|  |  |
| --- | --- |
| M1 | 5 Marks |
| M2 | 2 |
| M3 | 15 |
| M4 | 15 |
| M5 | 8 |
| M6 | 15 |
| M7 | 10 |

M1:

1. Short Questions

a)Linker converts source program into machine code. (True/False) [1]

Ans False

b)What is volatile memory?

Ans. Volatile memory is a type of memory that maintains its data only while the device is powered. If the power is interrupted for any reason, the data is lost.

c)The OMR stands for \_\_\_\_\_\_\_\_\_\_.

Ans Optical Mark Recognition

d)What is the Full form of EEPROM?

Ans.Electrically Erasable Programmable Read-Only Memory

e)What is the program that translates source code into object code ?

Ans. Compiler

f)What is mnemonics in assembly Language? [1]

Ans. Mnemonics are the names for the symbols used in assembly language. A phrase, symbol, or word used to define or identify a computing function is known as a mnemonic

**5 Marks**

1. What is source code, object code and executable code? Give examples of each. [5]

Ans. **Source code** refers to high level code or assembly code which is generated by human/programmer. Source code is easy to read and modify. It is written by programmer by using any High Level Language or Intermediate language which is human-readable.

**Object code** refers to low level code which is understandable by machine. Object code is generated from source code after going through compiler or other translator.

**Executable** code (also called the Binary) is the output of a linker after it processes the object code. A machine code file can be immediately executable (i.e., runnable as a program), or it might require linking with other object code files (e.g. libraries) to produce a complete executable program.

For example f1.c source code, f1.obj object code, f1.exe executable code

1. Convert (29)10 to binary and (1011101)2 to hexadecimal number system. [5]

Ans. Binary 11101 Hexadecimal: 5D

**15 Marks**

1. Distinguish between compiler error and runtime error with the help of an example. [5]

-------------------------------------------------------------------------------------------------------------------------

|  |  |
| --- | --- |
| Compile-time | Runtime |
| The compile-time errors are the errors which are produced at the compile-time, and they are detected by the compiler. | The runtime errors are the errors which are not generated by the compiler and produce an unpredictable result at the execution time. |
| In this case, the compiler prevents the code from execution if it detects an error in the program. | In this case, the compiler does not detect the error, so it cannot prevent the code from the execution. |
| It contains the syntax and semantic errors such as missing semicolon at the end of the statement. | It contains the errors such as division by zero, determining the square root of a negative number. |

**M2** Introduction to ‘C’ Language

1. Short Question

a)int a=2;printf(“%d %d %d”,++a,a++,++a);

**Ans 5 3 3**

b)The expression -11%-3 evaluates to -----

Ans -2

c) What is sizeof() in C?

Ans. It is a compile-time unary operator which can be used to compute the size of its operand.

d) What is typecasting ?

Ans. In type casting, the compiler automatically changes one data type to another one depending on what we want the program to do. For instance, in case we assign a float variable (floating point) with an integer (int) value, the compiler will ultimately convert this int value into the float value.

e) ^ is a -----------operator.

Ans Bitwise Operator.

f) Which data type will throw an error on modulus operation(%)?

Ans. float or double

**5 Marks**

1. What is meant by identifier in C? What do you mean by conditional operator? Evaluate the expression: 6\*2/ (2+1 \* 2/3 + 6) + 8 \* (8/4)  **[2+2+1]**

**Ans.** Identifiers in C language represent the names of various entities such as arrays, functions, variables, user-defined data types, labels, etc. An identifier is a type of string of alphanumeric characters that always begins with either an alphabetic or underscore character.

The conditional operator in C is kind of similar to the if-else statement as it follows the same algorithm as of if-else statement but the conditional operator takes less space and helps to write the if-else statements in the shortest way possible. It is also known as the ternary operator in C as it operates on three operands.

The conditional operator can be in the form : variable = Expression1 ? Expression2 : Expression3;

Example: i)12>=0? printf("Positive"): printf("Negative");

ii) max=12>6?12:6;

Evaluate result: 17

1. What Bitwise operator? Give examples? [5]

Ans. A bitwise operation operates on two-bit patterns of equal lengths by positionally matching their individual bits. For example, a logical AND (&) of each bit pair results in a 1 if both the first AND second bits are 1. If only one bit is a 1, the result is 0.

1. Write a program to swap the values between two variables without using third variable. [5]

Ans. #include<stdio.h>

int main()

{

int a=10, b=20;

printf("Before swap a=%d b=%d",a,b);

a=a+b;

b=a-b;

a=a-b;

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

return 0;

}

4.i)Evaluate the following expressions, indicating which ones are true or false. [5]

1. 10 = = 9 + 1
2. 10 && 8
3. 8 || 0
4. 0 && 0
5. Let X = 10 and Y=9;

X >= 8 && Y <= X

Ans. True,True,True,False,False

ii) Explain different types of type conversion in C?[8]

ans-

**In C, there are two types of type conversion:**

* Implicit Conversion.
* Explicit Conversion.

iii) Which type of conversion is NOT accepted? [2]

Explanation: **Conversion of a float to pointer type** is not allowed.

5. Explain the type cast operator with example?[3]

Cast operator: ()  
  
A type cast **provides a method for explicit conversion of the type of an object in a specific situation**.

