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

Estd. 2000

IQAC

LECTURE HANDOUTS

L - 1

CSE

I/II

Course Name with Code : 16CSC02 & Advanced C Programming

Course Faculty : N.Anand

Unit : I-Arrays **Date of Lecture:**

Topic of Lecture: Declaring and initializing One-Dimensional Array

Introduction :

- One dimensional array is an array that has only one subscript specification that is needed to specify a particular element of an array. A one-dimensional array is a structured collection of components (often called array elements) that can be accessed individually by specifying the position of a component with a single index value.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Basics of Computer
- Programming Skill
- Coding Knowledge

Detailed content of the Lecture:

One Dimensional Array in C with Examples

One Dimensional Array in C with Examples. Please read our previous articles, where we discussed the basics of **Array in C** Langauge.

One Dimensional Array in C:

Syntax: data-type arr_name[array_size];

Rules for Declaring One Dimensional Array

1. An array variable must be declared before being used in a program.
2. The declaration must have a data type(int, float, char, double, etc.), variable name, and subscript.
3. The subscript represents the size of the array. If the size is declared as 10, programmers can store 10 elements.
4. An array index always starts from 0. For example, if an array variable is declared as s[10], then it ranges from 0 to 9.
5. Each array element stored in a separate memory location.

Initialization of One-Dimensional Array in C

An array can be initialized at either following states:

1. At compiling time (static initialization)
2. Dynamic Initialization

Compiling time initialization:

The compile-time initialization means the array of the elements are initialized at the time the program is written or array declaration.

Syntax: data_type array_name [array_size]=(list of elements of an array);

Example: int n[5]={0, 1, 2, 3, 4};

Program:

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

```c
int main()
{
int n[5]={0, 1, 2, 3, 4};
printf("%d", n[0]);
printf("%d", n[1]);
printf("%d", n[2]);
printf("%d", n[3]);
printf("%d", n[4]);
}
```

Output: 0 1 2 3 4

Video Content / Details of website for further learning (if any):
https://dotnettutorials.net/lesson/one-dimensional-array-in-c/
http://www.griet.ac.in/nodes/UNIT-III(QA)_cp.pdf

Important Books/Journals for further learning including the page nos.:
E Balagurusamy," Programming in ANSI C", Tata McGraw Hill, 2012
Yuksel Uckan, "Problem Solving Using C", McGraw Hill, 1999

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

Estd. 2000

IQAC

LECTURE HANDOUTS

L - 2

CSE

I/II

Course Name with Code : 16CSC02 & Advanced C Programming

Course Faculty : N.Anand

Unit : I-Arrays **Date of Lecture:**

Topic of Lecture: Array Operations

Introduction :

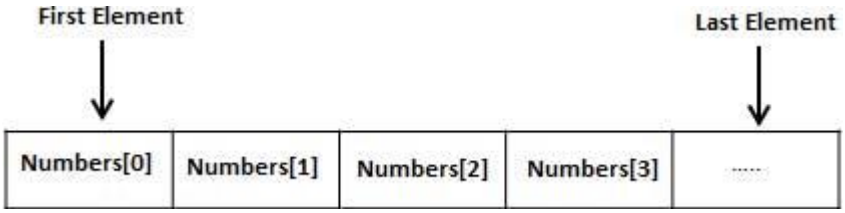
- Arrays a kind of data structure that can store a fixed-size sequential collection of elements of the same type. An array is used to store a collection of data, but it is often more useful to think of an array as a collection of variables of the same type.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Basics of Computer
- Programming Skill
- Coding Knowledge

Detailed content of the Lecture:

All arrays consist of contiguous memory locations. The lowest address corresponds to the first element and the highest address to the last element.



Declaring Arrays

To declare an array in C, a programmer specifies the type of the elements and the number of elements required by an array as follows −

type arrayName [arraySize];

This is called a *single-dimensional* array. The **arraySize** must be an integer constant greater than zero and **type** can be any valid C data type. For example, to declare a 10-element array called **balance** of type double, use this statement −

double balance[10];

Here *balance* is a variable array which is sufficient to hold up to 10 double numbers.

Initializing Arrays

You can initialize an array in C either one by one or using a single statement as follows −

double balance[5] = {1000.0, 2.0, 3.4, 7.0, 50.0};

The number of values between braces { } cannot be larger than the number of elements that we declare for the array between square brackets [ ].

If you omit the size of the array, an array just big enough to hold the initialization is created. Therefore, if you write −

double balance[] = {1000.0, 2.0, 3.4, 7.0, 50.0};

You will create exactly the same array as you did in the previous example. Following is an example to assign a single element of the array −

balance[4] = 50.0;

The above statement assigns the 5th element in the array with a value of 50.0. All arrays have 0 as the index of their first element which is also called the base index and the last index of an array will be total size of the array minus 1. Shown below is the pictorial representation of the array we discussed above −

	0	1	2	3	4
balance	1000.0	2.0	3.4	7.0	50.0

Accessing Array Elements

An element is accessed by indexing the array name. This is done by placing the index of the element within square brackets after the name of the array. For example −

double salary = balance[9];

Video Content / Details of website for further learning (if any):
https://www.tutorialspoint.com/cprogramming/c_arrays.html
https://www.studymite.com/blog/operation-on-arrays-in-c-1/

Important Books/Journals for further learning including the page nos.:
E Balagurusamy," Programming in ANSI C", Tata McGraw Hill, 2012
Yuksel Uckan, "Problem Solving Using C", McGraw Hill, 1999

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/II

Course Name with Code : 16CSC02 & Advanced C Programming

Course Faculty : N.Anand

Unit : I-Arrays **Date of Lecture:**

Topic of Lecture: Two-Dimensional Array and its Operation, Insertion, Deletion

Introduction :

- Arrays can be defined as collection of elements or data that are of similar or different data types, which is implemented in one or more dimensions with respect to the requirement provided to the program developer. 2-D or two dimensional array are represented as 'datatype variable[n][n]', where datatype can be an int, char, etc, and the [n][n] is n*n to represent the position of the value of the variable in the array. A few basic operations necessary for all the two dimensional array are 'initializing the array', 'inserting the value in the array', 'updating the value in the array', and 'deleting a value from the array'. In this article will see about 2-D Arrays in C.

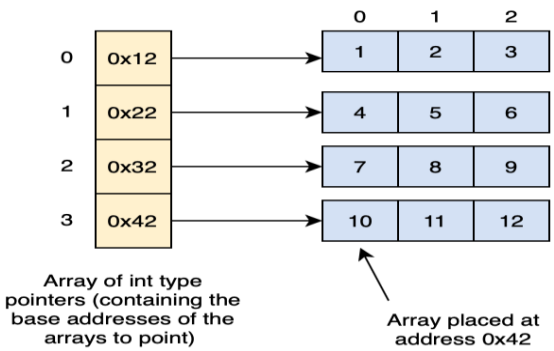
Prerequisite knowledge for Complete understanding and learning of Topic:

- Basics of Computer
- Programming Skill
- Coding Knowledge

Detailed content of the Lecture:

A Two Dimensional Array is an array of *references* that holds references to other arrays. These arrays are preferably used if you want to put together data items in a table or matrix-like structure. Matrices are widely used in the field of Game Development, where you are required to store and update the location of the player at each second.

Take a look at the figure below to get a good understanding of what a *Two Dimensional Array* looks like:



Explanation: If we take a look at the structure of the Two Dimensional Array, we get the idea that a Two Dimensional Array is basically an array of One Dimensional Array; in other words, every element points to the first element of a different One Dimensional Array.

It is important to note that in 2D arrays, all values must have the same data type. This means that you can't store an array of integers next to an array of strings and vice versa. For example, if one array is declared of type int, then its pointer can't point to the string type array. Each element must be of the same data type.

Ex:

```
class TwoDArray {
    public static void main( String args[] ) {
        int [][] my2DArray = new int[3][4];

      // add "10" at Row 0 and Column 1
        my2DArray[0][1] = 10;
        System.out.println(my2DArray[0][1]);
    }
}
```

Output: 10

Video Content / Details of website for further learning (if any):
https://www.tutorialspoint.com/data_structures_algorithms/array_data_structure.htm
https://www.educba.com/2-d-arrays-in-c/
https://www.guru99.com/array-data-structure.html

Important Books/Journals for further learning including the page nos.:
E Balagurusamy," Programming in ANSI C", Tata McGraw Hill, 2012
Yuksel Uckan, "Problem Solving Using C", McGraw Hill, 1999

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

Estd. 2000

IQAC

LECTURE HANDOUTS

L - 4

CSE

I/II

Course Name with Code : 16CSC02 & Advanced C Programming

Course Faculty : N.Anand

Unit : I-Arrays **Date of Lecture:**

Topic of Lecture: Matrix addition operation

Introduction :

- Matrix addition is the operation of adding two matrices by adding the corresponding entries together. However, there are other operations which could also be considered addition for matrices, such as the direct sum and the Kronecker sum.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Basics of Computer
- Programming Skill
- Coding Knowledge

Detailed content of the Lecture:

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. For example, if the order is 2, 2, i.e., two rows and two columns and the matrices are:

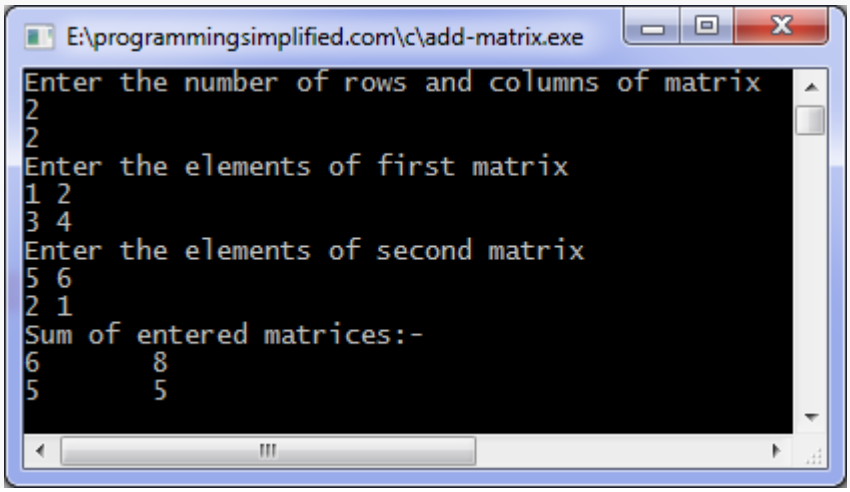
First matrix:
1 2
3 4
Second matrix:
4 5
-1 5
The output is:
5 7
2 9

Addition of two matrix in C

C program for matrix addition:

```
#include <stdio.h>

int main()
{
  int m, n, c, d, first[10][10], second[10][10], sum[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("Sum of entered matrices:-\n");
  for (c = 0; c < m; c++) {
    for (d = 0 ; d < n; d++) {
      sum[c][d] = first[c][d] + second[c][d];
      printf("%d\t", sum[c][d]);
    }
    printf("\n");
  }

  return 0;
}
```



Video Content / Details of website for further learning (if any):
https://en.wikipedia.org/wiki/Matrix_addition#:~:text=In%20mathematics%2C%20matrix%20addition%20is,sum%20and%20the%20Kronecker%20sum.
https://stattrek.com/matrix-algebra/matrix-addition.aspx

Important Books/Journals for further learning including the page nos.:
E Balagurusamy," Programming in ANSI C", Tata McGraw Hill, 2012
Yuksel Uckan, "Problem Solving Using C", McGraw Hill, 1999

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

Estd. 2000

IQAC

LECTURE HANDOUTS

L - 5

CSE

I/II

Course Name with Code : 16CSC02 & Advanced C Programming

Course Faculty : N.Anand

Unit : I-Arrays **Date of Lecture:**

Topic of Lecture: Multi-Dimensional Arrays

Introduction :

- A multi-dimensional array is **an array of arrays**. 2-dimensional arrays are the most commonly used. They are used to store data in a tabular manner.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Basics of Computer
- Programming Skill
- Coding Knowledge

Detailed content of the Lecture:

Multidimensional arrays in simple words as an array of arrays. Data in multidimensional arrays are stored in tabular form (in row-major order). The general form of declaring N-dimensional arrays:

data_type array_name[size1][size2]....[sizeN];

data_type: Type of data to be stored in the array.
Here data_type is valid C/C++ data type
array_name: Name of the array
size1, size2,... ,sizeN: Sizes of the dimensions
Examples:

Take a step-up from those "Hello World" programs. Learn to implement data structures like Heap, Stacks, Linked List and many more! Check out our **Data Structures in C** course to start learning today.

Two dimensional array:

```
int two_d[10][20];
```

Three dimensional array:

```
int three_d[10][20][30];
```

Size of multidimensional arrays

The total number of elements that can be stored in a multidimensional array can be calculated by multiplying the size of all the dimensions. For example: The array **int x[10][20]** can store total (10*20) = 200 elements. Similarly array **int x[5][10][20]** can store total (5*10*20) = 1000 elements.

Two-Dimensional Array

Two – dimensional array is the simplest form of a multidimensional array. We can see a two – dimensional array as an array of one – dimensional array for easier understanding.

- The basic form of declaring a two-dimensional array of size x, y: **Syntax:**

data_type array_name[x][y];

data_type: Type of data to be stored. Valid C/C++ data type.

- We can declare a two-dimensional integer array say 'x' of size 10,20 as:

int x[10][20];

Initializing Two – Dimensional Arrays: There are two ways in which a Two-Dimensional array can be initialized.

First Method:

int x[3][4] = {0, 1 ,2 ,3 ,4 , 5 , 6 , 7 , 8 , 9 , 10 , 11}

The above array has 3 rows and 4 columns. The elements in the braces from left to right are stored in the table also from left to right. The elements will be filled in the array in order, the first 4 elements from the left in the first row, the next 4 elements in the second row

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

https://www.geeksforgeeks.org/multidimensional-arrays-c-cpp/

https://www.mathworks.com/help/matlab/math/multidimensional-arrays.html

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

E Balagurusamy," Programming in ANSI C", Tata McGraw Hill, 2012

Yuksel Uckan, "Problem Solving Using C", McGraw Hill, 1999

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

Estd. 2000

IQAC

LECTURE HANDOUTS

L - 6

CSE

I/II

Course Name with Code : 16CSC02 & Advanced C Programming

Course Faculty : N.Anand

Unit : I-Arrays **Date of Lecture:**

Topic of Lecture: Drawbacks of Linear Arrays.

Introduction :

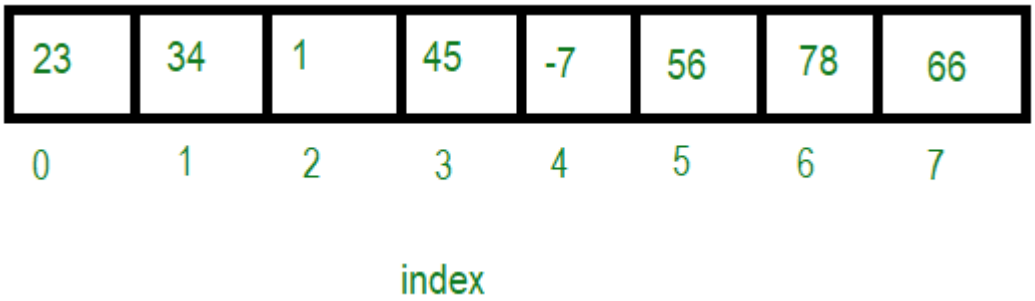
- An array is a collection of similar types of elements. For example, an integer array holds the elements of int types while a character array holds the elements of char types.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Basics of Computer
- Programming Skill
- Coding Knowledge

Detailed content of the Lecture:

An array is a collection of similar types of elements. For example, an integer array holds the elements of int types while a character array holds the elements of char types. Below is the representation of the array:



We provide nothing but the best curated videos and practice problems for our students. Check out the C Foundation Course and master the C language from basic to advanced level. Wait no more, start learning today!

Though, array got its own set of advantages and disadvantages.

Advantages of Arrays

Below are some advantages of the array:

- In an array, accessing an element is very easy by using the index number.
- The search process can be applied to an array easily.
- 2D Array is used to represent matrices.
- For any reason a user wishes to store multiple values of similar type then the Array can be used and utilized efficiently.

Disadvantages of Arrays

Now let's see some disadvantages of the array and how to overcome it: Array size is fixed: The array is static, which means its size is always fixed. The memory which is allocated to it cannot be increased or decreased.
Video Content / Details of website for further learning (if any):
Important Books/Journals for further learning including the page nos.: E Balagurusamy," Programming in ANSI C", Tata McGraw Hill, 2012 Yuksel Uckan, "Problem Solving Using C", McGraw Hill, 1999

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/II

Course Name with Code : 16CSC02 & Advanced C Programming

Course Faculty : N.Anand

Unit : II- Pointers & Preprocessor Directives **Date of Lecture:**

Topic of Lecture: Pointers - Introduction and Features of Pointers

Introduction :

- Pointers store address of variables or a memory location.

Syntax:

Take a step-up from those "Hello World" programs. Learn to implement data structures like Heap, Stacks, Linked List and many more! **Data Structures in C** course to start learning today.

datatype *var_name;

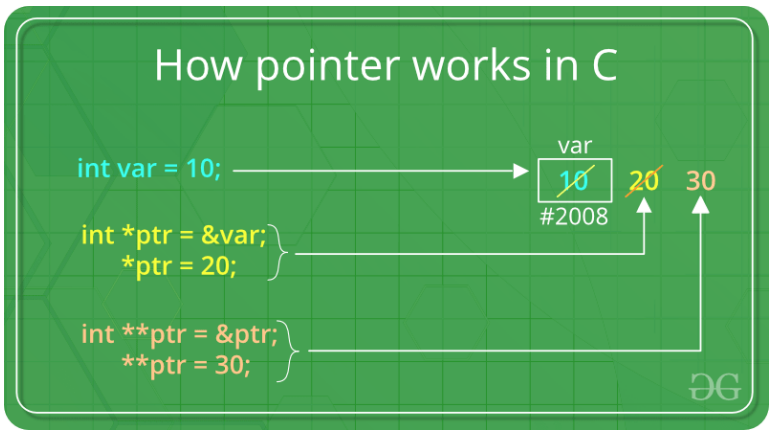
Prerequisite knowledge for Complete understanding and learning of Topic:

- Data Structure
- C Language
- Computer Knowledge

Detailed content of the Lecture:

pointer "ptr" holds address of an integer variable or holds address of a memory whose value(s) can be accessed as integer values through "ptr"

int *ptr;



Features of Pointers:

1. Pointers save memory space.
2. Execution time with pointers is faster because data are manipulated with the address, that is, direct access to memory location.
3. Memory is accessed efficiently with the pointers. The pointer assigns and releases the memory as well. Hence it can be said the Memory of pointers is dynamically allocated.

4. Pointers are used with data structures. They are useful for representing two-dimensional and multi-dimensional
   arrays.
5. An array, of any type can be accessed with the help of pointers, without considering its subscript range.
6. Pointers are used for file handling.
7. Pointers are used to allocate memory dynamically.
8. In C++, a pointer declared to a base class could access the object of a derived class. However, a pointer to a derived class cannot access the object of a base class.

Uses of pointers:

1. To pass arguments by reference
2. For accessing array elements
3. To return multiple values
4. Dynamic memory allocation
5. To implement data structures
6. To do system level programming where memory addresses are useful

Video Content / Details of website for further learning (if any):
https://www.geeksforgeeks.org/features-and-use-of-pointers-in-c-c/
https://www.studytonight.com/c/pointers-in-c.php

Important Books/Journals for further learning including the page nos.:
E Balagurusamy," Programming in ANSI C", Tata McGraw Hill, 2012
Yuksel Uckan, "Problem Solving Using C", McGraw Hill, 1999

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/II

Course Name with Code : 16CSC02 & Advanced C Programming

Course Faculty : N.Anand

Unit : II- Pointers & Preprocessor Directives **Date of Lecture:**

Topic of Lecture: Declaration of Pointer- Void Pointers

Introduction :

- A void pointer is a pointer that has no associated data type with it. A void pointer can hold address of any type and can be typcasted to any type.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Data Structure
- C Language
- Computer Knowledge

Detailed content of the Lecture:

Take a step-up from those "Hello World" programs. Learn to implement data structures like Heap, Stacks, Linked List and many more! Check out our Data Structures in C course to start learning today.

```
int a = 10;
char b = 'x';

void *p = &a;  // void pointer holds address of int 'a'
p = &b; // void pointer holds address of char 'b'
```

Advantages of void pointers:
1) malloc() and calloc() return void * type and this allows these functions to be used to allocate memory of any data type (just because of void *)

```
int main(void)
{
   // Note that malloc() returns void * which can be
   // typecasted to any type like int *, char *, ..
   int *x = malloc(sizeof(int) * n);
}
```

Note that the above program compiles in C, but doesn't compile in C++. In C++, we must explicitly typecast return value of malloc to (int *).

2) void pointers in C are used to implement generic functions in C. For example compare function which is used in qsort().
Some Interesting Facts:
1) void pointers cannot be dereferenced. For example the following program doesn't compile.

```
#include<stdio.h>
int main()
{
    int a = 10;
    void *ptr = &a;
    printf("%d", *ptr);
    return 0;
}
```

Output:

Compiler Error: 'void*' is not a pointer-to-object type

The following program compiles and runs fine.

```
#include<stdio.h>
int main()
{
    int a = 10;
    void *ptr = &a;
    printf("%d", *(int *)ptr);
    return 0;
}
```

Output:

10

2) The C standard doesn't allow pointer arithmetic with void pointers. However, in GNU C it is allowed by considering the size of void is 1. For example the following program compiles and runs fine in gcc.

```
#include<stdio.h>
int main()
{
    int a[2] = {1, 2};
    void *ptr = &a;
    ptr = ptr + sizeof(int);
    printf("%d", *(int *)ptr);
    return 0;
}
```

Output:

2

Video Content / Details of website for further learning (if any): **https://www.tutorialspoint.com/void-pointer-in-c#:~:text=The%20void%20pointer%20in%20C,return%20void%20*%20or%20generic%20pointers.** **https://www.geeksforgeeks.org/void-pointer-c-cpp/**
Important Books/Journals for further learning including the page nos.: E Balagurusamy," Programming in ANSI C", Tata McGraw Hill, 2012 Yuksel Uckan, "Problem Solving Using C", McGraw Hill, 1999

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

Estd. 2000

IQAC

LECTURE HANDOUTS

L - 9

CSE

I/II

Course Name with Code : 16CSC02 & Advanced C Programming

Course Faculty : N.Anand

Unit : II- Pointers & Preprocessor Directives **Date of Lecture:**

Topic of Lecture: Array of Pointers- Pointers to Pointers

Introduction :

- In computer programming, an array of pointers is an indexed set of variables, where the variables are pointers (referencing a location in memory). ... The value of each integer is printed by dereferencing the pointers. In other words, this code prints the value in memory of where the pointers point.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Data Structure
- C Language
- Computer Knowledge

Detailed content of the Lecture:

- The pointer to an array of five integers. We use parenthesis to pronounce pointer to an array. Since subscript has higher priority than indirection, it is crucial to encase the indirection operator and pointer name inside brackets.

```
// C program to demonstrate
// pointer to an array.

#include <stdio.h>

int main()
{

        // Pointer to an array of five numbers
        int(*a)[5];

        int b[5] = { 1, 2, 3, 4, 5 };

        int i = 0;

        // Points to the whole array b

        a = &b;

        for (i = 0; i < 5; i++)
```

```
        printf("%d\n", *(*a + i));

    return 0;
}
```

Output:

1

2

3

4

5

Video Content / Details of website for further learning (if any):
https://www.geeksforgeeks.org/pointer-array-array-pointer/
http://books.gigatux.nl/mirror/cinanutshell/0596006977/cinanut-CHP-9-SECT-4.html

Important Books/Journals for further learning including the page nos.:
E Balagurusamy," Programming in ANSI C", Tata McGraw Hill, 2012
Yuksel Uckan, "Problem Solving Using C", McGraw Hill, 1999

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

Estd. 2000

IQAC

LECTURE HANDOUTS

L - 10

CSE

I/II

Course Name with Code : 16CSC02 & Advanced C Programming

Course Faculty : N.Anand

Unit : II- Pointers & Preprocessor Directives **Date of Lecture:**

Topic of Lecture: Introduction - #define and #undef Directives- #include ,#line Directive

Introduction :

- Preprocessor directives, such as #define and #ifdef , are typically used to make source programs easy to change and easy to compile in different execution environments. Directives in the source file tell the preprocessor to take specific actions

Prerequisite knowledge for Complete understanding and learning of Topic:

- Data Structure
- C Language
- Computer Knowledge

Detailed content of the Lecture:

Pre-Processor:

Take a step-up from those "Hello World" programs. Learn to implement data structures like Heap, Stacks, Linked List and many more! Check out our **Data Structures in C** course to start learning today.

- Pre-processor is a program that performs before the compilation.
- It only notices the # started statement.
- # is called preprocessor directive.
- Each preprocessing directive must be on its own line.
- The word after # is called the preprocessor command.

