



# MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L-1

CSE

I / I

Course Name with Code : Programming For Problem Solving Using C - 19GES01

Course Faculty : M.Ganthimathi

Unit : I - Introduction to C Programming Date of Lecture:

**Topic of Lecture:** Introduction to computer software

**Introduction :**

- Software is a set of programs, which is designed to perform a well-defined function.
- A program is a sequence of instructions written to solve a particular problem

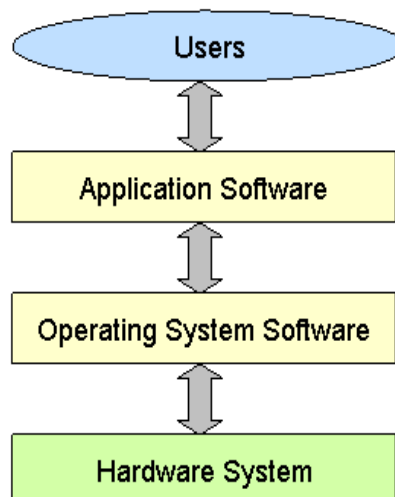
**Prerequisite knowledge for Complete understanding and learning of Topic:**

1. Knowledge of computer
2. Knowledge of software

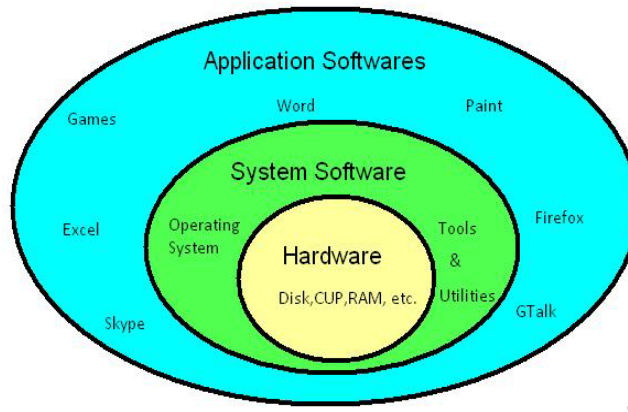
**Detailed content of the Lecture:**

- Computer: A programmable electronic device designed to accept data, perform prescribed mathematical and logical operations at high speed, and display the accurate results of these operations.
- It can expanded as C – Common, O- Operating, M- Machine, P- Purposely, U - Used for , T-Technological , E –Educational and R -Research

Components of Computer System



## Computer comprises of software and Hardware



## Computer – Software

- Software is a set of programs, which is designed to perform a well-defined function
- A program is a sequence of instructions written to solve a particular problem

## There are two types of software

### 1. System Software :

- The system software is a collection of programs designed to operate, control, and extend the processing capabilities of the computer itself
- System software is generally prepared by the computer manufacturers
- These software written in low-level languages, which interact with the hardware at a very basic level
- System software serves as the interface between the hardware and the end users
- Examples : Operating System, Compilers, Interpreter, Assemblers, etc.

### 2. Application Software

- Application software products are designed to satisfy a particular need of a particular environment
- It is a collection of programs, often called a software package, which work together to accomplish user task, such as a spreadsheet package
- Some examples: Payroll Software , Student Record Software , Income Tax Software and Railways Reservation Software

## Difference between hardware and Software :

<b>Hardware</b>	<b>Software</b>
Hardware is further divided into four main categories: <ul style="list-style-type: none"><li>• Input Devices</li><li>• Output Devices</li><li>• Secondary Storage Devices</li><li>• CPU</li></ul>	Software is further divided into two main categories: <ul style="list-style-type: none"><li>• Application Software</li><li>• System Software</li></ul>
Developed using electronic and other materials	Developed by using a programming language

When damaged, it can be replaced with a new component	When damaged it can be installed once more using a backup copy
Hardware is physical in nature and hence one can touch and see hardware	The software cannot be physically touched but still can be used and seen
Hardware cannot be infected by Viruses	The software can be infected by Viruses
An example of Hardware is hard drives, monitors, CPU, scanners, printers etc..	An example of software is Windows 10, Adobe Photoshop, Google Chrome etc..

Difference between System Software and Application Software:

Key	System Software	Application Software
Definition	System Software is the type of software which is the interface between application software and system	Application Software is the type of software which runs as per user request. It runs on the platform which is provide by system software
Development Language	low level language	high level language
Usage	System software is used for operating computer hardware	Application software is used by user to perform specific task
Installation	Installed on the computer when operating system is installed	Application software are installed according to user's requirements
Dependency	System software can run independently, It provides platform for running application software	Application software can't run independently. They can't run without the presence of system software
Examples	compiler, assembler, debugger, driver, etc	word processor, web browser, media player, etc.

**Video Content / Details of website for further learning (if any):**

<https://www.youtube.com/watch?v=wvfAT2ahIcU>

//www.centralacademy.ac.in/software-its-types

**Important Books/Journals for further learning including the page nos.:**

Computer Fundamentals and Programming in C - Reema Thareja: Oxford University Press, Second Edition.pp 1-5

Course Faculty

Verified by HOD





# MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L-2

CSE

I / I

Course Name with Code : Programming For Problem Solving Using C - 19GES01

Course Faculty : M.Ganthimathi

Unit : I - Introduction to C Programming Date of Lecture:

**Topic of Lecture:** Program Design Tools: Algorithms, Flowcharts, Pseudo codes

**Introduction :**

Program Design tools are the tools used to develop a program. Popular tools and technique used to represent the programs are Algorithm, Flowchart, and Pseudocode

**Prerequisite knowledge for Complete understanding and learning of Topic:**

1. Knowledge on Software
2. Concept of Program

**Detailed content of the Lecture:**

Types of Computer Language

- Computer language is defined as code or syntax which is used to write programs or any specific applications
- The computer language is used to communicate with computers
- Three categories assembly language, machine language, and high-level language

1. Machine Language

- The Machine language is considered a low-level language
- Other name -machine code or object code
- Which is set of binary digits 0 and 1
- These binary digits are understood and read by a computer system
- Example of machine language for the text "Hello World".

01001000 0110101 01101100 01101100 01101111 00100000 01010111 01101111 01110010  
01101100 01100100

2. Assembly Language

- Intermediate-level language for microprocessors
- It is second-generation language

3. High-Level Language

- The high-level language is easy to understand and human-readable program
- Examples: C++, C, JAVA, FORTRAN, etc..

## Algorithm :

- An algorithm in general is a sequence of steps to solve a particular problem
- Algorithms are universal
- The algorithm you use in C programming language is also the same algorithm you use in every other language

## Qualities of a good algorithm

1. Input and output should be defined precisely
2. Each steps in algorithm should be clear and unambiguous
3. Algorithm should be most effective among many different ways to solve a problem
4. An algorithm shouldn't have computer code
5. Instead, the algorithm should be written in such a way that, it can be used in similar programming languages

## Examples of Algorithms In Programming

Write an algorithm to add two numbers entered by user

Step 1: Start

Step 2: Declare variables num1, num2 and sum.

Step 3: Read values num1 and num2.

Step 4: Add num1 and num2 and assign the result to sum  
 $sum \leftarrow num1 + num2$


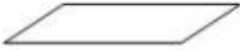





Step 5: Display sum

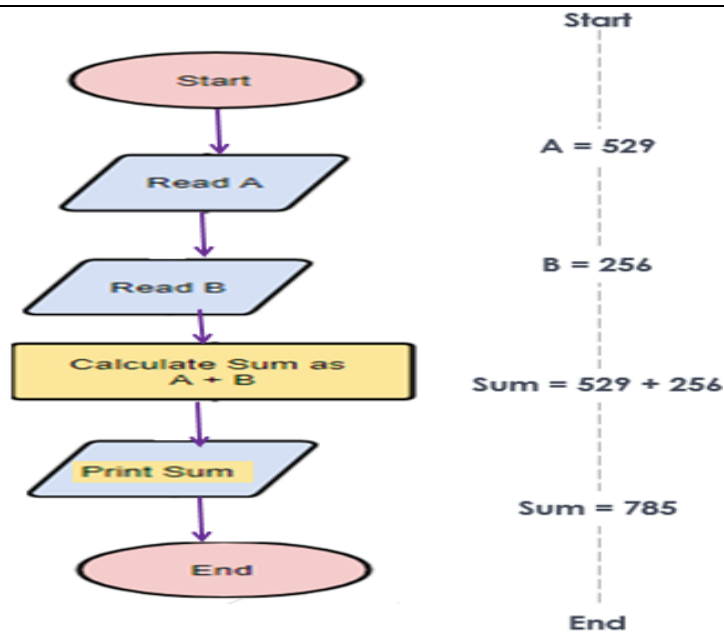
Step 6: Stop

## Flowchart

- A flowchart is a type of diagram that represents a workflow or process
- A flowchart can also be defined as a diagrammatic representation of an algorithm, a step-by-step approach to solving a task.
- The flowchart shows the steps as boxes of various kinds, and their order by connecting the boxes with arrows

Some of the symbols used for flowchart and Example is shown below:

Symbol	Name	Function
	Process	Indicates any type of internal operation inside the Processor or Memory
	input/output	Used for any Input / Output (I/O) operation. Indicates that the computer is to obtain data or output results
	Decision	Used to ask a question that can be answered in a binary format (Yes/No, True/False)
	Connector	Allows the flowchart to be drawn without intersecting lines or without a reverse flow.
	Predefined Process	Used to invoke a subroutine or an Interrupt program.
	Terminal	Indicates the starting or ending of the program, process, or interrupt program
	Flow Lines	Shows direction of flow.



### Pseudocode

- It cannot be compiled or run like a regular program.
- Pseudocode can be written how you want.
- Syntax is a set of rules on how to use and organize statements in a programming language

### Advantages of Pseudocode

- Improves the readability of any approach
- It's one of the best approaches to start implementation of an algorithm
- Acts as a bridge between the program and the algorithm or flowchart
- Also works as a rough documentation, so the program of one developer can be understood easily when a pseudo code is written out
- In industries, the approach of documentation is essential, and that's where a pseudo-code proves vital
- The main goal of a pseudo code is to explain what exactly each line of a program should do, hence making the code construction phase easier for the programmer

### Examples of Pseudocode

```

If student's grade is greater than or equal to 60
    Print "passed"
else
    Print "failed"
  
```

### **Video Content / Details of website for further learning (if any):**

<https://www.youtube.com/watch?v=NXQOekzLwFA>

<https://www.edrawsoft.com/flowchart/program-flowchart-definition.html>

<https://www.edureka.co/blog/introduction-to-c-programming-algorithms/>

**Important Books/Journals for further learning including the page nos.:** Computer Fundamentals and Programming in C - Reema Thareja: Oxford University Press, Second Edition.pp 429-431

Course Faculty

Verified by HOD







# MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



## LECTURE HANDOUTS

L-3

CSE

I / I

Course Name with Code : Programming For Problem Solving Using C - 19GES01

Course Faculty : M.Ganthimathi

Unit : I - Introduction to C Programming Date of Lecture:

**Topic of Lecture:** Structure of a C program, Writing the first C program

### Introduction :

It represents the way how C Program is written and what is the part present in its structure.

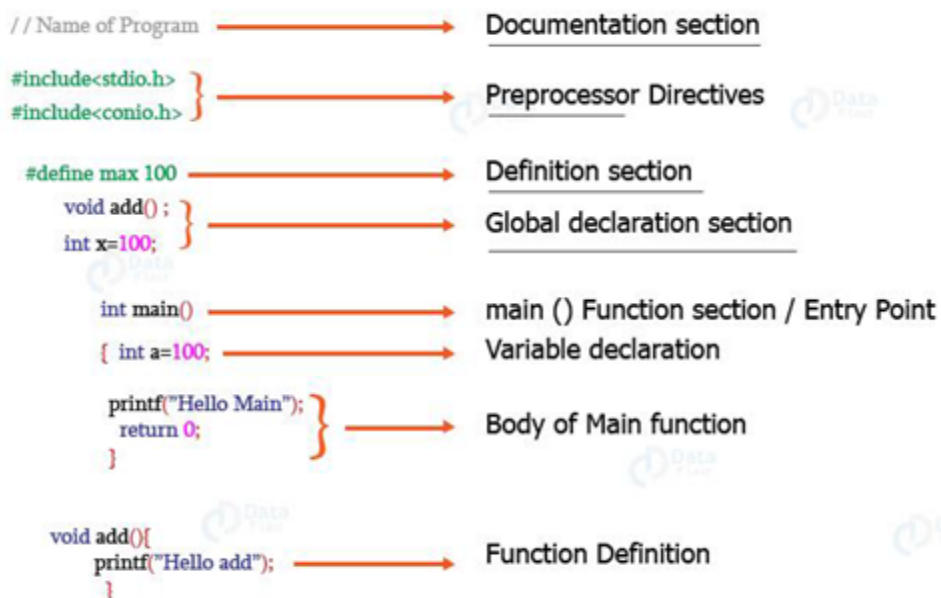
### Prerequisite knowledge for Complete understanding and learning of Topic:

- Program Design Tools
- Basic concept of Program software

### Detailed content of the Lecture:

Structure of a C program :C program basically consists of the following parts

- Comments
- Preprocessor Commands
- Functions
- Variables
- Statements & Expressions



### Documentation section:

- This is a comment block, used by others to understand the code. and
- It is ignored by the compiler.
- Comment can be used anywhere in the program to add info about the program or code block,

- Which will be helpful for developers to understand the existing code in the future easily?

// Comment - single Comment line

/\* Comment \*/ - multiple comment line

#### Preprocessor command

- The first line of the program  
#include <stdio.h>
- is a preprocessor command, which tells a C compiler to include stdio.h file before going to actual compilation

#### Definition section

#define

- variable value –used to define values for the variables globally.

#### Main function-

- main() is the main function where the program execution begins.
- It consists of Declaration part & Execution part / Body of main function

#### Declaration part

- Is used to declare all variables that will be used within the program.

#### Execution part / Body of main function

- There needs to be at least one statement in the executable part, and these two parts are declared within the opening and closing curly braces of the main().

#### Function

- The sub-program section deals with all user-defined functions that are called from the main()
- These user-defined functions are declared and usually defined after the main() function  
printf(...)
- is another function available in C which causes the message to be displayed on the screen

#### Writing the first C program:

```
#include <stdio.h> // Preprocessor command
int main()          // main function
{
    /* Our first simple C basic program */ -// Comment line
    printf("Hello World! ");
    getch();
    return 0;
}
```

OUTPUT: Hello World!

#### **Video Content / Details of website for further learning (if any):**

<https://www.youtube.com/watch?v=mLXsoqkTiiA>

<https://www.studytonight.com/c/first-c-program.php>

#### **Important Books/Journals for further learning including the page nos.:**

Computer Fundamentals and Programming in C - Reema Thareja: Oxford University Press, Second Edition.pp 14-16

**Course Faculty**

**Verified by HOD**



# MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



## LECTURE HANDOUTS

L-4

CSE

I / I

Course Name with Code : Programming For Problem Solving Using C - 19GES01

Course Faculty : M.Ganthimathi

Unit : I - Introduction to C Programming Date of Lecture:

Topic of Lecture: Keywords, Identifiers, Basic Data Types in C

### Introduction :

Characters are grouped together to form meaningful words and these meaningful words are termed as Tokens. Tokens are broadly categorized into following categories and they are as follows: Keywords, Identifiers, Constants, Operators, Special symbols

### Prerequisite knowledge for Complete understanding and learning of Topic:

- Structure of a C program
- Knowledge on C Tokens

### Detailed content of the Lecture:

#### Keywords in C

- Keywords are predefined, reserved words used in programming
- It have a special meaning
- Keywords are part of the syntax and they cannot be used as an identifier

For example: `int a;` // Here, `int` is a keyword

Keywords in C Programming			
auto	break	case	char
const	continue	default	do
double	else	enum	extern
float	for	goto	if
int	long	register	return
short	signed	sizeof	static
struct	switch	typedef	union
unsigned	void	volatile	while

#### C Identifiers

- Identifier refers to name given to entities such as variables, functions, structures etc.
- Identifiers must be unique
- They are created to give a unique name to an entity to identify it during the execution of the program called variable
- To indicate the storage area, each variable should be given a unique name (identifier)

For example:

```
int a, b; // Here, a and b are identifiers
```

### Rules for naming identifiers

1. A valid identifier can have letters (both uppercase and lowercase letters), digits and underscores
2. The first letter of an identifier should be either a letter or an underscore, it can be followed by letter/digit.
3. You cannot use keywords as identifiers.
4. It should not have space in the middle.

Example

valid (Name, RollNo, A123, CSE\_A)

invalid ( 12Abc, CSE#, Roll No...)

### Data Types in C

- A data type specifies the type of data that a variable can store such as integer
- There are the following data types in C language.

