Name Priyadarshi Prabhakar SAP ID 590029237

EXPERIMENT 7 : STRUCTURES & UNIONS

Activity 1: *Write a C program that uses functions to perform the following operations:*

a. Reading a complex number.

b. Writing a complex number.

c. Addition and subtraction of two complex numbers.

Note: *Represent complex number using a structure.*

ALGORITHM:

STEP 1: START

STEP 2: Define structure Complex with real and imag.

STEP 3: Create function **readComplex()** to read real and imaginary parts.

STEP 4: Create function **writeComplex()** to print complex number.

STEP 5: Create **addComplex()** to add two complex numbers.

STEP 6: Create **subComplex()** to subtract two complex numbers.

STEP 7: Read first complex number into **c1.**

STEP 8: Read second complex number into **c2.**

STEP 9: Compute **sum = addComplex(c1, c2).**

STEP 10: Compute **diff = subComplex(c1, c2).**

STEP 11: Display **c1, c2, sum,** and **diff** using **writeComplex().**

STEP 12: END

PSEUDOCODE :

```
STRUCT Complex
real : float
imag : float
ENDSTRUCT
FUNCTION readComplex() RETURNS Complex
DECLARE c AS Complex
PRINT "Enter real part: "
READ c.real
PRINT "Enter imaginary part: "
READ c.imag
RETURN c
ENDFUNCTION
FUNCTION writeComplex(c AS Complex)
IF c.imag >= 0 THEN
PRINT formatted: c.real + " + " + c.imag + "i"
ELSE
PRINT formatted: c.real + " - " + abs(c.imag) + "i"
ENDIF
ENDFUNCTION
FUNCTION addComplex(c1 AS Complex, c2 AS Complex) RETURNS Complex
DECLARE sum AS Complex
sum.real = c1.real + c2.real
sum.imag = c1.imag + c2.imag
```

```
RETURN sum
ENDFUNCTION
FUNCTION subComplex(c1 AS Complex, c2 AS Complex) RETURNS Complex
DECLARE diff AS Complex
diff.real = c1.real - c2.real
diff.imag = c1.imag - c2.imag
RETURN diff
ENDFUNCTION
MAIN
DECLARE c1, c2, sum, diff AS Complex
PRINT "Enter first complex number"
c1 = readComplex()
PRINT "Enter second complex number"
c2 = readComplex()
sum = addComplex(c1, c2)
diff = subComplex(c1, c2)
PRINT "First complex number: "
writeComplex(c1)
PRINT "Second complex number: "
writeComplex(c2)
PRINT "Sum: "
writeComplex(sum)
PRINT "Difference: "
```

writeComplex(diff)

ENDMAIN

CODE :

```
#include <stdio.h>
struct Complex {
float real;
float imag;
};
struct Complex readComplex() {
struct Complex c;
printf("Enter real part: ");
scanf("%f", &c.real);
printf("Enter imaginary part: ");
scanf("%f", &c.imag);
return c;
}
void writeComplex(struct Complex c) {
if (c.imag >= 0)
printf("%.2f + %.2fi\n", c.real, c.imag);
else
printf("%.2f - %.2fi\n", c.real, -c.imag);
}
```

```
struct Complex addComplex(struct Complex c1, struct Complex c2) {
struct Complex sum;
sum.real = c1.real + c2.real;
sum.imag = c1.imag + c2.imag;
return sum;
}
struct Complex subComplex(struct Complex c1, struct Complex c2) {
struct Complex diff;
diff.real = c1.real - c2.real;
diff.imag = c1.imag - c2.imag;
return diff;
}
int main() {
struct Complex c1, c2, sum, diff;
printf("Enter first complex number\n");
c1 = readComplex();
printf("\nEnter second complex number:\n");
c2 = readComplex();
sum = addComplex(c1, c2);
diff = subComplex(c1, c2);
printf("\nFirst complex number: ");
writeComplex(c1);
printf("Second complex number: ");
```

writeComplex(c2);

printf("\nSum: ");

writeComplex(sum);

printf("Difference: ");

writeComplex(diff);

return 0;

