Wednesday, 19 January 2022

Q1 – Declare a class employee.

Solution - ../Programs/00\_Class.cpp

*// Declare a class for Employees*

#include <iostream>

using namespace std ;

class *Employee* {

    int id ;

    char name[30] ;

    float ctc ;

} ;

int main () {

*Employee* roy, agr ;

    return 0;

}

Thursday, 20 January 2022

Q1 Program to find the largest among three integer.

Solution - ../Program/04\_Practice.cpp

#include <iostream>

using namespace std;

int max(int a, int b, int c)

{

    if (a > b && a > c)

    {

        return a;

    }

    else if (b > a && b > c)

    {

        return b;

    }

    else

    {

        return c;

    }

}

int main()

{

    int a, b, c;

    cout << "Enter a, b, c " << endl;

    cin >> a >> b >> c;

    cout << "Largest number is " << max(a, b, c) << endl;

    return 0;

}

Q2 Write a Program to find the sum of natural numbers.

Solution – ../Program/04\_Practice.cpp

#include <iostream>

using namespace std;

int sumN(int n)

{

    if (n == 0)

    {

        return 0;

    }

    else

    {

        return n + sumN(n - 1);

    }

}

int main()

{

    int n;

    cout << "Enter no. of terms " << endl;

    cin >> n;

    cout << "Sum of " << n << " natural number is " << sumN(n) << endl;

    return 0;

}

Q3 WAP to check if a number is prime or not.

Solution – ../Program/04\_Practice.cpp

#include <iostream>

using namespace std;

int isPrime(int a)

{

    if (a == 1)

    {

        return 0;

    }

    for (int i = 2; i <= a / 2; i++)

    {

        if (a % i == 0)

        {

            return 0;

        }

    }

    return 1;

}

int main()

{

    int n;

    cout << "Enter Number " << endl;

    cin >> n;

    cout << "Isprime : " << isPrime(n) << endl;

    return 0;

}

Q4 WAP to display the Fibonacci series.

Solution – ../Program/04\_Practice.cpp

#include <iostream>

using namespace std;

int Fibonnaciterm(int n)

{

    if (n == 1)

    {

        return 0;

    }

    else if (n == 2)

    {

        return 1;

    }

    else

    {

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

    }

}

int main()

{

    int n;

    cout << "Enter n " << endl;

    cin >> n;

    for (int i = 1; i <= n; i++)

    {

        cout << Fibonnaciterm(i) << " "  ;

    }

    return 0;

}

Q5 – What is a class?

Answer – A class is a collection of different data and functions.

Q7 – What is an object?

Answer - An object is an instance of class referring to a real entity.

Friday, 21 January 2022

Q1 – Class to represent the details of a n students.

Solution –

#include <iostream>

using namespace std;

class student

{

private:

int roll;

float percentage;

public:

void input();

void output();

// char grade(float a);

char grade();

};

void student ::input()

{

cout << "Enter roll " << endl;

cin >> roll;

cout << "Enter Percentage " << endl;

cin >> percentage;

}

char student ::grade()

{

if (percentage >= 90)

{

return 'A';

}

else if (percentage >= 80)

{

return 'B';

}

else if (percentage >= 70)

{

return 'C';

}

else

{

return 'D';

}

}

void student ::output()

{

cout << "Roll - " << roll << " Percentage - " << percentage << endl;

cout << "Grade - " << this->grade() << endl;

}

int main()

{

int n;

cout << "Enter the no. of students " << endl;

cin >> n;

student S[n];

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

{

cout << "Student " << i + 1 << " : " << endl;

S[i].input();

}

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

{

cout << "Student " << i + 1 << " : " << endl;

S[i].output();

}

return 0;

}

Q2 – Class to represent the details of 2 students

Solution –

#include <iostream>

using namespace std;

class student

{

private:

int roll;

float percentage;

public:

void input();

void output();

char grade();

} harsh, utkarsh;

void student ::input()

{

cout << "Enter roll " << endl;

cin >> roll;

cout << "Enter Percentage " << endl;

cin >> percentage;

}

char student ::grade()

{

if (percentage >= 90)

{

return 'A';

}

else if (percentage >= 80)

{

return 'B';

}

else if (percentage >= 70)

{

return 'C';

}

else

{

return 'D';

}

}

void student ::output()

{

cout << "Roll - " << roll << " Percentage - " << percentage << endl;

cout << "Grade - " << this->grade() << endl;

}

int main()

{

harsh.input();

utkarsh.input();

harsh.output();

utkarsh.output();

return 0;

}

Q3 – WAP to represent the details of a student.

