**C++**

**Programs**

**File**

**-**

|  |  |  |  |
| --- | --- | --- | --- |
| **INDEX** | | | |
| **S.no** | **Practicals** | **Page.no** | **Sign** |
| 1 | Practical-1 Welcome Program | 1 |  |
| * Write a Program to print “Welcome”. |
| 2 | Practical-2 Operators | 2 |  |
| * Write a Program to show the use of Arithmetical Operators. |
| 3 | Practical-3 Control Structures | 3-17 |  |
| * 3.1 Write a Program to Print value only for even numbers in for loop using if Statement. * 3.2 Write a Program to check whether input integer is even number or not using if-else Statement. * 3.3 Write a Program to find if an integer is even or odd or neither using nested if-else Statement. * 3.4 Write a Program to perform arithmetic operations on two numbers using if-else ladder. * 3.5 Write a Program to check whether entered character is vowel or not using switch statement. * 3.6 Write a Program to perform arithmetic operations on two numbers using switch statement. * 3.7 Write a Program to Print 1-10 values using for loop. * 3.8 Write a Program to Print 1-10 values using while loop. * 3.9 Write a Program to Print 1-10 values using do-while loop. * 3.10 Write a Program to show the useage and implementation of goto Statement. * 3.11 Write a Program to show the useage and implementation of break Statement. * 3.12 Write a Program to Print value only for even numbers in for loop using Continue Statement. |
| 4 | Practical-4 Arrays | 18-23 |  |
| * 4.1 Write a Program to implement 1-D array. * 4.2 Write a Program to implement 2-D array. * 4.3 Write a Program to print pascal triangle. * 4.4 Write a Program to print Star pattern. |
| 5 | Practical-5 Functions | 24-31 |  |
| * 5.1 Write a Program to implement a function with no argument and no return value. * 5.2 Write a Program to implement a function with no argument and return value. * 5.3 Write a Program to implement a function with arguments and no return value. * 5.4 Write a Program to implement a function with arguments and return value. * 5.5 Write a Program to find a factorial of a given number using function. * 5.6 Write a Program to swap two number using call by value. * 5.7 Write a Program to swap two number using call by reference. * 5.8 Write a Program to find factorial of a number using recursion. |
| 6 | Practical-6 Pointers | 32-40 |  |
| * 6.1 Write a Program to implement a pointer. * 6.2 Write a Program to show the increment of a pointer. * 6.3 Write a Program to show the decrement of a pointer. * 6.4 Write a Program to add two numbers using a pointer. * 6.5 Write a Program to subtract two numbers using a pointer. * 6.6 Write a Program to show the use of a pointer to pointer. * 6.7 Write a Program to show the use of array of pointer. * 6.8 Write a Program to show the use of dynamic memory allocation using pointer. |
| 7 | Practical-7 Strings | 41-47 |  |
| * 7.1 Write a Program to initialise and display a string. * 7.2 Write a Program to find length of a string. * 7.3 Write a Program to concatenate two strings. * 7.4 Write a Program to copy two strings. * 7.5 Write a Program to compare two strings. * 7.6 Write a Program to reverse a string and perform lowercase and uppercase string function on string |
| 8 | Write a Program to define the member function inside the class | 48-49 |  |
| 9 | Write a Program to define the member function outside the class. | 50-51 |  |
| 10 | Write a Program to show the use of array of object. | 52-53 |  |
| 11 | Write a Program to pass object as an argument. | 54-55 |  |
| 12 | Write a Program to return object as an argument. | 56-57 |  |
| 13 | Write a Program to show the use of Friend Function. | 58-59 |  |
| 14 | Write a Program to show the use of Constant Member Function. | 60-61 |  |
| 15 | Write a Program to show the use of Static Variable in Function. | 62 |  |
| 16 | Write a Program to show the use of Static Data Members and Static Member functions in Class. | 63-64 |  |
| 17 | Write a Program to show the use of default constructor. | 65 |  |
| 18 | Write a program To Show The Use Of Parameterized Constructor. | 66-67 |  |
| 19 | Write a program To Show Constructor Overloading. | 68-69 |  |
| 20 | Write a program To Show The Use Of Constructor With Default Parameters. | 70-71 |  |
| 21 | Write a program To Show The Use Of Copy Constructor. | 72-73 |  |
| 22 | Write a program To Show The Use Of Dynamic Constructor. | 74-75 |  |
| 23 | Write a Program To Show The Use Of Explicit Constructor | 76-77 |  |
| 24 | Write a program To Show The Use Of Destructor. | 78 |  |
| 25 | Write a program to use Constructor in Array of Objects. | 79-80 |  |
| 26 | Write a program to use Constructor in Array of Objects. | 81 |  |
| 27 | Write a program to show the use of Nested Class. | 82 |  |
| 28 | Write a program to show the use of Friend Class. | 83 |  |
| 29 | Write a program to show the use of Pointers to Object. | 84-85 |  |
| 30 | Write a program to define local, global and static object. | 86 |  |
| 31 | Write a program to define Local class. | 87-88 |  |
| 32 | Write a program to overload Unary Operator | 89-90 |  |
| 33 | Write a program to overload Binary Operator. | 91-92 |  |
| 34 | Write a program to overload Binary Operator using Friend Function. | 93-94 |  |
| 35 | Write a program to overload Relational Operator. | 95-96 |  |
| 36 | Write a program to overload memory management operator | 97-98 |  |
| 37 | Write a program to demonstrate the typecasting of basic data type to class type | 99-100 |  |
| 38 | Write a program to demonstrate the typecasting of class type to basic data type | 101-102 |  |
| 39 | Write a program to demonstrate the typecasting of class type to another class type | 103-105 |  |
| 40 | Write a program to implement single inheritance | 106-107 |  |
| 41 | Write a program to implement multi-level inheritance | 108-109 |  |
| 42 | Write a program to implement multiple inheritance | 110-111 |  |
| 43 | Write a program to implement Hierarchical inheritance | 112-113 |  |
| 44 | Write a program to implement Hybrid inheritance | 114-115 |  |
| 45 | Write a program to resolve ambiguity in multiple inheritance | 116-117 |  |
| 46 | Write a Program to show the use of virtual base class | 118-119 |  |
| 47 | Write a program to show the order of default constructor in inheritance | 120-121 |  |
| 48 | Write a program to show the order parameterized constructor in inheritance | 122-123 |  |
| 49 | Write a program to show the order of destructor | 124-125 |  |
| 50 | Write a program to show the use of composition | 126-128 |  |
| 51 | Write a program to Implement virtual function. | 129-130 |  |
| 52 | Write a program to show use of abstract class or pure virtual function | 131-132 |  |
| 53 | Write a program to implement Virtual Destructor | 133-134 |  |
| 54 | Write a program to implement This Pointer | 135-136 |  |
| 55 | Write a program to implement Exception Handling Mechanism | 137-138 |  |
| 56 | Write a program to implement Multiple catch block | 139-140 |  |
| 57 | Write a program to implement Catch all exception | 141-142 |  |
| 58 | Write a program to implement Rethrow an exception | 143-144 |  |
| 59 | Write a program to implement Exception specification | 145-146 |  |
| 60 | Write a Program to implement a template | 147 |  |
| 61 | Write a Program function to use template with more than one generic type | 148 |  |
| 62 | Write a Program to overload function template | 149 |  |
| 63 | Write a Program to show use of class template | 150-151 |  |
| 64 | Write a Program to open and close a file | 152 |  |
| 65 | Write a Program to write content in the file | 153 |  |
| 66 | Write a Program to read the contents of a file | 154 |  |
| 67 | Write a Program to read the content of the file using get function | 155 |  |
| 68 | Write a Program to find the size of a file | 156 |  |
|  | | | |

**Program no.1**

**Program in C++ to print Welcome**

#include<iostream>

using namespace std;

int main()

{

cout<<”Welcome”;

return 0;

}

**Output:**



**Program no. 2**

**Program to add, subtract, multiply and divide two numbers**

#include<iostream>

using namespace std;

int main(){

int a = 20;

int b = 5;

cout<<"The sum of two numbers is "<<a+b<<endl;

cout<<"The difference of two numbers is "<<a-b<<endl;

cout<<"The multiplication of the two numbers is "<<a\*b<<endl;

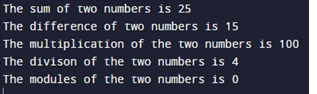
cout<<"The divison of the two numbers is "<<a/b<<endl;

cout<<"The modules of the two numbers is "<<a%b<<endl;

return 0;

