**OPERATOR AND CONTROL STRUCTURES**

1. **Write a program to read in two integers and perform the following operations on them: addition, subtraction, multiplication, division, and modulo.**

**PROGRAM:**

#include<iostream>

using namespace std;

int main()

{

int n1,n2;

cout<<"enter number 1:";

cin>>n1;

cout<<"enter number 2:";

cin>>n2;

int sum=n1+n2;

cout<<"sum:"<<sum<<endl;

int difference=n1-n2;

cout<<"difference:"<<difference<<endl;

int product=n1\*n2;

cout<<"product:"<<product<<endl;

int division=n1/n2;

cout<<"division:"<<division<<endl;

int modulo=n1%n2;

cout<<"modulo:"<<modulo<<endl;

return 0;

}

1. **Program to determine the integer is odd or even**

**PROGRAM:**

#include<iostream>

using namespace std;

int main()

{

int num;

cout<<"enter the number:";

cin>>num;

if(num%2==0){

cout<<"even number";

}

else

{

cout<<"odd number";

}

return 0;

}

1. **Program to compute the average of three integers**

**PROGRAM:**

#include<iostream>

using namespace std;

int main()

{

int n1,n2,n3;

cout<<"enter number1:";

cin>>n1;

cout<<"enter number2:";

cin>>n2;

cout<<"enter number3:";

cin>>n3;

int average=(n1+n2+n3)/3;

cout<<"average of"<<n1<<","<<n2<<","<<n3<<"is:"<<average<<endl;

return 0;

}

1. **Program to check two numbers are equal or not**

**PROGRAM:**

#include<iostream>

using namespace std;

int main()

{

int n1, n2;

cout<<"enter number1:";

cin>>n1;

cout<<"enter number2:";

cin>>n2;

if(n1==n2){

cout<<"equal";

}

else{

cout<<"not equal";

}

return 0;

}

1. **Write a program to read in two Floating numbers and perform the following operations on them: addition, subtraction, multiplication, division, and modulo.**

**PROGRAM:**

#include<iostream>

using namespace std;

int main()

{

float n1,n2;

cout<<"enter number 1:";

cin>>n1;

cout<<"enter number 2:";

cin>>n2;

float sum=n1+n2;

cout<<"sum:"<<sum<<endl;

float difference=n1-n2;

cout<<"difference:"<<difference<<endl;

float product=n1\*n2;

cout<<"product:"<<product<<endl;

float division=n1/n2;

cout<<"division:"<<division<<endl;

//modulo is not defined for the floating points

return 0;

}

1. **Program to check the character is a vowel or consonant**

**PROGRAM:**

#include<iostream>

using namespace std;

int main()

{

char ch;

cout<<"enter a character:";

cin>>ch;

if(ch=='a'||ch=='e'||ch=='i'||ch=='o'||ch=='u'){

cout<<"character is a vowel";

}

else{

cout<<"character is a consonant";

}

return 0;

}

1. **Program to check the number is positive, negative or zero**

**PROGRAM:**

#include<iostream>

using namespace std;

int main()

{

int num;

cout<<"enter the number:";

cin>>num;

if(num>0){

cout<<"the number is positive";

}

if(num<0){

cout<<"the number is negative";

}

if(num==0){

cout<<"the number is zero";

}

return 0;

}

1. **Program to determine which number is greater among two integers**

**PROGRAM:**

#include<iostream>

using namespace std;

int main(){

int n1,n2;

cout<<"enter number 1:";

cin>>n1;

cout<<"enter number 2:";

cin>>n2;

if(n1>n2){

cout<<"greater among two integers is:"<<n1<<endl;

}

else{

cout<<"greater among two integers is:"<<n2<<endl;

}

return 0;

}

1. **Program to read a floating-number and round it to the nearest integer using the floor an ceil functions.**

**PROGRAM:**

#include<iostream>

#include<cmath>

using namespace std;

int main()

{

double num;

cout<<"enter a floating point number:";

cin>>num;

int roundeddown=floor(num);

int roundedup=ceil(num);

cout<<"original number:"<<num<<endl;

cout<<"rounded down(floor):"<<roundeddown<<endl;

cout<<"rounded up(ceil):"<<roundedup<<endl;

return 0;

}

1. **Program to swap two numbers using bitwise XOR operator**

**PROGRAM:**

#include<iostream>

using namespace std;

int main()

{

int a=3;

int b=4;

a=a^b;

b=a^b;

a=a^b;

cout<<"after swapping: a="<<a<<",b="<<b<<endl;

return 0;

}

1. **Largest among three numbers using ternary conditional operator**

**PROGRAM:**

#include<iostream>

using namespace std;

int main()

{

int n1,n2,n3;

cout<<"enter three numbers:";

cin>>n1>>n2>>n3;

int max=(n1>n2)?((n1>n3)?n1:n2):((n2>n3)?n2:n3);

cout<<"the largest number is:"<<max<<endl;

return 0;

}

1. **Program to check two numbers are equal or not using ternary conditional operator**

**PROGRAM:**

#include<iostream>

using namespace std;

int main(){

int n1,n2;

cout<<"enter the first number:";

cin>>n1;

cout<<"enter the second number:";

cin>>n2;

cout<<(n1==n2 ?"the numbers are equal":"the numbers are not equal")<<endl;

return 0;

}

1. **Program to check the integer is divisible by 3 or not using ternary conditional operator**