#define:

The #define directive defines an identifier and a character sequence (a set of characters) that will be substituted for the identifier each time it is encountered in the source file.

Syntax:

#define macro-name char-sequence.

The identifier is referred to as a macro name and replacement process as a macro replacement.

Example:

#define PI 3.14

Here. PI is the macro-name and 3.14 is the char-sequence.

Video Content / Details of website for further learning (if any): https://codeforwin.org/2018/11/c-preprocessor-directives-include-define-undef-conditional-directives.html https://www.geeksforgeeks.org/define-vs-undef-in-c-language/
Important Books/Journals for further learning including the page nos.: E Balagurusamy," Programming in ANSI C", Tata McGraw Hill, 2012 Yuksel Uckan, "Problem Solving Using C", McGraw Hill, 1999

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

Estd. 2000

IQAC

LECTURE HANDOUTS

L - 11

CSE

I/II

Course Name with Code : 16CSC02 & Advanced C Programming

Course Faculty : N.Anand

Unit : II- Pointers & Preprocessor Directives **Date of Lecture:**

Topic of Lecture: Predefined macros in ANSI C- Standard I/O Predefined Streams in stdio.h

Introduction :

- The **stdio.h** header defines three variable types, several macros, and various functions for performing input and output.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Data Structure
- C Language
- Computer Knowledge

Detailed content of the Lecture:

Library Variables

Following are the variable types defined in the header stdio.h −

Sr.No.	Variable & Description
1	**size_t** This is the unsigned integral type and is the result of the **sizeof** keyword.
2	**FILE** This is an object type suitable for storing information for a file stream.
3	**fpos_t** This is an object type suitable for storing any position in a file.

Library Macros

Following are the macros defined in the header stdio.h −

Sr.No.	Macro & Description
1	**NULL** This macro is the value of a null pointer constant.
2	**_IOFBF, _IOLBF** and **_IONBF** These are the macros which expand to integral constant expressions with distinct values and suitable for the use as third argument to the **setvbuf** function.
3	**BUFSIZ** This macro is an integer, which represents the size of the buffer used by the **setbuf** function.
4	**EOF** This macro is a negative integer, which indicates that the end-of-file has been reached.
5	**FOPEN_MAX** This macro is an integer, which represents the maximum number of files that the system can guarantee to be opened simultaneously.
6	**FILENAME_MAX** This macro is an integer, which represents the longest length of a char array suitable for holding the longest possible filename. If the implementation imposes no limit, then this value should be the recommended maximum value.
7	**L_tmpnam** This macro is an integer, which represents the longest length of a char array suitable for holding the longest possible temporary filename created by the **tmpnam** function.
8	**SEEK_CUR, SEEK_END,** and **SEEK_SET** These macros are used in the **fseek** function to locate different positions in a file.
9	**TMP_MAX** This macro is the maximum number of unique filenames that the function **tmpnam** can generate.

10	**stderr, stdin,** and **stdout** These macros are pointers to FILE types which correspond to the standard error, standard input, and standard output streams.

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

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

E Balagurusamy," Programming in ANSI C", Tata McGraw Hill, 2012

Yuksel Uckan, "Problem Solving Using C", McGraw Hill, 1999

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/II

Course Name with Code : 16CSC02 & Advanced C Programming

Course Faculty : N.Anand

Unit : II- Pointers & Preprocessor Directives **Date of Lecture:**

Topic of Lecture: Predefined macros in ctype.h

Introduction :

- h header file contains **inbuilt functions to handle Strings in C/C++**, the ctype. h/<cctype> contains inbuilt functions to handle characters in C/C++ respectively. Characters are of two types: Printable Characters: The characters that are displayed on the terminal.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Data Structure
- C Language
- Computer Knowledge

Detailed content of the Lecture:

As string.h header file contains inbuilt functions to handle Strings in C/C++, the ctype.h/<cctype> contains inbuilt functions to handle characters in C/C++ respectively.
Characters are of two types:

Take a step-up from those "Hello World" programs. Learn to implement data structures like Heap, Stacks, Linked List and many more! Check out our Data Structures in C course to start learning today.

1. Printable Characters: The characters that are displayed on the terminal.
2. Control Characters: The characters that are initiated to perform a specific operation.

The arguments passed to character functions should be of integer type. If we pass characters instead of an integer, the characters are typecasted into integers(corresponding ASCII values) and those integers are passed as arguments.

The below functions under ctype.h/<cctype> header file are applied on normal characters. Wide character functions are used for the characters of type wchar_t.

S.No	Function	Description	Return Values
1.	isalnum()	This function identifies the alphanumeric characters	Returns 0 if the passed argument is non – alphanumeric character Returns non zero value if the passed argument is alphanumeric character

2.	isalpha()	This function identifies the alphabets from other characters	Returns 0 if the passed argument is not an alphabet Returns non zero value if the passed argument is an alphabet
3.	isblank()	This function identifies the blank spaces from other characters	Returns 0 if the passed argument is not a blank space Returns nonzero value if the passed argument is a blank space
4.	iscntrl()	This function identifies the control characters(\n, \b, \t, \r).	Returns 0 if the passed argument is not a control character Returns nonzero value if the passed argument is a control character
5.	isdigit()	This function identifies numbers in character.	Returns 0 if the passed argument is not a number Returns nonzero value if the passed argument is a number
6.	islower()	This function identifies the lowercase alphabets.	Returns 0 if the passed argument is not a lowercase alphabet Returns nonzero value if the passed argument is a lowercase alphabet
7.	isprint()	This function identifies the printable characters.	Returns 0 if the passed argument is a non printable character Returns nonzero value if the passed argument is a printable character
8.	ispunct()	This function identifies punctuation characters (characters that are neither alphanumeric nor space).	Returns 0 if the passed argument is not a punctuation character Returns nonzero value if the passed argument is a punctuation character
9.	isspace()	This function identifies white-space characters.	Returns 0 if the passed argument is not a white-space character Returns nonzero value if the passed argument is a white-space character
10.	isupper()	This function identifies the uppercase alphabets.	Returns 0 if the passed argument is not an uppercase alphabet Returns nonzero value if the passed argument is an uppercase alphabet

11.	isxdigit()	This function identifies the hexadecimal digit.	Returns 0 if the passed argument is not a hexadecimal digit Returns nonzero value if the passed argument is an hexadecimal digit
12.	tolower()	This function converts uppercase alphabet to lowercase alphabet.	Returns lowercase alphabet of the corresponding uppercase alphabet
13.	toupper()	This function convert lowercase alphabet to uppercase alphabet.	Returns uppercase alphabet of the corresponding lowercase alphabet

Video Content / Details of website for further learning (if any):
https://fresh2refresh.com/c-programming/c-function/c-ctype-h-library-functions/
https://www.geeksforgeeks.org/ctype-hcctype-library-in-c-c-with-examples/

Important Books/Journals for further learning including the page nos.:
E Balagurusamy," Programming in ANSI C", Tata McGraw Hill, 2012
Yuksel Uckan, "Problem Solving Using C", McGraw Hill, 1999

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 - 13

CSE

I/II

Course Name with Code : 16CSC02 & Advanced C Programming

Course Faculty : N.Anand

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

Topic of Lecture: Basics of Functions - Built-in and user defined Functions

Introduction :

- A large program is subdivided into a number of smaller programs or subprograms. Each subprogram specifies one or more actions to be performed for a large program. such subprograms are functions. ... Built-in functions that predefined and supplied along with the compiler are known as built-in functions.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Function Call
- Basics of function
- String

Detailed content of the Lecture:

Library function:

These function are the built-in functions i.e., they are predefined in the library of the C. These are used to perform the most common operations like calculations, updatation, etc. Some of the library functions are printf, scanf, sqrt, etc. To use this functions in the program the user have to use associate header file associated to the corresponding function in the program.

For Example:

If, the user have to use print the data or scan the data using input stream then we have to use functions printf() and scanf() in C program and cin and cout in C++ program. To use these functions the user have to include #include<stdio.h> preprocesser directive in C program and #include<iostream> preprocesser directive in C++ program.

Want to learn from the best curated videos and practice problems, check out the C++ Foundation Course for Basic to Advanced C++ and C++ STL Course for the language and STL. To complete your preparation from learning a language to DS Algo and many more, please refer Complete Interview Preparation Course.

```
// C program to illustrate inbuilt function
#include <stdio.h>

// Driver Code
int main()
{

        // Print Statement
        printf("GeeksforGeeks!");
```

```
        return 0;
}
```

Output:
GeeksforGeeks!

User-defined Functions	Library Functions
These functions are not predefined in the Compiler.	These functions are predefined in the compiler of C language.
These function are created by user as per their own requirement.	These functions are not created by user as their own.
User-defined functions are not stored in library file.	Library Functions are stored in special library file.
There is no such kind of requirement to add the particular library.	In this if the user wants to use a particular library function then the user have to add the particular library of that function in header file of the program.
Execution of the program begins from the user-define function.	Execution of the program does not begin from the library function.
Example: sum(), fact(),…etc.	**Example:** printf(), scanf(), sqrt(),…etc.

Video Content / Details of website for further learning (if any):
https://www.geeksforgeeks.org/difference-between-user-defined-function-and-library-function-in-c-c/
http://www.geekinterview.com/question_details/3434

Important Books/Journals for further learning including the page nos.:
E Balagurusamy," Programming in ANSI C", Tata McGraw Hill, 2012
Yuksel Uckan, "Problem Solving Using C", McGraw Hill, 1999

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/II

Course Name with Code : 16CSC02 & Advanced C Programming

Course Faculty : N.Anand

Unit : III : Functions **Date of Lecture:**

Topic of Lecture: Using String, Math and other built-in functions, Advantages of using Functions

Introduction :

- Built-in functions are those that are already defined in Python libraries and we can call them directly. User defined functions are those that we define ourselves in our program and then call them wherever we want.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Function Call
- Basics of function
- String

Detailed content of the Lecture:

Functions are used because of following reasons –

a) To improve the readability of code.
b) Improves the reusability of the code, same function can be used in any program rather than writing the same code from scratch.
c) Debugging of the code would be easier if you use functions, as errors are easy to be traced.
d) Reduces the size of the code, duplicate set of statements are replaced by function calls.

Types of functions

1) Predefined standard library functions
Standard library functions are also known as built-in functions. Functions such as puts(), gets(), printf(), scanf() etc are standard library functions. These functions are already defined in header files (files with .h extensions are called header files such as stdio.h), so we just call them whenever there is a need to use them.

For example, printf() function is defined in <stdio.h> header file so in order to use the printf() function, we need to include the <stdio.h> header file in our program using #include <stdio.h>.

2) User Defined functions
The functions that we create in a program are known as user defined functions or in other words you can say that a function created by user is known as user defined function.

Now we will learn how to create user defined functions and how to use them in C Programming

Syntax of a function

```
return_type function_name (argument list)
{
   Set of statements – Block of code
}
```

return_type: Return type can be of any data type such as int, double, char, void, short etc. Don't worry you will understand these terms better once you go through the examples below.

function_name: It can be anything, however it is advised to have a meaningful name for the functions so that it would be easy to understand the purpose of function just by seeing it's name.

argument list: Argument list contains variables names along with their data types. These arguments are kind of inputs for the function. For example – A function which is used to add two integer variables, will be having two integer argument.

Block of code: Set of C statements, which will be executed whenever a call will be made to the function.

Do you find above terms confusing? – Do not worry I'm not gonna end this guide until you learn all of them :)
Lets take an example – Suppose you want to create a function to add two integer variables.

Let's split the problem so that it would be easy to understand –
Function will add the two numbers so it should have some meaningful name like sum, addition, etc. For example lets take the name addition for this function.

Video Content / Details of website for further learning (if any):
https://beginnersbook.com/2014/01/c-functions-examples/
https://www.codingeek.com/tutorials/c-programming/functions-and-its-advantages-in-c-language/

Important Books/Journals for further learning including the page nos.:
E Balagurusamy," Programming in ANSI C", Tata McGraw Hill, 2012
Yuksel Uckan, "Problem Solving Using C", McGraw Hill, 1999

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 - 15

CSE

I/II

Course Name with Code : 16CSC02 & Advanced C Programming

Course Faculty : N.Anand

Unit : III : Functions **Date of Lecture:**

Topic of Lecture: Working of a Function- Declaring, Defining and calling user defined Functions

Introduction :

- User-defined functions are functions that you use to organize your code in the body of a policy. ... The syntax for a function declaration is the Function keyword followed by the name of the function and a comma-separated list of runtime parameters.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Function Call
- Basics of function
- String

Detailed content of the Lecture:

User-defined functions are functions that you use to organize your code in the body of a policy.

Once you define a function, you can call it in the same way as the built-in action and parser functions. Variables that are passed to a function are passed by reference, rather than by value. This means that changing the value of a variable within a function also changes the value of the variable in the general scope of the policy.

User-defined functions cannot return a value as a return parameter. You can return a value by defining an output parameter in the function declaration and then assigning a value to the variable in the body of the function. Output parameters are specified in the same way as any other parameter.

You can also declare your own functions and call them within a policy. User-defined functions help you encapsulate and reuse functionality in your policy.

The syntax for a function declaration is the Function keyword followed by the name of the function and a comma-separated list of runtime parameters. The list of runtime parameters is followed by a statement block that is enclosed in curly braces.

Unlike action and parser functions, you cannot specify a return value for a user-defined function. However, because the scope of variables in IPL policy is global, you can approximate this functionality by setting the value of a return variable inside the function.

Function declarations must appear in a policy before any instance where the function is called. The best

practice is to declare all functions at the beginning of a policy.

The following example shows how to declare a user-defined function called GetNodeByHostname. This function looks up a node in an external data source by using the supplied host name.

```
Function GetNodeByHostName(Hostname) {

        DataType = "Node";
        Filter = "Hostname ='" + Hostname + "'";
        CountOnly = False;

        MyNodes = GetByFilter(DataType, Filter, CountOnly);
        MyNode = MyNodes[0];
}
```

You call user-defined functions in the same way that you call other types of functions. The following example shows how to call the function.

```
GetNodeByHostName("ORA_HOST_01");
```

Here, the name of the node that you want to look up is ORA_HOST_01. The function looks up the node in the external data source and returns a corresponding data item named MyNode.

Video Content / Details of website for further learning (if any):
https://www.ibm.com/docs/SSSHYH_6.1.1.3/com.ibm.netcoolimpact.doc/common/dita/user_defined_functions_c.html
https://www.programiz.com/c-programming/c-user-defined-functions

Important Books/Journals for further learning including the page nos.:
E Balagurusamy," Programming in ANSI C", Tata McGraw Hill, 2012
Yuksel Uckan, "Problem Solving Using C", McGraw Hill, 1999

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 - 16

CSE

I/II

Course Name with Code : 16CSC02 & Advanced C Programming

Course Faculty : N.Anand

Unit : III : Functions **Date of Lecture:**

Topic of Lecture: The return Statement- Call by Value and call by Reference

Introduction :

- Call By Reference. While calling a function, **we pass values of variables to it**. Such functions are known as "Call By Values".
- While calling a function, instead of passing the values of variables, we pass address of variables(location of variables) to the function known as "Call By References.

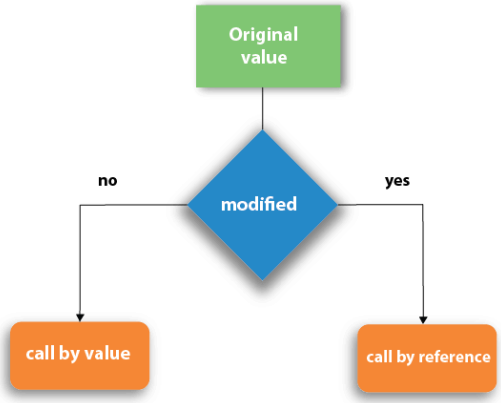
Prerequisite knowledge for Complete understanding and learning of Topic:

- Function Call
- Basics of function
- String

Detailed content of the Lecture:

Call by value and Call by reference in C :

There are two methods to pass the data into the function in C language, i.e., *call by value* and *call by reference*.



Let's understand call by value and call by reference in c language one by one.

Call by value in C

- In call by value method, the value of the actual parameters is copied into the formal parameters. In other words, we can say that the value of the variable is used in the function call in the call by value method.
- In call by value method, we can not modify the value of the actual parameter by the formal parameter.
- In call by value, different memory is allocated for actual and formal parameters since the value of the actual parameter is copied into the formal parameter.
- The actual parameter is the argument which is used in the function call whereas formal parameter is the argument which is used in the function definition.

Let's try to understand the concept of call by value in c language by the example given below:

```
#include<stdio.h>
void change(int num) {
   printf("Before adding value inside function num=%d \n",num);
   num=num+100;
   printf("After adding value inside function num=%d \n", num);
}
int main() {
   int x=100;
   printf("Before function call x=%d \n", x);
   change(x);//passing value in function
   printf("After function call x=%d \n", x);
return 0;
}
```

Output

```
Before function call x=100
Before adding value inside function num=100
After adding value inside function num=200
After function call x=100
```

Video Content / Details of website for further learning (if any): **https://www.guru99.com/call-by-value-vs-call-by-reference.html#:~:text=In%20Call%20by%20value%2C%20a,in%20the%20same%20memory%20location.** **https://www.geeksforgeeks.org/difference-between-call-by-value-and-call-by-reference/**
Important Books/Journals for further learning including the page nos.: E Balagurusamy," Programming in ANSI C", Tata McGraw Hill, 2012 Yuksel Uckan, "Problem Solving Using C", McGraw Hill, 1999

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

Estd. 2000

IQAC

LECTURE HANDOUTS

L - 17

CSE

I/II

Course Name with Code : 16CSC02 & Advanced C Programming

Course Faculty : N.Anand

Unit : III : Functions **Date of Lecture:**

Topic of Lecture: Function as an Argument- Recursion

Introduction :

- When a function calls itself, that's called a recursion step. The basis of recursion is function arguments that make the task so simple that the function does not make further calls. A recursively-defined data structure is a data structure that can be defined using itself.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Function Call
- Basics of function
- String

Detailed content of the Lecture:

A function is *recursive* if the function, in order to compute its result, ends up "calling itself". This idea can seem paradoxical when you're just getting started, so we'll go through an example of how the computer really handles it in order to explain away the paradox (the upshot is that we have the same function, yes, but it is one *call of the function* that in turn makes a separate *call* to the same function, but with different arguments). To do this, we consider a concrete example: a recursive function for computing the factorial of a number.

```
int factorial(int n) {
  if (n == 1)
    return 1;
  else
    return n * factorial(n-1);
}
```

```
int main() {
  int k = 4;
  int f = factorial(4);
  cout << f << endl;
  return 0;
}
```

The key point you need to learn with recursion is that there is no contradiction in having multiple active calls to a function (like factorial above) because each function call is essentially its own copy of the function. So we end up with a "stack" of function calls that includes many copies of the factorial function, but they're all called with different argument values.

Video Content / Details of website for further learning (if any): https://publications.gbdirect.co.uk/c_book/chapter4/recursion_and_argument_passing.html https://www.usna.edu/Users/cs/nchamber/courses/si204/s18/lec/l24/lec.html
Important Books/Journals for further learning including the page nos.: E Balagurusamy," Programming in ANSI C", Tata McGraw Hill, 2012 Yuksel Uckan, "Problem Solving Using C", McGraw Hill, 1999

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/II

Course Name with Code : 16CSC02 & Advanced C Programming

Course Faculty : N.Anand

Unit : III : Functions **Date of Lecture:**

Topic of Lecture: Advantages and Disadvantages of Recursion

Introduction :

- Recursive functions **are generally slower than** non-recursive function. 2. It may require a lot of memory space to hold intermediate results on the system stacks.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Function Call
- Basics of function
- String

Detailed content of the Lecture:

Advantages of Recursion

On the other hand, recursion has the following advantages:

- For a recursive function, you only need to define the base case and recursive case, so the code is simpler and shorter than an iterative code.
- Some problems are inherently recursive, such as Graph and Tree Traversal.

Disadvantages of Recursion

Recursion, broadly speaking, has the following disadvantages:

- A recursive program has **greater space requirements** than an iterative program as each function call will remain in the stack until the base case is reached.
- It also has **greater time requirements** because each time the function is called, the stack grows and the final answer is returned when the stack is popped completely.

Video Content / Details of website for further learning (if any): https://www.collegenote.net/curriculum/data-structures-and-algorithms/41/454/ https://www.educative.io/courses/recursion-for-coding-interviews-in-cpp/qAx1lwQYDNG
Important Books/Journals for further learning including the page nos.: E Balagurusamy," Programming in ANSI C", Tata McGraw Hill, 2012 Yuksel Uckan, "Problem Solving Using C", McGraw Hill, 1999

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 - 19

CSE

I/II

Course Name with Code : 16CSC02 & Advanced C Programming

Course Faculty : N.Anand

Unit : IV- Structure And Union **Date of Lecture:**

Topic of Lecture: Introduction and Features of Structures

Introduction :

- Structured programming is a program written with only the structured programming constructions: **(1) sequence, (2) repetition, and (3) selection**. Sequence. Lines or blocks of code are written and executed in sequential order.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Functions
- Arrays
- Structures

Detailed content of the Lecture:

Structural features

Feature	Purpose	Effect on the reader
openings	The start of a text must interest the reader.	Comment on how the writer introduces ideas and raises questions.
focus	This is what the writer focuses on as the text develops.	Analyse what is implied, eg a gloomy landscape implies an unhappy situation - what is causing that unhappiness? What will happen next?
shifts	Changes in ideas and perspectives, eg outside to inside.	Comment on how this change is effective, eg creates contrast.
contrast	The differences between two things.	Comment on the effect a drastic difference produces.
repetition or patterns	When words, phrases or ideas are repeated for effect.	Repetitive features can highlight key meanings, indicate a development or show a lack of change.

pace	The feeling of speed in the writing – are events and ideas revealed to the reader slowly or quickly?	Ask what effect is created by altering the pace, eg a slow pace builds tension or suggests boredom, a quicker pace may suit a piece about things happening at speed.
temporal references	References to time.	Comment on how time is used to speed up or slow down the pace of the text.
order of events	This could be chronological or writers might choose to start at the end, in the middle, or with flashbacks / flash forwards.	Comment on how the order of events introduces and prioritises key ideas – and how this engages the reader.
endings	The conclusion of a text may be neat or leave us with questions.	Think about how the reader feels at the end. Have their feelings changed since the opening?
withholding information	Clues and hints are given without revealing everything at once.	Analyse what is implied by hints – how does this build the reader's expectations?
dialogue	Conversations and speech.	How does dialogue move the text forward?
headings, subheadings and questions	Divides the content of texts into topics and sub topics, can signal the start of new points.	How do they guide readers through a text?
bullets	Bullets can summarise and simplify a range of ideas.	Why does the writer summarise certain points?
sentence structures	Varied types of sentences, eg simple, compound and complex.	Comment on how sentence structures affect the fluency of the text, eg a sudden short sentence could reveal shocking information.
paragraph lengths	These vary like sentences eg, to highlight significant points or to provide a detailed account.	Comment on how paragraph lengths affect the development of the text, eg a final paragraph might summarise key points in an argument.

Structure of a non-fiction text

The structure of a non-fiction piece could be:

- chronological – in date or time order
- prioritised – the most important facts first (like a news article)
- separated into blocks by subheadings – eg in a feature article
- question and answer – eg in information leaflets
- problem and solution – eg in agony aunt columns, or self-help guides
- letter structure – a salutation (Dear…) and an appropriate ending (Yours sincerely…)
- starting in the middle of an event, then providing further information to give several possible

viewpoints
Video Content / Details of website for further learning (if any): https://www.studytonight.com/c/structures-in-c.php https://www.codingeek.com/tutorials/c-programming/features-of-structures-in-c-programming-language/
Important Books/Journals for further learning including the page nos.: E Balagurusamy," Programming in ANSI C", Tata McGraw Hill, 2012 Yuksel Uckan, "Problem Solving Using C", McGraw Hill, 1999

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

I/II

Course Name with Code : 16CSC02 & Advanced C Programming

Course Faculty : N.Anand

Unit : IV- Structure And Union **Date of Lecture:**

Topic of Lecture: Declaration and Initialization of Structures

Introduction :

- Structure Initialization

  Just after structure **declaration put the braces** (i.e. {}) and inside it an equal sign (=) followed by the values must be in the order of members specified also each value must be separated by commas. The example below will show how to initialize structure variable in C programming.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Functions
- Arrays
- Structures

Detailed content of the Lecture:

Declaring structure variable along with structure declaration

Syntax

```
struct name/tag
{
   //structure members

} variables;
```

Example

```
struct car
{
     char  name[100];
     float price;
} car1;
```

Declaring structure variable using struct keyword.