Types	Data Types
Basic DT	int, char, float and double
Derived DT	Array, pointer, structure and union
Enumeration DT	enum
Void DT	void

Data Types	Memory Size	Range
char	1 byte	-128 to 127
signed char	1 byte	-128 to 127
unsigned char	1 byte	0 to 255
short	2 byte	-32,768 to 32,767
signed short	2 byte	-32,768 to 32,767
unsigned short	2 byte	0 to 65,535
int	2 byte	-32,768 to 32,767
signed int	2 byte	-32,768 to 32,767
unsigned int	2 byte	0 to 65,535
<b>short int</b>	2 byte	-32,768 to 32,767
signed short int	2 byte	-32,768 to 32,767
unsigned short int	2 byte	0 to 65,535
<b>long int</b>	4 byte	-2,147,483,648 to 2,147,483,647
signed long int	4 byte	-2,147,483,648 to 2,147,483,647
unsigned long int	4 byte	0 to 4,294,967,295
<b>float</b>	4 byte	1.2E-38 to 3.4E+38
<b>double</b>	8 byte	2.3E-308 to 1.7E+308
<b>long double</b>	10 byte	3.4E-4932 to 1.1E+4932

- int: Integers are whole numbers that can have both zero, positive and negative values but no decimal values. Ex : int a=2;
- float and double: float and double are used to hold real numbers, Ex : float a=2.345;
- char: Keyword char is used for declaring character type variables. Ex: char test = 'h';
- void: void is an incomplete type. It means "nothing" or "no type"

```
void main ( )
```

### Example : Integer Output

```
#include <stdio.h> int main()
{
    int testInteger = 5;
    printf("Number = %d", testInteger); return 0;
}
```

Output: Number = 5

We use %d format specifier to print int types.

Here, the %d inside the quotations will be replaced by the value of testInteger

### Example : float and double Output

```
#include <stdio.h>
int main()
{ float number1 = 13.5;
  double number2 = 12.4;
  printf("number1 = %f\n", number1);
  printf("number2 = %lf", number2);
  return 0; }
```

Output :    number1 = 13.500000  
              number2 = 12.400000

To print float, we use %f format specifier. Similarly, we use %lf to print double values.

### Example 4: Print Characters

```
#include <stdio.h> int main()
{
char chr = 'a';
printf("character = %c.", chr);
return 0;
}
```

Output

character = a

To print char, we use %c format specifier.

### Enumeration constants

- Keyword enum is used to define enumeration data types
- Enumeration is a user defined data type in C. It is mainly used to assign names to integral constants

For example:

```
enum color {yellow, green, black, white};
```

Here, color is a variable and yellow, green, black and white are the enumeration constants having value 0, 1, 2 and 3 respectively.

**An example program to demonstrate working  
// of enum in C**

```
#include<stdio.h>
enum week{Mon, Tue, Wed, Thur, Fri, Sat, Sun};
int main( )
{
enum week day;
day = Wed;
printf("%d",day);
return 0;
}
Output: 2
```

**Video Content / Details of website for further learning (if any):**

<https://intellipaat.com/blog/tutorial/c-tutorial/c-data-types/>  
<https://www.youtube.com/watch?v=HzNmyCPmJvU>  
<https://www.geeksforgeeks.org/variables-and-keywords-in-c/>

**Important Books/Journals for further learning including the page nos.:** Computer Fundamentals and Programming in C - Reema Thareja: Oxford University Press, Second Edition.pp 19-21

**Course Faculty**

**Verified by HOD**



# MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L-5

CSE

I / I

Course Name with Code : Programming For Problem Solving Using C - 19GES01

Course Faculty : M.Ganthimathi

Unit : I - Introduction to C Programming Date of Lecture:

**Topic of Lecture:** Variables, Constants

### **Introduction :**

Characters are grouped together to form meaningful words and these meaningful words are termed as Tokens. Tokens are broadly categorized into following categories and they are as follows: Keywords, Identifiers, Constants, Operators, Special symbols

### **Prerequisite knowledge for Complete understanding and learning of Topic:**

- Structure of a C program
- Knowledge on C Tokens

### **Detailed content of the Lecture:**

#### **Variables in C**

- A variable is a name of the memory location.
- It is used to store data.
- Its value can be changed, and it can be reused many times.
- It is a way to represent memory location through symbol so that it can be easily identified.

The example of declaring the variable is given below:

1. int a;
2. float b;
3. char c; Here, a, b, c are variables. The int, float, char are the data types.

- We can also provide values while declaring the variables as given below:

1. int a=10,b=20;//declaring 2 variable of integer type
2. float f=20.8;
3. char c='A';

#### **Constants**

- Constants refer to fixed values that the program may not alter during its execution.
- These fixed values are also called literals.
- Constants can be of any of the basic data types like an integer constant, a floating constant, a character constant, or a string literal.

we can define constants in two ways as shown below:

1. Using #define preprocessor directive
2. Using a const keyword

#### Using #define preprocessor directive:

- This directive is used to declare an alias name for existing variable or any value.
- declare a constant as shown below

```
#define identifierName value
```

identifierName: It is the name given to constant

value: This refers to any value assigned to identifierName

```
#include<stdio.h>
#define val 10
int main()
{
    printf("Integer Constant: %d\n",val);
    return 0;
}
```

Output: Integer Constant: 10

#### using a const keyword:

- Using const keyword to define constants is as simple as defining variables, the difference is you will have to precede the definition with a const keyword

```
#include <stdio.h>
int main()
{
    const int intVal = 10;                // int constant
    const float floatVal = 4.14;         // Real constant
    const char charVal = 'A';           // char constant
    const char stringVal[10] = "ABC";    // string constant
    printf("Integer constant:%d \n", intVal );
    printf("Floating point constant: %.2f\n", floatVal );
    printf("Character constant: %c\n", charVal );
    printf("String constant: %s\n", stringVal);
    return 0;
}
```

#### **Output:**

Integer constant: 10  
Floating point constant: 4.14  
Character constant: A  
String constant:ABC

#### **Video Content/ Details of website for further learning (if any):**

<https://www.youtube.com/watch?v=bS6uNMmIoQ0>

[https://www.tutorialspoint.com/cprogramming/c\\_constants.htm](https://www.tutorialspoint.com/cprogramming/c_constants.htm)

**Important Books/Journals for further learning including the page nos.:** Computer Fundamentals and Programming in C - Reema Thareja: Oxford University Press, Second Edition.pp 22-24

Course Faculty

Verified by HOD





# MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L-6

CSE

I / I

Course Name with Code : Programming For Problem Solving Using C - 19GES01

Course Faculty : M.Ganthimathi

Unit : I - Introduction to C Programming Date of Lecture:

**Topic of Lecture:** Input / Output Statements in C

**Introduction :**

C programming language provides many built-in functions to read any given input and to display data on screen when there is a need to output the result.

**Prerequisite knowledge for Complete understanding and learning of Topic:**

- Structure of a C program
- Knowledge on functions

**Detailed content of the Lecture:**

Input Output (I/O) Statements:

- scanf() -Commonly used function to take input from the user
- scanf - is used when we enter data by using an input device  
Syntax: scanf ("format string", &arg1, &arg2, .....);

C Output

- printf() - is one of the main output function
- This function is used for displaying the output on the screen i.e the data is moved from the computer memory to the output device  
Syntax: printf("format string", arg1, arg2, .....);

Example :

```
#include<stdio.h>
void main()
{
    int a,b,c;
    printf("Enter any two numbers: \n");
    scanf("%d %d", &a, &b);
    c = a + b;
    printf("The addition of two number is: %d", c);
}
```

Enter any two numbers:

12

3

The addition of two number is:15

Format Specifiers for I/O

As you can see from the above examples, we use

- %d for int
- %f for float
- %lf for double
- %c for char

Here's a list of commonly used C data types and their format specifiers.

Data Type	Format Specifier
int	%d
char	%c
float	%f
double	%lf
short int	%hd
unsigned int	%u
long int	%li
long long int	%lli
unsigned long int	%lu
unsigned long long int	%llu
signed char	%c
unsigned char	%c

**Video Content / Details of website for further learning (if any):**

<https://www.studytonight.com/c/c-input-output-function.php>

[https://www.slideshare.net/rulza\\_9621/basic-input-and-output](https://www.slideshare.net/rulza_9621/basic-input-and-output)

<https://www.youtube.com/watch?v=66mubvPdaQA>

**Important Books/Journals for further learning including the page nos.:** Computer Fundamentals and Programming in C - Reema Thareja: Oxford University Press, Second Edition.pp 24-31

**Course Faculty**

**Verified by HOD**



# MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



## LECTURE HANDOUTS

L-7

CSE

I / I

Course Name with Code : Programming For Problem Solving Using C - 19GES01

Course Faculty : M.Ganthimathi

Unit : I - Introduction to C Programming Date of Lecture:

**Topic of Lecture:** Operators in C-Arithmetic, Relational

**Introduction :**  
Operators can be defined as basic symbols that help us work on logical and mathematical operations

**Prerequisite knowledge for Complete understanding and learning of Topic:**

- Structure of a C program
- Knowledge about operator

**Detailed content of the Lecture:**  
Operators in C

- C programming has various operators to perform tasks including arithmetic, conditional and bitwise operations.
- Operands are variables or expressions which are used in operators to evaluate the expression.
- Combination of operands and operators form an Expression.
- For instance  $a = b + c$ ; denote an expression in which there are 3 operands a, b, c and two operator + and =.
- The association of expressions and keywords is called Statements.

For instance `int a = b + c;` denote a statement..

Types of Operators	Description
Arithmetic_operators + - * / %	These are used to perform mathematical calculations like addition, subtraction, multiplication, division and modulus
Assignment_operators = += -= *= /= %=	These are used to assign the values for the variables in C programs.
Relational operators < > <= >= == !=	These operators are used to compare the value of two variables.
Logical operators &&    !	These operators are used to perform logical operations on the given two variables.

<p>Bit wise operators &amp;   ^ ~ &lt;&lt; &gt;&gt;</p>	<p>These operators are used to perform bit operations on given two variables.</p>
<p>Conditional (ternary) operators ?:</p>	<p>Conditional operators return one value if condition is true and returns another value if condition is false.</p>
<p>Increment/decrement operators ++ --</p>	<p>These operators are used to either increase or decrease the value of the variable by one.</p>
<p>Special operators &amp; * sizeof</p>	<p>&amp;, *, sizeof() and ternary operators.</p>

### Arithmetic Operators :

**Assume variable A holds 10 and variable B holds 20**

Operator	Description	Example
+	Adds two operands.	A + B = 30
-	Subtracts second operand from the first.	A - B = -10
*	Multiplies both operands.	A * B = 200
/	Divides numerator by de-numerator.	B / A = 2
%	Modulus Operator and remainder of after an integer division.	B % A = 0
++	Increment operator increases the integer value by one.	A++ = 11
--	Decrement operator decreases the integer value by one.	B--=19

```
#include <stdio.h>
main()
{
    int a = 9,b = 4, c;
    c = a+b;    printf("a+b = %d \n",c);
    c = a-b;    printf("a-b = %d \n",c);
    c = a*b;    printf("a*b = %d \n",c);
    c = a/b;    printf("a/b = %d \n",c);
    c = a%b;    printf("Remainder when a divided by b = %d \n",c);
}
```

#### **Output :**

a+b = 13  
a-b = 5  
a\*b = 36  
a/b = 2  
Remainder when a divided by b=1

Relational Operators:Assume variable A holds 10 and variable B holds 20

Operator	Description	Example
==	Checks if the values of two operands are equal or not. If yes, then the condition becomes true.	(A == B) is not true.
!=	Checks if the values of two operands are equal or not. If the values are not equal, then the condition becomes true.	(A != B) is true.
>	Checks if the value of left operand is greater than the value of right operand. If yes, then the condition becomes true.	(A > B) is not true.
<	Checks if the value of left operand is less than the value of right operand. If yes, then the condition becomes true.	(A < B) is true.
>=	Checks if the value of left operand is greater than or equal to the value of right operand. If yes, then the condition becomes true.	(A >= B) is not true.
<=	Checks if the value of left operand is less than or equal to the value of right operand. If yes, then the condition becomes true.	(A <= B) is not true.

```
#include <stdio.h>
int main()
{
    int a = 5, b = 5, c = 10;
    printf("%d == %d is %d \n", a, b, a == b);
    printf("%d > %d is %d \n", a, b, a > b);
    printf("%d < %d is %d \n", a, b, a < c);
    printf("%d != %d is %d \n", a, b, a != c);
    printf("%d >= %d is %d \n", a, b, a >= b);
    printf("%d <= %d is %d \n", a, b, a <= c);
}
```

**Output:**

```
5 == 5 is 1
5 > 5 is 0
5 < 10 is 1
5 != 10 is 1
5 >= 5 is 1
5 <= 10 is 1
```

Bit Wise Operators in C:

- These operators are used to perform bit operations.
- Decimal values are converted into binary values which are the sequence of bits and bit wise operators work on these bits.
- Bit wise operators in C language are & (bitwise AND), | (bitwise OR), ~ (bitwise NOT), ^ (XOR), << (left shift) and >> (right shift).

Consider x=40 and y=80. Binary form of these values are given below.

x = 00101000 (40)

y= 01010000 (80)

All bit wise operations for x and y are given below.

1. x&y = 00000000 (binary) = 0 (decimal)

2.  $x|y = 01111000$  (binary) = 120 (decimal)
3.  $\sim x = 11010111 = -41$  (decimal)
4.  $x^y = 01111000$  (binary) = 120 (decimal)
5.  $x \ll 1 = 01010000$  (binary) = 80 (decimal)
6.  $x \gg 1 = 00010100$  (binary) = 20 (decimal)

**Video Content/ Details of website for further learning (if any):**

<https://www.youtube.com/watch?v=WGQRInmOBM8>

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

**Important Books/Journals for further learning including the page nos.:** Computer Fundamentals and Programming in C - Reema Thareja: Oxford University Press, Second Edition.pp 32-35

**Course Faculty**

**Verified by HOD**



# MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



## LECTURE HANDOUTS

L-8

CSE

I / I

Course Name with Code : Programming For Problem Solving Using C - 19GES01

Course Faculty : M.Ganthimathi

Unit : I - Introduction to C Programming Date of Lecture:

Topic of Lecture: Logical, Conditional

### Introduction :

Operators can be defined as basic symbols that help us work on logical and mathematical operations

### Prerequisite knowledge for Complete understanding and learning of Topic:

- Structure of a C program
- Knowledge about operator

### Detailed content of the Lecture:

#### Logical Operators :

- variable A holds 1 and variable B holds 0, then

Operator	Description	Example
&&	Called Logical AND operator. If both the operands are non-zero, then the condition becomes true.	(A && B) is false.
	Called Logical OR Operator. If any of the two operands is non-zero, then the condition becomes true.	(A    B) is true.
!	Called Logical NOT Operator. It is used to reverse the logical state of its operand. If a condition is true, then Logical NOT operator will make it false.	! A =0

```
#include <stdio.h>
int main()
{
    int a = 5, b = 5, c = 10, result;
    result = (a == b) && (c > b);
    printf("(a == b) && (c > b) is %d \n", result);
    result = (a == b) || (c < b);
    printf("(a == b) || (c < b) is %d \n", result);
    result = !(a != b);
    printf("(a == b) is %d \n", result);
}
```

(a == b) && (c > b) is 1  
(a == b) || (c < b) is 1  
!(a != b) is 1  
!(a == b) is 0

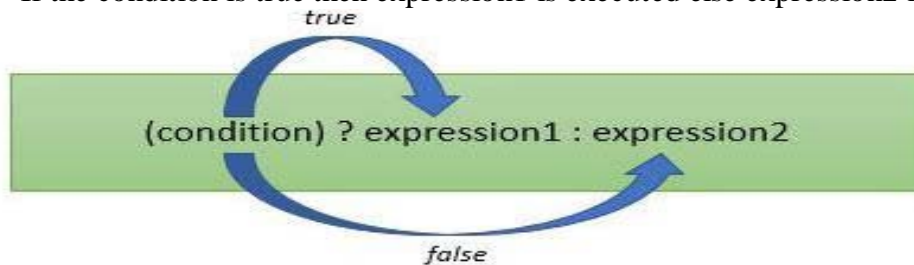
#### Conditional operator:

- Conditional operators return one value if condition is true and returns another value if condition is false.
- This operator is also called as ternary operator.

Syntax : (Condition? true\_value: false\_value);

Example : (A > 100 ? 0 : 1);

If the condition is true then expression1 is executed else expression2 is executed



```
#include <stdio.h>
main()
{
    int mark;
    printf("Enter mark: ");    scanf("%d", &mark);
    puts(mark >= 40 ? "Passed" : "Failed");
}
```

Enter mark: 39 Failed

#### **Video Content / Details of website for further learning (if any):**

<https://www.geeksforgeeks.org/conditional-or-ternary-operator-in-c-c/>

<http://www.trytoprogram.com/c-programming/c-conditional-operator/>

#### **Important Books/Journals for further learning including the page nos.:**

Computer Fundamentals and Programming in C - Reema Thareja: Oxford University Press, Second Edition. pp 35-45

**Course Faculty**

**Verified by HOD**





# MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L-9

CSE

I / I

Course Name with Code : Programming For Problem Solving Using C - 19GES01

Course Faculty : M.Ganthimathi

Unit : I - Introduction to C Programming Date of Lecture:

**Topic of Lecture:** Type conversion and Typecasting

**Introduction :**

In type casting, a data type is converted into another data type by a programmer using casting operator

**Prerequisite knowledge for Complete understanding and learning of Topic:**

- Knowledge about data type
- Precedence of data type

**Detailed content of the Lecture:**

- The type conversion process in C is basically converting one type of datatype to other
- The conversion is done only between those datatypes wherein the conversion is possible  
ex – char to int (lowest data type to highest data type)

Types of conversion:

1. Implicit Type Conversion
2. Explicit Type Conversion

Implicit Type Conversion

- This type of conversion is usually performed by the compiler when necessary without any commands by the user.
- Thus it is also called "Automatic Type Conversion".
- The compiler usually performs this type of conversion when a particular expression contains more than one data type.