}

**Output:**



**Program no. 3.1**

**Program to check if the number is positive, negative or zero using if statement.**

#include<iostream>

using namespace std;

int main(){

int a ;

cout<<"Enter the number"<<endl;

cin>>a;

if (a>0){

cout<<"The number is positive."<<endl;

}

if(a<0){

cout<<"The number is negative";

}

if(a==0){

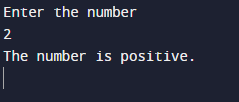
cout<<"The number is zero";

}

return 0;

}

**Output:**



**Program no. 3.2**

**Program to check if a number is even or odd using if – else statement.**

#include<iostream>

using namespace std;

int main(){

int a ;

cout<<"Enter the number"<<endl;

cin>>a;

if (a%2==0){

cout<<"The number is even."<<endl;

}

else{

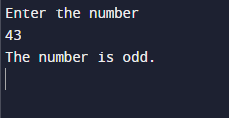
cout<<"The number is odd. "<<endl;

}

return 0;

}

**Output:**



**Program no. 3.3**

**Program to find the largest number using nested if statement.**

#include<iostream>

using namespace std;

int main(){

int a, b , c;

cout<<"Enter the three numbers "<<endl;

cin>>a>>b>>c;

if(a>=b) {

if (a >= c) {

cout << "The largest number is " << a << endl;

} else

{

cout << "The largest number is " << c << endl;

}

} else

{

if (b >= a) {

if (b >= c) {

cout << "The largest number is " << b << endl;

} else {

cout << "The largest number is " << c << endl;

}

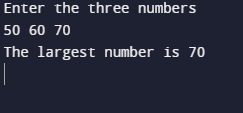
}

}

return 0;

}

**Output :**

****

**Program no. 3.4**

**Program in C++ to show the use of ladder else- if statement.**

#include<iostream>

using namespace std;

int main(){

int a;

cout<<"Enter a positive number"<<endl;

cin>>a;

if(a>=0 && a<=10){

cout<<"The number is between 0 and 10"<<endl;

} else if(a>=11 && a<=20){

cout<<"The number is between 11 and 20"<<endl;

}else if(a>=21 && a<=30){

cout<<"The number is between 21 and 30"<<endl;

}else if(a>30){

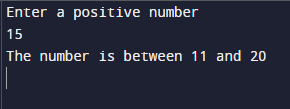
cout<<"The number is greater than 30"<<endl;

}

return 0;

}

**Output :**

****

**Program no. 3.5**

**Program in C++ to use Switch Statement.**

#include<iostream>

using namespace std;

int main(){

int n ;

cout<<"Enter a number from 1 to 5"<<endl;

cin>>n;

switch (n) {

case 1:

cout<<"Your choice is 1";

break;

case 2:

cout<<"Your choice is 2";

break;

case 3:

cout<<"Your choice is 3";

break;

case 4:

cout<<"Your choice is 4";

break;

case 5:

cout<<"Your choice is 5";

break;

default:

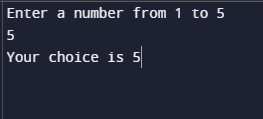
cout<<"Wrong Input";

}

return 0;

}

**Output :**

****

**Program no. 3.6**

**Program to add, subtract, multiply and divide two number using switch statement.**

#include<iostream>

using namespace std;

int main(){

int a = 40;

int b = 5;

char ch ;

cout<<"Enter\n '+' for addition \n '-' for subtraction \n '\*' for multiply \n or '/' for division" <<endl;

cin>>ch;

switch (ch) {

case '+':

cout<<"The sum is "<<a+b<<endl;

break;

case '-':

cout<<"The difference is "<<a-b<<endl;

break;

case '\*':

cout<<"The product is "<<a\*b;

break;

case '/':

cout<<" The division is "<<a/b;

break;

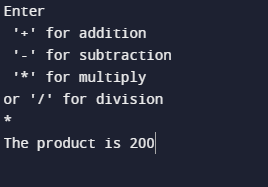
default:

cout<<"Wrong Input";

} return 0;

}

**Output :**



**Program no. 3.7**

**Program to print first n natural numbers using for loop.**

#include<iostream>

using namespace std;

int main(){

int n ;

cout<<"Enter the value of n"<<endl;

cin>>n;

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

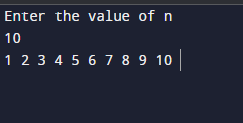
cout<<i<<" ";

}

return 0;

}

**Output :**



**Program no. 3.8**

**Program to print first n natural numbers using while loop.**

#include<iostream>

using namespace std;

int main(){

int n ;

int i = 1;

cout<<"Enter the value of n"<<endl;

cin>>n;

while(i<=n){

cout<<i<<" ";

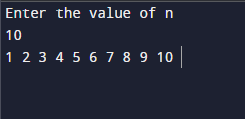
i++;

}

return 0;

}

**Output :**



**Program no. 3.9**

**Program to print first n natural numbers using do-while loop.**

#include<iostream>

using namespace std;

int main(){

int n ;

int i = 1;

cout<<"Enter the value of n"<<endl;

cin>>n;

do{

cout<<i<<" ";

i++;

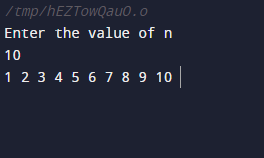
}

while(i<=n);

return 0;

}

**Output :**



**Program no. 3.10**

**Program in C++ to use goto statement.**

#include<iostream>

using namespace std;

int main(){

int n;

cout<<"Enter a number "<<endl;

cin>>n;

if(n%2 == 0){

goto print;

}

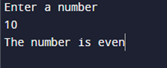
print:

cout<<"The number is even";

return 0;

}

**Output :**



**Program no. 3.11**

**Program in C++ to use break statement.**

#include<iostream>

using namespace std;

int main(){

int n = 50;

while(n>0){

cout<<n<<endl;

n--;

if(n==45){

break;

}

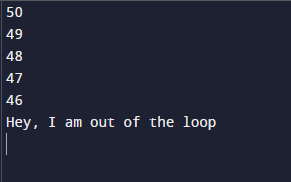
}

cout<<"Hey, I am out of the loop"<<endl;

return 0;

}

**Output :**



**Program no. 3.12**

**Program in C++ to use continue statement.**

#include<iostream>

using namespace std;

int main(){

for(int n = 250;n>240;n--){

if(n==247){

continue;

}

cout<<n<<endl;

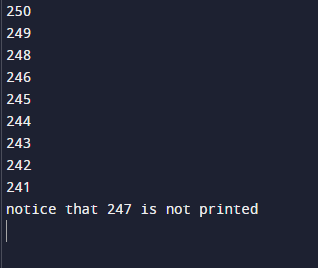
}

cout<<"notice that 247 is not printed"<<endl;

return 0;

}

**Output :**



**Program no. 4.1**

**Program in C++ to use 1d array.**

#include<iostream>

using namespace std;

int main(){

int n;

cout<<"Enter the number of elements you want to store in the array"<<endl;

cin>>n;

int arr[n];

cout<<"Enter the "<<n<<" elements"<<endl;

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

cin>>arr[i];

}

cout<<"Your array is "<<endl;

cout<<"{";

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

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

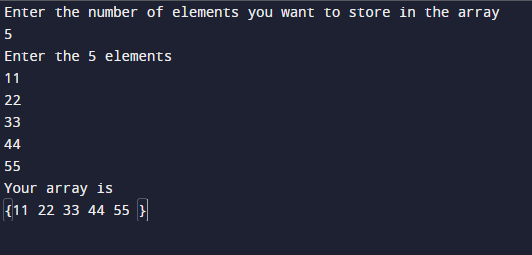
}

cout<<"}";

return 0;

}

**Output :**



**Program no. 4.2**

**Program in C++ to use 2-d array.**

#include<iostream>

using namespace std;

int main(){

int rows;

int columns;

cout<<"Enter the number of rows and columns "<<endl;

cin>>rows>>columns;

int arr[rows][columns];

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

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

cout << "Enter the element of " << i+1 << " row and " << j+1 << " column" << endl;

cin >> arr[i][j];

}

}

cout<<"Your array is "<<endl;

