Storage Classes in C

From compiler's point of view, a variable name identifies some physical location within the computer, where the string of bits representing the variable value is stored. A variable's storage class tells us:

a) Where the variable would be stored.

b) What would be the scope of the variable i.e. in which fun^n the value of the variable would be available.

Tuesday • June 21

c) What will be the initial value of the variable.

d) What is the life of the variable i.e. how long would the variable exist.

There are Four Storage classes in C

→ (i) Automatic Storage Class.
→ (ii) Register Storage Class.
→ (iii) Static Storage Class.
→ (iv) External storage Class

Relative Difference between these classes

2011 22 June • Wednesday

	Automatic	Static	Register	External
Storage	Memory	Memory	CPU Register	Memory
Default Initial value	Garbage	Zero	Garbage	Zero
Scope	Local to the block in which the variable is defined	Local to the block in which the variable is defined	Local to the block in which the variable is defined.	Global throughout the program
Life	Till the control remains within the block in which the variable is defined	value of the variable persists between different function Calls	Till the control remains within the block in which the variable is defined	As long as the programs execution does not come to an end.

23 June • Thursday

Which is to use when

⇒ Use |Static| storage class only if we want the value of a variable to persist between different fun calls.

⇒ Use |register| storage class for only those variables that are being used very often in a program.

⇒ Use |extern| storage class for only these variables that are being used by almost all the functions in the program.

Saturday • June 25

⇒ If we donot have any of the express needs mentioned above, then we use the |auto Storage| Class. In fact most of the times we use auto variables. This key word auto is not mendatory to use in the program.

Sunday 26

Different Examples :-

(Automatic Storage Class)

①
```
main()
{
auto int i, j;
printf("%d %d", i, j);
}
```

⋆ O/P → It will print garbage / unpredictable value.

②
```
main()
{ auto int i = 1;
  { auto int i = 2;
    {
      auto int i = 3;
      printf("%d", i);
    }
    printf("%d", i);
  }
  printf("%d", i);
}
```

June • Tuesday

28

⋆ It will print 3 2 1

Register Storage Class

(1)
```
main()
{ register int i;
  for(i=1; i<=5; i++)
      printf("%d", i);
}
```

(2) O/p → 1 2 3 4 5

Static

Thursday • June 30

Static storage class

(1)
```
main()
{ for(int i=1; i<=5; i++)
    increment();
}

increment()
{
  auto int i=0;
  printf("%d", i);
  i++;
}
```

(2) O/p → 0 0 0 0 0

(2)
```
main()
{
for(i=1; i<=5; i++)
   increment();
}

increment()
{
  static int i;
  printf("%d", i);
   i++;
}
```

(*) O/P → 0 1 2 3 4

External Storage Class :- 02

July • Saturday

```
int i;
main()
{ printf("i=%d", i);
  increment();
  increment();
  decrement();
  decrement();
}
increment()
{
  i++;
  printf("\n on incrementing i=%d", i);
}
decrement()
{
  i = i-1;
  printf("\n on decrementing i=%d", i);
}
```

Sunday 03

(*) O/P → i = 0
on incrementing i = 1
on incrementing i = 2
on decrementing i = 1
on decrementing i = 0