LAB 9
STRUCTURES

Faculty of Electronic Engineering Technology
Universiti Malaysia Perlis

1. OBJECTIVES:

1.1 To be able to define, declare and initialize structures.
1.2 To be able to access members of structures.
1.3 To be able to pass structures to functions.
1.4 To be able to create and use structures, arrays of structures and structures within structures in programs.

2. INTRODUCTION:

Structures are collections of related data items called components (or members) that are not necessarily of the same data type. Commonly used to define records to be stored in files.

Usually collections of related data item are characteristics of an object.

Object	Characteristics
■ Book	title, price, number of pages, year published
■ Car	price, model, colour
■ Student	name, matric_no, grade

2.1 Definition and Declaration of Structures

Before a structure can be used, we need to define the structure

■ Syntax Format:

```
struct <StructureTypeName>
{
    structure member declaration list
};
```

Example:

```
struct book
{ char title[20];
    float price;
    int numpages;
    int year;
  };
```

Note: **struct** is the keyword to define a structure.

There are two ways to declare a structure. For example, given a variable name **fiction_book**, it can be declared as:

■ struct <structure name> <variable_name>;

struct book **fiction_book;** or

■ It can be written directly after the structures definition

```
struct book
        { char title[20];
          float price; int
          numpages; int
          year;
        } fiction_book;
```

2.2 Arrays of Structures

We can also make structures to be in arrays, it is called arrays of structures. For example,

```
struct employeeType
{
    char
    firstName[20];
    char lastName[20];
    int personID; char
    deptID[10];
    double monthlySalary
    double monthlyBonus;
};
```

```
struct employeeType employees[50]; //We declare 50 elements of array type structure of
                                   //employeeType
```

2.3 Accessing Members of Structures

To access members of structures, we use the "**.**" (dot) operator. Refer to the previous **fiction_book** for the following examples.

i. To assign the **numpages** member of **fiction_book** with 120,
   fiction_book.numpages = 120; or
   scanf("%d",**&fiction_book.numpages**);

ii. To assign the **price** member of **fiction_book** with RM 1.00
   fiction_book.price = 1.00; or
   scanf ("%d",**&fiction_book.price**);

iii. To print members of the structure printf("Number of pages in fiction book is %d",**fiction_book.numpages**); printf("The price for fiction book is %f",**fiction_book.price**);

2.4 Structures Within a Structure

Structures within a structure format can be illustrated by the following example:

```
struct nameType
{   char first[30];
      char middle[30];
      char last[20];
};

struct addressType
{   char address1[40];
      char
      address2[40];
      char city[30]; char
      state;
      char zip;
};

struct dateType
{     char month[2];
      char day[2];
      char year[4];
 };
```

```
struct contactType
{   char phone[12];
     char cellphone[12];
      char fax[12];
      char pager[12];
      char email[50]; };


struct employeeType
{     nameType name;
      char emplID[12];
      addressType address;
      dateType hiredate;
      dateType quitdate;
       contactType contact;
      char deptID[10];
      double salary;
};
```

The **struct employeeType** contains structures of **nameType**, **addressType**, **dateType** and also **contactType**.

To access their members, use:

Declare a variable name **newEmployee** => **struct** employeeType **newEmployee;**

- **newEmployee.salary = 45678.00;**
- **newEmployee.name.first = "Mary";**
- **newEmployee.name.middle = "Beth";**
- **newEmployee.name.last = "Simmons";**

3. TASKS:

3.1 a. What is the main 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?

The difference between Array and Structure is Array consist to hold on type of data where Structure can hold humongous type of data.

b. Define a structure according to the information provided:

Structure name - **studentType**
Structure members -

- name: type char, has 50 characters
- no_ID: type integer
- test1: type float
- test2: type float
- total: type float

```
typedef struct studentType
{
   char name [50];
   int no_id;
   float test1;
   float test2;
} studentType;
```

c. Declare the following variables:

i. **Jason** type of **studentType**.
studentType Jason;

ii. **new_students** is an array of **studentType**, with 5 elements.
studentType new_students [5];

3.2 a. Define a structure type sSquareColumnT to represent a square timber column. Components should include the type of wood (a string of less than 20 characters), the length and width of the column in inches, and the maximum compressive strength of the wood in whole pounds per square inch. Here is a variable sq_col that it should be possible to declare using your data type.

sq_col

acType	Douglas fir
dLength	84.0
dWidth	6.5
iMaxCompStrength	445

```
typedef sq_col

{

   char acType [50];

   float dLength;

   float dWidth;

   float iMaxCompStrength;

}sq_col;
```

(b) Define a structure type inventory containing character array acPartName[30], integer iPartNumber, floating point fPrice, integer iStock, and integer iReorder.

```
typedef struct icontain
{
   char acPartName[30];
   int iPartNumber;
   float fPrice;
   int iStock;
   int iReorder;
}icontain;
```

(c) Define a struct called address, containing character arrays acStreetAddress[25], acCity[20], acState[3] and acZipcode[6].

```
typedef  struct icontain
{
   char acStreetAddress[25];
   char acCity [20]
   char acState [3]
   char acZipCode[6];
}icontain;
```

(d) Define a structure type cat_num_t to represent a catalog number consisting of a two- or threecharacter category code followed by an integer. Then define a structure type cat_entry_t to represent a catalog entry. Each entry has a catalog number, a description that is a string of up to 19 characters, a wholesale price, and a retail price.

```
typedef struct cat_num_t
{
   char acCode[25];
   int catalog;
}cat_num_t ;

typedef struct cat_entry_t
{
   cat_num_t catalog;
   char description [20];
   float wPrice;
   float rPrice;
}cat_entry_t ;
```

(e) Define two output functions for the data types you defined in Question 3.2.d: print_cat_num and print_cat_entry. Output of a catalog entry should resemble the following example.

