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;

}

Wednesday, 09 February 2022

Tutorial Work –

Q1. Write the difference between –

1. Structure and Union

|  |  |
| --- | --- |
| 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 |

1. Structure and Array

|  |  |
| --- | --- |
| Structure   * Structure is a user defined data type. * Structure elements are accessed using the dot operator. * Structures can also contain functions. * Memory allocation in structure is not continuous and uniform. * Structure can contain different types of data members. | Array   * Array is not a user defined array type. * Array elements are accessed using the [] operator. * Arrays cannot contain function. * Memory allocation in array is uniform and continuous. * Array contains only one type of data. |

1. Call by value, Call by reference and call by address –

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.

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.

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.

1. Inline and non inline member function –

The member functions which are declared inside and defined inside the class block only are inline by default.

The member function which are declared inside the class block and defined outside the class are non-inline by default.

Inorder to make the non-inline member function inline, inline keyword is added before the function declaration inside the class.

1. Public, Private and Protected access specifiers

Public, private and Protected are keywords used to specify the access of data members and member function of a class.

Public access is used to make the members accessible by program. Private is used to make the members inaccessible. Private members can only be access by the same class function or friend functions. Protected members are similar to private in reference to accessibility but however a private data cant be inherited by the derived class while a protected data can be inherited.

Q2 – Write a short note on recursion.

Answer –

* A function that calls itself is called a recursive function, and the process is called recursion.
* A recursive function has a condition which controls the recursion switch and this condition is called recursive condition.
* A recursion technique can be used to solve a lot of complex problems but mostly it is used to solve problems with hierarchy conditions.
* It helps reduce the code length.
* Although it is helpful in solving complex problems, Recursion takes a lot of space in stack since function is called again and again and thus takes a lot of time and Cpu resource.

Q3 – What is 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.

Q4 – What is Enum ?

Answer –

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

Q5 – Write a short note on static data and function.

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.

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

Q6 – Write a short note on friend function and class.

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.

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.

Thursday, 10 February 2022

Q1. What is a local variable?

Answer –

Variables defined within a function or block are said to be local to those functions.

• Anything between ‘{‘ and ‘}’ is said to inside a block.

• Local variables do not exist outside the block in which they are declared, i.e. they can not be accessed or used outside that block.

• Declaring local variables: Local variables are declared inside a block.

Q2. What is a global variable?

Answer –

As the name suggests, Global Variables can be accessed from any part of the program.

They are available through out the life time of a program.

They are declared at the top of the program outside all of the functions or blocks.

Declaring global variables: Global variables are usually declared outside of all of the functions and blocks, at the top of the program. They can be accessed from any portion of the program.

Q3 – Write the output of following program.

Ans-

// scope of var

#include <iostream>

using namespace std;

int a = 1;

int main()

{

int a = 2;

cout << a << " "; // 2

{

int a = 3; // 3 -> 4 in line 17

cout << a << " "; // 3

{

a = 4; // Assigns 4 to the var a in the block (Not a global)

cout << a << " "; // 4

}

cout << a << " "; // prints a from line 13 (4)

}

cout << a << " "; // Prints a from line 10 (2)

return 0;

Wednesday, 16 February 2022

Q1. What is a default argument?

Answer –

Default Arguments in C++ A default argument is a value provided in a function declaration that is automatically assigned by the compiler if the caller of the function doesn't provide a value for the argument with a default value.

If the caller provides a value, the default parameter is over written by the provided value.

Rule to declare the default argument :

1. The default argument is placed from last to first order.

Q2 – What is function overloading?

Answer –

More than one function sharing the same name but different parameters is called function overloading.

The signature of the function should be different :

1. Sequence of parameter
2. No. of parameter
3. Different data types of data types

Function overloading helps in remembering the name of function easily.

Q3 – What is a pointer?

Answer –

A pointer is a special variable which stores the address of other variables of same data type.

The memory taken by any kind of pointer is equal to the memory taken by unsigned int.

\* Indirection, De-refrencing, Value at address

& Address, And, Ampersand

Q4 – What is recursion?

Answer –

// Recursion

// A function which calls itself is called a recursive function

// The process of calling itself is called recursion.

// A recursive function should hava a base condition to terminate the

// recursion process.