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

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

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

}

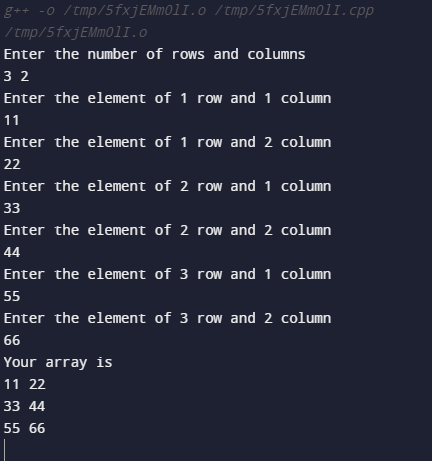
cout<<endl;

}

return 0;

}

**Output :**

****

**Program no. 4.3**

**Program in C++ to print a pattern.**

#include<iostream>

using namespace std;

int main(){

int n;

cout<<"Enter the value of n"<<endl;

cin>>n;

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

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

cout<<j<<" ";

}

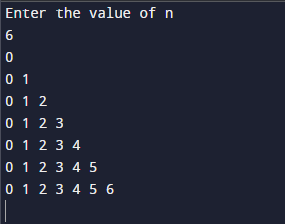
cout<<endl;

}

return 0;

}

**Output :**

****

**Program no. 4.4**

**Program in C++ to print the star pattern**

#include<iostream>

using namespace std;

int main(){

int n;

cout<<"Enter the value of n"<<endl;

cin>>n;

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

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

cout<<"\*"<<" ";

}

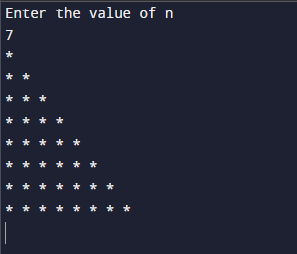
cout<<endl;

}

return 0;

}

**Output :**

****

**Program no. 5.1**

**Program to print two numbers using function with no arguments and no return value.**

#include<iostream>

using namespace std;

void print(){

int a = 30;

int b = 10;

cout<<endl;

cout<<"The sum of a and b is "<<a+b<<endl;

cout<<endl;

}

int main(){

print();

return 0;

}

**Output :**

****

**Program no. 5.2**

**Program to add two numbers using function with arguments but no return value.**

#include<iostream>

using namespace std;

void print(int a , int b){

cout<<endl;

cout<<"The sum of a and b is "<<a+b<<endl;

cout<<endl;

}

int main(){

int a = 50;

int b = 15;

print(a, b);

return 0;

}

**Output :**

****

**Program no. 5.3**

**Program to add two numbers using function with no arguments but return value.**

#include<iostream>

using namespace std;

int print(){

int a = 90;

int b = 10;

return a+b;

}

int main(){

int sum;

sum = print();

cout<<endl;

cout<<"The sum of a and b is "<<sum<<endl;

cout<<endl;

return 0;

}

**Output :**

****

**Program no. 5.4**

**Program to add two numbers using function with arguments and return value.**

#include<iostream>

using namespace std;

int print(int x, int y){

return x+y;

}

int main(){

int a = 15;

int b = 10;

int sum;

sum = print(a, b);

cout<<endl;

cout<<"The sum of a and b is "<<sum<<endl;

cout<<endl;

return 0;

}

**Output :**

****

**Program no. 5.5**

**Program to print factorial of a number using function in C++.**

#include<iostream>

using namespace std;

int fact(int a){

int factorial = 1;

while(a>0){

factorial = factorial \*a;

a--;

}

return factorial;

}

int main(){

int n;

cout<<"Enter the value of n"<<endl;

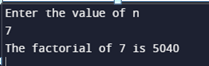
cin>>n;

int result = fact(n);

cout<<"The factorial of "<<n<< " is "<<result<<endl;

}

**Output :**

****

**Program no. 5.6**

**Program to sWrite a Program two numbers using call by value**

#include<iostream>

using namespace std;

void sWrite a Program p(int a, int i){

int temp = a;

a = i;

i = temp;

cout<<"After sWrite a Program ping the value of a is "<<a<<" and b is "<<i<<endl;

}

using namespace std;

int main(){

int a = 15;

int b = 10;

cout<<endl;

cout<<"Before sWrite a Program ping the value of a is "<<a<<" and b is "<<b<<endl;

sWrite a Program p(a,b);

return 0;

}

**Output :**



**Program no. 5.7**

**Program to sWrite a Program two numbers using call by reference**

#include<iostream>

using namespace std;

void sWrite a Program p(int \*a, int \*i){

int temp = \*a;

\*a = \*i;

\*i = temp;}

using namespace std;

int main(){

int a = 25;

int b = 30;

int \*p = &a;

int \*pp = &b;

cout<<endl;

cout<<"Before sWrite a Program ping the value of a is "<<a<<" and b is "<<b<<endl;

sWrite a Program p(p,pp);

cout<<"Before sWrite a Program ping the value of a is "<<a<<" and b is "<<b<<endl;

return 0;

}

**Output :**



**Program no. 5.8**

**Program to print Fibonacci series using recursion**

#include<iostream>

using namespace std;

int fib(int n){

if(n==0){

return 0;

} else if(n ==1){

return 1;

} else{

return fib(n-1) + fib(n-2);

}

}

int main(){

int n;

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

cin>>n;

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

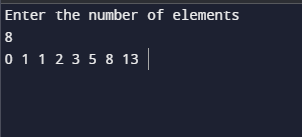
cout<<fib(i)<<" ";

}

return 0;

}

**Output :**



**Program no. 6.1**

**Program in C++ to show the use of pointer**

#include<iostream>

using namespace std;

int main(){

int \*ptr;

int var = 3;

ptr = &var;

cout<<"The value of variable is "<<var<<endl;

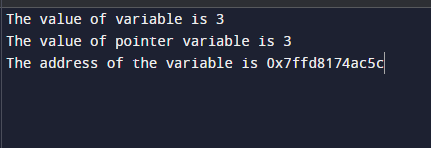
cout<<"The value of pointer variable is "<<\*ptr<<endl;

cout<<"The address of the variable is "<<ptr;

return 0;

}

**Output :**



**Program no. 6.2**

**Program in C++ to show increment of a pointer**

#include<iostream>

using namespace std;

int main(){

int \*ptr;

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

ptr = arr;

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

cout<<"The value of the pointer at index "<<i<<" is "<<\*ptr<<endl;

cout<<"The address of the pointer at index "<<i<<" is "<<ptr<<endl;

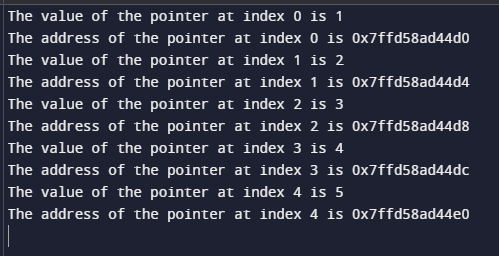
ptr++;

}

return 0;

}

**Output :**



**Program no. 6.3**

**Program in C++ to show decrement of pointer**

#include<iostream>

using namespace std;

int main(){

int \*ptr;

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

ptr = &arr[4];

for(int i = 4;i>=0;i--){

cout<<"The value of the pointer at index "<<i<<" is "<<\*ptr<<endl;

cout<<"The address of the pointer at index "<<i<<" is "<<ptr<<endl;

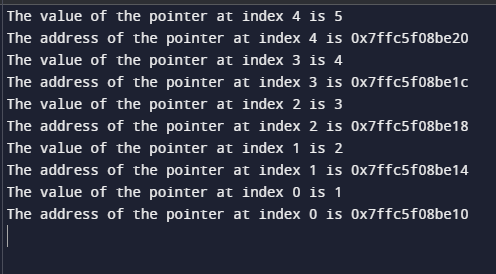
ptr--;

}

return 0;

}

**Output :**



**Program no. 6.4**

**Program to add two numbers using pointers.**

#include<iostream>

using namespace std;

int main(){

int a ;

int b ;

cout<<endl;

cout<<"Enter the two numbers "<<endl;

cin>>a>>b;

int \*p1 = &a;

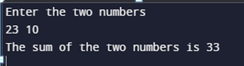
int \*p2 = &b;

cout<<"The sum of the two numbers is "<<\*p1 + \*p2<<endl;

return 0;

}

**Output :**



**Program no. 6.5**

**Program to subtract two numbers using pointers.**

#include<iostream>

using namespace std;

int main(){

int a ;

int b ;

cout<<endl;

cout<<"Enter the two numbers "<<endl;

cin>>a>>b;

int \*p1 = &a;

int \*p2 = &b;

cout<<"The difference of the two numbers is "<<\*p1 - \*p2<<endl;

return 0;

