**Tutorial - Week 6- Solution**

**Objectives:** To practise with

* Struct , Union, and Enumeration
* User Defined Data Types

1. **What is the primary difference between a structure and an array? Which would you use to store the catalog description of a course? To store the names of students in the course?**

**Solution:**

A **structure can** have components/members of **different types**, but an **array’s elements** **must all be of the same type**. Use a structure for the catalog item and an array of strings for the list of student names.

1. **Define a struct data type date\_t and a function displayDate , that will output its date\_t type parameter in the form dd/mm/year.**

**Example: Assuming that a variable currentDate is of type date and its contents is**

**day month year**

**9 5 2016**

**The function call displaydate (currentDate) will output 9/5/2016**

**Solution:**

/\*--data type definition--\*/

**typedef** struct

{

char day; char month; int year;

} **date\_t**;

/\*-- function prototype --\*/

**void** displayDate(date\_t cDate)

/\* --function definition--\*/

**void** displayDate( **date\_t** cDate )

{

**printf**(“ %d/%d/%d”, cDate**.**day, cDate.month, cDate.year);

**return;**

}

1. **Considering the following C program segment**

**typedef struct**

{

char name[20];

int id;

float mark[5];

}person\_t;

person\_t groupOne[10];

Indicate whether the following statements are valid or invalid

* 1. person\_t.id =2907;
  2. groupOne[5].id = 2645;
  3. groupOne[0].mark[4]= 45.7;
  4. printf( "%d\n", groupOne.id );

**Solution:**

1. person\_t.id =2907;

Invalid as person\_t is a struct data type and not a struct variable

1. groupOne[5].id = 2645;

Correct as the array element groupOne[5] is a variable of person\_t data type whose one of its members is *id*

1. groupOne[0].mark[4]= 45.7;
2. printf( "%d\n", groupOne.id );

Invalid as groupOne is an array name and not an array element

1. **a) Define a type named long\_lat\_t that would be appropriate for storing longitude**

**or latitude values. The type comprises components named degrees (an integer), minutes (an integer), and direction (one of the characters 'N' , 'S' , 'E' , or 'W' ).**

**b) The following is a type to represent a geographic location and a variable of this hierarchical structure type. We will assume that STRSIZ means 20.**

typedef struct {

char place[STRSIZ];

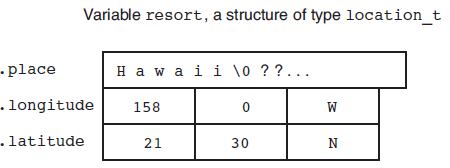
long\_lat\_t longitude,

latitude;

} location\_t;

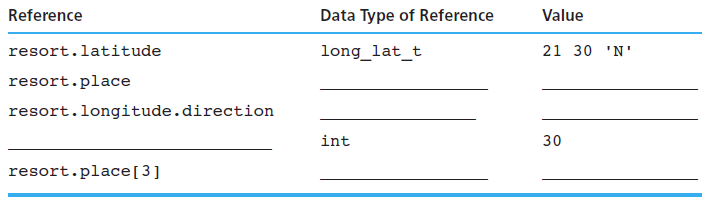
location\_t resort;

Figure 1 gives the content of the variable resort in the memory



**Figure 1**

**Complete the following table .**



**Solution:**

[[1]](#footnote-1)typedef struct {

int degrees;

int minutes;

char direction;

} long\_lat\_t;

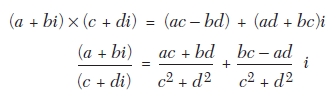
char place[STRSIZ] "Hawaii"

char 'W'

resort.latitude.minutes

char 'a' (the 4th char of Hawaii)

1. **Write functions multiply\_complex and divide\_complex to implement the operations of multiplication and division of complex numbers defined as follows:**



**Hint: Define a struct data type named complex\_t, with two double components, real and imag to hold the real and imaginary part of the complex number, respectively.**

**Solution:**

**typedef struct** {

**double** real, imag;

} **complex\_t;**

/\*

\* Returns product of complex values c1 and c2.

\*/

**complex\_t** multiply\_complex(**complex\_t** c1, **complex\_t** c2)

{

**complex\_t** cmult;

cmult.real = c1.real \* c2.real ‑ c1.imag \* c2.imag;

cmult.imag = c1.real \* c2.imag + c1.imag \* c2.real;

**return** (cmult);

}

/\*

\* Returns quotient of complex values (c1 / c2).

\*/

**complex\_t** divide\_complex(**complex\_t** c1, **complex\_t** c2)

{

**complex\_t** cdiv;

**double** denom;

denom = c2.real \* c2.real + c2.imag \* c2.imag;

cdiv.real = (c1.real \* c2.real + c1.imag \* c2.imag)/denom;

cdiv.imag = (c1.imag \* c2.real ‑ c1.real \* c2.imag)/denom;

**return** (cdiv);

}

1. **Given the following definitions:**

**typedef struct** {

**char** fst\_name[20],last\_name[20];

**int** score;

**char** grade;

} student\_t;

. . .

student\_t stu1, stu2;