// Usually done via if else condition control

#include <iostream>

using namespace std;

class Num

{

public:

int Factorial(int a);

};

int Num ::Factorial(int a)

{

if (a == 1 || a == 0)

{

return 1;

}

return a \* Factorial(a - 1);

}

int main()

{

Num F;

cout << "Factorial 5 : " << F.Factorial(5) << endl;

return 0;

}

Q5 – Pointers

Answer –

// Pointers - Pointers are Variables which stores address of other variable

// of same data type.

// Any poiter takes the size equal to the size of unsigned interger

// Because the address is in the form of unsigned interger only.

#include <iostream>

using namespace std;

int main()

{

int a = 20;

int \*p = &a;

// \* --> Indirection, derefrence , value of operator

// & --> Ampersand , address of , And operartor

cout << a << endl;

cout << \*p << endl;

cout << p << endl;

cout << sizeof(p) << endl; // Size of unsigned int

return 0;

}

Friday, 18 February 2022

Q1. What are the types of pointers ?

Answers – There are five types of pointers :

1. Null Pointers
2. Wild pointers
3. Generic pointers
4. Constant pointer
5. Dangling pointer

Q2. What is Null Pointer?

Answer – It is a pointer that does not point to any memory location.

There are two ways to declare a Null Pointer :

1. Int \*ptr = NULL ;
2. Int \*ptr = 0 ;

It is invalid to dereference a null pointer .

Q3. What is a wild pointer?

Answer – It is a pointer which is declared but not yet initialised.

Eg-

int \*p ; // p is a wild pointer

int a ;

a = 10 ;

Q4. What is a dangling pointer?

Answer –

An initialised pointer whose variable is destroyed is called a dangling pointer.

Q5. What is a constant pointer?

Answer –

A constant pointer is a pointer which is initialised with an address and cannot be changed is called a constant pointer.

Syntax - int \* const ptr = &a ;

A constant pointer must be initialised while declaration.

Q6. What is a generic Pointer?

Answer –

A generic pointer is a pointer that can hold the address of different data types.

Type casting is req to access var .

Tuesday, 22 February 2022

Q1. Discuss Pointers and Arrays.

Answer –

A pointer can point towards an array using the following notation:

Int a[] = {2, 43} ;

1. Int \*p = a ;
2. Int \*p = a[];
3. Int \*p = a[0];

We can use pointer arithmetic on p as well as a .

To access the ith element :

1. \*(p+i) ;
2. \*(a+i) ;

However, the name of the array cannot be reassigned to a new memory location, it is of constant pointer type.

Q2. Discuss Pointers with class

Answer –

We can create a pointer as a data member of the class as well as define a pointer to the object of the class.

Class A {

Int \*a ;

Int b ;

};

A obj1 ;

A \* ptr1 = &obj1;

We can access the public members with pointer using the following syntax :

Int A::\*p = &A::a;

P is a pointer, pointing towards a data member a of class A .

Q3 – Practice program –

Ans –

// Pointer to class-Objects

#include <iostream>

using namespace std;

class A

{

int a;

public:

int b;

inline void getdata(int a, int b);

void displaydata()

{

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

}

};

void A ::getdata(int a, int b)

{

this->a = a;

this->b = b;

}

int main()

{

A shyam;

shyam.getdata(3, 5);

shyam.displaydata();

// Pointer to object ;

A \*ptr = &shyam;

// Access data members

cout << ptr->b << " " << (\*ptr).b << endl;

// Access memberfuntion

ptr->displaydata();

(\*ptr).displaydata();

// we can also declare a genral pointer for the public data members and

// member funtion of a class

int A::\*p = &A::b;

// It's similar to declaring a pointer inside the class as a data member

// int \*p = &b --> inside class

// int A::\*q() = &A::displaydata;

return 0;

}

// #include <iostream>

// using namespace std;

// class X

// {

// public:

// int a;

// void f(int b)

// {

// cout << "The value of b is " << b << endl;

// }

// };

// int main()

// {

// // declare pointer to data member

// int X::\*ptiptr = &X::a;

// // declare a pointer to member function

// void (X::\*ptfptr)(int) = &X::f; // void (\*pf)()

// // create an object of class type X

// X xobject;

// // initialize data member

// xobject.\*ptiptr = 10;

// cout << "The value of a is " << xobject.\*ptiptr << endl;