**PROGRAM:**

#include<iostream>

using namespace std;

int main()

{

int num;

cout<<"enter the number:";

cin>>num;

cout<<(num%3==0 ? "divisible by 3":"not divisible by 3")<<endl;

return 0;

}

**14.Program to print numbers from 1 to 10 using for loop**

**PROGRAM:**

#include<iostream>

using namespace std;

int main()

{

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

cout<<i<<" ";

return 0;

}

**15.Factorial of a number using for loop**

**PROGRAM:**

#include<iostream>

using namespace std;

int main()

{

int i,fact=1,num;

cout<<"enter a number:";

cin>>num;

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

fact=fact\*i;

}

cout<<"factorial of a number is:"<<fact<<endl;

return 0;

}

**16.Print multiplication table using for loop**

**PROGRAM:**

#include<iostream>

using namespace std;

int main()

{

int num;

cout<<"enter the number:";

cin>>num;

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

cout<<num<<"\*"<<i<<"="<<num\*i<<endl;

}

return 0;

}

**17.Fibonacci series using for loop**

**PROGRAM:**

#include<iostream>

using namespace std;

int main()

{

int n,t1=0,t2=1,t3=0;

cout<<"enter the number of terms:";

cin>>n;

cout<<"fibonacci series:";

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

if(i==1){

cout<<t1<<",";

continue;

}

if(i==2){

cout<<t2<<",";

continue;

}

t3=t1+t2;

t1=t2;

t2=t3;

cout<<t3<<",";

}

return 0;

}

**18.Prime number using for loop**

**PROGRAM:**

#include<iostream>

using namespace std;

int main()

{

int n,i,m=0,flag=0;

cout<<"enter the number:";

cin>>n;

m=n/2;

for(i=2;i<=m;i++)

{

if(n%i==0)

{

cout<<"not a prime number"<<endl;

flag=1;

break;

}

}

if(flag==0)

cout<<"prime number"<<endl;

return 0;

}

**19.Check the string is palindrome or not using while loop**

**PROGRAM:**

**20.Sum of all digits using while loop (n=123 output:1+2+3=6)**

**PROGRAM:**

#include<iostream>

using namespace std;

int main(){

int n,sum=0,m;

cout<<"enter a number:";

cin>>n;

while(n>0)

{

m=n%10;

sum=sum+m;

n=n/10;

}

cout<<"sum is:"<<sum<<endl;

return 0;

}

**21.GCD of two numbers using do-while loop**

**PROGRAM:**

#include<iostream>

using namespace std;

int main()

{

int n1,n2;

cout<<"enter the first number:";

cin>>n1;

cout<<"enter the second number:";

cin>>n2;

int gcd;

do{

int temp=n1;

n1=n2;

n2=temp%n2;

}

while(n2!=0);

gcd=n1;

cout<<"GCD of n1 and n2 is:"<<gcd<<endl;

return 0;

}

**22.Check whether the number is perfect or not**

**PROGRAM:**

#include<iostream>

using namespace std;

int main(){

int n,sum=0;

cout<<"enter a number:";

cin>>n;

for(int i=1;i<=n/2;i++){

if(n%i==0){

sum+=i;

}

}

if(sum==n){

cout<<"perfect number"<<endl;

}

else{

cout<<"not a perfect number"<<endl;

}

return 0;

}

**23.Armstrong number**

**PROGRAM:**

#include<iostream>

using namespace std;

int main(){

int n,r,sum=0,temp;

cout<<"enter the number:";

cin>>n;

temp=n;

while(n>0)

{

r=n%10;

sum=sum+(r\*r\*r);

n=n/10;

}

if(temp==sum)

cout<<"armstrong number"<<endl;

else

cout<<"not an armstrong number"<<endl;

return 0;

}

**24.Harshad number**

**PROGRAM:**

#include<iostream>

using namespace std;

int sumofdigits(int num){

int sum=0;

while(num>0){

sum+=num%10;

num/=10;

}

return sum;

}

int main()

{

int num;

cout<<"enter a number:";

cin>>num;

int sum=sumofdigits(num);

if(num%sum==0){

cout<<"harshad number"<<endl;

}

else{

cout<<"not a harshad number"<<endl;

}

return 0;

}

**25.Happy number**

**PROGRAM:**

#include<iostream>

using namespace std;

int check(int num)

{

int rem=0,sum=0;

while(num>0)

{

rem=num%10;

sum=sum+(rem\*rem);

num/=10;

}

return sum;

}

void ishappynumber(int num)

{

int result=num;

while(result!=1&& result!=4)

{

result=check(result);

}

if(result==1)

cout<<"happy number"<<endl;

else

if(result==4)

cout<<"not a happy number"<<endl;

}

int main(){

int num;

cout<<"enter a number:";

cin>>num;

ishappynumber(num);

return 0;

}

**26.strong number**

**PROGRAM:**

#include<iostream>

using namespace std;

int main()

{

int i, num, temp, rem;

long sum = 0, factorial = 1;

cout << "enter a number:";

cin >> num;

for(temp = num; temp > 0; temp = temp / 10 )

{

factorial = 1;

rem = temp % 10;

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

{

factorial = factorial \* i;

}

cout << "\nThe Factorial of "<< rem << " = " << factorial;

sum = sum + factorial;

}

cout << "\n\nThe Sum of the Factorials of " << num << " is = " << sum << "\n\n";

if(num == sum)

