**C++ Storage Classes**

Storage class is used to define the lifetime and visibility of a variable and/or function within a C++ program.

Lifetime refers to the period during which the variable remains active and visibility refers to the module of a program in which the variable is accessible.

There are five types of storage classes, which can be used in a C++ program

1. Automatic
2. Register
3. Static
4. External
5. Mutable

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Storage Class** | **Keyword** | **Lifetime** | **Visibility** | **Initial Value** |
| Automatic | auto | Function Block | Local | Garbage |
| Register | register | Function Block | Local | Garbage |
| Mutable | mutable | Class | Local | Garbage |
| External | extern | Whole Program | Global | Zero |
| Static | static | Whole Program | Local | Zero |

**Automatic Storage Class**

It is the default storage class for all local variables. The auto keyword is applied to all local variables automatically.

1. {
2. auto int y;
3. float y = 3.45;
4. }

The above example defines two variables with a same storage class, auto can only be used within functions.

**Register Storage Class**

The register variable allocates memory in register than RAM. Its size is same of register size. It has a faster access than other variables.

It is recommended to use register variable only for quick access such as in counter.

Note: We can't get the address of register variable.

1. register int counter=0;

**Static Storage Class**

The static variable is initialized only once and exists till the end of a program. It retains its value between multiple functions call.

The static variable has the default value 0 which is provided by compiler.

1. #include <iostream>
2. using namespace std;
3. void func() {
4. static int i=0; //static variable
5. int j=0; //local variable
6. i++;
7. j++;
8. cout<<"i=" << i<<" and j=" <<j<<endl;
9. }
10. int main()
11. {
12. func();
13. func();
14. func();
15. }

Output:

i= 1 and j= 1

i= 2 and j= 1

i= 3 and j= 1

**External Storage Class**

The extern variable is visible to all the programs. It is used if two or more files are sharing same variable or function.

1. extern int counter=0;

NOTES – II

Every variable in C++ has two features: type and storage class.

Type specifies the type of data that can be stored in a variable. For example: int, float, char etc.

And, storage class controls two different properties of a variable: lifetime (determines how long a variable can exist) and scope (determines which part of the program can access it).

Depending upon the storage class of a variable, it can be divided into 4 major types:

* [Local variable](https://www.programiz.com/cpp-programming/storage-class#local_variable)
* [Global variable](https://www.programiz.com/cpp-programming/storage-class#global_variable)
* [Static local variable](https://www.programiz.com/cpp-programming/storage-class#static_variable)
* [Register Variable](https://www.programiz.com/cpp-programming/storage-class#register variable)
* [Thread Local Storage](https://www.programiz.com/cpp-programming/storage-class#thread_local_storage)

## Local Variable

A variable defined inside a function (defined inside [function](https://www.programiz.com/cpp-programming/function) body between braces) is called a local variable or automatic variable.

Its scope is only limited to the function where it is defined. In simple terms, local variable exists and can be accessed only inside a function.

The life of a local variable ends (It is destroyed) when the function exits.

### Example 1: Local variable

#include <iostream>

using namespace std;

void test();

int main()

{

// local variable to main()

int var = 5;

test();

// illegal: var1 not declared inside main()

var1 = 9;

}

void test()

{

// local variable to test()

int var1;

var1 = 6;

// illegal: var not declared inside test()

cout << var;

}

The variable var cannot be used inside test() and var1 cannot be used inside main() function.

Keyword auto was also used for defining local variables before as: auto int var;

But, after [C++11 auto](http://en.cppreference.com/w/cpp/language/auto) has a different meaning and should not be used for defining local variables.

## Global Variable

If a variable is defined outside all functions, then it is called a global variable.

The scope of a global variable is the whole program. This means, It can be used and changed at any part of the program after its declaration.

Likewise, its life ends only when the program ends.

### Example 2: Global variable

#include <iostream>

using namespace std;

// Global variable declaration

int c = 12;

void test();

int main()

{

++c;

// Outputs 13

cout << c <<endl;

test();

return 0;

}

void test()

{

++c;

// Outputs 14

cout << c;

}

**Output**

13

14

In the above program, c is a global variable.

This variable is visible to both functions main() and test() in the above program.

### Static Local variable

Keyword static is used for specifying a static variable. For example:

... .. ...

int main()

{

static float a;

... .. ...

}

A static local variable exists only inside a function where it is declared (similar to a local variable) but its lifetime starts when the function is called and ends only when the program ends.

The main difference between local variable and static variable is that, the value of static variable persists the end of the program.

### Example 3: Static local variable

#include <iostream>

using namespace std;

void test()

{

// var is a static variable

static int var = 0;

++var;

cout << var << endl;

}

int main()

{

test();

test();

return 0;

}

**Output**

1

2

In the above program, test() function is invoked 2 times.

During the first call, variable var is declared as static variable and initialized to 0. Then 1 is added to var which is displayed in the screen.

When the function test() returns, variable var still exists because it is a static variable.

During second function call, no new variable var is created. The same var is increased by 1 and then displayed to the screen.

**Output of above program if var was not specified as static variable**

1

1

## Register Variable (Deprecated in C++11)

Keyword register is used for specifying register variables.

Register variables are similar to automatic variables and exists inside a particular function only. It is supposed to be faster than the local variables.

If a program encounters a register variable, it stores the variable in processor's register rather than memory if available. This makes it faster than the local variables.

However, this keyword was deprecated in C++11 and should not be used.

## Thread Local Storage

Thread-local storage is a mechanism by which variables are allocated such that there is one instance of the variable per extant thread.

Keyword thread\_local is used for this purpose.