1. The variables are stored in the memory and need to declare the variables with data types. Demonstrate with program to find the Size of fundamental data types using C++ compiler.

#include <iostream>

Using namespace std;

int main()

{

cout << "\n\n Find Size of fundamental data types :\n";

cout << "------------------------------------------\n";

cout << " The sizeof(char) is : " << sizeof(char) << " bytes \n" ;

cout << " The sizeof(short) is : " << sizeof(short) << " bytes \n" ;

cout << " The sizeof(int) is : " << sizeof(int) << " bytes \n" ;

cout << " The sizeof(long) is : " << sizeof(long) << " bytes \n" ;

cout << " The sizeof(long long) is : " << sizeof(long long) << " bytes \n";

cout << " The sizeof(float) is : " << sizeof(float) << " bytes \n" ;

cout << " The sizeof(double) is : " << sizeof(double) << " bytes \n";

cout << " The sizeof(long double) is : " << sizeof(long double) << " bytes \n";

// cout << " The sizeof(bool) is : " << sizeof(bool) << " bytes \n\n";

return 0;

}

**Sample Output:**Find Size of fundamental data types :  
------------------------------------------  
The sizeof(char) is : 1 bytes  
The sizeof(short) is : 2 bytes  
The sizeof(int) is : 4 bytes  
The sizeof(long) is : 8 bytes  
The sizeof(long long) is : 8 bytes  
The sizeof(float) is : 4 bytes  
The sizeof(double) is : 8 bytes  
The sizeof(long double) is : 16 bytes  
The sizeof(bool) is : 1 bytes

1. There will be range for data values in storage now we will find check the upper and lower limit of all data types in C++.

#include <iostream>

#include <limits>

using namespace std;

int main()

{

cout << "\n\n Check the upper and lower limits of integer :\n";

cout << "--------------------------------------------------\n";

cout << " The maximum limit of int data type : " << INT\_MAX << endl;

cout << " The minimum limit of int data type : " << INT\_MIN << endl;

cout << " The maximum limit of unsigned int data type : " << UINT\_MAX << endl;

cout << " The maximum limit of long long data type : " << LLONG\_MAX << endl;

cout << " The minimum limit of long long data type : " << LLONG\_MIN << endl;

cout << " The maximum limit of unsigned long long data type : " << ULLONG\_MAX << endl;

cout << " The Bits contain in char data type : " << CHAR\_BIT << endl;

cout << " The maximum limit of char data type : " << CHAR\_MAX << endl;

cout << " The minimum limit of char data type : " << CHAR\_MIN << endl;

cout << " The maximum limit of signed char data type : " << SCHAR\_MAX << endl;

cout << " The minimum limit of signed char data type : " << SCHAR\_MIN << endl;

cout << " The maximum limit of unsigned char data type : " << UCHAR\_MAX << endl;

cout << " The minimum limit of short data type : " << SHRT\_MIN << endl;

cout << " The maximum limit of short data type : " << SHRT\_MAX << endl;

cout << " The maximum limit of unsigned short data type : " << USHRT\_MAX << endl;

cout << endl;

return 0;

}

**Expected Output:**  
Check the upper and lower limits of integer :  
--------------------------------------------------  
The maximum limit of int data type : 2147483647  
The minimum limit of int data type : -2147483648  
The maximum limit of unsigned int data type : 4294967295  
The maximum limit of long long data type : 9223372036854775807  
The minimum limit of long long data type : -9223372036854775808  
The maximum limit of unsigned long long data type : 18446744073709551615  
The Bits contain in char data type : 8  
The maximum limit of char data type : 127  
The minimum limit of char data type : -128  
The maximum limit of signed char data type : 127  
The minimum limit of signed char data type : -128  
The maximum limit of unsigned char data type : 255  
The minimum limit of short data type : -32768  
The maximum limit of short data type : 32767  
The maximum limit of unsigned short data type : 65535

1. C++ is an Object oriented program, we will write and illustrate with a program to create an object of a class and access class attributes.