Rules...

if operand is :

1. char + int = int
2. Float+ double float =double float
3. Float+int=float

**Example 1:**

```
int a = 20;
```

```
double b = 20.5; a + b;
```

Here, first operand is int type and other is double. So, converted to double.

Therefore, the final answer is double a + b = 40.500000.

**Example 2:**

```
char ch='a';
int c =13;
a + c;
answer is ch + c = 97 + 13 = 110.
```

**Example 3:**

```
char ch='A';
int a =60; a * b;
int variable, 65 + 60 = 125.
```

**Explicit Type Conversion**

- compiler for converting one data type to another instead the user explicitly defines within the program the datatype of the operands in the expression.

Syntax : (Datatype)variable\_name

**Example:**

```
double da = 4.5;
double db = 4.6;
double dc = 4.9;
int result = (int)da + (int)db + (int)dc;
printf("result = %d", result);
```

Output result is :12

**C - Type Casting**

- Type casting is a way to convert a variable from one data type to another data type

Syntax : (type\_name) expression

```
#include <stdio.h>
main()
{
    int sum = 17, count = 5; double mean;
    mean = (double) sum / count;
    printf("Value of mean : %f\n", mean );
}
Value of mean : 3.400000
```

**Integer promotion**

- If the operands still have different types, then they are converted to the type that appears highest in the following hierarchy

**Video Content / Details of website for further learning (if any):**

<https://www.geeksforgeeks.org/difference-between-type-casting-and-type-conversion/>  
<https://www.youtube.com/watch?v=t3gAwC277a8>

**Important Books/Journals for further learning including the page nos.:**

Computer Fundamentals and Programming in C - Reema Thareja: Oxford University Press, Second Edition.pp 46-48

Course Faculty

Verified by HOD



**MUTHAYAMMAL ENGINEERING COLLEGE**  
(An Autonomous Institution)



(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to  
Anna University)  
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

**LECTURE HANDOUTS**

L-10

**CSE**

**I / I**

**Course Name with Code : Programming For Problem Solving Using C - 19GES01**

**Course Faculty : M.Ganthimathi**

**Unit : II - Conditional and Looping Statements Date of Lecture:**

**Topic of Lecture:** Conditional branching statements- if statement

**Introduction :**

- Conditional branching Statements help to jump from one part to the program to another depending on whether a particular condition is satisfied or not
- There are following types of conditional statements in C.
  - if statement
  - if-Else statement
  - Nested if-else statement
  - if-Else if ladder
  - Switch statement
- **Conditional Statements in C** programming are used to make decisions based on the conditions

**Prerequisite knowledge for Complete understanding and learning of Topic:**

- Basic Concepts of Programming
- Translate the algorithms to programs in C language and execute them if statement

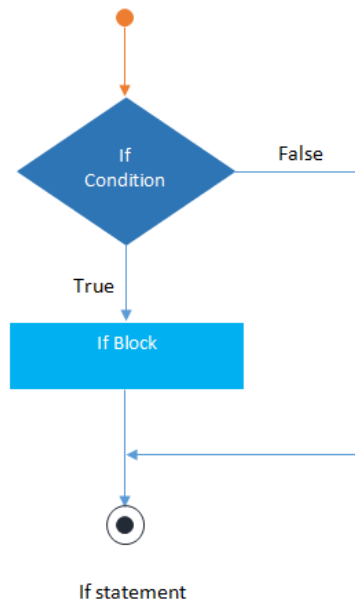
**Detailed content of the Lecture:**

**if statement**

The single if statement in C language is used to execute the code if a condition is true. It is also called one-way selection statement

**Syntax**

```
if(expression)
{
//code to be executed
}
```



### How "if" statement works

- If the expression is evaluated to nonzero (true) then if block statement(s) are executed.
- If the expression is evaluated to zero (false) then Control passes to the next statement following it.

### if Statement Example

```

#include<stdio.h>
#include<conio.h>
void main()
{
int num=0;
printf("enter the number");
scanf("%d",&num);
if(n%2==0)
{
printf("%d number in even",num);
}
getch();
}

```

### Video Content / Details of website for further learning (if any):

[https://www.tutorialspoint.com > c programming](https://www.tutorialspoint.com/c-programming)

<http://www.tutorialspoint.net/2015/12/conditional-branching-statements-in-c.html>

**Important Books/Journals for further learning including the page nos.:** Reema Thareja Computer Fundamentals and Programming in C Second Edition, Oxford University Press 2015.pp.205-207

**Course Faculty**

**Verified by HOD**



**MUTHAYAMMAL ENGINEERING COLLEGE**  
(An Autonomous Institution)



(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

**LECTURE HANDOUTS**

L-11

**CSE**

**I / I**

**Course Name with Code : Programming For Problem Solving Using C - 19GES01**

**Course Faculty : M.Ganthimathi**

**Unit : II - Conditional and Looping Statements Date of Lecture:**

**Topic of Lecture:** if-else statements

**Introduction :**

- The if statement may have an optional else block
- An **if** statement can be followed by an optional **else** statement, which executes when the Boolean expression is false

**Prerequisite knowledge for Complete understanding and learning of Topic:**

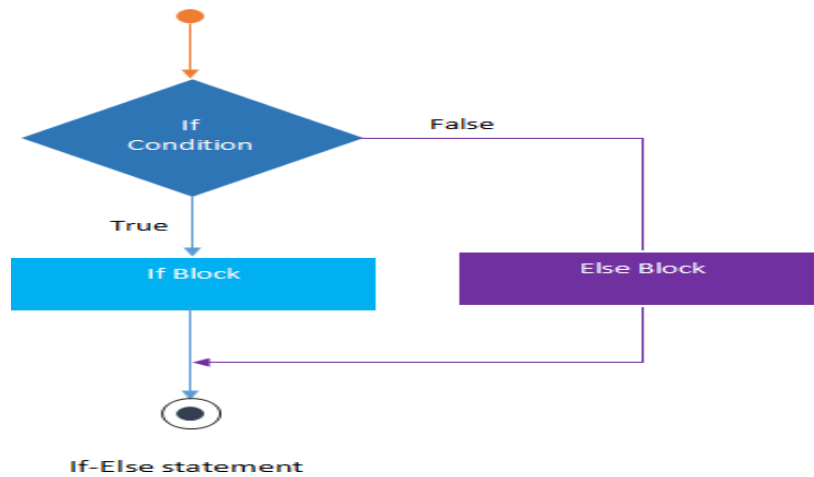
- Basic Concepts of Programming
- Translate the algorithms to programs in C language and execute them if statement

**Detailed content of the Lecture:**

**if-else statements**

**Syntax**

```
if(expression)
{
// Statements
}
else
{
// Statements
}
```



### How "if..else" statement works.

- If the expression is evaluated to nonzero (true) then if block statement(s) are executed.
- If the expression is evaluated to zero (false) then else block statement(s) are executed.

### if..else Statement Example

```

#include<stdio.h>
#include<conio.h>
void main()
{
int num=0;
printf("enter the number");
scanf("%d",&num);
if(n%2==0)
{
printf("%d number in even", num);
}
else
{
printf("%d number in odd",num);
}
getch();
}

```

### Video Content / Details of website for further learning (if any):

[https://www.tutorialspoint.com > c programming](https://www.tutorialspoint.com/c-programming)

[https://www.tutorialspoint.com/cprogramming/if\\_else\\_statement\\_in\\_c.htm](https://www.tutorialspoint.com/cprogramming/if_else_statement_in_c.htm)

**Important Books/Journals for further learning including the page nos.:** Reema Thareja Computer Fundamentals and Programming in C Second Edition, Oxford University Press 2015.pp.212-213

**Course Faculty**

**Verified by HOD**



**MUTHAYAMMAL ENGINEERING COLLEGE**  
(An Autonomous Institution)



(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to  
Anna University)  
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

**LECTURE HANDOUTS**

L-12

**CSE**

**I / I**

**Course Name with Code : Programming For Problem Solving Using C - 19GES01**

**Course Faculty : M.Ganthimathi**

**Unit : II - Conditional and Looping Statements Date of Lecture:**

**Topic of Lecture:** if-else-if statements

**Introduction :**

- The if-else-if statement is used to execute one code from multiple conditions
- It is also called multipath decision statement
- It is a chain of if..else statements in which each if statement is associated with else if statement and last would be an else statement

**Prerequisite knowledge for Complete understanding and learning of Topic:**

- Basic Concepts of Programming
- Translate the algorithms to programs in C language and execute them if statement if-else Statements

**Detailed content of the Lecture:**

**Syntax**

```
if(condition1)
{
//statements
}
else if(condition2)
{
//statements
}
else if(condition3)
{
//statements
}
else
{
//statements
}
```

**Example**

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a;
printf("enter a number");
scanf("%d",&a);
if( a%5==0 && a%8==0)
{
printf("divisible by both 5 and 8");
}
else if( a%8==0 )
{
printf("divisible by 8");
}
else if(a%5==0)
{
printf("divisible by 5");
}
else
{
printf("divisible by none");
}
getch();
}
```

**Video Content / Details of website for further learning (if any):**

[https://www.tutorialspoint.com > c programming](https://www.tutorialspoint.com/c-programming)

[https://www.tutorialspoint.com/cprogramming/if\\_else\\_if-statement\\_in\\_c.htm](https://www.tutorialspoint.com/cprogramming/if_else_if-statement_in_c.htm)

**Important Books/Journals for further learning including the page nos.:** Reema Thareja Computer Fundamentals and Programming in C Second Edition, Oxford University Press 2015.pp.216-218

**Course Faculty**

**Verified by HOD**





Course Name with Code : Programming For Problem Solving Using C - 19GES01

Course Faculty : M.Ganthimathi

Unit : II - Conditional and Looping Statements Date of Lecture:

**Topic of Lecture:** switch statements

**Introduction :**

- Switch statement is a control statement that allows us to choose only one choice among the many given choices
- The expression in switch evaluates to return an integral value, which is then compared to the values present in different case.
- It executes that block of code which matches the case value. If there is no match, then default block is executed(if present)

**Prerequisite knowledge for Complete understanding and learning of Topic:**

- Basic Concepts of Programming
- Translate the algorithms to programs in C language and execute them if statement
- if-else Statements
- if-else-if Statements

**Detailed content of the Lecture:**

**switch statements**

- switch statement acts as a substitute for a long if-else-if ladder that is used to test a list of cases.
- A switch statement contains one or more case labels which are tested against the switch expression.
- When the expression match to a case then the associated statements with that case would be executed.

**Syntax**

**Switch (expression)**

```
{
case value1:
//Statements
break;
case value 2:
//Statements
break;
case value 3:
//Statements
case value n:
//Statements
break;
Default:
//Statements
}
```

**Switch Statement Example**

```
#include <stdio.h>
int main()
{
int x = 2;
switch (x)
{
case 1: printf("Choice is 1");
break;
case 2: printf("Choice is 2");
break;
case 3: printf("Choice is 3");
break;
default: printf("Choice other than 1, 2 and 3");
break;
}
return 0;
}
```

**Output:**

Choice is 2

**Video Content / Details of website for further learning (if any):**

[https://www.tutorialspoint.com > c programming](https://www.tutorialspoint.com/c-programming)

[https://www.tutorialspoint.com/cprogramming/switch-statement\\_in\\_c.htm](https://www.tutorialspoint.com/cprogramming/switch-statement_in_c.htm)

**Important Books/Journals for further learning including the page nos.:** Reema Thareja Programming in C Second Edition, Oxford University Press 2015.pp.221-224

**Course Faculty**

**Verified by HOD**



MUTHAYAMMAL ENGINEERING COLLEGE  
(An Autonomous Institution)



(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to  
Anna University)  
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

LECTURE HANDOUTS

L-14

CSE

I/I

Course Name with Code : Programming For Problem Solving Using C - 19GES01

Course Faculty : M.Ganthimathi

Unit : II - Conditional and Looping Statements Date of Lecture:

**Topic of Lecture:** Iterative statements -while statement

**Introduction :**

- Looping statement are the statements execute one or more statement repeatedly several number of times
- In C programming language there are three types of loops; while, for and do-while
- When you need to execute a block of code several number of times then you need to use looping concept in C language

**Prerequisite knowledge for Complete understanding and learning of Topic:**

- Basic Concepts of Programming
- if-else Statements
- **Switch Statements**

**Detailed content of the Lecture:**

Iterative statements -while statement

- In **While Loop in C** First check the condition if condition is true then control goes inside the loop body otherwise goes outside the body.
- **while loop** will be repeats in clock wise direction.

**Syntax**

```
Assignment;  
while(condition)  
{  
Statements;  
.....  
Increment/decrements (++ or --);  
}
```

**Example of while loop**

```
#include<stdio.h>
#include<conio.h>
void main()
{
int i; clrscr(); i=1;
while(i<5)
{
printf("\n%d",i); i++;
}
getch();
}
```

**Output**

```
1
2
3
4
```

**Video Content / Details of website for further learning (if any):**

[https://www.tutorialspoint.com > c programming](https://www.tutorialspoint.com/c-programming)

<https://www.tutorialspoint.com/cprogramming/while-loop-statement-in-c.htm>

**Important Books/Journals for further learning including the page nos.:** Reema Thareja Computer Fundamentals and Programming in C Second Edition, Oxford University Press 2015.pp. 232-233

**Course Faculty**

**Verified by HOD**



Course Name with Code : Programming For Problem Solving Using C - 19GES01

Course Faculty : M.Ganthimathi

Unit : II - Conditional and Looping Statements Date of Lecture:

**Topic of Lecture:**do-while statement

**Introduction :**

- A **do-while Loop in C** is similar to a while loop, except that a do-while loop is executed at least one time
- A do while loop is a control flow statement that executes a block of code at least once, and then repeatedly executes the block, or not, depending on a given condition at the end of the block (in while)

**Prerequisite knowledge for Complete understanding and learning of Topic:**

- Basic concepts of programming
- If Statements
- While Loop

**Detailed content of the Lecture:**

**Syntax**

```
do
{
Statements;
.....
Increment/decrement (++ or --)
} while();
```

**Example of do..while loop** #include<stdio.h> #include<conio.h>

```
void main()
{
int i; clrscr(); i=1;
do
{
printf("\n%d",i); i++;
}
while(i<5); getch();
}
```

**Output**

1  
2  
3  
4

**Video Content / Details of website for further learning (if any):**

[https://www.tutorialspoint.com > c programming](https://www.tutorialspoint.com/c-programming)

[https: <https://www.tutorialspoint.com/cprogramming/do-while-loop-statement-in-c.htm>](https://www.tutorialspoint.com/cprogramming/do-while-loop-statement-in-c.htm)

**Important Books/Journals for further learning including the page nos.:** Reema Thareja Computer Fundamentals and Programming in C Second Edition, Oxford University Press 2015.pp. 225-227

**Course Faculty**

**Verified by HOD**



LECTURE HANDOUTS

L -16

CSE

I/I

Course Name with Code : Programming For Problem Solving Using C - 19GES01

Course Faculty : M.Ganthimathi

Unit : II - Conditional and Looping Statements Date of Lecture:

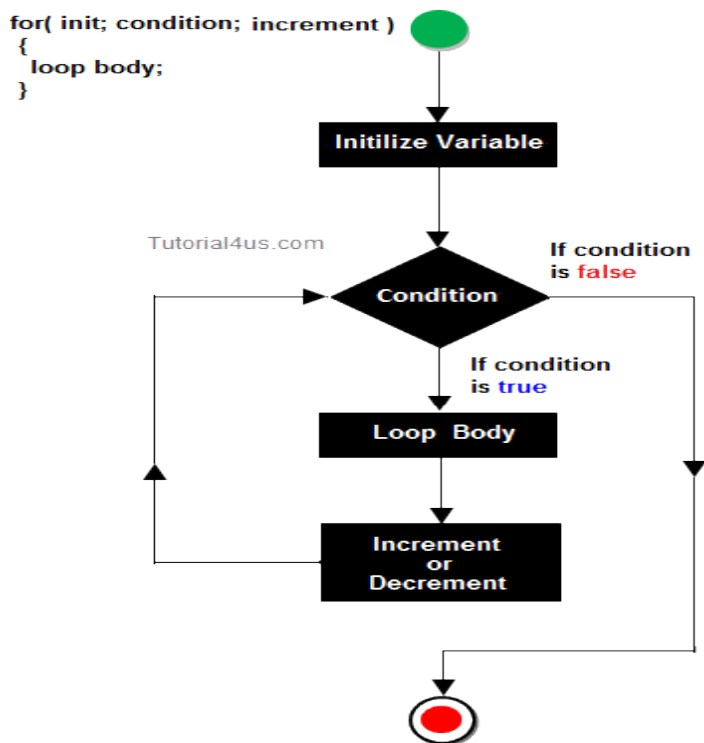
Topic of Lecture: for loop statements

Introduction :  
• Iteration statements or loop statements allow us to execute a block of statements as long as the condition is true.  
• Loops statements are used when we need to run same code again and again, each time with a different value

Prerequisite knowledge for Complete understanding and learning of Topic:  
• Basic concepts of Programming  
• While loop  
• Do-while loop

Detailed content of the Lecture:

For Loop in C is a statement which allows code to be repeatedly executed. For loop 3 arts Initialization, Condition and Increment or Decrements.



**Example of for loop**

