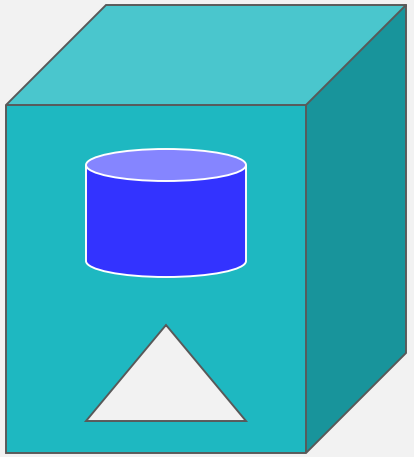
trungtt@soict.hust.edu.vn

Topic of this week

- Structure
  - Class Lecture Review
    + Structure declaration
    + Using typedef
    + Accessing field of structure variables.
  - Programming Exercises

Structure

- A structure in C is a collection of items of different types.
- Structures, or structs, are very useful in creating data structures larger and more complex than the ones we have discussed so far.



Defining a `struct` in C

```
struct struct-name
{

    field-type1 field-name1;
    field-type2 field-name2;
    field-type3 field-name3;

    ...
};
```

Example

- We can define a type for representing a student who has a name, age and a grade like this

```
struct student {
    char name[20];
    int age;
    float grade;
};
```

Example

- To create a new user defined type for cars:
  - A car must have a brandname, a model (string) and a year of production (a number)

```
struct car {
    char* make;
    char* model;
    int   year;
};
```

Variable declaration and Initialisation

- You must use keyword struct in the declaration

```
struct student s1;

struct car mycar;


struct student s1 = {“Nguyen Le”, 19, 8.0};

struct car mycar = {“Fiat”, "Punto", 2004};
```

Structure declaration with typedef

```
typedef struct student {
        char name[20];
        int age;
        float grade;
} student_t;

typedef struct car {
            char* make;
             char* model;
            int   year;
} car_t;
```

Now the program has a new types - **student_t and car_t**

Variable declaration

- With the usage of typedef, we don't have to write "**`struct student`**" every time!
- For example, we can define two complex numbers in the following line:

```
car_t mycar;

student_t excellentP;
```

Accessing Members of a Structure

- Use a dot between the structure name and the field name .

```
car_t mycar;
mycar.year = 2004;

student_t excellentp;
excellentp.age = 18;
excellentp.grade = 7.8;
```

Exercises 14.1

1. Create a structure named Date for storing date concerning variables. Each date has a day, a month and a year.
2. Write a function for the input of variable of this type. Remember to check the validation of data
3. Write a function to datecmp to compare two date which return
   - -1 if the first date (parameter) is before the second
   - 0 if two date are identical.
   - 1 if the first date (parameter) is after the second
4. Write a program asking user to for two date and print out the results of the comparison.

For example: 2/10/1997 is after 23/8/1997

Clear Buffer when reading structure's data

- Windows env:
```
fflush(stdin);
```
- Every env:
```
void clear_buffer() {
    int ch;
    while ((ch=getchar()) !='\n' && ch!=EOF);

}
```

Exercise 14.2

- Write a program that uses a structure to store the following weather data for a particular month:
  - Total Rainfall
  - High Temperature
  - Low Temperature
  - Average Temperature
- The program should have an array of 12 structures to hold weather data for an entire year. When the program run, it ask the user to enter data for each month and then calculate and display the average rain fall, the total rainfall of the year, the highest and lowest temperatures for the year.
- Input validation: Only accept temperature within the range -40 and 50 degrees Celcius.

Exercise 14.3

- Write a student management program using this structure:

```
typedef struct
{
   char id[6];
   char name[31];
   float grade;
   char level;
} student;
```

- Students are classified according to their grade in respect to this criteria :
  - from 9 to 10: A (Excellent)
  - from 8 to 9: B (Good)
  - from 6.5 to 8: C (Medium)
  - < 6.5 : D (Bad)

Exercise 14.3

- The program should read from keyboard data for n students, then print the list of student in order descending of grade like this:

```
Name            Grade       Level
Dao Tiem        9.3           A
Dinh Lan        8.2           B
Bui Luu Van     5.7           D
```

Exercise 14.4

- Define a new type for representing the fractions. Then use it to write a fraction manipulation program. This program have the following functionalities.
  - Input for an array of fraction
  - Print the content of the fraction array
  - Inverse all the fraction in the array
  - Compare two fraction

Exercise 14.5

- Develop others following userful fraction manipulation functions:
  - convert a fraction to simplified fraction simple
  - convert array to array of simplified fraction
  - mutiply, add two fractions
- And integrate them into the program in exercise 14.4