/\* C++ program to create an object of a class and access class attributes \*/

#include <iostream>

#include <string>

Using namespace std;

class Student {

public:

int rollNo;

char stdName[];

float perc;

};

int main()

{

Student std; // object creation

// Accessing attributes and setting the values

std.rollNo = 132;

std.stdName = "Subhakar M";

std.perc = 98.20f;

// Printing the values

cout << "Student's Roll No.: " << std.rollNo << "\n";

cout << "Student's Name: " << std.stdName << "\n";

cout << "Student's Percentage: " << std.perc << "\n";

return 0;

}

**Output**

Student's Roll No.: 132

Student's Name: Subhakar M

Student's Percentage: 98.2

1. C++ program to create a class to demonstrate the use of objects with a program in C++ to read and add two distance

class Distance

{

    private:

int feet;

int inch;

    public:

Distance (); //Constructor

void getDist ();

void showDist ();

Distance addDist( Distance d2 );

Distance subDist( Distance d2 );

};

Distance:: Distance ()

{

feet = 0; inch = 0;

}

void Distance:: getDist()

{

cout << "Enter Value of feets : "; cin >> feet;

cout << "Enter value of inches : "; cin >> inch;

inch = (inch >= 12) ? 12 : inch;

}

void Distance:: showDist()

{

cout << endl << "\tFeets : " << feet;

cout << endl << "\tInches: " << inch;

}

Distance Distance:: addDist( Distance d2 )

{

Distance temp;

temp.feet = feet + d2.feet;

temp.inch = inch + d2.inch;

if( temp.inch >= 12)

{

temp.feet++;

temp.inch -= 12;

}

return temp;

}

Distance Distance:: subDist( Distance d2 )

{

Distance temp;

temp.feet = feet - d2.feet;

temp.inch = inch - d2.inch;

if( temp.inch < 0 )

{

temp.feet--;

temp.inch = 12 + temp.inch;

}

return temp;

}

int main()

{

Distance d1;

Distance d2;

Distance d3;

Distance d4;

cout << "Enter Distance1 : " << endl;

d1.getDist();

cout << "Enter Distance2 : " << endl;

d2.getDist();

d3 = d1.addDist(d2);

d4 = d1.subDist(d2);

cout << endl << "Distance1 : " ;

d1.showDist();

cout << endl << "Distance2 : " ;

d2.showDist();

cout << endl << "Distance3 : " ;

d3.showDist();

cout << endl << "Distance4 : " ;

d4.showDist();

cout << endl;

return 0;

}

**Output**

Enter Distance1 :

Enter Value of feets : 10

Enter value of inches : 7

Enter Distance2 :

Enter Value of feets : 15

Enter value of inches : 8

Distance1 :

Feets : 10

Inches: 7

Distance2 :

Feets : 15

Inches: 8

Distance3 :

Feets : 26

Inches: 3

1. Member functions are used as a part of object in class of OOPs concept, let’s see how to use functions in the program of C++.

#include <iostream.h>

class car

{

private:

int car\_number;

char car\_model[10];

public:

void getdata()

{

cout<<"Enter car number: "; cin>>car\_number;

cout<<"\n Enter car model: "; cin>>car\_model;

}

void showdata()

{

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

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

}

};

// main function starts

int main()

{

car c1;

c1.getdata();

c1.showdata();

return 0;

}

**Output**

Enter car number : 9999

Enter car model : Sedan

Car number is 9999

Car model is Sedan

1. Depending on the number and type of arguments passed, the corresponding display() function is called for Overloading Using Different Number of Parameters

#include <iostream.h>

// function with 2 parameters

void display(int var1, double var2) {

cout << "Integer number: " << var1;

cout << " and double number: " << var2 << endl;

}

// function with double type single parameter

void display(double var) {

cout << "Double number: " << var << endl;

}

// function with int type single parameter

void display(int var) {

cout << "Integer number: " << var << endl;

}

int main() {

int a = 5;

double b = 5.5;

// call function with int type parameter

display(a);

// call function with double type parameter

display(b);

// call function with 2 parameters

display(a, b);

return 0;

}

**Output**

Integer number: 5

Float number: 5.5

Integer number: 5 and double number: 5.5

1. The three types of the parametric constructor, one for initialization of name only, second to initialization of age only, and third to initialize both name and age. Program to be created three different types of objects and initialized them in different ways, and printed values for each of them.