```
#include<stdio.h>
#include<conio.h>
void main()
{
int i; clrscr();
for(i=1;i<5;i++)
{
printf("\n%d",i);
}
getch();
}
```

**Output**

```
1
2
3
4
```

**Video Content / Details of website for further learning (if any):**

[https://www.tutorialspoint.com > c programming](https://www.tutorialspoint.com/c-programming)

[https://www.tutorialspoint.com/cprogramming/fore-loop-statement\\_in\\_c.htm](https://www.tutorialspoint.com/cprogramming/fore-loop-statement_in_c.htm)

**Important Books/Journals for further learning including the page nos.:** Reema Thareja Computer Fundamentals and Programming in C Second Edition, Oxford University Press 2015.pp. 227-228

**Course Faculty**

**Verified by HOD**





MUTHAYAMMAL ENGINEERING COLLEGE  
(An Autonomous Institution)



(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to  
Anna University)  
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

LECTURE HANDOUTS

L-17

CSE

I/I

Course Name with Code : Programming For Problem Solving Using C - 19GES01

Course Faculty : M.Ganthimathi

Unit : II - Conditional and Looping Statements Date of Lecture:

**Topic of Lecture:** Nested Loop

**Introduction :**

- In Nested loop one loop is place within another loop body. when we need to repeated loop body itself n number of times use nested loops
- Nested loops can be design up to 255 blocks

**Prerequisite knowledge for Complete understanding and learning of Topic:**

- Basic concepts of Programming
- While loop
- Do-while loop

**Detailed content of the Lecture:**

Nested loop one loop is place within another loop body. when we need to repeated loop body itself n number of times use nested loops

```
for (initialization; condition; increment/ decrement)
{
    statement(s);
    for (initialization; condition; increment/ decrement)
    {
        statement(s);
        ....
    }
    ....
}
```

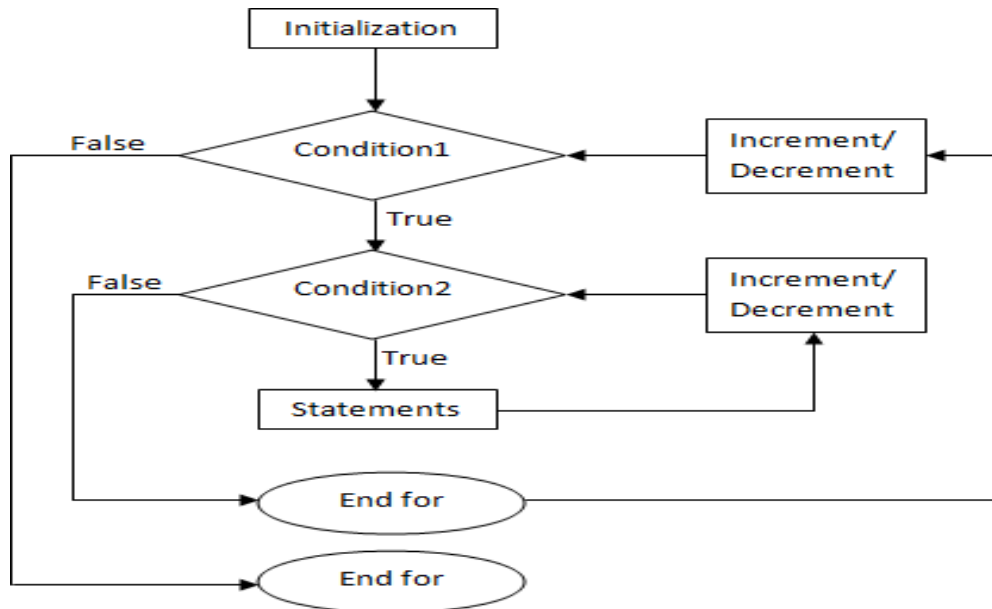


Fig: Flowchart for nested for loop

### EXAMPLE PROGRAM:

```

#include <stdio.h> int main()
{
for (int i=0; i<2; i++)
{
for (int j=0; j<4; j++)
{
printf("%d, %d\n",i ,j);
}
}
return 0;
}
  
```

### OUTPUT:

```

0, 0
0, 1
0, 2
0, 3
1, 0
1, 1
1, 2
1, 3
  
```

### Video Content / Details of website for further learning (if any):

[https://www.tutorialspoint.com > c programming](https://www.tutorialspoint.com/c-programming)

[https://www.tutorialspoint.com/cprogramming/nested-loop-statement\\_in\\_c.htm](https://www.tutorialspoint.com/cprogramming/nested-loop-statement_in_c.htm)

**Important Books/Journals for further learning including the page nos.:** Reema Thareja Computer Fundamentals and Programming in C Second Edition, Oxford University Press 2015.pp. 228-229

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE  
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to  
Anna University)  
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L-18

CSE

I/I

Course Name with Code : Programming For Problem Solving Using C - 19GES01

Course Faculty : M.Ganthimathi

Unit : II - Conditional and Looping Statements Date of Lecture:

**Topic of Lecture:** break and continue statements

**Introduction :**

- In any loop break is used to jump out of loop skipping the code below it without caring about the test condition
- It interrupts the flow of the program by breaking the loop and continues the execution of code which is outside the loop
- The common use of break statement is in switch case where it is used to skip remaining part of the code

**Prerequisite knowledge for Complete understanding and learning of Topic:**

- Basic concepts of Programming
- While loop
- Do-while loop
- For loop

**Detailed content of the Lecture:**

**In while loop**

```
while (test_condition)
{
statement1;
if (condition )
break;
statement2;
}
```

**In do...while loop**

```
do
{
statement1; if(condition)
break;
statement2;
} while (test_condition);
```

**In for loop**

```
For(int-exp:test-exp:update-exp)
{
statement1; if(condition)
break; statement2;
}
```

In above program, while is an infinite loop which will be repeated forever and there is no exit from the loop. So the program will ask for input repeatedly until the user will input 0. When the user enters zero, the if condition will be true and the compiler will encounter the break statement which will cause the flow of execution to jump out of the loop

## Continue Statement

1. Like a break statement, continue statement is also used with 2.if condition inside the loop to alter the flow of control.

3. When used in while, for or do...while loop, it skips the remaining statements in the body of that loop and performs the next iteration of the loop.

Unlike break statement, continue statement when encountered doesn't terminate the loop, rather interrupts a particular iteration.

### **In while loop**

```
while (test_condition)
{
    statement1;
    if (condition) continue;
    statement2;
}
```

### **In do...while loop**

```
do
{
    statement1; if (condition)
    continue; statement2;
}while (test_condition);
```

### **In for loop**

```
for (int-exp; test-exp; update-exp)
{
    statement1; if (condition)
    continue; statement2;
}
```

In above structures, if test\_condition is true then the continue statement will interrupt the flow of control and block of statement2 will be skipped, however, termination of the loop will be continued.

**Example:** C program to print sum of odd numbers between 0 and 10

```
#include <stdio.h>
int main ()
{
    int a, sum = 0;
    for (a = 0; a < 10; a++)
    {
        if ( a % 2 == 0 ) continue;
        sum = sum + a;
    }
    printf("sum = %d", sum);
    return 0;
}
```

### **Output**

sum = 25

### **Video Content / Details of website for further learning (if any):**

<https://www.tutorialspoint.com/c-programming>

[https://www.tutorialspoint.com/cprogramming/continue-statement\\_in\\_c.htm](https://www.tutorialspoint.com/cprogramming/continue-statement_in_c.htm)

**Important Books/Journals for further learning including the page nos.:** Reema Thareja Computer Fundamentals and Programming in C Second Edition, Oxford University Press 2015.pp.231-232

**Course Faculty**

**Verified by HOD**



**MUTHAYAMMAL ENGINEERING COLLEGE**  
(An Autonomous Institution)



(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to  
Anna University)  
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

L-19

**LECTURE HANDOUTS**

**Course Name with Code : Programming For Problem Solving Using C - 19GES01**

**Course Faculty : M.Ganthimathi**

**Unit : III- Functions and Arrays Date of Lecture:**

**Topic of Lecture: : Function Declaration/Function Prototype**

**Introduction:**

- C enables its programmers to break up a program into segments commonly known as functions, each of which can be written more or less independently of the others.
- Every function in the program is supposed to perform a well-defined task. Therefore, the program code of one function is completely insulated from that of other functions.

**Prerequisite knowledge for Complete understanding and learning of Topic:**

- Syntax of function
- Arguments

**Detailed content of the Lecture:**

Every function has a name which acts as an interface to the outside world in terms of how information is transferred to it and how results generated by the function are transmitted back from it.

Dividing the program into separate well-defined functions facilitates each function to be written and tested separately. This simplifies the process of getting the total program to work.

Understanding, coding and testing multiple separate functions are far easier than doing the same for one huge function. If a big program has to be developed without the use of any function (except main ()), then there will be countless lines in the main (). All the libraries in C contain a set of functions that the programmers are free to use in their programs.

These functions have been prewritten and pre-tested, so the programmers use them without worrying about their code details. This speed up program development. A function, f that uses another function g, is known as the calling function and g is known as the called function. The inputs that the function takes are known as arguments. When a called function returns some result back to the calling function, it is said to return that result. The calling function may or may not pass parameters to the called

function. If the called function accepts arguments, the calling function will pass parameters, else not. main() is the function that is called by the operating system and therefore, it is supposed to return the result of its processing to the operating system. Function Declaration/Function Prototype. Function declaration is a declaration statement that identifies a function with its name, a list of arguments that it accepts and the type of data it returns.

- The general format for declaring a function that accepts some arguments and returns some value as result can be given as: `return_data_type function_name(data_type variable1,data_type variable2,..)`; No function can be declared within the body of another function. Example, `float avg ( int a, int b)`;

Function definition consists of a function header that identifies the function, followed by the body of the function containing the executable code for that function. When a function is defined, space is allocated for that function in the memory.

- The syntax of a function definition can be given as:

```
return_data_type function_name(data_type variable1, data_type variable2,..)
```

```
{ ..... statements ..... return( variable); }
```

- The no. and the order of arguments in the function header must be same as that given in function declaration statement.

**Video Content / Details of website for further learning (if any):**

<https://www.youtube.com/watch?v=GY6Q2f2kvY0>

<https://www.khanacademy.org/math/cc-eighth-grade-math/cc-8th-linear-equations-functions/cc-8th-function-intro/v/relations-and-functions>

**Important Books/Journals for further learning including the page nos.:**

Reema Thareja, Computer Fundamentals and Programming in C, Oxford University Press, Second Edition

John V Guttag Introduction to Computation and Programming Using Python Revised and expanded Edition, MIT Press, 2013 (Page No: 123-127)

Reema Thareja, Programming in C, Oxford University Press, Second Edition (Page No: 141-147)

**Course Faculty**

**Verified by HOD**



**MUTHAYAMMAL ENGINEERING COLLEGE**  
(An Autonomous Institution)



(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

**LECTURE HANDOUTS**

L-20

**CSE CSE**

**I/II/I**

Course Name with Code : Programming For Problem Solving Using C - 19GES01

Course Faculty : M.Ganthimathi

Unit : III- Functions and Arrays Date of Lecture:

**Topic of Lecture:**Function definition, Function call, passing parameters to functions

**Introduction:**

- The function call statement invokes the function.
- When a function is invoked the compiler jumps to the called function to execute the statements that are a part of that function.
- Once the called function is executed, the program control passes back to the calling function.
- Function call statement has the following syntax. `function_name(variable1, variable2, ...)`

**Prerequisite knowledge for Complete understanding and learning of Topic:**

- Function Declaration
- Function Prototype

**Detailed content of the Lecture:**

- Function name and the number and type of arguments in the function call must be same as that given in the function declaration and function header of the function definition
- Names (and not the types) of variables in function declaration, function call and header of function definition may vary
- Arguments may be passed in the form of expressions to the called function. In such a case, arguments are first evaluated and converted to the type of formal parameter and then the body of the function gets executed
- If the return type of the function is not void, then the value returned by the called function may be assigned to some variable as given below. `variable_name = function_name(variable1, variable2, ...)`

Example: Program using Functions

```

#include int sum(int a, int b);
int main() { int num1, num2, total = 0;
printf("\n Enter the first number : ");
scanf("%d", &num1);
printf("\n Enter the second number : ");
scanf("%d", &num2); total = sum(num1, num2);
// FUNCTION CALL printf("\n Total = %d", total);
return 0; } // FUNCTION DEFINITION
int sum ( int a, int b) // FUNCTION HEADER
{ // return (a + b); }

```

### Passing Parameters to Functions

- There are two ways in which arguments or parameters can be passed to the called function.
- Call by value in which values of the variables are passed by the calling function to the called function.
- Call by reference in which address of the variables are passed by the calling function to the called function.
- In the Call by Value method, the called function creates new variables to store the value of the arguments passed to it. Therefore, the called function uses a copy of the actual arguments to perform its intended task.
- If the called function is supposed to modify the value of the parameters passed to it, then the change will be reflected only in the called function. In the calling function no change will be made to the value of the variables.

### Example: Program for Functions using Call by Value method

```

#include<stdio.h>
void add( int n);
int main()
{
int num = 2;
printf("\n The value of num before calling the function = %d", num);
add(num);
printf("\n The value of num after calling the function = %d", num);
return 0;
}
void add(int n)
{
n = n + 10;
printf("\n The value of num in the called function = %d", n);
}

```

The output of this program is:

The value of num before calling the function = 2

The value of num in the called function = 20

The value of num after calling the function = 2



## Call by Reference

- When the calling function passes arguments to the called function using call by value method, the only way to return the modified value of the argument to the caller is explicitly using the return statement. The better option when a function can modify the value of the argument is to pass arguments using call by reference technique.
- In call by reference, we declare the function parameters as references rather than normal variables. When this is done any changes made by the function to the arguments it received are visible by the calling program.
- To indicate that an argument is passed using call by reference, an ampersand sign (&) is placed after the type in the parameter list. This way, changes made to that parameter in the called function body will then be reflected in its value in the calling program.

## Example Program for Functions using Call by Reference method

```
#include<stdio.h>
void add( int &n);
int main()
{
int num = 2;
printf("\n The value of num before calling the function = %d", num);
add(num);
printf("\n The value of num after calling the function = %d", num);
return 0;
}
void add( int &n)
{  n = n + 10;
printf("\n The value of num in the called function = %d", n); }
```

### The output of this program is:

The value of num before calling the function = 2

The value of num in the called function = 20

The value of num after calling the function = 20

### Video Content / Details of website for further learning (if any):

<https://www.youtube.com/watch?v=4-xX9vmPDsc>

<https://www.youtube.com/watch?v=CMiuV0WvJjw>

### Important Books/Journals for further learning including the page nos.:

Reema Thareja, Computer Fundamentals and Programming in C, Oxford University Press, Second Edition

John V Guttag Introduction to Computation and Programming Using Python Revised and expanded Edition, MIT Press, 2013(Page No: 151-153)

Reema Thareja, Programming in C, Oxford University Press, Second Edition(Page No: 122-125)

**Course Faculty**

**Verified by HOD**



MUTHAYAMMAL ENGINEERING COLLEGE  
(An Autonomous Institution)



(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to  
Anna University)  
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

LECTURE HANDOUTS

L-21

CSE

I/I

Course Name with Code : Programming For Problem Solving Using C - 19GES01

Course Faculty : M.Ganthimathi

Unit : III- Functions and Arrays Date of Lecture:

**Topic of Lecture:** Declaration of arrays, accessing the elements of an array

**Introduction:**

- An array is a collection of similar data elements.
- These data elements have the same data type.
- The elements of the array are stored in consecutive memory locations and are referenced by an index (also known as the subscript).

**Prerequisite knowledge for Complete understanding and learning of Topic:**

- One Dimensional Array
- Array Syntax

**Detailed content of the Lecture:**

Declaring an array means specifying three things:

The data type- what kind of values it can store ex, int, char, float Name- to identify the array

The size- the maximum number of values that the array can hold

Arrays are declared using the following syntax. type name[size];

Accessing the elements of an Array To access all the elements of the array, you must use a loop. That is, we can access all the elements of the array by varying the value of the subscript into the array.

But note that the subscript must be an integral value or an expression that evaluates to an integral value.

Address of data element,  $A[k] = BA(A) + w(k - \text{lower\_bound})$

Here, A is the array k is the index of the element of which we have to calculate the address BA is the base address of the array A.

w is the word size of one element in memory, for example, size of int is 2. Storing values in arrays

**Step:1**

Initialization of Arrays

Arrays are initialized by writing,

type array\_name[size]={ list of values };

```
int marks[5]={90, 82, 78, 95, 88};
```

**Step:2**

Inputting values to the Arrays

**Step:3**

Assigning the values to the Arrays

Example:

```
#include<stdio.h>
#include<conio.h>
int main()
{
int i=0, n, arr[20];
clrscr();
printf("\n Enter the number of elements : ");
scanf("%d", &n);
for(i=0;i<n;i++)
{
printf("\n Arr[%d] = ", i);
scanf("%d",&arr[i]);
Page53
}
printf("\n The array elements are ");
for(i=0;i<n;i++)
printf("Arr[%d] = %d\t", i, arr[i]);
return 0;
}
```

**Video Content / Details of website for further learning (if any):**

<https://www.youtube.com/watch?v=IRgKavUxvKY>

<https://www.youtube.com/watch?v=00m3YhLT6fA>

**Important Books/Journals for further learning including the page nos.:**

Reema Thareja, Computer Fundamentals and Programming in C, Oxford University Press, Second Edition

John V Guttag Introduction to Computation and Programming Using Python Revised and expanded Edition, MIT Press, 2013 (Page No: 142-145)

Reema Thareja, Programming in C, Oxford University Press, Second Edition(Page No: 152-155)

**Course Faculty**

**Verified by HOD**



MUTHAYAMMAL ENGINEERING COLLEGE  
(An Autonomous Institution)



(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to  
Anna University)  
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

LECTURE HANDOUTS

L-22

CSE

I/I

Course Name with Code : Programming For Problem Solving Using C - 19GES01

Course Faculty : M.Ganthimathi

Unit : III- Functions and Arrays Date of Lecture:

<b>Topic of Lecture:</b> storing values in arrays, operations on 1-d arrays
<b>Introduction:</b> <ul style="list-style-type: none"><li>Initialization of Arrays Arrays are initialized by writing, type array_name[size]={list of values}; int marks[5]={90, 82, 78, 95, 88}</li></ul>
<b>Prerequisite knowledge for Complete understanding and learning of Topic:</b> <ul style="list-style-type: none"><li>Array Syntax</li><li>1-Dimensional Array</li><li>2-Dimensional Array</li></ul>
<b>Detailed content of the Lecture:</b> <p>Step 1: Read the values from array Step:2 Inputting values to the Arrays Step:3 Assigning the values to the Arrays</p> <pre>#include #include int main() { int i=0, n, arr[20]; clrscr(); printf("\n Enter the number of elements : ");  scanf("%d", &amp;n); for(i=0;i&lt;n;i++) { printf("\n Arr[%d] = ", i); scanf("%d",&amp;num[i]); } printf("\n The array elements are ");</pre>

```
for(i=0;i<n;i++)
printf("Arr[%d] = %d\t", i, arr[i]);
return 0; }
```

### **Operations on 1-d arrays – Inserting an Element of an array**

Algorithm to insert a new element to the end of the array.