}

**Output :**

****

**Program no. 6.6**

**Program to show the pointer to pointer variable in C++.**

#include<iostream>

using namespace std;

int main(){

int var = 30;

int \*pInt = &var;

int \*\*pInt1 = &pInt;

cout<<"The value of the variable is "<<var<<endl;

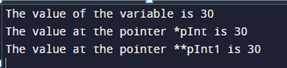
cout<<"The value at the pointer \*pInt is "<<\*pInt<<endl;

cout<<"The value at the pointer \*\*pInt1 is "<<\*\*pInt1<<endl;

return 0;

}

**Output :**



**Program no. 6.7**

**Program in C++ to use array of pointers.**

#include<iostream>

using namespace std;

int main(){

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

int \*ptr[5];

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

ptr[i] = &arr[i];

}

cout<<"The elements of the pointer array are "<<endl;

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

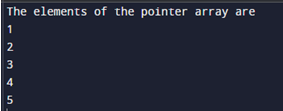
cout<<\*ptr[i]<<endl;

}

return 0;

}

**Output :**



**Program no. 6.8**

**Program in C++ to show dynamic memory allocation.**

#include<iostream>

using namespace std;

int main(){

int \*ptr = NULL;

ptr = new int;

\*ptr = 80;

cout<<"The value of the pointer is "<<\*ptr<<endl;

delete ptr;

}

**Output :**



**Program no. 6.9**

**Program to show the use of void pointer**

#include<iostream>

using namespace std;

int main(){

void \*ptr;

int a = 33;

float b = 99.11;

ptr = &a;

cout<<endl;

cout<<"The value of the pointer is "<<\*(int\*)ptr<<endl;

ptr = &b;

cout<<"The value of the pointer is "<<\*(float\*)ptr<<endl;

}

**Output :**



**Program no. 7.1**

**Program to initialize and display a string**

#include<iostream>

using namespace std;

int main(){

char s[50] = "Gurpreet Singh";

cout<<"The string is "<<endl;

cout<<s;

}

**Output :**



**Program no. 7.2**

**Program to find the length of a string**

#include<iostream>

#include<string.h>

using namespace std;

int main(){

char s[50] = "GURPREET SINGH";

int x;

x = strlen(s);

cout<<endl;

cout<<"The length of the string is "<<x<<endl;

}

**Output :**



**Program no. 7.3**

**Program to concatenate two strings**

#include<iostream>

#include<string.h>

using namespace std;

int main(){

char s[50] = "Good ";

char a[50] = "Night";

cout<<"The concatenated string is "<<endl;

cout<<strcat(s,a);

}

**Output:**

****

**Program no. 7.4**

**Program to copy a string in C++**

#include<iostream>

#include<string.h>

using namespace std;

int main(){

char s[50] = "Good ";

char a[50] = "Luck";

strcpy(s,a);

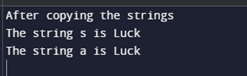
cout<<"After copying the strings"<<endl;

cout<<"The string s is "<<s<<endl;

cout<<"The string a is "<<a<<endl;

}

**Output:**



**Program no. 7.5**

**Program to compare two strings in C++**

#include<iostream>

#include<string.h>

using namespace std;

int main(){

char s1[50] = "Good";

char s2[50] = "Morning";

char s3[50] = "Good";

if(strcmp(s1, s2) == 0){

cout<<"Strings are equal"<<endl;

} else{

cout<<"String are not equal"<<endl;

}

if(strcmp(s1, s3) == 0){

cout<<"Strings are equal"<<endl;

} else{

cout<<"String are not equal"<<endl;

}

}

**Output:**



**Program no. 7.6**

**Program to reverse a string, convert a string to uppercase, and to lowercase in C++**

#include<iostream>

#include<cstring>

using namespace std;

int main() {

string str="Carryminati";

int i;

cout<<"string in lowercase\n";

for(i = 0; i<=str.length()-1; i++) {

if(str[i]>=97&&str[i]<=122){

cout<<str[i];

}

else{

str[i]=str[i]+32;

cout<<str[i];

}

}

cout<<endl;

cout<<"string in Uppercase\n";

for(i = 0; i<=str.length()-1; i++) {

if(str[i]>=65&&str[i]<=90){

cout<<str[i];

}

else{

str[i]=str[i]-32;

cout<<str[i];

}

}

cout<<"\nPrinting string in reverse\n";

for(i = str.length() - 1; i >= 0; i--) {

cout<<str[i];

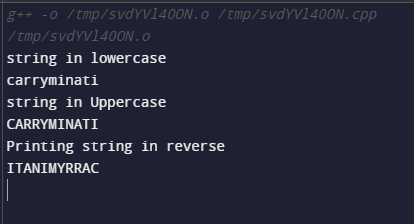
}

cout<<endl;

return 0;

}

**Output:**

****

**Program no. 8**

**Program to define the member functions inside the class in C++**

#include <iostream>

using namespace std;

class car

{

private:

int car\_number;

char car\_model[10];

public:

void getdata()

{

cout<<"Enter car number: ";

cin>>car\_number;

cout<<"\n Enter car model: ";

cin>>car\_model;

}

void showdata()

{

cout<<"Car number is "<<car\_number;

cout<<"\n Car model is "<<car\_model;

}

};

int main()

{

car c1;

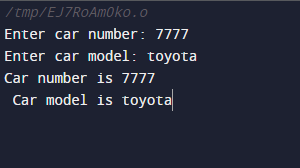
c1.getdata();

c1.showdata();

return 0;

}

**Output:**

****

**Program no. 9**

**Program to define the member functions outside the class in C++**

#include <iostream>

using namespace std;

class car

{

private:

int car\_number;

char car\_model[10];

public:

void getdata();

void showdata();

};

// function definition

void car::getdata()

{

cout<<"Enter car number: ";

cin>>car\_number;

cout<<"\n Enter car model: ";

cin>>car\_model;

}

void car::showdata()

{

cout<<"Car number is "<<car\_number;

cout<<"\n Car model is "<<car\_model;

}

int main()

{

car c1;

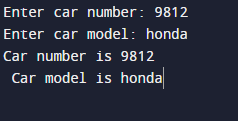
c1.getdata();

c1.showdata();

return 0;

}

**Output:**

****

**Program no. 10**

**Program to show the use of array of an object in C++**

#include <iostream>

using namespace std;

class Rectangle

{

public:

int length;

int breadth;

Rectangle( int l, int b )

{

length = l;

breadth = b;

}

int printArea()

{

return length \* breadth;

}

};

int main()

{

Rectangle rt1( 8, 6 );

Rectangle rt2( 7, 9 );

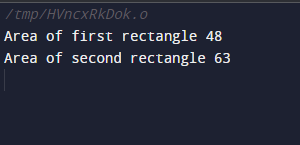
cout << "Area of first rectangle " << rt1.printArea() << endl;

cout << "Area of second rectangle " << rt2.printArea() << endl;

return 0;

}

Output:



**Program no. 11**

**Program to pass object as an argument in C++**

#include <iostream>

using namespace std;

class Demo

{

private:

int a;

public:

void set(int x)

{

a = x;

}

void sum(Demo ob1, Demo ob2)

{

a = ob1.a + ob2.a;

}

void print()

{

cout<<"Value of A : "<<a<<endl;

}

};

int main()

{

//object declarations

Demo d1;

Demo d2;

Demo d3;

//assigning values to the data member of objects

d1.set(30);

d2.set(60);

//passing object d1 and d2

d3.sum(d1,d2);

//printing the values

d1.print();

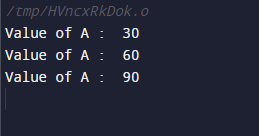
d2.print();

d3.print();

return 0;

}

**Output:**

****

**Program no. 12**

**Program to return object as an argument in C++**

#include <iostream>

using namespace std;

class Point {

int x;

int y;

public:

Point(int x1 = 0, int y1 = 0) {

x = x1;

y = y1;

}

Point addPoint(Point p) {

Point temp;

temp.x = x + p.x;

temp.y = y + p.y;

return temp;

}

void display() {

cout<<"x = "<< x <<"\n";

cout<<"y = "<< y <<"\n";

}

};

int main() {

Point p1(8,4);

Point p2(13,6);

Point p3;

cout<<"Point 1\n";

p1.display();

cout<<"Point 2\n";

p2.display();

p3 = p1.addPoint(p2);

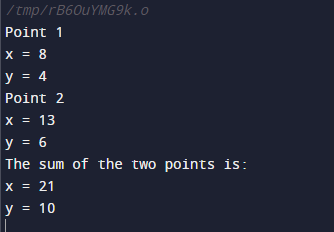
cout<<"The sum of the two points is:\n";

p3.display();

return 0;