Differentiate between Type Casting and Type Conversion with example.[7]

| **S.NO** | **TYPE CASTING** | **TYPE CONVERSION** |
| --- | --- | --- |
| 1. | In type casting, a data type is converted into another data type by a programmer using casting operator. | Whereas in type conversion, a data type is converted into another data type by a compiler. |
| 2. | Type casting can be applied to **compatible data types** as well as **incompatible data types**. | Whereas type conversion can only be applied to **compatible datatypes**. |
| 3. | In type casting, casting operator is needed in order to cast a data type to another data type. | Whereas in type conversion, there is no need for a casting operator. |
| 4. | In typing casting, the destination data type may be smaller than the source data type, when converting the data type to another data type. | Whereas in type conversion, the destination data type can’t be smaller than source data type. |
| 5. | Type casting takes place during the program design by programmer. | Whereas type conversion is done at the compile time. |
| 6. | Type casting is also called narrowing conversion because in this, the destination data type may be smaller than the source data type. | Whereas type conversion is also called widening conversion because in this, the destination data type can not be smaller than the source data type. |
| 7. | Type casting is often used in coding and competitive programming works. | Whereas type conversion is less used in coding and competitive programming as it might cause incorrect answer. |
| 8. | Type casting is more efficient and reliable. | Whereas type conversion is less efficient and less reliable. |

Differentiate between Embedded C and C

| **Parameters** | **C** | **Embedded C** |
| --- | --- | --- |
| **General** | * C is a general-purpose programming language, which can be used to design any type of desktop-based application. * It is a type of high-level language. | * Embedded C is simply an extension of C language and it is used to develop micro-controller-based applications. * It is nothing but an extension of C. |
| **Dependency** | * C language is a hardware-independent language. * C compilers are OS-dependent. | * Embedded C is a fully hardware-dependent language. * Embedded C is OS-independent. |
| **Compiler** | * For C language, the standard compilers can be used to compile and execute the program. * Popular Compiler to execute a C language program are:   + GCC (GNU Compiler collection)   + Borland turbo C,   + Intel C++ | * For Embedded C, specific compilers that are able to generate particular hardware/micro-controller-based output are used. * Popular Compiler to execute an Embedded C language program are:   + Keil compiler   + BiPOM ELECTRONIC   + Green Hill Software |
| **Usability and Applications** | * C language has a free format of program coding. * It is specifically used for desktop applications. * Optimization is normal. * It is very easy to read and modify the C language. * Bug fixing is very easy in a C language program. * It supports other various programming languages during application. * Input can be given to the program while it is running. * Applications of C Program:   + Logical programs   + System software programs | * Formatting depends upon the type of microprocessor that is used. * It is used for limited resources like RAM and ROM. * High level of optimization. * It is not easy to read and modify the Embedded C language. * Bug fixing is complicated in an Embedded C language program. * It supports only the required processor of the application and not the programming languages. * Only the pre-defined input can be given to the running program. * Applications of Embedded C Program:   + DVD   + TV   + Digital camera |

**M3: Conditional Statements and loops :**

1. a) How many times will the loop be executed?

for(i=1;i<=10;i++){

printf(“\nHello”);

i+=3;

}

**Ans. 4**

b)How many times is a do while loop guaranteed to loop?

**Ans 1**

c)What will be the output of the following code segment?

Int a=0;

Switch(a)

{

Case 0:

Default: printf(“default”);

Case 1: printf(“a=%d”,a);

}

Ans default a=0

d) What is the value of r after this code is executed?

r=2;

k=8;

if (r>3 || k>6 && r<5 ||k>10)

r=9;

else

r=6

Ans. 9

1. What is the final value of x when the code int x; for(x=0; x<10; x++) {} is executed?

Ans. 10

1. Find output of the following code:

if(printf(“C program is ”))

printf(“easy”);

else

printf(“hard”);

Ans. C program is easy

### What is fall down property?

In C Programming Language, switch case statement follows the fall down property. It means when case block is executed and break statement is not used after the block statements, then it will execute next case or default statements until break not reached or switch not finished.

### Which loop statement is executed at least once even loop test condition if false?

**do while loop** executes once even loop test condition if false. Since **do while** is an exit controlled loop and in this type of loop, loop body execute once than loop test condition checks.

**5 Marks**

**1.Difference between switch and if else statement. [5]**

a)A switch statement is significantly faster than an if-else ladder if there are many nested if-else's involved. This is due to the creation of a jump table for switch during compilation. As a result, instead of checking which case is satisfied throughout execution, it just decides which case must be completed. The number of comparisons made is lesser hence, reducing the compile time. Hence, the switch would work better while selecting from a large set of values.

b)When compared to if-else statements, it is more readable. You can also see this in the examples given above. In the if-else code, you can't clearly see the months which have 30 days; however, in switch, it's easily highlighted.

**2.Write a program which asks the user a number between 1 and 7 and prints the**

**corresponding day of the week. (1: Sunday, 2: Monday, …) using switch statement.**

#include <stdio.h>

int main()

{

int week;

printf("Enter week number (1-7): ");

scanf("%d", &week);

switch (week)

{

case 1:

printf("Monday");

break;

case 2:

printf("Tuesday");

break;

case 3:

printf("Wednesday");

break;

case 4:

printf("Thursday");

break;

case 5:

printf("Friday");

break;

case 6:

printf("Saturday");

break;

case 7:

printf("Sunday");

break;

default:

printf("\n Please enter Valid Number between 1 to 7");

}

return 0;

}

**3.Write program to generate multiplication table for first ‘n’ number, where ‘n’ is a user input.**

#include <stdio.h>

int main() {

int n;

printf("Enter an integer: ");

scanf("%d", &n);

for (int i = 1; i <= 10; ++i) {

printf("%d \* %d = %d \n", n, i, n \* i);

}

return 0;

}

4.

a)**Difference between i)break and continue ii)entry controlled loop and exit controlled loop.[8]**

**b) Write a 'C' program to check whether given number is prime or not.[7]**

**i)**a)The Break statement is used to exit from the loop constructs.

The continue statement is not used to exit from the loop constructs.

b)The break statement is usually used with the switch statement, and it can also use it within the while loop, do-while loop, or the for-loop.

The continue statement is not used with the switch statement, but it can be used within the while loop, do-while loop, or for-loop.

ii)Entry Controlled loops: In this type of loops the test condition is tested before entering the loop body. For Loop and While Loop are entry controlled loops. 2. Exit Controlled Loops: In this type of loops the test condition is tested or evaluated at the end of loop body.

b)#include <stdio.h>

int main() {

int n, i, f = 0;

printf("Enter any number: ");

scanf("%d", &n);

for (i = 2; i <n; i++) {

if (n % i == 0) {

f++;

break;

}

}

if (f == 0) {

printf("%d is a Prime number.\n", n);

} else {

printf("%d is not a Prime number.\n", n);

}

return 0;

}

**5.a) Write and explain the action of WHILE statement. Develop a program in ‘C’ language to compute the average of every third integer number lying between 1 and 100. [8]**

**b) Write a ‘C’ program to display following pattern: [7]**

A

B B

C C C

D D D D

Ans. a)A while loop in C programming repeatedly executes a target statement as long as a given condition is true.

