শিক্ষা নিয়ে গড়বো দেশ তথ্য-প্রযুক্তির বাংলাদেশ

Bangabandhu Sheikh Mujibur Rahman Digital University, Bangladesh

Bangabandhu Sheikh Mujibur Rahman
Digital University

LAB REPORT-01

COURSE NO.- PROG 112
COURSE TITLE- OBJECT ORIENTED PROGRAMMING SESSIONAL

SUBMITTED BY

Mehrin Farzana
ID: 2101013
Department of IRE
Session :2021-2022
Bangabandhu Sheikh Mujibur Rahman Digital University, Bangladesh

SUBMITTED TO

Md.Toukir Ahmed
Lecturer
Department of IRE
Bangabandhu Sheikh Mujibur Rahman Digital University, Bangladesh

---

Date of Submission: 16 July, 2023

Problem No.: 01

Problem Name: Introduction to Structures

Code:

```
#include <stdio.h>
struct {
        char *engine;
        char *fuel_type;
        int fuel_tank_capacity;
        int seating_capacity;
        float city_mileage;
}car1, car2;

int main() {
        car1.engine="DDis 190 Engine";
        car1.fuel_type="Petrol";
        car1.fuel_tank_capacity=37;
        car1.seating_capacity=5;
        car1.city_mileage=19.74;

        car2.engine="1.2 L Kappa Dual VTVT";
        car2.fuel_type="Diesel";
        car2.fuel_tank_capacity=35;
        car2.seating_capacity=5;
        car2.city_mileage=22.54;

        printf("Car1 specifications\nEngine: %s\nFuel Type: %s\nFuel Tank
Capacity: %d\nSeating Capacity: %d\nCity Mileage: %.2f kmpl\n\n",
car1.engine,car1.fuel_type,car1.fuel_tank_capacity,car1.seating_capacity,car1.city_m
ileage);

        printf("Car2 specifications\nEngine: %s\nFuel Type: %s\nFuel Tank
Capacity: %d\nSeating Capacity: %d\nCity Mileage: %.2f kmpl\n",
car2.engine,car2.fuel_type,car2.fuel_tank_capacity,car2.seating_capacity,car2.city_m
ileage);
   return 0;
}
```

Output:

```
Car1 specifications
Engine: DDis 190 Engine
Fuel Type: Petrol
Fuel Tank Capacity: 37
Seating Capacity: 5
City Mileage: 19.74 kmpl

Car2 specifications
Engine: 1.2 L Kappa Dual VTVT
Fuel Type: Diesel
Fuel Tank Capacity: 35
Seating Capacity: 5
City Mileage: 22.54 kmpl
```

Explanation:

In a garage with two cars, it is intended to store all the information of the cars and then print them on the screen.
The above code creates 2 variables, car1 and car2, of type Car, which is a structure of 5 members, 2 char, 2 int and a float, where we store the car1 and car2 specifications and then printing them on the screen.
Car1 specifications:

Engine: DDis 190 Engine
Fuel Type: Petrol
Fuel Tank Capacity: 37
Seating Capacity: 5
City Mileage: 19.74 kmpl

Car2 specifications:

Engine: 1.2 L Kappa Dual VTVT
Fuel Type: Diesel
Fuel Tank Capacity: 35
Seating Capacity: 5
City Mileage: 22.54 kmpl

Problem No.: 02

Problem Name: Structure Types, Using Structure Tags

Code:

```
#include <stdio.h>

struct employee{
   char *name;
   int age;
   int salary;
};

int manager(){
   struct employee manager;
   manager.age=27;
   if(manager.age>30)
      manager.salary = 65000;
   else
      manager.salary = 55000;
   return manager.salary;
}

int main(){
   struct employee emp1, emp2;
   printf("Enter the salary of employee1: ");
   scanf("%d", &emp1.salary);
   printf("Enter the salary of employee2: ");
   scanf("%d", &emp2.salary);
   printf("Employee 1 salary is: %d\n", emp1.salary);
   printf("Employee 2 salary is: %d\n", emp2.salary);
   printf("Manager's salary is: %d\n", manager());
   return 0;
}
```

Input:

```
Enter the salary of employee1: 20000
Enter the salary of employee2: 25000
```

Output:

```
Employee 1 salary is: 20000
Employee 2 salary is: 25000
Manager's salary is: 55000
```

Explanation:

The above code creates a global struct with a tag, employee, for it to access within a function to create local variables of that struct type with the help of the tag.