Solution –

#include <iostream>

using namespace std;

class student

{

private:

int roll;

float percentage;

public:

void input();

void output();

// char grade(float a);

char grade() ;

};

void student ::input()

{

cout << "Enter roll " << endl;

cin >> roll;

cout << "Enter Percentage " << endl;

cin >> percentage;

}

char student ::grade()

{

if (percentage >= 90)

{

return 'A';

}

else if (percentage >= 80)

{

return 'B';

}

else if (percentage >= 70)

{

return 'C';

}

else

{

return 'D';

}

}

void student ::output()

{

cout << "Roll - " << roll << " Percentage - " << percentage << endl;

cout << "Grade - " << this->grade() << endl;

}

int main()

{

student harsh;

harsh.input();

harsh.output();

return 0;

}

Q4 – What are access specifiers in C++?

Answer –

Access Specifiers define how the members of a class can be accessed.

There are three access specifiers in cpp –

1. Public : Members are accessible from outside the class.
2. Private : Members cannot be accessed or viewed from outside the class scope.
3. Protected : Members cannot be accessed or viewed from outside the class however

Can be accessed from inherited class.

Example –

class MyClass {  
  **public:**    // Public access specifier  
    int x;   // Public attribute  
  **private:**   // Private access specifier  
    int y;   // Private attribute  
};  
  
int main() {  
  MyClass myObj;  
  myObj.x = 25;  // Allowed (public)  
  myObj.y = 50;  // Not allowed (private)  
  return 0;  
}

Q5 – What is an object?

Answer –

An Object is an instance of a Class. When a class is defined, no memory is allocated but when it is instantiated (i.e. an object is created) memory is allocated. Defining Class and Declaring Objects. A class is defined in C++ using keyword class followed by the name of class

Here,

Class student {

int data ;

}s1 ;

Student is the class , and s1 is the object.

Q6 – How to define a member function?

Answer –

There are two ways to define a member :

1. Inside the class declaration : We can define a function inside a class along with its

declaration. We can define a function inside a class if the function definition is

less than 4 lines. However it is not a standard practice to define a function inside a class

definition.

Example –

Class student {

int roll ;

void input () {

cout << “Enter Roll” << endl;

cin >> roll;

}

}

Here, the function input is defined within the class declaration, It’s wont give an

Error since there are only two line of code in input function.

1. Outside the class definition : We can Define a function outside the class definition using the

Scope resolution operator. In this method we declare the function with appropriate arguments and define it outside the class block.

Syntax : <return type> <class name > :: <function name> () {

// code goes here

}

This is a standard way of defining member function.

Thursday, 27 January 2022

Q1 What is structure ?

Answer – Structure is a user defined data type used to collect different types of data member. In c++, We have member function for structure too.

The size of a structure is equal to the sum of the size of its data members, however an empty structure takes up one byte of memory. Structures in c++ has two access specifiers, Public and Private. By default, the access specifiers of structure is set to public.

Q2 – What is union ?

Answer – Unions are user defined data type which uses a common memory location for its data members. The size of an union data type is equal to the size of largest data member.

Unions are used for efficient management of space. We can initialise a data member during declaration, but at max only one initialisation is allowed in union.

Q3- What is enum ?

Answer – An enumeration is a user defined data type consisting of set of named constant called enumerators.

Q4- Difference between a class and a structure.

Answer –

|  |  |
| --- | --- |
| Class   * A class in c++ can be defined as a collection of related variables and function encapsulated in a single structure. * Keyword : class * Three access specifiers : public, private and protected. * Default access specifiers : Private. * Has features like data abstraction, inheritance etc. * Used for large amount of data. | Structure   * A structure can be referred to as a user defined data type possessing its own operations. * Keyword : struct * Two access specifiers : public and private. * Default access specifiers : Public * Grouping of data * Used for smaller amount of data. |

Q5 – Difference between structure and union.