Syntax

```
struct name variables;
```

Example

```
struct car
{
    char  name[100];
    float price;
};

struct car car1, car2, car3;
```

In general, we declare variables like,

<data type> variables;

Example

```
int i;
float f;
```

In structure, data type is <struct name>. So, the declaration will be

<struct name> variables;

Example

```
struct car car1;
```

Initializing structure members

We can initialize the structrue members directly like below,

Example

```
struct car
{
    char  name[100];
    float price;
};

//car1 name as "xyz"
//price as 987432.50

struct car car1 ={"xyz", 987432.50};
```

Video Content / Details of website for further learning (if any): https://www.log2base2.com/C/structure/declaration-and-initialization-of-structure-in-c.html https://www.geeksforgeeks.org/structures-c/
Important Books/Journals for further learning including the page nos.: E Balagurusamy," Programming in ANSI C", Tata McGraw Hill, 2012 Yuksel Uckan, "Problem Solving Using C", McGraw Hill, 1999

Course Faculty

Verified by HoD

LECTURE HANDOUTS

L - 21

CSE

I/II

Course Name with Code : 16CSC02 & Advanced C Programming

Course Faculty : N.Anand

Unit : IV- Structure And Union **Date of Lecture:**

Topic of Lecture: Array of Structures

Introduction :

An array of structures is simply **an array in which each element is a structure of the same type**. The referencing and subscripting of these arrays (also called structure arrays) follow the same rules as simple arrays.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Functions
- Arrays
- Structures

Detailed content of the Lecture:

An array of structure in C programming is a collection of different datatype variables, grouped together under a single name.

General form of structure declaration

The structural declaration is as follows −

```
struct tagname{
   datatype member1;
   datatype member2;
   datatype member n;
};
```

Here, **struct** is the keyword

tagname specifies name of structure

member1, member2 specifies the data items that make up structure.

Example

The following example shows the usage of array of structures in C programming −

```
struct book{
   int pages;
```

```
  char author [30];
  float price;
};
```

Array of structures

- The most common use of structure in C programming is an array of structures.
- To declare an array of structure, first the structure must be defined and then an array variable of that type should be defined.
- For Example − struct book b[10]; //10 elements in an array of structures of type ‘book’

Example

The following program shows the usage of array of structures.

```
#include <stdio.h>
#include <string.h>
struct student{
  int id;
  char name[30];
  float percentage;
};
int main(){
  int i;
  struct student record[2];
  // 1st student's record
  record[0].id=1;
  strcpy(record[0].name, "Bhanu");
  record[0].percentage = 86.5;
  // 2nd student's record
  record[1].id=2;
  strcpy(record[1].name, "Priya");
  record[1].percentage = 90.5;
  // 3rd student's record
  record[2].id=3;
  strcpy(record[2].name, "Hari");
  record[2].percentage = 81.5;
  for(i=0; i<3; i++){
    printf(" Records of STUDENT : %d \n", i+1);
    printf(" Id is: %d \n", record[i].id);
    printf(" Name is: %s \n", record[i].name);
    printf(" Percentage is: %f\n\n",record[i].percentage);
  }
  return 0;
```

```
}
```

Output

When the above program is executed, it produces the following result −

Records of STUDENT : 1

Id is: 1

Name is: Bhanu

Percentage is: 86.500000

Records of STUDENT : 2

Id is: 2

Name is: Priya

Percentage is: 90.500000

Records of STUDENT : 3

Id is: 3

Name is: Hari

Percentage is: 81.500000

Video Content / Details of website for further learning (if any):
https://www.tutorialspoint.com/explain-the-array-of-structures-in-c-language
https://www.geeksforgeeks.org/array-of-structures-vs-array-within-a-structure-in-c-and-cpp/

Important Books/Journals for further learning including the page nos.:
E Balagurusamy," Programming in ANSI C", Tata McGraw Hill, 2012
Yuksel Uckan, "Problem Solving Using C", McGraw Hill, 1999

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

Estd. 2000

IQAC

LECTURE HANDOUTS

L - 22

CSE

I/II

Course Name with Code : 16CSC02 & Advanced C Programming

Course Faculty : N.Anand

Unit : IV- Structure And Union **Date of Lecture:**

Topic of Lecture: Pointers to Structure

Introduction :

- Pointer to structure **holds the add of the entire structure**. It is used to create complex data structures such as linked lists, trees, graphs and so on. The members of the structure can be accessed using a special operator called as an arrow operator (->).

Prerequisite knowledge for Complete understanding and learning of Topic:

- Functions
- Arrays
- Structures

Detailed content of the Lecture:

Pointer to structure holds the add of the entire structure.

It is used to create complex data structures such as linked lists, trees, graphs and so on.

The members of the structure can be accessed using a special operator called as an arrow operator (->).

Declaration

Following is the declaration for pointers to structures in C programming −

struct tagname *ptr;

For example − struct student *s −

Accessing

It is explained below how to access the pointers to structures.

Ptr-> membername;

For example − s->sno, s->sname, s->marks;

Example Program

The following program shows the usage of pointers to structures −

```
#include<stdio.h>
struct student{
```

```
  int sno;
  char sname[30];
  float marks;
};
main ( ){
  struct student s;
  struct student *st;
  printf("enter sno, sname, marks:");
  scanf ("%d%s%f", & s.sno, s.sname, &s. marks);
  st = &s;
  printf ("details of the student are");
  printf ("Number = %d\n", st ->sno);
  printf ("name = %s\n", st->sname);
  printf ("marks =%f\n", st ->marks);
  getch ( );
}
```

Output

Let us run the above program that will produce the following result −

enter sno, sname, marks:1 Lucky 98

details of the student are:

Number = 1

name = Lucky

marks =98.000000

Video Content / Details of website for further learning (if any):
https://www.tutorialspoint.com/what-are-pointers-to-structures-in-c-language
https://www.programiz.com/c-programming/c-structures-pointers

Important Books/Journals for further learning including the page nos.:
E Balagurusamy," Programming in ANSI C", Tata McGraw Hill, 2012
Yuksel Uckan, "Problem Solving Using C", McGraw Hill, 1999

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

Estd. 2000

IQAC

LECTURE HANDOUTS

L - 23

CSE

I/II

Course Name with Code : 16CSC02 & Advanced C Programming

Course Faculty : N.Anand

Unit : IV- Structure And Union **Date of Lecture:**

Topic of Lecture: Typedef, Enumerated Data Type

Introduction :

- For example: enum { Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday } weekday; **defines the variable weekday** , which can be assigned any of the specified enumeration constants. However, you cannot declare any additional enumeration variables using this set of enumeration constants.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Functions
- Arrays
- Structures

Detailed content of the Lecture:

There are several possibilities and conventions to name an enumeration. The first is to use a *tag name* just after the `enum` keyword.

```
enum color
{
   RED,
   GREEN,
   BLUE
};
```

This enumeration must then always be used with the keyword *and* the tag like this:

```
enum color chosenColor = RED;
```

If we use `typedef` directly when declaring the `enum`, we can omit the tag name and then use the type without the `enum` keyword:

```
typedef enum
{
   RED,
   GREEN,
   BLUE
} color;
```

```
color chosenColor = RED;
```

But in this latter case we cannot use it as `enum color`, because we didn't use the tag name in the definition. One common convention is to use both, such that the same name can be used with or without `enum` keyword. This has the particular advantage of being compatible with C++

```
enum color              /* as in the first example */
{
   RED,
   GREEN,
   BLUE
};
typedef enum color color; /* also a typedef of same identifier */

color chosenColor  = RED;
enum color defaultColor = BLUE;
```

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

https://stackoverflow.com/questions/34323130/the-importance-of-c-enumeration-typedef-enum#:~:text=A%20typedef%20is%20a%20mechanism,valid%20values%20of%20that%20type.
https://nios.ac.in/media/documents/330srsec/online_course_material_330/Theory/Lesson_17.pdf

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

E Balagurusamy," Programming in ANSI C", Tata McGraw Hill, 2012
Yuksel Uckan, "Problem Solving Using C", McGraw Hill, 1999

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

Estd. 2000

IQAC

LECTURE HANDOUTS

L - 24

CSE

I/II

Course Name with Code : 16CSC02 & Advanced C Programming

Course Faculty : N.Anand

Unit : IV- Structure And Union **Date of Lecture:**

Topic of Lecture: Union, Union of Structures

Introduction :

- A union is **a special data type available in C that allows to store different data types in the same memory location**. You can define a union with many members, but only one member can contain a value at any given time. Unions provide an efficient way of using the same memory location for multiple-purpose

Prerequisite knowledge for Complete understanding and learning of Topic:

- Functions
- Arrays
- Structures

Detailed content of the Lecture:

A union is a memory location that is shared by several variables of different data types in C programming language.

Syntax

The syntax for union of structure is as follows −

```
union uniontag{
  datatype member 1;
  datatype member 2;
  ----
  ----
  datatype member n;
};
```

Example

The following example shows the usage of union of structure −

```
union sample{
  int a;
  float b;
```

```
  char c;
};
```

Declaration of union variable

Following is the declaration for union variable. It is of three types as follows −

Type 1

```
union sample{
  int a;
  float b;
  char c;
}s;
```

Type 2

```
union{
  int a;
  float b;
  char c;
}s;
```

Type 3

```
union sample{
  int a;
  float b;
  char c;
};
union sample s;
```

- When union is declared, the compiler automatically creates largest size variable type to hold variables in the union.
- At any time, only one variable can be referred.

Initialization and accessing

- Same syntax of structure is used to access a union member.
- The dot operator is for accessing members.
- The arrow operator (->) is used for accessing the members using pointer.

Video Content / Details of website for further learning (if any): **https://www.tutorialspoint.com/what-is-union-of-structure-in-c-language#:~:text=A%20union%20is%20a%20memory,types%20in%20C%20programming%20language.** **https://www.tutorialspoint.com/cprogramming/c_unions.htm**
Important Books/Journals for further learning including the page nos.: E Balagurusamy," Programming in ANSI C", Tata McGraw Hill, 2012 Yuksel Uckan, "Problem Solving Using C", McGraw Hill, 1999

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/II

Course Name with Code : 16CSC02 & Advanced C Programming

Course Faculty : N.Anand

Unit : V-Files **Date of Lecture:**

Topic of Lecture: Introduction- File Operations

Introduction :

- The various operations which can be implemented on a file such as read, write, open and close etc. are called file operations. These operations are performed by the user by using the commands provided by the operating system.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Strings
- Files
- Character

Detailed content of the Lecture:

A **file** represents a sequence of bytes on the disk where a group of related data is stored. File is created for permanent storage of data. It is a ready made structure.

In C language, we use a structure **pointer of file type** to declare a file.

```
FILE *fp;
```

Copy

C provides a number of functions that helps to perform basic file operations. Following are the functions,

Function	Description
fopen()	create a new file or open a existing file
fclose()	closes a file
getc()	reads a character from a file
putc()	writes a character to a file
fscanf()	reads a set of data from a file
fprintf()	writes a set of data to a file

getw()	reads a integer from a file
putw()	writes a integer to a file
fseek()	set the position to desire point
ftell()	gives current position in the file
rewind()	set the position to the begining point

Opening a File or Creating a File

The fopen() function is used to create a new file or to open an existing file.

General Syntax:

```
*fp = FILE *fopen(const char *filename, const char *mode);
```

Copy

Here, *fp is the FILE pointer (FILE *fp), which will hold the reference to the opened(or created) file.

Video Content / Details of website for further learning (if any):
https://www.analyticsvidhya.com/blog/2021/10/introduction-to-file-operations-in-python/
https://www.learnconline.com/2013/12/introduction-to-file-operations-in-c-programming.html

Important Books/Journals for further learning including the page nos.:
E Balagurusamy," Programming in ANSI C", Tata McGraw Hill, 2012
Yuksel Uckan, "Problem Solving Using C", McGraw Hill, 1999

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/II

Course Name with Code : 16CSC02 & Advanced C Programming

Course Faculty : N.Anand

Unit : V-Files **Date of Lecture:**

Topic of Lecture: Opening a File, Reading a File,Closing a File

Introduction :

- Function fopen() is used both to open or create a file, and fclose() is used to close an already opened file. File 'read' and 'write' operations can be performed after opening a file. The fputc() and fputs() functions are used to write characters and strings respectively in a file.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Strings
- Files
- Character

Detailed content of the Lecture:

C File management

A File can be used to store a large volume of persistent data. Like many other languages 'C' provides following file management functions,

1. Creation of a file
2. Opening a file
3. Reading a file
4. Writing to a file
5. Closing a file

Following are the most important file management functions available in 'C,'

function	purpose
fopen ()	Creating a file or opening an existing file
fclose ()	Closing a file
fprintf ()	Writing a block of data to a file
fscanf ()	Reading a block data from a file
getc ()	Reads a single character from a file
putc ()	Writes a single character to a file
getw ()	Reads an integer from a file
putw ()	Writing an integer to a file
fseek ()	Sets the position of a file pointer to a specified location
ftell ()	Returns the current position of a file pointer
rewind ()	Sets the file pointer at the beginning of a file

To create a file in a 'C' program following syntax is used,

```
FILE *fp;
fp = fopen ("file_name", "mode");
```

File Mode	Description
r	Open a file for reading. If a file is in reading mode, then no data is deleted if a file is already present on a system.
w	Open a file for writing. If a file is in writing mode, then a new file is created if a file doesn't exist at all. If a file is already present on a system, then all the data inside the file is truncated, and it is opened for writing purposes.
a	Open a file in append mode. If a file is in append mode, then the file is opened. The content within the file doesn't change.
r+	open for reading and writing from beginning
w+	open for reading and writing, overwriting a file
a+	open for reading and writing, appending to file

Video Content / Details of website for further learning (if any):
https://www.programiz.com/c-programming/c-file-input-output
https://www.guru99.com/c-file-input-output.html

Important Books/Journals for further learning including the page nos.:
E Balagurusamy," Programming in ANSI C", Tata McGraw Hill, 2012
Yuksel Uckan, "Problem Solving Using C", McGraw Hill, 1999

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/II

Course Name with Code : 16CSC02 & Advanced C Programming

Course Faculty : N.Anand

Unit : V-Files **Date of Lecture:**

Topic of Lecture: Text Modes- Binary Modes

Introduction :

- The two file types may look the same on the surface, but they encode data differently. While both binary and text files contain data stored as a series of bits (binary values of 1s and 0s), the bits in text files represent characters, while the bits in binary files represent custom data.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Strings
- Files
- Character

Detailed content of the Lecture:

Text mode, usually by default, and binary mode. Obviously, in text mode, the program writes data to file as text characters, and in binary mode, the program writes data to files as 0/1 bits. While it sounds trivial to distinguish the two modes, people sometimes got confused.

Text mode	Binary mode
In text mode various character translations are performed i.e; "\r+\f" is converted into "\n"	In binary mode, such translations are not performed.

To write in the files:
ofstream ofs ("file.txt");
Or
ofstream ofs;
ofs.open("file.txt");
	to write in the files:
	ofstream ofs("file.txt",ios::binary);
	or
	ofstream ofs;
	ofs.open("file.txt", ios::binary);

To add text at the end of the file: Ofstream ofs("file.txt",ios::app); or ofstream ofs;	To add text at the end of the file: Ofstream ofs("file.txt",ios::app|ios::binary); or ofstream ofs;

ofs.open(“file.txt”, ios::app);	ofs.open(“file.txt”, ios::app|ios::binary);
To read files: ifstream in (“file.txt”); or ifstream in ; in.open(“file.txt”);	To read files: ifstream in (“file.txt”, ios::binary); or ifstream in ; in.open(“file.txt”, ios::binary);

Example

We have a `signed int` `-10000`, an `unsigned shot` `100`, and a C string `WE`. Their binary sequence representations in an 64-bit computer is as follows.

`signed int` `-10000`:

```
11111111 11111111 11011000 11110000
```

`std::string` `-10000`:

```
00101101 00110001 00110000 00110000 00110000 00110000
```

`unsigned shot` `100`:

```
00000000 01100100
```

`std::string` `100`:

```
00110001 00110000 00110000
```

C string `WE`:

```
01010111 01000101 00000000
```

Video Content / Details of website for further learning (if any): **https://www.tutorialspoint.com/difference-between-files-written-in-binary-and-text-mode-in-cplusplus** **https://www.codingeek.com/tutorials/c-programming/text-files-vs-binary-files-in-c-programming-language/**
Important Books/Journals for further learning including the page nos.: E Balagurusamy," Programming in ANSI C", Tata McGraw Hill, 2012 Yuksel Uckan, "Problem Solving Using C", McGraw Hill, 1999

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/II

Course Name with Code : 16CSC02 & Advanced C Programming

Course Faculty : N.Anand

Unit : V-Files **Date of Lecture:**

Topic of Lecture: File Functions, fprintf(), fscanf(),getc()

Introduction :

- fprintf **() function writes formatted data to a file**. fscanf () fscanf () function reads formatted data from a file. fputchar () fputchar () function writes a character onto the output screen from keyboard input.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Strings
- Files
- Character

Detailed content of the Lecture:

Input means to provide the program with some data to be used in the program and Output means to display data on the screen or write the data to a printer or a file.

The C programming language provides standard library functions to read any given input and to display data on the console.

Before moving forward with input and output in C language, check these topics to understand the concept better :

- C Syntax Rules
- Compile and Run C Program
- Data Types in C
- Variables in C

The functions used for standard input and output are present in the stdio.h header file. Hence to use the functions we need to include the stdio.h header file in our program, as shown below.

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

Copy

Following are the functions used for standard input and output:

1. printf() function - Show Output
2. scanf() function - Take Input
3. getchar() and putchar() function

4. gets() and puts() function

In C Language, computer monitor, printer, etc. output devices are treated as files and the same process is followed to write output to these devices as would have been followed to write the output to a file.

printf() function - Show Output

The printf() function is the most used function in the C language. This function is defined in the stdio.h header file and is used to show output on the console (standard output).

This function is used to print a simple text sentence or value of any variable which can be of int, char, float, or any other datatype.

printf() Example - Print a statement

Let's print a simple sentence using the printf() function.

```
#include <stdio.h>



int main() {

   // using printf()

   printf("Welcome to Studytonight");



   return 0;

}
```

Copy

```
Welcome to Studytonight
```

Video Content / Details of website for further learning (if any): **https://fresh2refresh.com/c-programming/c-file-handling/fscanf-fprintf-ftell-rewind-functions-c/** **https://www.studytonight.com/c/c-input-output-function.php**
Important Books/Journals for further learning including the page nos.: E Balagurusamy," Programming in ANSI C", Tata McGraw Hill, 2012 Yuksel Uckan, "Problem Solving Using C", McGraw Hill, 1999

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 - 29

CSE

I/II

Course Name with Code : 16CSC02 & Advanced C Programming

Course Faculty : N.Anand

Unit : V-Files **Date of Lecture:**

Topic of Lecture: putc(), fgetc(),fputc(),fseek(), feof()

Introduction :

- fgetc() and fputc() in C. fgetc() is **used to obtain input from a file single character at a time**. This function returns the ASCII code of the character read by the function. It returns the character present at position indicated by file pointer.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Strings
- Files
- Character

Detailed content of the Lecture:

File operation	Declaration & Description
getc()	Declaration: int getc(FILE *fp) getc functions is used to read a character from a file. In a C program, we read a character as below. **getc** (fp);

putc()	Declaration: int putc(int char, FILE *fp)
	putc function is used to display a character on standard output or is used to write into a file. In a C program, we can use putc as below.
	
	putc(char, stdout);
	putc(char, fp);

EXAMPLE PROGRAM FOR GETC(), PUTC() FUNCTIONS IN C PROGRAMMING LANGUAGE:
This file handling C program illustrates how to read the contents of a file. Assume that, a file called "test.c" contains the following data "Hi, How are you?". Let's read this data using following C program using getc() and putc() functions.

Please note that putc() function can be used to write a character into a file also instead of stdout. For

that, please use putc(char, fp).

```
C1 #include <stdio.h>
 2  int main()
 3  {
 4     char ch;
 5     FILE *fp;
 6     if (fp = fopen("test.c", "r"))
 7     {
 8       ch = getc(fp);
 9       while (ch != EOF)
10       {
11          putc(ch, stdout);
12          ch = getc(fp);
13       }
14       fclose(fp);
15       return 0;
16     }
17     return 1;
18  }
```

OUTPUT:

```
Hi, How are you?
```

Video Content / Details of website for further learning (if any):
https://www.geeksforgeeks.org/fgetc-fputc-c/
https://fresh2refresh.com/c-programming/c-file-handling/fscanf-fprintf-ftell-rewind-functions-c/

Important Books/Journals for further learning including the page nos.:
E Balagurusamy," Programming in ANSI C", Tata McGraw Hill, 2012
Yuksel Uckan, "Problem Solving Using C", McGraw Hill, 1999

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/II

Course Name with Code : 16CSC02 & Advanced C Programming

Course Faculty : N.Anand

Unit : V-Files **Date of Lecture:**

Topic of Lecture: Command Line Arguments

Introduction :

Properties of Command-Line arguments in C:-

- argv[0] prints the name of the program.
- argv[1] prints the first argument which is provided by the user.
- argv[argc] is a null pointer.
- Always passed to the main() function.
- Command Line Arguments are passed by the user from the terminal.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Strings
- Files
- Character

Detailed content of the Lecture:

The command line to your C programs when they are executed. These values are called **command line arguments** and many times they are important for your program especially when you want to control your program from outside instead of hard coding those values inside the code.

The command line arguments are handled using main() function arguments where **argc** refers to the number of arguments passed, and **argv[]** is a pointer array which points to each argument passed to the program. Following is a simple example which checks if there is any argument supplied from the command line and take action accordingly −

```
#include <stdio.h>

int main( int argc, char *argv[] )  {

  if( argc == 2 ) {
    printf("The argument supplied is %s\n", argv[1]);
  }
  else if( argc > 2 ) {
    printf("Too many arguments supplied.\n");
  }
  else {
    printf("One argument expected.\n");
```

```
   }
}
```

When the above code is compiled and executed with single argument, it produces the following result.

```
$./a.out testing
The argument supplied is testing
```

When the above code is compiled and executed with a two arguments, it produces the following result.

```
$./a.out testing1 testing2
Too many arguments supplied.
```

When the above code is compiled and executed without passing any argument, it produces the following result.

```
$./a.out
One argument expected
```

It should be noted that **argv[0]** holds the name of the program itself and **argv[1]** is a pointer to the first command line argument supplied, and *argv[n] is the last argument. If no arguments are supplied, argc will be one, and if you pass one argument then **argc** is set at 2.

You pass all the command line arguments separated by a space, but if argument itself has a space then you can pass such arguments by putting them inside double quotes "" or single quotes ". Let us re-write above example once again where we will print program name and we also pass a command line argument by putting inside double quotes −

```
#include <stdio.h>

int main( int argc, char *argv[] )  {

   printf("Program name %s\n", argv[0]);

   if( argc == 2 ) {
      printf("The argument supplied is %s\n", argv[1]);
   }
   else if( argc > 2 ) {
      printf("Too many arguments supplied.\n");
   }
   else {
      printf("One argument expected.\n");
   }
}
```

Video Content / Details of website for further learning (if any): https://www.geeksforgeeks.org/command-line-arguments-in-c-cpp/ https://www.tutorialspoint.com/cprogramming/c_command_line_arguments.htm
Important Books/Journals for further learning including the page nos.: E Balagurusamy," Programming in ANSI C", Tata McGraw Hill, 2012 Yuksel Uckan, "Problem Solving Using C", McGraw Hill, 1999

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.

Department of Computer Science and Engineering
Question Bank - Academic Year (2020-21)

Course Code & Course Name : 16CSC02 & Advanced C Programming

Name of the Faculty : N.Anand

Year/Sem/Sec : I/II/A

Unit-I : Arrays
Part-A (2 Marks)

1. What is an array? Write the syntax for array.
2. What is an array? Write the syntax for array.
3. Identify the way to assign an array to another array.
4. List out the advantages of Arrays.
5. What will happen when you access the array more than its dimension?
6. Point out an example code to express two dimensional array.
7. What are the different ways of initializing array?
8. Distinguish between one dimensional and two dimensional arrays.
9. List out the any four functions that are performed on character strings.
10. Specify any two applications of Array.

Part-B (16 Marks)

1. (i)Write a C Program to take 5 values from the user and store them in an array (16)
   (ii)Write a C program to re-order a one-dimensional array of numbers in descending order.
2. Write a C program for transpose of a matrix. (16)
3. Write a C program to calculate median for an array of elements. (16)
4. Explain with an example how to copy all elements of an array into another array (16)
5. Explain about the String Arrays and its manipulation in detail (16)

Unit-II : Pointers & Preprocessor Directives

Part-A (2 Marks)

1. Define pointer. How can you declare it?
2. What is pointer to pointer?
3. What is pointer arithmetic?
4. Define pointer array.
5. How can you read a string through keyboard?
6. What is array of strings?
7. Display string ―pepper‖ in reverse order
8. Discriminate puts() and gets()
9. Discriminate putchar() and getchar()
10. How can you compare two strings?

Part-B (16 Marks)

1. Define pointer. How to declare and initialize it. (16)
2. Write a C program to illustrate the use of indirection operator to access the value pointed by a pointer. (16)
3. (a)What are the features of pointers? Write a C program to print address of a variable (16)
   (b) Explain the declaration of pointers and pointer to pointer with examples.
4. (a) Explain the concept of functions returning pointers with example. (16)
   (b) Write a C program to read and print an array of elements using pointers.
5. (a)Explain the concept of array of pointers with examples. (16)
   (b) Write a C program to read and display multiple strings using pointers.