// // call member function

// (xobject.\*ptfptr)(20);

// }

Q4 – Discuss this pointer

Answer –

This pointer stores the address of current object

// This pointer

// this pointer stores the address of current object

#include <iostream>

using namespace std;

class A

{

int a = 8;

public:

void setdata(int a)

{

cout << this->a << endl;

this->a = a;

cout << a << endl;

}

void display()

{

cout << a << endl;

cout << this << endl;

}

};

int main()

{

A shyam;

shyam.setdata(4);

shyam.display();

cout << &shyam << endl;

A ram;

ram.display();

return 0;

}

// #include <iostream>

// using namespace std;

// class X

// {

// private:

// int a;

// public:

// void Set\_a(int a)

// {

// // The 'this' pointer is used to retrieve 'xobj.a'

// // hidden by the automatic variable 'a'

// cout << this << endl;

// this->a = a; // fnc formal para

// }

// void Print\_a()

// {

// cout << "a = " << a << endl;

// }

// };

// int main()

// {

// X xobj, x2;

// int a = 5;

// xobj.Set\_a(a);

// xobj.Print\_a();

// x2.Set\_a(10);

// x2.Print\_a();

// }

Wednesday, 23 February 2022

Q1 – What is a constructor ?

Answer –

Constructor is a special member function having name as that of its class which is used to initialize some valid calues to the member variable .

Constructor is called automatically when an object is created.

It does not have any return type .

We can declare more than one constructor.

If we do not declare any constructor the compiler generated a default constructor (Expects no parameter and has an empty body) .

Constructor is declared within the public section of the class.

Q2 – What is a default constructor ?

Answer – A constructor with no parameters is called a default constructor

Q3 – WAP with constructor

Answer –

// Constructor

#include <iostream>

using namespace std;

class Student

{

int roll;

string fullName;

int age;

string section;

public:

// Default constructor

Student()

{

section = "K21GP";

}

// Parameterised constructor

Student(int a)

{

section = "K21GP";

roll = a;

}

// Parameterised constructor using scope resolution

Student(int a, int b);

void setData(string a, int b);

void displayData();

};

Student ::Student(int a, int b)

{

roll = a;

age = b;

}

void Student ::setData(string a, int b)

{

fullName = a;

age = b;

}

void Student ::displayData()

{

cout << "Student Name : " << fullName << endl;

cout << "Section : " << section << endl;

cout << "Roll Number : " << roll << endl;

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

}

int main()

{

Student S[5] = {33}; // Using parameterised constructor with array of

// objects

// (S[0]).displayData();

Student S1(32);

Student S2(33, 18);

S2.displayData();

return 0;

}

// #include <iostream>

// using namespace std;

// class Rectangle

// {

// private:

// int length, breadth;

// public:

// Rectangle() // def const

// {

// cout << "\n enter l and b";

// cin >> length >> breadth;

// // length=5;breadth=6;

// }

// void area()

// {

// int a = length \* breadth;

// cout << "area is " << a << endl;

// }

// };

// main()

// {

// Rectangle r1;

// r1.area();

// Rectangle r2;

// r2.area();

// }

Thursday, 03 March 2022

Q1. What is a string?

Answer –

String is a sequence of characters terminated by a null character

There are two styles to declare a string :

1. C-Style : Using char pointer or array
2. C++ Style : Using string class

Syntax : char \* st ;

Char str[20] = “”;

String strin ;

In c++ string is a class which creates a char obj for each character dynamically with continuous memory location.

Initialisation –

1. Default constructor : String s1 ;
2. Parameterised constructor : String s2(“sh”) ;
3. S1 = s2

Q2 – comparison of strings

// String is compared using characters

#include <iostream>

#include <string.h>

using namespace std;

int main()

{

string s1 = "Shubh";

string s2 = "Abc";

string s3 = "shubh";

// Compare method is case sensitive

// Compare the whole string

cout << s1.compare(s2) << endl; // 1 --> s1 > s2

cout << s2.compare(s3) << endl; // -1 --> s2 < s3

cout << s1.compare(s1) << endl; // 0 --> strings are equal

// Comparing substrings

cout << s1.compare(1, 2, s3, 1, 2);