Syntax

The syntax of a while loop in C programming language is −

while(condition) {

statement(s);

}

Here, statement(s) may be a single statement or a block of statements. The condition may be any expression, and true is any nonzero value. The loop iterates while the condition is true.

When the condition becomes false, the program control passes to the line immediately following the loop.

#include<stdio.h>

void main()

{

int a,b,s=0;

for(a=1,b=0;a<=100;a+=3)

{

s+=a;

b++;

}

printf("\n\tAverage of every third integer between 1 to 100 is %d",s/b);

}

b)

#include <stdio.h>

int main() {

int i, j, rows;

char x='A';

printf("Enter the number of rows: ");

scanf("%d", &rows);

for (i = 1; i <= rows; ++i) {

for (j = 1; j <= i; ++j) {

printf("%c ", x);

}

x++;

printf("\n");

}

return 0;

}

6. a) Write a program to generate first 15 numbers of the following sequence:

1,3,4,7,11,18,29 [8]

b) Write an interactive ‘C’ program to evaluate the series: 1 – 1/2 + 1/3 – 1/4 …±1/n, where n is

entered from keyboard.

Ans.

a)……….

int a = 1, b= 3, c;

printf("%d %d ",n1,n2);

for(int i = 3; i < 15; ++i)

{

c=a+b ;

printf("%d ",c);

a=b;

b=c;

}

b)

double sums = 0.0, t,f=1;

for (i = 1; i <= n; ++i) {

t = 1 / (f)\*pow(i, i);

f=f\*(-1);

sums += t;

}

printf(“Result: %lf”,sums);

**M4 Arrays:**

**1. Short Questions.**

a) Arrays automatically allocate space when declared(State True/False)

Ans True

b)Null character is represented by --------

Ans \0

c) C performs bound checking for array.(state true/False)

Ans. False

d) which header file should be included to use functions like strlen() and strrev()?

Ans. string.h

e) The index of an array starts from -----

Ans 0

f)printf(“%s”,”C””question””bank”); Output will be-----

Ans. Cquestionbank

g) printf(“%d %d”,sizeof(“string”),strlen(“string”)); Output will be-----

Ans. 7 6

h) Binary search works on sorted arrays. (State True/False)

Ans. True

**i**) What is use of getchar() and putchar() in C?

int getchar() = Returns the next character from the standard input (stdin). Basically read one character at a time. putchar(int) = Writes a character to the standard output (stdout).

**5 Marks**

**1.How are one dimensional and two dimensional arrays stored in computer memory? Illustrate with an example [5]**

**One Dimensional Array:**

It is a list of the variable of similar data types.

It allows random access and all the elements can be accessed with the help of their index.

The size of the array is fixed.



Two Dimensional Array:

It is a list of lists of the variable of the same data type.

It also allows random access and all the elements can be accessed with the help of their index.

It can also be seen as a collection of 1D arrays. It is also known as the Matrix.

Its dimension can be increased from 2 to 3 and 4 so on.



2.Write a ‘C’ program to find largest element of an Array [5]

Ans.

#include <stdio.h>

int main() {

int n,i,m;

int arr[100];

printf("Enter the number of elements (1 to 100): ");

scanf("%d", &n);

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

printf("Enter number%d: ", i + 1);

scanf("%d", &arr[i]);

}

m=arr[0];

for ( i = 1; i < n; ++i)

if (m < arr[i])

m = arr[i];

printf("Largest element = %.d", m);

return 0;

}

3.What is the difference between linear search and binary search? [5]

|  |  |  |
| --- | --- | --- |
| Basis of comparison | Linear search | Binary search |
| Definition | The linear search starts searching from the first element and compares each element with a searched element till the element is not found. | It finds the position of the searched element by finding the middle element of the array. |
| Sorted data | In a linear search, the elements don't need to be arranged in sorted order. | The pre-condition for the binary search is that the elements must be arranged in a sorted order. |
| Implementation | The linear search can be implemented on any linear data structure such as an array, linked list, etc. | The implementation of binary search is limited as it can be implemented only on those data structures that have two-way traversal. |
| Approach | It is based on the sequential approach. | It is based on the divide and conquer approach. |
| Size | It is preferrable for the small-sized data sets. | It is preferrable for the large-size data sets. |
| Efficiency | It is less efficient in the case of large-size data sets. | It is more efficient in the case of large-size data sets. |
| Worst-case scenario | In a linear search, the worst- case scenario for finding the element is O(n). | In a binary search, the worst-case scenario for finding the element is O(log2n). |
| Best-case scenario | In a linear search, the best-case scenario for finding the first element in the list is O(1). | In a binary search, the best-case scenario for finding the first element in the list is O(1). |

**1.a) Write a program to reverse an array of numbers.[8]**

**b)What do you mean by an Array ? How to declare and initialize 1-D array in C ? What are the advantages and disadvantages of an Array?[2+2+3]**

Ans a)

#include <stdio.h>

int main() {

int arr[30];

int n,m,i;

printf("Enter the size of the array: ");

scanf("%d", &n);

m=n;

printf("Enter an array: ");

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

scanf("%d", &arr[i]);

}

n--;

for(i=0;i<n;i++,n--)

{

t=a[i];

a[i]=a[n];

a[n]=t;

}

printf("Reversed array: ");

for ( i = 0;i<m; i++){

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

}

return 0;

}

b)An array is a collection of one or more values of the same data type stored in contiguous memory locations. The data type can be user-defined or even any other primitive data-type. Elements of an array can be accessed with the same array name by specifying the index number as the location in memory.