{

cout<< "Strong Number";

}

else

{

cout<< "Not a Strong Number";

}

return 0;

}

**27.buzz number**

**PROGRAM:**

#include <iostream>

#include <conio.h>

using namespace std;

int main()

{

int n;

cout<<"Enter a number ";

cin>>n;

if(n%7==0 || n%10==7)

{

cout<<"Buzz number";

}

else

{

cout<<"Not a buzz number";

}

return 0;

}

**28.neon number**

**PROGRAM:**

#include <iostream>

using namespace std;

int main()

{

int num;

cout << "Enter the number: ";

cin >> num;

int square = num \* num;

int sum = 0;

while (square > 0)

{

int lastDigit = square % 10;

sum = sum + lastDigit;

square = square / 10;

}

if (sum == num)

cout<<"Neon number";

else

cout<<"NOT a Neon number";

return 0;

}

**29.abundant number**

**PROGRAM:**

#include <iostream>

using namespace std;

int sumOfDivisors(int num) {

int sum = 1;

for (int i = 2; i <= num / 2; ++i) {

if (num % i == 0) {

sum += i;

}

}

return sum;

}

bool isAbundant(int num) {

return sumOfDivisors(num) > num;

}

int main() {

int num;

cout << "Enter a number: ";

cin >> num;

if (isAbundant(num)) {

cout<<"abundant number." <<endl;

} else {

cout<<"not an abundant number." <<endl;

}

return 0;

}

**30.narcissistic number**

**PROGRAM:**

#include<iostream>

using namespace std;

int main(){

int n,r,sum=0,temp;

cout<<"enter the number:";

cin>>n;

temp=n;

while(n>0)

{

r=n%10;

sum=sum+(r\*r\*r);

n=n/10;

}

if(temp==sum)

cout<<"narcissistic number"<<endl;

else

cout<<"not an narcissistic number"<<endl;

return 0;

}

**31.print the pattern 1 22 333 4444 55555**

**PROGRAM:**

#include<iostream>

using namespace std;

int main(){

int rows=5;

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

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

cout<<i<<" ";

}

cout<<endl;

}

return 0;

}

**32.print the pattern \* \*\* \*\*\* \*\*\*\* \*\*\*\*\***

**PROGRAM:**

#include<iostream>

using namespace std;

int main(){

int rows;

cout<<"enter the number of rows:";

cin>>rows;

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

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

cout<<"\*";

}

cout<<"\n";

}

return 0;

}

**33.Print pascal triangle pattern nested for loop**

**PROGRAM:**

#include<iostream>

using namespace std;

int main(){

int rows;

cout<<"enter the number of rows:";

cin>>rows;

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

int coefficient=1;

for(int space=1;space<=rows-i;space++){

cout<<" ";

}

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

cout<<coefficient<<" ";

coefficient=coefficient\*(i-j)/(j+1);

}

cout<<endl;

}

return 0;

}

**34.Print diamond pattern with \* using nested for loop**

**PROGRAM:**

#include<iostream>

using namespace std;

int main(){

int n;

cout<<"enter the number of rows:";

cin>>n;

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

for(int j=1;j<=n-i;j++){

cout<<" ";

}

for(int k=1;k<=2\*i-1;k++){

cout<<"\*";

}

cout<<endl;

}

for(int i=n-1;i>=1;i--){

for(int j=1;j<=n-i;j++){

cout<<" ";

}

for(int k=1;k<=2\*i-1;k++){

cout<<"\*";

}

cout<<endl;

}

return 0;

}

**35.Program to reverse the elements in an array**

**PROGRAM:**

#include<iostream>

using namespace std;

int main(){

int arr[10],n,temp,i,j;

cout<<"enter the number of elements:";

cin>>n;

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

{

cout<<"enter the elements"<<i+1<<":";

cin>>arr[i];

}

for(i=0,j=n-1;i<n/2;i++,j--)

{

temp=arr[i];

arr[i]=arr[j];

arr[j]=temp;

}

cout<<"\n reverese all elements of the array"<<endl;

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

{

cout<<arr[i]<<" ";

}

return 0;

}

**36.Program to insert an element in an array at a specific position**

**PROGRAM:**

#include<iostream>

using namespace std;

int main(){

int arr[50],i,elem,pos,tot;

cout<<"enter the size of the array:";

cin>>tot;

cout<<"enter"<<tot<<"array elements:";

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

cin>>arr[i];

cout<<"\nenter element to be inserted:";

cin>>elem;

cout<<"enter the position to insert:";

cin>>pos;

for(i=tot;i>=pos;i--)

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

arr[i]=elem;

tot++;

cout<<"\nthe new array is:\n";

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

cout<<arr[i]<<" ";

cout<<endl;

return 0;

}

**37.Program to Delete an element in an array at a specific position**

**PROGRAM:**

#include<iostream>

using namespace std;

int main()

{

int arr[50], tot, i, elem, j, found=0;

cout<<"Enter the Size: ";

cin>>tot;

cout<<"Enter "<<tot<<" Array Elements: ";

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

cin>>arr[i];

cout<<"\nEnter Element to Delete: ";

cin>>elem;

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

{

if(arr[i]==elem)

{

for(j=i; j<(tot-1); j++)

arr[j] = arr[j+1];

found=1;

i--;

tot--;