```
Step 1: Set upper_bound = upper_bound + 1
Step 2: Set A[upper_bound] = VAL
Step 3; EXIT
```

```
Step 1: [INITIALIZATION] SET I = POS
Step 2: Repeat Steps 3 and 4 while I <= N - 1
Step 3:         SET A[I] = A[I + 1]
Step 4:         SET I = I + 1
           [End of Loop]
Step 5: SET N = N - 1
Step 6: EXIT
```

### **Operations on 1-d arrays –Deleting an Element from an array**

Algorithm to delete an element from the end of the array

```
Step 1: Set upper_bound = upper_bound - 1
Step 2: EXIT
```

### **Video Content / Details of website for further learning (if any):**

<https://www.youtube.com/watch?v=HyDE7T6YPsU>

<https://www.youtube.com/watch?v=cscUPrRWnmQ>

<https://www.youtube.com/watch?v=VgONsOtEuZw>

### **Important Books/Journals for further learning including the page nos.:**

Reema Thareja, Computer Fundamentals and Programming in C, Oxford University Press, Second Edition

John V Guttag Introduction to Computation and Programming Using Python Revised and expanded Edition, MIT Press, 2013(Page No: 122-123)

Reema Thareja, Programming in C, Oxford University Press, Second Edition (Page No: 111-121)

Course Faculty

Verified by HOD



LECTURE HANDOUTS

L-23

CSE

I/I

Course Name with Code : Programming For Problem Solving Using C - 19GES01

Course Faculty : M.Ganthimathi

Unit : III- Functions and Arrays Date of Lecture:

**Topic of Lecture:** Inserting an Element of an array

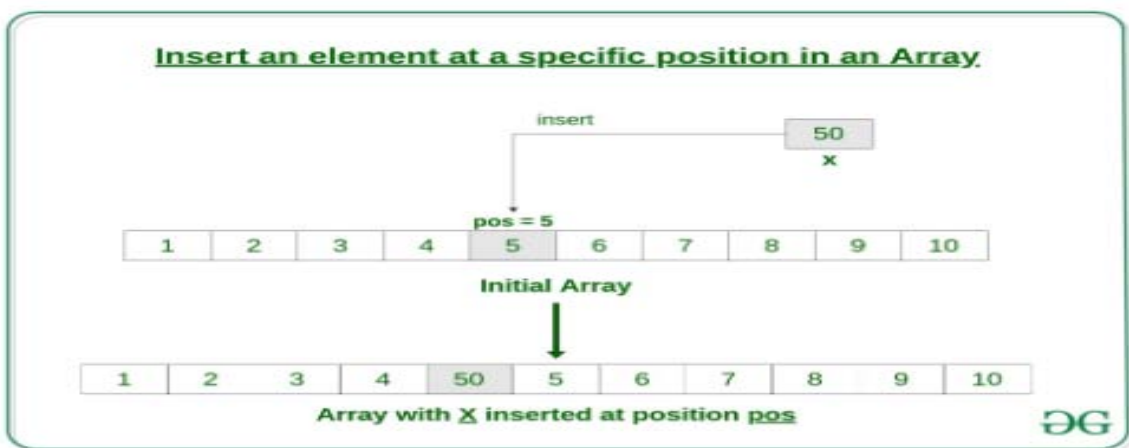
**Introduction:**

- An array is a collection of items stored at contiguous memory locations. In this article, we will see how to insert an element in an array in C.
- Given an array arr of size n, this article tells how to insert an element x in this array arr at a specific position pos.

**Prerequisite knowledge for Complete understanding and learning of Topic:**

- Length of the array
- Mid value find out
- First value and last value

**Detailed content of the Lecture:**



**Here's how to do it.**

First get the element to be inserted, say x

1. Then get the position at which this element is to be inserted, say pos
2. Then shift the array elements from this position to one position forward, and do this for all the other elements next to pos.
3. Insert the element x now at the position pos, as this is now empty.

// C Program to Insert an element

```
// at a specific position in an Array
```

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    int arr[100] = { 0 };
```

```
    int i, x, pos, n = 10;
```

```
    // initial array of size 10
```

```
    for (i = 0; i < 10; i++)
```

```
        arr[i] = i + 1;
```

```
    // print the original array
```

```
    for (i = 0; i < n; i++)
```

```
        printf("%d ", arr[i]);
```

```
    printf("\n");
```

```
    // element to be inserted
```

```
    x = 50;
```

```
    // position at which element
```

```
    // is to be inserted
```

```
    pos = 5;
```

```
    // increase the size by 1
```

```
    n++;
```

```
    // shift elements forward
```

```
    for (i = n-1; i >= pos; i--)
```

```
        arr[i] = arr[i - 1];
```

```
    // insert x at pos
```

```
    arr[pos - 1] = x;
```

```
    // print the updated array
```

```
    for (i = 0; i < n; i++)
```

```
        printf("%d ", arr[i]);
```

```
    printf("\n"); return 0;}
```

Output:

```
1 2 3 4 5 6 7 8 9 10
```

```
1 2 3 4 50 5 6 7 8 9 10
```

#### **Video Content / Details of website for further learning (if any):**

<https://www.geeksforgeeks.org/c-program-to-insert-an-element-in-an-array/>

<https://www.youtube.com/watch?v=yw0LsuV4GjQ>

#### **Important Books/Journals for further learning including the page nos.:**

Reema Thareja, Computer Fundamentals and Programming in C, Oxford University Press, Second Edition

John V Guttag Introduction to Computation and Programming Using Python Revised and expanded Edition, MIT Press, 2013 (Page No: 99-101)

Reema Thareja, Programming in C, Oxford University Press, Second Edition (Page No: 95-98)

**Course Faculty**

**Verified by HOD**



MUTHAYAMMAL ENGINEERING COLLEGE  
(An Autonomous Institution)



(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to  
Anna University)  
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

LECTURE HANDOUTS

L-24

CSE

I/I

Course Name with Code : Programming For Problem Solving Using C - 19GES01

Course Faculty : M.Ganthimathi

Unit : III- Functions and Arrays Date of Lecture:

**Topic of Lecture:** Deleting an Element from an Array

**Introduction:**

- C program to delete an element in an array: This program deletes or removes an element from an array. A user will enter the position at which the array element deletion is required. Deleting an element does not affect the size of the array. It also checks whether deletion is possible or not, for example, if an array contains five elements and user wants to delete the element at the sixth position, it isn't possible.

**Prerequisite knowledge for Complete understanding and learning of Topic:**

- Array position
- Indexing value
- Calculation for first value and last value

**Detailed content of the Lecture:**

```
#include <stdio.h>
int main()
{
    int array[100], position, c, n;

    printf("Enter number of elements in array\n");
    scanf("%d", &n);

    printf("Enter %d elements\n", n);

    for (c = 0; c < n; c++)
        scanf("%d", &array[c]);

    printf("Enter the location where you wish to delete element\n");
    scanf("%d", &position);

    if (position >= n+1)
        printf("Deletion not possible.\n");
    else
```



```

{
    for (c = position - 1; c < n - 1; c++)
        array[c] = array[c+1];

    printf("Resultant array:\n");

    for (c = 0; c < n - 1; c++)
        printf("%d\n", array[c]);
}

return 0;
}

```

### C program to delete element from array output:

```

E:\programmingsimplified.com\c\delete-array.exe
Enter number of elements in array
5
Enter 5 elements
4
6
8
10
7
Enter the location where you wish to delete element
2
Resultant array is
4
8
10
7

```

#### Video Content / Details of website for further learning (if any):

<https://www.programmingsimplified.com/c/source-code/c-program-delete-element-from-array>  
<https://www.programmingsimplified.com/c-program-examples>

#### Important Books/Journals for further learning including the page nos.:

Reema Thareja, Computer Fundamentals and Programming in C, Oxford University Press, Second Edition

John V Guttag Introduction to Computation and Programming Using Python Revised and expanded Edition, MIT Press, 2013 (Page No: 67-69)

Reema Thareja, Programming in C, Oxford University Press, Second Edition (Page No: 94- 96)

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE  
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to  
Anna University)  
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L-25

CSE

I/I

Course Name with Code : Programming For Problem Solving Using C - 19GES01

Course Faculty : M.Ganthimathi

Unit : III- Functions and Arrays Date of Lecture:

**Topic of Lecture:** Searching for a Value in an Array, two-dimensional arrays

**Introduction:**

- Input size and elements in array from user.
- Input number to search from user in some variable say to Search.
- Define a flag variable as found = 0.
- Run loop from 0 to size.
- Inside loop check if current array element is equal to searched number or not.

**Prerequisite knowledge for Complete understanding and learning of Topic:**

- Array operation
- Insertion and Deletion

**Detailed content of the Lecture:**

**Using Standard Method**

Read the array size and store that value into the variable n.

2) Read the entered elements and store those elements into an array a[] using scanf statement and the for loop. scanf(“%d”,&a[i]) indicates scanf statement reads the entered element and assigned that element to a[i].

3) Read the key which we want to search in an array.

4) Compare the key with each element of the array as a[i]==key and print element found, if the key matches with any element of the array otherwise print element not found.

```
#include <conio.h>
int main()
{
    int a[10000],i,n,key;

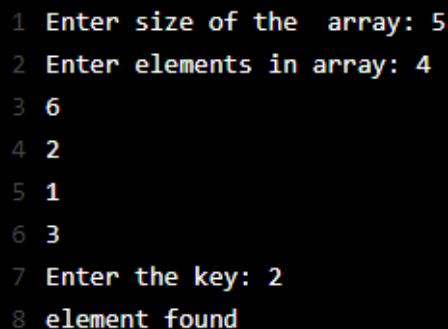
    printf("Enter size of the array : ");
    scanf("%d", &n);
    printf("Enter elements in array : ");
```

```
for(i=0; i<n; i++)
{
    scanf("%d",&a[i]);
}
printf("Enter the key : ");
scanf("%d", &key);

for(i=0; i<n; i++)
{
    if(a[i]==key)
    {
        printf("element found ");
        return 0;
    }
}

printf("element not found"); }
```

### **Output:**



```
1 Enter size of the array: 5
2 Enter elements in array: 4
3 6
4 2
5 1
6 3
7 Enter the key: 2
8 element found
```

### **Video Content / Details of website for further learning (if any):**

<https://javatutoring.com/c-program-search-element-array/>  
<https://www.programmingsimplified.com/c-program-examples>

### **Important Books/Journals for further learning including the page nos.:**

Reema Thareja, Computer Fundamentals and Programming in C, Oxford University Press, Second Edition

John V Guttag Introduction to Computation and Programming Using Python Revised and expanded Edition, MIT Press, 2013 (Page No: 96-98)

Reema Thareja, Programming in C, Oxford University Press, Second Edition (Page No: 57-60)

**Course Faculty**

**Verified by HOD**



MUTHAYAMMAL ENGINEERING COLLEGE  
(An Autonomous Institution)



(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to  
Anna University)  
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

LECTURE HANDOUTS

L-26

CSE

I/I

Course Name with Code : Programming For Problem Solving Using C - 19GES01

Course Faculty : M.Ganthimathi

Unit : III- Functions and Arrays Date of Lecture:

**Topic of Lecture:** operations on two dimensional arrays

**Introduction:**

- The two-dimensional array can be defined as an array of arrays. The 2D array is organized as matrices which can be represented as the collection of rows and columns. However, 2D arrays are created to implement a relational database lookalike data structure. It provides ease of holding the bulk of data at once which can be passed to any number of functions wherever required.

**Prerequisite knowledge for Complete understanding and learning of Topic:**

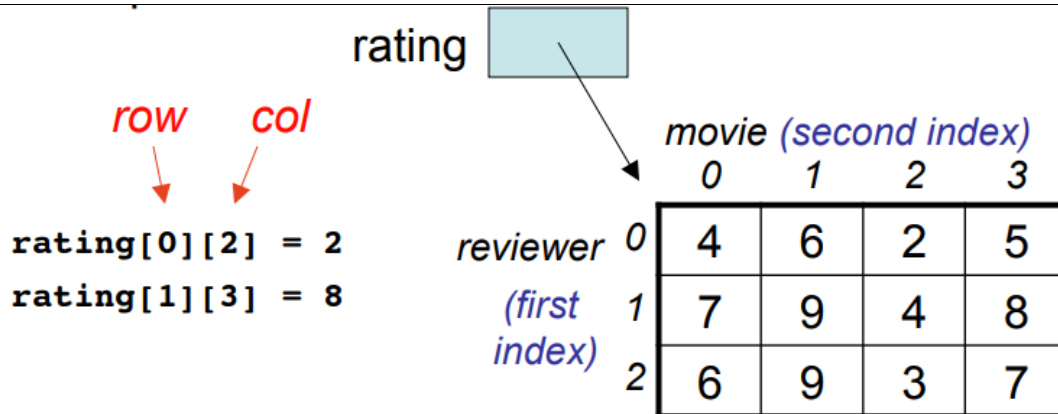
- 1- Dimensional Array
- Rows and Column's

**Detailed content of the Lecture:**

Declaration of two dimensional Array in C

The syntax to declare the 2D array is given below.

**data\_type array\_name[rows][columns];**



- Find the average rating by the reviewer in row 2.

```
int sum = 0;

for (int col = 0; col <= 3; col++) {
    sum += rating[2][col];
}
double average = (double) sum / 4;
```

	movie			
	0	1	2	3
reviewer 0	4	6	2	5
1	7	9	4	8
2	6	9	3	7

Summer 2010

15-110 (Reid-Miller)

```
#include<stdio.h>

int main(){
    /* 2D array declaration*/
    int disp[2][3];
    /*Counter variables for the loop*/
    int i, j;
    for(i=0; i<2; i++) {
        for(j=0;j<3;j++) {
            printf("Enter value for disp[%d][%d]:", i, j);
            scanf("%d", &disp[i][j]);
        }
    }
    //Displaying array elements
    printf("Two Dimensional array elements:\n");
    for(i=0; i<2; i++) {
        for(j=0;j<3;j++) {
```

```
printf("%d ", disp[i][j]);
if(j==2){
    printf("\n");
}
}
}
return 0;
}
```

**Output:**

Enter value for disp[0][0]:1

Enter value for disp[0][1]:2

Enter value for disp[0][2]:3

Enter value for disp[1][0]:4

Enter value for disp[1][1]:5

Enter value for disp[1][2]:6

**Two Dimensional array elements:**

1 2 3

4 5 6

**Video Content / Details of website for further learning (if any):**

<https://www.javatpoint.com/two-dimensional-array-in-c>

<https://beginnersbook.com/2014/01/2d-arrays-in-c-example/>

<https://www.programmingsimplified.com/c-program-examples>

**Important Books/Journals for further learning including the page nos.:**

Reema Thareja, Computer Fundamentals and Programming in C, Oxford University Press, Second Edition

John V Guttag Introduction to Computation and Programming Using Python Revised and expanded Edition, MIT Press, 2013 (Page No: 103-121)

Reema Thareja, Programming in C, Oxford University Press, Second Edition (Page No: 100-107)

**Course Faculty**

**Verified by HOD**





MUTHAYAMMAL ENGINEERING COLLEGE  
(An Autonomous Institution)



(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to  
Anna University)  
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

LECTURE HANDOUTS

L-27

CSE

I/I

Course Name with Code : Programming For Problem Solving Using C - 19GES01

Course Faculty : M.Ganthimathi

Unit : III- Functions and Arrays Date of Lecture:

**Topic of Lecture:** Sum, Difference

**Introduction:**

- Matrix addition in C language to add two matrices, i.e., compute their sum and print it. A user inputs their orders (number of rows and columns) and the matrices.
- C code to subtract matrices of any order. This program finds the difference between corresponding elements of two matrices and then print the resultant matrix.