}

**Output:**



**Program no. 13**

**Program to show the use of friend function in C++**

#include <iostream>

using namespace std;

class Temperature

{

int celsius;

public:

Temperature()

{

celsius = 0;

}

friend int temp( Temperature ); // declaring friend function

};

int temp( Temperature t ) // friend function definition

{

t.celsius = 45;

return t.celsius;

}

int main()

{

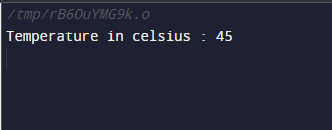
Temperature tm;

cout << "Temperature in celsius : " << temp( tm ) << endl;

return 0;

}

**Output;**

****

**Program no. 14**

**Program to show the use of constant member function in C++**

#include <iostream>

using namespace std;

class Numbers

{

private:

int a;

int b;

public:

Numbers()

{

a = 0;

b = 0;

}

void set\_a(int num1)

{

a = num1;

}

void set\_b(int num2)

{

b = num2;

}

int get\_a(void) const

{

return a;

}

int get\_b(void) const

{

return b;

}

};

int main()

{

Numbers Num;

Num.set\_a(50);

Num.set\_b(50);

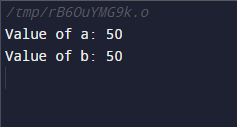
cout<<"Value of a: "<<Num.get\_a()<<endl;

cout<<"Value of b: "<<Num.get\_b()<<endl;

return 0;

}

**Output;**

****

**Program no. 15**

**Program to show the use of static variable in a class in C++**

#include <iostream>

using namespace std;

void counter()

{

static int count=0;

cout << count++<<" ";

}

int main()

{

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

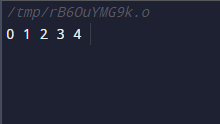
{

counter();

}

}

**Output;**

****

**Program no. 16**

**Program to show the use of static data member & static member function in class in C++**

#include<iostream>

using namespace std;

class Test{

static int val;

public:

Test(){

cout<<"Constructor called"<<endl;

val++;

}

static void print(){

cout<<"The number of objects is "<<val<<endl<<endl;

}

};

int Test::val = 0;

int main(){

cout<<endl;

Test::print();

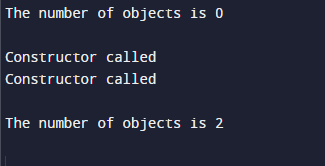
Test t,t1;

cout<<endl; Test::print();

return 0;

}

**Output**

****

**Program no. 17**

**Program to show the use of default constructor C++**

#include <iostream>

using namespace std;

class Cube

{

public:

int side;

Cube()

{

side = 30;

}

};

int main()

{

Cube c;

cout << c.side;

}

**Output:**

****

**Program no. 18**

**Program to show the use of parameterized constructor C++**

#include <iostream>

using namespace std;

class Cube

{

public:

int side;

Cube(int x)

{

side=x;

}

};

int main()

{

Cube c1(60);

Cube c2(70);

Cube c3(80);

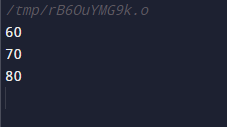
cout << c1.side<<endl;

cout << c2.side<<endl;

cout << c3.side<<endl;

}

**Output:**

****

**Program no. 19**

**Program to show the constructor overloading in C++**

#include<iostream>

using namespace std;

class Area{

int side;

int length , breadth;

float radius;

public:

Area(int side){

this->side = side;

cout<<"The area of the square is "<<side\*side<<endl;

}

Area(int length, int breadth){

this->length = length;

this->breadth = breadth;

cout<<"The area of the rectangle is "<<length\*breadth<<endl;

}

Area(float radius){

this->radius = radius;

cout<<"The area of the circle is "<<radius\*2\*3.14<<endl;

}

};

int main(){

cout<<endl;

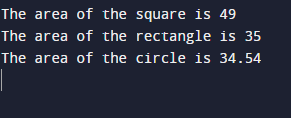
Area a(4);

Area b(4, 3);

Area c(3.3f);

}

**Output;**

****

**Program no. 20**

**Program to show the use of copy constructor in C++**

#include <iostream>

using namespace std;

class Wall {

private:

double length;

double height;

public:

Wall(double len, double hgt)

{

length = len;

height = hgt;

}

Wall(Wall &obj)

{

length = obj.length;

height = obj.height;

}

double calculateArea() {

return length \* height;

}

};

int main() {

Wall wall1(9.5, 8.1);

cout << "Area of Room 1: " << wall1.calculateArea() << endl;

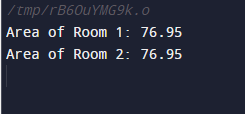
Wall wall2 = wall1;

cout << "Area of Room 2: " << wall2.calculateArea() << endl;

return 0;

}

Output:



**Program no. 21**

**Program to show the use of dynamic constructor in C++**

#include <iostream>

using namespace std;

class yes {

const char\* p;

public:

yes()

{

p = new char[6];

p = "mario”;

}

void display()

{

cout << p << endl;

}

};

int main()

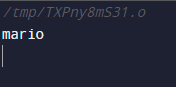
{

yes obj ;

obj.display();

}

**Output:**

****

**Program no. 22**

**Program to show the use of constructor with default parameters**

#include<iostream>

using namespace std;

class Para{

int a;

int b;

public:

Para(int a,int b = 20){

this->a = a;

this->b = b;

}

void show(){

cout<<"The values are "<< a<<" and "<<b<<endl;

}

};

int main(){

Para p1(40,60);

Para p2(90);

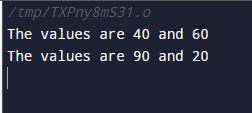
p1.show();

p2.show();

return 0;

}

**Output:**

****

**Program no. 23**

**Program to show the use of explicit constructor**

#include <iostream>

using namespace std;

class demo

{

int x,y;

public:

demo(int a ,int b)

{

x=a;

y=b;

}

void show ()

{

cout<<"\n Values are \n"<<x<<endl;

cout<<y<<endl;

}

};

int main ()

{

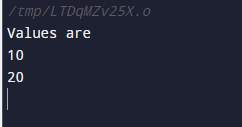
demo d = demo (10,20);

d.show();

return 0;

}

**Output:**

****

**Program no. 24**

**Program to show the use of destructor**

#include<iostream>

using namespace std;

class Des{

public:

Des(){

cout<<"constructor called"<<endl;

}

void print(){

cout<<"function called"<<endl;

}

~Des(){

cout<<"destructor called"<<endl;

}

};

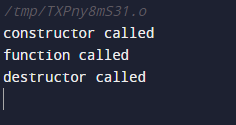
int main(){

Des d;

d.print();

return 0;

} **Output:**

****

**Program no. 25**

**Program to show the use of constructor in array of objects.**

#include<iostream>

using namespace std;

class Sum{

int a;

int b;

public:

Sum(int a, int b){

this->a = a;

this->b = b;

}

void add(){

cout<<"The addition is "<<a + b<<endl;

}

};

int main(){

Sum arr[3] = {Sum(10,20), Sum(30,79), Sum(56,8)};

for(int i= 0;i<3;i++){

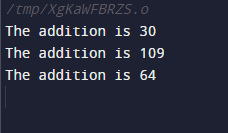
arr[i].add();

}

return 0;

}

**Output:**

****

**Program no. 26**

**Program to show the use of initializer list**

#include<iostream>

using namespace std;

class Demo{

int a;

int b;

public:

Demo(int first , int second):a(first), b(second){

cout<<"The value of a and b is "<<a<<" "<<b;

}

};

int main(){

Demo d(45, 78);

}

**Output:**



**Program no. 27**

**Program to show the use of nested class**

#include<iostream>

using namespace std;

class A{

public:

class B{

int a;

public:

void setData(int a){

this->a =a ;

}

void getData(){

cout<<"The value of a is "<<a;

}

};

};

int main(){

A::B object;

object.setData(100);

object.getData();

}

**Output:**

****

**Program no. 28**

**Program to show the use of friend class**

#include<iostream>

using namespace std;

class sum{

int a;

int b;

public:

sum(int a , int b){

this->a = a;

this->b = b;

}

friend int add(sum s);

};

int add(sum s){

return s.a + s.b;

}

int main(){

sum S1(10,15);

cout<<"The sum is "<<add(S1)<<endl;