Unit-III : Functions

Part-A (2 Marks)

1. What is a function? Write the types of functions.
2. Express the difference between function declaration and function definition.
3. What is function call?
4. Write and explain the syntax of function?
5. What is #include, #define directives.
6. Write any two applications of recursive function.
7. Why is scope of variable necessary in function?
8. Differentiate between call by value and call reference.
9. What is meant by library function?
10. Develop no argument and no return value in a function.

Part-B (16 Marks)

1. Describe about user defined function and predefined function with an example. (16)
2. Write a code in C to get the smallest element of an array using function. Analyze the code with sample input 34,2,6,11 and 46. (16)
3. Apply a recursive function in C for reverse a sentence. (16)
4. Discuss about the classification of functions depending upon their inputs and output (parameters). (8)
   (8)
5. Discuss about passing arrays to function. (16)

Unit-IV : Structure And Union

Part-A (2 Marks)

1. What is structure?
2. Where is Union used in C?
3. How the members of structure object is accessed?
4. How many bytes in memory taken by the following C structure?
```
#include <stdio.h>
struct test
{ int k;
char c;
};
```
5. What is a nested structure?
6. How typedef is used in structure?
7. Interpret the term Union in C.
8. What is the output of this program?
```
#include<stdio.h>
void main()
{
enum days {MON=-1, TUE, WED=4, THU,FRI, SAT};
printf("%d, %d, %d, %d, %d, %d", MON, TUE, WED,
THU, FRI, SAT);
}
```
9. Point out the meaning of array of structures.
10. Specify the use of typedef.

Part-B (16 Marks)

1. Describe about the functions and structures. (16)
2. Explain about the structures and its operations. (16)
3. Examine the differences between nested structures and array of structures. (16)
4. Write a C program using structures to prepare the students mark statement. (16)
5. i)Express a structure with data members of various types and declare two structure variables. Write a program to read data into these and print the same. (16)
   (ii)Justify the need for structured data type.

Unit-V : Files

Part-A (2 Marks)

1. What are the Different file operations?
2. Write about Sequential file handling functions?
3. Write about Random file handling functions?
4. Write about different file modes?
5. Write about different error handling functions on files?
6. Define i. fgets() ii. fputs()
7. File handling functions:
   a. fseek() b. ftell() c. rewind() d. feof()
8. Write the syntax for opening a file .
9. Write the steps for reads the name of a file
10. Write the steps for display the content of the file

Part-B (16 Marks)

1. Write a C program to count no.of characters, spaces, lines, words of a file. (16)
2. Write a C program to copy the contents from one file to another file. (16)
3. Write the syntax for opening a file with various modes and closing a file. (16)
   (b) Explain the following file handling functions:
   a. fseek() b. ftell() c. rewind() d. feof()
4. (a)Discuss command line arguments in detail with examples. (16)
   (b) Write a short notes on
   i. fgets() ii. fputs()
5. Write a program in C that reads the name of a file and displays the contents of the file on the user screen. (16)

Course Faculty **HoD**

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

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

Estd. 2000

OFFICE OF THE CONTROLLER OF EXAMINATIONS

CIA-I

Degree & Branch	**:**	**B.E & CSE**	**Max Marks**	**:**	**50**
Year/Sem	**:**	**I/II**	**Duration**	**:**	**1.30 Hrs**
Course Code & Course Name	**:**	**16CSC02 & Advanced C Programming**	**Date**	**:**	

16CSC02.CO1 : Student will able to develop programs using single dimensional and multi dimensional arrays

16CSC02.CO2 : Students will be able to Perform memory access operations using pointers

PART-A (5x2=10Marks)

Answer all the questions

Q.No.	Questions	BT Level	Course Outcome
1.	Define an array?	K1	CO1
2.	Write any four features of arrays.	K2	CO1
3.	List out the types of arrays in C?	K1	CO2
4.	Write about pointer?	K4	CO2
5.	Distinguish between Library function and user defined function?	K5	CO2

PART – B (1x10=10 Marks)

Compulsory Question

Q.No.	Questions	BT Level	Course Outcome
6.	Implement an array to identify max and min element using C programming.	K2	CO1

PART – C (2x15=30 Marks)

Answer all the questions

Q.No.		Questions	BT Level	Course Outcome
7.	a)	Explain all steps in Multiplication of two matrix with an example program.	K1	CO1
		OR		
	b)	Implement student database using array.	K3	CO1
8.	a)	Explain #include and #define preprocessor directives.	K2	CO2
		OR		
	b)	Discuss in detailed working principle of void pointers, null pointers, and array pointers with an example program.	K4	CO2

Bloom's Taxonomy Level

Bloom's Taxonomy Level	K1 Remembering	K2 Understanding	K3 Applying	K4 Analyzing	K5 Evaluating	K6 Creating	Total
% of Questions	24	34	19	21	3	0	100%

Course Faculty **HoD**

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

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

ASSIGNMENT – 1 ANSWER KEY

Degree & Branch	**:**	**B.E & CSE**	**Max Marks**	**:**	**50**
Year / Sem	**:**	**I/II**	**Duration**	**:**	**1.30 Hrs**
Course Code & Course Name	**:**	**16CSC02 & Advanced C Programming**	**Date**	**:**	

16CSC02.CO1 : Student will able to develop programs using single dimensional and multi dimensional arrays

16CSC02.CO2 : Students will be able to Perform memory access operations using pointers

PART-A (5x2=10 Marks)

Answer all the questions

Q.No. **Questions**

1. Define an array?

Array is the collection of similar data items which stores data in adjacent memory locations.

Example:

```
int a[5]={1,2,3,4,5};
```

2. Write any four features of arrays.

- An array is a derived data type
- It is used to represent a collection of elements of the same data type.
- The elements can be accessed with base address.
- The starting memory location(Base address) are represented by the array name.

3. List out the types of arrays in C?

There are three types of arrays in C. They are

- One dimensional Array
- Two Dimensional Arrays and
- Three Dimensional Arrays

4. Define is Function?

A function is a self contained block or a sub program of one or more statements that performs a special task when called.

Types:

- Pre defined functions.
    - User Defined functions

5. Distinguish between Library function and user defined function?

Library Functions	User-defined Functions
a) Library functions are pre-defined set of functions that are defined in C libraries. b) User can only use the function but cannot change (or) modify this function.	a) The User-defined functions are the functions defined by the user according to his/her requirement. b) User can use this type of function. User can also modify this function.

PART – B (1x10=10 Marks)

Compulsory Question

Q.No. **Questions**

6. Implement an array to identify max and min element using C programming.

1.Create two intermediate variables max and min to store the maximum and minimum element of the array.

2. Assume the first array element as maximum and minimum both, say max = arr[0] and min = arr[0].

3. Traverse the given array arr[].

4. If the current element is smaller than min, then update the min as the current element.

5. If the current element is greater than the max, then update the max as the current element.

6. Repeat the above two steps 4 and 5 for the element in the array.

Input: arr[] = {1, 2, 4, -1}
Output:
The minimum element is -1
The maximum element is 4
Input: arr[] = {-1, -1, -1, -1}
Output:
The minimum element is -1
The maximum element is -1

PART – C (2x15=30 Marks)

Answer all the questions

Q.No. **Questions**

7. (a)Explain all steps in Multiplication of two matrix with an example program.

(Square Matrix Multiplication)

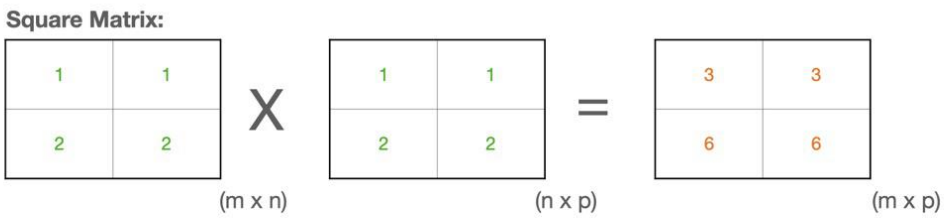
Input : mat1[m][n] = {
{1, 1},
{2, 2}
}
mat2[n][p] = {
{1, 1},
{2, 2}
}

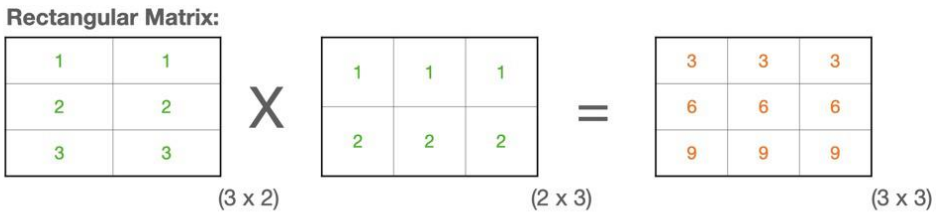
Output : result[m][p] = {
{3, 3},
{6, 6}
}

(Rectangular Matrix Multiplication)

Input : mat1[3][2] = {
{1, 1},
{2, 2},
{3, 3}
}
mat2[2][3] = {
{1, 1, 1},
{2, 2, 2}
}

Output : result[3][3] = {
{3, 3, 3},
{6, 6, 6},
{9, 9, 9}
}





(b)Implement student database using array.

Arrays possess many properties. In order to work with them, we need to remember some key

points:

1. An array can only have elements of the same data type.
2. The system stores the elements of an array in subsequent memory locations.
3. An array index starts with 0 and goes up to n-1 where n is the number of elements.
4. In C programming, it supports multidimensional arrays. For example, a two-dimensional array.
5. In the case of two-dimensional arrays, the system stores the elements row by row in subsequent memory locations.
6. The array name refers to the address of the first element.
7. Array size is always constant and not a variable.
8. If we have to pass an array to a function, we can refer to it by defining a pointer to the specific array
9. We generate the pointer to the first element of an array by the array name without index.
10. In C we have a function that can return an array.

8. (a)Explain #include and #define preprocessor directives.

The C preprocessor versus the compiler
Please remember that, C preprocessor is not a part of C compiler. It has different syntax from normal C statements compiled by the compiler, for example:

It start with hash/pound # character.

C preprocessor is line oriented. Each statement start in a separate line. While it is not mandatory for other C statements to start in separate line.

These statements end with new line. While, other statements are terminated by semicolon.

C preprocessor is a **Micro preprocessor** which compiles the code before the compilation.

List of all preprocessor directives:

#include preprocessor directive
#define and #undef preprocessor directive
Parameterized Macros (Function like Macros)
#ifdef, #ifndef and #endif preprocessor directive
#if...#elif...#else...#endif
Stringize operator (#)
Token pasting (##)
#pragma preprocessor directive
#error preprocessor directive
#null preprocessor directive

(b)Discuss in detailed working principle of void pointers, null pointers, and array pointers with an example program.

The **Pointer** in C, is a variable that stores address of another variable. A pointer can also be used to refer to another pointer function. A pointer can be incremented/decremented, i.e., to point to the next/ previous memory location. The purpose of pointer is to save memory space and achieve

faster execution time.

If you print the address of a variable on the screen, it will look like a totally random number (moreover, it can be different from run to run).

Let's try this in practice with pointer in C example

```c
#include <stdio.h>

int main() {
    int v = 0;
    printf("%d\n", &v);
    return 0;
}
```

The output of this program is -480613588.

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

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

DESIGNING YOUR FUTURE
Estd. 2000

OFFICE OF THE CONTROLLER OF EXAMINATIONS
CIA-II

Degree & Branch	**:**	**B.E & CSE**	**Max Marks**	**:**	**50**
Year/Sem	**:**	**I/II**	**Duration**	**:**	**1.30 Hrs**
Course Code & Course Name	**:**	**16CSC02 & Advanced C Programming**	**Date**	**:**	

16CSC02.CO3 : Student will able to develop programs using single dimensional and multi dimensional arrays

16CSC02.CO4 : Students will be able to Perform memory access operations using pointers

PART-A (5x2=10Marks)
Answer all the questions

Q.No.	Questions	BT Level	Course Outcome
1.	Define is Function?	K1	CO3
2.	List out the steps in writing a function in a program.	K2	CO3
3.	Give the syntax for using user-defined functions in a program.	K3	CO3
4.	List the similarities and differences between structure and union	K2	CO4
5.	Define Structures.	K1	CO4

PART – B (1x10=10 Marks)
Compulsory Question

Q.No.	Questions	BT Level	Course Outcome
6.	7. Briefly explain about function prototypes	K2	CO3

PART – C (2x15=30 Marks)
Answer all the questions

Q.No.		Questions	BT Level	Course Outcome
8.	a)	Explain cal by value and call by reference with c program	K1	CO3
		OR		
	b)	Briefly explain about String function with examples	K3	CO3
9.	a)	Write a c program for student mark sheet using structure	K4	CO4
		OR		
	b)	Explain about Enumerated Data Type	K1	CO4

Bloom's Taxonomy Level

Bloom's Taxonomy Level	K1 Remembering	K2 Understanding	K3 Applying	K4 Analyzing	K5 Evaluating	K6 Creating	Total
% of Questions	24	34	19	21	3	0	100%

Course Faculty **HoD**

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

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

OFFICE OF THE CONTROLLER OF EXAMINATIONS

CIA-II-(Answer Key)

Degree & Branch	:	**B.E & CSE**	**Max Marks**	:	**50**
Year/Sem	:	**I/II**	**Duration**	:	**1.30 Hrs**
Course Code & Course Name	:	**16CSC02 & Advanced C Programming**	**Date**	:	

16CSC02.CO3 : Student will able to develop programs using single dimensional and multi dimensional arrays

16CSC02.CO4 : Students will be able to Perform memory access operations using pointers

PART-A (5x2=10Marks)

Answer all the questions

Q.No.	Questions	BT Level	Course Outcome
1.	Define is Function?	K1	CO3

A function is a self contained block or a sub program of one or more statements that performs a special task when called.

Types:

- Pre defined functions.
- User Defined functions

2.	List out the steps in writing a function in a program.	K2	CO3

a) Function Declaration (Prototype declaration):

Every user-defined functions has to be declared before the main().

b) Function Callings:

The user-defined functions can be called inside any functions like main(), user-defined function, etc.

c) Function Definition:

The function definition block is used to define the user-defined functions with statements.

3.	Give the syntax for using user-defined functions in a program.	K3	CO3

Syntax for using user-defined functions in a program

Syntax:

```
function declaration;          function definition;
main()                          main()
 {                              {
```

function calling; } function calling;}

function definition;

4. List the similarities and differences between structure and union K2 CO4

S.No	Structure	Union
1.	The size of the structure is sum of the size of each member in the struchture.	size of the union is size of Largest member in the union because union members are overlaps on each other in memory.
2.	Structure elements are of same size.	Union element can be of different sizes
3.	Keyword struct is used.	Keyword union is used.
4.	All members in structure may be initialized	First member in union may be initialized

5. Define Structures. K1 CO4

A structure is a collection of one or more variables of different data types grouped together under a single name. It contains different data types.
Syntax:

struct struct-name

{

type variable 1; type variable 2; type variable n;

} structure_variables;

PART – B (1x10=10 Marks)

Compulsory Question

Q.No.	Questions	BT Level	Course Outcome
6.	Briefly explain about function prototypes		

The Function prototype serves the following purposes –
1) It tells the return type of the data that the function will return.
2) It tells the number of arguments passed to the function.
3) It tells the data types of each of the passed arguments.
4) Also it tells the order in which the arguments are passed to the function.
Therefore essentially, the function prototype specifies the input/output interlace to the function i.e. what to give to the function and what to expect from the function.
The prototype of a function is also called the signature of the function. K2 CO3
What if one doesn't specify the function prototype?
The output of the below kinds of programs is generally asked at many places.

Take a step-up from those "Hello World" programs. Learn to implement data structures like Heap, Stacks, Linked List and many more! Check out our **Data Structures in C** course to start learning today.

c

```
int main()
{
  foo();
  getchar();
  return 0;
}
void foo()
{
  printf("foo called");
}
```

If one doesn't specify the function prototype, the behavior is specific to the C standard (either C90 or C99) that the compilers implement. Up to the C90 standard, C compilers assumed the return type of the omitted function prototype as int. And this assumption at the compiler side may lead to unspecified program behavior.
Later C99 standard specified that compilers can no longer assume return type as int. Therefore, C99 became more restricted in type checking of function prototypes. But to make C99 standard backward compatible, in practice, compilers throw the warning saying that the return type is assumed as int. But they go ahead with compilation. Thus, it becomes the responsibility of programmers to make sure that the assumed function prototype and the actual function type matches.
To avoid all these implementation specifics of C standards, it is best to have a function prototype.

PART – C (2x15=30 Marks)
Answer all the questions

Q.No.		Questions	BT Level	Course Outcome
7.	a)	Explain cal by value and call by reference with c program **Call By Value**: In this parameter passing method, values of actual parameters are copied to function's formal parameters and the two types of parameters are stored in different memory locations. So any changes made inside functions are not reflected in actual parameters of	K1	CO3

the caller.

Call by Reference: Both the actual and formal parameters refer to the same locations, so any changes made inside the function are actually reflected in actual parameters of the caller.

Call By Value	Call By Reference
While calling a function, we pass values of variables to it. Such functions are known as “Call By Values”.	While calling a function, instead of passing the values of variables, we pass address of variables(location of variables) to the function known as “Call By References.
In this method, the value of each variable in calling function is copied into corresponding dummy variables of the called function.	In this method, the address of actual variables in the calling function are copied into the dummy variables of the called function.
With this method, the changes made to the dummy variables in the called function have no effect on the values of actual variables in the calling function.	With this method, using addresses we would have an access to the actual variables and hence we would be able to manipulate them.

```
// C program to illustrate
// call by value

#include

// Function Prototype
void swapx(int x, int y);

// Main function
int main()
{
    int a = 10, b = 20;

    // Pass by Values
    swapx(a, b);

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

    return 0;
```

```
// C program to illustrate
// Call by Reference

#include

// Function Prototype
void swapx(int*, int*);

// Main function
int main()
{
    int a = 10, b = 20;

    // Pass reference
    swapx(&a, &b);

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

    return 0;
```

```
}
```

```
// Swap functions that swaps
// two values
void swapx(int x, int y)
{
   int t;

   t = x;
   x = y;
   y = t;

   printf("x=%d y=%d\n", x, y);
}
```

Output:
x=20 y=10
a=10 b=20

Thus actual values of a and b remain unchanged even after exchanging the values of x and y.

In call by values we cannot alter the values of actual variables through function calls.

Values of variables are passes by Simple technique.

```
}
```

```
// Function to swap two variables
// by references
void swapx(int* x, int* y)
{
   int t;

   t = *x;
   *x = *y;
   *y = t;

   printf("x=%d y=%d\n", *x, *y);
}
```

Output:
x=20 y=10
a=20 b=10

Thus actual values of a and b get changed after exchanging values of x and y.

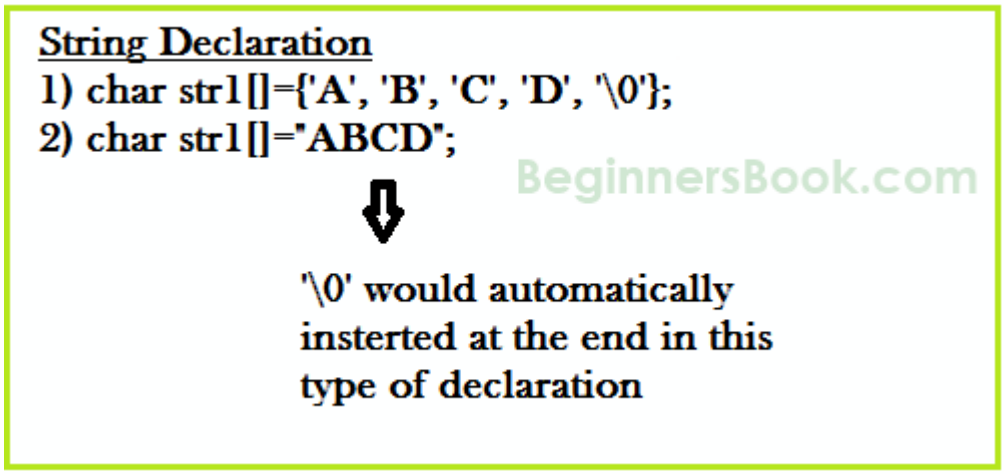
In call by reference we can alter the values of variables through function calls.

Pointer variables are necessary to define to store the address values of variables.

OR

b) Briefly explain about String function with examples CO3

String is an array of characters. In this guide, we learn how to declare strings, how to work with strings in C programming and how to use the pre-defined string handling functions. K3
We will see how to compare two strings, concatenate strings, copy one string to another & perform various string manipulation operations. We can perform such operations using the pre-defined functions of “string.h” header file. In order to use these string functions you must include string.h file in your C program.



Method 1:

```
char address[]={'T', 'E', 'X', 'A', 'S', '\0'};
```

Method 2: The above string can also be defined as –

```
char address[]="TEXAS";
```

In the above declaration NULL character (\0) will automatically be inserted at the end of the string.

```
#include <stdio.h>
#include <string.h>
int main()
{
   /* String Declaration*/
   char nickname[20];

   printf("Enter your Nick name:");

   /* I am reading the input string and storing it in nickname
    * Array name alone works as a base address of array so
    * we can use nickname instead of &nickname here
    */
   scanf("%s", nickname);

   /*Displaying String*/
   printf("%s",nickname);

   return 0;
}
```

Output:

```
Enter your Nick name:Negan
Negan
```

8. a) Write a c program for student mark sheet using structure K4 CO4

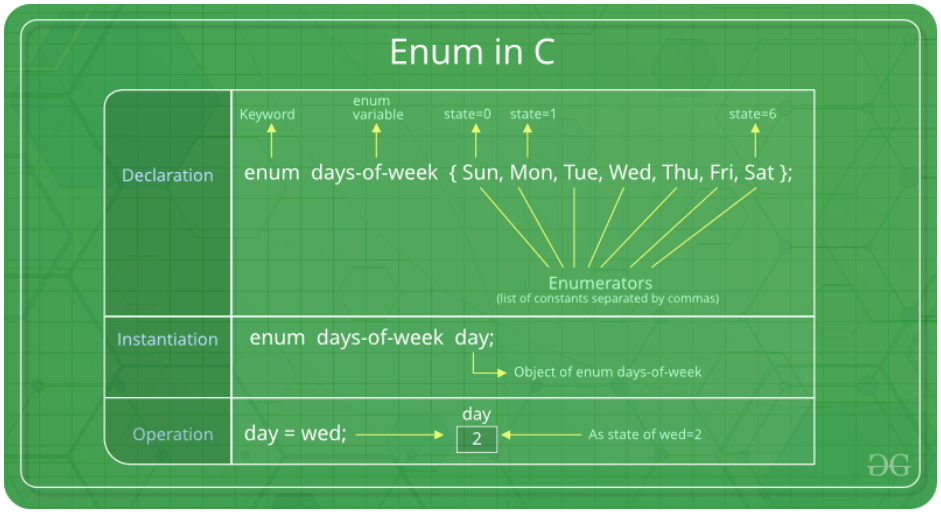
```
#include<stdio.h>
#include<conio.h>

struct stu
{
     int rn,grade,a[5];
     float avg;
  }s[2];
```

```
void main()
{
      int i,j,sum,n;
      float avg;
      clrscr();
      printf("\t STUDENT MARKSHEET USING STRUCTURES\n\n");
      printf("Enter the no of students");
      scanf("%d",&n);

for(i=0;i<n;i++)
{
      scanf("%d",s[i].rn);

for(j=0;j<=5;j++)
{
      scanf("%d",&s[i].a[j]);
   }
 }

for(i=0;i<n;i++)
{
       sum=0;

for(j=0;j<5;j++)
{
        sum=sum+s[i].a[j];
        s[i].avg=(float)(sum/5);
        if(s[i].avg>=60)
        s[i].grade=1;
        else if(s[i].avg>50&&s[i].avg<60)
        s[i].grade=2;
else
       s[i].grade=3;
  }
}

       printf("****************************************\n");
       printf("rn\ts1\ts2\ts3\ts4\ts5\tavg\t grade\n");
       printf("****************************************\n");

for(i=0;i<n;i++)
{
       printf("%d\t",s[i].rn);

for(j=0;j<5;j++)
{
       printf("%d\t",s[i].a[j]);
    }
      printf("%f\t%d\n",s[i].avg,s[i].grade);
  }
  getch();
}
```

OR

b) Explain about Enumerated Data Type

Enumeration (or enum) is a user defined data type in C. It is mainly used to assign names to integral constants, the names make a program easy to read and maintain.



Take a step-up from those "Hello World" programs. Learn to implement data structures like Heap, Stacks, Linked List and many more! Check out our **Data Structures in C** course to start learning today.

enum State {Working = 1, Failed = 0};

The keyword ‘enum’ is used to declare new enumeration types in C and C++. Following is an example of enum declaration.