**Prerequisite knowledge for Complete understanding and learning of Topic:**

- One-Dimensional Array
- Two-Dimensional Array
- Array operations

**Detailed content of the Lecture:**

```
#include <stdio.h>
int main() {
    int r, c, a[100][100], b[100][100], sum[100][100], i, j;
    printf("Enter the number of rows (between 1 and 100): ");
    scanf("%d", &r);
    printf("Enter the number of columns (between 1 and 100): ");
    scanf("%d", &c);

    printf("\nEnter elements of 1st matrix:\n");
    for (i = 0; i < r; ++i)
        for (j = 0; j < c; ++j) {
            printf("Enter element a%d%d: ", i + 1, j + 1);
            scanf("%d", &a[i][j]);
        }

    printf("Enter elements of 2nd matrix:\n");
    for (i = 0; i < r; ++i)
        for (j = 0; j < c; ++j) {
            printf("Enter element a%d%d: ", i + 1, j + 1);
            scanf("%d", &b[i][j]);
        }
}
```



```

    }

    // adding two matrices
    for (i = 0; i < r; ++i)
        for (j = 0; j < c; ++j) {
            sum[i][j] = a[i][j] + b[i][j];
        }

    // printing the result
    printf("\nSum of two matrices: \n");
    for (i = 0; i < r; ++i)
        for (j = 0; j < c; ++j) {
            printf("%d ", sum[i][j]);
            if (j == c - 1) {
                printf("\n\n");
            }
        }
    }

    return 0;
}

```

### Output

Enter the number of rows (between 1 and 100): 2

Enter the number of columns (between 1 and 100): 3

#### Enter elements of 1st matrix:

Enter element a11: 2

Enter element a12: 3

Enter element a13: 4

Enter element a21: 5

Enter element a22: 2

Enter element a23: 3

#### Enter elements of 2nd matrix:

Enter element a11: -4

Enter element a12: 5

Enter element a13: 3

Enter element a21: 5

Enter element a22: 6

Enter element a23: 3

#### Sum of two matrices:

-2 8 7

10 8 6

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    int m, n, c, d, first[10][10], second[10][10], difference[10][10];
```

```
    printf("Enter the number of rows and columns of matrix\n");
```

```
    scanf("%d%d", &m, &n);
```

```
    printf("Enter the elements of first matrix\n");
```

```

for (c = 0; c < m; c++)
    for (d = 0 ; d < n; d++)
        scanf("%d", &first[c][d]);

printf("Enter the elements of second matrix\n");
for (c = 0; c < m; c++)
    for (d = 0; d < n; d++)
        scanf("%d", &second[c][d]);

printf("Difference of entered matrices:-\n");
for (c = 0; c < m; c++) {
    for (d = 0; d < n; d++) {
        difference[c][d] = first[c][d] - second[c][d];
        printf("%d\t",difference[c][d]);
    }
    printf("\n");
}

return 0;
}

```

### Output of program:

```

E:\programmingsimplified.com\c\subtract-matrices.exe
Enter the number of rows and columns of matrix
2
2
Enter the elements of first matrix
4 3
2 1
Enter the elements of second matrix
1 2
1 1
difference of entered matrices:-
3      1
1      0

```

### Video Content / Details of website for further learning (if any):

<https://www.programmingsimplified.com/c-program-examples>  
<https://www.programiz.com/c-programming/examples/add-matrix>  
<https://www.programmingsimplified.com/c-program-add-matrices>

### Important Books/Journals for further learning including the page nos.:

Reema Thareja, Computer Fundamentals and Programming in C, Oxford University Press, Second Edition  
 John V Guttag Introduction to Computation and Programming Using Python Revised and expanded Edition, MIT Press, 2013 (Page No: 161-165)  
 Reema Thareja, Programming in C, Oxford University Press, Second Edition (Page No: 102-108)

Course Faculty

Verified by HOD



# MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)  
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



## LECTURE HANDOUTS

L - 28

CSE

I / I

**Course Name with Code : Programming For Problem Solving Using C - 19GES01**

**Course Teacher : M Ganthimathi**

**Unit : IV-Strings and Structures Date of Lecture :**

**Topic of Lecture :** Strings: Introduction

### **Introduction :**

Strings are actually one-dimensional array of characters terminated by a null character '\0'. Thus, a null-terminated string contains the characters that comprise the string followed by a null.

### **Prerequisite knowledge for Complete understanding and learning of Topic:**

**( Max. Four important topics)**

1. Basic concepts in C
2. Data types
3. Arrays
4. Looping statements

### **Detailed content of the Lecture:**

- The following declaration and initialization create a string consisting of the word "Hello".
- To hold the null character at the end of the array, the size of the character array containing the string is one more than the number of characters in the word "Hello."
- `char greeting[6] = {'H', 'e', 'l', 'l', 'o', '\0'};`
- If you follow the rule of array initialization, then you can write the above statement as follows:
- `char greeting[] = "Hello";`

- Following is the memory presentation of the above defined string in C

Index	0	1	2	3	4	5
Variable	H	e	l	l	o	\0
Address	0x23451	0x23452	0x23453	0x23454	0x23455	0x23456

Actually, you do not place the null character at the end of a string constant. The C compiler automatically places the '\0' at the end of the string when it initializes the array. Let us try to print the above mentioned string:

```
#include <stdio.h>
int main ()
{
char greeting[6] = {'H', 'e', 'l', 'l', 'o', '\0'};
printf("Greeting message: %s\n", greeting );
return 0;
}
```

When the above code is compiled and executed, it produces the following result:

Greeting message: Hello

C supports a wide range of functions that manipulate null-terminated strings:

**Video Content / Details of website for further learning (if any):**

<https://www.youtube.com/watch?v=SEZpQLG3AZ4>

**Important Books/Journals for further learning including the page nos.:**

Computer Fundamentals and Programming in C - Reema Thareja: Oxford University Press, Second Edition.  
 Programming in C – Reema Thareja, Oxford University Press, Second Edition. Page no. 317-320.



# MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Verified by HOD



LECTURE HANDOUTS

L - 29

CSE

I / I

**Course Name with Code : Programming For Problem Solving Using C - 19GES01**

**Course Teacher : M Ganthimathi**

**Unit : IV – Strings and Structures Date of Lecture :**

**Topic of Lecture :** Operations on Strings

## **Introduction :**

The C programming language has a set of functions implementing operations on strings (character strings and byte strings) in its standard library. Various operations, such as copying, concatenation, tokenization and searching are supported. There are various library functions to carry out the operations on strings.

## **Prerequisite knowledge for Complete understanding and learning of Topic:**

1. Basic concepts in C
2. Data types
3. Arrays
4. Looping statements

## **Detailed content of the Lecture:**

### **ARITHMETIC OPERATIONS ON STRINGS**

C supports a wide range of functions that manipulate null-terminated strings:

Characters in C can be used just like integers when used with arithmetic operators. This is nice, for example, in low memory applications because unsigned chars take up less memory than do regular integers as long as your value does not exceed the rather limited range of an unsigned char.

S.N.	Function & Purpose
1	<b>strcpy(s1, s2);</b> Copies string s2 into string s1.
2	<b>strcat(s1, s2);</b> Concatenates string s2 onto the end of string s1.
3	<b>strlen(s1);</b> Returns the length of string s1.
4	<b>strcmp(s1, s2);</b> Returns 0 if s1 and s2 are the same; less than 0 if s1<s2; greater than 0 if s1>s2.
5	<b>strchr(s1, ch);</b> Returns a pointer to the first occurrence of character ch in string s1.
6	<b>strstr(s1, s2);</b> Returns a pointer to the first occurrence of string s2 in string s1.

**For example: consider the following piece of code charmath.c:**

```
int answer;
printf("%d\n", val1);
#include <stdio.h>
void main() {
unsigned char val1 = 20;
unsigned char val2 = 30;
printf("%d\n", val2);
answer = val1 + val2;
printf("%d + %d = %d\n", val1, val2, answer);
val1 = 'a';
answer = val1 + val2;
printf("%d + %d = %d\n", val1, val2, answer);
```

}

- First we make two unsigned character variables and give them number values. We then add them together and put the answer into an integer variable. We can do this without a cast because characters are an alphanumeric data type. Next, we set var1 to an expected character value, the letter lowercase a. Now this next addition adds 97 to 30, why?
- Because the ASCII value of lowercase a is 97. So, it adds 97 to 30, the current value in var2. Notice it did not require casting the characters to integers or having the compiler complain. This is because the compiler knows when to automatically change between characters and integers or other numeric types.

**Video Content / Details of website for further learning (if any):**

<https://www.youtube.com/watch?v=-pSyzCWsBA8>

**Important Books/Journals for further learning including the page nos.:**

1. Computer Fundamentals and Programming in C - Reema Thareja: Oxford University Press, Second Edition. Page No.322-331.

Programming in C – Reema Thareja, Oxford University Press, Second Edition. Page No. 180-183.

**Course Faculty**

**Verified by HOD**







# MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)  
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



## LECTURE HANDOUTS

L - 30

CSE

I / I

**Course Name with Code : Programming For Problem Solving Using C - 19GES01**

**Course Teacher : M Ganthimathi**

**Unit : IV – Strings and Structures Date of Lecture :**

**Topic of Lecture :** Finding the length of a String

**Introduction :** The C programming language has a set of functions implementing operations on strings (character strings and byte strings) in its standard library. Various operations, such as copying, concatenation, tokenization and searching are supported. Built in function is used to find out the length of the string. The number of characters in the string determines the length of the string.

**Prerequisite knowledge for Complete understanding and learning of Topic:**

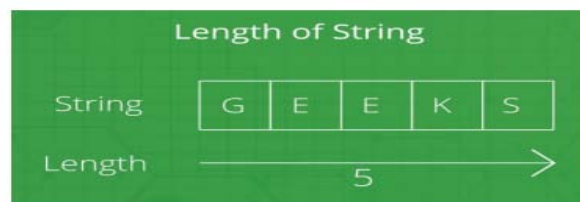
1. Basic concepts in C
2. Data types
3. Arrays
4. Looping statements

**Detailed content of the Lecture:**

C supports a wide range of functions that manipulate null-terminated strings:

**FINDING THE LENGTH OF A STRING :**

Given a string str. The task is to find the length of the string.



**Examples:**

Input: str = "Geeks"

Output: Length of Str is : 4

Input: str = "GeeksforGeeks"

Output: Length of Str is : 13

In the below program, to find the length of the string str, first the string is taken as input from the user using scanf in str, and then the length of Str is calculated using loop and using strlen() method .

Example 1: Using loop to calculate the length of string.

// C program to find the length of string

```
#include <stdio.h>
#include <string.h>
int main()
{
    char Str[1000]; int i;
    printf("Enter the String: ");
    scanf("%s", Str);
    for (i = 0; Str[i] != '\0'; ++i);
    printf("Length of Str is %d", i);
    return 0;
}
```

**Output:**

Enter the String: Geeks

Length of Str is 5

**Video Content / Details of website for further learning (if any):**

<https://www.youtube.com/watch?v=SEZpQLG3AZ4>

**Important Books/Journals for further learning including the page nos.:**

1. Computer Fundamentals and Programming in C - Reema Thareja: Oxford University Press, Second Edition. Page No.322-331.
2. Programming in C – Reema Thareja, Oxford University Press, Second Edition. Page No. 180-183.

**Course Faculty**

**Verified by HOD**



# MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



## LECTURE HANDOUTS

L - 31

CSE

I / I

**Course Name with Code : Programming For Problem Solving Using C - 19GES01**

**Course Teacher : M Ganthimathi**

**Unit : IV – Strings and Structures Date of Lecture :**

**Topic of Lecture :** Converting characters of a string into upper case

### Introduction :

The C programming language has a set of functions implementing operations on strings (character strings and byte strings) in its standard library. Various operations, such as copying, concatenation, tokenization and searching are supported. There are built in functions to convert the string in lower case to upper case and vice versa

### Prerequisite knowledge for Complete understanding and learning of Topic:

1. Basic concepts in C
2. Data types
3. Arrays
4. Looping statements

### Detailed content of the Lecture:

#### CONVERTING CHARACTERS OF A STRING INTO UPPERCASE :

- The C Strupr function is one of the String Function, which is used to convert the given characters or string into Uppercase letters.
- C Strupr Function syntax
- The following function will accept the characters as the parameter and convert all the characters in a string to uppercase using the built-in String function Strupr in C.
- Remember, you have to include the #include<string.h> header before using any string function.

**strupr(chars);**

The `strupr` method is used to convert all the characters in a given string to uppercase.

**This program will help you to understand the same.**

```
#include <stdio.h>
#include <string.h>
int main()
{
char str[] = "C ProgramminG Tutorial at Tutorial GateWay";
char str1[] = "c laGUagE";
char str2[] = "Java Programming Language";
char str3[] = "c PRogramming WOrld";
char str4[] = "TrY tO ReAd ThIs SenTEncE";
printf("\n Upper Case String is = %s", strupr(str));
printf("\n Upper Case String is = %s", strupr(str1));
printf("\n Upper Case String is = %s", strupr(str2));
printf("\n Upper Case String is = %s", strupr(str3));
printf("\n Upper Case String is = %s", strupr(str3));
}
```

**Convert a String to Uppercase in C without using `strupr` ()**

**Example :**

```
#include <stdio.h>
#include <string.h>
int main() {
char s[100]; int i;
printf("\nEnter a string : ");
gets(s);
for (i = 0; s[i]!='\0'; i++) {
if(s[i] >= 'a' && s[i] <= 'z') {
s[i] = s[i] -32;
} }
printf("\nString in Upper Case = %s", s);
return 0;
```

```
}
```

**Output :**

Enter a string : hello world!

String in Upper Case = HELLO WORLD!

- In the program, the actual code of conversion of string to upper case is present in main() function. An array of char type s[100] is declared which will store the entered string by user.
- Then, for loop is used to convert the string into upper case string and if block is used to
- check that if characters are in lower case then, convert them in upper case by subtracting 32 from their ASCII value.

**Video Content / Details of website for further learning (if any):**

<https://www.youtube.com/watch?v=w2dNnBlg5uw>

**Important Books/Journals for further learning including the page nos.:**

1. Computer Fundamentals and Programming in C - Reema Thareja: Oxford University Press, Second Edition. Page No.322-331.
2. Programming in C – Reema Thareja, Oxford University Press, Second Edition. Page No. 180-183.

**Course Faculty**

**Verified by HOD**





# MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



## LECTURE HANDOUTS

L - 32

CSE

I / I

**Course Name with Code : Programming For Problem Solving Using C - 19GES01**

**Course Teacher : M Ganthimathi**

**Unit : IV-Strings and Structures Date of Lecture :**

**Topic of Lecture :** Converting characters of a string into lower case.

### **Introduction :**

String is set of characters. The string can be in lowercase or uppercase. There are built in functions to convert strings in lowercase to uppercase and vice versa. The C Strlwr function is one of the String Function, which is used to convert the user specified characters or string into Lowercase letters.

### **Prerequisite knowledge for Complete understanding and learning of Topic:**

1. Basic concepts in C
2. Data types
3. Arrays
4. Looping statements

### **Detailed content of the Lecture:**

The C Strlwr function is one of the String Function, which is used to convert the user specified characters or string into Lowercase letters.

#### **C Strlwr Function syntax**

The following C strlwr function will accept the characters as the parameter and convert all the characters in a string to lowercase using the built-in String function Strlwr. strlwr(chars)

#### **Strlwr in C Programming Example**

The C strlwr method is used to convert all the characters in a given string into lowercase.

This program will help you to understand the same.

//strlwr in C Programming

```
#include <stdio.h>
```

```
#include<string.h>
```

```
int main()
```

```
{char str[] = "C LanGUAGE Tutorial AT TUTORIal GATEWaN";
```

```

char str1[] = "C LaGUagE";
char str2[] = "Java Programming Language";
char str3[] = "c PRogramms";
char str4[] = "TrY tO ReAd ThIs SenTEncE";
printf("\n Lower Case String is = %s", strlwr(str));
printf("\n Lower Case String is = %s", strlwr(str1));
printf("\n Lower Case String is = %s", strlwr(str2));
printf("\n Lower Case String is = %s", strlwr(str3));
printf("\n Lower Case String is = %s", strlwr(str3));
}

```

### **C Program to convert string to Lowercase without Strupr**

This program will help you to understand, how to write a C program to convert the given string into Lower case without using the built-in string function strlwr.