Array Declaration

While declaring a one-dimensional array in C, the data type can be of any type, and also, we can give any name to the array, just like naming a random variable.

Example

int arr[5];

Array Initialization

In static uninitialized arrays, all the elements initially contain garbage values, but we can explicitly initialize them at their declaration.

Example:

int nums[5] = {0, 1, 2, 3, 4};

Advantages of Arrays

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.

Arrays have low overhead.

C provides a set of built-in functions for manipulating arrays, such as sorting and searching.

C supports arrays of multiple dimensions, which can be useful for representing complex data structures like matrices.

Arrays can be easily converted to pointers, which allows for passing arrays to functions as arguments or returning arrays from functions.

Disadvantages of Arrays

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.

1. **a)Write a ‘C’ program to find out sum of diagonal elements of a matrix.[8]**

**b) What is a string? How to declare and initialize a string? Explain the use of strcpy(),strcat() function[7]**

Ans. a)

#include <stdio.h>

int main()

{

int i, j, m = 3, n = 3, a = 0, sum = 0;

int matrix[3][3]

= { { 1, 2, 3 }, { 4, 5, 6 }, { 7, 8, 9 } };

for (i = 0; i < m; ++i) {

// calculating the main diagonal sum

sum = sum + matrix[i][i];

// calculating the off diagonal sum

a = a + matrix[i][m - i - 1];

}

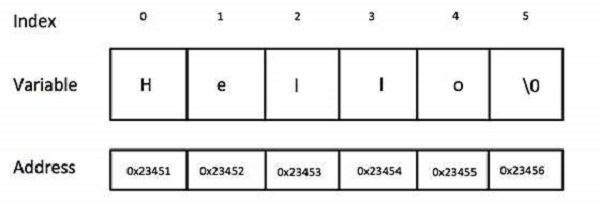
printf("\nMain diagonal elements sum is = %d\n", sum);

printf("Off-diagonal elements sum is = %d\n", a);

return 0;

}

b)

A string in C is actually a character array. As an individual character variable can store only one character, we need an array of characters to store strings. Thus, in C string is stored in an array of characters. Each character in a string occupies one location in an array. The null character ‘\0’ is put after the last character. This is done so that program can tell when the end of the string has been reached. For example, the string “Hello” is stored as follows. 

Since the string contains 5 characters. it requires a character array of size 6 to store it. the last character in a string is always a NULL('\0') character.

Initialization:

char str[] = "hello";

char str[10] = "hello";

char str[14] = { 'h','e','l','l','o','\0'};

**strcpy()** is one of the most popular String functions in c language . This function is used store a value in a string variable

Syntax :

strcpy(Str2,Str1);

Str2 is the string variable to store the value.

Str1 is the string value to be stored in string variable Str2.

We can’t use assignment operator = to assign a value to string variable. So we need to use strcpy() function to do that.

char name[20];

strcpy(name,”Amit”);

strcat() function is used to combine values of two string variables .

**Syntax**

**strcat(Str2,Str1);**

**Str2** is the string variable whose value will be combined with another string value.

**Str1** is the string value to be combined with string variable Str2.

**3.a) Write a program to delete an element from a given location of an array of integers.[7]**

**b)Write a program to check whether a given string is palindrome or not.[8]**

**Ans.a)**

**Int main(){**

**int arr[20],n=10,i;**

**printf("Enter the elements of the array : ");**

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

**scanf("%d",&arr[i]);**

**}**

**printf("Enter the position of elements to be deleted : ");**

**scanf("%d",&pos);**

**for(i=pos;i<n-1;i++) {**

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

**}**

**// Finally print new array after deletion of new element**

**printf("\nThe new array is : ");**

**for(i=0;i<n-1;i++) {**

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

**}**

**return 0;**

**}**

**b)**

**#include <stdio.h>**

**#include <string.h>**

**int main() {**

**char string1[20];**

**int i, length;**

**int flag = 0;**

**printf("Enter a string: ");**

**scanf("%s", string1);**

**length = strlen(string1);**

**for (i = 0; i < length / 2; i++) {**

**if (string1[i] != string1[length - i - 1]) {**

**flag = 1;**

**break;**

**if (flag)**

**printf("%s is not a palindrome\n", string1);**

**else**

**printf("%s is a palindrome\n", string1);**

**return 0;**

**}**

4.a) Write a program that input a string and Count no of vowels .[8]

b)Write a program that input a string and converts into uppercase.[7]

a)#include <stdio.h>

int main()

{

int c = 0, count = 0;

char s[1000];

printf("Input a string\n");

gets(s);

while (s[c] != '\0') {

if (s[c] == 'a' || s[c] == 'A' || s[c] == 'e' || s[c] == 'E' || s[c] == 'i' || s[c] == 'I' || s[c] =='o' || s[c]=='O' || s[c] == 'u' || s[c] == 'U')

count++;

c++;

}

printf("Number of vowels in the string: %d", count);

return 0;

}

b) #include <stdio.h>

int main() {

char s[100];

int i;

printf("Enter a string : ");

gets(s);

for (i = 0; s[i]!='\0'; i++) {

if(s[i] >= 'a' && s[i] <= 'z') {

s[i] = s[i] -32;

}

}

printf("String in Upper Case = %s", s);

return 0;

}

5.a) Write a C program to find the transpose of a matrix [7]

b) What are the commonly used input functions in C? Write their syntax and explain the purpose of each. [8]

Ans.

a) #include <stdio.h>

int main(){

int m, n, i, j, matrix[10][10], transpose[10][10];

printf("Enter rows and columns :");

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

printf("Enter elements of the matrix");

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

for (j = 0; j < n; j++)

scanf("%d", &matrix[i][j]);

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

for (j = 0; j < n; j++)

transpose[j][i] = matrix[i][j];

printf("Transpose of the matrix:");

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

for (j = 0; j < m; j++)

printf("%d\t", transpose[i][j]);

printf("\n");

}

return 0;

}

### b) Formatted

The formatted functions basically present or accept the available data (input) in a specific format. The standard library in C contains various functions for the input-output operations. The scanf() and printf() out of these functions help a programmer format the functions in their desired format. The program can use these functions for reading any form of data, like a real number, an integer, a character, and many more.