}

}

if(found==0)

cout<<"\nElement doesn't found in the Array!";

else

{

cout<<"\nElement Deleted Successfully!";

cout<<"\n\nThe New Array is:\n";

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

cout<<arr[i]<<" ";

}

cout<<endl;

return 0;

}

**38.Find the sum of all elements in an array**

**PROGRAM:**

#include <iostream>

using namespace std;

int main()

{

int arr[100],i,size,n,sum=0;

cout<<"Enter the number of elements: ";

cin>>size;

cout<<"Enter the value of elements: "<<endl;

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

{

cin>>arr[i];

}

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

{

sum=sum+arr[i];

}

cout<<"Sum of elements in an array is: "<<sum;

return 0;

}

**39.Find the average of all elements in an array**

**PROGRAM:**

#include <iostream>

using namespace std;

int main()

{

int n, i;

float num[100], sum=0.0, average;

cout << "Enter the size of array: ";

cin >> n;

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

{

cout << i + 1 << ". Enter number: ";

cin >> num[i];

sum += num[i];

}

average = sum / n;

cout << "Average = " << average;

return 0;

}

**40.Find the second largest element in an array**

**PROGRAM:**

#include <iostream>

using namespace std;

int main(){

int n, num[50], largest, second;

cout<<"Enter number of elements: ";

cin>>n;

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

cout<<"Enter Array Element"<<(i+1)<<": ";

cin>>num[i];

}

if(num[0]<num[1]){

largest = num[1];

second = num[0];

}

else{

largest = num[0];

second = num[1];

}

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

if (num[i] > largest) {

second = largest;

largest = num[i];

}

else if (num[i] > second && num[i] != largest) {

second = num[i];

}

}

cout<<"Second Largest Element in array is: "<<second;

return 0;

}

**41.Find the number of occurrences of a value in an array**

**PROGRAM:**

#include <iostream>

using namespace std;

int countOccurrences(int arr[], int size, int target) {

int count = 0;

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

if (arr[i] == target) {

count++;

}

}

return count;

}

int main() {

int size, target;

cout << "Enter the size of the array: ";

cin >> size;

if (size <= 0) {

cout << "Array size should be greater than 0." << endl;

return 1;

}

int arr[size];

cout << "Enter the elements of the array:" << endl;

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

cout << "Enter element " << i + 1 << ": ";

cin >> arr[i];

}

cout << "Enter the value to find occurrences: ";

cin >> target;

int occurrences = countOccurrences(arr, size, target);

cout << "The number of occurrences of " << target << " in the array is: " << occurrences << endl;

return 0;

}

**42.Merge two array**

**PROGRAM:**

#include<iostream>

using namespace std;

int main()

{

int arrOne[50], arrTwo[50], arrMerge[100];

int size1, size2, i, k;

cout<<"Enter the Size for First Array: ";

cin>>size1;

cout<<"Enter Elements for First Array: ";

for(i=0; i<size1; i++)

{

cin>>arrOne[i];

arrMerge[i] = arrOne[i];

}

k = i;

cout<<"\nEnter the Size for Second Array: ";

cin>>size2;

cout<<"Enter Elements for Second Array: ";

for(i=0; i<size2; i++)

{

cin>>arrTwo[i];

arrMerge[k] = arrTwo[i];

k++;

}

cout<<"\nThe New Array (Merged Array):\n";

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

cout<<arrMerge[i]<<" ";

cout<<endl;

return 0;

}

**43.Create a dynamic array using pointers and display the values**

**PROGRAM:**

#include<iostream>

using namespace std;

int main(){

int size;

cout<<"enter the size of the dynamic array:";

cin>>size;

int\*dynamicarray=new int[size];

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

dynamicarray[i]=i\*2;

}

cout<<"value of the dynamic array:";

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

cout<<dynamicarray[i]<<" ";

}

delete[] dynamicarray;

return 0;

}

**44.Create a dynamic 2D (Two dimensional) array using pointers and display the values**

**PROGRAM:**

#include <iostream>

using namespace std;

int main() {

int rows, cols;

cout << "Enter the number of rows: ";

cin >> rows;

cout << "Enter the number of columns: ";

cin >> cols;

int\*\* dynamicArray = new int\*[rows];

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

dynamicArray[i] = new int[cols];

}

int value = 1;

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

for (int j = 0; j < cols; ++j) {

dynamicArray[i][j] = value++;

}

}

cout << "Dynamic 2D Array:" << endl;

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

for (int j = 0; j < cols; ++j) {

cout << dynamicArray[i][j] << " ";

}

cout << endl;

}

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

delete[] dynamicArray[i];

}

delete[] dynamicArray;

return 0;

}

**45.Add 2 matrices**

**PROGRAM:**

#include<iostream>

using namespace std;

int main()

{

int rows,cols;

cout<<"enter the number of rows:";

cin>>rows;

cout<<"enter the number of columns:";

cin>>cols;

int matrix1[rows][cols],matrix2[rows][cols],result[rows][cols];

cout<<"enter elements of first matrix:\n";

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

for(int j=0;j<cols;j++){

cin>>matrix1[i][j];

}

}

cout<<"enter elements of second matrix:\n";

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

for(int j=0;j<cols;j++){

cin>>matrix2[i][j];

}

}

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

for(int j=0;j<cols;j++){

result[i][j]=matrix1[i][j]+matrix2[i][j];

}

}

cout<<"resultant matrix after addition:\n";

