**1.a .Sum using while loop**

#include <iostream>

using namespace std;

int main() {

int num,rem,sum=0;

cout<<"enter the number:";

cin>>num;

while(num>0){

rem=num%10;

sum+=rem;

num=num/10;

}

cout<<"\n sum of digits="<<sum;

cout<<endl;

return 0;

}

**1.b. stack using class.**

#include<iostream>

#define MAX\_SIZE 5

using namespace std;

class stack{

private:int item, i;

int arr\_stack[MAX\_SIZE];

int top;

public:

stack(){

top=0;

}

void push(){

if(top==MAX\_SIZE)

cout<<"\n##Stack is full!";

else{

cout<<"\nEnter the value to be pushed:";

cin>>item;

cout<<"\n##Position:"<<top<<"'Pushed Value:"<<item;

arr\_stack[top++]=item;

}

};

void pop(){

if (top==0)

cout<<"\n##Stack is Empty!";

else{

--top;

cout<<"\n##Position:"<<top<<"'Popped Value:"<<arr\_stack[top];

}}

void display(){

cout<<"\n#3Stack Size:"<<top;

for(i=(top-1);i>=0;i--)

cout<<"\n##Position:"<<i<<"'Value:"<<arr\_stack[i];

}};

int main(){

int choice,exit\_p=1;

stack obj;

cout<<"\nSimple Stack example-class and member functions-C++";

do{

cout<<"\n\nStack Main Menu";

cout<<"\n1.Push\n2.Pop\n3.Display\nOthers to exit";

cout<<"\nEnter Your Choice:";

cin>>choice;

switch(choice){

case 1:

obj.push();

break;

case 2:

obj.pop();

break;

case 3:

obj.display();

break;

default:

exit\_p=0;

break;

}

}while(exit\_p);

return 0;

}

**2.a.Sum of digits using for loop**

#include <iostream>

using namespace std;

int main() {

int num,rem,sum=0;

cout<<"enter the digits:-";

cin>>num;

for(sum=0;num>0;num=num/10){

rem=num%10;

sum+=rem;

}

cout<<"\n sum of two digits:-"<<sum;

cout<<endl;

return 0;

}

**2.b.Multiple inheritance**

#include <iostream>

using namespace std;

class m{

protected:

int m;

public:

void get\_m(int);

};

class n{

protected:

int n;

public:

void get\_n(int);

};

class p:public m,public n{

public:

void display(void);

};

void m::get\_m(int x){

m=x;

}

void n::get\_n(int y){

n=y;

}

void p:: display(void){

cout<<"m="<<m<<"\n";

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

cout<<"m\*n="<<m\*n<<"\n";

}

int main() {

p p;

p.get\_m(10);

p.get\_n(20);

p.display();

return 0;

}

**3.a.Sum of digits user defined function**

#include <iostream>

using namespace std;

int sum\_of\_digits(int n){

int sum=0;

while(n!=0){

sum+=n%10;

n/=10;

}

return sum;

}

int main() {

int n;

cout<<"enter the digits:-";

cin>>n;

cout<<"\n sum of two digits:-"<<sum\_of\_digits(n);

cout<<endl;

return 0;

}

**3.b.Display 2 number constructor**

#include <iostream>

using namespace std;

class point {

private:

int x,y;

public:

point(int x1,int y1){

x=x1;

y=y1;

}

point(const point&p1){

x=p1.x;

y=p1.y;

}

int getx(){

return x;

}

int gety(){

return y;

}

};

int main() {

point p1(10,15);

point p2=p1;

cout<<"p1.x="<<p1.getx()<<",p1.y="<<p1.gety();

cout<<"\n";

cout<<"p2.x="<<p2.getx()<<",p2.y="<<p2.gety();

return 0;

}

**4.a.Demonstrate and working of inline function**

#include <iostream>

using namespace std;

inline int area (int length,int breadth){

return length \* breadth;

