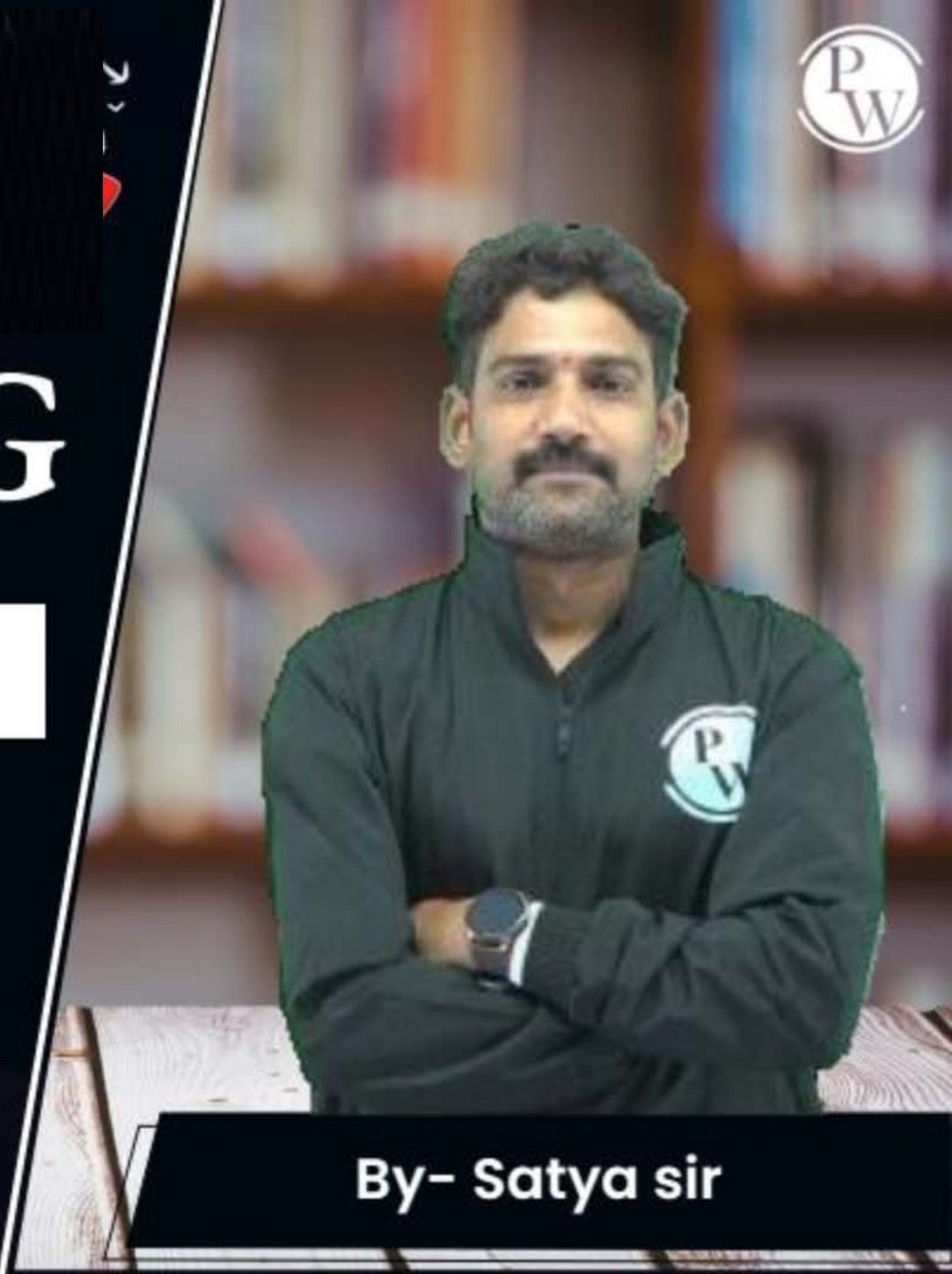
CS & IT ENGINEERING

'C' Programming

Structures & Unions

Lecture No.- 01



By- Satya sir

Recap of Previous Lecture

- Storage classes
  - It defines,
    Scope, Lifetime, Default value, Memory location
- Static Scoping vs Dynamic Scoping
- PYQ Practice on Recursion