Answer –

|  |  |
| --- | --- |
| Structure   * A structure can be referred to as a user defined data type possessing its own operations. * Keyword : struct * Two access specifiers : public and private. * Default access specifiers : Public * Grouping of data * Used for smaller amount of data. | Union   * A union is a user defined data type, with its data member sharing a common memory location. * Keyword : union * No access specifiers. * No default access specifiers. * No member function. * Max one initialisation allowed. * Used for better memory management |

Q6 – WAP to display to display the student info -

#include <iostream>

using namespace std;

struct student

{

private:

int roll;

char name[20];

public:

void setdata();

void display();

};

void student ::setdata()

{

cout << "Enter name : ";

cin >> name;

cout << "Enter roll : ";

cin >> roll;

}

void student ::display()

{

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

}

union stud

{

int registration;

int roll;

};

enum day

{

mon,

tue = 8,

wed,

thur = 5,

fri,

sat,

sun

};

int main()

{

student S1;

S1.setdata();

S1.display();

stud S2;

cout << "Enter registration no. :: ";

cin >> S2.registration;

cout << "Resgistration number is " << S2.registration << endl;

cout << "Enter Roll : ";

cin >> S2.roll;

cout << "Roll is : " << S2.roll << endl;

day x, y, z;

x = mon;

y = fri;

z = wed;

cout << x << endl;

cout << x + 3 << endl;

cout << y << endl;

cout << z << endl;

return 0;

}

***Sunday, 28 January 2022***

Q1 – What are inline functions ?

Ans – Functions are sets of instruction to perform a particular task. When a function is called the control of program jumps from calling block to function block and hops back after the execution of function. This hoping of control consumes cpu resources.

So, if the function contains less than 4 lines, it consumes unnecessary CPU resource to hop back and forth thus its is recommended to use code blocks instead of function call, but then code reusability is jeopardized. To overcome this problem, we use inline function. When an Inline function is called, compiler replaces the function call with lines of codes avoiding the control jump while running.

While declaration an inline keyword is used to make the function inline. However, inline is just a request to compiler not a command which a compiler may refuse.

Syntax: inline <return\_type> <Function\_name> () ;

Q2 – What are the conditions for Inline functions ?

Answer - Conditions for inline Function :

1. Function should not contain loops.
2. Function should not be recursive.
3. Contains less than 5 lines of code.
4. Return statement for non void function.
5. Function should not contain jump statements.
6. Function should not have static members/variables.

Q3 – What are non-inline functions ?

Answer – Functions which are not inline are called non inline function .

Q4 – What are inline member functions ?

Answer – There are two methods to declare a function, First: Declare and define the member function inside the class definition block itself and Second: Declaring the member function inside the class block and defining the function outside the class block using the scope resolution operator.

By default, the functions declared and defined inside the class block are inline and the ones declared outside the block are non-inline functions.

To make the functions inline member functions we use the keyword inline while declaring the member function.

Q5 -What are static data members?

Answer – The variables which have the same memory location for all the objects in a class is called a static variable.

Static variables are initialised with default value 0. The initialisation of static variable inside the class is not allowed.

The static variable is visible to the class members only but the life of these variables are throughout the program.

Q6 – What are static functions ?

Answer – A static function can have access to only static member declared in same class.

Q7 – WAP including Inline, non-inline functions & static member and function.

Answer –

// Practice/ Homework - 28 Jan 2022

#include <iostream>

using namespace std;

class items

{

int id;

int cost;

static int count;

public:

void setitem();

void display()

{

cout << "ID - : " << id << ", Cost - : " << cost << endl;

}

inline static void getcount();

};

int items ::count = 1000;

void items ::setitem()

{

cout << "Enter cost : ";

cin >> cost;

id = count;

count++;

}

void items ::getcount()

{

cout << "count is " << count << endl;

}

int main()

{

items x, y, z;

x.setitem();

y.setitem();

z.setitem();

items ::getcount();

x.display();

y.display();

z.display();

return 0;

}

Wednesday, 02 February 2022

Q1 – What is a reference variable ?

Answer – Reference variable is a special variable which will take reference of another variable.

Or, Reference variable is an alternate name of already existing variable. It cannot be changed to refer another variable and should be initialised at the time of declaration and cannot be NULL.