}

int main() {

int length ,breadth,result;

cout<<"enter the length of the rectangle;";

cin>>length;

cout<<"enter the breadth of the rectangle;";

cin>>breadth;

result=area(length,breadth);

cout<<"\n the area of the rectangle is"<<result;

cout<<"\n\n";

return 0;

}

**4.b.Create student object**

#include<iostream>

using namespace std;

#define MAX 10

class student

{

private:

char name[30];

int rollno;

int total;

float perc;

public:

//member function to get student's details

void getDetails(void);

//member function to print student's details

void putDetails(void);

};

void student::getDetails(void){

cout<<"Enter name:";

cin>>name;

cout<<"Enter roll number:";

cin>>rollno;

cout<<"Enter total marks out of 500:";

cin>>total;

perc=(float)total/500\*100;

}

void student::putDetails(void){

cout<<"Student Deatails:\n";

cout<<"Name:"<<name<<",Roll Number:"<<rollno<<",Total:"<<total<<",Percentage:"<<perc;

}

int main()

{

student std[MAX]; //array of objects creation

int n, loop;

cout<<"Enter total number of students:";

cin>>n;

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

cout<<"Enter details of student"<<loop+1<<":\n";

std[loop].getDetails();

}

cout<<endl;

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

cout<<"Details of student"<<(loop+1)<<":\n";

std[loop].putDetails();

}

return 0;

}

**5.a.Sum of digit using class**

#include <iostream>

using namespace std;

class name{

private:

int sum,num,res;

public:

void getnumber();

int findsumofdigit();

};

void name::getnumber(){

cout<<"enter the number:";

cin>>num;

}

int name::findsumofdigit(){

sum=0;

while(num>0){

res=num%10;

sum+=res;

num/=10;

}

return sum;

}

int main(){

name c;

int sum;

c.getnumber();

sum=c.findsumofdigit();

cout<<"\n sum of digits="<<sum;

return 0;}

**5.b.destructor**

#include <iostream>

using namespace std;

int count=0;

class Test

{

public:

Test(){

count++;

cout<<"\n no of object ceated:"<<count;

}

~Test(){

cout<<"\n no of object destroyed:"<<count;

--count;

}

};

int main() {

Test t,t1,t2,t3;

return 0;

}

**6.a.Friend function**

#include <iostream>

using namespace std;

class Distance {

private:

int meter;

// friend function

friend int addFive(Distance);

public:

Distance() : meter(0) {}

};

// friend function definition

int addFive(Distance d) {

//accessing private members from the friend function

d.meter += 5;

return d.meter;

}

int main() {

Distance D;

cout << "Distance: " << addFive(D);

return 0;

}

**7.a.Demonstrate of inline function**

#include <iostream>

using namespace std;

inline void displayNum(int num)

{

cout<<num<<endl;

}

int main() {

displayNum(5);

displayNum(8);

displayNum(666);

return 0;

}

**7.b.marks add**

#include <iostream>

using namespace std;

class student{

protected:

int roll\_number;

public:

void get\_number(int);

void put\_number(void);

};

void student :: get\_number(int a){

roll\_number=a;

}

void student :: put\_number(){

cout<<"roll number:"<<roll\_number<<"\n";

}

class test : public student{

protected:

float sub1;

float sub2;

public:

void get\_marks(float,float);

void put\_marks(void);

};

void test :: get\_marks(float x,float y){

sub1=x;

sub2=y;

}

void test :: put\_marks(){

cout<<"marks in sub1"<<sub1<<"\n";

cout<<"marks in sub2"<<sub2<<"\n";

}

class result : public test{

float total;

public:

void display(void);

};

void result :: display(void){

total =sub1+sub2;

put\_number();

put\_marks();

cout<<"total="<<total<<"\n";

}

int main() {

result student1;

student1.get\_number(111);

student1.get\_marks(75.0,59.5);

student1.display();

return 0;