Topics to be Covered

- Storage classes
  - auto, register, static, extern
- PYQ Practice
- Structures & unions

Topic : Storage Classes

Lifetime: Active limits

↳ The extent upto which a Variable created is active (alive in Memory)

- within the block (or) Through out Program

4 storage classes in 'C' Programming:

- Automatic - auto
- Register - register
- Static - static
- External - extern

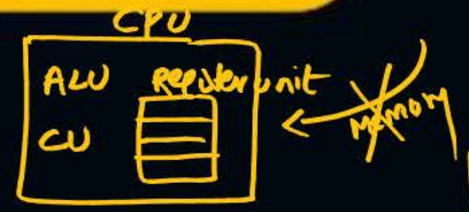
Syntax: Storage class keyword datatype Variable;

Ex:
```
auto int a;
static char b;
register float c;
extern double d;
```

Topic : Storage Classes

Storage class	Default value	Memory location	Scope	Life time
Automatic (Default)	Garbage value	RAM	Block scope	With in the block
Register	Garbage value	CPU Registers	Block scope	With in the block
Static	Zero	RAM	Block scope	Throughout the Program
External	Zero	RAM	File scope	Throughout the Program

Topic : Storage Classes



Example :

```
Void f( )
{
  int i=10; // auto int i=10;
        (OR)
  register int i=10;

  i= i+5;
  Printf("%d", i);
}
Void main( )
{
  int j;
  for(j=1; j<=5; j++)
    f();
}
```

o/p: 15 15 15 15 15

j=1 f() ~~i=10 15~~
j=2 f() ~~i=10 15~~
j=3 f() i=10 15
j=4 f() i=10 15
j=5 f() i=10 15

- Static (or) External Variables are initialized only once.

o/p: 15 20 25 30 35

```
Void f( )
{
  static int i=10; // extern int i=10;

  i=i+5;
  Printf("%d", i);
}
Void main( )
{
  int j;
  for(j=1; j<=5; j++)
    f();
}
```

j=1 f() ~~i=10~~ 15
j=2 f() ~~i=15~~ 20
j=3 f() i=20 25
j=4 f() ~~i=25~~ 30
j=5 f() i=30 35

- Register Storage class ⇒ when a Variable is accessed repeatedly many times.
- Automatic Storage class ⇒ Preferred, for Variables that are accessed less Number of times.
- External Storage class ⇒ If a Variable accen is required globally (within the directory (folder))
- Static Storage class ⇒ If all changes (Modifications) Performed on variable, need to be Preserved

Topic : Storage Classes

ISRO 2017

#Q. What is the output of the following program?

```
#include<stdio.h>
int tmp=20;
main()
{
  printf("%d", tmp);
  func();
  printf("%d", tmp);

}
func()
{
 static int tmp=10;
 printf("%d", tmp);
}
```

a) 20 10 10
b) 20 10 20 ✓
c) 20 20 20
d) 10 10 10

o/p: 20 10 20

Topic : Storage Classes

GATE 2019

#Q. Consider the following C program:

```
#include <stdio.h>
int r() {
    static int num=7;
    return num--;
}
int main() {
    for (r();r();r())
        printf("%d",r());
    return 0;
}
```

⇒ return num, num=num-1

r() return 7
r() return 6 (True)
r() return 5, Printed 5
r() return 4
r() return 3 (True)
r() return 2, Printed 2
r() return 1
r() return 0 (False)

Which one of the following values will be displayed on execution of the programs?