The manager() function creates a variable of type employee to store the salary via conditional statements based on the given age.

Reads input from the user for emplyee1 and employee2's salary. And then finally prints the salaries on the screen.

Problem No.: 03

Problem Name: Application of Unions

Code:

```
#include <stdio.h>
#pragma pack(1)

struct store{
    double price;
    union{
        struct{
            char *title;
            char *author;
            int num_pages;
        }book;
        struct{
            int color;
            int size;
            char  *design;
        }shirt;
    }item;
};

int main(){
    struct store s;
    s.item.book.title = "The Alchemist";
    s.item.book.author = "Paulo Coelho";
    s.item.book.num_pages = 197;
    printf("%s\n",s.item.book.title);
    printf("%ld\n", sizeof(s));

    return 0;
}
```

Output:

```
The Alchemist
28
```

Explanation:

A store sells two kinds of items.
1. Books
2. Shirts

Store owner wants to keep records of above mentioned items along with the relevent information.

Books: Title, Author, number of pages, price
Shirt: Color, Size, Design, price

The above code creates a structure, with tag being "store", of having members of a double and a union having two structures as its member.

Then declares a variable of 'store' type and stores information in its union member, in its struct book member variable.

Then prints the value stored in one of its members and the size of the structure.

Problem No.: 04

Problem Name: Program to Find Area of Rectangle Using Structures

Code:

```
#include <stdio.h>

struct point{
   int x;
   int y;
};

struct rectangle{
   struct point upper_left;
   struct point lower_right;
};

int area(struct rectangle r){
   int length, breadth;
   length = r.lower_right.x - r.upper_left.x;
   breadth = r.upper_left.y - r.lower_right.y;
   return length*breadth;
}

int main(){
   struct rectangle r;
   printf("Enter the upper left coordinates of the rectangle: \n");
   scanf("%d%d", &r.upper_left.x, &r.upper_left.y);
   printf("Enter the lower right coordinates of the rectangle: \n");
   scanf("%d%d", &r.lower_right.x, &r.lower_right.y);
   printf("Area of the rectangle: %d\n", area(r));
}
```

Input:

```
Enter the upper left coordinates of the rectangle:
2 5
Enter the lower right coordinates of the rectangle:
5 1
```

Output:

```
Area of the rectangle: 12
```

Explanation:

The following structures are designed to store information about objects on a graphics screen:
struct point { int x, y; };
struct rectangle {"struct point upper_left, lower_right; };
A point structure stores the x and y coordinates of a point on the screen. A rectangle structure stores the coordinates of the upper left and lower right corners of the rectangle.

The above code writes a function that accepts rectangle structure r as an argument and computes the area of r and prints it onto the screen.

Problem No.: 05

Problem Name: Program to find the size of the Union

Code:

```
#include <stdio.h>

union test1 {
	int x;
	int y;
} Test1;

union test2 {
	int x;
	char y;
} Test2;

union test3 {
	int arr[10];
	char y;
} Test3;

int main()
{
	int size1 = sizeof(Test1);
	int size2 = sizeof(Test2);
	int size3 = sizeof(Test3);

	printf("Sizeof test1: %d\n", size1);
	printf("Sizeof test2: %d\n", size2);
	printf("Sizeof test3: %d", size3);
	return 0;
}
```

Output:

```
Sizeof test1: 4
Sizeof test2: 4
Sizeof test3: 40
```

Explanation:

The above code declares three unions named Test1, Test2 and Test3. The first union has two integer members named x and y. The second union has an integer member named x and a character member named y. The third union has an array of ten integers named arr and a character member named y. The driver code then uses the sizeof() operator to find the size of each union and prints it to the console.

