***Getting Started with C++***

**1.** **Program of Syntax in CPP**

#include<iostream>

using namespace std;

int main()

{

int a=12; //variable declaration and definition.

cout<<a; //printing in the cosole.

cout<<endl<<"hello world";

return 0;

}

**Output:**

12

hello world

**2. Operators program in CPP**

#include<iostream>

using namespace std;

int main()

{

int a=12; int b=20;

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

cout<<"subraction result of a from b is: "<<b-a<<endl;

cout<<"the multiplication of a and b is: "<<a\*b<<endl;

cout<<" the divide result of the a from b is: "<<b/a<<endl;

return 0;

}

**Output:**

the sum of a and b is: 32

subraction result of a from b is: 8

the multiplication of a and b is: 240

the divide result of the a from b is: 1

**3. Functions program in CPP**

#include<iostream>

using namespace std;

int sum(int a, int b) { return a+b;}

int main()

{

cout<<"the sum of 12 and 15 is: "<<sum(12,15)<<endl;

return 0;

}

Output:

the sum of 12 and 15 is: 27

**4. Pointers program in CPP**

#include <iostream>

using namespace std;

int main()

{

// pointer arithmetic operation

int marks[] = {12, 34, 23, 12, 34};

int \*mpointer = marks;

cout << \*mpointer << endl<< endl;

mpointer++;

cout << \*mpointer << endl

<< endl;

int a = 12;

int \*b;

b = &a;

cout << \*b<<endl;

//\* is known as the dereference operator

// & is known as the address of operator

// pointer to pointer

int \*\*c = &b;

cout << \*\*c;

int \*\*\*d = &c;

return 0;

}

**Output:**

12

34

12

12

**5. Recursion program in CPP**

#include<iostream>

using namespace std;

int factorial(int a)

{

if (a==1)

{

return 1;

}

else{

return (a\*factorial(a-1));

}

}

int main()

{

cout<<"the factorial of 5 is: "<<factorial(5);

return 0;

}

**Output:**

the factorial of 5 is: 120

**6. Arrays program in CPP**

#include<iostream>

using namespace std;

int main()

{

int a[]={12,12,14,45,654,23,2131,432}; //syntax to declare the array in cpp

for (int i = 0; i < 8; i++)

{

cout<<a[i]<<"\t";

}

return 0;

}

**Output:**

12 12 14 45 654 23 2131 432

**7. Structures program in CPP**

#include<iostream>

using namespace std;

struct studentRecored

{

int id;

string name;

}s1;

int main()

{

cout<<"Enter the id and the name of the student: ";

cin>>s1.id>>s1.name;

cout<<"id of the student is: "<<s1.id<<endl;

cout<<"name of the student is: "<<s1.name<<endl;

return 0;

}

**Output:**

Enter the id and the name of the student: 12 Shubham

id of the student is: 12

name of the student is: Shubham

***Classes & Objects***

**8. Program of Concept of class in CPP**

#include<iostream>

using namespace std;

class student {

public:

int id;

string name;

public:

void getdata()

{

cout<<"Enter the id and name of the student.";

cin>>id>>name;

}

void display(){

cout<<"id of the student is "<<id<<" and name of the student is "<<name<<endl;

}

};

int main()

{

student st1;

st1.getdata();

st1.display();

return 0;

}

**Output:**

Enter the id and name of the student.1 Shubham\_Dahiya

id of the student is 1 and name of the student is Shubham\_Dahiya

**9. This pointer program in cpp**

#include<iostream>

using namespace std;

class calculator {

public:

int a;

int b;

int sum(int a, int b)

{

this->a = a;

this->b = b;

return a+b;

}

};

int main()

{

calculator c1;

cout<<"the sum of 12 and 12 is: "<<c1.sum(12,12);

return 0;

}

**Output:**

the sum of 12 and 12 is: 24

**10. Function Overloading program in cpp**

#include <iostream>

using namespace std;

int add(int a, int b)

{

return a + b;

}

int add(int a, int b, int c)

{

return a + b + c;

}

int main()

{ int a = 12;

int b = 12;

int c = 12;

cout << add(a, b) << endl;

cout << add(a, b, c);

return 0;