}

**Output:**

****

**Program no. 29**

**Program to show the use of pointer to object**

#include<iostream>

using namespace std;

class Pointer{

int a;

public:

void setData(int a){

this->a = a;

}

void getData(){

cout<<"The value of a is "<<a<<endl;

}

};

int main(){

Pointer p;

p.setData(20);

p.getData();

Pointer\* ptr;

ptr = &p;

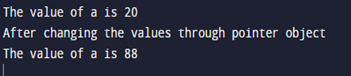
cout<<"After changing the values through pointer object"<<endl;

ptr->setData(88);

ptr->getData();

}

**Output:**

****

**Program no. 30**

**Program to define local object, global object and static object**

#include<iostream>

using namespace std;

class Demo{

string str;

public:

Demo(string str){

this->str = str;

cout<<"Constructor called for "<<str<<" object"<<endl;

}

};

Demo d3("global");

int main(){

Demo d1("local");

static Demo d2("static");

return 0;

}

**Output:**

****

**Program no. 31**

**Program to define a local class in c++**

#include<iostream>

using namespace std;

void myFunction(){

class Local{

int a;

int b;

public:

Local(int a, int b){

this->a = a;

this->b = b;

}

void add(){

cout<<"The sum is "<<a+b<<endl;

}

};

int a,b;

cout<<"Enter the values of a and b"<<endl;

cin>>a>>b;

Local l(a,b);

l.add();

}

int main(){

myFunction();

return 0;

}

**Output:**

****

**Program no. 32**

**Program to overload unary operator**

#include<iostream>

using namespace std;

class Unary{

int age;

public:

Unary(int age){

this->age = age;

cout<<"The age is "<<age<<endl;

}

Unary operator -(){

age-=1;

cout<<"The using the operator the age is "<<age<<endl;

}

};

int main(){

Unary u(40);

-u;

return 0;

}

**Output:**

****

**Program no. 33**

**Program to overload binary operator**

#include<iostream>

using namespace std;

class Binary{

string str;

public:

Binary(){

}

Binary(string str){

this->str = str;

};

Binary operator +(Binary b){

Binary bin;

bin.str = this->str +" " + b.str;

return bin;

}

void display(){

cout<<"The string is "<<str<<endl;

}

};

int main(){

Binary b1("Good");

Binary b2("Luck");

Binary b3 = b1 + b2;

b3.display();

return 0;

}

**Output:**

****

**Program no. 34**

**Program to overload binary operator using friend function**

#include<iostream>

using namespace std;

class Binary{

int str;

public:

Binary(){

}

Binary(int str){

this->str = str;

};

friend Binary operator +(Binary b, Binary c);

void display(){

cout<<"The sum is "<<str<<endl;

}

};

Binary operator +(Binary b, Binary c){

Binary bin;

bin.str = b.str + c.str;

return bin;

}

int main(){

Binary b1(125);

Binary b2(60);

Binary b3 = b1 + b2;

b3.display();

return 0;

}

**Output:**

****

**Program no. 35**

**Program to overload relational operators**

#include<iostream>

using namespace std;

class Height{

int feet;

int inch;

public:

Height(int feet = 0, int inch = 0){

this->feet = feet;

this->inch = inch;

}

bool operator ==(Height h1){

if(feet == h1.feet){

if(inch == h1.inch){

return true;

}

}

return false;

}

};

int main(){

Height h1(8,6);

Height h2(4,6);

if(h1 == h2){

cout<<"The heights are same"<<endl;

} else{

cout<<"The heights are not same "<<endl;

}

return 0;

}

**Output:**

****

**Program no. 36**

**Program to overload memory management operator**

#include<iostream>

#include<stdlib.h>

using namespace std;

class student

{

string name;

int age;

public:

student()

{

cout<< "Constructor is called\n" ;

}

student(string name, int age)

{

this->name = name;

this->age = age;

}

void display()

{

cout<< "Name:" << name << endl;

cout<< "Age:" << age << endl;

}

void \* operator new(size\_t size)

{

cout<< "Overloading new operator with size: " << size << endl;

void \* p = ::new student();

return p;

}

void operator delete(void \* p)

{

cout<< "Overloading delete operator " << endl;

free(p);

}

};

int main()

{

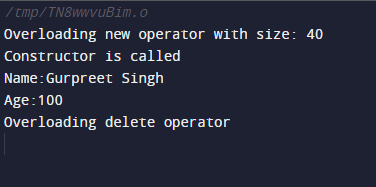
student \* p = new student("Gurpreet Singh", 100);

p->display();

delete p;

}

Output:-



**Program no. 37**

**Program to demonstrate the typecasting of basic data type to class type**

#include<iostream>

using namespace std;

class Time

{

int hrs,min;

public:

Time(int);

void display();

};

Time :: Time(int t)

{

cout<<"Basic Type to ==> Class Type Conversion..."<<endl;

hrs=t/60;

min=t%60;

}

void Time::display()

{

cout<<hrs<< ": Hours(s)" <<endl;

cout<<min<< " Minutes" <<endl;

**}**

int main()

{

int duration;

cout<<"Enter time duration in minutes";

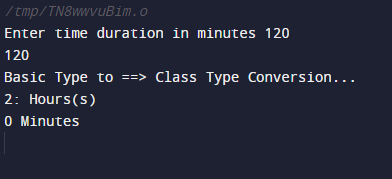
cin>>duration;

Time t1=duration;

t1.display();

}

**Output:**

****

**Program no. 38**

**Program to demonstrate the typecasting of class type to basic data type**

#include<iostream>

using namespace std;

class Time

{

int hrs,min;

public:

// constructor

Time(int a,int b)

{

cout<<"Constructor called with two parameters...\n";

hrs=a;

min=b;

}

// casting operator function

operator int()

{

cout<<"\nClass Type to Basic Type Conversion...";

return(hrs\*60+min);

}

// destructor

~Time()

{

cout<<"\nDestructor called..."<<endl;

}

};

int main()

{

int h,m,duration;

cout<<"\nEnter Hours ";

cin>>h;

cout<<"\nEnter Minutes ";

cin>>m;

Time t(h,m); // construct object

duration = t; // casting conversion OR duration = (int)t

cout<<"\nTotal Minutes are "<<duration;

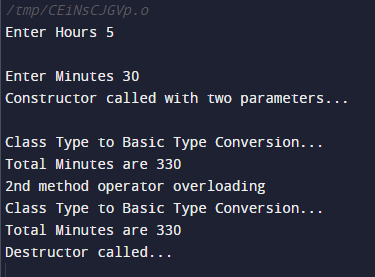
cout<<"\n2nd method operator overloading ";

duration = t.operator int();

cout<<"\nTotal Minutes are "<<duration;

}

Output:



**Program no. 39**

**Program to demonstrate the typecasting of class type to another class type**

#include<iostream>

using namespace std;

class Time

{

int hrs,min;

public:

Time(int h,int m)

{

hrs=h;

min=m;

}

Time()

{

cout<<"\n Time's Object Created";

}

int getMinutes()

{

int tot\_min = ( hrs \* 60 ) + min ;

return tot\_min;

}

void display()

{

cout<<"Hours: "<<hrs<<"\n";

cout<<" Minutes : "<<min <<"\n";

}

};

class Minute

{

int min;

public:

Minute()

{

min = 0;

}

void operator=(Time T)

{

min=T.getMinutes();

}

void display()

{

cout<<"\n Total Minutes : " <<min<<"\n";

}

};

int main()

{

Time t1(6,40);

t1.display();

Minute m1;

m1.display();

//conversion from Time to Minute

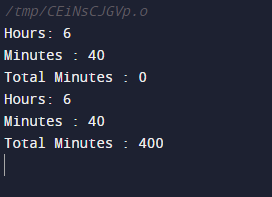
m1 = t1;

t1.display();

m1.display();

}

**Output:**

****

**Program no. 40**

**Program to implement single inheritance**

#include <iostream>

using namespace std;

class A {

public:

A(){

cout<<"Constructor of A class"<<endl;

}

};

class B: public A {

public:

B(){

cout<<"Constructor of B class";

}

};

int main() {

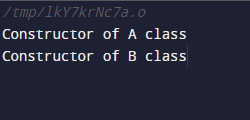
//Creating object of class B

B obj;

return 0;

}

**Output:**

****

**Program no. 41**

**Program to implement multi-level inheritance**

#include <iostream>

using namespace std;

class A {

public:

A(){

cout<<"Constructor of A class"<<endl;

}

};

class B: public A {

public:

B(){

cout<<"Constructor of B class"<<endl;