// The name of enumeration is "flag" and the constant K1 CO4

// are the values of the flag. By default, the values

// of the constants are as follows:

// constant1 = 0, constant2 = 1, constant3 = 2 and

// so on.

enum flag{constant1, constant2, constant3, ... };

Variables of type enum can also be defined. They can be defined in two ways:

// 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;
```

}

Bloom's Taxonomy Level

Bloom's Taxonomy Level	K1 Remembering	K2 Understanding	K3 Applying	K4 Analyzing	K5 Evaluating	K6 Creating	Total
% of Questions	24	34	19	21	3	0	100%

Course Faculty **HoD**

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

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

OFFICE OF THE CONTROLLER OF EXAMINATIONS

CIA-III

Degree & Branch	**:**	**B.E & CSE**	**Max Marks**	**:**	**100**
Year/Sem	**:**	**I/II**	**Duration**	**:**	**3.00 Hrs**
Course Code & Course Name	**:**	**16CSC02 & Advanced C Programming**	**Date**	**:**	

16CSC02.CO1	:	Students will able to develop programs using single dimensional and multi dimensional arrays
16CSC02.CO2	:	Students ability to perform memory access operations using pointers
16CSC02.CO3	:	Students able to solve real time applications using Functions
16CSC02.CO4	:	Students have utilize memory efficiently using structures and union
16CSC02.CO5	:	Students able to design programs to perform operations on files

PART-A (10x2=20Marks)

Answer all the questions

Q. No.	Questions	BT Level	Course Outcome
1.	Write the value of b[0] in the following program?	K1	CO1
2.	Give an example for compile-time array initialization.	K1	CO1

3.	Determine the output for the following printf statements?
	main()
	{
	char *ptr=”string”;
	printf(“%c”,*ptr++);
	printf(“%c”,*(++ptr));
	printf(“%c”,(*ptr)++);
	printf(“%c”,++*ptr);
	
	}
		K4	CO2

4.	Write the output of the program?	K3	CO2
5.	Define recursion?	K1	CO3
6.	Write the purpose of the function main()?	K4	CO3
7.	Define Union.	K1	CO4
8.	List out the pre-processor directives?	K5	CO4
9.	State file opening?	K1	CO5
10.	Define file pointer?	K6	CO5

PART – B (5x16=80 Marks)

Answer all the questions

Q.No.		Questions	BT Level	Course Outcome
11	(a)	Implement student database using array.	K2	CO1
		(OR)		
	(b)	Explain all steps to Subtraction of two matrix with an example program.	K4	CO1
12	(a)	Predefined macros in ANSI C	K3	CO2
		(OR)		
	(b)	Write the Pointers and Double Pointers concept.	K2	CO2
13	(a)	Explain about Recursive Function with suitable C program.	K3	CO3
		(OR)		
	(b)	Briefly explain about function prototypes	K4	CO3
14	(a)	Write a c program for library management system using union	K6	CO4
		(OR)		
	(b)	Briefly explain about union with example	K6	CO4
15	(a)	What is the purpose of rewind() ?	K2	CO5
		(OR)		
	(b)	Write a c program for reading a file and closing a file.	K5	CO5

Bloom's Taxonomy Level

Bloom's Taxonomy Level	K1 Remembering	K2 Understanding	K3 Applying	K4 Analyzing	K5 Evaluating	K6 Creating	Total
% of Questions	9	28	18	19	9	18	100%

Course Faculty **HoD**

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

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

OFFICE OF THE CONTROLLER OF EXAMINATIONS
CIA-III-ANSWER KEY

Degree & Branch	**:**	**B.E & CSE**	**Max Marks**	**:**	**100**
Year/Sem	**:**	**I/II**	**Duration**	**:**	**3.00 Hrs**
Course Code & Course Name	**:**	**16CSC02 & Advanced C Programming**	**Date**	**:**	

16CSC02.CO1	:	Students will able to develop programs using single dimensional and multi dimensional arrays
16CSC02.CO2	:	Students ability to perform memory access operations using pointers
16CSC02.CO3	:	Students able to solve real time applications using Functions
16CSC02.CO4	:	Students have utilize memory efficiently using structures and union
16CSC02.CO5	:	Students able to design programs to perform operations on files

PART-A (10x2=20Marks)
Answer all the questions

Q. No.	Questions	BT Level	Course Outcome

1.	Write the value of b[0] in the following program?
	main()
	{
	Int a[5]={1,3,6,7,0};
	Int *b;
	b=&a[2];
	}
	b[0] is address of a + 4 bytes4.
		K1	CO1

2.	Give an example for compile-time array initialization. Example for compile-time array initialization: int rank_list [5] = {0,0,0,0,0} ; /* Declares & initializes an array */	K1	CO1
3.	Determine the output for the following printf statements? main() { char *ptr=”string”; printf(“%c”,*ptr++); printf(“%c”,*(++ptr)); printf(“%c”,(*ptr)++);	K4	CO2

```
printf("%c",++*ptr);
}
```

Output:

Srrn

4. Write the output of the program? K3 CO2

```
main() junk(int i, int j)
{ {
int i=5;j=2; i=i*j; junk(i,j); j=i*j; printf("\n %d %d",i,j); }
}
```

Output:

1. 2 2.

5. Define recursion? K1 CO3

If a function calls itself again and again, then that function is called Recursive function

6. Write the purpose of the function main()? K4 CO3

The function main () invokes other functions within it. It is the first function to be called when the program starts execution.

7. Define Union. K1 CO4

Union is a collection of variables similar to structure. The union requires bytes that are equal to number of bytes required for the largest number

8. List out the pre-processor directives? K5 CO4

· Macro Inclusion

· Conditional Inclusion

· File Inclusion

9. State file opening? K1 CO5

The action of connecting a program to a file is called opening of a file. This requires creating an I/O stream before reading or writing the data.

10. Define file pointer? K6 CO5

The pointer to

a FILE data type is called as a stream pointer or a file pointer. A file pointer points to the block of information of the stream that had just been opened.

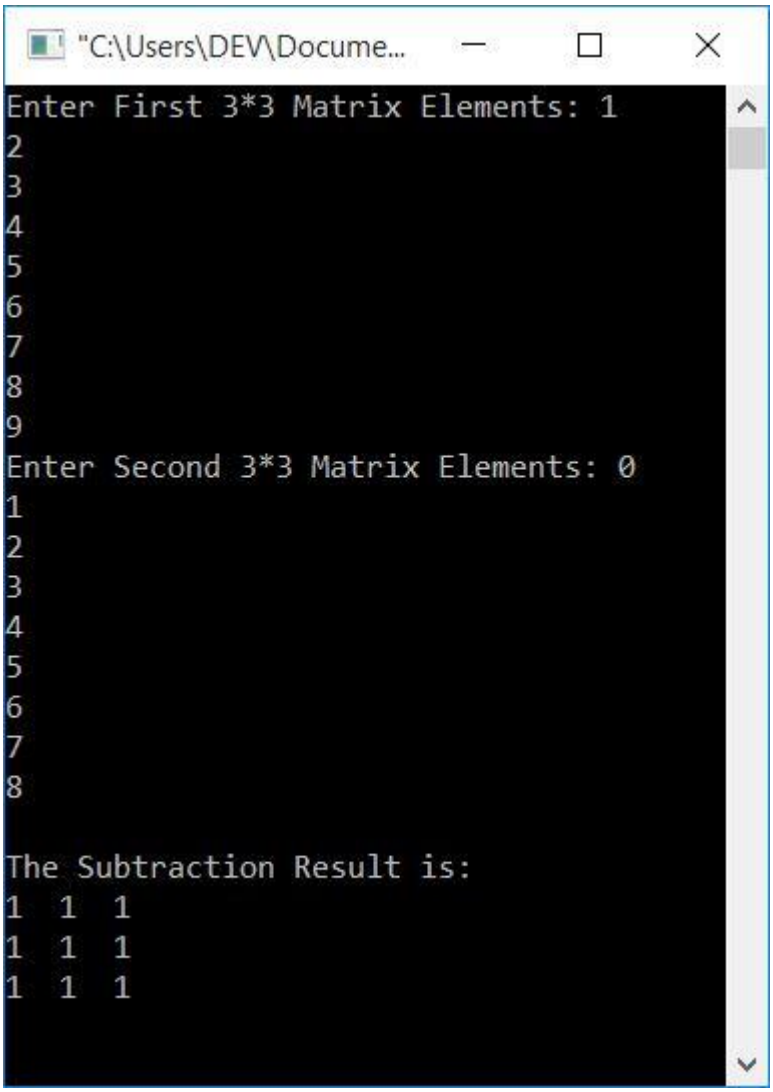
PART – B (5x16=80 Marks)

<u>Answer all the questions</u>

Q.No.		Questions	BT Level	Course Outcome
11	(a)	Implement student database using array. Arrays possess many properties. In order to work with them, we need to remember some key points: 1. An array can only have elements of the same data type. 2. The system stores the elements of an array in subsequent memory locations. 3. An array index starts with 0 and goes up to n-1 where n is the number of elements. 4. In C programming, it supports multidimensional arrays. For example, a two-dimensional array. 5. In the case of two-dimensional arrays, the system stores the elements row by row in subsequent memory locations. 6. The array name refers to the address of the first element. 7. Array size is always constant and not a variable. 8. If we have to pass an array to a function, we can refer to it by defining a pointer to the specific array 9. We generate the pointer to the first element of an array by the array name without index. 10. In C we have a function that can return an array.	K2	CO1
		(OR)		

	(b)	Explain all steps to Subtraction of two matrix with an example program.
		
		#include<stdio.h>
		#include<conio.h>
		int main()
		{
		int mat1[3][3], mat2[3][3], matSub[3][3], i, j;
		printf("Enter First 3*3 Matrix Elements: ");
		for(i=0; i<3; i++)
		{
		for(j=0; j<3; j++)
		scanf("%d", &mat1[i][j]);
		}
		printf("Enter Second 3*3 Matrix Elements: ");
		for(i=0; i<3; i++)
		{
		for(j=0; j<3; j++)
		scanf("%d", &mat2[i][j]);
		}
		for(i=0; i<3; i++)
		{
		for(j=0; j<3; j++)
		matSub[i][j] = mat1[i][j] - mat2[i][j];
		}
		printf("\nThe Subtraction Result is:\n");
		for(i=0; i<3; i++)
		{
		for(j=0; j<3; j++)
		printf("%d ", matSub[i][j]);
			K4	CO1

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



12 (a) Predefined macros in ANSI C K3 CO2

1. **__LINE__ Macro**: __LINE__ macro contains the current line number of the program in the compilation. It gives the line number where it is called. It is used in generating log statements, error messages, throwing exceptions and debugging codes. Whenever the compiler finds an error in compilation it first generates the line number at which error occurred using __LINE__ and prints error message along with line number so that user can easily fix that error easily.

- C

```
#include <stdio.h>
int main()
{
    printf("Line number is: %d\n",
           __LINE__);
    return 0;
```

```
}
```

2. **Output:**
3. `Line number is: 5`

4. **__FILE__ Macro**: __FILE__ macro holds the file name of the currently executing program in the computer. It is also used in debugging, generating error reports and log messages.

- C

```
#include <stdio.h>
int main()
{
    printf("File name of this"
           " program is: %s\n",
           __FILE__);
    return 0;
}
```

5. **Output:**
6. *File name of this program is: /usr/share/IDE_PROGRAMS/C/other/703ad0b087fbd7d18cde5ea81f148f36/703ad0b087fbd7d18cde5ea81f148f36.c*

7.
8.
9. **__DATE__ Macro**: __DATE__ macro gives the date at which source code of this program is converted into object code. Simply put, it returns the date at which the program was compiled. Date is in the format *mmm dd yyyy*. mmm is the abbrevated month name.

- C

```
#include <stdio.h>
int main()
{
    printf("Program Compilation Date: %s\n",
           __DATE__);
    return 0;
}
```

10. **Output:**
11. `Program Compilation Date: Dec 26 2019`

- __

```
#include <stdio.h>
int main()
{
    printf("Time of compilation is: %s\n",
           __TIME__);
    return 0;
}
```

12. **Output:**
13. `Time of compilation is: 13:17:20`

-

```
#include <stdio.h>
int main()
{
    printf("Compiler Standard Number: %d\n",
            __STDC__);
    return 0;
}
```

14. **Output:**
15. `Compiler Standard Number: 1`
16. **__STDC__HOSTED Macro**: This macro holds the value 1 if the compiler's target is a hosted environment. A hosted environment is a facility in which a third-party holds the compilation data and runs the programs on its own computers. Generally, the value is set to 1.

- C

```
#include <stdio.h>
int main()
{
    printf("STDC_HOSTDED Number: %d\n",
            __STDC_HOSTED__);
    return 0;
}
```

Output:

`Compiler Standard VERSION Number: 201112`

(OR)

(b) Write the Pointers and Double Pointers concept. K2 CO2

```
#include <stdio.h>
int main()
{
    int num=123;

    //A normal pointer pr2
    int *pr2;

    //This pointer pr2 is a double pointer
    int **pr1;

    /* Assigning the address of variable num to the
     * pointer pr2
     */
    pr2 = #

    /* Assigning the address of pointer pr2 to the
     * pointer-to-pointer pr1
     */
    pr1 = &pr2;

    /* Possible ways to find value of variable num*/
    printf("\n Value of num is: %d", num);
```

```
    printf("\n Value of num using pr2 is: %d", *pr2);
    printf("\n Value of num using pr1 is: %d", **pr1);

    /*Possible ways to find address of num*/
    printf("\n Address of num is: %p", &num);
    printf("\n Address of num using pr2 is: %p", pr2);
    printf("\n Address of num using pr1 is: %p", *pr1);

    /*Find value of pointer*/
    printf("\n Value of Pointer pr2 is: %p", pr2);
    printf("\n Value of Pointer pr2 using pr1 is: %p", *pr1);

    /*Ways to find address of pointer*/
    printf("\n Address of Pointer pr2 is:%p",&pr2);
    printf("\n Address of Pointer pr2 using pr1 is:%p",pr1);

    /*Double pointer value and address*/
    printf("\n Value of Pointer pr1 is:%p",pr1);
    printf("\n Address of Pointer pr1 is:%p",&pr1);

    return 0;
}
```

Output:

```
Value of num is: 123
Value of num using pr2 is: 123
Value of num using pr1 is: 123
Address of num is: XX771230
Address of num using pr2 is: XX771230
Address of num using pr1 is: XX771230
Value of Pointer pr2 is: XX771230
Value of Pointer pr2 using pr1 is: XX771230
Address of Pointer pr2 is: 66X123X1
Address of Pointer pr2 using pr1 is: 66X123X1
Value of Pointer pr1 is:  66X123X1
Address of Pointer pr1 is: XX661111
```

13 (a) Explain about Recursive Function with suitable C program K3 CO3

Recursion is the process of repeating items in a self-similar way. In programming languages, if a program allows you to call a function inside the same function, then it is called a recursive call of the function.

```
void recursion() {
  recursion(); /* function calls itself */
}

int main() {
  recursion();
}
```

Fibonacci Series

The following example generates the Fibonacci series for a given number using a recursive function −

```
#include <stdio.h>

int fibonacci(int i) {

   if(i == 0) {
      return 0;
   }

   if(i == 1) {
      return 1;
   }
   return fibonacci(i-1) + fibonacci(i-2);
}

int  main() {

   int i;

   for (i = 0; i < 10; i++) {
      printf("%d\t\n", fibonacci(i));
   }

   return 0;
}
```

0 1 1 2 3 5 8 13

(OR)

(b) Briefly explain about function prototypes K4 CO3

Function Prototypes:

i. Function without arguments and without return type

ii. Function with arguments and without return type

iii. Function without arguments and with return type

iv. Function with arguments and with return type

i) Function without arguments and without return type

o In this type no argument is passed through the function call and no output is return to main function

o The sub function will read the input values perform the operation and print

the result in the same block

ii) Function with arguments and without return type

o Arguments are passed through the function call but output is not return to the main function

iii) Function without arguments and with return type

o In this type no argument is passed through the function call but output is return to the main function.

iv) Function with arguments and with return type

In this type arguments are passed through the function call and output is return to the main function

14 (a) Write a c program for library management system using union K6 CO4

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<string.h>
struct library{
  char bookname[50];
  char author[50];
  int noofpages;
  float price;
};
int main(){
  struct library lib[100];
  char bookname[30];
  int i,j, keepcount;
  i=j=keepcount = 0;
  while(j!=6){
    printf("\n1. Add book information\n");
    printf("2.Display book information\n");
```

```
    printf("3. no of books in the library\n");
    printf("4. Exit");
    printf ("\n\nEnter one of the above : ");
    scanf("%d",&j);
    switch (j){
      /* Add book */
      case 1:
        printf ("Enter book name = ");
        scanf ("%s",lib[i].bookname);
        printf ("Enter author name = ");
        scanf ("%s",lib[i].author);
        printf ("Enter pages = ");
        scanf ("%d",&lib[i].noofpages);
        printf ("Enter price = ");
        scanf ("%f",&lib[i].price);
        keepcount++;
        i++;
        break;
      case 2:
        printf("you have entered the following information\n");
        for(i=0; i<keepcount; i++){
          printf ("book name = %s\n",lib[i].bookname);
          printf ("\t author name = %s\n",lib[i].author);
          printf ("\t pages = %d\n",lib[i].noofpages);
          printf ("\t price = %f\n",lib[i].price);
        }
        break;
      case 3:
        printf("\n No of books in library : %d", keepcount);
        break;
      case 4:
        exit (0);
    }
  }
  return 0;
}
```

OUTPUT:

. Add book information

2.Display book information

3. no of books in the library

4. Exit

Enter one of the above : 1

Enter book name = HarryPotter

Enter author name = hp

Enter pages = 250

Enter price = 350.6

(OR)

(b) Briefly explain about union with example K6 CO4

A **union** is a special data type available in C that allows to store different data types in the same memory location. You can define a union with many members, but only one member can contain a value at any given time. Unions provide an efficient way of using the same memory location for multiple-purpose.

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

a union type named Data having three members i, f, and str –

```
union Data {
   int i;
   float f;
   char str[20];
} data;
```

Example:

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

union Data {
   int i;
   float f;
   char str[20];
};

int main( ) {
```

```
  union Data data;

  printf( "Memory size occupied by data : %d\n", sizeof(data));

  return 0;
}
```

15 (a) What is the purpose of rewind() ? K2 CO5

The function rewind is used to bring the file pointer to the beginning of the file.

Rewind(fp);

Where fp is a file pointer.Also we can get the same effect by

feek(fp,0,0);

File handling functions	Description
fopen ()	fopen () function creates a new file or opens an existing file.
fclose ()	fclose () function closes an opened file.
getw ()	getw () function reads an integer from file.
putw ()	putw () functions writes an integer to file.
fgetc ()	fgetc () function reads a character from file.
fputc ()	fputc () functions write a character to file.
gets ()	gets () function reads line from keyboard.
puts ()	puts () function writes line to o/p screen.
fgets ()	fgets () function reads string from a file, one line at a time.
fputs ()	fputs () function writes string to a file.
feof ()	feof () function finds end of file.
fgetchar ()	fgetchar () function reads a character from keyboard.
fprintf ()	fprintf () function writes formatted data to a file.
fscanf ()	fscanf () function reads formatted data from a file.
fputchar ()	fputchar () function writes a character onto the output keyboard input.
fseek ()	fseek () function moves file pointer position to given locatio
SEEK_SET	SEEK_SET moves file pointer position to the beginning of t
SEEK_CUR	SEEK_CUR moves file pointer position to given location.
SEEK_END	SEEK_END moves file pointer position to the end of file.
ftell ()	ftell () function gives current position of file pointer.

rewind ()	rewind () function moves file pointer position to the beginning of the file.
getc ()	getc () function reads character from file.
getch ()	getch () function reads character from keyboard.
getche ()	getche () function reads character from keyboard and echoes to o/p screen.
getchar ()	getchar () function reads character from keyboard.
putc ()	putc () function writes a character to file.
putchar ()	putchar () function writes a character to screen.
printf ()	printf () function writes formatted data to screen.
sprinf ()	sprinf () function writes formatted output to string.
scanf ()	scanf () function reads formatted data from keyboard.
sscanf ()	sscanf () function Reads formatted input from a string.
remove ()	remove () function deletes a file.
fflush ()	fflush () function flushes a file.

(OR)

(b) Write a c program for reading a file and closing a file. K5 CO5

```
#include <stdio.h>
int main()
{
    /* Pointer to the file */
    FILE *fp1;
    /* Character variable to read the content of file */
    char c;

    /* Opening a file in r mode*/
    fp1= fopen ("C:\\myfiles\\newfile.txt", "r");