}

**Output:**

24

36

**11. Constructor and destructor program in cpp**

#include<iostream>

using namespace std;

class greet{

int a;

public:

greet()

{

cout<<"constructor method"<<endl;;

}

~greet() {

cout<<" destructor method"<<endl;

}

};

int main()

{

greet g1;

return 0;

}

**Output:**

constructor method

destructor method

**12. Default value function program in cpp**

#include<iostream>

using namespace std;

int sum(int a, int b=12)

{

return a + b;

}

int main()

{

cout<<"the sum of a and b is: "<<sum(23);

return 0;

}

**Output:**

the sum of a and b is: 35

**13. Dynamic memory allocation program in CPP**

#include<iostream>

using namespace std;

int main()

{

cout<<"enter the size of array";

int size;

cin>>size;

int \*a=new int[size];

cout<<"enter the elements of array";

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

{

cin>>a[i];

}

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

{

cout<<a[i]<<"\t";

}

return 0;

}

**Output:**

enter the size of array3

enter the elements of array12 12 12

12 12 12

**14. Static members program in CPP**

#include<iostream>

using namespace std;

class shop

{

int id;

int itemPrice;

static int totalPrice;

public:

void getData(int id,int price)

{

this->id=id;

this->itemPrice=price;

this->totalPrice=totalPrice+itemPrice;

}

void displayData()

{

cout<<"the id of the item is: "<<this->id<<endl;

cout<<"the price of the item is:"<<this->itemPrice<<endl;

cout<<"the total price of all the items is: "<<this->totalPrice<<endl;

}

};

int shop ::totalPrice=0;

int main()

{

shop item1;

shop item2;

item1.getData(1,200);

item1.displayData();

item2.getData(1,200);

item2.displayData();

return 0;

}

**Output:**

the id of the item is: 1

the price of the item is:200

the total price of all the items is: 200

the id of the item is: 1

the price of the item is:200

the total price of all the items is: 400

**15. Inheritance program in CPP**

#include<iostream>

using namespace std;

class base

{

public:

void display()

{

cout<<"function from the base class.";

}

int sum( int a,int b)

{

return a +b;

}

};

class derived:public base

{

};

int main()

{

derived d1;

d1.display();

cout<<endl<<"the sum of 12 and 12 is: "<<d1.sum(12,12);

return 0;

}

**Output:**

function from the base class.

the sum of 12 and 12 is: 24

**16. Method overriding program in CPP**

#include<iostream>

using namespace std;

class base

{

public:

virtual void display()

{

cout<<"function from the base class.";

}

int sum( int a,int b)

{

return a +b;

}

};

class derived:public base

{

public:

void display ()

{

cout<<"function from the derived class.";

}

};

int main()

{

base b1;

derived \*d1;

d1= (derived \*)&b1;

d1->display();

return 0;

}

**Output:**

function from the base class.

**17. Abstract class program in CPP**

#include <iostream>

using namespace std;

class base

{

virtual void display();

};

class derived : public base

{

public:

void display() { cout << "definition of the virtual function of the base class in derived class."; }

};

int main()

{

derived d1;

d1.display();

return 0;

}

**Output:**

redefinition of the virtual function of the base class in derived class.

***Inheritance***

**18. Single inheritance**

#include <iostream>

using namespace std;

class Animal {

public:

void fun1() {

cout<<"I am an animal"<<endl;

}

};

class Dog : public Animal {

public:

void fun2() {

cout<<"I am a dog"<<endl;

}

};

int main() {

Dog obj;

obj.fun1();

obj.fun2();

return 0;

}

**Output:**

I am an animal  
I am a dog

**19. Multiple inheritance**

#include <iostream>

using namespace std;

class A {

protected:

int a;

public:

void seta(int x) {

a = x;

}

};

class B {

protected:

int b;

public:

void setb(int y) {

b = y;

}

};

class C : public A, public B {

public:

int add() {

cout<<"Addition of two numbers = "<<a+b;

}

};

int main() {

C obj;

obj.seta(4);

obj.setb(9);

obj.add();

return 0;