### Unformatted

The unformatted functions are not capable of controlling the format that is involved in writing and reading the available data. Thus, these functions constitute the most basic forms of output and input. The supply of input or the display of output isn’t allowed in the user format – thus, we call these functions *unformatted* functions for input and output.

The unformatted input-output functions further have two categories:

* The character functions
* The string functions

We use the character input functions for reading only a single character from the input device (the keyboard). On the other hand, we use the character output functions for writing just a single character on the output source (the screen). Here, the getchar(), getche(), and the getch() refer to the input functions of unformatted type, while the putchar() and putch() refer to the output functions of unformatted type.

**M5: Pointers**

**1**

a)Which of the following expression is equivalent to ++\*ptr? A) (\*ptr)++ B) ++\*(ptr) C) (ptr)\*++ D) (ptr)++\* [1]

Ans B

b) Size of a pointer is equal to the data type it points to (State True/False)

Ans False

c) Output of the following code

int a[]={1,2,9,8,6,3,5,7,8,9};

int \*p=a+1;

int \*q=a+6;

printf(“%d”,(q-p));

Ans.5

d)Two pointer variables cannot be added or multiplied. Is it True?

Ans True

e)With every use of a memory allocation function, what function should be used to release allocated memory which is no longer needed?

Ans. free()

f)How we can combine the following two statements into one using char \*p; p = (char\*) malloc(100);

Ans. char \*p=(char\*) malloc(100);

**5 Marks**

1. What will be the output of the program? Explain step by step.

#include void fun(int\*, int\*);

int main() {

int i=5, j=2;

fun(&i, &j);

printf("%d, %d", i, j);

return 0;

}

void fun(int \*i, int \*j)

{ \*i = \*i\*\*i;

\*j = \*j\*\*j;

}

Ans 25 4

2.Write a note on pointers and its uses. What do you mean by referencing and de-referencing of a pointer variable ?

Ans.

Pointers are used to store and manage the addresses of dynamically allocated blocks of memory. Such blocks are used to store data objects or arrays of objects. Most structured and object-oriented languages provide an area of memory, called the heap or free store, from which objects are dynamically allocated.

Making a pointer variable to point other variables by providing address of that variable to the pointer is known as referencing of pointer.

It is also known as initialization of pointers. For proper use of pointer, pointer variables must point to some valid address and it is important to note that without referencing, pointer variables are meaningless.

General syntax for referencing of pointer is:

pointer\_variable = &normal\_variable;

The operator \* (star) used in front of the name of the pointer variable is known as pointer or dereferencing or indirection operator. After valid referencing of pointer variable, \* pointer\_variable gives the value of the variable pointed by pointer variable and this is known as dereferencing of pointer. For the sake of simplicity, \*pointer\_variable after referencing instructs compilers that go to the memory address stored by pointer\_variable and get value from that memory address.

Examples of Dereferencing of Pointer:

int a=5;

int \*ptr;

ptr = &a;

printf(“a = %d”, \*ptr);

1. a)Find the output of the following:

char p[] = "%d\n";

p[1] = 'c';

printf(p, 65);

b) #include <stdio.h>

void main()

{

int k = 5;

int \*p = &k;

int \*\*m = &p;

printf("%d%d%d\n", k, \*p, \*\*m);

}

# Ans a) A b) 5 5 5

# 4.a)What are the advantages and disadvantages of using pointers in C?[5]

b)What is dynamic memory allocation? Mention four functions used for dynamic memory manipulation. [5]

c) Explain the use of malloc() and calloc() in dynamic memory allocation. [5]

Ans.

a)Advantages

* Pointers provide direct access to memory
* Pointers provide a way to return more than one value to the functions
* Reduces the storage space and complexity of the program
* Reduces the execution time of the program
* Provides an alternate way to access array elements
* Pointers can be used to pass information back and forth between the calling function and called function.
* Pointers allows us to perform dynamic memory allocation and deallocation.
* Pointers helps us to build complex data structures like linked list, stack, queues, trees, graphs etc.
* Pointers allows us to resize the dynamically allocated memory block.
* Addresses of objects can be extracted using pointers

Disadvantages of pointers in c:

* Uninitialized pointers might cause segmentation fault.
* Dynamically allocated block needs to be freed explicitly.  Otherwise, it would lead to memory leak.

b)

Dynamic memory allocation is the process of assigning the memory space during the execution time or the run time. Reasons and Advantage of allocating memory dynamically: When we do not know how much amount of memory would be needed for the program beforehand

Malloc(),calloc(),free(),realloc()

c) **malloc()** allocates a memory block of given size (in bytes) and returns a pointer to the beginning of the block. malloc() doesn’t initialize the allocated memory. If you try to read from the allocated memory without first initializing it, then you will invoke [undefined behavior](https://www.geeksforgeeks.org/undefined-behavior-c-cpp/), which usually means the values you read will be garbage values.

**calloc()** allocates the memory and also initializes every byte in the allocated memory to 0. If you try to read the value of the allocated memory without initializing it, you’ll get 0 as it has already been initialized to 0 by calloc().

