1. **Ways to pass arguments to functions: pass by value and pass by reference**.

When a function is called, the arguments in a function can be **passed by value** or **passed by reference**.

*pass by value* means you are making a copy in memory of the actual parameter's value that is passed in, a copy of the contents of the actual parameter. Use pass by value when when you are only "using" the parameter for some computation, not changing it for the client program.

**In *pass by reference*** (also called pass by address), a copy of the address of the actual parameter is stored. Use pass by reference when you are changing the parameter passed in by the client program.

1. **Strings. Functions strcpy and strncpy.**

**strcpy**( ) **function** copies whole content of one string into another string. Whereas, **strncpy**( ) **function** copies portion of contents of one string into another string. If destination string length is less than source string, entire/specified source string value won't be copied into destination string in both cases.

1. **A sequence of positive integers is entered. 0 is the end of the sequence. For each element of the sequence, determine and display the number that will be obtained after writing the digits of the original numbers in reverse order.**

**Write a program with function**

#include <stdio.h>

/\* Iterative function to reverse digits of num\*/

int reversDigits(int num)

{

int rev\_num = 0;

while (num > 0) {

rev\_num = rev\_num \* 10 + num % 10;

num = num / 10;

}

return rev\_num;

}

/\*Driver program to test reversDigits\*/

int main()

{

int num = 4562;

printf("Reverse of no. is %d", reversDigits(num));

getchar();

return 0;

}

1. **Write a program that determines whether the given sequence contains at least one Armstrong number or not.**
2. **C program to print contents of file.**

#include <stdio.h>

#include <stdlib.h> // For exit()

int main()

{

FILE \*fptr;

char filename[100], c;

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

scanf("%s", filename);

// Open file

fptr = fopen(filename, "r");

if (fptr == NULL)

{

printf("Cannot open file \n");

exit(0);

}

// Read contents from file

c = fgetc(fptr);

while (c != EOF)

{

printf ("%c", c);

c = fgetc(fptr);

}

fclose(fptr);

return 0;

}

1. **Pointers. Addresses, &operator, %p format.**

A **pointer** is a variable that stores the address of another variable. The data type of pointer and the variable must match, an int pointer can hold the address of int variable, similarly a pointer declared with float data type can hold the address of a float variable.

The bitwise AND operator (**&**) compares each bit of the first operand to the corresponding bit of the second operand. If both bits are 1, the corresponding result bit is set to 1. Otherwise, the corresponding result bit is set to 0.

%p. This is used to print the pointer type data.

1. **The Standart Files**

When a C program begins its execution, it has to access three files.  
  
Ø  Standard input file  
Ø  Standard output file  
Ø  Standard error file  
  
Standard input file:    The first file is the standard input file from which the input is received, usually it is a keyboard. Input can also be redirected from a disk file. As long                                         as the program reads the input from the beginning to its end, it need not be concerned about where the characters are coming from.  
  
standard output file:  The second file is the standard output file to which the output is sent; usually it is the visual display unit (i.e. screen). Output can also be redirected                                         to store into the disk file. As long as the program reads the input from the beginning to its end, it need not be concerned about where the characters                                         are coming from.  
  
standard error file:    The third file is the standard error file. The purpose of this file is to keep error messages separate from other program output (i.e. if the standard                                             output is displayed in the screen) so that the error messages are stored in the disk file and vice-versa.

The stdin, stdout, and stderr are file type pointers defined in the header file stdio.h.

The standard input device (keyboard), standard output and error device (VDU) are called as the stdin, stdout, and stderr respectively. These file type pointers are treated as files in C.

File related operations can be performed on these files just as other ordinary files.

For example  
  
    @    The statement x=fgetc(stdin); accepts a single character as input from the standard input device (i.e. keyboard) and assigns it to the variable x.  
  
    @    The statement fputc(x, stdout); displays the character as output on the standard output device (i.e screen) which is assigned to the variable x.  
  
    @    The statement fputc(x, stderr); also displays the character as output on the screen which is assigned to x. Even though the VDU is the output and error device,                     there are differences these two files.

1. **Write a C program with function to determine whether a given number is prime or not.**

#include <stdio.h>

#include <conio.h>

void main()

{

int num,res=0;

clrscr();

printf("\nENTER A NUMBER: ");

scanf("%d",&num);

res=prime(num);

if(res==0)

printf("\n%d IS A PRIME NUMBER",num);

else

printf("\n%d IS NOT A PRIME NUMBER",num);

getch();

}

int prime(int n)

{

int i;

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

{

if(n%i!=0)

continue;

else

return 1;

}

return 0;

}

1. **A program for reading information from an input file and displaying it on the screen as on a standart output device.**

#include <iostream>

using namespace std;

int main()

{

char sample[] = "GeeksforGeeks";

cout << sample << " - A computer science portal for geeks";

return 0;

}

1. **Enter a sequence of integers. 0 is the end of the sequence. Determine if the sequence contains at least one perfect number. A perfect number is equal to the sum of all its divisors that do not exceed this number. For example, 6 = 1+2+3 or 28 = 1 +2 + 4 +7 +14.**

**Write a program with a function.**

1. **Recursion. Recursion vs. Iteration.**

The process in which a function calls itself directly or indirectly is called recursion and the corresponding function is called as recursive function. Using recursive algorithm, certain problems can be solved quite easily. Examples of such problems are [Towers of Hanoi (TOH)](http://quiz.geeksforgeeks.org/c-program-for-tower-of-hanoi/), [Inorder/Preorder/Postorder Tree Traversals](https://www.geeksforgeeks.org/tree-traversals-inorder-preorder-and-postorder/), [DFS of Graph](https://www.geeksforgeeks.org/depth-first-traversal-for-a-graph/), etc.