    /* Infinite loop –I have used break to come out of the loop*/
    while(1)
    {
      c = fgetc(fp1);
      if(c==EOF)
         break;
      else
         printf("%c", c);
    }
    fclose(fp1);
    return 0;
}
```

Bloom's Taxonomy Level

Bloom's Taxonomy Level	K1 Remembering	K2 Understanding	K3 Applying	K4 Analyzing	K5 Evaluating	K6 Creating	Total
% of	9	28	18	19	9	18	100%

Questions							

Course Faculty **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.

MUST KNOW CONCEPTS

MKC

CSE

2016-17

Course Code & Course Name : 16CSC02 & Advanced C Programming

Year/Sem/Sec : I/II/A

S.No.	Term	Notation (Symbol)	Concept / Definition / Meaning / Units / Equation / Expression	Units
Unit-I : Arrays				
1.	Arrays		An array is a collection of similar data elements.The elements of the array are stored in consecutive memory locations and are referenced by an index (also known as the subscript).	
2.	Linear Search		Linear search is also called sequential search. Linear search is a method for searching a value within a array.	
3.	Binary Search		Binary search works on sorted arrays. Binary search begins by comparing an element in the middle of the array with the target value.	
4.	Two Dimension(2D-Array)		A two dimensional array is specified using two subscripts where one subscript denotes row and the other denotes column.	
5.	int main		int main means that our function needs to return some integer at the end of the execution and we do so by returning 0 at the end of the program.	
6.	Types of function		Predefined functions User defined functions	

7.	How to read the Matrix		for (c = 0; c < m; c++)
			for (d = 0 ; d < n; d++)
			scanf("%d", &second[c][d]);	

8.	Formula for Addition of two matrix		sum[c][d] = first[c][d] + second[c][d];	
9.	/n		New Line	
10.	scanf( )		scanf( ) allows to read more than just a single character at a time.	

11.	Why header files are included in ‘C’ programming?		Each header file has ‘h’ extension and include using ’# include’ directive at the beginning of a program.	
12.	Define delimiters in ‘C’.		: ; () [] {} # ,	
13.	What is meant by Recursive function?		If a function calls itself again and again, then that function is called Recursive function.	
14.	Is it possible to place a return statement anywhere in ‘C’ program?		Yes. The return statement can occur anywhere.	
15.	types of errors occurred in C program		1. Syntax errors 2. Runtime errors 3. Logical errors 4. Latent errors	
16.	What are the types of Arrays?		1.One-Dimensional Array 2. Two-Dimensional Array 3. Multi-Dimensional Array	
17.	typedef		It is used to create a new data using the existing type. Syntax: typedef data type name;	
18.	Operator overloading		Operator overloading is a compile-time polymorphism in which the operator is overloaded to provide the special meaning to the user-defined data type	
19.	Function overriding		Function overriding is a feature that allows us to have a same function in child class which is already present in the parent class.	
20.	Arrays		An array is a collection of similar data elements.The elements of the array are stored in consecutive memory locations and are referenced by an index (also known as the subscript).	
21.	Linear Search		Linear search is also called sequential search. Linear search is a method for searching a value within a array.	
22.	Binary Search		Binary search works on sorted arrays. Binary search begins by comparing an element in the middle of the array with the target value.	
23.	Two Dimension(2D-		A two dimensional array is specified using two subscripts where one	

	Array)		subscript denotes row and the other denotes column.	
24.	int main		int main means that our function needs to return some integer at the end of the execution and we do so by returning 0 at the end of the program.	
25.	Types of function		Predefined functions User defined functions	
Unit-II : Pointers & Preprocessor Directives				
26.	Pointers		A pointer is a variable whose value is the address of another variable	
27.	Declaring a pointer		type. Datatype * pointer-name;	
28.	Accessing a Variable Through its Pointer		The indirection operator (*) is used to access the value of a variable by its ptr * can be remembered as value at address	
29.	Null pointer		A pointer is said to be null pointer when its right value is 0. A null pointer can never point to valid data.	
30.	Pointer to pointer		chain of pointers. int **var;	
31.	& var		Address of var variable	
32.	*var		Value of *ip variable	
33.	Pointers		A pointer is a variable whose value is the address of another variable	
34.	Declaring a pointer		type. Datatype * pointer-name;	
35.	Accessing a Variable Through its Pointer		The indirection operator (*) is used to access the value of a variable by its ptr * can be remembered as value at address	
36.	int main		int main means that our function needs to return some integer at the end of the execution and we do so by returning 0 at the end of the program.	
37.	Types of function		Predefined functions User defined functions	

38.	How to read the Matrix		for (c = 0; c < m; c++)
			for (d = 0 ; d < n; d++)
			scanf("%d", &second[c][d]);	

39.	Formula for Addition of two matrix		sum[c][d] = first[c][d] + second[c][d];	

40.	/n		New Line	
41.	scanf( )		scanf( ) allows to read more than just a single character at a time.	
42.	Why header files are included in ‘C’ programming?		Each header file has ‘h’ extension and include using ’# include’ directive at the beginning of a program.	
43.	Define delimiters in ‘C’.		: ; () [] {} # ,	
44.	What is meant by Recursive function?		If a function calls itself again and again, then that function is called Recursive function.	
45.	Is it possible to place a return statement anywhere in ‘C’ program?		Yes. The return statement can occur anywhere.	
46.	int main		int main means that our function needs to return some integer at the end of the execution and we do so by returning 0 at the end of the program.	
47.	Types of function		Predefined functions User defined functions	

48.	How to read the Matrix		for (c = 0; c < m; c++)
			for (d = 0 ; d < n; d++)
			scanf("%d", &second[c][d]);	

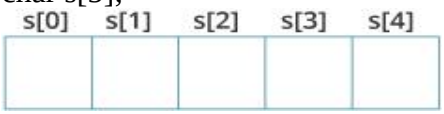
49.	Formula for Addition of two matrix		sum[c][d] = first[c][d] + second[c][d];	
50.	/n		New Line	
Unit-III : Functions				
51.	Functions		A function is a self-contained block or a sub-program of one or more statements that performs a special task when called.	
52.	Function Declaration		Function declaration 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.	
53.	Void Function		A function with void result type ends either by reaching the end of the function or by executing a return statement with no returned value.	
54.	Function Call		A function call is a request made by a program that performs a predetermined function	

55.	Call By Value		Call by value in which values of the variables are passed by the calling function to the called function.	
56.	Call By Reference		Call by reference in which address of the variables are passed by the calling function to the called function.	
57.	Arrays		An array is a collection of similar data elements.The elements of the array are stored in consecutive memory locations and are referenced by an index (also known as the subscript).	
58.	Linear Search		Linear search is also called sequential search. Linear search is a method for searching a value within a array.	
59.	Binary Search		Binary search works on sorted arrays. Binary search begins by comparing an element in the middle of the array with the target value.	
60.	Two Dimension(2D-Array)		A two dimensional array is specified using two subscripts where one subscript denotes row and the other denotes column.	
61.	int main		int main means that our function needs to return some integer at the end of the execution and we do so by returning 0 at the end of the program.	
62.	Types of function		Predefined functions User defined functions	

63.	How to read the Matrix		for (c = 0; c < m; c++)
			for (d = 0 ; d < n; d++)
			scanf("%d", &second[c][d]);	

64.	Formula for Addition of two matrix		sum[c][d] = first[c][d] + second[c][d];	
65.	/n		New Line	
66.	scanf( )		scanf( ) allows to read more than just a single character at a time.	
67.	Why header files are included in ‘C’ programming?		Each header file has ‘h’ extension and include using ’# include’ directive at the beginning of a program.	
68.	Define delimiters in ‘C’.		: ; () [] {} # ,	
69.	What is meant by Recursive function?		If a function calls itself again and again, then that function is called Recursive function.	

70.	Is it possible to place a return statement anywhere in 'C' program?		Yes. The return statement can occur anywhere.	
71.	types of errors occurred in C program		1. Syntax errors 2. Runtime errors 3. Logical errors 4. Latent errors	
72.	What are the types of Arrays?		1.One-Dimensional Array 2. Two-Dimensional Array 3. Multi-Dimensional Array	
73.	typedef		It is used to create a new data using the existing type. Syntax: typedef data type name;	
74.	Operator overloading		Operator overloading is a compile-time polymorphism in which the operator is overloaded to provide the special meaning to the user-defined data type	
75.	Function overriding		Function overriding is a feature that allows us to have a same function in child class which is already present in the parent class.	
Unit-IV : Structure And Union				
76.	String		sequence of characters	
77.	String Functions in C		1strcpy(s1,s2); 2strcat(s1, s2); 3strlen(s1); 4strcmp(s1,s2); 5strchr(s1,ch); 6strstr(s1,s2);	
78.	typedef		It is used to create a new data using the existing type. Example: typedef int hours: hours hrs;/* Now, hours can be used as new datatype */	
79.	Structure		Use to combine data of different types together	

80.	How to find the length of a string		Given a string str. Using function strlen(Str)	
81.	Converting String Into Uppercase		Convert a String to Uppercase in C using function strupr ()	
82.	Converting string into lower case		Convert a String to Lowercase in C using function strlwr(chars)	
83.	Declare a string		char s[5];  	
84.	Initialize strings		char c[] = "abcd"; char c[50] = "abcd";	
85.	Passing Strings to Functions		Strings can be passed to a function in a similar way as arrays.	
86.	Strings and Pointers		Similar like arrays, string names are "decayed" to pointers. Hence, you can use pointers to manipulate elements of the string.	
87.	Structure		Use to combine data of different types together	

88.	Declaring Structure Variables		struct structure _ name
			{
			structure_element 1;
			structure_element 2;
			structure_element 3;
			};
			struct structure_name v1,v2…vn;
			
			v1,v2….vn are structure variable	

89.	Declaring structure using pointer variable		struct student *report, rep;	
90.	Accessing structure members using normal variable		report.mark; report.name; report.average;	
91.	Accessing structure members using pointer variable		eport -> mark; report -> name; report -> average;	
92.	Rules for declaring a structure		The template is terminated with a semicolon The tag name such as book _ bank can be used to declare structure variables of its type, later in the program.	
93.	Accessing structure elements		After declaring the structure type, variables and members, the member of the structure can be accessed by using the structure variable along with the dot(.) operator.	
94.	Access pointer members of a structure		-> - Structure pointer operator	

95.	Structure Initialization		Like any other data type, a structure variable can be initialized at compile time.
			
			main()
			{
			struct
			{
			int weight; float height;
			}
			student ={60, 180.75};	

96.	Nested structures		Structure with in another structure is called nested structure	
97.	Copying and Comparing Structure Variable		Two variables of the same structure type can be copied the same way as ordinary variables. If e1 and e2 belong to the same type, then the following statement is valid. e1 = e2, and e2 = e1;	
98.	Structure Can Be Accessed In two Ways In a C Program		Using normal structure variable Using pointer variable	
99.	Pointers to Structures		struct Books *struct_pointer;	
100.	Structure Padding		In order to align the data in memory one or more empty bytes (addresses) are inserted (or left empty) between memory addresses which are allocated for other structure members while memory allocation	
Unit-V : Files				
101.	FILE		File is a collection of bytes that is stored on secondary storage devices like disk	
102.	Opening/Creating a file		fopen() – To open a file FILE *fopen (const char *filename, const char *mode)	
103.	Closing a file		fclose() – To close a file Declaration int fclose(FILE *fp);	
104.	Reading a file		fgets() – To read a file char *fgets(char *string, int n, FILE *fp)	
105.	Writing in a file		fprintf() – To write into a file Declaration: int fprintf(FILE *fp, const char *format);	

106.	File Processing		A file represents a sequence of bytes, regardless of it being a text file or a binary file.	
107.	Opening Files		fopen( ) function is used to create a new file or to open an existing file	
108.	fseek ()		fseek () function moves file pointer position to given location.	
109.	SEEK_SET		SEEK_SET moves file pointer position to the beginning of the file	
110.	SEEK_CUR		SEEK_SET moves file pointer position to the beginning of the file.	
111.	SEEK_END		SEEK_END moves file pointer position to the end of file.	
112.	ftell ()		ftell () function gives current position of file pointer	
113.	rewind ()		rewind () function moves file pointer position to the beginning of the file.	
114.	remove ()		remove () function deletes a file.	
115.	fflush ()		fflush () function flushes a file.	
116.	File mode		r- Opens an existing text file for reading purpose w- Opens a text file for writing **a-** appending mode r+ / w+ - Opens file for both reading and writing	
117.	Sequential access file		A sequential access file is such that data are saved in sequential order: one data is placed into the file after another	
118.	Random access file		If the amount of data stored in a file is fairly large, the use of random access will allow you to search through it quicker.	
119.	FILE		File is a collection of bytes that is stored on secondary storage devices like disk	
120.	Opening/Creating a file		fopen() – To open a file FILE *fopen (const char *filename, const char *mode)	
121.	Closing a file		fclose() – To close a file	

			Declaration int fclose(FILE *fp);	
122.	Reading a file		fgets() – To read a file char *fgets(char *string, int n, FILE *fp)	
123.	Writing in a file		fprintf() – To write into a file Declaration: int fprintf(FILE *fp, const char *format);	
124.	File Processing		A file represents a sequence of bytes, regardless of it being a text file or a binary file.	
125.	Opening Files		fopen( ) function is used to create a new file or to open an existing file	
Placement Questions				
126.	iterator protocol		iter()- To create an iterator next()- To iterate to the next element	
127.	Tuple packing		we place value into a new tuple	
128.	Tuple unpacking		we extract those values back into variables.	
129.	frozen set		Frozen set is immutable ,we cannot change its values.	
130.	Dogpile effect		In case the cache expires, what happens when a client hits a website with multiple requests is what we call the dogpile effect.	
131.	JSON		JSON stands for JavaScript Object Notation.	
132.	Garbage collection		form of automatic memory management which attempts to reclaim no longer use of memory	
133.	sub()		This looks for all substrings where the regex pattern matches, and replaces them with a different string	
134.	subn()		Like sub(), this returns the new string and the number of replacements made	
135.	map()		This function applies a function to each element in the iterable.	
136.	filter()		This function lets us keep the values that satisfy some conditional logic.	

137.	reduce()		This function reduces a sequence pair	
138.	Lamda()		A lambda function is a small anonymous function.It can take any number of arguments, but can only have one expression.	
139.	Is C call-by-value or call-by-reference?		C is neither call-by-value, nor call-by-reference. It is call-by-object-reference	
140.	__init__()		__init__() is what we need to initialize a class when we initiate it.	
141.	Case sensitive		Python is a case-sensitive language. This means, Variable and variable are not the same	
142.	JSON		JSON stands for JavaScript Object Notation.	
143.	Extension of python file		PY is a script file format used by Python	
144.	pointer		Variable that contains address of another variable	
145.	Structure		Structure is another user defineddata type available in C that allows to combine data items of different kinds.	
146.	Union		A union is a special data type available in C that allows to store different data types in the same memory location.	
147.	Parameter		It refers to any declaration within the parentheses following the function name in a function definition;	
148.	argument		It refers to any expression within the parentheses of a function call.	
149.	Formal Parameter		A variable and its type as they appear in the prototype of the function or method.	
150.	Actual Parameter		The variable corresponding to a formal parameter that appears in the function or method call in the calling environment.	

Faculty Team Prepared **Signatures** **HoD**

1.

2.

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.

Department of Computer Science and Engineering
Useful Website / E-Content Details - Academic Year (2020-21)

Course Code & Course Name : **16CSC02 & Advanced C Programming**

Name of the Faculty : **N.Anand**

Year/Sem/Sec : **I /II/A**

Unit-I : Arrays

1. https://www.youtube.com/watch?v=3lqgdqoY83o&list=PLBlnK6fEyqRi0Va6znG73P52rFfXD5fhs&index=1
2. https://www.youtube.com/watch?v=HEiPxjVR8CU&list=PLBlnK6fEyqRi0Va6znG73P52rFfXD5fhs&index=4
3. https://www.programiz.com/c-programming/c-arrays-functions
4. https://www.dummies.com/programming/c/how-to-use-arrays-and-functions-together-in-c-programming/
5. https://www.journaldev.com/30808/arrays-in-c

Unit-II : Pointers & Preprocessor Directives

1. https://www.youtube.com/watch?v=H9dECCBeLQk
2. https://www.guru99.com/c-loop-statement.html
3. https://www.youtube.com/watch?v=JxjUyJkjGHI
4. https://link.springer.com/article/10.1007/s40753-019-00108-2
5. https://www.youtube.com/watch?v=Bv1LcqhqnZs

Unit-III : Functions

1. https://www.youtube.com/watch?v=3lqgdqoY83o&list=PLBlnK6fEyqRi0Va6znG73P52rFfXD5fhs&index=1
2. https://www.youtube.com/watch?v=HEiPxjVR8CU&list=PLBlnK6fEyqRi0Va6znG73P52rFfXD5fhs&index=4
3. https://www.programiz.com/c-programming/c-arrays-functions
4. https://www.dummies.com/programming/c/how-to-use-arrays-and-functions-together-in-c-programming/
5. https://www.journaldev.com/30808/arrays-in-c

Unit-IV : Structure And Union

1. https://www.journaldev.com/35071/strings-in-c-programming
2. https://www.hackerearth.com/practice/notes/array-and-strings-code-monk/
3. https://m.youtube.com/watch?v=AefKSoNpZtQ
4. https://www.youtube.com/watch?v=OuOlrC1WYPo
5. https://www.journaldev.com/35071/strings-in-c-programming

Unit-V : Files

1. https://www.youtube.com/watch?v=mKrwrQMT0a4
2. https://www.programiz.com/c-programming/c-pointers-arrays
3. https://www.guru99.com/c-function-pointers.html
4. https://www.programiz.com/c-programming/c-file-examples
5. https://www.youtube.com/watch?v=mKrwrQMT0a4

Course Faculty **HoD**

Reg. No. :

07/06/18

Question Paper Code : 40027

B.E./B.Tech. DEGREE EXAMINATION, APRIL/MAY 2018

Second Semester

Computer Science and Engineering

CS 8251 – PROGRAMMING IN C

(Common to : Computer and Communication Engineering/ B.Tech. Information Technology)

(Regulations 2017)

Time : Three Hours | Maximum : 100 Marks

Answer ALL questions

PART – A (10×2=20 Marks)

1. What is external storage class ?
2. How does a preprocessor work ?
3. What is an array ? Write the syntax for multi-dimensional array.
4. Design a C program for compare any two string.
5. List the advantages of recursion.
6. When null pointer is used ?
7. State the meaning of the root word struct.
8. Specify the use of typedef.
9. How can you restore a redirected standard stream ?
10. What does argv and argc indicate in command-line arguments ?

PART – B (5×13=65 Marks)

11. a) i) Explain the different types of operators used in C with necessary program. (8)

ii) Write a C program to check the integer is Palindrome or not. (5)

(OR)

b) Describe the decision making statements and looping statements in C with an example. (13)

12. a) Write the C program to multiply two matrices (two-dimensional array) which will be entered by a user. The user will enter the order of a matrix and then its elements and similarly input the second matrix. If the entered orders of two matrices are such that they can't be multiplied by each other, then an error message is displayed on the screen. (13)

(OR)

b) i) What are the different types of string function ? Describe with their purpose. (5)

ii) Write the C program to find the number of Vowels, Consonants, Digits and white space in a string. (8)

13. a) i) Explain the purpose of a function prototype. And specify the difference between user defined function and built-in functions. (8)

ii) Write the C program to find the value of sin(x) using the series up to the given accuracy (without using user defined function) also print sin(x) using library function. (5)

(OR)

b) What is difference between pass by value and pass by reference ? Write the C coding for swapping two numbers using pass by reference. (13)

14. a) Define structure in C. Also specify the pointer and structure with example. (13)

(OR)

b) i) Write a C program for accessing structure member through pointer using dynamic memory allocation. (6)

ii) Write a short note on singly linked list and specify how the node are created in singly linked list. (7)

15. a) Explain the types of file processing with necessary examples. (13)

(OR)

b) Write the C coding for finding the average of number stored in sequential access file. (13)

PART – C (1×15=15 Marks)

16. i) Write the case study of "How sequential Access file is differ from Random Access file". (10)

ii) Write a C program to write all the members of an array of structures to a file using fwrite (). Read the array from the file and display on the screen. (5)

Reg. No. : | | | | | | | | | | | | |

Question Paper Code : 80093

B.E./B.Tech. DEGREE EXAMINATIONS, APRIL/MAY 2019.

Second Semester

Computer Science and Engineering

CS 8251 — PROGRAMMING IN C

(Common to Computer and Communication Engineering/Information Technology)

(Regulation 2017)

Time : Three hours | Maximum : 100 marks

Answer ALL questions.

PART A — (10 × 2 = 20 marks)

1. Differentiate between formatted and unformatted input statements. Give one example for each.
2. What is the use of preprocessor directive?
3. Define an array.
4. Write a C function to compare two strings.
5. What is the need for functions?
6. What is the output of the following code fragment?

```
int x= 456, *p1, **p2;
p1=&x; p2=&p1;
printf ("Value of x is : %d\n", x);
printf("Value of *p1 is : %d\n", *p1);
printf ("Value of *p2 is : %d\n", *p2);
```

7. Compare and contrast a structure with an array.
8. What is the output of the following code fragment?