1. **Identify the following statements as possibly valid or definitely invalid. If invalid, explain why.**
2. student\_t stulist[30];
3. printf("%s", stu1);
4. printf("%d %c", stu1.score, stu1.grade);
5. stu2 = stu1;
6. if (stu2.score == stu1.score)

printf("Equal");

1. if (stu2 == stu1)

printf("Equal structures");

1. scan\_student(&stu1);
2. stu2.last\_name = "Martin";

**Solution:**

1. Valid
2. Invalid: printf does not accept a C-struct variable.
3. Valid, access to the members of stu1 using dot operator.
4. Valid, unlike comparison, C-struct variables of the same datatype can be assigned to each other
5. Valid, comparing the members of C-struct variables (of a primitive data type) is permissible
6. Invalid: Equality operators cannot be used with C-struct variables
7. Valid (assuming parameter type is student\_t \* )
8. Invalid: Cannot copy strings with = except when intialising the variable during the declaration (to copy a literal string after declaration use the built-in function **strcpy**)
9. **Identify the type of each of the following references:**
10. stu1
11. stu2.score
12. stu2.fst\_name[3]
13. stu1.grade

**Solution:**

* 1. student\_t
  2. int
  3. char, as it is fst\_name which is an array of char
  4. char

1. **Write a statement that displays the initials of stu1 (with periods).**

**Solution:**

printf("%c.%c.", stu1.fst\_name[0],stu1.last\_name[0]);

1. **How many components does variable stu2 have?**

**Solution:**

Four

1. **Declare an array of 40 student\_t structures, and write a code segment that displays on separate lines the names (*last name, first name)* of all the students in the list.**

**Solution:**

student\_t students[40];

for (i = 0; i < 40; ++i)

printf("%s, %s\n", students[i].last\_name,

students[i].fst\_name);

1. **Write functions scan\_student and print\_student for type student\_t variables.**

**Solution:**

**The following code gives the entire program and not just the requested function defintions.**

#include<stdio.h>

**typedef struct** {

**char** fst\_name[20],last\_name[20];

**int** score;

**char** grade;

} **student\_t;**

**student\_t** scan\_student();

**void** print\_student(**student\_t** stu);

**int main**(**void**){

**student\_t** st;

st=scan\_student();

**if**(st.score!=-1) print\_student(st);

**else** printf("Error\n");

}

**student\_t** scan\_student()**/\* output - student structure to fill \*/**

{

**student\_t** error;

error.score=-1;

**student\_t** stu;

**int** status;

status = **scanf**("%s%s%d %c", stu.fst\_name,stu.last\_name,&stu.score, &stu.grade);

**// space before %c is needed to “swallow” the white space after user’s input integer**

**if** (status == 4) {

**return** (stu);

}

**else**  {

**return** error;

}

}

**void** print\_student(**student\_t** stu) /\* input - student structure

to display \*/

{

**printf**("Student: %s, %s\n", stu.fst\_name,stu.last\_name);

printf(" Score: %d Grade: %c\n", stu.score,stu.grade);

}

1. **What output is produced by the following program?**

typedef union /\* can store only one value at a time \*/

{

char var1;

int var2; double var3;

} mixData\_t;

int main(void)

{

mixData\_t myData;

myData.var1 = 'A'; myData.var2 = 35;

if(myData.var1 == 'A')

myData.var3 = 5.0;

else myData.var3 = -1.5;

printf (“%f”, myData.var3);

}

**Solution:**

Output: -1.5 as the character ‘A’ gets overwritten by 35 when setting var2 member value; therefore, the else statement gets executed

1. **Electromagnetic spectrum is subdivided into several bands:**

LF, MF, HF, VHF, UHF

**Define a new data type band\_t that can take only these values. Declare array channels of type band\_t that can store 44 elements. Write a function initChannels()that initializes all elements of the array with the value VHF using a for loop.**

/\*--data type definition--\*/

**typedef enum** { LF, MF, HF, VHF, UHF } band\_t;

# #define NUM\_OF\_CHAN 44

/\* -- array of bands --\*/

band\_t channels [ NUM\_OF\_CHAN];

/\*-- function prototype --\*/

void initChannels(band\_t chan[], int size );

/\* --function definition--\*/

**void** initChannels(band\_t chan[], **int** size )

{

**int** i;

**for**(i=0; i<size; i++)

chan[i] = VHF;

**return;**

}

1. **Write a function print\_day for enumerated type day\_t that displays its argument as a string.**

**void print\_day(day\_t cur\_day);**

***Hint:* Use a switch statement to select the appropriate printf statement.**

**typedef** enum {SUN, MON, TUE, WED, THU,FRI,SAT} day\_t;

**void** print\_day(day\_t cur\_day)

{

**switch** (cur\_day) {

**case** SUN: **printf**("Sunday\n");

**break;**

**case** MON: **printf**("Monday\n");

**break;**

**case** TUE: **printf**("Tuesday\n");

**break;**

**case** WED: **printf**("Wednesday\n");

**break;**

**case** THU: **printf**("Thursday\n");

**break;**

**case** FRI: **printf**("Friday\n");

**break;**

**case** SAT: **printf**("Saturday\n");

}

}

To test it, you can write

int main(void){

day\_t cur=SUN;

print\_day(cur);

}

1. https://gisgeography.com/latitude-longitude-coordinates/ [↑](#footnote-ref-1)