}

};

class C: public B {

public:

C(){

cout<<"Constructor of C class"<<endl;

}

};

int main() {

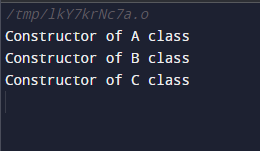
//Creating object of class C

C obj;

return 0;

}

**Output:**

****

**Program no. 42**

**Program to implement multiple inheritance**

#include <iostream>

using namespace std;

class A {

public:

A(){

cout<<"Constructor of A class"<<endl;

}

};

class B {

public:

B(){

cout<<"Constructor of B class"<<endl;

}

};

class C: public A, public B {

public:

C(){

cout<<"Constructor of C class"<<endl;

}

};

int main() {

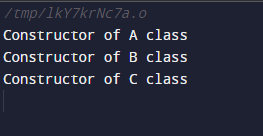
//Creating object of class C

C obj;

return 0;

}

**Output:**

****

**Program no. 43**

**Program to implement Hierarchica**l  **inheritance**

#include <iostream>

using namespace std;

class A {

public:

A(){

cout<<"Constructor of A class"<<endl;

}

};

class B: public A {

public:

B(){

cout<<"Constructor of B class"<<endl;

}

};

class C: public A{

public:

C(){

cout<<"Constructor of C class"<<endl;

}

};

int main() {

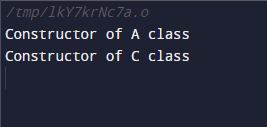
//Creating object of class C

C obj;

return 0;

**}**

**Output**

****

**Program no. 44**

**Program to implement Hybrid**  **inheritance**

**#**include <iostream>

using namespace std;

class A

{

public:

int x;

};

class B : public A

{

public:

B() //constructor to initialize x in base class A

{

x = 40;

}

};

class C

{

public:

int y;

C() //constructor to initialize y

{

y = 8;

}

};

class D : public B, public C //D is derived from class B and class C

{

public:

void sum()

{

cout << "Sum= " << x + y;

}

};

int main()

{

D obj1; //object of derived class D

obj1.sum();

return 0;

}

**Output:-**

****

**Program no. 45**

**Program to resolve ambiguity in multiple inheritance**

#include<iostream>

using namespace std;

class A

{

public:

void show()

{

cout<<"\nClass A"<<endl;

}

};

class B

{

public:

void show()

{

cout<<"\nClass B"<<endl;

}

};

class C:public A,public B

{

//body of class

};

int main ()

{

C obj;

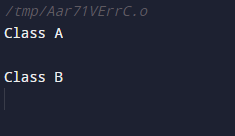
obj.A::show();

obj.B::show();

return 0;

}

**Output:**

****

**Program no. 46**

**Program to show the use of virtual base class**

#include <iostream>

using namespace std;

class A {

public:

int a;

A(){

a = 50;

}

};

class B : public virtual A {

};

class C : public virtual A {

};

class D : public B, public C {

};

int main(){

//creating class D object

D object;

cout << "a = " << object.a << endl;

return 0;

}

**Output:**

****

**Program no. 47**

**Program to show the order of default constructor in inheritance**

#include <iostream>

using namespace std;

// base class

class Parent

{

    public:

    // base class constructor

    Parent()

    {

        cout << "Inside base class" << endl;

    }

};

// sub class

class Child : public Parent

{

    public:

    //sub class constructor

    Child()

    {

        cout << "Inside sub class" << endl;

    }

};

// main function

int main() {

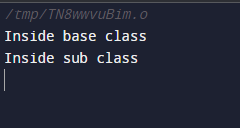
    // creating object of sub class

    Child obj;

    return 0;

}

**Ouput:**

****

**Program no. 48**

**Program to show the order parameterized constructor in inheritance**

#include <iostream>

using namespace std;

// base class

class Parent

{

public:

// base class's parameterised constructor

Parent(int i)

{ int x =i;

cout << "Inside base class's parameterised constructor" << endl;

}

};

// sub class

class Child : public Parent

{

public:

// sub class's parameterised constructor

Child(int j): Parent(j)

{

cout << "Inside sub class's parameterised constructor" << endl;

}

};

// main function

int main() {

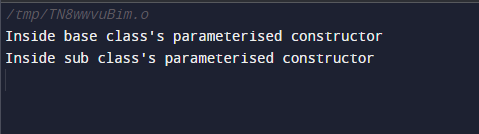
// creating object of class Child

Child obj1(10);

return 0;

}

Output:



**Program no. 49**

**Program to show the order of destructor**

#include<iostream>

using namespace std;

class Base

{

public:

Base()

{

cout<<"Constructor Base \n";

}

~Base()

{

cout<<"Destructor Base \n";

}

};

class Derived: public Base

{

public:

Derived()

{

cout<<"Constructor Derived \n";

}

~Derived()

{

cout<<"Destructor Derived \n";

}

};

int main()

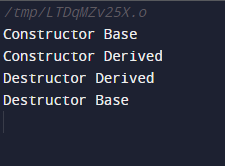
{

Derived d;

return 0;

}

Output:

****

**Program no. 50**

**Program to show the use of composition**

#include <iostream>

using namespace std;

class date

{

int dd,mm,yy;

public:

void read()

{

cout<<"Enter the Date:-"<<endl;

cin>>dd;

cout<<"Enter the month:-"<<endl;

cin>>mm;

cout<<"Enter the year:-"<<endl;

cin>>yy;

}

void show()

{

cout<<"\n Date is:-"<<dd<<"/"<<mm<<"/"<<yy;

}

};

class student

{

int rollno;

char name[20];

date dob;

public:

void read()

{

cout<<"Enter a Roll No:-";

cin>>rollno;

cout<<"Enter a Name:-";

cin>>name;

dob.read();

}

void show()

{

cout<<"\n Rollno="<<rollno<<endl;

cout<<"\n Name="<<name<<endl;

dob.show();

}

};

int main()

{

student s;

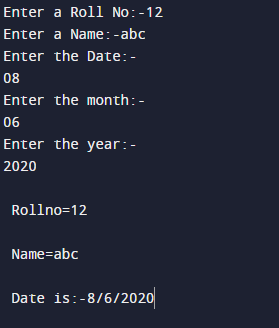
s.read();

s.show();

return 0;

}

**Output:**

****

**Program no. 51**

**Program to implement virtual function**

#include <iostream>

using namespace std;

class BaseClass

{

public:

virtual void Display1()=0;

virtual void Display2()=0;

void Display3()

{

cout<<"\nThis Display3() method of Base Class";

}

};

class DerivedClass:public BaseClass

{

public:

void Display1()

{

cout<<"\nThis Display1() method of Base Class";

}

void Display2()

{

cout<<"\nThis Display2() method of Base Class";

}

};

int main()

{

DerivedClass D;

D.Display1();

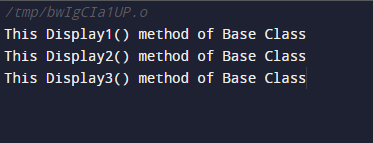
D.Display2();

D.Display3();

return 0;

}

**Output:-**

****

**Program no. 52**

**Program to show the use abstract class pure virtual function**

#include <iostream>

using namespace std;

class BaseClass //abstract class

{

public:

virtual void Display1()=0;

virtual void Display2()=0;

void Display3()

{

cout<<"\n\t This Display3() method of base class";

}

};

class DerivedClass:public BaseClass

{

public:

void Display1()

{

cout<<"\n\t This Display1() method of Derived class";

}

void Display2()

{

cout<<"\n\t This Display2() method of Derived class";

}

};

int main()

{

DerivedClass D;

D.Display1();

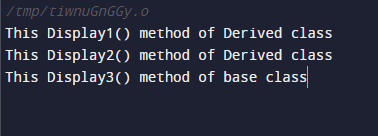
D.Display2();

D.Display3();

return 0;

}

**OUTPUT:**

****

**Program no. 53**

**Program to implement virtual destructor**

#include <iostream>

using namespace std;

class Base

{

public:

Base()

{

cout<<"\nBase class constructor";

}

virtual ~Base()

{

cout<<"\nBase class destructor";

}

};

class Derived:public Base

{

public:

Derived()

{

cout<<"\n derived class constructor";

}

~Derived()

{

cout<<"\n Derived class Destructor";

}

};

int main()

{

Base \*ptr;

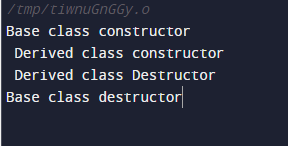
ptr=new Derived ;

delete ptr;

return 0;