Syntax : int &b = a ; // a should be an existing variable.

Q2 - What is call by value ?

Answer – When we pass a copy of variable as arguments to a function, it is called call by value.

Every argument we pass is stored on a different memory location and operated upon, so the original arguments/variable remains unchanged.

Q3 – What is call by address ?

Answer – When we pass a pointers to variable in the function as arguments, it is called call by address.

We pass the address of variable instead of copying the content of variable and access the content in function using the dereferencing operator.

Any change in variable reflects at original argument since we are dereferencing to the same memory location.

Q4 – What is call by reference ?

Answer – When we pass the reference of variables as arguments to the function, it is called call by reference.

The reference is just another name for same memory location, that’s why if we edit the variable in the functions, it gets reflected to the main function .

Q5 – WAP to swap two numbers using all the three calling method.

Answer –

// Function calling - value , refrence and address

// HW - 2 Feb 2022

#include <iostream>

using namespace std;

// call by value -> Passing a copy of variable as argument

void swap0(int x, int y)

{

int temp = x;

x = y;

y = temp;

}

// call by address -> Passing pointers to the variable as an argument

void swap1(int \*x, int \*y)

{

int temp = \*x;

\*x = \*y;

\*y = temp;

}

// call by refrence -> Passing refrence variable as an argument

void swap2(int &x, int &y)

{

int temp = x;

x = y;

y = temp;

}

int main()

{

int a = 10, b = 20;

cout << a << " " << b << endl;

// Call by value

swap0(a, b);

cout << " Call by value : ";

cout << a << " " << b << endl;

a = 10, b = 20;

// call by address

swap1(&a, &b);

cout << " Call by address : ";

cout << a << " " << b << endl;

a = 10, b = 20;

// Call by refrence

swap2(a, b);

cout << " Call by Refrence : ";

cout << a << " " << b << endl;

a = 10, b = 20;

return 0;

}

Thursday, 03 February 2022

Q1 – What is a friend function ?

Answer – A friend function is a non-member function which can access the private, public and protected data members of a class .

Syntax : friend <function declaration> ;

Rules to define a friend function :

* Friend function is declared inside the class block with a friend keyword and defined outside the class block.
* One function can be declared as a friend of more than one function.
* The function can be declared as a friend function inside a class in any section of class (public, private or protected).

Purpose of using friend function : We use friend function to access the private members of a class in case we need to.

Q2 – What is a friend class ?

Answer – A class whose function can access the private members of another function is called friend class.

Friendship of classes is not mutual ( i.e if A is a friend of B then B might not be a friend of A ).

Syntax : friend class <class name> ;

Purpose of using a friend class :

It is very complicated and error prone method to make an individual member function a friend function, since there are a lot of rules for declaring and defining member friend function. So in case we need to declare a friend member function, we rather choose to make the whole class a friend and avoid potential errors.

Q3 – WAP to swap two numbers of two different class using friend class and function.

Answer –

[16:15, 2/3/2022] Shubhansu Kr: #include <iostream>

using namespace std;

class A;

class B;

class B

{

int b = 76;

public:

friend void display(A &X, B &Y);

void swap(A &X);

};

class A

{

int a = 43;

public:

// friend class B;

friend void B ::swap(A &X);

friend void display(A &X, B &Y);

};

void B ::swap(A &X)

{

int temp = b;

b = X.a;

X.a = temp;

}

void display(A &X, B &Y)

{

cout << "A : " << X.a << " B : " << Y.b << endl;

cout << endl;

}

int main()

{

A obj1;

B obj2;

display(obj1, obj2);

obj2.swap(obj1);

display(obj1, obj2);

return 0;

}

Q 4 – WAP to find the sum using friend class.

Answer –

[16:16, 2/3/2022] Shubhansu Kr: #include <iostream>

using namespace std;

class A;

class B;

class A

{

int a = 33, b = 54;

public:

friend class B; // B is a friend now

};

class B

{

int s;

public:

void sum(A &X);

float Avg()

{

return s / 2.0;

};

};

void B ::sum(A &X)

{

s = X.a + X.b;

}

int main()

{

A obj1;

B obj2;

obj2.sum(obj1);

cout << "Average : " << obj2.Avg() << endl;

return 0;

}

Q5 – WAP to find the area of rectangle .

