|  |
| --- |
| Close-up image showing the leaf-sides of two oversized books side-by-side on a bookshelf, with additional books in soft focus background |
| [Document title]  [Document subtitle] |
| |  |  |  | | --- | --- | --- | | Ranjan-EXT, Rajeev | 1/17/13 | [Course title] | |

**Contents**

[1. Storage classes in C 2](#_Toc519605446)

[2. Difference between global and static variable 2](#_Toc519605447)

[3. Structure and unions 2](#_Toc519605448)

[4. Size of structure 2](#_Toc519605449)

[5. structure padding 2](#_Toc519605450)

[6. Functions(call by value, call by reference) 3](#_Toc519605451)

[7. Recursion and stack overflow with example 3](#_Toc519605452)

[8. Pre-processors directives(macro) 3](#_Toc519605453)

[9. C's memory map Execution of c program(flow chart) 3](#_Toc519605454)

[10. Dynamic memory allocation 3](#_Toc519605455)

[11. Typedef 3](#_Toc519605456)

[12. Difference between #define and constant 4](#_Toc519605457)

[13. Difference constant and volatile(Qualifiers) 4](#_Toc519605458)

[14. Pointers(NULL, Wild, Dangling) 4](#_Toc519605459)

[15. Endianness(little, Big) 4](#_Toc519605460)

[16. Pointer arithmetic 4](#_Toc519605461)

[17. Function pointers 4](#_Toc519605462)

[18. Pointers and array 5](#_Toc519605463)

# Storage classes in C

<https://cprogrammingcodes.blogspot.com/2012/02/comparison-storage-classes.html>

<http://cs-fundamentals.com/c-programming/storage-classes-in-c-and-storage-class-specifiers.php>

<https://www.geeksforgeeks.org/storage-classes-in-c/>

# Difference between global and static variable

Global variables are declared outside of all functions. Global variables are stored in Data Segment of process. Global variable’s life is until the life of program and it can be accessed by other files using extern keyword.

Static variable can be declared outside of all functions or within a function using static keyword before the data type of variable. If static variables are declared outside of all functions it will have global scope or it is declared within a function it will have scope within a function and it will retain until the life of program. Static variables are initialized only once at the time of declaration only. Static variables are not accessible by other files using extern

<https://www.geeksforgeeks.org/static-variables-in-c/>

<https://www.geeksforgeeks.org/g-fact-19-redeclaration-of-global-variable-in-c/>

<https://www.codingunit.com/c-tutorial-functions-and-global-local-variables>

<https://www.tutorialspoint.com/cprogramming/c_scope_rules.htm>

# Structure and unions

<https://www.geeksforgeeks.org/difference-structure-union-c/>

<https://www.geeksforgeeks.org/structures-c/>

# Size of structure

<https://www.geeksforgeeks.org/structure-member-alignment-padding-and-data-packing/>

# structure padding

https://fresh2refresh.com/c-programming/c-structure-padding/

Datatypes

<https://www.geeksforgeeks.org/data-types-in-c/>

<https://www.tutorialspoint.com/cprogramming/c_data_types.htm>

# Functions(call by value, call by reference)

<https://www.geeksforgeeks.org/functions-in-c/>

<https://beginnersbook.com/2014/01/c-function-call-by-value-example/>

<https://beginnersbook.com/2014/01/c-function-call-by-reference-example/>

# Recursion and stack overflow with example

<https://www.geeksforgeeks.org/recursion/>

<https://www.tutorialspoint.com/cprogramming/c_recursion.htm>

<https://www.geeksforgeeks.org/heap-overflow-stack-overflow/>

# Pre-processors directives(macro)

<https://www.geeksforgeeks.org/cc-preprocessors/>

<https://www.tutorialspoint.com/cprogramming/c_preprocessors.htm>

https://fresh2refresh.com/c-programming/c-preprocessor-directives/

# C's memory map Execution of c program(flow chart)

<https://www.geeksforgeeks.org/memory-layout-of-c-program/>

<https://www.geeksforgeeks.org/compiling-a-c-program-behind-the-scenes/>

<https://www.javatpoint.com/flow-of-c-program>

# Dynamic memory allocation

<https://www.tutorialspoint.com/cprogramming/c_memory_management.htm>

<https://www.programiz.com/c-programming/c-dynamic-memory-allocation>

<https://www.geeksforgeeks.org/what-is-dynamic-memory-allocation/>

https://www.w3schools.in/c-tutorial/dynamic-memory-allocation/

https://www.geeksforgeeks.org/c-language-2-gq/dynamic-memory-allocation-gq/

# Typedef

<https://www.tutorialspoint.com/cprogramming/c_typedef.htm>

<https://www.geeksforgeeks.org/typedef-versus-define-c/>

https://fresh2refresh.com/c-programming/c-typedef/

# Difference between #define and constant

https://www.geeksforgeeks.org/diffference-define-const-c/

# Difference constant and volatile(Qualifiers)

<https://www.geeksforgeeks.org/understanding-volatile-qualifier-in-c/>

The const qualifier tells the compiler that the variable's value should not be changed once it has been initialized. If we declare a const variable as

 Const int k=55;

then any subsequent attempt to modify the variable k will be flagged as an error by the compiler. The const qualifier can also be used by the compiler to perform certain compiler optimizations like placing these variables in a special read-only memory block.

The Volatile qualifier tells the compiler not to perform any optimizations on the variable. One of the optimizations that is performed by most modern computers is to place some variables in the cache memory instead of the main memory to optimize speed of access. The volati1 e qualifier specifies that the variable is heavily used and could be shared by some other programs running in parallel (either in a multithreaded environment or using some interrupt based scheme). Therefore, variables qualified as vo1atil e should not be optimized in any manner and should be stored in the main memory at all times.

# Pointers(NULL, Wild, Dangling)

<https://www.geeksforgeeks.org/few-bytes-on-null-pointer-in-c/>

<https://www.geeksforgeeks.org/what-are-wild-pointers-how-can-we-avoid/>

https://www.geeksforgeeks.org/dangling-void-null-wild-pointers/

# Endianness(little, Big)

<https://www.geeksforgeeks.org/little-and-big-endian-mystery/>

# Pointer arithmetic

<https://www.tutorialspoint.com/cprogramming/c_pointer_arithmetic.htm>

<https://www.geeksforgeeks.org/pointers-in-c-and-c-set-1-introduction-arithmetic-and-array/>

# Function pointers

<https://www.geeksforgeeks.org/function-pointer-in-c/>

<https://beginnersbook.com/2014/01/c-function-pointers/>

# Pointers and array

<https://www.programiz.com/c-programming/c-pointers-arrays>

<https://www.geeksforgeeks.org/pointer-array-array-pointer/>

<https://www.studytonight.com/c/pointers-with-array.php>

Basic C programs (reverse a number, palindrome, count no. of bits, find 2r0 largest no. in array, factorial, Fibonacci series, sum of digits etc etc)