Number: EXC-412
Description: 2400 BAUD MODEM
Wholesale price: $79.86
Retail price: $119.99

```
typedef struct cat_num_t
{
   char acCode[25];
   int catalog;
}cat_num_t ;

typedef struct cat_entry_t
{
   cat_num_t catalog;
   char description [20];
   float wPrice;
   float rPrice;
}cat_entry_t ;


int detail ( cat_entry_t iDetail)
{
   printf ("Number: %s - %d ",iDetail.catalog.acCode,iDetail.catalog.catalog);
   printf ("Description: %s",iDetail.description);
   printf ("Wholesale price: %.2f",iDetail.wPrice);
   printf ("Retail price: %.2f",iDetail.rPrice);

}
```

(f) Define an interactive input function for catalog numbers. Name the function fnScanCatNum. The function should prompt the user as follows.

Enter catalog number:
Enter letters before dash>
Enter digits after dash>

Function fnScanCatNum should take a single structured output argument through which it stores the catalog number entered. The function should return 1 for successful input, 0 for input error, and EOF if endfile is encountered before any data is obtained.

```
typedef struct cat_num_t
{
   char acCode[25];
   int catalog;
}cat_num_t ;

typedef struct cat_entry_t
{
   cat_num_t catalog;
   char description [20];
   float wPrice;
   float rPrice;
}cat_entry_t ;


int fnScanCatNum ( cat_num_t iInput)
{


   while (FileName!= EOF)
   {
      scanf ("%s",&iInput.acCode);
      scanf ("%d",&iInput.catalog);
      if (iInput.catalog != 1)
      {
         printf ("Error");
      }

      printf("successful input");
   }

}
```

3.3 Problem Solving Question.

Book's Title	Price (RM)	Quantity Sold
C_Programming	79.10	40
Dictionary	49.50	20
EasyAnalog	70.50	30
Electronics	125.60	10
Linux101	59.95	50

Table 10.1

Table 10.1 shows a list of best seller books in a book store. Write a program to calculate the total price for all the books sold. If the quantity sold is greater than or equal to 40 copies, a discount of 10% will be deducted from the book's price. Display book's title, price, quantity, total price, discounted amount and price after discount for each book, and finally calculate the total of amount payable to the book's supplier.

Requirement:

1. Define and declare **struct bookType** with **title** (char), **price** (float), **qty** (int) and **total** (float) as members.
2. Declare an array named **discount** to store calculated discount amount.
3. Declare an array named **after_discount** to store calculated total amount after discount.
4. Declare an array of structure **bookType** named **book** to store all the information for five (5) books.

Sample Output:

```
Enter book title : C_Programming

Enter book price : RM 79.10

Enter quantity ordered : 40
…….
…….
…….

Book's Title          Price    Quantity    Total      Discount    After Discount
  C_Programming       79.10       40      3164.00     316.40        2847.60
  Dictionary          49.50       20       990.00       0.00         990.00
  EasyAnalog          70.50       30      2115.00       0.00        2115.00
  Electronics        125.60       10      1256.00       0.00        1256.00
Linux101              59.95       50      2997.50     299.75        2697.75

Total amount payable : RM 9906.35
```

```
#include <stdio.h>


typedef struct bookType
{
   char bTitle [20];;
   float iPrice;
   int Quant;
   float iTotal;
}bookType;


int main ()
{
  bookType book [5];


  int i;
  float disc[5], afDisc[5],pay;



// Input
  for (i=0;i<5;i++)
  {
      printf("Enter book title :");
      scanf("%s",&book[i].bTitle);
      printf("Enter book price :");
      scanf("%f",&book[i].iPrice);
      printf("Enter quantity ordered :");
      scanf("%d",&book[i].Quant);
      printf("\n");
      book[i].iTotal = book[i].iPrice * book[i].Quant;


  }
```

```
// Calculation for Discount
  for (i=0;i<5;i++)
  {
      if (book[i].Quant >= 40)
      {
         disc [i] = book[i].iTotal * 0.1;
      }
  }

  // Calculation After Discount
  for (i=0;i<5;i++)
  {
     afDisc [i] = book[i].iTotal - disc[i];
  }

  // Output Program

  printf("Book's Title\t\tPrice\t\tQuantity\t\tTotal\t\tDiscount\tAfter Discount\n");

  for (i=0;i<5;i++)
  {

printf("%s\t\t%.2f\t\t%d\t\t%.2f\t\t%.2f\t\t%.2f\n",book[i].bTitle,book[i].iPrice,book[i].Quant,book[i].iTotal,di
sc[i],afDisc[i]);
  }

  for (i=0;i<5;i++)
  {
     pay += afDisc[i];
  }

  printf("Total amount payable : RM %.2f",pay);

}
```