Answer –

#include <iostream>

using namespace std;

class Area;

class Dimensions

{

int length = 42;

int breadth = 32;

public:

friend Area;

friend void display(Dimensions &X, Area &Y);

};

class Area

{

int ar;

public:

void area(Dimensions &X);

friend void display(Dimensions &X, Area &Y);

};

void Area::area(Dimensions &X)

{

ar = X.breadth \* X.length;

}

void display(Dimensions &X, Area &Y)

{

cout << "Length : " << X.length << " Breadth : " << X.breadth << " ";

cout << "Area : " << Y.ar << endl;

cout << endl;

}

int main()

{

Dimensions D1;

Area A1;

A1.area(D1);

display(D1, A1);

return 0;

}

Tuesday, 08 February 2022

Q1 – What is a Function ?

Answer –

Sets of instruction in a particular block to perform a specific task is called a function.

Functions are used to increase the code reusability and increase debugging efficiency of code .

There are two types of functions :

1. Library Functions : The functions which are already provided by the header file and included using header files in the program.
2. User defined functions : These functions are created by users to perform a specific task according to the need.

Components of a Function :

1. Declaration : The declaration of function is done to specify the return type , name and parameters of a function.
2. Function calling : The only function which calls itself on its own is main, rest all the functions needs to be called in order to be executed.
3. Function Definition : It is the block of instruction that a function executes when called upon.

Q2 – WAP to find the square root of a number.

Answer –

// Wap to find the square root of a number

#include <iostream>

#include <math.h>

using namespace std;

void root()

{

int n;

cout << "Enter n " << endl;

cin >> n;

cout << "Square root : " << sqrt(n) << endl;

}

void root(int a)

{

cout << "Square root of " << a << " is : " << sqrt(a) << endl;

}

float rootRet()

{

cout << "Enter n " << endl;

int n;

cin >> n;

return sqrt(n);

}

float rootRet(int a)

{

return sqrt(a);

}

int main()

{

int n;

cout << "Enter number : ";

cin >> n;

root(n);

root();

float root = rootRet();

root = rootRet(n);

return 0;

}

Q3 – WAP to find the power of a number.

Answer –

// Find power

#include <iostream>

using namespace std;

int power(int &a, int &x)

{

int p = 1;

while (x > 0)

{

p \*= a;

x--;

}

return p;

}

int main()

{

int n, x;

cout << "Enter n " << endl;

cin >> n;

cout << "Enter power " << endl;

cin >> x;

cout << "Answer : " << power(n, x) << endl;

return 0;

}

Q4 – WAP to swap two numbers.

Answer –

// Swapping of two numbers

#include <iostream>

using namespace std;

int a = 10, b = 20;

void swap()

{

int temp = a;

a = b;

b = temp;

}

void swap(int &a, int &b)

{

int temp = a;

a = b;

b = temp;

}

int swap1()

{

int temp = a;

a = b;

b = temp;

return 1;

}

int swap1(int &a, int &b)

{

int temp = a;

a = b;

b = temp;

return 1;

}

int main()

{

int a = 100, b = 200;

// cout << "Enter a " << endl;

// cin >> a;

// cout << "Enter b " << endl;

// cin >> b;

// Without passing the argument we cannot swap the local variable

cout << "Before swapping global var " << endl;

cout << ::a << " " << ::b << endl;

swap();

cout << "Global variable after swap " << endl;

cout << ::a << " " << ::b << endl;

cout << endl;

// Swapping using parameters

cout << "Before swapping local var " << endl;

cout << a << " " << b << endl;

swap(a, b);

cout << "After swapping : " << endl;

cout << a << " " << b << endl;

cout << endl;

// Return type swapping

cout << "Before swapping global var " << endl;

cout << ::a << " " << ::b << endl;

swap1();

cout << "Global variable after swap " << endl;

cout << ::a << " " << ::b << endl;

cout << endl;

// Swapping using parameters and return type

cout << "Before swapping local var " << endl;

cout << a << " " << b << endl;

swap1(a, b);

cout << "After swapping : " << endl;

cout << a << " " << b << endl;

cout << endl;

return 0;

}