// index of s1, no. of char , comparing string, index, no. of char

// strcmp(s1, s2) it is used to comapare strings, without case sensitivity

// cout << strcmp(s1, s3) << endl; we need to pass c style string

char c[4] = "abc";

cout << c[3] << "hell";

return 0;

}

// / greater than >

// if (s1 > s2)

// cout

// << "Greater" << endl;

// else cout << "Not greater" << endl;

// // less than <

// if (s1 < s2)

// cout << "smaller" << endl;

// else

// cout << "Not smaller" << endl;

// // not Equal to !=

// if (s1 != s2)

// cout << "not Equal" << endl;

// else

// cout << "equal" << endl;

// s1 = "Rahul";

// s2 = "Ramu";

// int c = s1.compare(s2);

// cout << c;

// cout << s1.compare(0, 2, s2, 0, 2);

// }

Q3 Searching in a string

// Finding Elements in string

#include <iostream>

using namespace std;

int main()

{

string s1 = "ABAABCDA";

// Find\_first\_of('char') - return the index of first occurence

// Find\_last\_of('char') - returns the occurence of last occurence

cout << "First of " << endl;

cout << "A " << s1.find\_first\_of('B') << endl;

cout << "B " << s1.find\_first\_of('A') << endl;

cout << "C " << s1.find\_first\_of('C') << endl;

cout << "D " << s1.find\_first\_of('D') << endl;

cout << "E " << s1.find\_first\_of('E') << endl; // gv

// gives garbage if element is not present

cout << "Find last of " << endl;

cout << "A " << s1.find\_last\_of('A') << endl;

cout << "B " << s1.find\_last\_of('B') << endl;

cout << "C " << s1.find\_last\_of('C') << endl;

cout << "D " << s1.find\_last\_of('D') << endl;

cout << "E " << s1.find\_last\_of('E') << endl; // gv

// gives garbage if element is not present

// cannot find string, it takes the first char of string and use

// it as target

cout << "BC " << s1.find\_first\_of("BC") << endl;

// Here the function takes B as target and returns the first occurrence

// index of B (ie 1 in this case ) ;

// Find first not of gives the index of first element which is not

// specified in the parametes

cout << "Find not first" << endl;

cout << "A Not - " << s1.find\_first\_not\_of('A') << endl;

cout << "B Not - " << s1.find\_first\_not\_of('B') << endl;

cout << "C Not - " << s1.find\_first\_not\_of('C') << endl;

cout << "D Not - " << s1.find\_first\_not\_of('D') << endl;

cout << "E Not - " << s1.find\_first\_not\_of('E') << endl;

// Passing multiple parameters -

cout << "A, B Not - " << s1.find\_first\_not\_of("AB") << endl;

cout << endl;

// find - returns the index of matching target starting

cout << "Find " << endl;

cout << "A " << s1.find('A') << endl;

cout << "A " << s1.find("BC") << endl;

// rfind - searches the string from the last

cout << "A " << s1.rfind('A') << endl;

return 0;

}

Q4 – String info

// String Info

#include <iostream>

using namespace std;

int main()

{

string s1("Shubhansu"), s2("Kumar"), s3;

string s4("Shubhansu Kumar Singh");

// capacity is used to find the number of characters a string can

// store before reallocating more memory

// length() is used to get the length of string

// size () is used for the same

cout << s1.capacity() << " " << s1.length() << endl;

cout << s2.capacity() << " " << s2.size() << endl;

cout << s3.capacity() << " " << s3.length() << endl;

cout << s4.capacity() << " " << s4.length() << endl;

// Default capacity of string is 15 depending upon hardware

// Clear is used to erase all the characters of string

s4.clear();

cout << s4.capacity() << " " << s4.length() << endl;

// capacity remains the same even after clearing

// empty() is used to check if string is empty or not

// if empty - 1 else 0

cout << "s1 - " << s1.empty() << endl;

cout << "s2 - " << s2.empty() << endl;

cout << "s3 - " << s3.empty() << endl; // 1

cout << "s4 - " << s4.empty() << endl; // 1

// swap() - takes a string obj and swaps the strings

cout << "Before Swap " << s1 << " " << s2 << endl;

s1.swap(s2);

// s1.swap("abc"); Error since we have to pass an string object

cout << "After Swap " << s1 << " " << s2 << endl;

return 0;