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

for(int j=0;j<cols;j++){

cout<<result[i][j]<<" ";

}

cout<<endl;

}

return 0;

}

**46.Multiply 2 matrices**

**PROGRAM:**

#include <iostream>

using namespace std;

int main()

{

int a[10][10],b[10][10],mul[10][10],r,c,i,j,k;

cout<<"enter the number of row=";

cin>>r;

cout<<"enter the number of column=";

cin>>c;

cout<<"enter the first matrix element:\n";

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

{

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

{

cin>>a[i][j];

}

}

cout<<"enter the second matrix element:\n";

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

{

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

{

cin>>b[i][j];

}

}

cout<<"multiply of the matrix:\n";

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

{

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

{

mul[i][j]=0;

for(k=0;k<c;k++)

{

mul[i][j]+=a[i][k]\*b[k][j];

}

}

}

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

{

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

{

cout<<mul[i][j]<<" ";

}

cout<<"\n";

}

return 0;

}

**47.Find the sum of diagonals of a matrix**

**PROGRAM:**

#include<iostream>

using namespace std;

int main()

{

int i, j, rows, columns, sum = 0;

cout << "\n Enter the rows and Columns:";

cin >> i >> j;

int sumDgnalArr[i][j];

cout << "\nEnter the elements:\n";

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

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

cin >> sumDgnalArr[rows][columns];

}

}

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

{

sum = sum + sumDgnalArr[rows][rows];

}

cout << "\nThe Sum of Diagonal Elements:" << sum;

return 0;

}

**Functions in C++**

1. **Find factorial using function**

**PROGRAM:**

#include<iostream>

using namespace std;

void fact();

int main()

{

fact();

}

void fact()

{

int n,i,fact=1;

cout<<"enter the number:";

cin>>n;

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

fact=fact\*i;

cout<<"\nfactorial of"<<n<<"is\n"<<fact;

}

1. **Find prime number using function**

**PROGRAM:**

#include<iostream>

using namespace std;

int prime(int);

int main()

{

int n,s;

cout<<"enter the number:";

cin>>n;

s=prime(n);

s==2? cout<<"prime number":cout<<"not a prime number";

}

int prime(int n)

{

int i,count=0;

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

if(n%i==0)

count++;

return count;

}

1. **Find the reverse of a string using function**

**PROGRAM:**

#include<iostream>

#include<cstring>

using namespace std;

void reverse(char[]);

int main()

{

char c[20];

cout<<"enter the string:";

gets(c);

reverse(c);

}

void reverse(char c[])

{

cout<<"reverse of string:"<<strrev(c);

}

1. **Find minimum and maximum element in an array using function**

**PROGRAM:**

#include<iostream>

using namespace std;

#define n 6

void max\_min(int[]);

int main()

{

int i,a[20];

cout<<"enter the elements of size 6:";

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

cin>>a[i];

max\_min(a);

}

void max\_min(int a[])

{

int max=a[0],min,i;

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

{

if(max<a[i])

max=a[i];

if(min>a[i])

min=a[i];

}

cout<<"\nmaximum:"<<max;

cout<<"\nminimum:"<<min;

}

1. **Find GCD of two number using function**

**PROGRAM:**

#include<iostream>

using namespace std;

void gcd();

int main()

{

gcd();

}

void gcd()

{

int n1,n2,g,i;

cout<<"enter two numbers:";

cin>>n1>>n2;

for(i=1;i<=n1||i<=n2;i++)

if(n1%i==0&&n2%i++)

g=i;

cout<<"gcd of n1 and n2 is:"<<g<<endl;

}

1. **Function to count the no of elements in a string**

**PROGRAM:**

#include<iostream>

#include<string.h>

using namespace std;

int count(char[]);

int main()

{

int a;

char c[100];

cout<<"enter the string:";

gets(c);

a=count(c);

cout<<"\nthe given string contains\n"<<a<<"words";

}

int count(char c[])

{

int i,count=1;

for(i=0;i<strlen(c);i++)

{

if(c[i]==' ')

count++;

}

return count;

}

1. **Convert Celsius and Fahrenheit using function**

**PROGRAM:**

#include<iostream>

using namespace std;

int temp(int c);

int main(){

int a,c;

cout<<"enter the temp in celsius:";

cin>>c;

a=temp(c);

cout<<"temp in fahrenheit:"<<a;

}

int temp(int c)

{

int f=(c\*9/5)+32;

return f;

}

1. **Find the area of a circle using function**

**PROGRAM:**

#include<iostream>

using namespace std;

#define pi 3.14

int area(int);

int main(){

int a,r;

cout<<"enter the radius:\n";

cin>>r;

a=area(r);

cout<<"radius:"<<a;

}

int area(int r)

{

int a=pi\*r\*r;

return a;

}

1. **Check whether the string is palindrome or not**

**PROGRAM:**

**Constructor and destructor**

1. **Write a c++ program to create a class for a bank account with a constructor and a destructor**

**PROGRAM:**

#include <iostream>

#include <string>

using namespace std;

class BankAccount {

private:

string accountHolder;

double balance;

public:

BankAccount(string holder, double initialBalance) : accountHolder(holder), balance(initialBalance) {

cout << "Creating a bank account for " << accountHolder << " with initial balance: $" << balance <<endl;

}

~BankAccount() {

cout << "Destroying the bank account for " << accountHolder <<endl;

