C Programming Introduction

week 14:Structure

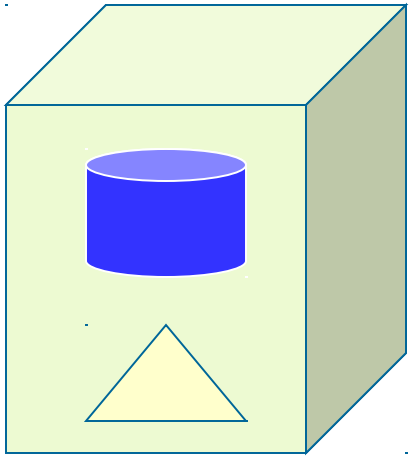
Topic of this week

- 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

a) Create a structure named Date for storing date concerning variables. Each date has a day, a month and a year.

b) Write a function for the input of variable of this type. Remember to check the validation of data

c) 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

d) 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

Solution: Structure Declaration

```
#include <stdio.h>
typedef struct date
{
    unsigned char day;
    unsigned char month;
    int nam;
}date_t;
```

Solution: Input function

```
date input_date(){
    date tmp;
    do {
        printf("The day (between 1 and 31):");
        scanf("%u",&tmp.day);
    } while ((tmp.day <1) || (tmp.day >31))
    do {
        printf("The month (between 1 and 12):");
        scanf("%u",&tmp.month);
    } while ((tmp.day <1) || (tmp.day >12))
    do {
        printf("The year (between 1 and 10000):");
        scanf("%d",&tmp.year);
    } while ((tmp.day <1) || (tmp.day >1000))

    return tmp;
}
```

Solution: Date compare function

```
int datecmp(date d1, date d2){
  if (d1.year < d2.year) return -1;
  else if (d1.year > d2.year) return 1;
  else {
  if (d1.month < d2.month) return -1;
  else if (d1.month > d2.month) return 1;
  else {
  if (d1.day < d2.day) return -1;
  else if (d1.day > d2.day) return 1;
  else return 0;
  }
  }
}
```

Solution: main program

```
int main(){
   date date1, date2;
   int m;
   printf("Enter the first date.\n");
   date1= input_date();
   printf("Enter the second date.\n");
   date2= input_date();
   m = datecmp(date1, date2);
   if (m==0) printf("Two date are identical.\n");
   else if (m<0) printf("%d/%d/%d is before %d/%d/%d\n",
   date1.day, date1.month, date1.year, date2.day, date2.month,
   date2.year);
   else printf("%d/%d/%d is after %d/%d/%d\n", date1.day,
   date1.month, date1.year, date2.day, date2.month, date2.year);
   return 0;
}
```

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.

solution

```
#include <stdio.h>
typedef struct wt{
   int total_rain; // in mm
   int high_temp; // in celcius
   int low_temp;
}wearther;
const int MONTHS = 12;
```

solution

```
int main() {
   weather data[MONTHS];
   int i;
   char months[MONTHS][10]={"January", "February",
   "March", "April"};
   int total_rain = 0;
   int max_temp = -100;
   int max_month = 0;
   int min_temp = 100;
   int min_month = 0;
// read input data for each month
   for (i = 0; i < MONTHS; i++) {
   printf("Weather Data for %d: \n", months[i]);
   printf("  Total Rainfall (mm): ");
   scanf("%d",&data[i].total_rain);
   printf("  High Temperature (C): ");
   scanf("%d",&data[i].high_temp);
   printf("  Low Temperature (C): ");
   scanf("%d",&data[i].low_temp);
   }
```

solution

```
    // find min, max temperature
    for (i = 0; i < MONTHS; i++) {
    total_rain += data[i].total_rain;
    if (max_temp < data[i].high_temp) {
    max_temp = data[i].high_temp;
    max_month = i;
    }
    if (min_temp > data[i].low_temp) {
    min_temp = data[i].low_temp;
    min_month = i;
    }
    }
    printf ("\nWeather Statistic for the Year:\n";
    printf ("  Total Rainfall: %d mm\n", total_rain);
    printf (" Monthly Rainfall: %2.3f mm\n",
    double(total_rain)/MONTHS );
    printf (" Highest Temperature: " : %d °C in %s \n",
    max_temp, months[max_month]);
    printf (" Lowest Temperature: " : %d °C in %s \n",
    min_temp, months[min_month]);
    return 0;
}
```

Exercise 14.3

- Write a student management program using this structure:

```
typedef struct
{
 char id[6];
 char name[31];
 float grade;
 char classement
} 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	Classment
Dao Tiem	9.3	A
Dinh Lan	8.2	B
Bui Luu Van	5.7	D

Solution:

```
#include <stdio.h>
#include <string.h>
#define MAX 100
typedef struct {
 char id[6];
 char name[31];
 float grade;
 char classement
} student;

void printStudent(student s)
{
printf("ID | Name    | Grade | Classment\n");
printf("%s | %s      | %1.1f | %c\n",s.id,s.name,
  s.grade,s.classement);
}
```

Solution:

```
int main(){
   int i, n;
   student std_list[MAX], tmp;
   printf("Enter the number of student (>0):");
   scanf("%d", &n);
   for(i=0; i<n; i++){
   printf("ID:"); gets(std[i].id);
   printf("name:"); gets(std[i].name);
   printf("Grade:"); scanf("%f",&std[i].grade);
   if (std[i].grade >= 9 && std[i].grade <= 10)
   std[i].classment = 'A';
   else if (std[i].grade >= 8 && std[i].grade < 9)
   std[i].classment = 'B';
   else if (std[i].grade >= 6.5 && std[i].grade < 8)
   std[i].classment = 'C';
   else std[i].classment = 'D';
   }
```

Solution: Sort the student list

```
    for(i=0; i<n-1; i++)
      for (j=i+1; j<n; j++)
           if (std[i].grade < std[j].grade){
                tmp=std[i];
                std[i]=std[j];
                std[j]=tmp;
           }
    for(i=0; i<n; i++) printStudent(std[i]);
    return 0;
}
```

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

Solution:

```
#include <stdio.h>
#define MAX 100
typedef struct {
   int numerator;
   int Denominator;
}fraction;

void fractionInput(fraction *ps);
void fractionOutput (fraction ps);
void fractionArrayInput(fraction dsps[], int n);
void fractionArrayOutput(fraction dsps[], int n);
int  fractionCmp(fraction ps1, fraction ps2);
fraction inverse(fraction ps);
void inverseArray(fraction dsps[], int n);
```

Solution:

```
void fractionInput(fraction *ps){
   int n,d;
   printf("Numerator:"); scanf("%d",&n);
   ps->numerator = n;
   do {
        printf("Denominator:"); scanf("%d",&d);
   } while (d==0);
   ps->denominator = d;
}
void fractionArrayInput(fraction dsps[], int n){
   int i;
   for (i=0; i<n; i++){
        printf("Data input for the i-th fraction:\n");
        fractionInput(dsps[i]);
   }
}
```

Solution:

```
void fractionOutput(fraction ps){
  printf(" %d/%d ", ps.numerator,
  ps.denominator);
}
void fractionArrayOutput(fraction dsps[], int n){
  int i;
  printf("Data output for the array of
  fraction:\n");
  for (i=0; i<n; i++){
  fractionOutput(dsps[i]);
  }
  printf("\n");
}
```

Solution:

```
fraction inverse(fraction ps){
   fraction tmp;
   if (ps.numerator ==0) {
   printf("Can not have an inverse fraction of 0!\n");
   exit(1);
   }
   tmp.numerator = ps.denominator; tmp.denominator=ps.numerator;
   return tmp;
}
int fractionCmp(fraction ps1, fraction ps2){
   long smd =(ps1.numerator*ps2.denominator -
   ps2.numerator*ps1.denominator )*ps1.denominator*ps2.denominator;
   if(smd> 0)return 1;
   else if (smd<0) return -1;
   else return 0;
}
```

Solution:

```
void inverseArray(fraction dsps[], int n){
   int i;
   for(i=0; i<n; i++) {
         dsps[i]= inverse(dsps[i]);
   }
}
int main(){
   int n;
    fraction a[MAX], max, s, p;
    printf("Enter the number of fractions:");
   scanf("%d",&n);
    fractionArrayInput(a, n);
   fractionArrayOutput(a, n);
    printf("Inverse all fractions in the array.\n");
   inverseArray(a,n);
```

Solution:

```
    printf("Enter first fraction to compare:");
    fractionInput(s);
    printf("Enter second fraction to compare:");
    fractionInput(p);
    if (fractionCmp(s,p) ==0)
    printf("Two fractions are equals.\n");
    else if (fractionCmp(s,p) < 0){
    fractionOutput(s);
    printf("is smaller than");
    fractionOutput(p);
    printf("\n");
    }
    return 0;
}
```

Solution:

```
    s=TongCacPS(a, n);
     printf("\nTong gia tri cac phan so co trong mang: ");
     XuatPS(s);

     p=TichCacPS(a, n);
     printf("\nTich gia tri cac phan so co trong mang: ");
     XuatPS(p);

     NghichDaoCacPS(a, n);
     printf("\nMang phan so sau khi nghich dao cac phan tu: ");
     XuatMangPS(a, n);
     return 0;
}
```

Exercise 14.5

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