Difference:

**Usage:** Usage of either of these techniques is a trade-off between time complexity and size of code. If time complexity is the point of focus, and number of recursive calls would be large, it is better to use iteration. However, if time complexity is not an issue and shortness of code is, recursion would be the way to go.

**Overhead:** Recursion has a large amount of Overhead as compared to Iteration.

**Infinite Repetition:** Infinite Repetition in recursion can lead to CPU crash but in iteration, it will stop when memory is exhausted.

**Time Complexity:** Finding the Time complexity of Recursion is more difficult than that of Iteration.

1. **Structures. Arrays of structures.**

**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**.

1. To declare an **array** of **structure**, first the **structure** must be defined and then an **array** variable of that type should be defined.
2. For Example − **struct** book b[10]; //10 elements in an **array** of **structures** of type 'book'
3. **A sequence of N positive integers is entered. Determine if the sequence contains at least one prime number. Write a program with function.**
4. **Write a program which to implement a function for sine calculations. The function argument is the variable x, and the row boundary is defined as a constant**

#include <iostream>

Using namespace std;

double sinfunction(int N, int x)

{

int sum=0;

int i,q=x;

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

{

sum+=q;

q=q\*(-1)\*x\*x\(2\*i)\(2\*i+1);

}

return sum;

}

int main()

{

int N, x;

cout<<"Enter the value of N:";

cin>>N;

cout<<"Enter the value of x:";

cin>>x;

sinfunction(N;x)

return 0;

}

1. **Write a program for finding the sum of the elements of a matrix above the main diagonal.burda below diagonalda var kodda onu yazmassiz bu sual gelse**

#include<stdio.h>

int main()

{

int i,j,m,n,d1=0,d2=0,a[5][5];

printf("How many rows and columns:");

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

printf("Enter matrix elements:\n");

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

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

{

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

if(j>i)

d1+=a[i][j];

else

if(i>j)

d2+=a[i][j];

}

printf(\n"Sum of elements above the diagonal=%d\n",d1);

printf("Sum of elements below the diagonal=%d",d2);

return 0;

}

1. **Relationship Between Pointers and Arrays**

An **array** is represented by a variable that is associated **with the** address of its first storage location. A **pointer** is also the address of a storage location **with a** defined type, so D permits the use of the **array** [ ] index notation with both **pointer** variables and **array** variables. **Pointers** are not **arrays**, however, if you assign an **array to** a **pointer**, the **pointer** virtually becomes the **array**. The **pointer** will contain the address **of** the first element **of** the **array** and it can read and write each element **of** the **array** by using **pointer** arithmetics and/or **array**-style indexing.

1. **Searching Algorithms**

There are many different types of searching algorithms. Two of them are **linear search** and **binary search**.

Linear Search

Linear search is a very basic and simple search algorithm. In Linear search, we search an element or value in a given array by traversing the array from the starting, till the desired element or value is found.

**Binary Search**

Binary Search is used with sorted array or list. In binary search, we follow the following steps:

Binary Search is useful when there are large number of elements in an array and they are sorted.

So a necessary condition for Binary search to work is that the list/array should be sorted.

1. **Write a program in C to find the factorial of a number using recursion.**

#include<stdio.h>

long int multiplyNumbers(int n);

int main() {

int n;

printf("Enter a positive integer: ");

scanf("%d",&n);

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

return 0;

}

long int multiplyNumbers(int n) {

if (n>=1)

return n\*multiplyNumbers(n-1);

else

return 1;

}

Initially, multiplyNumbers() is called from main() with 6 passed as an argument.

Then, 5 is passed to multiplyNumbers() from the same function (recursive call). In each recursive call, the value of argument n is decreased by 1.

When the value of n is less than 1, there is no recursive call and the factorial is returned ultimately to the main() function.

