Assignment - 10

Q) What is the concept of structure? explain with example.

- structure is considered as user defined datatype
- structure can hold any heterogenious datatype based on programmar need.
- this known as user defined datatype because its contains are based on the programmars needs & performance is user of that programming language.
- The concept of structure is extended in C++ & java where it is known as class
- class is most important datatype in any object oriented programming language
- structure is userdefined datatype which can store any primitive, Derived and userdefined datatype.
- Declaration of structure also called as structure prototype

- when we declare the structure the compiler will predict the no. of bytes of that structure but at the time of declaration memory is not allocation.

- memory for structure gets Allocated when we create the object of structure.

- to Access or fetch the member of structure we use . operator (dot) operator called as direct Accessing operator

e.g

```
# include <stdio.h>
Struct Demo
{
    int no;
    float f;
    int x;
};
int main()
{
 Struct Demo obj; // object creation
  obj. no = 10;
  obj. f = 10.5f;
  obj. x = 20;

printf("sizeof(Demo) Structure is = %d\n",
        sizeof (obj));
```

```
struct Demo obj2;

obj2.na = 20;
obj2.f = 20.5f;
obj2.x = 30;

printf("sizeof(Demo) structure is = %d",
                sizeof(obj2));

return 0;
}
```

2) Explain the concept of padding in memory allocation of structure

→

- padding is considered as allocated memory inside the object that we can not Access it is considered as westage of memory to Access the demand in fast way compiler will Assign some extra memory is

- Padding also known as Structure padding or data structure padding is technique that inserts empty bytes between data members in a structure to align them in memory.

Q3) Write the difference between structure and union

→

Structure	union
1) the size of Structure is equal to or greater than total size of its all member	1) the size of union the size of its largest member.
2) the structure can store the data for multiple member at a same time	2) union can contain data for only one time Appl^{n}:- Binary tree we use union
3) All field Share the different memory allocation	3) Each member have the same independent memory allocation
4) more storage space is required	4) minimum storage space is required

Q4) which type of data can we store in structure and union

→

- structure and union are the user defined datatypes.
- structure and union can hold or store any hetrogenius datatype based on programmer need

Q5) what are the different ways to initialize the members of structure

→

1) we can create the object of structure with structure declare at the end of structure.

for e.g:-

```
struct Demo
{
    int no;
    float f;
}; str obj, strobj2
```

2) we can create object of structure independantly whenever require

for e.g: Strud Demo obj;

3) we can initalize the object of structure by using the concept of member initalization list.

for e.g:- Struct Demo Strobj 3 { 2, 60.5f, 700, 700.5f };

4) we can create an array as a member of structure

for e.g:-

```
# include <stdio.h>
Struct Demo
{
  int no;
  int Arr[3]; ← Array declar
  float f;
};
int main()
{
  struct Demo obj;
    obj.no = 10;

  obj.Arr[0] = 20; } member by
  obj.Arr[1] = 30; } member
  obj.Arr[2] = 40; } initali
                    zation
  return 0;
```

Q6] why we can't initalize members of structure at the time of declaration?

→ - when we declare the structure the compiler will predict the no of bytes of that structure but at the time of declaration memory is not Allocated

- At the time of declaration of structure we can not initalize id member with its values

- initalization not allowed because at that point there is no memory Allocation for structure

- the memory Allocation is done after creating the object of structure.

Q7) write the difference between structure and Array

→

structure	Array
1) structure can store any type hetrogenius datatype	1) Array can store only homongenius elements in it.
2) structure is user defined datatype	2) Array is Derived datatype.
3) memory is Allocated to the structure is after creating the object	3) memory for all the elements of Array its Allocated in sequence way
4) padding is included in structure	4) Padding is not in Array

Q8) write the different between direct access (.) & indirect access(→) operator with example & program

Direct Access (.) operator

- when we have object of structure or union then we can Access its value using (.) operator it is called as direct Accessing operator).

- in c++ the dot operator is used to access class, structure or union members. the member is specified

e.g:-

```
# include <stdio.h>
 Struct Demo {
{
    int x;
};
int main()
{
    Struct Demo obj = {30};
    printf("%d", obj.A);

    return 0;
}
```

Indirect Accessing operator (→)

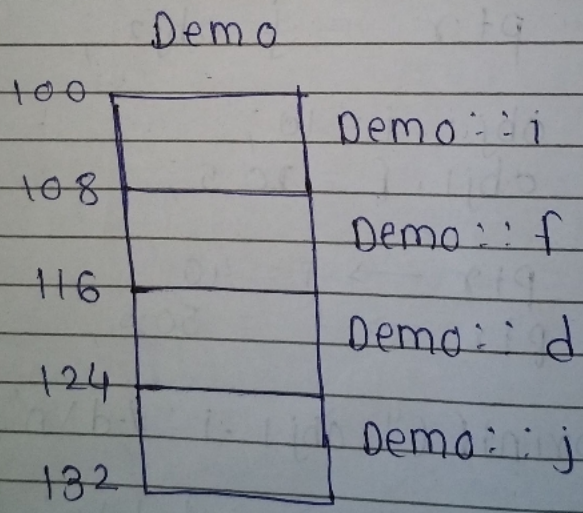
- when we have a pointer which point to the object of structure or union the Access to its member we use → (Arrow operator) which is called indirect Accessing operator.

```
# include <stdio.h>
Struct Demo
{
    int i;
    float f;
};
int main ()
{
  Struct Demo obj1;

  Struct Demo obj2;

  Struct Demo* ptr = NULL;
      ptr = &obj2;

      obj1.i = 10;
      obj1.f = 20.5;

      ptr -> i = 40
      ptr -> f = 50.5;

  printf (" obj1.i : %d \n", obj1.i);

  printf ("obj1.i : %d \n", obj2.i);

  printf (" ptr -> i : %d \n", ptr -> i);

  return 0;
}
```

Q 9) Draw the memory layout for the below structure

```
Struct Demo
{
    int i;
    float f;
    double d;
    int j;
};
```

memory layout



Q10] find out the problem in the below syntax and correct it, and draw the memory layout for below structure

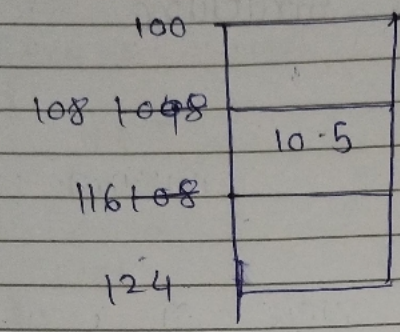
a]
```
struct Demo
{
   int i;
  float f = 10.5;
   double d;
};
```

- in the above program variable f is initalized ~~in~~ at the time of declaration, this is not allowed in structure becaus the memory get Allocated after creating object so we can initalize the value at the time of object creation

correction:

```
struct Demo
{
   int i;
   float f;
  double d;
};
int main() {
struct Demo obj;

  obj.f = 10.5f;
   return 0;
        }
```

memory layout



b]

```
Struct Demo
{
   int i;
  float f = 10.5;
  double d;
};
```

Same as above syntax

correction:-

```
Struct Demo
{
   int i;
  float f;
  double d;
};
int main ()
Struct Demo obj;
  obj.f = 10.5f;
  return 0;
}
```