}

OUTPUT :

```
PS E:\Cprogramming works\LAB REPORT CODE> gcc .\complex.c
PS E:\Cprogramming works\LAB REPORT CODE> .\a.exe
Enter first complex number
Enter real part: 6
Enter imaginary part: 3

Enter second complex number:
Enter real part: 3
Enter imaginary part: 9

First complex number: 6.00 + 3.00i
Second complex number: 3.00 + 9.00i

Sum: 9.00 + 12.00i
Difference: 3.00 - 6.00i
PS E:\Cprogramming works\LAB REPORT CODE>
```

Activity 2 : *WAP to read a list of integers and store it in a single dimensional array. Write a C program to count and display positive, negative, odd, and even numbers in an array.*

ALGORITHM:

STEP 1: START

STEP 2: Declare structure Employee with **name, basic,** and **gross.**

STEP 3: Declare an array **emp[100].**

STEP 4: Read number of employees **n.**

STEP 5: Repeat for each employee **(i = 0 to n−1):**

a) Read employee name

b) Read basic pay

c) Compute **DA = 52% of basic pay**

d) Compute **gross = basic + DA**

STEP 6: Repeat for each employee

Print name and **gross salary**

STEP 7: END

PSEUDOCODE :

```
BEGIN
DECLARE Employee array emp[100]
DECLARE integer n, i
DECLARE float DA
PRINT "Enter number of employees"
READ n
FOR i = 0 TO n - 1 DO
PRINT "Enter name of employee i"
READ emp[i].name
PRINT "Enter basic pay of employee"
READ emp[i].basic
DA = 0.52 * emp[i].basic
emp[i].gross = emp[i].basic + DA
```

END FOR

PRINT "Name Gross Salary"

FOR i = 0 TO n - 1 DO

PRINT emp[i].name, emp[i].gross

END FOR

END

CODE :

```
#include <stdio.h>
struct Employee {
char name[50];
float basic, gross;
};
int main() {
struct Employee emp[100];
int n, i;
float da;
printf("Enter number of employees (max 100): ");
scanf("%d", &n);
for (i = 0; i < n; i++) {
printf("\nEnter name of employee %d: ", i + 1);
scanf("%s", emp[i].name);
```

```
printf("Enter basic pay of %s: ", emp[i].name);
scanf("%f", &emp[i].basic);
da = 0.52 * emp[i].basic;
emp[i].gross = emp[i].basic + da;
}
printf("\nName\t\tGross Salary\n");
printf("---------------------------\n");
for (i = 0; i < n; i++) {
printf("%s\t\t%.2f\n", emp[i].name, emp[i].gross);
}
return 0;
```

OUTPUT :

```
PS E:\Cprogramming works\LAB REPORT CODE> .\a.exe
Enter number of employees (max 100): 3

Enter name of employee 1: x
Enter basic pay of x: 5000

Enter name of employee 2: y
Enter basic pay of y: 4000

Enter name of employee 3: z
Enter basic pay of z: 9000

Name            Gross Salary
---------------------------
x               7600.00
y               6080.00
z               13680.00
PS E:\Cprogramming works\LAB REPORT CODE>
```

Activity 3 : *Create a Book structure containing book_id, title, author name and price. Write a C program to pass a structure as a function argument and print the book details.*

ALGORITHM:

STEP 1: START
STEP 2: Define structure **Book** with fields: **book_id, title, author, price**
STEP 3: Declare variable **b1** of type **Book**
STEP 4: Read Book ID into **b1.book_id**
STEP 5: Read Book Title into **b1.title**
STEP 6: Read Author Name into **b1.author**
STEP 7: Read Book Price into **b1.price**
STEP 8: Call function **displayBook(b1)** to print the details
STEP 9: END