}

**OUTPUT:**

****

**Program no. 54**

**Program to show use of this pointer**

#include <iostream>

using namespace std;

class A

{

int num;

char ch;

public:

void Values(int num, char ch)

{

this->num =num;

this->ch=ch;

}

void displayValues()

{

cout<<num<<endl;

cout<<ch;

}

};

int main()

{

A obj;

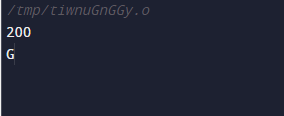
obj.Values(200, 'G');

obj.displayValues();

return 0;

}

**OUTPUT-:**

****

**Program no. 55**

**Program to implement exception handling**

#include<iostream>

using namespace std;

int main ()

{

int a=20,b=0,c;

try

{

if (b==0)

{

throw " Divison by zero is not possible";

}

else

{

c=a/b;

cout<<"value is"<<c;

}

}

catch(const char\* ex)

{

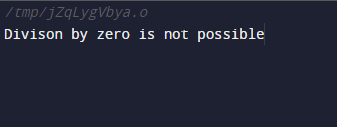
cout<<ex;

}

return 0;

}

**OUTPUT-:**

****

**Program no. 56**

**Program to implement multiple Catch block**

#include<iostream>

using namespace std;

void test(int x)

{

try

{

if(x>0)

{

throw x;

}

else

{

throw 'x';

}

}

catch(int x)

{

cout<<"\n Catch an integer :"<<x;

}

catch (char x)

{

cout<<"\n Catch a character:"<<x;

}

}

int main()

{

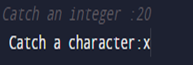
test (20);

test(0);

return 0;

}

**OUTPUT:-**

****

**Program no. 57**

**Program to implement Catch all exception**

#include<iostream>

using namespace std;

void test(int x)

{

try

{

if(x>0)

{

throw x;

}

else

{

throw 'x';

}

}

catch(...)

{

cout<<"\n Exception";

}

}

int main()

{

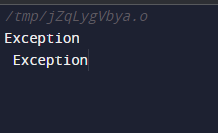
test (30);

test(0);

return 0;

}

**Output:-**

****

**Program no. 58**

**Program to implement Rethrow an exception**

#include<iostream>

using namespace std;

void Demo()

{

try

{

throw 10;

}

catch(int i)

{

cout<<"\n Caught an exception inside function";

throw;

}

}

int main()

{

try

{

Demo();

}

catch(int i)

{

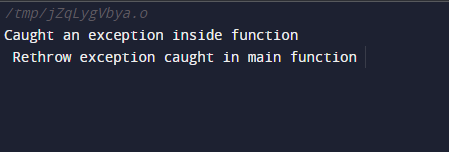
cout<<"\n Rethrow exception caught in main function ";

}

return 0;

}

**Output:-**

****

**Program no. 59**

**Program to implement Exception specification**

#include <iostream>

using namespace std;

void test(int val) throw(int,char)

{

if (val==0) //1= = 0 f

throw 'a';

else

throw 10;

}

int main()

{

try

{

test(1);

}

catch(int)

{

cout<<"\n Int type Exception";

}

catch(char)

{

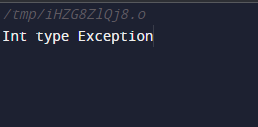
cout<<"\n Char type Exception";

}

return 0;

}

**Output-:**

****

**Program no. 60**

**Program to implement a template**

#include <iostream>

using namespace std;

template<class T>

T add (T a,T b)

{

T c;

c=a+b;

return c;

}

int main()

{

int x=20,y=20;

float d=10.5,f=5.5;

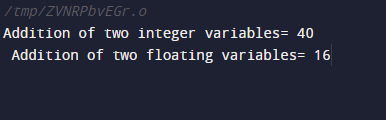
cout<<"\n Addition of two integer variables= "<<add(x,y);

cout<<"\n Addition of two floating variables= "<<add(d,f);

return 0;

}

**Output:**

****

**Program no. 61**

**Program to use function template with more than one generic type**

#include <iostream>

using namespace std;

template <class T1 , class T2>

void display(T1 a, T2 b)

{

cout<<"\na:"<<a<<"\n b:"<<b;

}

int main ()

{

int i =5;

float f=20.5;

display(i,f);

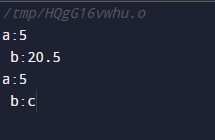
char y='c';

display(i,y);

return 0;

}

**Output:**

****

**Program no. 62**

**Program to overload function template**

#include <iostream>

using namespace std;

template<class X>

void fun(X a)

{

cout << "Value of a is : " <<a<<endl;

}

template<class X,class Y>

void fun(X b ,Y c)

{

cout << "Value of b is : " <<b<< endl;

cout << "Value of c is : " <<c<< endl;

}

int main()

{

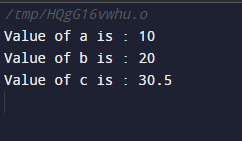
fun(10);

fun(20,30.5);

return 0;

}

**Output:**

****

**Program no. 63**

**Program to show use of class template**

#include <iostream>

        #include <string>

        using namespace std;

        template <class T>

        class Stack

        {

        public:

                Stack();

                void push(T i);

                T pop();

        private:

                int top;

        T st[100];

        };

        template <class T>

        Stack<T>::Stack()

        {

                top = -1;

        }

        template <class T>

        void Stack<T>::push(T i)

        {

                st[++top] = i;

        }

        template <class T>

        T Stack<T>::pop()

        {

                return st[top--];

        }

        int main ()

        {

                Stack<int> int\_stack;

                Stack<string> str\_stack;

                int\_stack.push(67);

                str\_stack.push("Hello");

                str\_stack.push("Codezclub");

                cout << int\_stack.pop() << endl;

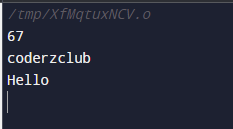
                cout << str\_stack.pop() << endl;

                cout << str\_stack.pop() << endl;

                return 0;

        }

**Output**

****

**Program no. 64**

**Program to open and close a file**

#include<iostream>

#include<fstream>

using namespace std;

int main(){

fstream obj("my\_file");

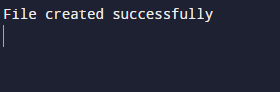
cout<<"File created successfully"<<endl;

obj.close();

return 0;

}

**Output:-**

****

**Program no. 65**

**Program to write content in the file**

#include<iostream>

#include<fstream>

using namespace std;

int main(){

string name;

int roll;

cout<<"Enter name and roll number"<<endl;

cin>>name>>roll;

ofstream obj("my\_file");

obj<<name<<endl;

obj<<roll<<endl;

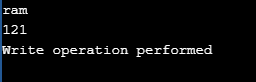
cout<<"Write operation performed"<<endl;

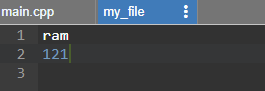
obj.close();

return 0;

}

**Output:-**

****

****

**Program no. 66**

**Program to read the contents of a file**

#include<iostream>

#include<fstream>

using namespace std;

int main(){

ifstream obj("my\_file");

if(!obj){

cerr<<"Cannot open the file"<<endl;

exit(0);

} else{

string name;

int roll\_no;

obj>>name;

obj>>roll\_no;

cout<<"The details in the file are"<<endl;

cout<<name<<endl;

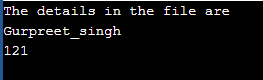
cout<<roll\_no<<endl;

}

return 0;

}

Output:-



**Program no. 67**

**Program to read the content of the file using get function**

#include<iostream>

#include<fstream>

using namespace std;

int main(){

ifstream obj("my\_file");

if(!obj){

cerr<<"Cannot open the file"<<endl;

exit(0);

} else {

char ch;

cout << "The details in the file are" << endl;

while (obj.eof() == 0) {

obj.get(ch);

cout << ch;

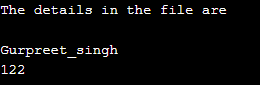
}

}

return 0;

}

**Output:-**

****

**Program no. 68**

**Program to find the size of a file**

#include<iostream>

#include<fstream>

using namespace std;

int main(){

fstream obj("my\_file");

if(!obj){

cerr<<"Cannot open the file"<<endl;

exit(0);

} else {

obj.seekg(0,ios::end);

cout<<"The size of the file is "<<endl;

cout<<obj.tellg();

}

return 0;

}

**Output:-**

****