5.a)**Output of following program?[8]**

#include <stdio.h>

int main()

{

int \*ptr;

int x;

ptr = &x;

\*ptr = 0;

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

printf(" \*ptr = %d\n", \*ptr);

\*ptr += 5;

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

printf(" \*ptr = %d\n", \*ptr);

(\*ptr)++;

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

printf(" \*ptr = %d\n", \*ptr);

return 0;

}

**b) Difference between array and pointer.[7]**

Ans a)0 0 5 5 6 6

## b) Difference between Array and Pointer

|  |  |  |
| --- | --- | --- |
| S.No. | Array | Pointer |
| 1. | It stores the values of a homogeneous data type. | It stores the address of variables. |
| 2. | An array is defined as a collection of similar datatypes. | Pointer is a variable which stores address of another variable. |
| 3. | The number of variables that can be stored is decided by the size of the array. | Pointer can store the address of only a single variable. |
| 4. | The initialization of arrays can be done while defining them. | Pointers cannot be initialized while defining them. |
| 5. | The nature of arrays is static. | The nature of pointers is dynamic. |
| 6. | Arrays cannot be resized according to user's requirements. | Pointers can be resized at any point of time. |
| 7. | The allocation of array is done at compile time. | The allocation of pointer is done at run time. |

5.a)What is a function pointer? Explain with example [8]

b) What is the difference between array of pointer and pointer to an array . [7]

Ans.

a) The pointer variable that holds a function’s address is called a function pointer. The basic advantage of a function pointer is that one function can be passed as a parameter to another function. Function pointer calls are faster than normal functions.

We already discussed that, in C Programming Language, we can create a pointer of any data type such as int, char, float, etc. In C Programming Language, we can also create a pointer pointing to a function. In that case, the code of that function always resides in the memory. That means the function now has some address. We can get the address of memory by using the function pointer.

#include <stdio.h>

int Fun()

{

int x = 10;

}

int main()

{

printf ("Address of Main Function : %p", main);

printf ("\nAddress of Fun Function : %p", Fun);

return 0;

}

b) Pointer to an array: Pointer to an array is also known as array pointer. We are using the pointer to access the components of the array.

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

int \*ptr = a;

We have a pointer ptr that focuses to the 0th component of the array. We can likewise declare a pointer that can point to whole array rather than just a single component of the array. Syntax:

data type (\*var name)[size of array];

Declaration of the pointer to an array:

// pointer to an array of five numbers

int (\* ptr)[5] = NULL;

The above declaration is 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.

Array of pointers: “Array of pointers” is an array of the pointer variables. It is also known as pointer arrays. Syntax:

int \*var\_name[array\_size];

Declaration of an array of pointers:

int \*ptr[3];

We can make separate pointer variables which can point to the different values or we can make one integer array of pointers that can point to all the values.

**M6:Function**

**1 Short question**

a)A function can be defined within another function. State True/False

Ans False

b)main() is a library function defined in stdio.h file. State True/False

Ans False

c)By default any user defined function returns -----type of data.

Ans int

d)----- is the default storage class of any declared variable.

Ans auto

e) The value obtained in the function is given back to main by using \_\_\_\_\_\_\_\_ keyword.

Ans return

f)Every C Program should contain which function?

Ans main()

g)The default parameter passing mechanism is-----

Ans Call by Value

h)What is full form of LIFO?

Ans. Last In First Out

i)When an array is passed to a function as a function argument, it passes the ----- address of the array.

Ans base

**Marks 5**

1.What is the purpose of using functions in ‘C’ programming? Differentiate declaration and definition of a function. [5]

Ans. In programming, functions allow us to repeat a sequence of code without having to write the code over again. This allows us to simplify a complex action by using only one command.

The declaration establishes the names and characteristics of a function but does not allocate storage for it, while the definition specifies the body for a function, associates an identifier with the function, and allocates storage for it.

2.How will you pass parameters to a function? Write a function that will compute y=x^n where x and n are positive integer numbers. [5]

1. Pass by Value. Pass by Value, means that a copy of the data is made and stored by way of the name of the parameter.
2. Pass by Reference. A reference parameter "refers" to the original data in the calling function.

#include <stdio.h>

int power(int x, int n)

{

int i,f=1;

for(i=1;i<=n;i++)

f=f\*x;

return f;

}

int main()

{

    int x = 2;

    int n = 3;

     printf("%d\n", power(x, n));

}

3.What do you mean by call by value & call by reference? Explain with examples. [5]

Call By Value in C: 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 caller. In other words, in this parameter passing method, values of actual parameters are copied to function's formal parameters, and the parameters are stored in different memory locations. So any changes made inside functions are not reflected in actual parameters of the caller.

#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;

}

Call by reference in C Call by reference method copies the address of an argument into the formal parameter. In this method, the address is used to access the actual argument used in the function call. It means that changes made in the parameter alter the passing argument. In this method, the memory allocation is the same as the actual parameters. All the operations in the function are performed on the value stored at the address of the actual parameter, and the modified value will be stored at the same address. Means, both the actual and formal parameters refer to same locations, so any changes made inside the function are actually reflected in actual parameters of caller.

#include<stdio.h>

void change(int \*num) {

    printf("Before adding value inside function num=%d \n",\*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 reference in function

    printf("After function call x=%d \n", x);

return 0;

}

**4.Explain the storage classes available in C. What do you mean by scope of the variable? [4+1]**

There are four different types of storage classes that we use in the C language:

* Automatic Storage Class
* External Storage Class
* Static Storage Class
* Register Storage Class

Use of Storage Class in C

A variable given in a C program will have two of the properties: storage class and type. Here, type refers to any given variable’s data type, while the storage class determines that very variable’s lifetime, visibility, and also its scope.

Summary of Storage Classes in C

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Class | Name of Class | Place of Storage | Scope | Default Value | Lifetime |
| auto | Automatic | RAM | Local | Garbage Value | Within a function |
| extern | External | RAM | Global | Zero | Till the main program ends. One can declare it anywhere in a program. |
| static | Static | RAM | Local | Zero | Till the main program ends. It retains the available value between various function calls. |
| register | Register | Register | Local | Garbage Value | Within the function |

In simple terms, scope of a variable is its lifetime in the program. This means that the scope of a variable is the block of code in the entire program where the variable is declared, used, and can be modified.