}

};

int main() {

BankAccount myAccount("John Doe", 1000.0);

return 0;

}

1. **Write a c++ program to create a class for a car with a constructor and a destructor**

**PROGRAM:**

#include <iostream>

#include <string>

using namespace std;

class Car {

private:

string make;

string model;

public:

Car(const string& make, const string& model) : make(make), model(model) {

cout <<"Creating a car object."<<endl;

}

~Car() {

cout <<"Destroying the car object."<<endl;

}

};

int main() {

Car myCar("Toyota", "Camry");

return 0;

}

1. **Write a c++ program to create a class for a rectangle with a constructor and a destructor**

**PROGRAM:**

#include<iostream>

using namespace std;

class rectangle{

private:

double width;

double length;

public:

rectangle(double l,double w):length(l),width(w){

cout<<"creating a rectangle with length"<<length<<"and width"<<width<<endl;

}

~rectangle(){

cout<<"destroying a rectangle with length"<<length<<"and width"<<width<<endl;

}

double area(){

return length\*width;

}

};

int main(){

rectangle myrectangle(2.5,4.2);

double area=myrectangle.area();

cout<<"area of rectangle is:"<<area<<endl;

return 0;

}

1. **Write a c++ program to create a class for a book with a constructor and a destructor**

**PROGRAM:**

#include<iostream>

#include<string>

using namespace std;

class book{

private:

string title;

string author;

public:

book(string title,string author):title(title),author(author){

cout<<"creating a object book with"<<title<<"by"<<author<<endl;

}

~book(){

cout<<"destroying a object book with"<<title<<"by"<<author<<endl;

}

};

int main()

{

book mybook("c++","manasa");

return 0;

}

**5.Write a c++ program to create a class for student with a constructor and a destructor**

**PROGRAM:**

#include<iostream>

#include<string>

using namespace std;

class student{

private:

string name;

int age;

public:

student(string studentname, int studentage):name(studentname),age(studentage){

cout<<"creating a object student with" << name<<"and age" << age<<endl;

}

~student(){

cout<<"destroying an object student with" << name<<"and age" << age<<endl;

}

};

int main()

{

student mystudent("manasa",20);

return 0;

}

**Inheritance and pointers**

1. **Create a base class called Shape with data members for height and width. Derive two classes Rectangle and Triangle from the base class. Write member functions to calculate the area and perimeter of each class**

**PROGRAM:**

1. **Create a base class called vehicle with data members for make, model, and year. Derive two classes Car and Truck from the base class. The Car class should have additional data members for seating capacity and fuel type, while the Truck class should have additional data members for payload capacity and towing capacity. Write member functions to get and set the data members for each class**

**PROGRAM:**

#include<iostream>

using namespace std;

class vehicle{

string make;

string model;

int year;

public:

void get\_p()

{

cout<<"enter the make:";

cin>>make;

cout<<"enter the model:";

cin>>model;

cout<<"enter the year:";

cin>>year;

}

void display\_p()

{

cout<<"MAKE:"<<make<<"\nMODEL:"<<model<<"\nYEAR:"<<year<<endl;

}

};

class car:public vehicle

{

int capacity;

string fuel;

public:

void get\_s1()

{

get\_p();

cout<<"enter the seating capacity:";

cin>>capacity;

cout<<"enter the type of fuel used:";

cin>>fuel;

}

void display\_s1()

{

display\_p();

cout<<"SEATING CAPACITY:"<<capacity<<"\nTYPE OF FUEL:"<<fuel<<endl;

}

};

class truck:public vehicle

{

int payload;

int towing;

public:

void get\_s2()

{

get\_p();

cout<<"enter the payload capacity:";

cin>>payload;

cout<<"enter the towing capacity:";

cin>>towing;

}

void display\_s2()

{

display\_p();

cout<<"PAYLOAD CAPACITY:"<<payload<<"\nTOWING CAPACITY:"<<towing<<endl;

}

};

int main(){

car c;

c.get\_s1();

c.display\_s1();

truck t;

t.get\_s2();

t.display\_s2();

return 0;

}

1. **Create a base class called Animal with data members for name, species, and age. Derive two classes Cat and Dog from the base class. The Cat class should have additional data members for color and breed, while the Dog class should have additional data members for weight and breed. Write member functions to get and set the data members for each class**
2. **Create a base class called Employee with data members for name, d, and salary Derive two classes Manager and Engineer from the base class. The Manager class should have additional data members for department and bonus, while the Engineer class should have additional data members for specialty and hours. Write member functions to get and set the data members for each class**
3. **Create a base class called Person with data members for name, age, and gender. Derive two classes Student and Teacher from the base class. The Student class should have additional data members for roll number and class, while the Teacher class should have additional data members for subject and salary. Write member functions to get and set the data members for each class.**
4. **Write a C++ program to create a pointer to an integer and display its value.**

**PROGRAM:**

#include <iostream>

using namespace std;

int main() {

int myInteger = 42;

int\* myPointer = &myInteger;

cout << "Value using pointer: " << \*myPointer <<endl;

cout << "Value using variable: " << myInteger <<endl;

return 0;

}

1. **Write a C++ program to create a pointer to a float and display its value.**

**PROGRAM:**

#include <iostream>

using namespace std;

int main() {

float myFloat = 2.13;

float\* myPointer = &myFloat;

cout << "Value using pointer: " << \*myPointer << endl;

cout << "Value using variable: " << myFloat << endl;

return 0;

