CS & IT ENGINEERING

'C' Programming

Structures & Unions

Lecture No.- 02

By- Satya sir

Recap of Previous Lecture

- Storage classes, Static Scoping Vs Dynamic Scoping [C]
- Structure / union
  - Collection of dissimilar data
  - Secondary / Derived datatype
- Structure / union Declaration
- Members, Variables
- Member access
  - Variable . member
  - Variable -> member
- Nested Structure / union

Topics to be Covered

- Size of Structure / Variable / Member
- Size of Union / Variable / member
- Array of structure Variables
- Limitations

Topic : Structures & Unions - 2

- Size of Structure == Size of Structure Variable == Σ (members size)

- Size of Union == Size of Union Variable == Max (members size)

- In Computer, Memory is Word-addressable (Every word is accessed in 1 clock cycle)

- So, To keep Number of clock cycles as minimum as Possible, while accessing data from memory,

  Memory will follow / implement alignment (or) Padding.

  - 1 word == 16-bits for 16-bit Processor
    == 32-bits for 32-bit Processor
    == 64-bits for 64-bit Processor
    | :

Topic : Structures & Unions - 2

Ignoring Alignment Considerations

Ex:

```
Struct ABC
{
    int i;      = 2 Bytes
    Char j;     = 1 Byte
    float K;    = 4 Bytes
                ---------
                Σ = 7 Bytes
} V1, V2, *V3;
```

Let 2 Bytes Per integer

Size of (V1) = 7 Bytes
Size of (V2) = 7 Bytes
Size of (V3) = 2 Bytes == Pointer size is Equal to int size
Size of (Struct ABC) = 7 Bytes

Ex:

```
Union ABC
{
    int i;      = 2 Bytes
    Char j;     = 1 Byte
    float K;    = 4 Bytes
                ---------
                Max = 4 Bytes
} V1, V2, *V3;
```

Size of (V1) = 4 Bytes
Size of (V2) = 4 Bytes
Size of (V3) = 2 Bytes
Sizeof (Union ABC) = 4 Bytes

Topic : Structures & Unions - 2

Ex:

```
Struct ABC
{   int i;                    ---> 2 Bytes
    float j;                  ---> 4 Bytes
    Struct xyz                ---> 9 Bytes
    {   char P;   = 1 Byte        Σ = 15 Bytes
        double d; = 8 Bytes
                  Σ = 9 Bytes
    } V1, *V2;
} V3, *V4;
```

sizeof(V1) = 9 Bytes sizeof(V2) = 2 Bytes

sizeof(V3) = 15 Bytes sizeof(V4) = 2 Bytes

sizeof(Struct xyz) = 9 Bytes sizeof(Struct ABC) = 15 Bytes

```
Union ABC
{   int i;                    ---> 2 Bytes
    float j;                  ---> 4 Bytes
    Union xyz                 ---> 8 Bytes
    {   char P;   ---> 1 Byte     Max = 8 Bytes
        double d; ---> 8 Bytes
                  Max = 8 Bytes
    } V1, *V2;
} V3, *V4;
```

V1 = 8 Bytes V2 = 2 Bytes

V3 = 8 Bytes V4 = 2 Bytes

xyz = 8 Bytes ABC = 8 Bytes

Topic : Structures & Unions - 2

Ex:

```
Struct ABC
{
    int i;          ----------> 2 Bytes
    float j;        ----------> 4 Bytes
    union XYZ       ----------> 8 Bytes
    {                           Σ = 14 Bytes
        char P;     -> 1B
        double d;   -> 8B
                    Max = 8B
    } V1, *V2;
} V3, *V4;
```

V1 = 8 Bytes V2 = 2 Bytes

V3 = 14 Bytes V4 = 2 Bytes

XYZ = 8 Bytes ABC = 14 Bytes

```
Union ABC
{
    int i;          ----------> 2B
    float j;        ----------> 4B
    Struct XYZ      ----------> 9B
                                Max = 9 Bytes
    {
        char P;     -> 1B
        double d;   -> 8B
                    Σ = 9 Bytes
    } V1, *V2;
} V3, *V4;
```

V1 = 9 Bytes V2 = 2 Bytes

V3 = 9 Bytes V4 = 2 Bytes

XYZ = 9 Bytes ABC = 9 Bytes

Topic : Structures & Unions - 2

Considering Memory Alignment

Let 32-bit Processor

⇒ 1 Word == 4 Bytes

⇒ 1 int = 4 Bytes

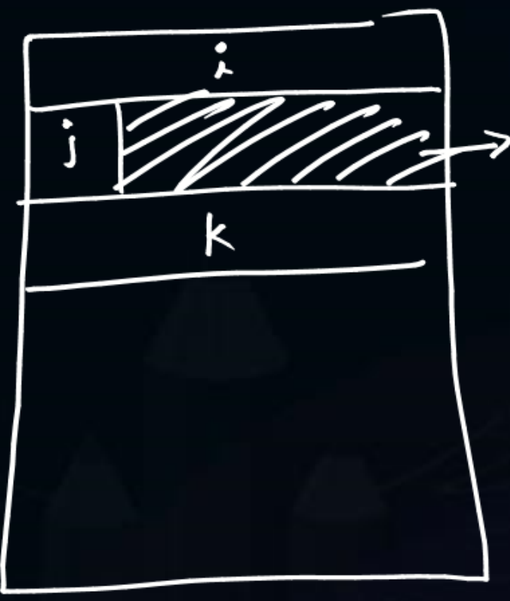
Struct ABC

```
{
1 cycle   int  i;   → 1 cycle
1 cycle   Char j;   → 1 cycle
1 cycle   float k;  → 2 cycles (1 cycle (word 1) + 1 cycle (word 2))
} V1;
```