```
struct point
{
int x;
int y;
};
struct point origin, *pp;
main ( )
{
pp = & origin;
printf (" origin is (%d% d)\n", (*pp).x,pp → y);
}
```

9. Why files are needed?
10. What is the use of command line argument?

PART B — (5 × 16 = 80 marks)

11. (a) (i) What is the purpose of a looping statement? Explain in detail the operation of various looping statements in C with suitable examples. (12)

(ii) Write a C program to find the sum of 10 non-negative numbers entered by the user. (4)

Or

(b) (i) What is a storage class? Explain the various storage classes in C along with suitable example. (12)

(ii) Write a C program to find the largest among 3 numbers entered by the user. (4)

12. (a) Explain binary search procedure. Write a C program to perform binary search and explain. (16)

Or

(b) Discuss how you can evaluate the mean, median, mode for an array of numbers. Write the C program to evaluate the mean, median and mode for an array of numbers and explain. (16)

13. (a) What is recursion? Explain the procedure to compute $\sin(x)$ using recursive functions. Write the C code for the same. (16)

Or

(b) What is pass by reference? Explain swapping of 2 values using pass by reference in 'C'. (16)

14. (a) What is dynamic memory allocation? Explain various C functions that are used for the same with examples. (16)

Or

(b) What is a self-referential structures? Explain with suitable examples. (16)

15. (a) Explain in detail various operations that can be done on file giving suitable examples. (16)

Or

(b) Explain in detail random access in files along with the functions used for the same in C. Give suitable examples. (16)

———————

Reg. No. :

07/06/18

Question Paper Code : 40027

B.E./B.Tech. DEGREE EXAMINATION, APRIL/MAY 2018

Second Semester

Computer Science and Engineering

CS 8251 – PROGRAMMING IN C

(Common to : Computer and Communication Engineering/ B.Tech. Information Technology)

(Regulations 2017)

Time : Three Hours | Maximum : 100 Marks

Answer ALL questions

PART – A (10×2=20 Marks)

1. What is external storage class ?
2. How does a preprocessor work ?
3. What is an array ? Write the syntax for multi-dimensional array.
4. Design a C program for compare any two string.
5. List the advantages of recursion.
6. When null pointer is used ?
7. State the meaning of the root word struct.
8. Specify the use of typedef.
9. How can you restore a redirected standard stream ?
10. What does argv and argc indicate in command-line arguments ?

PART – B (5×13=65 Marks)

11. a) i) Explain the different types of operators used in C with necessary program. (8)

ii) Write a C program to check the integer is Palindrome or not. (5)

(OR)

b) Describe the decision making statements and looping statements in C with an example. (13)

12. a) Write the C program to multiply two matrices (two-dimensional array) which will be entered by a user. The user will enter the order of a matrix and then its elements and similarly input the second matrix. If the entered orders of two matrices are such that they can't be multiplied by each other, then an error message is displayed on the screen. (13)

(OR)

b) i) What are the different types of string function ? Describe with their purpose. (5)

ii) Write the C program to find the number of Vowels, Consonants, Digits and white space in a string. (8)

13. a) i) Explain the purpose of a function prototype. And specify the difference between user defined function and built-in functions. (8)

ii) Write the C program to find the value of sin(x) using the series up to the given accuracy (without using user defined function) also print sin(x) using library function. (5)

(OR)

b) What is difference between pass by value and pass by reference ? Write the C coding for swapping two numbers using pass by reference. (13)

14. a) Define structure in C. Also specify the pointer and structure with example. (13)

(OR)

b) i) Write a C program for accessing structure member through pointer using dynamic memory allocation. (6)

ii) Write a short note on singly linked list and specify how the node are created in singly linked list. (7)

15. a) Explain the types of file processing with necessary examples. (13)

(OR)

b) Write the C coding for finding the average of number stored in sequential access file. (13)

PART – C (1×15=15 Marks)

16. i) Write the case study of "How sequential Access file is differ from Random Access file". (10)

ii) Write a C program to write all the members of an array of structures to a file using fwrite (). Read the array from the file and display on the screen. (5)

11/19/AP

Reg. No. :

Question Paper Code : 90150

B.E./B.Tech. DEGREE EXAMINATIONS, NOVEMBER/DECEMBER 2019

Second Semester

Computer Science and Engineering

CS 8251 – PROGRAMMING IN C

(Common to Computer and Communication Engineering/Information Technology)

(Regulations 2017)

Time : Three Hours | Maximum : 100 Marks

Answer ALL questions

PART – A **(10×2=20 Marks)**

1. Differentiate break and continue statement in C programming language.
2. List the various types of C operators.
3. Name any two library functions used for string handling.
4. Given an array int a [10] = {101, 012, 103, 104, 105, 106, 107, 108, 109, 110}. Show the memory representation and calculate its length.
5. What are the steps in writing a function in a program ?
6. Is it better to use a macro or a function ?
7. What is register storage in storage class ?
8. Write the syntax for pointers to structure.
9. What are two main ways a file can be organized ?
10. List the File Operations in C paradigm.

PART – B **(5×16=80 Marks)**

11. a) i) Describe the structure of a C program with an example. **(8)**

ii) Explain the various operators used in C. **(8)**

(OR)

b) Explain about the various decision making and branching statements in C programming language.

12. a) i) Describe the following with respect to arrays :- need of an array, declaration of array and accessing an array element. (8)

ii) Explain in algorithm and write a C program to re-order a one-dimensional array of numbers in descending order. (8)

(OR)

b) Write a C program to find the transpose of a matrix.

13. a) i) Discuss on recursive function . Write a C program to find factorial of n using recursion. (8)

ii) Write a C program to reverse a string using recursion. (8)

(OR)

b) Explain the concept of pass by value and pass by reference with suitable example in C programming language.

14. a) Write a C program using structures to prepare the students mark statement. The number of records is created based on the user input.

(OR)

b) Write a C program using structures to prepare the employee pay roll of a company. The number of records is created based on the user input.

15. a) Explain in detail about command line arguments with an example of generating Fibonacci series of a number in C programming language.

(OR)

b) i) Write short notes on File functions in C,

1. fseek()
2. ftell()
3. rewind()
4. feof()
5. fscanf().

ii) Discuss about the modes of file handling.

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.

Department of Computer Science and Engineering
List of COs, POs, PSOs Mapping with Attainment - Academic Year (2016-17)

Course Code & Course Name	:	**16CSC02 & Advanced C Programming**
Name of the Faculty	:	**N.Anand**
Year/Sem/Sec	:	**I /II/A**

Course Outcomes (COs)

1. Develop programs using single dimensional and multi dimensional arrays
2. Perform memory access operations using pointers
3. Solve real time applications using Functions
4. Utilize memory efficiently using structures and union
5. Design programs to perform operations on files

Programme Outcomes (POs)

1. **Engineering Knowledge:** Graduates can apply mathematics, science, computing and engineering knowledge to computer science related problems.
2. **Problem Analysis:** An ability to analyze a problem, interprets data, and defines the computing system requirements which would be appropriate to the solution.
3. **Design/development of solutions :**An ability to design, implements, and evaluate a computer-based system, process, component, or program to meet desired needs.
4. **Conduct investigations of complex problems :**An ability to apply creativity in the design of systems which would help to investigate the complex problem and provide software solution.
5. **Modern Tool usage: An** ability to use the computing techniques, skills, and modern system tools necessary for practice as a Computer Science and Engineering professional.
6. **The Engineer and Society :**An ability to analyze the local and global impact of computing on individuals, organizations, and society.
7. **Environment and Sustainability :**An ability to develop and use the software systems within realistic constraints environmental, health and safety, manufacturability, and sustainability considerations.
8. **Ethics :**An Ability to understanding of professional, ethical, legal, security and social issues and responsibilities.
9. **Individual and Team work :**An ability to function effectively on teams and individually to accomplish a common goal.

10. **Communication:** An ability to communicate effectively with a range of audiences by written and oral.

11. **Project management and finance :** Ability to plan, organize and follow best practices and standards so that the project is completed as successfully by meeting performance, quality at CMM level, budget and time.

12. **Lifelong learning:** An ability to engage in Lifelong learning and continuing professional development.

Programme Specified Outcomes (PSOs)

1. **Professional Skill Development:** To understand and analyze the principles of recent technologies in Computer Science and Engineering domain and apply it to develop a new algorithm and program

2. **Analytical Skill and Problem Solving Expertise:** To understand the new challenges in Computer Science and Engineering field for career development by their analytical skill application

3. **Project development skill:** To identify the social issues and problems to develop a new product with ethics

Programme Educational Objectives (PEOs)

1. **Foundation:** To develop the students with programming skill sets with a sound foundation in mathematical, scientific and engineering fundamentals, necessary for the core concepts, focusing on knowledge up-gradation leading to technical innovations.

2. **Analytical Skill:** Capable of analyzing and specifying the requirement of the Computer Science and Information Technology system to design and develop using the contemporary tools.

3. **Leadership Skill:** The Graduates of the programme will have the competencies for communicating, planning, coordinating, organizing and decision making and they will have interpersonal skills and ethical responsibility.

4. **Employability Skill**: The graduates will practice and demonstrate the ability to use the knowledge and expertise through the continuous performances which will contribute to the society through active engagement.

Mapping of COs with POs and PSOs

Course Code & Course Name	Description of Course Outcomes	POs												PSOs		
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**1**	**2**	**3**
16CSC02 & Advanced C Programming	Able to identify the Develop programs using single dimensional and multi dimensional arrays	2	3	3	3	2	2	2	1	1	1	2	3	2	2	3
	Perform memory access operations using pointers	2	3	3	3	2	2	2	1	1	1	2	3	2	2	3
	Solve real time applications using Functions	2	3	3	3	2	2	2	1	-	1	2	3	2	2	3
	Utilize memory efficiently using structures and union	2	3	2	3	2	2	2	-	-	1	2	3	2	2	3
	Compare efficiency of various Design programs to perform operations on files	2	3	2	3	2	2	2	-	-	1	2	3	2	2	3
*(***Details**; High-3, Medium-2, Low-1, "-" No Correlation)*																

Attainment of COs with POs and PSOs

S.No.	Roll Number	Name of the Student	End Semester Result Grade
1.			
2.			
3.			
4.			
5.			
6.			
7.			
8.			
9.			
10.			
11.			
12.			
13.			
14.			
15.			
16.			
17.			
18.			
19.			
20.			
21.			
22.			
23.			
24.			
25.			
Number of Students with "S" Grade			
Number of Students with "A" Grade			
Number of Students with "B" Grade			
Number of Students with "C" Grade			
Number of Students with "D" Grade			

Summary

Target Level	**:**	**60% Students must Achieve "C" and Above**
Total Number of Students	**:**	
Total Number of Students with "C" and above "C" Grade	**:**	
% of Students with "C" and above "C" Grade	**:**	
Attainment Level	**:**	
Attainment Status (Yes/No)	**:**	
***Details**; Target level-1 (50% of "C") Target level-2 (55% of "C") , Target level-3 (60% of "C")*		

Course Faculty **HoD**

Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Engineering 15 (2011) 2932 – 2936

Procedia Engineering

www.elsevier.com/locate/procedia

Advanced in Control Engineering and Information Science

Research and Development of C Language Programming Experiment Assistant Management Platform Based on Hybrid Architecture

Ye Chen[a], Ren Zhikao[a],Chen Chunping[b]a*

[a]College of Information, Qingdao University of science & technology ,Qingdao, China
[b]Center of Modern Education Technology , Qingdao University of science & technology, Qingdao, China

Abstract

C language experiment management and the control model based on hybrid architecture of C/S and B/S are presented by studying C language programming experiment assistant platform, and apply these results to control and manage experimental process and teaching effects, and realize paperless mode of C language programming teaching and experiment process, at the same time, in a certain extent, to improve experiment teaching quality and teaching effect.

Selection and/or peer-review under responsibility of [CEIS 2011]

Keywords: C/S；B/S；C language programming ; Experimental teaching aids；Control model;

1. INTRODUCTION

C language programming is an important foundation course of computer teaching, and experiment teaching is a key step of learning in science and engineering Colleges[1], as a course that interrelated with computer, at present, by investigation, most colleges adopt experiment teaching mode is: teachers arrange experiment content in advance, and explain experiment key points in experiment teaching class, students do experiment operation freely according to arrangements of teachers, and teachers site guidance, at the same time, experiment reports as students experimental results still rest on foundation of paper ,because requirements writing content is more, experiment reports are often added to complete after school, so it is difficult to do site submitted, such mode should waste many manpowers, financial and material resources, and also exist many problems in writing and review, management is not convenient, experiment reports

* Ye Chen. Tel.: +0-86-13780629192; fax: +0-86-0532-88959036.
E-mail address: ych_321@126.com.

doi:10.1016/j.proeng.2011.08.552

also easy are lost[2]. In the information age of today, this traditional mode will be replaced by information management based on computer. In order to enable experiment teaching, reflect the characteristics of information, make its more standardized and more scientific, so designed and developed a set of C language assistant platform, it's very good to a certain extent solved the above problems, the platform is simple and practical, is good tool for teachers teaching and experiment management and student experiment of network course learning under the network environment.

2. C LANGUAGE PROGRAMMING EXPERIMENT ASSISTANT MANAGEMENT MODEL BASED ON HYBIRD ARCHITECTURE

In order to being favor of experiment teaching management and process control, design of experiment assistant platform use a hybrid architecture control model, the specific model is expressed as H, the model H includes two levels ,first-level C is C/S framework, second level B is B/S framework.

$$H=C+B \tag{1}$$

Where C/S framework achieve control of experiment process, such as student attendance statistics, student experiment status control and track locking control function to users. B/S framework can realize process control of C language experiment teaching, include assess module of experiment preview and module of releasing experiment content and module of experiment content management, module of experiment reports management and online review module of experiment reports and statistics and summary module of experiment reports scores etc.

2.1. Design of C/S framework

Here, Programs transmission protocol on the sockets interface of Windows to realize listening and connection between the server and clients. The server would collect IP address of the client computer automatically, after starting the client computer, the user would enter login page of experiment assistant platform, students can login this system after input some related information, then these students login information have been send to the server. The experiment control model of C/S framework shows as figure1.

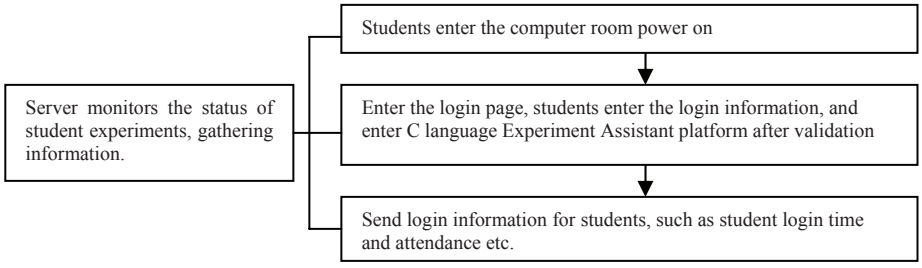


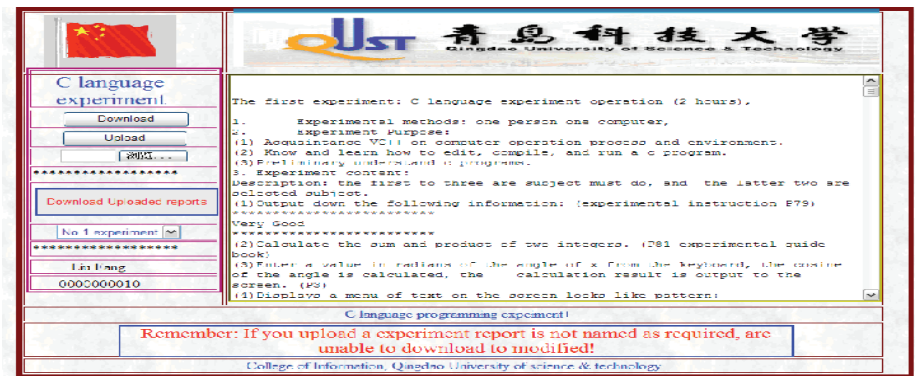
Figure 1. Control model of experiment

2.2. Experiment application process based on B/S framework

C language experiment teaching process control and management use B/S framework based on WEB, students can access all function module of the system according to experiment process arrangements by using browser after enter this system. Firstly enter examination module of experiment preview situation, then investigate students preview situation and to give examination scores, and these can provide basis for targeted counseling of teachers, at the same time, supervise and urge student doing good preview works before class, and then enter download module of experiment content, so students can read experiment instruction, download experiment requires and contents, and student can operate experiment over teacher

unification instruction. In this process, students can begin experiment operating freely according to teachers provide experiment requires by experiment content management module, and write experiment reports in term of fixed experiment report pattern, and send these reports to the server, and also can download those uploaded report to do modification operation. The management module of experiment content shows as figure 2.

Figure 2. Modules for experiment content management interface



In order to making experiment reports can be managed and reviewed expediently, the system design especially automatic filtering function to experiment reports, experiment reports that do not conform to naming regular would give error warning, these can make uploaded experiment reports more according with criterion, and management of experiment reports also more ordered. The naming rule of experiment report shows as follows:

Experiment report file name=Student ID+Name+Experiment report+Experiment serial number （2）

Where student ID select latter ten number of student in the school's roll number, the name is student name, and experiment serial number, according to experiment schedule arrangement, default is from one to ten. The reference format of experiment report is : *09001001023-Wang haihong-experiment report one.doc* . Experiment report naming detection in addition to testing the file name, and file extension name can also be detected, so as to avoid occurring of naming double extension, such as **.doc.doc*, experiment reports standardized naming would benefit of automatic classification management of experiment report.

C language programming experiment assistant management platform can assess experiment report by using fuzzy control model, and assess key points use mode of classified management and random display, can make assess results are more objective and fair. Specific algorithm realization are that each experiment set 30 small questions, and serial number from 1 to 30, when experiment began, each student use student ID to login assistant system, as student ID is 0818050123, last two bits of student ID is serial number of student, range is for 01~30, where student ID express as "Sid", actual of questions ID express as *"A"*, questions ID of logic display express to *"L"*, *L* express as following:

$$L=(A*Sid)\%31 \qquad (3)$$

Where % express modulo operation. 31 is the smallest prime number of close to 30, so preview questions of each adjacent t student have different display order, at some extent can prevent students cheat each other. When students have completed experiment preview, must to submit preview results to the server, then students can operate next experiment contents. This system can do scoring for students submitted experiment preview answers automatically, and sum to save to the server for teacher providing basis to grading on the experiment.

2.3. Design of experiment control and management

Control and management of the experiment process embody in an experiment management subsystem in B/S mode, main functions include: ① the basic maintenance of the system; ② experiment content releasing; ③ experimental report callback; ④ review of experiment report; ⑤ sum of experiment scores.

The basic maintenance of the system are completed by system administrator, it is responsible for the setting of the system, database management, addition and remove of users etc. This system can realize automatic collection and classification function of experiment reports according to course and class , the collection path of experiment report is designed as follows figure 3.

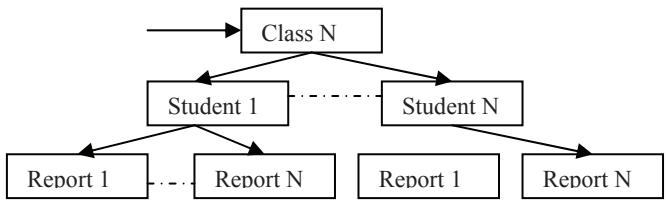


Figure 3. Save path of experiment report

Teachers use their own ID to login experiment management subsystem, then release experiment content and requirements, update experiment content, online and interact with students, review experiment report of courses, output sum of experiment scores and experiment results etc. Students registration form record student attendance of the experiment, scores of experiment preview and review scores of experiment report, as well as the final scores of the experiment, the evaluation model is set to C:.

$$C = \frac{K1}{N} * 20 + \left(\sum_{N}^{i=1} s_i / M\right) * \frac{20}{100} + \left(\sum_{N}^{i=1} b_i w_i / M\right) * \frac{60}{100} \quad (4)$$

Where C is last experiment scores, $K1$ is experimental attendance number, N is the total number of attend experiment, S_i is preview scores of each experiment, W_i is the weight of the experiment, B_i is review scores of each experiment report. This system can realize rationalization and effectiveness of experiment scores evaluation over performance evaluation model, and can reflect effect of experiment teaching and true level of student experiment process objectively.

3. APPLICATION AND REALIZING OF THE SYSTEM PLATFORM

Design of the system platform is oriented to experiment teaching of C language programming in our school, Visual Studio 2005 development platform is used, and database structure and function are design by Microsoft SQL2000/2005 database system in the system[3,4,,5,6], program codes is generated after in-depth of system feasibility research and project developed, and the system functions are achieved after tested repeatedly and make out network performance, then this system have be applied in 2010-2011 second semester of school year. In applying process of C language programming experiment assistant platform, in order to grasping situation of students application accurately, and knowing actual opinions of students to this system, so as to improve and perfect this system, where questionnaire survey are provided in three class student of pharmaceutical profession, and provide ninety one questionnaires, and take back ninety one questionnaires. The statistical results of selected issues in statistic questionnaire shows as table 3.1.

Table 3.1 Student questionnaires of C language programming experiment assistant platform

Survey content	Option A Check rate（%）	Option B Check rate（%）	Option C Check rate（%）
1、Do you think the application of the system for the C language to improve the quality of experiment teaching ?	Necessary, 100	General , 0	The effect is not significant 0
2、Do you think using experiment teaching system of this form	Relatively new , 71.4	General , 26.4	Need for further innovation 2.2
3、Do you think this way of teaching for the situation of knowledge	Some help , 86.8	General , 9.9	Still not a perfect 3.3
4、Do you think this way for experiment teaching	Useful supplement 93.2	Have some effect 4.4	The effect is not significant 2.4
5、System application for the effect of experimental teaching	Some help, 92.3	General,,,5.5	The effect is not significant 2.2
6、System application for experiment teaching content	Some complementary role , 94.3	General, 3.4	The effect is not significant 2.3

It can see from investigation results: Application of the system improve experiment teaching quality of C language programming, this one point get consistent recognition for investigated students, it reflext reform of C language programming course is successful, although exist some problems in actual application, helpful role is significantly for students learning and teaching of teachers, the system would be amended according to students pertinent advice.

4. Conclusions

Research thought of this system is based on rationalizing experiment teaching process and standardizing experiment teaching mode for starting point, experiment process are standardized by using Man-computer dialogue, this can simplify tasks of teachers in experimental teaching, highlight guidance role of teachers, makes teachers can real-time understand students experimental status, instruct student effectively in time.This system can create experiment report model automatically, and teachers can also read and review students experiment reports by this system, experiment report scores can reflect real student situation justly and accurately, so make teachers hold first-hand information of students experimental process truly, at the same time ,the system can analyze and statistics experimental report, these can provide teachers strong support of experiment sum and follow-up experiment guidance after class, to some extent standardize experiment steps, enhance experiment effect and efficiency, these provide teacher and students creating experimental environment of more science standardization.

References

[1] Ren Zhikao, Ye Chen.Liu Guozhu, Application and Research of C Language Programming Examination System Based on B/S, Third International Symposium on Information Processing,2010: 316-319

[2] Visual Studio.NET2003 document［OL］Microsoft Corporation, 2003.

[3] Karli Watson Christian Nagel A programmer's Introduction to C# [M], 2006.5.1

[4] YAN Wei, CAO Baoxing, XIA Xiaona, "Design and implementation of postgraduate online exam system"[J], App lica tion Re sea rch of Computers. NO 2, Vol 26, Feb 2009, pp:637-640.

[5] DU Feng, ZHANG Ying-jie, CHEN Lu, "Research on general paperless examination system"[J]. Journal of ZheJiang University ot Technology.NO 4, Vol 36, Aug 2009,pp:382-385.

[6] Li Ze-Zhong, DENG Pu. Research on Intelligent Test Paper Composition Based on Genetic Algorithm[J], Journal of Chongqing Electric Power College. NO 4, Vol 13, Dec.2008,pp:25-28.

C Programming MCQ Questions & Answers Pdf

Question: 1

In C++, a function contained within a class is called

(A) a method

(B) a class function

(C) member function

(D) none of these

Ans: C

member function

Question: 2

The && and || operators compare two

(A) boolean values

(B) boolean value

(C) numeric values

(D) numeric value

Ans; B

boolean value

Question: 3

A pointer is

(A) address of a variable

(B) a variable for sorting address

(C) data type of an address variable

(D) indication of the variable to be accessed next

Ans: B

a variable for sorting address

Question: 4

The function abort() is declared in the header file

(A) <math.h>

(B) <iostream.h>

(C) <stdio.h>

(D) <stdlib.h>

Ans: B

<iostream.h>

Question: 5

In C++, when accessing a structure member, the identifier to the left of the dot operator is the name of

(A) structure tag

(B) structure member

(C) structure variable

(D) keyword struck

Ans: C

structure variable

Survey of Programming Languages

C Lecture 1 : Getting Started: in C

Modified from Dr. Robert Siegfried's Presentation

Objective

- Intro to C
- Tools we will use
- Program file structure
- Variables
- Read from screen and print to screen
- Decisions (If)

C Orientation

- Created in 1972 to write operating systems (Unix in particular)
  - By Dennis Ritchie
  - Bell Labs
- Evolved from B
- Can be portable to other hardware (with careful design – use Plauger's The Standard C Library book)
- Built for performance and memory management – operating systems, embedded systems, real-time systems, communication systems

C Standardization

- 1989 ANSI and ISO -> Standard C
- 1999 C99
- 2011 C11

- Don’t get thrown when you lookup information on websites and find conflicts based upon standards

Later Languages

- 1979 C++ by Bjarn Stroustrup also at Bell
  - Object orientation
- 1991 Java by Sun
  - Partial compile to java bytecode: virtual machine code
  - Write once, run anywhere
  - Memory manager – garbage collection
  - Many JVMs written in C / C++

A First Program

makes input and output available to us

```
#include    <stdio.h>

int         main(void)
{
        printf("This is my first C program.\n");
        return(0);
}
```

header

open and close braces mark the beginning and end

statements

A First Program – What Does It Do?

```
printf("This is my first C program.\n");
return(0);
```

Prints the message

```
This is my first C program.
```

Ends the program

Ends the line

Java Reminder

Program	C	Java
hello, world	#include<stdio.h>	public class HelloWorld {
	int main(void) {	public static void main(String[] args) {
	printf("Hello\n");	System.out.println("Hello");
	return 0;	}
	}	}

C Program Phases

- Editor - code by programmer
- Compiling using gcc:
  - Preprocess – expand the programmer's code
  - Compiler – create machine code for each file
  - Linker – links with libraries and all compiled objects to make executable
- Running the executable:
  - Loader – puts the program in memory to run it
  - CPU – runs the program instructions

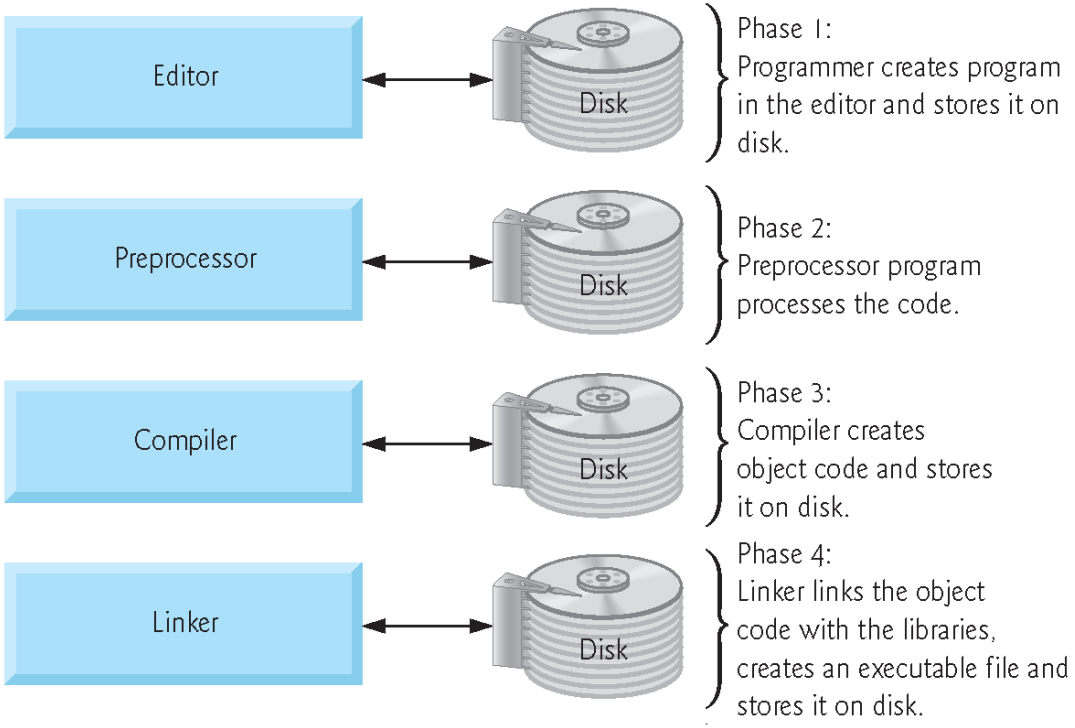


Fig. 1.7 | Typical C development environment. (Part 1 of 3.)

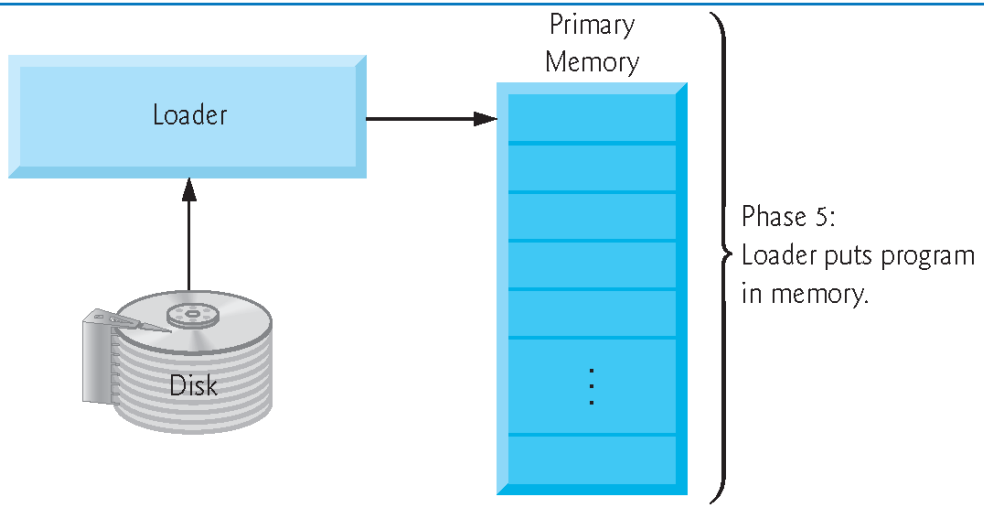


Fig. 1.7 | Typical C development environment. (Part 2 of 3.)

Run First Program

- Write in notepad++
- Transfer with Filezilla
- Connect to panther as terminal (putty) using SSH (Secure Shell)
- More filename to see the file
- gcc filename -o filename without c -g (ex: gcc hello.c -o hello -g)
- ./hello

Using variables

```
#include    <stdio.h>
int   main(void)
{
    int     sum, value1, value2, value3;
    float   average;
    value1 = 2;
    value2 = 4;
    value3 = 6;
    sum = 2 + 4 + 6;
    average = sum / 3;
    printf("The average  of %d , %d, %d is %f\n",
    value1, value2, value3, average);
    return(0);
}
```

Print a float value from the rest of the parameter list

Variables and Identifiers

- Variables have names – we call these names ***identifiers***.
- An identifier must begin with a letter or an underscore **_**
- C is case sensitive upper case (capital) or lower case letters are considered different characters. **`Average`**, **`average`** and **`AVERAGE`** are three different identifiers.
- Numbers can also appear after the first character.
- However, C only considers the first 31 (external identifiers) or first 63 (internal identifiers) significant.
- Identifiers cannot be reserved words (special words like **`int`**, **`main`**, etc.)

User Input

- Let’s rewrite the average program so it can find the average any 3 numbers we try:
- We now need to:
  1. Find our three values
  2. Add the values
  3. Divide the sum by 3
  4. Print the result

Average3.c

```
#include    <stdio.h>

int         main(void)
{
      int         value1, value2, value3;
      float sum, average;

      printf("What is the first value? ");
      scanf("%d", &value1);

      printf("What is the second value? ");
      scanf("%d", &value2);
```

Read

The address of variable value1

Indicates that we are reading an integer