}

**8.a.Sum of digit using class**

#include <iostream>

using namespace std;

class name{

private:

int sum,num,res;

public:

void getnumber();

int findsumofdigit();

};

void name::getnumber(){

cout<<"enter the number:";

cin>>num;

}

int name::findsumofdigit(){

sum=0;

while(num>0){

res=num%10;

sum+=res;

num/=10;

}

return sum;

}

int main(){

name c;

int sum;

c.getnumber();

sum=c.findsumofdigit();

cout<<"\n sum of digits="<<sum;

return 0;

}

**8.b.Illustrate operator overloading**

#include <iostream>

using namespace std;

class count{

private:

int value;

public:

count():value(5){}

void operator++(){

++value;

}

void display(){

cout<<"count:"<<value<<endl;

}

};

int main() {

count count1;

++count1;

count1.display();

return 0;

}

**9.a.rectangle area**

#include <iostream>

using namespace std;

inline int area (int length,int breadth){

return length \* breadth;

}

int main() {

int length ,breadth,result;

cout<<"enter the length of the rectangle;";

cin>>length;

cout<<"enter the breadth of the rectangle;";

cin>>breadth;

result=area(length,breadth);

cout<<"\n the area of the rectangle is"<<result;

cout<<"\n\n";

return 0;

}

**9.b.paramaterised contructor**

#include <iostream>

using namespace std;

class point {

private:

int x,y;

public:

point(int x1,int y1){

x=x1;

y=y1;

}

int getx(){

return x;

}

int gety(){

return y;

}

};

int main() {

point p1(10,15);

cout<<"p1.x="<<p1.getx()<<",p1.y="<<p1.gety();

return 0;

}

**10.a.default constructor**

#include <iostream>

using namespace std;

class construct

{

public:

int a,b;

construct()

{

a=10;

b=20;

}

};

int main()

{

construct c;

cout<<"a="<<c.a<<endl<<"b="<<c.b;

return 1;

}

**10.b.illustrate function overloading**

#include <iostream>

using namespace std;

float absolute (float var){

if(var<0.0)

var=-var;

return var;

}

int absolute(int var){

if(var<0)

var=-var;

return var;

}

int main() {

cout<<"absolute value of -5="<<absolute(-5)<<endl;

cout<<"absolute value of5.5="<<absolute(5.5f)<<endl;

return 0;

}

**11.a.sum of two digit while loop**

#include <iostream>

using namespace std;

int main() {

int num,rem,sum=0;

cout<<"enter the number:";

cin>>num;

while(num>0){

rem=num%10;

sum+=rem;

num=num/10;

}

cout<<"\n sum of digits="<<sum;

cout<<endl;

return 0;

}

**11.b.multiple inheritance**

#include <iostream>

using namespace std;

class m{

protected:

int m;

public:

void get\_m(int);

};

class n{

protected:

int n;

public:

void get\_n(int);

};

class p:public m,public n{

public:

void display(void);

};

void m::get\_m(int x){

m=x;

}

void n::get\_n(int y){

n=y;

}

void p:: display(void){

cout<<"m="<<m<<"\n";

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

cout<<"m\*n="<<m\*n<<"\n";

}

int main() {

p p;

p.get\_m(10);

p.get\_n(20);

p.display();

return 0;

}

**12.a.demostrate concept of friend function**

#include <iostream>

using namespace std;

class Distance {

private:

int meter;

// friend function

friend int addFive(Distance);

public:

Distance() : meter(0) {}

};

// friend function definition

int addFive(Distance d) {

//accessing private members from the friend function

d.meter += 5;

return d.meter;

}

int main() {

Distance D;

cout << "Distance: " << addFive(D);

return 0;

}

**12.b.illustrate operator overloading**

#include <iostream>

using namespace std;

class count{

private:

int value;

public:

count():value(5){}

void operator++(){

++value;

}

void display(){

cout<<"count:"<<value<<endl;

}

};

int main() {

count count1;

++count1;

count1.display();

return 0;

}