a) 41 b) 52 c) 63 d) 630

o/p: 52

Topic : Storage Classes

GATE 2020

NAT

#Q. Consider the following C functions.

```
int fun1(int n) {
  static int i= 0;
  if (n > 0) {
    ++i;
    fun1(n-1);
  }
 return (i);          }
int fun2(int n) {
  static int i= 0;
  if (n>0) {
    i = i+ fun1 (n);
    fun2(n-1) ;
  }
return (i);          }
```

The return value of fun2(5) is 55

fun2(5) n=5
i = 0+5 = 5

fun2(4) n=4
i = 5+fun1(4)
= 5+9 = 14

fun2(3) n=3
i = 14+fun1(3)
= 14+12 = 26

fun2(2)
i = 26+fun1(2)
= 26+14 = 40

fun2(1)
i = 40+fun1(1)
= 40+15
= 55

fun2(0)
return 55 ✓

fun1(5) → fun1(4) → fun1(3) → fun1(2) → fun1(1) → fun1(0)
return 5

fun1(4) → fun1(3) → fun1(2) → fun1(1) → fun1(0)
return 9

fun1(3) → fun1(2) → fun1(1) → fun1(0)
return 12

fun1(2) → fun1(1) → fun1(0)
return 14

fun1(1) → fun1(0) return 15

Topic : Structures & Unions - 1

Structure / Union : Collection of dissimilar (or) Heterogeneous data

— Derived Data Type [Extension of Primary]

Ex:

1) Student_details

- Name, Father Name, location // Strings
- Gender M/F/T // character
- Mobile Number // long int
- GATE_AIR // int
- GATE_Percentage // float

2) Emp_Details

- Emp_Name, Department // Strings
- Gender // character
- Emp_ID, Salary // int
- Mobilenumber // long int

Topic : Structures & Unions - 1

Structure (or) Union Declaration

Syntax:

```
union/
struct  Name
{
    datatype1  data;
    datatype2  data;
    :
};
```

Members (defined within the block)

Ex:

```
union/struct  Student_info
{
    char Name[20], City[10];
    char gender;
    long int Mobile;
    int AIR;
};
```

Variable declaration

1) Along with structure/union

```
union/Struct  Name
{
    datatype1 data;
    datatype2 data;
    :
} Variables;
```

Ex:

```
Struct ABC
{
    int i;
    float j;
    char k;
} V1, V2, V3---;
```

2) Later Declaration

```
union/Struct  Name
{
    datatype1 data;
    :
};
    :
union/struct  Name  Variables;
```

Ex: `Struct ABC V4;`

Topic : Structures & Unions - 1

Access Members of Structure (or) Union

- Through variables only, members can be accessed.

If Normal variable ⇒ `Variable . member;`

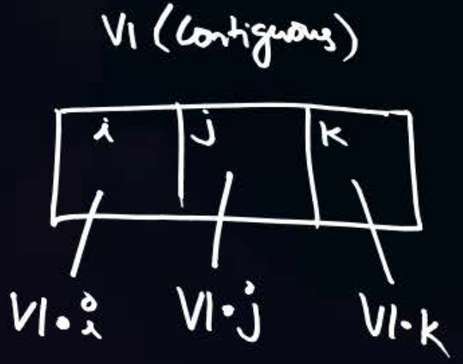
If Pointer Variable ⇒ `Variable -> member;`

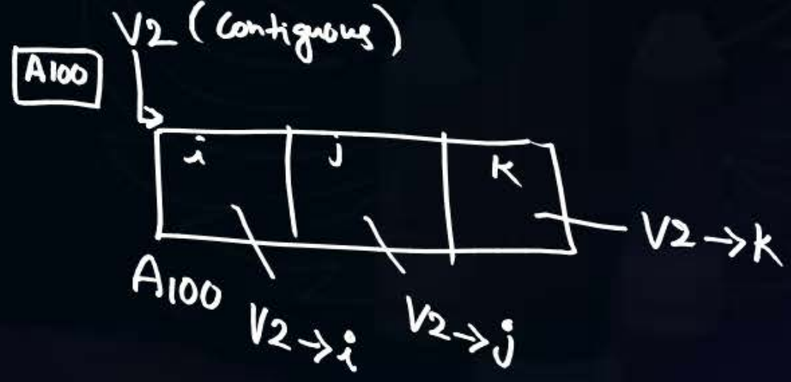
- All Members of one variable will be Stored in Contiguous locations.