1. **Write a program in C to count the total number of words in a string.**

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#define str\_size 100 //Declare the maximum size of the string

void main()

{

char str[str\_size];

int i, wrd;

printf("\n\nCount the total number of words in a string :\n");

printf("------------------------------------------------------\n");

printf("Input the string : ");

fgets(str, sizeof str, stdin);

i = 0;

wrd = 1;

/\* loop till end of string \*/

while(str[i]!='\0')

{

/\* check whether the current character is white space or new line or tab character\*/

if(str[i]==' ' || str[i]=='\n' || str[i]=='\t')

{

wrd++;

}

i++;

}

printf("Total number of words in the string is : %d\n", wrd-1);

}

1. **A sequence of positive integers is entered, 0 is end of the sequence. For each element of the sequence, determine and display the number that will be obtained after writing the digits of the original number in reverse order. Write a program using function.**
2. **Examples of recursion. Greatest common divisor. Towers of Hanoi.**
3. **String Library Functions**
4. **A sequence of N positive integers is entered. Count the number of palindrome numbers. Palindrome numbers are symmetrical about their midpoint, for example, 12021 or 454.**
5. **A program with functions for working with strings and symbols.**
6. **Sorting Arrays. Selection sort algorithm and program**
7. **Functions. Function Definitions with Multiple Parameters.**
8. **C – Memory Management. Calloc(), malloc(), realloc().**
9. **Write a program in C to find the factorial of a number using recursion**
10. **Write a program in C to count the total number of words in a string**
11. **A sequence of positive integers is entered, 0 is end of the sequence. For each element of the sequence, determine and display the number that will be obtained after writing the digits of the original number in the reverse order. Write a program using function.**
12. **C/C++ Bitwise Operators**
13. **Strings. Functions strcpy and strncpy**
14. **A sequence of N positive integers is entered. Determine if the sequence contains at least one prime number. Write a program with a function**
15. **Write a program in which to implement a function for sine calculations. The function argument is the variable x, and the row boundary is defined as a constant.**
16. **Write a program for finding the sum of the elements of a matrix above the main diagonal.**
17. **Recursion. Example of using recursion: Fibonacci Series.**
18. **Binary search in C/C++.**
19. **Write a program for implementation of insertion sort.**
20. **A sequence of N positive integers is entered. Check if the sequence contains at least one pair of adjacent friendly numbers. Two distinct natural numbers are friendly if the sum of all divisors of the first number (except for the number itself) is equal to the second number. For example, 220 and 284, 1184 and 1210, 2620 and 2924, 5020 and 5564. Write a program with a function**
21. **Write a program in which to implement a function for cosine calculations. The function argument is the variable x, and the row boundary is defined as a constant**
22. **Pointers and arrays.**
23. **The Standart Files**
24. **Shaker (Cocktail) sort algorithm and program.**
25. **Write a program to transform the original matrix so that the zero elements of each row is replaced by the arithmetic mean of the element in this row.**
26. **Three real numbers x,y,z are given. Find the value of min(max(x,y), max(y,z), x+y+z). Write a program with function.**
27. **Function Prototypes. Argument Promotion Rules**
28. **Structures. Defining a Structure**
29. **Write a program in C to check a number is a prime number or not using recursion.**
30. **Three integers a,b,c are given. Find their mean. Mean number equals to a+b+c – min(a,b,c)- max(a,b,c). Write a program with function.**
31. **Write a program with a function to calculate a number from Fibonacci sequences. The function argument is the ordinal number of the number in the sequence. Write a program with function.**
32. **The call by reference method of passing arguments to a function**
33. **Searching Arrays with Linear Search**
34. **Write a C program of insertion sort.**
35. **The matrix A(n,m) is given. Write a program to swap its maximum and minimum elements.**
36. **Search in arrays. Binary search program**
37. **Examples of recursion. Greatest common divisor. Towers of Hanoi.**
38. **Dynamic memory allocation with new and delete operator.**
39. **Write a program in C to find the factorial of a number using recursion.**
40. **Write a program in C to count the total number of words in a string.**
41. **A sequence of positive integeres is entered, 0 is end of the sequence. For each element of the sequence, determine and display the number that will be obtained after writing the digits of the original number in the reverse order. Write a program using function.**
42. **Functions. References and reference parameters.**
43. **Functions for working with files**
44. **A sequence of N positive integers is entered. Determine if the sequence contains at least one prime number. Write a program with a function**
45. **Write a program in which to implement a function for sine calculations. The function argument is the variable x, and the row boundary is defined as a constant**
46. **Write a program for finding the sum of the elements of a matrix above the main diagonal.**
47. **Functions. References and reference parameters.**
48. **Sorting Arrays with Insertion Sort.**
49. **A sequence of positive integers is entered, 0 is the end of the sequence. For each element of the sequence, determine and display the number that will be obtained after writing the digits of the original numbers in reverse order. Write a program with a function**
50. **Write a program that determines whether the given sequence contains at least one Armstrong number or not.**
51. **C program to print contents of file.**
52. **Pointers. Pointer arithmetic. Pointer variable declarations and initialization**
53. **Searching Algorithms**
54. **Write a C program with function to determine whether a given number is prime or not**
55. **A program for reading information from an input file and displaying it on the screen as on a standard output device.**
56. **Enter a sequence of integers, 0 is the end of the sequence. Determine if the sequence contains at least one perfect number. A perfect number is equal to the sum of all its divisors that do not exceed this number. For example, 6 = 1 + 2 + 3 or 28 = 1 + 2 + 4 + 7 +14. Wr**ite a program with function.