}

1. **Write a C++ program to create a pointer to a char and display its value.**

**PROGRAM:**

#include<iostream>

using namespace std;

int main(){

char mychar='A';

char\* mypointer=&mychar;

cout<<"the value using pointer:"<<mypointer<<endl;

cout<<"the value using variable:"<<mychar<<endl;

return 0;

}

1. **Write a C++ program to create a pointer to a double and display its value.**

**PROGRAM:**

#include<iostream>

using namespace std;

int main(){

double mydouble='3.14';

double\* mypointer=&mydouble;

cout<<"the value using pointer is:"<<mypointer<<endl;

cout<<"the value using variable is:"<<mydouble<<endl;

return 0;

}

1. **Write a C++ program to create a pointer to a string and display its value.**

**PROGRAM:**

#include <iostream>

#include <string>

using namespace std;

int main() {

string myString = "Hello, World!";

string\* ptrToString = &myString;

cout << "Value of myString: " << \*ptrToString <<endl;

return 0;

}

1. **Write a C++ program to create a pointer to an array of elements and display its value.**

**PROGRAM:**

#include<iostream>

using namespace std;

int main(){

string array[]={"manasa","orange","c++","saveetha"};

string\*ptr=array;

cout<<"values in the array using pointer:"<<endl;

for(int i=0;i<4;i++)

cout<<"value at index"<<i<<":"<<\*(ptr+i)<<endl;

return 0;

}

1. **Write a C++ program to create a pointer to an array of character and display its value.**

**PROGRAM:**

#include<iostream>

using namespace std;

int main(){

char array[]={'a','b','c','d','e'};

char\*ptr=array;

cout<<"values in the array using pointer:"<<endl;

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

cout<<"value at index"<<i<<":"<<\*(ptr+i)<<endl;

return 0;

}

1. **Write a C++ program to create a pointer to an array of floats and display its value.**
2. **Write a C++ program to create a pointer to an object and display its attributes.**
3. **Write a C++ program to create a pointer to a function and call the function using the pointer**.

**Polymorphism**

1. **Create a base class called Person with a virtual function work (). Derive two classes Employee and Manager from the base class. Implement the work () function for each class**

**PROGRAM:**

#include<iostream>

using namespace std;

class person{

public:

virtual void work(){

cout<<"the person is working"<<endl;

}

};

class employee:public person{

public:

void work() override{

cout<<"performing his duties"<<endl;

}

};

class manager:public person{

public:

void work() override{

cout<<"monitoring the employees"<<endl;

}

};

int main(){

employee employee;

manager manager;

cout<<"employee works";

employee.work();

cout<<"manager works";

manager.work();

person\*person1=&employee;

person\*person2=&manager;

cout<<"person1 work:";

person1->work();

cout<<"person2 work:";

person2->work();

return 0;

}

1. **Create a base class called Animal with a virtual function eat (). Derive two classes Herbivore and Carnivore from the base class. Implement the eat function for each class.**

**PROGRAM:**

#include<iostream>

using namespace std;

class animal{

public:

virtual void eat(){

cout<<"animal is eating"<<endl;

}

};

class herbivore:public animal{

public:

void eat() override{

cout<<" plants"<<endl;

}

};

class carnivore:public animal{

public:

void eat() override{

cout<<" other animals"<<endl;

}

};

int main(){

herbivore herbivore;

carnivore carnivore;

cout<<"herbivore eat";

herbivore.eat();

cout<<"carnivore eat";

carnivore.eat();

animal\*animal1=&herbivore;

animal\*animal2=&carnivore;

cout<<"animal1 eat:";

animal1->eat();

cout<<"aniaml2 eat:";

animal2->eat();

return 0;

}

1. **Create a base class called Shape with virtual functions area () and volume (). Derive two classes Sphere and Cylinder from the base class. Implement the area and volume () functions for each class**

**PROGARM:**

#include<iostream>

#include<cmath>

using namespace std;

class Shape {

public:

virtual double area() const = 0;

virtual double volume() const = 0;

};

class Sphere : public Shape {

private:

double radius;

public:

Sphere(double r) : radius(r) {}

double area() const override {

return 4 \* M\_PI \* pow(radius, 2);

}

double volume() const override {

return (4.0 / 3.0) \* M\_PI \* pow(radius, 3);

}

};

class Cylinder : public Shape {

private:

double radius;

double height;

public:

Cylinder(double r, double h) : radius(r), height(h) {}

double area() const override {

return 2 \* M\_PI \* radius \* (radius + height);

}

double volume() const override {

return M\_PI \* pow(radius, 2) \* height;

}

};

int main() {

Sphere sphere(2.0);

Cylinder cylinder(2.0, 5.0);

cout << "Sphere Area: " << sphere.area() << endl;

cout << "Sphere Volume: " << sphere.volume() << endl;

cout << "Cylinder Area: " << cylinder.area() << endl;

cout << "Cylinder Volume: " << cylinder.volume() << endl;

return 0;

}

1. **Create a base class called Person with a virtual function greet). Derive two classes Student and Teacher from the base class. implement the greet) function for each class**

**PROGRAM:**

#include<iostream>

using namespace std;

class person{

public:

virtual void greet(){

cout<<"person say hi"<<endl;

}

};

class student:public person{

public:

void greet() override{

cout<<"hello"<<endl;