But, Variables need not be Contiguous.

Ex:

```
struct ABC
{
    int i;
    float j;
    char k;
} V1, *V2;
```





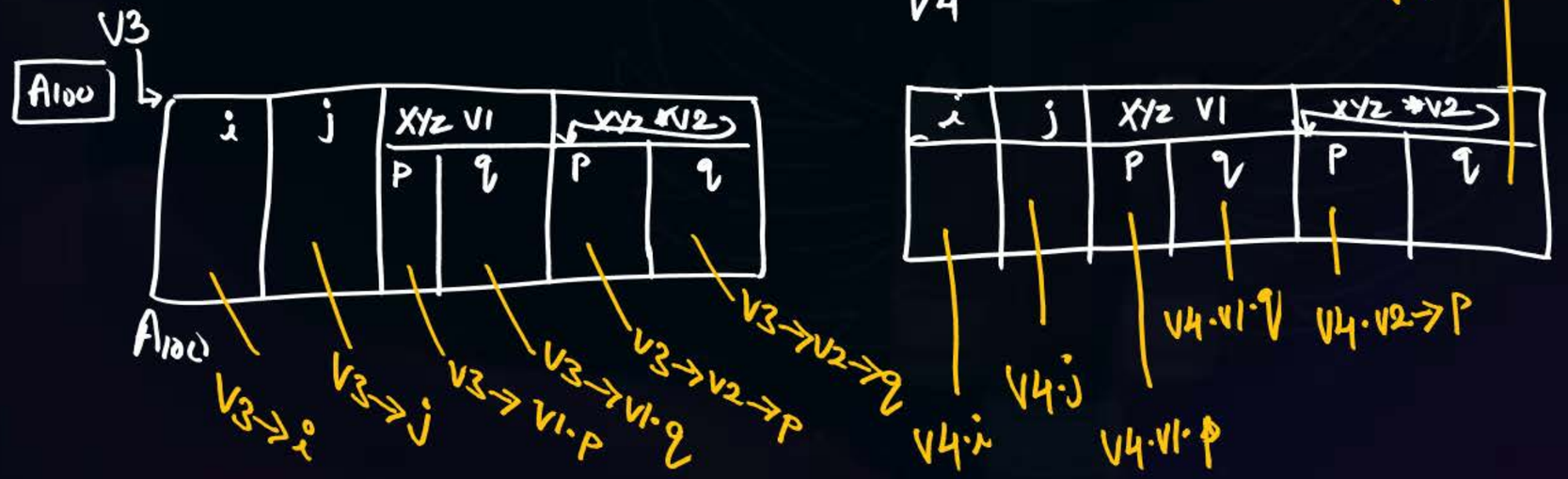
Topic : Structures & Unions - 1

Nested Structures / Unions

```
Struct                Struct {              Union {               Union
{                                                                 {
    Struct                Union {               Struct {              Union
    {                                                                     {
                                    }                     }                 }
    }                                                                 }
}                     }             }             }
```

Ex:

```
Struct ABC
{
    int i;
    float j;
    Struct xyz
    {
        int P;
        char q;
    } V1, *V2;
} *V3, V4;
```



2 mins Summary

- Storage classes (auto, register, static, extern)
- Structure & union
  - Definition
  - Declaration
  - Variable, Member
  - Member access
  - Nested Structures.

THANK - YOU