}

**Output:**

Addition of two numbers = 13

**20. Multilevel inheritance**

#include <iostream>

using namespace std;

class Animal {

public:

void fun1() {

cout<<"Animal"<<endl;

}

};

class PetAnimal : public Animal {

public:

void fun2() {

cout<<"Pet animal"<<endl;

}

};

class Dog : public PetAnimal {

public:

void fun3() {

fun1();

fun2();

cout<<"Dog"<<endl;

}

};

int main() {

Dog obj;

obj.fun3();

return 0;

}

**Output:**

Animal  
Pet animal  
Dog

**21. Hierarchial inheritance**

#include <iostream>

using namespace std;

class Values {

protected:

double a, b;

public:

void initialize(double x, double y) {

a = x;

b = y;

}

};

class A : public Values {

public:

void add() {

cout<<"addition = "<<a+b<<endl;

}

};

class B : public Values {

public:

void subtract() {

cout<<"subtraction = "<<a-b<<endl;

}

};

int main() {

A obj1;

B obj2;

obj1.initialize(4.5,8.7);

obj1.add();

obj2.initialize(3.6,11);

obj2.subtract();

return 0;

}

**Output:**

addition = 13.2  
subtraction = -7.4

Salary: 60000

Bonus: 5000

**22. Hybrid inheritance**

#include <iostream>

using namespace std;

class A {

protected:

float a;

public:

void seta(float n1) {

a = n1;

}

};

class B : public A {

public:

void modifyA() {

a/=2;

}

};

class C {

protected:

float c;

public:

void setc(float n2) {

c = n2;

}

};

class D : public B, public C {

public:

float modify() {

modifyA();

cout<<"Result = "<<a\*c;

}

};

int main() {

D obj;

obj.seta(15.6);

obj.setc(9.7);

obj.modify();

return 0;

}

**Output:**

Result = 75.66

**23. Friend function program in CPP**

#include<iostream>

using namespace std;

class greet

{

public:

friend void display();

};

void display(){

cout<<"I am the friend function.";

}

int main()

{

greet g1;

display();

return 0;

}

**Output:**

I am the friend function.

**24.** **Method overloading program in CPP**

#include<iostream>

using namespace std;

class calculator { public:

int sum(int a,int b) {return a+b;}

int sum(int a,int b,int c) {return a+b+c;}

};

int main()

{

calculator c1;

cout<<c1.sum(1,1)<<endl;

cout<<c1.sum(2,2,2);

return 0;

}

**Output:**

2

***Polymorphism***

**25. Runtime polymorphism with two derived classes**

#include <iostream>

using namespace std;

class Shape {                                        //  base class

    public:

virtual void draw(){                             // virtual function

cout<<"drawing..."<<endl;

    }

};

class Rectangle: public Shape                  //  inheriting Shape class.

{

 public:

 void draw()

   {

       cout<<"drawing rectangle..."<<endl;

    }

};

class Circle: public Shape                        //  inheriting Shape class.

{

 public:

 void draw()

   {

      cout<<"drawing circle..."<<endl;

   }

};

int main(void) {

    Shape \*s;                               //  base class pointer.

    Shape sh;                               // base class object.

       Rectangle rec;

        Circle cir;

      s=&sh;

     s->draw();

        s=&rec;

     s->draw();

    s=?

    s->draw();

}

**Output:**

drawing...

drawing rectangle...

drawing circle...

**26. Runtime polymorphism with data members**

#include <iostream>

using namespace std;

class Animal {                                          //  base class declaration.

    public:

    string color = "Black";

};

class Dog: public Animal                       // inheriting Animal class.