```

#include <stdio.h>
void String_Lower(char []);
int main()
{ char str[100];
printf("\n Please Enter a string to convert it into Lowercase\n");
gets(str);
String_Lower(str);
printf("\n Lower Case String is = %s", str);
return 0;
}
void String_Lower(char string[])
{ int i = 0;
while (string[i] != '\0')
{ if (string[i] >= 'A' && string[i] <= 'Z') { string[i] = string[i] + 32; }
i++; } }

```

### **Video Content / Details of website for further learning (if any):**

<https://www.youtube.com/watch?v=Oo-OyHTsZk4>

### **Important Books/Journals for further learning including the page nos.:**

1. Computer Fundamentals and Programming in C - Reema Thareja: Oxford University Press, Second Edition. Page No.322-331.
- 2.Programming in C – Reema Thareja, Oxford University Press, Second Edition.Page No. 180-183.

**Course Faculty**

**Verified by HOD**





# MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)  
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L - 33

CSE

I / I

**Course Name with Code : Programming For Problem Solving Using C - 19GES01**

**Course Teacher : M Ganthimathi**

**Unit : IV- Strings and Structures Date of Lecture :**

**Topic of Lecture :** Structures: Introduction to Structures

## **Introduction :**

Structure is a user-defined data type in C language which allows us to combine data of different types together. Structure helps to construct a complex data type which is more meaningful. It is somewhat similar to an Array, but an array holds data of similar type only. But structure on the other hand, can store data of any type, which is practical more useful.

## **Prerequisite knowledge for Complete understanding and learning of Topic:**

1. Basic concepts in C
2. Data types
3. Arrays
4. Looping statements

## **Detailed content of the Lecture:**

Structures are used to represent a record, suppose you want to keep track of your books in a library. You might want to track the following attributes about each book:

- Title
- Author
- Subject
- Book ID

## DEFINING A STRUCTURE

- To define a structure, you must use the struct statement. The struct statement defines a new data type, with more than one member for your program. The format of the struct statement is this :

```
struct [structure tag]
{ member definition; member definition;
...
member definition;
} [one or more structure variables];
```

- The structure tag is optional and each member definition is a normal variable definition, such as int i; or float f; or any other valid variable definition. At the end of the structure's definition, before the final semicolon, you can specify one or more structure variables but it is optional.

Here is the way you would declare the Book structure:

```
struct Books
{
char title[50]; char
author[50]; char
subject[100]; int
book_id;
} book;
```

## DECLARING STRUCTURE VARIABLES

Structure variable declaration is similar to the declaration of any normal variable of any other data type. Structure variables can be declared in following two ways:

### Declaring Structure variables separately

```
struct Student
{
char name[25];
int age;
char branch[10];
//F for female and M for male
char gender;
```

```
};  
struct Student S1, S2;  
//declaring  
variables of  
struct Student
```

### **Declaring Structure variables with structure definition**

```
struct Student  
{  
char name[25];  
int age;  
char branch[10];  
//F for female and M for male  
char gender;  
} S1, S2; //declaring variables of struct Student
```

Here S1 and S2 are variables of structure Student. However this approach is not much recommended.

### **Video Content / Details of website for further learning (if any):**

<https://www.youtube.com/watch?v=3loZ0NOc4Wc>

### **Important Books/Journals for further learning including the page nos.:**

1. Computer Fundamentals and Programming in C - Reema Thareja: Oxford University Press, Second Edition, Page no.386-388.
2. Programming in C – Reema Thareja, Oxford University Press, Second Edition. Page no.259-262.

**Course Faculty**

**Verified by HOD**





# MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)  
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L - 34

CSE

I / I

**Course Name with Code : Programming For Problem Solving Using C - 19GES01**

**Course Teacher : M Ganthimathi**

**Unit : IV-Strings and Structures Date of Lecture :**

**Topic of Lecture :** Copying structures

**Introduction :** Two variables of the same structure type can be copied the same way as ordinary variables.

If person1 and person2 belong to the same structure, then the following statements are valid.

```
person1 = person2;
```

```
person2 = person1;
```

**Prerequisite knowledge for Complete understanding and learning of Topic:**

1. Basic concepts in C
2. Data types
3. Arrays
4. Looping statements

**Detailed content of the Lecture:**

C does not permit any logical operators on structure variables. In case, we need to compare them, we may do so by comparing members individually.

```
person1 == person2
```

```
person1 != person2
```

The above statements are not permitted.

Example

```
{  
int number;  
char name[20];  
float marks;  
};  
main()  
{
```

```
int x;
structclass student1 = { 111,"Rao",72.50};
structclass student2 = { 222,"Reddy", 67.00};
structclass student3;
student3 = student2;// copying structures
x = ((student3.number == student2.number) &&
(student3.marks == student2.marks)) ? 1 : 0;
if(x == 1)
{
printf("\nstudent2 and student3 are same\n\n");
printf("%d %s %f\n", student3.number,
student3.name,
student3.marks);
}
else
printf("\nstudent2 and student3 are different\n\n");
}
```

#### Output

```
student2 and student3 are same
222 Reddy 67.000000
```

#### **Video Content / Details of website for further learning (if any):**

<https://www.youtube.com/watch?v=3loZ0NOc4Wc>

#### **Important Books/Journals for further learning including the page nos.:**

Computer Fundamentals and Programming in C - Reema Thareja: Oxford University Press, Second Edition, Page No.388-389.

Programming in C – Reema Thareja, Oxford University Press, Second Edition. Page No.262-265.

**Course Faculty**

**Verified by HOD**



# MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)  
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



## LECTURE HANDOUTS

L - 35

CSE

I / I

**Course Name with Code : Programming For Problem Solving Using C - 19GES01**

**Course Teacher : M Ganthimathi**

**Unit : IV-Strings and Structures Date of Lecture :**

**Topic of Lecture :** Comparing structures

### Introduction :

Structure is collection of data items of different data types. The values of one structure can be copied to another structures as well as compared with each other. Two variables of the same structure type can be compared the same way as ordinary variables.

### Prerequisite knowledge for Complete understanding and learning of Topic:

1. Basic concepts in C
2. Data types
3. Arrays
4. Looping statements

### Detailed content of the Lecture:

- C does not permit any logical operators on structure variables. In case, we need to compare them, we may do so by comparing members individually.

```
person1 == person2
```

```
person1 != person2
```

- The above statements are not permitted.

### Example

```
main()
{
int number;
char name[20];
float marks;
```

```
};
main()
{
int x;
structclass student1 = {111,"Rao",72.50};
structclass student2 = {222,"Reddy", 67.00};
structclass student3;
student3 = student2;
x = ((student3.number == student2.number) &&
(student3.marks == student2.marks)) ? 1 : 0; // comparing structures
if(x == 1)
{
printf("\nstudent2 and student3 are same\n\n");
printf("%d %s %f\n", student3.number,
student3.name,
student3.marks);
}
else
printf("\nstudent2 and student3 are different\n\n");
}
```

#### Output

```
student2 and student3 are same
222 Reddy 67.000000
```

#### **Video Content / Details of website for further learning (if any):**

<https://www.youtube.com/watch?v=3loZ0NOc4Wc>

#### **Important Books/Journals for further learning including the page nos.:**

Computer Fundamentals and Programming in C - Reema Thareja: Oxford University Press, Second Edition, Page No.388-389.

Programming in C – Reema Thareja, Oxford University Press, Second Edition. Page No.262-265.

**Course Faculty**

**Verified by HOD**





# MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)  
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



## LECTURE HANDOUTS

L -3 6

CSE

I / I

**Course Name with Code : Programming For Problem Solving Using C - 19GES01**

**Course Teacher : M Ganthimathi**

**Unit : IV –Strings and Structures Date of Lecture :**

**Topic of Lecture :** Nested structures.

**Introduction :** Nested structure in C is nothing but structure within structure. One structure can be declared inside other structure as we declare structure members inside a structure. The structure variables can be a normal structure variable or a pointer variable to access the data.

**Prerequisite knowledge for Complete understanding and learning of Topic:**

**( Max. Four important topics)**

- Basic concepts in C
- Data types
- Arrays
- Looping statements

**Detailed content of the Lecture:**

1. Structure within structure in C using normal variable
2. Structure within structure in C using pointer variable

**STRUCTURE WITHIN STRUCTURE USING NORMAL VARIABLE:**

This program explains how to use structure within structure in C using normal variable. “student\_college\_detail” structure is declared inside “student\_detail” structure in this program. Both structure variables are normal structure variables.

Please note that members of “student\_college\_detail” structure are accessed by 2 dot(.) operator and members of “student\_detail” structure are accessed by single dot(.) operator.

```
#include <stdio.h>
#include <string.h>
struct student_college_detail
{
    int college_id;
    char college_name[50];
};
struct student_detail
{
    int id;
    char name[20];
    float percentage;
    // structure within structure
    struct student_college_detail clg_data;
}stu_data;
int main()
{
    struct student_detail stu_data = {1, "Raju", 90.5, 71145,
    "Anna University"};
    printf(" Id is: %d \n", stu_data.id);
    printf(" Name is: %s \n", stu_data.name);
    printf(" Percentage is: %f \n\n", stu_data.percentage);
    printf(" College Id is: %d \n",
    stu_data.clg_data.college_id);
    printf(" College Name is: %s \n",
    stu_data.clg_data.college_name);
    return 0;
}
```

**OUTPUT:**

Id is: 1

Name is : Raju

Percentage is : 90.500000

College Id is : 71145

College Name is : Anna University

### **STRUCTURE WITHIN STRUCTURE USING POINTER VARIABLE:**

This program explains how to use structure within structure in C using pointer variable. "student\_college\_detail" structure is declared inside "student\_detail" structure in this program. one normal structure variable and one pointer structure variable is used in this program. Please note that combination of .(dot) and ->(arrow) operators are used to access the structure member which is declared inside the structure.

```
#include <stdio.h>
#include <string.h>
struct student_college_detail
{
    int college_id;
    char college_name[50];
};
struct student_detail
{
    int id;
    char name[20];
    float percentage;
    // structure within structure
    struct student_college_detail clg_data;
}stu_data, *stu_data_ptr;
int main()
{
    struct student_detail stu_data = {1, "Raju", 90.5, 71145,
    "Anna University"};
    stu_data_ptr = &stu_data;
    printf(" Id is: %d \n", stu_data_ptr->id);
    printf(" Name is: %s \n", stu_data_ptr->name);
    printf(" Percentage is: %f \n\n",
```

```
stu_data_ptr->percentage);  
printf(" College Id is: %d \n",  
stu_data_ptr->clg_data.college_id);  
printf(" College Name is: %s \n",  
stu_data_ptr->clg_data.college_name);  
return 0;  
}
```

**OUTPUT:**

Id is: 1

Name is: Raju

Percentage is: 90.500000

College Id is: 71145

College Name is: Anna University

**Video Content / Details of website for further learning (if any):**

<https://www.youtube.com/watch?v=3loZONOc4Wc>

**Important Books/Journals for further learning including the page nos.:**

Computer Fundamentals and Programming in C - Reema Thareja: Oxford University Press, Second Edition, Page No. 392.

Programming in C – Reema Thareja, Oxford University Press, Second Edition. Page No. 265-266.

**Course Faculty**

**Verified by HOD**



# MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)  
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L-37

CSE

I / I

Course Name with Code : Programming for Problem Solving using C-19GES01

Course Faculty : M.Ganthimathi

Unit : V - Pointers and File Processing Date of Lecture:

**Topic of Lecture:** Introduction to Pointers

### Introduction :

Pointers in C language is a variable that stores / points the address of another variable. A Pointer in C is used to allocate memory dynamically i.e. at run time. The pointer variable might be belonging to any of the data type such as int, float, char, double, short etc.

### Prerequisite knowledge for Complete understanding and learning of Topic:

1. C programming Data types
2. Variable declaration
3. Input and Output Statements
4. Conditional statements
5. Memory Management

### Detailed content of the Lecture:

- A pointer is a variable whose value is the address of another variable, i.e., direct address of the memory location. Like any variable or constant, you must declare a pointer before using it to store any variable address.

Syntax : data\_type \*var\_name;

- Example: int \*p; char \*p; Where, \* is used to denote that "p" is pointer variable and not a normal variable. Normal variable stores the value whereas pointer variable stores the address of the variable.
- The content of the C pointer always be a whole number i.e. address. Always C pointer is

initialized to null, i.e. `int *p = null`.

- The value of null pointer is 0. `&` symbol is used to get the address of the variable. `*` symbol is used to get the value of the variable that the pointer is pointing to. If a pointer in C is assigned to `NULL`, it means it is pointing to nothing.
- Two pointers can be subtracted to know how many elements are available between these two pointers.

### PROGRAMS

```
#include <stdio.h>

int main()
{
    int var = 5;
    printf("var: %d\n", var); // Notice the use of & before var
    printf("address of var: %p", &var);
    return 0;
}
```

```
#include <stdio.h>

int main()
{
    int *ptr, q;
    q = 50;
    /* address of q is assigned to ptr */
    ptr = &q;
    /* display q's value using ptr variable */
    printf("%d", *ptr);
    return 0;
}

#include<stdio.h>

int main(){
```

```
int number=50;
int *p;
p=&number;
printf("Address of p variable is %x \n",p);
printf("Value of p variable is %d \n",*p);
return 0;
}
```

**Video Content / Details of website for further learning (if any):**

<https://www.youtube.com/watch?v=sY-s7O0FiYE>

**Important Books/Journals for further learning including the page nos.:**

1. Computer Fundamentals and Programming in C - Reema Thareja: Oxford University Press, Second Edition. Page no.348.
2. Programming in C – Reema Thareja, Oxford University Press, Second Edition. Page no.214.

**Course Faculty**

**Verified by HOD**



# MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)  
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L-38

CSE

I / I

Course Name with Code : Programming for Problem Solving using C-19GES01

Course Faculty : M.Ganthimathi

Unit : V - Pointers and File Processing Date of Lecture:

**Topic of Lecture:** Declaring pointer variables

**Introduction :**

A pointer is a variable whose value is the address of another variable, i.e., direct address of the memory location. Like any variable or constant, you must declare a pointer before using it to store any variable address.

**Prerequisite knowledge for Complete understanding and learning of Topic:**

- C programming Data types
- Variable declaration
- Input and Output Statements
- Conditional statements
- Memory Management

**Detailed content of the Lecture:**

Data type of a pointer must be same as the data type of the variable to which the pointer variable is pointing. void type pointer works with all data types, but is not often used.

Syntax to declare a pointer

```
data_type *pointer_name;  
int *ptr;
```

Here, in this statement

- ptr is the name of pointer variable (name of the memory blocks in which address of another variable is going to be stored).



- The character asterisk (\*) tells to the compiler that the identifier ptr should be declare as pointer.
- The data type int tells to the compiler that pointer ptr will store memory address of integer type variable.
- Finally, ptr will be declared as integer pointer which will store address of integer type variable.
- Pointer ptr is declared, but it not pointing to anything; now pointer should be initialized by the address of another integer variable.

#### Example

```
int* pc, c;
c = 5;
pc = &c;
printf("%d", *pc);
```

Accessing address and value of x using pointer variable ptr. We can get the value of ptr which is the address of x (an integer variable)

- ptr will print the stored value (memory address of x).
- \*ptr will print the value which is stored at the containing memory address in the ptr (value of variable x).

```
#include <stdio.h>
int main()
{
    int x=20; //int variable
    int *ptr; //int pointer declaration
    ptr=&x; //initializing pointer
    printf("Memory address of x: %p\n",ptr);
    printf("Value x: %d\n",*ptr);

    return 0;
}
```

#### **Video Content / Details of website for further learning (if any):**

[www.youtube.com/watch?v=sj0g6rn-RSk](http://www.youtube.com/watch?v=sj0g6rn-RSk)

#### **Important Books/Journals for further learning including the page nos.:**

Computer Fundamentals and Programming in C - Reema Thareja: Oxford University Press, Second Edition. Page no.348.

**Course Faculty**

**Verified by HOD**



# MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)  
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L-39

CSE

I / I

Course Name with Code : Programming for Problem Solving using C-19GES01

Course Faculty : M.Ganthimathi

Unit : V - Pointers and File Processing Date of Lecture:

**Topic of Lecture:** Passing arguments to function using pointers

**Introduction :**

Pointers can be passed as argument to functions. Just like any other argument, pointers can also be passed to a function as an argument. When we pass a pointer as an argument instead of a variable then the address of the variable is passed instead of the value. So any change made by the function using the pointer is permanently made at the address of passed variable. This technique is known as call by reference in C.

**Prerequisite knowledge for Complete understanding and learning of Topic:**

- C Programming Fundamentals
- Functions in C
- Passing Arguments to functions
- Default arguments
- Pointers

**Detailed content of the Lecture:**

- Pointer as a function parameter is used to hold addresses of arguments passed during function call. This is also known as call by reference.
- When a function is called by reference any change made to the reference variable will effect the original variable.
- A function can also return a pointer to the calling function. In this case you must be careful, because local variables of function doesn't live outside the function.
- They have scope only inside the function. Hence if you return a pointer connected to a local variable, that pointer will be pointing to nothing when the function ends.