PSEUDOCODE :

START

DECLARE structure Book with:

book_id : integer

title : string

author : string

price : float

DECLARE b1 as Book

PRINT "Enter Book ID"

READ b1.book_id

PRINT "Enter Book Title"

READ b1.title

PRINT "Enter Author Name"

READ b1.author

PRINT "Enter Book Price"

READ b1.price

CALL displayBook(b1)

END

FUNCTION displayBook(b)

PRINT "Book Details:"

PRINT "Book ID :", b.book_id

PRINT "Title :", b.title

PRINT "Author :", b.author

PRINT "Price :", b.price

END FUNCTION

CODE :

```
#include <stdio.h>
struct Book {
int book_id;
char title[50];
char author[50];
```

```
float price;
};
void displayBook(struct Book b) {
printf("\nBook Details:\n");
printf("Book ID : %d\n", b.book_id);
printf("Title : %s\n", b.title);
printf("Author : %s\n", b.author);
printf("Price : %.2f\n", b.price);
}
int main() {
struct Book b1;
printf("Enter Book ID: ");
scanf("%d", &b1.book_id);
printf("Enter Book Title: ");
scanf(" %[^\n]", b1.title);
printf("Enter Author Name: ");
scanf(" %[^\n]", b1.author);
printf("Enter Book Price: ");
scanf("%f", &b1.price);
displayBook(b1);
return 0;
}
```

OUTPUT :

```
PS E:\Cprogramming works\LAB REPORT CODE> gcc .\book.c
PS E:\Cprogramming works\LAB REPORT CODE> .\a.exe
Enter Book ID: 55
Enter Book Title: LetusC
Enter Author Name: Iitian
Enter Book Price: 250

Book Details:
Book ID   : 55
Title     : LetusC
Author    : Iitian
Price     : 250.00
PS E:\Cprogramming works\LAB REPORT CODE>
```

Activity 4 : *Create a union containing 6 strings: name, home_address, hostel_address, city, state, and zip. Write a C program to display your present address..*

ALGORITHM:

STEP 1: START
STEP 2: Declare a union **Address** with fields:
name, home_address, hostel_address, city, state, zip
STEP 3: Declare a variable **addr** of type **Address**
STEP 4: Read name into **addr.name**
STEP 5: Read **home_address** into **addr.home_address**
STEP 6: Read hostel_address (present address) into **addr.hostel_address**
STEP 7: Read city into **addr.city**
STEP 8: Read state into **addr.state**
STEP 9: Read zip into **addr.zip**
STEP 10: Print the present address stored in **addr.hostel_address**
STEP 11: END

PSEUDOCODE :

START

DECLARE union Address with:

name : string

home_address : string

hostel_address : string

city : string

state : string

zip : string

DECLARE addr as Address

PRINT "Enter your name"

READ addr.name

PRINT "Enter your home address"

READ addr.home_address

PRINT "Enter your present (hostel) address"

READ addr.hostel_address

PRINT "Enter your city"

READ addr.city

PRINT "Enter your state"

READ addr.state

PRINT "Enter your zip"

READ addr.zip

PRINT "Present Address:", addr.hostel_address

END

CODE :

```
#include <stdio.h>
#include <string.h>
union Address {
char name[50];
char home_address[100];
char hostel_address[100];
char city[50];
char state[50];
char zip[10];
};
int main() {
union Address addr;
printf("Enter your name: ");
scanf("%s", addr.name);
printf("Enter your home address: ");
scanf(" %[^\n]", addr.home_address);
printf("Enter your present address (hostel address): ");
scanf(" %[^\n]", addr.hostel_address);
printf("Enter your city: ");
```

```
scanf("%s", addr.city);
printf("Enter your state: ");
scanf("%s", addr.state);
printf("Enter your zip: ");
scanf("%s", addr.zip);
printf("\nPresent Address: %s\n", addr.hostel_address);
return 0;
```

OUTPUT :