Problem No.: 06

Problem Name: Program to check if its possible to store data in multiple Union

Code:

```
#include <stdio.h>

union test {
	int x, y;
};

int main()
{

	union test t;

	t.x = 2;
	printf("After making x = 2:\n x = %d, y = %d\n\n", t.x,
		t.y);

	t.y = 10;
	printf("After making y = 10:\n x = %d, y = %d\n\n", t.x,
		t.y);
	return 0;
}
```

Output:

```
After making x = 2:
 x = 2, y = 2

After making y = 10:
 x = 10, y = 10
```

Explanation:

The code you provided declares a union named test with two integer members named x and y. The driver code then declares a union variable t and assigns the value 2 to its member x. Since x and y share the same memory location, y also gets the value 2. The program then prints the values of x and y. The driver code then assigns the value 10 to t's member y. Since x and y share the same memory location, x is also updated to 10. The program then prints the values of x and y again.

Problem No.: 07

Problem Name: Program to illustrate differences between Structure and Union

Code:

```
#include <stdio.h>
#include <string.h>

struct struct_example {
        int integer;
        float decimal;
        char name[20];
};

union union_example {
        int integer;
        float decimal;
        char name[20];
};

void main()
{
        struct struct_example s = { 18, 38, "geeksforgeeks" };

        union union_example u = { 18, 38, "geeksforgeeks" };

        printf("structure data:\n integer: %d\n"
                "decimal: %.2f\n name: %s\n",
                s.integer, s.decimal, s.name);
        printf("\nunion data:\n integer: %d\n"
                "decimal: %.2f\n name: %s\n",
                u.integer, u.decimal, u.name);
        printf("\nsizeof structure : %d\n", sizeof(s));
        printf("sizeof union : %d\n", sizeof(u));
        printf("\n Accessing all members at a time:");
        s.integer = 183;
        s.decimal = 90;
        strcpy(s.name, "geeksforgeeks");

        printf("structure data:\n integer: %d\n "
                "decimal: %.2f\n name: %s\n",
                s.integer, s.decimal, s.name);
        u.integer = 183;
        u.decimal = 90;
        strcpy(u.name, "geeksforgeeks");

        printf("\nunion data:\n integer: %d\n "
                "decimal: %.2f\n name: %s\n",
                u.integer, u.decimal, u.name);

        printf("\n Accessing one member at time:");

        printf("\nstructure data:");
        s.integer = 240;
        printf("\ninteger: %d", s.integer);

        s.decimal = 120;
        printf("\ndecimal: %f", s.decimal);

        strcpy(s.name, "C programming");
        printf("\nname: %s\n", s.name);

        printf("\n union data:");
        u.integer = 240;
        printf("\ninteger: %d", u.integer);

        u.decimal = 120;
        printf("\ndecimal: %f", u.decimal);

        strcpy(u.name, "C programming");
        printf("\nname: %s\n", u.name);


        printf("\nAltering a member value:\n");
        s.integer = 1218;
        printf("structure data:\n integer: %d\n "
                " decimal: %.2f\n name: %s\n",
                s.integer, s.decimal, s.name);

        u.integer = 1218;
        printf("union data:\n integer: %d\n"
                " decimal: %.2f\n name: %s\n",
                u.integer, u.decimal, u.name);
}
```

Output:

```
structure data:
 integer: 18
decimal: 38.00
 name: geeksforgeeks

union data:
 integer: 18
decimal: 0.00
 name: _x0012_

sizeof structure : 28
sizeof union : 20

 Accessing all members at a time:structure data:
 integer: 183
 decimal: 90.00
 name: geeksforgeeks

union data:
 integer: 1801807207
 decimal: 277322871721159507258114048.00
 name: geeksforgeeks

 Accessing one member at time:
structure data:
integer: 240
decimal: 120.000000
name: C programming

 union data:
integer: 240
decimal: 120.000000
name: C programming

Altering a member value:
```

```
structure data:
 integer: 1218
  decimal: 120.00
 name: C programming
union data:
 integer: 1218
 decimal: 0.00
 name: �
```

Explanation:

This code defines two user-defined data types: `struct struct_example` and `union union_example`.

The structure `struct_example` has three members: an integer named `integer`, a float named `decimal`, and a character array named `name`.

The union `union_example` has three members: an integer named `integer`, a float named `decimal`, and a character array named `name`.

The code initializes both the structure and union with the same values. It then prints out the values of each member variable in both the structure and union. Finally, it changes the value of one member variable in each data type and prints out the new values.