**#include <iostream>**

using namespace std;

**class** **Person**{

*// declaring private class data members*

**private**:

string name;

**int** age;

**public**:

*// declaring parameterized constructor of three different types*

Person(string person\_name)

{

cout<<"Constructor to set name is called"<<endl;

name = person\_name;

age = 12;

}

Person(**int** person\_age)

{

cout<<"Constructor to set age is called"<<endl;

name = "Student";

age = person\_age;

}

Person(string person\_name, **int** person\_age)

{

cout<<"Constructor for both name and age is called"<<endl;

name = person\_name;

age = person\_age;

}

*// display function to print the class data members value*

**void** **display**()

{

cout<<"Name of current object: "<<name<<endl;

cout<<"Age of current object: "<<age<<endl;

cout<<endl;

}

};

**int** **main**()

{

*// creating objects of class using parameterized constructor*

Person **obj1**("First person");

*// printing class data members for first object*

obj1.display();

Person **obj2**(25);

*// printing class data members for second object*

obj2.display();

Person **obj3**("Second person",15);

*// printing class data members for third object*

obj3.display();

**return** 0;

}

**Output**

Constructor to set name is called

Name of current object: First person

Age of current object: 12

Constructor to set age is called

Name of current object: Student

Age of current object: 25

Constructor for both name and age is called

Name of current object: Second person

Age of current object: 15

1. C++ program to read and print employee information with department and of information using hierarchical inheritance.

#include <iostream>

#include <stdio.h>

using namespace std;

//Base Class - basicInfo

class basicInfo {

protected:

char name[30];

int empId;

char gender;

public:

void getBasicInfo(void)

{

cout << "Enter Name: ";

cin.ignore(1);

cin.getline(name, 30);

cout << "Enter Emp. Id: ";

cin >> empId;

cout << "Enter Gender: ";

cin >> gender;

}

};

//Base Class - deptInfo

class deptInfo : private basicInfo {

protected:

char deptName[30];

char assignedWork[30];

int time2complete;

public:

void getDeptInfo(void)

{

getBasicInfo(); //to get basic info of an employee

cout << "Enter Department Name: ";

cin.ignore(1);

cin.getline(deptName, 30);

cout << "Enter assigned work: ";

fflush(stdin);

cin.getline(assignedWork, 30);

cout << "Enter time in hours to complete work: ";

cin >> time2complete;

}

void printDeptInfo(void)

{

cout << "Employee's Information is: " << endl;

cout << "Basic Information...:" << endl;

cout << "Name: " << name << endl; //accessing protected data

cout << "Employee ID: " << empId << endl; //accessing protected data

cout << "Gender: " << gender << endl

<< endl; //accessing protected data

cout << "Department Information...:" << endl;

cout << "Department Name: " << deptName << endl; //accessing protected data

cout << "Assigned Work: " << assignedWork << endl; //accessing protected data

cout << "Time to complete work: " << time2complete << endl; //accessing protected data

}

};

//another Base Class : loadInfo

class loanInfo : private basicInfo {

protected:

char loanDetails[30];

int loanAmount;

public:

void getLoanInfo(void)

{

getBasicInfo(); //to get basic info of an employee

cout << "Enter Loan Details: ";

cin.ignore(1);

cin.getline(loanDetails, 30);

cout << "Enter loan amount: ";

cin >> loanAmount;

}

void printLoanInfo(void)

{

cout << "Employee's Information is: " << endl;

cout << "Basic Information...:" << endl;

cout << "Name: " << name << endl; //accessing protected data

cout << "Employee ID: " << empId << endl; //accessing protected data

cout << "Gender: " << gender << endl

<< endl; //accessing protected data

cout << "Loan Information...:" << endl;

cout << "Loan Details: " << loanDetails << endl; //accessing protected data

cout << "Loan Amount : " << loanAmount << endl; //accessing protected data

}

};

int main()