}

Q5 – String Manipulation

// Manipulation of string

#include <iostream>

using namespace std;

int main()

{

string s1, s2;

s1.insert(0, "Shubh");

s2.insert(0, "Kumar");

cout << s1 << " " << s2 << endl;

string s3 = s1 + s2;

cout << s3 << endl;

// Insert takes two parameter - (Insertion index, "String") ;

// Erase takes two parameter - (Deletion index , No. of characters to be erased)

s3.erase(5, 5);

cout << "S3 after erase - " << s3 << endl;

// Replace takes three parameter - (start index , end index , string) ;

s3.append("ansu Kumar");

cout << s3 << endl;

// Replace 'kumar' with 'singh' in s3

s3.replace(9, 6, " Singh");

cout << s3 << endl;

// Clear is used to erase the content of string

s3.clear();

cout << s3 << endl;

return 0;

}

Q6 – String

// Strings - Sequence of characters terminated by a null character

// There are two styles to declare a string

// c-style ---> Array of character or character pointer

// C++-style ---> Using String class

#include <iostream>

using namespace std;

int main()

{

string S1 = "Kumar";

string S2("Shubhansu Kumar");

char \*title = "Singh";

char name[20] = "Shubh";

// cout << S1 + title << endl;

return 0;

}

// #include <iostream>

// #include <string>

// using namespace std;

// main()

// {

// // initialization of a string

// string s1 = "ABC";

// cout << "S1 string:" << s1 << endl;

// string s2("ABCD"); // another method to initialize a string

// cout << "S2 string:" << s2 << endl;

// string s3 = "STUDY";

// cout << "S3 string:" << s3 << endl;

// // String Functions

// // length function or size

// cout << "length of s1 string is:" << s1.length() << endl; // 3

// cout << "length of s2 string is:" << s2.size() << endl; // 4

// cout << "capacity" << s1.capacity() << endl; // 3 15 20

// // swap

// swap(s1, s2);

// cout << "S1 string: after swap " << s1 << endl;

// cout << "S2 string: after swap " << s2 << endl;

// // find\_last\_of()

// s1 = "ABCDAAB";

// cout << "find\_last\_of('A') " << s1.find\_last\_of('A') << endl; // 5

// cout << "find\_first\_of('A') " << s1.find\_first\_of('A') << endl; // 0

// // clear function

// s3.clear(); // clear() deletes all the characters availabe in string.

// cout << "length of s3 after clear function:" << s3.length() << endl;

// // empty function

// cout << "empty function:" << s3.empty() << endl; // empty() tells whether our string is empty or not. 1-empty and 0- non empty

// // append function

// s3 = "LPU\*UNIVERSITY";

// s3.append("\*JALANDHAR"); // append() string at the end.

// cout << "s3 string after append:" << s3 << endl;

// // erase(a,b) function

// s3.erase(4, 10); // start erasing from 4th index and erase 10 characters after it.

// cout << "after erase:" << s3 << endl;

// // replace function

// s3.replace(5, 10, "JAL"); // from 5 index it is replacing 10 characters with "JAL" .

// cout << "string after replace:" << s3 << endl;

// // substr function s3: abc1234abc

// cout << "substring using two arguments:" << s3.substr(5, 3) << endl; // from 5th index it is giving 3 characters.

// cout << "substring using single argument:" << s3.substr(2) << endl; // it is returning all the characters afters 2nd index.

// // insert function

// cout << "after insert:" << s3.insert(4, "UNIVERSITY") << endl; // start inserting from 4th index.

// // find and rfind

// cout << "location of U using Find:" << s3.find("U") << endl; // abUxUyz 2

// cout << "location of U using rFind:" << s3.rfind("U") << endl; // 4

// // string operators

// // concatenation +

// s1 = "\*PHAGWARA";

// s3 = s3 + s1; // concatination of two strings

// cout << "Concatinated string:" << s3 << endl;

// string s5 = "PUNJAB";

// cout << "using character:" << s3 + '\*' + s5 << endl;

// // assignment =

// string s4 = s3; // assinging s3 to s4 by assignment operator.

// cout << "String assignment:" << s4 << endl;

// // equlaity ==

// if (s1 == s2)

// {

// cout << "Equal" << endl;

// }

// else

// {

// cout << "Not equal" << endl;

// }

// return 0;

// }