{

 public:

    string color = "Grey";

};

int main(void) {

     Animal d= Dog();

    cout<<d.color;

}

**Output:**

Black

**27. Operator overloading program in CPP**

#include <iostream>

using namespace std;

class Test

{

   private:

      int num;

   public:

       Test(): num(8){}

       void operator ++()         {

          num = num+2;

       }

       void Print() {

           cout<<"The Count is: "<<num;

       }

};

int main()

{

    Test tt;

    ++tt;  // calling of a function "void operator ++()"

    tt.Print();

    return 0;

}

**Output:**

The Count is: 10

**28. Operator overloading binary operators.**

#include <iostream>

using namespace std;

class A

{

    int x;

      public:

      A(){}

    A(int i)

    {

       x=i;

    }

    void operator+(A);

    void display();

};

void A :: operator+(A a)

{

    int m = x+a.x;

    cout<<"The result of the addition of two objects is : "<<m;

}

int main()

{

    A a1(5);

    A a2(4);

    a1+a2;

    return 0;

}

**Output:**

The result of the addition of two objects is : 9

**29. Exception Handling**

#include <iostream>

#include<conio>

using namespace std;

int main()

{

   int x = -1;

   // Some code

   cout << "Before try \n";

   try {

      cout << "Inside try \n";

      if (x < 0)

      {

         throw x;

         cout << "After throw (Never executed) \n";

      }

   }

   catch (int x ) {

      cout << "Exception Caught \n";

   }

   cout << "After catch (Will be executed) \n";

   return 0;

}

**OUTPUT:**

Before try

Inside try

Exception Caught

After catch (Will be executed)

**30. Program to add two numbers using function templates:**

#include <iostream>

using namespace std;

template <typename T>

T add(T num1, T num2) {

return (num1 + num2);

}

int main() {

int result1;

double result2;

// calling with int parameters

result1 = add<int>(2, 3);

cout << "2 + 3 = " << result1 << endl;

// calling with double parameters

result2 = add<double>(2.2, 3.3);

cout << "2.2 + 3.3 = " << result2 << endl;

return 0;

}

**OUTPUT:**

2 + 3 = 5

2.2 + 3.3 = 5.5

**31. Simple calculator using class Templates**

#include <iostream>

using namespace std;

template <class T>

class Calculator {

private:

T num1, num2;

public:

Calculator(T n1, T n2) {

num1 = n1;

num2 = n2;

}

void displayResult() {

cout << "Numbers: " << num1 << " and " << num2 << "." << endl;

cout << num1 << " + " << num2 << " = " << add() << endl;

cout << num1 << " - " << num2 << " = " << subtract() << endl;

cout << num1 << " \* " << num2 << " = " << multiply() << endl;

cout << num1 << " / " << num2 << " = " << divide() << endl;

}

T add() { return num1 + num2; }

T subtract() { return num1 - num2; }

T multiply() { return num1 \* num2; }

T divide() { return num1 / num2; }

};

int main() {

Calculator<int> intCalc(2, 1);

Calculator<float> floatCalc(2.4, 1.2);

cout << "Int results:" << endl;

intCalc.displayResult();

cout << endl

<< "Float results:" << endl;

floatCalc.displayResult();

return 0;

}

**OUTPUT:**

Int results:

Numbers: 2 and 1.

2 + 1 = 3

2 - 1 = 1

2 \* 1 = 2

2 / 1 = 2

Float results:

Numbers: 2.4 and 1.2.

2.4 + 1.2 = 3.6

2.4 - 1.2 = 1.2

2.4 \* 1.2 = 2.88

2.4 / 1.2 = 2

**32. Template overloading program to overload square of different parameters**

#include <iostream>

#include <conio.h>

using namespace std;

template<class t1>

void sum(t1 a,t1 b,t1 c)

{

cout<<"Template function 1: Sum = "<<a+b+c<<endl;

}

template <class t1,class t2>

void sum(t1 a,t1 b,t2 c)

{

cout<<"Template function 2: Sum = "<<a+b+c<<endl;

}

void sum(int a,int b)

{

cout<<"Normal function: Sum = "<<a+b<<endl;

}

int main()

{

int a,b;

float x,y,z;

cout<<"Enter two integer data: ";

cin>>a>>b;

cout<<"Enter three float data: ";

cin>>x>>y>>z;

sum(x,y,z); // calls first template function

sum(a,b,z); // calls first template function

sum(a,b); // calls normal function

getch();

return 0;

}

**OUTPUT:**

Enter two integer data: 5 9

Enter three float data: 2.3 5.6 9.5

Template function 1: Sum = 17.4

Template function 2: Sum = 23.5

Normal function: Sum = 14