{

//read and print department information

deptInfo objD;

objD.getDeptInfo();

objD.printDeptInfo();

cout << endl

<< endl;

//read and print loan information

loanInfo objL;

objL.getLoanInfo();

objL.printLoanInfo();

return 0;

}

**Output**

Enter Name: Mickey

Enter Emp. Id: 1121

Enter Gender: F

Enter Department Name: Testing

Enter assigned work: To test login form

Enter time in hours to complete work: 20

Employee's Information is:

Basic Information...:

Name: Mickey

Employee ID: 1121

Gender: F

Department Information...:

Department Name: Testing

Assigned Work: To test login form

Time to complete work: 20

Enter Name: Mickey

Enter Emp. Id: 1121

Enter Gender: F

Enter Loan Details: HOME LOAN

Enter loan amount: 150000

Employee's Information is:

Basic Information...:

Name: Mickey

Employee ID: 1121

Gender: F

Loan Information...:

Loan Details: HOME LOAN

Loan Amount : 150000

1. C++ program to read and print students information using two classes and simple inheritance

#include <iostream>

using namespace std;

class std\_basic\_info { // Base class

private:

char name[30];

int age;

char gender;

public:

void getBasicInfo(void);

void putBasicInfo(void);

};

void std\_basic\_info::getBasicInfo(void) // function definitions

{

cout << "Enter student's basic information:" << endl;

cout << "Name?: ";

cin >> name;

cout << "Age?: ";

cin >> age;

cout << "Gender?: ";

cin >> gender;

}

void std\_basic\_info::putBasicInfo(void)

{

cout << "Name: " << name << ",Age: " << age << ",Gender: " << gender << endl;

}

class std\_result\_info : public std\_basic\_info { // Derived class

private:

int totalM;

float perc;

char grade;

public:

void getResultInfo(void);

void putResultInfo(void);

};

void std\_result\_info::getResultInfo(void) // Function definitions

{

cout << "Enter student's result information:" << endl;

cout << "Total Marks?: ";

cin >> totalM;

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

cout << "Grade?: ";

cin >> grade;

}

void std\_result\_info::putResultInfo(void)

{

cout << "Total Marks: " << totalM << ",Percentage: " << perc << ",Grade: " << grade << endl;

}

int main()

{

std\_result\_info std; // Create object of derived class

// Read student basic and result information

std.getBasicInfo();

std.getResultInfo();

//print student basic and result information

std.putBasicInfo();

std.putResultInfo();

return 0;

}

**Output**

Enter student's basic information:

Name?: Mickey

Age?: 26

Gender?: F

Enter student's result information:

Total Marks?: 455

Grade?: A

Name: Mickey,Age: 26,Gender: F

Total Marks: 455,Percentage: 91,Grade: A

1. Implement the exception handling blocks for both exceptions with a program that can throw integer and double exceptions in the same try block.

#include <iostream>

int main()

{

try

{

std::cout << "Throwing an int exception..." << '\n';

throw 123;

std::cout << "Throwing a double exception..." << '\n';

throw 456.789;

}

catch (int ex)

{

std::cout << "Integer exception: " << ex << " caught and handled."

<< '\n';

}

catch (double ex)

{

std::cout << "Double exception: " << ex << " caught and handled."

<< '\n';

}

}

#### ****Output:****

Throwing an int exception...

Integer exception: 123 caught and handled.

1. Demonstrate the process of working with file handling through a program in c++ to reading and writing data on a file.

#include<iostream>

#include<fstream>

using namespace std;

main()

{

int rno, fee;

char name[50];

cout<<"Enter the Roll Number:";

cin>>rno;

cout<<"\nEnter the Name:";

cin>>name;

cout<<"\nEnter the Fee:";

cin>>fee;

ofstream fout("d:/student.doc");

fout<<rno<<"\t"<<name<<"\t"<<fee; //write data to the file student

fout.close();

ifstream fin("d:/student.doc");

fin>>rno>>name>>fee; //read data from the file student

fin.close();

cout<<endl<<rno<<"\t"<<name<<"\t"<<fee;

return 0;

}