Without Padding

CPU ⟵ 4 Bytes in 1 cycle

Without alignment



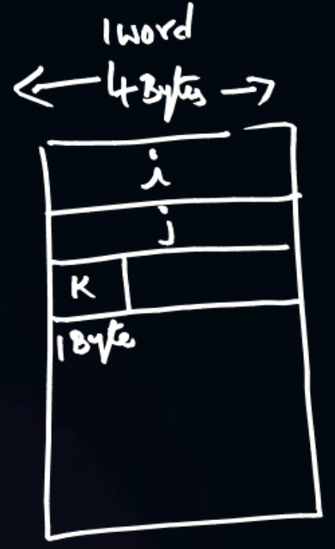
With Alignment

Size of (V1) == Size of (Struct ABC) == 12 Bytes (with Padding)

== 9 Bytes (without Padding)

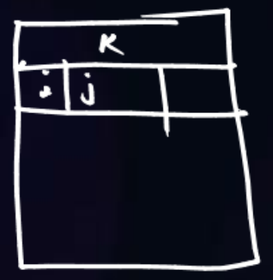
Topic : Structures & Unions - 2

```
Struct ABC
{  int i;   // 1 cycle
   float j; // 1 cycle
   char K;  // 1 cycle
} V1;
```



Sizeof(V1) = 9 Bytes

Sizeof(Struct ABC) = 9 Bytes



NOTE: Size of Structure is Not Needed to be Sum of all Members Size.

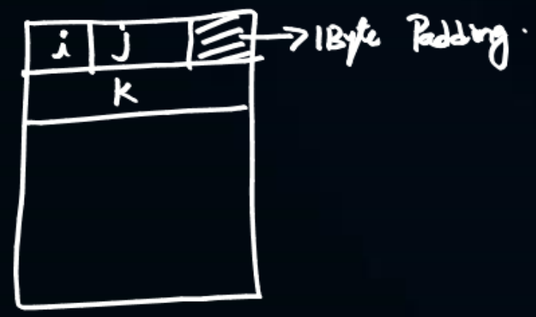
```
Struct ABC
{  float k;   (OR)  float k;
   int j;           char i;
   char i;          int j;
};
```

Size = 7 Bytes

[32-bit Processor]

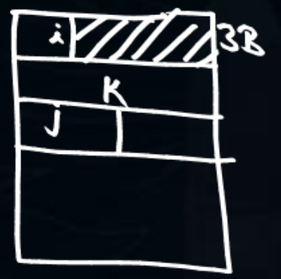
Let CPU can access 4 Bytes in a cycle. But, int size = 2 Bytes [16-bit Compiler]

```
Struct ABC
{  char i;
   int j;
   float K;
} V1;
```



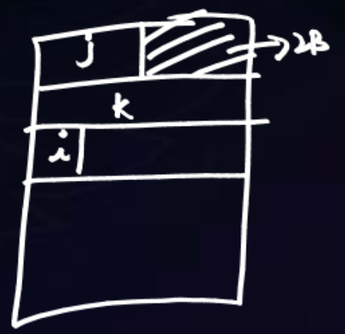
Sizeof(V1) == sizeof(Struct ABC) == 8 Bytes

```
Struct ABC
{  char i;
   float K;
   int j;
};
```



Size = 10 Bytes

```
Struct ABC
{  int j;
   float K;
   char i;
};
```



Size = 9 Bytes

Topic : Structures & Unions - 2

```
1   Struct ABC
2   {
        int i=10; // Error
3       char j;
4       float k;
    };
5   Void main( )
    {
6   Struct ABC V1={9, '@', 4.25}; // Memory allocated
7       Printf(" %d", V1.i);
8       Printf (" %c", V1.j);
9       Printf (" %f", V1.k);
    }
```

Structure members cannot be initialized at the time of declaration.

Topic : Structures & Unions - 2

```
Struct ABC
{    static int i;   // i value need to be assigned as zero.
     char j;
     float k;
 };
 Void main( )
{
   Struct ABC V1;
   Printf("%d", V1.i);
}
```

// Error

NOTE: Structure (or) Union members cannot be Static, extern, register

2 mins Summary

- Structures & Unions
  - Memory Size
  - Alignment Considerations
- Limitations

To be Contd . . .

THANK - YOU