5.a)State and explain various types of standard function with example. [5]

b)State and explain different phases used in user defined function. [5]

c)Explain function with return and function with arguments with example.[5]

Ans.

a) abs stdlib.h int abs(int n); Calculates the absolute value of an integer argument n.

pow math.h double pow(double x, double y); Calculates the value x to the power y.

printf stdio.h int printf(const char \*format-string, arg-list); Formats and prints characters and values to stdout.

scanf stdio.h int scanf(const char \*format-string, arg-list); Reads data from stdin into locations given by arg-list.

b) Different Types of User-defined Functions in C

There are four types of user-defined functions divided on the basis of arguments they accept and the value they return:

Function with no arguments and no return value

Function with no arguments and a return value

Function with arguments and no return value

Function with arguments and with return value

c) #include <stdio.h>

int sum(int x, int y) { return x + y; }

int main()

{

int x, y;

printf("Enter x and y\n");

scanf("%d %d", &x, &y);

printf("Sum of %d and %d is: %d", x, y, sum(x, y));

return 0;

}

**6.a)What is function? What do you mean by the following: Function declaration, Function calling, Function definition.[8]**

**b)When is it necessary to declare prototype of a function? What is Actual parameter and formal parameter [3+4]**

Ans. a)Functions in C are the basic building blocks of a C program. A function is a set of statements enclosed within curly brackets ({}) that take inputs, do the computation, and provide the resultant output. You can call a function multiple times, thereby allowing reusability and modularity in C programming

A function declaration tells the compiler about a function's name, return type, and parameters. A **function definition** provides the actual body of the function.

A **function declaration** tells the compiler about a function name and how to call the function. The actual body of the function can be defined separately.

A function declaration has the following parts −

return\_type function\_name( parameter list );

For the above defined function max(), the function declaration is as follows −

int max(int num1, int num2);

Parameter names are not important in function declaration only their type is required, so the following is also a valid declaration −

int max(int, int);

**Calling a Function**

While creating a C function, you give a definition of what the function has to do. To use a function, you will have to call that function to perform the defined task.

#include <stdio.h>

/\* function declaration \*/

int max(int num1, int num2);

int main () {

/\* local variable definition \*/

int a = 100;

int b = 200;

int ret;

/\* calling a function to get max value \*/

ret = max(a, b);

printf( "Max value is : %d\n", ret );

return 0;

}

/\* function definition \*/

int max(int num1, int num2) {

/\* local variable declaration \*/

int result;

if (num1 > num2)

result = num1;

else

result = num2;

return result;

}

b)

The function prototypes are used to tell the compiler about the number of arguments and about the required datatypes of a function parameter, it also tells about the return type of the function. By this information, the compiler cross-checks the function signatures before calling it. If the function prototypes are not mentioned, then the program may be compiled with some warnings, and sometimes generate some strange output.

If some function is called somewhere, but its body is not defined yet, that is defined after the current line, then it may generate problems. The compiler does not find what is the function and what is its signature. In that case, we need to function prototypes. If the function is defined before then we do not need prototypes.

7. a)What is the purpose of main function? What are the difference between void type and other data type used in function definition?[5]

b)What do you mean by recursion? Merits and Demerits of recursion.[5]

c) Write a program to calculate factorial of a number using recursion.[5]

Ans.

a) The main function serves as the starting point for program execution. It usually controls program execution by directing the calls to other functions in the program. Several restrictions apply to the main function that don't apply to any other C functions.

Void functions are created and used just like value-returning functions except they do not return a value after the function executes. In lieu of a data type, void functions use the keyword "void." A void function performs a task, and then control returns back to the caller--but, it does not return a value. You may or may not use the return statement, as there is no return value. Even without the return statement, control will return to the caller automatically at the end of the function. A good utilization of a void function would be to print a header/footer to a screen or file.

for example : int main()

Every C program must write this statement. This statement (main) specifies the starting point of the C program execution. Here, main is a user-defined method which tells the compiler that this is the starting point of the program execution. Here, int is a data type of a value that is going to return to the Operating System after completing the main method execution. If we don't want to return any value, we can use it as void.

b) RECURSION - A method of programming whereby a function directly or indirectly calls itself. Recursion is often presented as an alternative to iteration. Recursion is simply the use of a function which, as part of it's own execution code, invokes itself.Advantages of recursion

1. The code may be easier to write.

2. To solve such problems which are naturally recursive such as tower of Hanoi.

3. Reduce unnecessary calling of function.

4. Extremely useful when applying the same solution.

5. Recursion reduce the length of code.

6. It is very useful in solving the data structure problem.

7. Stacks evolutions and infix, prefix, postfix evaluations etc.

Disadvantages of recursion

1. 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.

3. Hard to analyze or understand the code.

4. It is not more efficient in terms of space and time complexity.

5. The computer may run out of memory if the recursive calls are not properly checked.

c) #include<stdio.h>

long int fact(int n);

int main() {

int n;

printf("Enter a positive integer: ");

scanf("%d",&n);

printf("Factorial of %d = %ld", n, fact(n));

return 0;

}

long int fact(int n) {

if (n>=1)

return n\*fact(n-1);

else

return 1;

}

8.a) How does passing an array as an argument to a function differ from call by value? [7]

b) Write a complete program to evaluate the given series using recursive function sum( ). Here n is user dependent. 1 + 2 + 3 +…+ n [8]

Ans.

* 1. Call by value creates a copy of the variables passed as argument whereas while passing an array, we pass the base address ie the name of the array itself,,,so whatever is done to the array inside the function is reflected in the actual array.

Example:

#include <stdio.h>

void printArray(int arr[], int size)

{

printf("Array Elements: ");

for (int i = 0; i < size; i++) {

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

}

}

int main()

{

int arr[8] = { 12, 4, 5, 3, 7, 8, 11, 45 };

int size = sizeof(arr) / sizeof(arr[0]);

printArray(arr, size);

return 0;

}

b) #include <stdio.h>

int addNumbers(int n);

int main() {

int num;

printf("Enter a positive integer: ");

scanf("%d", &num);

printf("Sum = %d", addNumbers(num));

return 0;

}

int addNumbers(int n) {

if (n != 0)

return n + addNumbers(n - 1);

else

return n;

}

**M7 File Processing**

**1 Short questions**

a)The fopen() function reurns -------- when the file is not opened.

Ans NULL

b)Function used to close a file is ------

Ans fclose()

c)What is the meaning of opening a file in w mode?