}

};

class teacher:public person{

public:

void greet() override{

cout<<"good morning"<<endl;

}

};

int main(){

student student;

teacher teacher;

cout<<"student greet";

student.greet();

cout<<"teacher greet";

teacher.greet();

person\*person1=&student;

person\*person2=&teacher;

cout<<"person1 greets:";

person1->greet();

cout<<"person2 greets:";

person2->greet();

return 0;

}

1. **Create a base class called Shape with virtual functions area( ) and perimeter(). Derive two classes Rectangle and Triangle from the base class. Implement the area () and perimeter () functions for each class.**

**PROGRAM:**

#include <iostream>

using namespace std;

class Shape {

public:

virtual double area() const = 0;

virtual double perimeter() const = 0;

};

class Rectangle : public Shape {

private:

double length;

double width;

public:

Rectangle(double l, double w) : length(l), width(w) {}

double area() const override {

return length \* width;

}

double perimeter() const override {

return 2 \* (length + width);

}

};

class Triangle : public Shape {

private:

double side1;

double side2;

double side3;

public:

Triangle(double s1, double s2, double s3) : side1(s1), side2(s2), side3(s3) {}

double area() const override {

double s = (side1 + side2 + side3) / 2;

return (s \* (s - side1) \* (s - side2) \* (s - side3));

}

double perimeter() const override {

return side1 + side2 + side3;

}

};

int main() {

Rectangle rectangle(5.0, 3.0);

Triangle triangle(3.0, 4.0, 5.0);

cout << "Rectangle Area: " << rectangle.area() << endl;

cout << "Rectangle Perimeter: " << rectangle.perimeter() <<endl;

cout << "Triangle Area: " << triangle.area() <<endl;

cout << "Triangle Perimeter: " << triangle.perimeter() <<endl;

return 0;

}

1. **Create a base class called Vehicle with a virtual function drive(). Derive two classes Car and Truck from the base class. Implement the drive() function for each class.**

**PROGRAM:**

#include<iostream>

using namespace std;

class vehicle{

public:

virtual void drives(){

cout<<"vehicle is being driven";

}

};

class car:public vehicle{

public:

void drives() override{

cout<<"by a man"<<endl;

}

};

class truck:public vehicle{

public:

void drives() override{

cout<<"with a load"<<endl;

}

};

int main(){

car car;

truck truck;

cout<<"car drives";

car.drives();

cout<<"truck drives";

truck.drives();

vehicle\*vehicle1=&car;

vehicle\*vehicle2=&truck;

cout<<"vehicle1 drives:";

vehicle1->drives();

cout<<"vehicle2 drives:";

vehicle2->drives();

return 0;

}

1. **Create a base class called Employee with a virtual function calculate Pay(). Derive two classes Manager and Engineer from the base class. Implement the calculatePay () function for each class.**

**PROGRAM:**

#include <iostream>

using namespace std;

class Employee {

public:

virtual double calculatePay() {

return 0.0;

}

};

class Manager : public Employee {

public:

double calculatePay() override {

return 5000.0;

}

};

class Engineer : public Employee {

public:

double calculatePay() override {

return 4000.0;

}

};

int main() {

Manager manager;

Engineer engineer;

cout << "Manager's monthly pay: $" << manager.calculatePay() <<endl;

cout << "Engineer's monthly pay: $" << engineer.calculatePay() <<endl;

Employee\* employee1 = &manager;

Employee\* employee2 = &engineer;

cout << "Employee 1's monthly pay: $" << employee1->calculatePay() <<endl;

cout << "Employee 2's monthly pay: $" << employee2->calculatePay() <<endl;

return 0;

}

1. **Create a base class called Animal with a virtual function speak(). Derive two classes Cat and Dog from the base class. Implement the speak() function for each class.**

**PROGRAM:**

#include <iostream>

using namespace std;

class Animal {

public:

virtual void speak() {

cout << "Animal speaks" <<endl;

}

};

class Cat : public Animal {

public:

void speak() override {

cout << "Meow!" <<endl;

}

};

class Dog : public Animal {

public:

void speak() override {

cout << "Woof!" <<endl;

}

};

int main() {

Cat cat;

Dog dog;

cout << "Cat says: ";

cat.speak();

cout << "Dog says: ";

dog.speak();

Animal\* animal1 = &cat;

Animal\* animal2 = &dog;

cout << "Animal 1 says: ";

animal1->speak();

cout << "Animal 2 says: ";

animal2->speak();

return 0;

}

1. **Create a base class called Shape with a virtual function area(). Derive two classes Rectangle and Circle from the base class. Implement the area() function for each class.**

**PROGRAM:**

#include <iostream>

using namespace std;

class Shape {

public:

virtual double area() const = 0;

virtual ~Shape() {}

};

class Rectangle : public Shape {

private:

double length;

double width;

public:

Rectangle(double l, double w) : length(l), width(w) {}

double area() const override {

return length \* width;

}

};

class Circle : public Shape {

private:

double radius;

public:

Circle(double r) : radius(r) {}

double area() const override {

return 3.14159 \* radius \* radius;

}

};

int main() {

Rectangle rectangle(4, 5);

Circle circle(3);

const Shape\* shape1 = &rectangle;

const Shape\* shape2 = &circle;

cout << "Area of Rectangle: " << shape1->area() << endl;

cout << "Area of Circle: " << shape2->area() << endl;

return 0;

}