Storage Classes in C

Storage classes in C are used to determine the lifetime, visibility, memory location, and initial value of a variable. There are four types of storage classes in C

* auto
* extern
* static
* register

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Storage Classes** | **Storage Place** | **Default Value** | **Scope** | **Lifetime** |
| auto | RAM | 0 | Local | Within function |
| extern | RAM | Zero | Global | Till the end of the main program Maybe declared anywhere in the program |
| static | RAM | Zero | Local | Till the end of the main program, Retains value between multiple functions call |
| register | Register | Garbage value | Local | Within the function |

**Automatic**

Automatic variables are allocated memory automatically at runtime.

The visibility of the automatic variables is limited to the block in which they are defined.

The automatic variables are initialized to garbage by default.

The memory assigned to automatic variables gets freed upon exiting from the block.

The keyword used for defining automatic variables is auto.

Every local variable is automatic in C by default.

**Example 1**

#include <stdio.h>

int main()

{

int a; //auto

float c;

printf("%d %f",a,c); // printing initial default value of automatic variables a, b, and c.

return 0;

}

**Output:**

0 0.000000

**Scope of a variable**

#include <stdio.h>

void display();

int a=30;

int main()

{

auto int a=10;

printf("function scope:%d\n",a);

{

int a=20;

printf("block scope%d\n",a);

}

display();

}

void display()

{

printf("program scope:%d",a);

}

Output:

function scope:10

block scope20

program scope:30

Static

* The variables defined as static specifier can hold their value between the multiple function calls.
* Static local variables are visible only to the function or the block in which they are defined.
* A same static variable can be declared many times but can be assigned at only one time.
* Default initial value of the static integral variable is 0 otherwise null.
* The visibility of the static global variable is limited to the file in which it has declared.
* The keyword used to define static variable is static.

Example

#include<stdio.h>

void sum()

{

static int a = 10;

static int b = 24;

int c=11;

printf("%d %d %d\n",a,b,c);

a++;

b++;

c++;

}

void main()

{

int i;

for(i = 0; i< 3; i++)

{

sum(); // The static variables holds their value between multiple function calls.

}

}

**Output:**

10 24 11

11 25 11

12 26 11

Register

* The variables defined as the register is allocated the memory into the CPU registers depending upon the size of the memory remaining in the CPU.
* We can not dereference the register variables, i.e., we can not use &operator for the register variable.
* The access time of the register variables is faster than the automatic variables.
* The initial default value of the register local variables is garbage.
* The register keyword is used for the variable which should be stored in the CPU register. However, it is compiler?s choice whether or not; the variables can be stored in the register.

Example 1

1. #include <stdio.h>
2. **int** main()
3. {
4. **register** **int** a; // variable a is allocated memory in the CPU register. The initial default value of a is 0.
5. printf("%d",a);
6. }

**Output:**

garbage value

Example 2

1. #include <stdio.h>
2. **int** main()
3. {
4. **register** **int** a = 0;
5. printf("%u",&a); // This will give a compile time error since we can not access the address of a register variable.
6. }

**Output:**

main.c:5:5: error: address of register variable ?a? requested

printf("%u",&a);

^~~~~~

External:

* The default initial value of external integral type is 0
* We can only initialize the extern variable globally, i.e., we can not initialize the external variable within any block or method.
* An external variable can be declared many times but can be initialized at only once.
* If a variable is declared as external then the compiler searches for that variable to be initialized somewhere in the program which may be If it

Example 1

#include <stdio.h>

int main()

{

extern int a; // Compiler will search here for a variable a defined and initialized somewhere in the //pogram or not.

printf("%d",a);

}

int a = 20;

**output:**

20

EXAMPLE2:

#include <stdio.h>

**int** a;

**int** main()

{

**extern** **int** a = 0; // this will show a compiler error since we can not use extern and initializer at same time

printf("%d",a);

}

**Output**

compile time error

main.c: In function ?main?:

main.c:5:16: error: ?a? has both ?extern? and initializer

extern int a = 0;