```
        printf("What is the third value? ");
        scanf("%d", &value3);

        sum = value1 + value2 + value3;
        average = sum / 3;

         printf("The average  of %d , %d, %d is
%f\n", value1, value2, value3, average);
        return(0);
}
```

scanf *needs the* **&** *before the identifier*

Scanf Conversion Characters

- https://wpollock.com/CPlus/PrintfRef.htm#scanfConv
  Doubles on our machine are read with a lf.
  (A double is a long float.)

Formatting %d and **%f**

- The specifiers **%d** and **%f** allow a programmer to specify how many spaces a number will occupy and how many decimal places will be used.
- **%*n*d** will use at least n spaces to display the integer value in decimal (base 10) format.
- **%*w*.*n*f** will use at least w spaces to display the value and will have exactly n decimal places.
- Example:
  - **printf("The average of %2d , %2d, %2d is %5.2f\n", value1, value2, value3, average);**

Changing the width

Number	Formatting	Print as:
182	`%2d`	182
182	`%3d`	182
182	`%5d`	``182
182	`%7d`	```182
-182	`%4d`	-182
-182	`%5d`	`-182
-182	`%7d`	```-182

Changing the width (continued)

Number	Formatting	Print as:
23	**`%1d`**	23
23	**`%2d`**	23
23	**`%6d`**	….23
23	**`%8d`**	……23
11023	**`%4d`**	11023
11023	**`%6d`**	.11023
-11023	**`%6d`**	-11023
-11023	**`%10d`**	….-11023

Changing The Precision

Number	Formatting	Prints as:
2.718281828	`%8.5f`	`` `2.71828 ``
2.718281828	`%8.3f`	`` ```2.718 ``
2.718281828	`%8.2f`	`` ``` `2.72 ``
2.718281828	`%8.0f`	`` ```` ```3 ``
2.718281828	`%13.11f`	`2.71828182800`
2.718281828	`%13.12f`	`2.718281828000`

Average – add comments

```
#include          <stdio.h>
/*
 * This program calculates average pay
 */
int          main(void)
{
      int          value1, value2, value3;
      float sum, average;
      string
// now get the first value
 ;
```

comments

Character Data

- All of our programs so far have used variables to store numbers, not words.
- We can store one or more characters by writing:

```
char  x, s[10];
```

  - **`x`** can hold one and only one character
  - **`s`** can hold up to nine characters (reserving 1 for ending null)
- For now, we use character data for input and output only.

A program that uses a character variable

```
#include     <stdio.h>

/* A very polite program that greets you by name */
int    main(void)
{
   char       name[25];

   /* Ask the user his/her name */
   printf("What is your name ? ");
   scanf("%s", name);

   /* Greet the user */
   printf("Glad to meet you, %s\n.", name);
   return(0);
}
```

Features so far

- Include
- Variable types: int, float, char
- Read using scanf
  - requires & for address of variable being read
- Print using printf
- Format strings: %f (float), %d (int), %u (unsigned int), %c (char), %s (character array)
- Comments /*.. */ or //

if and ***if-else*** and ***if-else if - else***

```
If (boolean_expression 1)
{ /statements }
 else if ( boolean_expression 2)
{ /* statements */ }
else if ( boolean_expression 3)
 { /* statements */ }
else
{ /* statements */ }
```

IsItNeg.c – illustrate if

```
#include     <stdio.h>
// Tell a user if a number is negative
int    main(void)
{ float number;
      /* Ask the user for a number */
       printf("Please enter a number ? ");
       scanf("%f", &number);
       // Print whether the number is negative or not
       if (number < 0){
         printf("%f is a negative number\n",  number); }
       else {
        printf("%f is NOT a negative number\n",  number); }
  return(0);  }
```

Relational operators

Operator	**Meaning**	**Example**
`==`	equals	`x == y`
`!=`	is not equal to	`1 != 0`
`>`	greater than	`x+1 > y`
`<`	less than	`x-1 < 2*x`
`>=`	greater than or equal to	`x+1 >= 0`
`<=`	less than or equal to	`-x +7 <= 10`

Integer Division

- Our compound interest program prints the values for every year where every ten or twenty years would be good enough.
- What we really want to print the results only if the year is ends in a 5. (The remainder from division by 10 is 5).

Integer Division Results

8 / 3 = 2	8 % 3 = 2
2 / 3 = 0	2 % 3 = 2
49 / 3 = 16	49 % 3 = 1
49 / 7 = 7	49 % 7 = 0
-8 / 3 = -2	-8 % 3 = -2
-2 / 3 = 0	-2 % 3 = -2
-2 / -3 = 0	-2 % -3 = -2
2 / -3 = 0	2 %-3 = 2
-49 / 3 = -16	-49 % 3 = -1

Choosing Data Types

- Sizes implementation dependent in limits.h
  - int -2147483648 to 2147483647
  - short -32768 to 32767
  - long -9223372036854775808 to 9223372036854775807
  - Float 1.17×10^{-38} to $3.4 * 10^{38}$
- Keyword unsigned starts at 0 but goes higher

Declaring Constants

- There are two ways of defining constants in C: using **`#define`** and **`const`**.
- #define is a compiler preprocessor which replaces each occurrence of the constant's name with its value:
- The general form of the constant declaration is:

```
#define ConstantName      ConstantValue
```

- Let's take a look at a few examples:

```
#define   withholding_rate    0.8
#define   prompt    'y'
#define   answer    "yes"
#define   maxpeople 15
#define   inchperft 12
#define   speed_limit    55
```

Declaring Constants

- The general form of the constant declaration is:

```
const datatype ConstantName =
                        ConstantValue,
      AnotherConstantName =
                AnotherConstantValue;
```

- Let's take a look at a few examples of constants:

```
const float    withholding_rate = 0.8;
const char     prompt = 'y',
               answer[] = "yes";
const int      maxpeople = 15,
               inchperft = 12;
               speed_limit = 55;
```

Java Comparison Thus Far

Feature	C	Java
type of language	function oriented / imperative	object oriented
file naming conventions	stack.c, stack.h	Stack.java - file name matches name of class
basic programming unit	function	class / Abstract Data Type
portability of source code	possible with discipline	yes
portability of compiled code	no, recompile for each architecture	yes, bytecode is "write once, run anywhere"
compilation	gcc hello.c creates machine language code	javac Hello.java creates Java virtual machine language bytecode
buffer overflow	segmentation fault, core dump, unpredicatable program	checked run-time error exception
boolean type	use int: 0 for false, nonzero for true OR include <stdbool.h> and use bool	boolean is its own type - stores value true or false
character type	char is usually 8 bit ASCII	char is 16 bit UNICODE
strings	'\0'-terminated character array	built-in immutable String data type
accessing a library	#include <stdio.h>	import java.io.File;

Credit: http://introcs.cs.princeton.edu/java/faq/c2java.html

More Java Comparison

Feature	C	Java
printing to standard output	printf("sum = %d", x);	System.out.println("sum = " + x);
formatted printing	printf("avg = %3.2f", avg);	System.out.printf("avg = %3.2f", avg)
reading from stdin	scanf("%d", &x);	int x = StdIn.readInt();
declaring constants	const and #define	final
for loops	for (i = 0; i < N; i++)	for (int i = 0; i < N; i++)
variable auto-initialization	not guaranteed	instance variables (and array elements) initialized to 0, null, or false, compile-time error to access uninitialized variables
casting	anything goes	checked exception at run-time or compile-time
demotions	automatic, but might lose precision	must explicitly cast, e.g., to convert from long to int
variable declaration	at beginning of a block	before you use it
variable naming conventions	sum_of_squares	sumOfSquares

Credit: http://introcs.cs.princeton.edu/java/faq/c2java.html

Summary

- Tools we will use
  - Notepad++
  - Filezilla
  - Panther (gcc)
  - Putty
- Program file structure
  - #include <> or “ “
  - Main function

Summary Cont.

- Variables
  - int, float, char
  - unsigned keyword
  - String defined as char array : char name[26]
  - For bool, include stdbool.h
  - Constant:
    - #define name value
    - const type name = ?
  - Get address of variable with &
  - Cast with (type) var
-

Summary Cont.

- Read from screen and print to screen
  - Scanf (control string, variable addresses)
  - Printf(string, variables to insert)
  - Format strings %2f, %d, %s, %u
  - #include <stdio.h>
- Decisions
  - If / else if / else

Exercise

- https://prof.beuth-hochschule.de/fileadmin/user/scheffler/Lehre/Think-C_v1.08.pdf
- Exercise 2.1

Regd. No.							

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

Question Paper Code:

B.E. DEGREE EXAMINATIONS, Apr/May-2016

COMPUTER SCIENCE AND ENGINEERING

FIFTH SEMESTER

16CSC02 & Advanced C Programming

(REGULATIONS 2016)

Answer Key

Time: Three hours **Maximum: 100 Marks**

Answer **ALL** the questions

PART – A (10 X 2 = 20 Marks)

1. Write any four features of arrays.

- An array is a derived data type
- It is used to represent a collection of elements of the same data type.
- The elements can be accessed with base address.
- The starting memory location(Base address) are represented by the array name.

2. List out the types of arrays in C?

There are three types of arrays in C. They are

- One dimensional Array
- Two Dimensional Arrays and

- Three Dimensional Arrays

3. Distinguish between Library function and user defined function?

Library Functions	User-defined Functions
a) Library functions are pre-defined set of functions that are defined in C libraries. b) User can only use the function but cannot change (or) modify this function.	a) The User-defined functions are the functions defined by the user according to his/her requirement. b) User can use this type of function. User can also modify this function.

4. Write about pointer?
   Pointer is a user defined data type which creates special types of variables which can hold the address of primitive data type like char, int, float, double
5. Define is Function?

   A function is a self contained block or a sub program of one or more statements that performs a special task when called.

   Types:

   - Pre defined functions.
   - User Defined functions
6. Give the syntax for using user-defined functions in a program.

   Syntax for using user-defined functions in a program

   Syntax:

```
function declaration;          function definition;
main()                          main()
 {                             {
function calling;    }           function calling;}
```

7. Define Structures.
   A structure is a collection of one or more variables of different data types grouped together under a single name. It contains different data types.
   Syntax:

```
struct struct-name
{
type variable 1; type variable 2; type variable n;
} structure_variables;
```

8. Define Union.

Union is a collection of variables similar to structure. The union requires bytes that are equal to number of bytes required for the largest number

9. State file opening?

The action of connecting a program to a file is called opening of a file. This requires creating an I/O stream before reading or writing the data.

10. Define file pointer?

The pointer to a FILE data type is called as a stream pointer or a file pointer. A file pointer points to the block of information of the stream that had just been opened.

PART – B (5 X 16 = 80 Marks)

Answer **ALL** the questions

11.(a) Implement an array to identify max and min element using C programming.

1.Create two intermediate variables max and min to store the maximum and minimum element of the array.

2. Assume the first array element as maximum and minimum both, say max = arr[0] and min = arr[0].

3. Traverse the given array arr[].

4. If the current element is smaller than min, then update the min as the current element.

5. If the current element is greater than the max, then update the max as the current element.

6. Repeat the above two steps 4 and 5 for the element in the array.

Input: arr[] = {1, 2, 4, -1}
Output:
The minimum element is -1
The maximum element is 4
Input: arr[] = {-1, -1, -1, -1}
Output:

The minimum element is -1
The maximum element is -1

OR

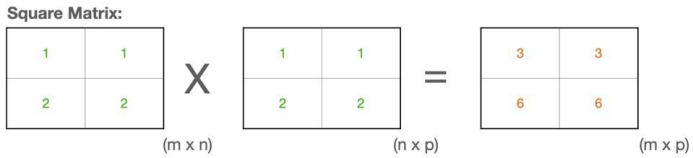
(b) Explain all steps in Multiplication of two matrix with an example program.

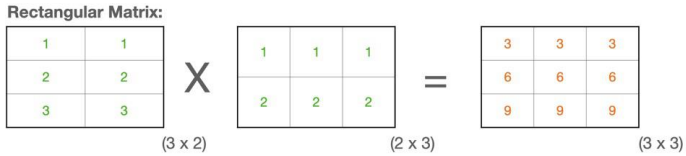
(Square Matrix Multiplication)

Input : mat1[m][n] = {
{1, 1},
{2, 2}
}
mat2[n][p] = {
{1, 1},
{2, 2}
}
Output : result[m][p] = {
{3, 3},
{6, 6}
}

(Rectangular Matrix Multiplication)

Input : mat1[3][2] = {
{1, 1},
{2, 2},
{3, 3}
}
mat2[2][3] = {
{1, 1, 1},
{2, 2, 2}
}
Output : result[3][3] = {
{3, 3, 3},
{6, 6, 6},
{9, 9, 9}
}





12.(a)Explain #include and #define preprocessor directives.

The C preprocessor versus the compiler
Please remember that, C preprocessor is not a part of C compiler. It has different syntax from normal C statements compiled by the compiler, for example:

- It start with hash/pound # character.
- C preprocessor is line oriented. Each statement start in a separate line. While it is not mandatory for other C statements to start in separate line.
- These statements end with new line. While, other statements are terminated by semicolon.

C preprocessor is a **Micro preprocessor** which compiles the code before the compilation.

List of all preprocessor directives:

- #include preprocessor directive
- #define and #undef preprocessor directive
- Parameterized Macros (Function like Macros)
- #ifdef, #ifndef and #endif preprocessor directive
- #if...#elif...#else...#endif
- Stringize operator (#)
- Token pasting (##)
- #pragma preprocessor directive
- #error preprocessor directive
- #null preprocessor directive

OR

(b) Discuss in detailed working principle of void pointers, null pointers, and array pointers with an example program.

The **Pointer** in C, is a variable that stores address of another variable. A pointer can also be used to refer to another pointer function. A pointer can be incremented/decremented, i.e., to point to the next/ previous memory location. The purpose of pointer is to save memory space and achieve faster execution time.

If you print the address of a variable on the screen, it will look like a totally random number (moreover, it can be different from run to run).

Let's try this in practice with pointer in C example

```c
#include <stdio.h>

int main() {
    int v = 0;
    printf("%d\n", &v);
    return 0;
}
```

The output of this program is -480613588.

13(a) Briefly explain about function prototypes

The Function prototype serves the following purposes –
1) It tells the return type of the data that the function will return.
2) It tells the number of arguments passed to the function.
3) It tells the data types of each of the passed arguments.
4) Also it tells the order in which the arguments are passed to the function.
Therefore essentially, the function prototype specifies the input/output interlace to the function i.e. what to give to the function and what to expect from the function.
The prototype of a function is also called the signature of the function.
What if one doesn't specify the function prototype?
The output of the below kinds of programs is generally asked at many places.

Take a step-up from those "Hello World" programs. Learn to implement data structures like Heap, Stacks, Linked List and many more! Check out our **Data Structures in C** course to start learning today.

- c

```
int main()
{
   foo();
   getchar();
   return 0;
}
void foo()
{
   printf("foo called");
}
```

OR

(b) Explain cal by value and call by reference with c program

Call By Value: In this parameter passing method, values of actual parameters are copied to function's formal parameters and the two types of parameters are stored in different memory locations. So any changes made inside functions are not reflected in actual parameters of the caller.

Call by Reference: Both the actual and formal parameters refer to the same locations, so any changes made inside the function are actually reflected in actual parameters of the caller.

Call By Value

While calling a function, we pass values of variables to it. Such functions are known as “Call By Values”.

In this method, the value of each variable in calling function is copied into corresponding dummy variables of the called function.

With this method, the changes made to the dummy variables in the called function have no effect on the values of actual variables in the calling function.

```
// C program to illustrate
// call by value

#include

// Function Prototype
void swapx(int x, int y);

// Main function
int main()
{
   int a = 10, b = 20;

   // Pass by Values
   swapx(a, b);

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

Call By Reference

While calling a function, instead of passing the values of variables, we pass address of variables(location of variables) to the function known as “Call By References.

In this method, the address of actual variables in the calling function are copied into the dummy variables of the called function.

With this method, using addresses we would have an access to the actual variables and hence we would be able to manipulate them.

```
// C program to illustrate
// Call by Reference

#include

// Function Prototype
void swapx(int*, int*);

// Main function
int main()
{
   int a = 10, b = 20;

   // Pass reference
   swapx(&a, &b);

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

Call By Value

```
    return 0;
}

// Swap functions that swaps
// two values
void swapx(int x, int y)
{
    int t;

    t = x;
    x = y;
    y = t;

    printf("x=%d y=%d\n", x, y);
}
```

Output:
x=20 y=10
a=10 b=20

Call By Reference

```
    return 0;
}

// Function to swap two variables
// by references
void swapx(int* x, int* y)
{
    int t;

    t = *x;
    *x = *y;
    *y = t;

    printf("x=%d y=%d\n", *x, *y);
}
```

Output:
x=20 y=10
a=20 b=10

14.(a) Write a c program for student mark sheet using structure

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

struct stu
{
    int rn,grade,a[5];
    float avg;
}s[2];
```

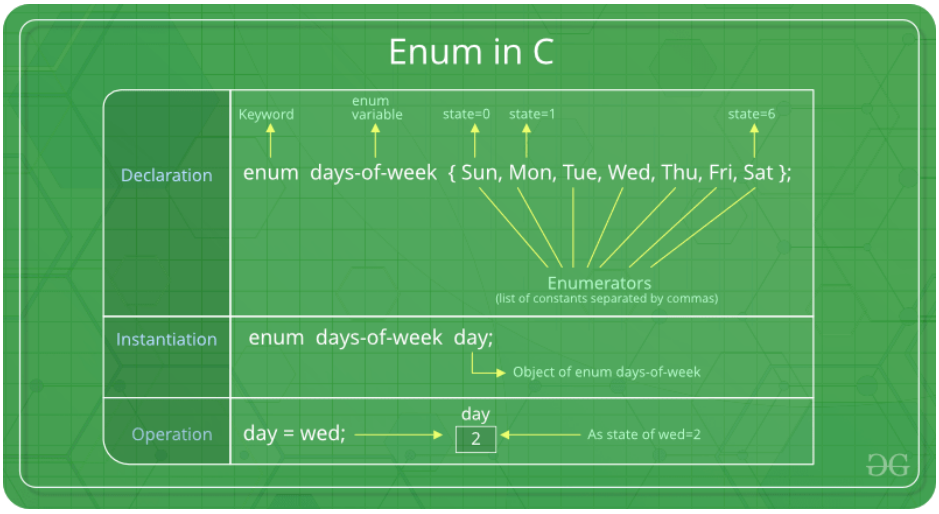
```
void main()
{
      int i,j,sum,n;
      float avg;
      clrscr();
      printf("\t STUDENT MARKSHEET USING STRUCTURES\n\n");
      printf("Enter the no of students");
      scanf("%d",&n);

for(i=0;i<n;i++)
{
      scanf("%d",s[i].rn);

for(j=0;j<=5;j++)
{
      scanf("%d",&s[i].a[j]);
   }
 }

for(i=0;i<n;i++)
{
       sum=0;

for(j=0;j<5;j++)
{
        sum=sum+s[i].a[j];
        s[i].avg=(float)(sum/5);
        if(s[i].avg>=60)
        s[i].grade=1;
        else if(s[i].avg>50&&s[i].avg<60)
        s[i].grade=2;
else
       s[i].grade=3;
  }
}

       printf("****************************************\n");
       printf("rn\ts1\ts2\ts3\ts4\ts5\tavg\t grade\n");
       printf("****************************************\n");

for(i=0;i<n;i++)
{
       printf("%d\t",s[i].rn);

for(j=0;j<5;j++)
```

```
{
      printf("%d\t",s[i].a[j]);
    }
     printf("%f\t%d\n",s[i].avg,s[i].grade);
  }
  getch();
}
```

OR

(b)Explain about Enumerated Data Type

Enumeration (or enum) is a user defined data type in C. It is mainly used to assign names to integral constants, the names make a program easy to read and maintain.



Take a step-up from those "Hello World" programs. Learn to implement data structures like Heap, Stacks, Linked List and many more! Check out our **Data Structures in C** course to start learning today.

enum State {Working = 1, Failed = 0};

The keyword ‘enum’ is used to declare new enumeration types in C and C++. Following is an example of enum declaration.

// The name of enumeration is "flag" and the constant

// are the values of the flag. By default, the values

// of the constants are as follows:

// constant1 = 0, constant2 = 1, constant3 = 2 and

// so on.

enum flag{constant1, constant2, constant3, ... };

Variables of type enum can also be defined. They can be defined in two ways:

```
// 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;
}
```

15(a)What is the purpose of rewind() ?

The function rewind is used to bring the file pointer to the beginning of the file.

Rewind(fp);

Where fp is a file pointer.Also we can get the same effect by

feek(fp,0,0);

File handling functions	Description
fopen ()	fopen () function creates a new file or opens an existing file.
fclose ()	fclose () function closes an opened file.
getw ()	getw () function reads an integer from file.
putw ()	putw () functions writes an integer to file.
fgetc ()	fgetc () function reads a character from file.
fputc ()	fputc () functions write a character to file.

gets ()	gets () function reads line from keyboard.
puts ()	puts () function writes line to o/p screen.
fgets ()	fgets () function reads string from a file, one line at a time.
fputs ()	fputs () function writes string to a file.
feof ()	feof () function finds end of file.
fgetchar ()	fgetchar () function reads a character from keyboard.
fprintf ()	fprintf () function writes formatted data to a file.
fscanf ()	fscanf () function reads formatted data from a file.
fputchar ()	fputchar () function writes a character onto the output screen from keyboard input.
fseek ()	fseek () function moves file pointer position to given location.
SEEK_SET	SEEK_SET moves file pointer position to the beginning of the file.
SEEK_CUR	SEEK_CUR moves file pointer position to given location.
SEEK_END	SEEK_END moves file pointer position to the end of file.
ftell ()	ftell () function gives current position of file pointer.
rewind ()	rewind () function moves file pointer position to the beginning of the file.
getc ()	getc () function reads character from file.
getch ()	getch () function reads character from keyboard.
getche ()	getche () function reads character from keyboard and echoes to o/p screen.
getchar ()	getchar () function reads character from keyboard.
putc ()	putc () function writes a character to file.
putchar ()	putchar () function writes a character to screen.
printf ()	printf () function writes formatted data to screen.
sprinf ()	sprinf () function writes formatted output to string.
scanf ()	scanf () function reads formatted data from keyboard.
sscanf ()	sscanf () function Reads formatted input from a string.
remove ()	remove () function deletes a file.
fflush ()	fflush () function flushes a file.

(OR)

(b)Write a c program for reading a file and closing a file.

```c
#include <stdio.h>
int main()
{
    /* Pointer to the file */
    FILE *fp1;
    /* Character variable to read the content of file */
    char c;

    /* Opening a file in r mode*/
    fp1= fopen ("C:\\myfiles\\newfile.txt", "r");

    /* Infinite loop –I have used break to come out of the loop*/
    while(1)
    {
      c = fgetc(fp1);
      if(c==EOF)
          break;
      else
          printf("%c", c);
    }
    fclose(fp1);
    return 0;
}
```

Regd. No.							

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

Question Paper Code:

B.E. DEGREE EXAMINATIONS, Apr/May-2016

COMPUTER SCIENCE AND ENGINEERING

SECOND SEMESTER

16CSC02 & Advanced C Programming

(REGULATIONS 2016)

Time: Three hours **Maximum: 100 Marks**

Answer **ALL** the questions

PART – A (10 X 2 = 20 Marks)

1. Write any four features of arrays.
2. List out the types of arrays in C?
3. Distinguish between Library function and user defined function?
4. Write about pointer?
5. Define is Function?
6. Give the syntax for using user-defined functions in a program Write the limitations of MANET?
7. Define Structures.
8. Define Union.
9. State file opening?
10. Define file pointer?

PART – B (5 X 16 = 80 Marks)

Answer **ALL** the questions

11.(a) Implement an array to identify max and min element using C programming.

Or

(b) Explain all steps in Multiplication of two matrix with an example program.

12.(a)Explain #include and #define preprocessor directives.

Or

(b) Discuss in detailed working principle of void pointers, null pointers, and array pointers with an example program.

13(a) Briefly explain about function prototypes

Or

(b) Explain cal by value and call by reference with c program

14.(a) Write a c program for student mark sheet using structure

Or

(b)Explain about Enumerated Data Type

15(a)What is the purpose of rewind() ?

Or

(b)Write a c program for reading a file and closing a file.

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.

Estd. 2000

IQAC

COURSE FILE - CHECK LIST

CoCHECK

DEPT	CSE

2016-17

Course Code & Course Name : **16CSC02 & Advanced C Programming**

Name of the Faculty : **N.Anand**

Year/Sem/Sec : **I /II/A**

S. No.	Content	Available (Yes/No)
1.	Syllabus Copy	
2.	Lesson Plan	
3.	Preamble	
4.	Subject Timetable	
5.	Minutes of the Course Committee Meeting (2 Meetings/Semester)	
6.	Course Material (Lecture Handouts)	
7.	Question Bank	
8.	CIA I – Question Paper, Answer Key & Sample Papers (3 Nos)	
9.	CIA II – Question Paper, Answer Key & Sample Papers (3 Nos)	
10.	Model Exam- Question Paper, Answer Key & Sample Papers (3 Nos)	
11.	MKC Material	
12.	Useful Websites / E-Content Details	
13.	End Semester Results and Analysis / Suggestions for Improvement in next Semester (if needed)	
14.	Previous End Semester Examination Question Papers	
15.	List CO, PO, PSO Mapping with Attainments	
16.	Any other Content	

Faculty Signature	:				
Verified by HoD	:		**Date of Verification**	:	

COURSE FILE

Academic Year:2016-2017

Course Code & Name : 16CSC02 & Advanced C Programming

Year/Semester/Sec : I / II /A

Name of the Faculty : N.Anand

Designation : ASSISTANT PROFESSOR

Department of the Faculty : CSE