```

#include <stdio.h>
void salaryhike(int *var, int b)
{ *var = *var+b; }
int main()
{ int salary=0, bonus=0;
  printf("Enter the employee current salary:");
  scanf("%d", &salary);
  printf("Enter bonus:");
  scanf("%d", &bonus);
  salaryhike(&salary, bonus);
  printf("Final salary: %d", salary);
  return 0;
}
#include <stdio.h>
void swapnum(int *num1, int *num2)
{ int tempnum;
  tempnum = *num1;
  *num1 = *num2;
  *num2 = tempnum;
}
int main()
{ int v1 = 11, v2 = 77 ;
  printf("Before swapping:");
  printf("\nValue of v1 is: %d", v1);
  printf("\nValue of v2 is: %d", v2);
  swapnum( &v1, &v2 );    /*calling swap function*/
  printf("\nAfter swapping:");
  printf("\nValue of v1 is: %d", v1);
  printf("\nValue of v2 is: %d", v2);
}

```

**Video Content / Details of website for further learning (if any):**

<https://www.youtube.com/watch?v=GY-H2jenPRo>

**Important Books/Journals for further learning including the page nos.:**

1. Computer Fundamentals and Programming in C - Reema Thareja: Oxford University Press, Second Edition. Page no.354
2. Programming in C – Reema Thareja, Oxford University Press, Second Edition. Page no.222-223.

**Course Faculty**

**Verified by HOD**



# MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)  
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L-40

CSE

I / I

Course Name with Code : Programming for Problem Solving using C-19GES01

Course Faculty : M.Ganthimathi

Unit : V-Pointers and File Processing Date of Lecture:

**Topic of Lecture:** Introduction to Files

**Introduction :** A file represents a sequence of bytes, regardless of it being a text file or a binary file. C programming language provides access on high level functions as well as low level (OS level) calls to handle file on your storage devices. This chapter will take you through the important calls for file management.

**Prerequisite knowledge for Complete understanding and learning of Topic:**

- Data types, Keywords, Identifier
- Input /Output Statements
- Control Statements
- Pointers
- Memory management

**Detailed content of the Lecture:**

- In programming, we may require some specific input data to be generated several numbers of times. Sometimes, it is not enough to only display the data on the console.
- The data to be displayed may be very large, and only a limited amount of data can be displayed on the console, and since the memory is volatile, it is impossible to recover the programmatically generated data again and again.
- However, if we need to do so, we may store it onto the local file system which is volatile and can be accessed every time. Here, comes the need of file handling in C.

File handling in C enables us to create, update, read, and delete the files stored on the local file

system through our C program. The following operations can be performed on a file.

- Creation of the new file, Opening an existing file, Reading from the file, Writing to the file and Deleting the file

The various functions for file handling are

1	fopen()	opens new or existing file
2	fprintf()	write data into the file
3	fscanf()	reads data from the file
4	fputc()	writes a character into the file
5	fgetc()	reads a character from file
6	fclose()	closes the file
7	fseek()	sets the file pointer to given position
8	fputw()	writes an integer to file
9	fgetw()	reads an integer from file
10	ftell()	returns current position
11	rewind()	sets the file pointer to the beginning of the file

**Video Content/ Details of website for further learning (if any):**

[//www.youtube.com/watch?v=wVDfRzBp8iE&list=PLfVsf4Bjg79BOMLYBRTwqCIkGPiOWb7xj](https://www.youtube.com/watch?v=wVDfRzBp8iE&list=PLfVsf4Bjg79BOMLYBRTwqCIkGPiOWb7xj)

**Important Books/Journals for further learning including the page nos.:**

1.Computer Fundamentals and Programming in C - Reema Thareja: Oxford University Press, Second Edition. Page no.415

2. Programming in C – Reema Thareja, Oxford University Press, Second Edition. Page no.290-291.

**Course Faculty**



# MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)  
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Verified by HOD



LECTURE HANDOUTS

L-41

CSE

I / I

Course Name with Code : Programming for Problem Solving using C-19GES01

Course Faculty : M.Ganthimathi

Unit : I -Pointers and File Processing Date of Lecture:

## Topic of Lecture: File Types

**Introduction :** File is collection of data. There are two types of files text files and binary files. There are appropriate functions for handling files. The operations performed with files are open, close, read, write etc.

## Prerequisite knowledge for Complete understanding and learning of Topic:

- C Programming Fundamentals
- Functions
- Passing Arguments to functions
- Pointers
- Passing pointers as arguments to functions

## Detailed content of the Lecture:

When a program is terminated, the entire data is lost. Storing in a file will preserve your data even if the program terminates.

If you have to enter a large number of data, it will take a lot of time to enter them all. However, if you have a file containing all the data, you can easily access the contents of the file using a few commands in C.

## Types of Files

When dealing with files, there are two types of files you should know about:

Text files

Binary files

### **Text files**

- Text files are the normal .txt files. You can easily create text files using any simple text editors such as Notepad. When you open those files, you'll see all the contents within the file as plain text. You can easily edit or delete the contents.
- They take minimum effort to maintain, are easily readable, and provide the least security and takes bigger storage space.

### Example-

#### 1.w(write):

This mode opens new file on the disk for writing.If the file exist,disk for writing.If the file exist, then it will be over written without then it will be over written without any confirmation.

```
fp=fopen("data.txt","w");
```

"data.txt" is filename

"w" is writemode.

#### 2. r (read)

This mode opens an preexisting file for reading.If the file doesn't Exist then the compiler returns a NULL to the file pointer

```
SYNTAX: fp=fopen("data.txt","r");
```

#### 3. w+(read and write)

This mode searches for a file if it is found contents are destroyed If the file doesn't found a new file is created.

```
SYNTAX: fp=fopen("data.txt","w+");
```

#### 4.a(append)

This mode opens a preexisting file for appending the data.

```
SYNTAX fp=fopen("data.txt","a");
```

#### 5.a+(append+read)

the end of the file.

```
SYNTAX: fp=fopen("data.txt","a+");
```

#### 6.r+ (read +write)

This mode is used for both Reading and writing

### Binary files

- Binary files are mostly the .bin files in your computer. Instead of storing data in plain text, they store it in the binary form (0's and 1's). They can hold a higher amount of data, are not readable easily, and provides better security than text files.
- A binary file is a file that uses all 8 bits of a byte for storing the information .It is the form which

can be interpreted and understood by the computer.

- The only difference between the text file and binary file is the data contain in text file can be recognized by the word processor while binary file data can't be recognized by a word processor.

#### 1.wb(write)

this opens a binary file in write mode.

SYNTAX: `fp=fopen("data.dat","wb");`

#### 2.rb(read)

this opens a binary file in read mode

SYNTAX: `fp=fopen("data.dat","rb");`

#### 3.ab(append)

this opens a binary file in a Append mode i.e. data can be added at the end of file.

SYNTAX: `fp=fopen("data.dat","ab");`

#### 4.r+b(read+write)

this mode opens preexisting File in read and write mode.

SYNTAX: `fp=fopen("data.dat","r+b");`

#### 5.w+b(write+read)

this mode creates a new file for reading and writing in Binary mode.

SYNTAX: `fp=fopen("data.dat","w+b");`

#### 6.a+b(append+write)

this mode opens a file in append mode i.e. data can be written at the end of file.

SYNTAX: `fp=fopen("data.dat","a+b");`

**Video Content / Details of website for further learning (if any):**

[https://www.youtube.com/watch?v=\\_KW\\_YBTXhN0](https://www.youtube.com/watch?v=_KW_YBTXhN0)

#### **Important Books/Journals for further learning including the page nos.:**

1.Computer Fundamentals and Programming in C - Reema Thareja: Oxford University Press, Second Edition. Page no.417

2.Programming in C – Reema Thareja, Oxford University Press, Second Edition. Page no.290-291.

**Course Faculty**

**Verified by HOD**







# MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)  
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L-42

CSE

I / I

Course Name with Code : Programming for Problem Solving using C-19GES01

Course Faculty : M.Ganthimathi

Unit : V-Pointers and File Processing Date of Lecture:

**Topic of Lecture:** File Modes

**Introduction :** A file represents a sequence of bytes, regardless of it being a text file or a binary file. C programming language provides access on high level functions as well as low level (OS level) calls to handle file on your storage devices.

**Prerequisite knowledge for Complete understanding and learning of Topic:**

- C Fundamentals
- Statements in C
- Functions
- Pointers
- Files

**Detailed content of the Lecture:**

For opening a file, fopen function is used with the required access modes. Some of the commonly used file access modes are

- “r” – Searches file. If the file is opened successfully fopen( ) loads it into memory and sets up a pointer which points to the first character in it. If the file cannot be opened fopen( ) returns NULL.
- “w” – Searches file. If the file exists, its contents are overwritten. If the file doesn’t exist, a new file is created. Returns NULL, if unable to open file.
- “a” – Searches file. If the file is opened successfully fopen( ) loads it into memory and sets up a pointer that points to the last character in it. If the file doesn’t exist, a new file is created. Returns NULL, if unable to open file.
- “r+” – Searches file. If is opened successfully fopen( ) loads it into memory and sets up a pointer which points to the first character in it. Returns NULL, if unable to open the file.

- “w+” – Searches file. If the file exists, its contents are overwritten. If the file doesn’t exist a new file is created. Returns NULL, if unable to open file.
- “a+” – Searches file. If the file is opened successfully fopen( ) loads it into memory and sets up a pointer which points to the last character in it. If the file doesn’t exist, a new file is created. Returns NULL, if unable to open file.

```
// C program to Open a File,  
// Write in it, And Close the File
```

```
# include <stdio.h>  
# include <string.h>  
int main( )  
{  
    // Declare the file pointer  
    FILE *filePointer ;  
    // Get the data to be written in file  
    char dataToBeWritten[50]  
        = "GeeksforGeeks-A Computer Science Portal for Geeks";  
    // Open the existing file GfgTest.c using fopen()  
    // in write mode using "w" attribute  
    filePointer = fopen("GfgTest.c", "w") ;  
    // Check if this filePointer is null  
    // which maybe if the file does not exist  
    if ( filePointer == NULL )  
    {  
        printf( "GfgTest.c file failed to open." ) ;  
    }  
    else  
    {  
        printf("The file is now opened.\n") ;  
        // Write the dataToBeWritten into the file  
        if ( strlen ( dataToBeWritten ) > 0 )  
        {  
            // writing in the file using fputs()  
            fputs(dataToBeWritten, filePointer) ;  
            fputs("\n", filePointer) ;  
        }  
        // Closing the file using fclose()  
        fclose(filePointer) ;  
        printf("Data successfully written in file GfgTest.c\n");  
        printf("The file is now closed." ) ;  
    }  
}
```

```
}  
    return 0;  
}
```

**Video Content/ Details of website for further learning (if any):**

<https://www.youtube.com/watch?v=lnmGb8AmS6Q>

**Important Books/Journals for further learning including the page nos.:**

3. Computer Fundamentals and Programming in C - Reema Thareja: Oxford University Press, Second Edition. Page no.417
4. Programming in C – Reema Thareja, Oxford University Press, Second Edition. Page no.290-291.

**Course Faculty**

**Verified by HOD**





# MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)  
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L-43

CSE

I / I

Course Name with Code : Programming for Problem Solving using C-19GES01

Course Faculty : M.Ganthimathi

Unit : I -Pointers and File Processing Date of Lecture:

**Topic of Lecture:** Read Data from Files

**Introduction :** File is collection of data. There are two types of files Text file and Binary File. Data can be written into and can be accessed from a file. A file need to be opened for reading, then data is read from the file, and then the file is closed. There are appropriate functions for handling files.

**Prerequisite knowledge for Complete understanding and learning of Topic:**

- C Fundamentals
- Statements in C
- Functions
- Pointers
- Files

**Detailed content of the Lecture:**

- Given below is the simplest function to read a single character from a file -

**int fgetc( FILE \* fp );**

1. The fgetc() function reads a character from the input file referenced by fp.
2. The return value is the character read, or in case of any error, it returns EOF.
3. The following function allows to read a string from a stream:

**char \*fgets( char \*buf, int n, FILE \*fp );**

1. The functions fgets() reads up to n-1 characters from the input stream referenced by fp.
2. It copies the read string into the buffer buf, appending a null character to terminate the string. If this function encounters a newline character '\n' or the end of the file EOF before they have read the maximum number of characters, then it returns only the characters read up to that point including the new line character.

3. You can also use `int fscanf(FILE *fp, const char *format, ...)` function to read strings from a file, but it stops reading after encountering the first space character.

```
#include <stdio.h>
main() {
FILE *fp;
char buff[255];
fp = fopen("/tmp/test.txt", "r");
fscanf(fp, "%s", buff);
printf("1 : %s\n", buff );
fgets(buff, 255, (FILE*)fp);
printf("2: %s\n", buff );
fgets(buff, 255, (FILE*)fp);
printf("3: %s\n", buff );
fclose(fp);
}
```

When the above code is compiled and executed, it reads the file created in the previous section and produces the following result:

Output:

1 : This  
2: is testing for fprintf...  
3: This is testing for fputs...

**Video Content/ Details of website for further learning (if any):**

<https://www.youtube.com/watch?v=lnmGb8AmS6Q>

**Important Books/Journals for further learning including the page nos.:**

1. Computer Fundamentals and Programming in C - Reema Thareja: Oxford University Press, Second Edition. Page no.417
2. Programming in C – Reema Thareja, Oxford University Press, Second Edition. Page no.290-291.

**Course Faculty**

**Verified by HOD**



# MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)  
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L-44

CSE

I / I

Course Name with Code : Programming for Problem Solving using C-19GES01

Course Faculty : M.Ganthimathi

Unit : V-Pointers and File Processing Date of Lecture:

**Topic of Lecture:** Writing data to Files

**Introduction :**

File is collection of data. There are two types of files Text file and Binary File. Data can be written into and can be accessed from a file. A file need to be opened for writing, then data is written into the file, and then the file is closed. There are appropriate functions for reading from and writing data into files.

**Prerequisite knowledge for Complete understanding and learning of Topic:**

- C Fundamentals
- Statements in C
- Functions
- Pointers
- Files

**Detailed content of the Lecture:**

The file is opened before writing, then the contents are written into the file and then the file is closed.

**PROGRAM 1**

```
#include <stdio.h>

int main()
{ char ch;
  FILE *fpw;
```



```

fpw = fopen("C:\\newfile.txt","w");

if(fpw == NULL)
{ printf("Error");
  exit(1);
}
printf("Enter any character: ");
scanf("%c",&ch);
/* You can also use fputc(ch, fpw);*/
fprintf(fpw,"%c",ch);
fclose(fpw);
return 0;
}

```

## PROGRAM 2

```

#include <stdio.h>
int main()
{ char ch;
  FILE *fpr, *fpw;    /* Pointer for both the file*/ /* Opening file FILE1.C in "r" mode for
reading */
  fpr = fopen("C:\\file1.txt", "r");/* Ensure FILE1.C opened successfully*/
  if (fpr == NULL)
  { puts("Input file cannot be opened  } /* Opening file FILE2.C in "w" mode for
writing*/
  fpw= fopen("C:\\file2.txt", "w"); /* Ensure FILE2.C opened successfully*/
  if (fpw == NULL)
  { puts("Output file cannot be opened"); }
  while(1)          /*Read & Write Logic*/
  {

```

```
ch = fgetc(fpr);
if (ch==EOF)
    break;
else
    fputc(ch, fpw);
} /* Closing both the files */
fclose(fpr);
fclose(fpw);
return 0;
}
```

**Video Content / Details of website for further learning (if any):**

[www.youtube.com/watch?v=38I\\_AUMpKpQhttps](https://www.youtube.com/watch?v=38I_AUMpKpQ)

**Important Books/Journals for further learning including the page nos.:**

1. Computer Fundamentals and Programming in C - Reema Thareja: Oxford University Press, Second Edition. Page no.418-421
2. Programming in C – Reema Thareja, Oxford University Press, Second Edition. Page no.294-299.

**Course Faculty**

**Verified by HOD**





# MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)  
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L-45

CSE

I / I

Course Name with Code : Programming for Problem Solving using C-19GES01

Course Faculty : M.Ganthimathi

Unit : V-Pointers and File Processing Date of Lecture:

**Topic of Lecture:** Operations on files

**Introduction :**

File is collection of data. There are two types of files Text file and Binary File. Data can be written into and can be accessed from a file. A file need to be opened for writing, then data is written into the file, and then the file is closed. There are appropriate functions for reading from and writing data into files. Records can be newly added into the file

**Prerequisite knowledge for Complete understanding and learning of Topic:**

- C Fundamentals
- Statements in C
- Functions
- Pointers
- Files

**Detailed content of the Lecture:**

1. Input file path from user to append data, store it in some variable say filePath.
2. Declare a FILE type pointer variable say, fPtr.
3. Open file in a (append file) mode and store referenc to fPtr using `fPtr = fopen(filePath, "a");`.
4. Input data to append to file from user, store it to some variable say dataToAppend.
5. Write data to append into file using `fputs(dataToAppend, fPtr);`.
6. Finally close file to save all changes. Use `fclose(fPtr);`.

```

#include <stdio.h>
int main()
{ FILE *fp;
  char ch;
  char *filename = "file_append.txt";
  char *content = "This text is appended later to the file, using C programming.";
  fp = fopen(filename, "r");          /* open for writing */
  printf("\nContents of %s -\n\n", filename);
  while ((ch = fgetc(fp)) != EOF)
  { printf ("%c", ch); }
  fclose(fp);
  fp = fopen(filename, "a");
  fprintf(fp, "%s\n", content); /* Write content to file */
  fclose(fp);
  fp = fopen(filename, "r");
  printf("\nContents of %s -\n", filename);
  while ((ch = fgetc(fp)) != EOF)
  { printf ("%c", ch); }
  fclose(fp);
  return 0;
}

```

### Output

Contents of file\_append.txt

This text was already there in the file.

Appending content to file\_append.txt...

Content of file\_append.txt after 'append' operation is -

This text was already there in the file.

This text is appended later to the file, using C programming.

### **Video Content / Details of website for further learning (if any):**

[:/ /www.youtube.com/watch?v=Hxhbp1WSDJA](https://www.youtube.com/watch?v=Hxhbp1WSDJA)

### **Important Books/Journals for further learning including the page nos.:**

1.Computer Fundamentals and Programming in C - Reema Thareja: Oxford University Press, Second Edition. Page no.418-421

2.Programming in C – Reema Thareja, Oxford University Press, Second Edition. Page no.300.

**Course Faculty**

**Verified by HOD**