Ans. The w mode opens the file for only writing. You can NOT read in this mode. The file pointer is placed at the start of the file. If the file exists, its content is truncated. If the file does not exist, it is created.

d) what is text file and binary file?

Ans. There are mainly two types of data files — text file and binary file. A text file consists of human readable characters, which can be opened by any text editor. On the other hand, binary files are made up of non-human readable characters and symbols, which require specific programs to access its contents.

e)FILE is of type \_\_\_\_\_\_

Ans. struct type

f)If there is any error while opening a file, fopen will return -------

Ans. NULL

**5 Marks**

**1.Discuss the use of fscanf() And fprintf() functions. Give some examples.**

Ans. Writing File : fprintf() function

The function fprintf () is utilized for writing a group of characters to a file and transmitting formatted output to a stream.

Example:

#include <stdio.h>

main(){

FILE \*fp;

fp = fopen("file.txt", "w");//opening file

fprintf(fp, "Hello file by fprintf...\n");//writing data into file

fclose(fp);//closing file

}

Reading File : fscanf() function

The f scanf() function function is designed to retrieve a sequence of characters from a file. It extracts a single word from the file and signals the end of the file by returning EOF.

#include<stdio.h>

main(){

FILE \*fp;

char buff[255];//creating char array to store data of file

fp = fopen("file.txt", "r");

while(fscanf(fp, "%s", buff)!=EOF){

printf("%s ", buff );

}

fclose(fp);

}

2.**What are the uses of following file handling function: fopen(), fclose(), fseek(), ftell(), rewind()**

fopen () fopen () function creates a new file or opens an existing file.

fclose () fclose () function closes an opened file.

ftell() ftell function is used to get current position of the file pointer.

rewind() rewind function is used to move file pointer position to the beginning of the file.

fseek () fseek () function moves file pointer position to given location.

1. **How to create a file in C? Why should a user close the file?**

Whenever you want to work with a file, the first step is to create a file. A file is nothing but space in a memory where data is stored.

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

FILE \*fp;

fp = fopen ("file\_name", "mode");

In the above syntax, the file is a data structure which is defined in the standard library.

fopen() is a standard function which is used to open a file.

If the file is not present on the system, then it is created and then opened.

If a file is already present on the system, then it is directly opened using this function.

fp is a file pointer which points to the type file. Mode may be r, w, a, r+, w+, a+

**Closing a file** after use is important because it frees up system resources that are being used by the file. When a file is open, the operating system allocates memory and other resources to the file, which can potentially impact the performance of the system if too many files are open at the same time

**4.a) What is a file in C ? Discuss various modes in which a file can be opened. Also discuss types of files.[8]**

**b)Write a program that copies contents of a text file into another file.[7]**

Ans. a) File is a collection of data that stored on secondary memory like hard disk of a computer.

C programming language supports two types of files and they are as follows...

Text Files (or) ASCII Files and Binary Files

Text File (or) ASCII File - The file that contains ASCII codes of data like digits, alphabets and symbols is called text file (or) ASCII file.

Binary File - The file that contains data in the form of bytes (0's and 1's) is called as binary file. Generally, the binary files are compiled version of text files.

In C programming language, there different modes are available to open a file and they are shown in the following table.

S. No. Mode Description

1 r Opens a text file in reading mode.

2 w Opens a text file in wirting mode.

3 a Opens a text file in append mode.

4 r+ Opens a text file in both reading and writing mode.

5 w+ Opens a text file in both reading and writing mode. It set the cursor position to the beginning of the file if it exists.

6 a+ Opens a text file in both reading and writing mode. The reading operation is performed from beginning and writing operation is performed at the end of the file.

b)

#include <stdio.h>

int main()

{

FILE \*fptr1, \*fptr2;

char filename[100], c;

printf("Enter the filename to open for reading \n");

scanf("%s", filename);

// Open one file for reading

fptr1 = fopen(filename, "r");

if (fptr1 == NULL)

{

printf("Cannot open file %s \n", filename);

exit(0);

}

printf("Enter the filename to open for writing \n");

scanf("%s", filename);

// Open another file for writing

fptr2 = fopen(filename, "w");

if (fptr2 == NULL)

{

printf("Cannot open file %s \n", filename);

exit(0);

}

// Read contents from file

c = fgetc(fptr1);

while (c != EOF)

{

fputc(c, fptr2);

c = fgetc(fptr1);

}

printf("\nContents copied to %s", filename);

fclose(fptr1);

fclose(fptr2);

return 0;

}

5.a) Short note on fseek() and rewind(). [8]

b)Write a program which creates a file named TEXT, enter 10 integers in the file and then read them from that file.[7]

Ans. a)C programming language provides various pre-defined methods to set the cursor position in files. The following are the methods available in c, to position cursor in a file.

rewind() and fseek()

rewind( \*file\_pointer ) - This function is used reset the cursor position to the beginning of the file.

fseek() in C language is used to move file pointer to a specific position. Offset and stream are the destination of pointer, given in the function parameters. If successful, it returns zero, else non-zero value is returned.

int fseek(FILE \*stream, long int offset, int whence)

Here are the parameters used in fseek(),

* stream − This is the pointer to identify the stream.
* offset − This is the number of bytes from the position.
* whence − This is the position from where offset is added.

whence is specified by one of the following constants.

* SEEK\_END − End of file.
* SEEK\_SET − Starting of file.
* SEEK\_CUR − Current position of file pointer.

b)

# include <stdio.h>

void main()

{

int num;

char reader;

FILE \*fp;

fp = fopen("TEXT", "w");

if(fp == NULL)

{

printf("Can not write file.");

return;

}

for(int i = 1; i<=10; i++)

{

printf("%d ",i);

scanf("%d", &num);

fprintf(fp, "%d ", num);

}

fclose(fp);

// To read our content from the file.

fp = fopen("TEXT", "r");

printf("\n Integers: \n\n");

while(fscanf(fp, "%d ", &num)!=EOF)

printf("%d\t", num);

fclose(fp);

}