INDEX

Roll No. SC55 - Shreyas Chavhan

===================================================

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Sr. no | Title | Pg.  no | Date  Of  Conduction | Date  Of  Submission | Sign |
| 1 | Implement a class Complex which represents the Complex Number data type. Implement the following operations:  1. A constructor (including a default constructor which creates the complex number 0+0i).  2. Overloaded operator+ to add two complex numbers.  3. Overloaded operator\* to multiply two complex numbers.  4. Overloaded << and >> to print and read Complex Number | 5 |  |  |  |
| 2 | Implement a class Quadratic that represents two degree polynomials i.e.,  polynomials of type ax2+bx+c. Your class will require three data member corresponding  to a, b and c. Implement the following operations:  1.A constructor(including a default constructor which creates the 0 polynomial).  2.Overloaded operator+ to add two polynomials of degree 2.  3.Overloaded << and >> to print and read polynomials.  To do this, you will need to decide what you want your input and output format to look like.  4.A function eval that computes the value of a polynomial for a given value of x.  5.A function that computes the two solutions of the equation ax 2 +bx+c=0 | 10 |  |  |  |
| 3 | Write a C++ program create a calculator for an arithmetic operator (+, -, \*, /). The program  should take two operands from user and performs the operation on those two operands  depending upon the operator entered by user. Use a switch statement to select the operation.  Finally, display the result.  Some sample interaction with the program might look like this:  Enter first number, operator, second number: 10 / 3  Answer = 3.333333  Do another (y/n)? y  Enter first number, operator, second number: 12 + 100  Answer = 112  Do another (y/n)? n | 17 |  |  |  |
| 4 | Develop an object oriented program in C++ to create a  database of student information system containing the  following information: Name, Roll number, Class, division,  Date of Birth, Blood group, Contact address, telephone  number, driving license no. etc Construct the database with  suitable member functions for initializing and destroying the  data with constructor, default constructor, Copy constructor,  destructor, static member functions, friend class, this pointer,  inline function.Code and dynamic memory allocation  operators-new and delete. | 21 |  |  |  |
| 5 | Write C++ Program with base class convert declares two variables, val1  and val2, which hold the initial and converted values, respectively. It also  defines the functions getinit( ) and getconv( ), which return the initial value  and the converted value. Int Val1 = 5 Explicitly convert the int to float  These elements of convert are fixed and applicable to all derived classes that  will inherit convert. However, the function that will actually perform the  conversion, compute ( ), is a pure virtual function that must be defined by  the classes derived from convert. The specific nature of compute ( ) will be  determined by what type of conversion is taking place.  Baseclass getinit() getcnv() | 29 |  |  |  |
| 6 | Create employee bio-data using following classes  i) Personal record  ii))Professional record  iii) Academic record  Assume appropriate data members and member function to accept required data & print bio-data.  Create bio-data using multiple inheritance using C++. | 35 |  |  |  |
| 7 | Create user defined exception to check the following conditions and throw the  exception if the criterion does not meet.  a. User has age between 18 and 55.  b. User stays has income between Rs. 50,000 - Rs. 1,00,000 per month  c. User stays in Pune/Mumbai/Banglore/Chennai  d. User has 4-wheeler  Accept age, Income, City, Vehicle from the user and check for the conditions  mentioned above. If any of the condition not met then throw the expception; | 40 |  |  |  |
| 8 | Assignment No. 8 -  Write a menu driven program that will create a data file containing the list of  telephone numbers in the following form  John 23456  Ahmed 9876  ........................  Use a class object to store each set of data, access the file created and implement  the follwoing tasks  I. Determine the telephone number of specified person  II. Determine the name if telephone number is known  III. Update the telephone number, whenever there is a change. | 43 |  |  |  |
| 9 | Write a C++ program that creates an output file, writes  information on it, closes the file and open it again as an input  file and read the information from the file. | 51 |  |  |  |
| 10 | Write a function template selection sort. Write a program that  inputs, sorts and outputs an integer array and a float array. | 55 |  |  |  |
|  |  |  |  |  |  |

===================================================

Roll No. SC55 - Shreyas Chavhan

Assignment No. 01

Roll No. SC55 - Shreyas Chavhan

===================================================

# Complex Number

## Problem Statement -

Implement a class Complex which represents the Complex Number data type. Implement the following operations:

1. A constructor (including a default constructor which creates the complex number 0+0i).

2. Overloaded operator+ to add two complex numbers.

3. Overloaded operator\* to multiply two complex numbers.

4. Overloaded << and >> to print and read Complex Number

## Outcomes -

To understand operator overloading concept.

## Theory -

### Operator Overloading:

You can redefine or overload most of the built-in operators available in

C++. Thus a programmer can use operators with user-defined types as well.

Overloaded operators are functions with special names the keyword operator followed by the

symbol for the operator being defined. Like any other function, an overloaded operator has a

return type and a parameter list.

Box operator+(const Box&amp;);

declares the addition operator that can be used to add two Box objects and returns final Box

object.

### Overloadable/Non-overloadableOperators:

Following is the list of operators which can be overloaded:

+ - \* / % ^

&amp; | ~ ! , =

Roll No. SC55 - Shreyas Chavhan

===================================================

+= -= /= %= ^= &amp;=

|= \*= &lt;&lt;= &gt;&gt;= [ ] ( )

-&gt; -&gt;\* new new [ ] delete delete [ ]

Operator that are not overloaded are follows

scope operator - ::

sizeof

member selector - .

member pointer selector - \*

ternary operator - ?:

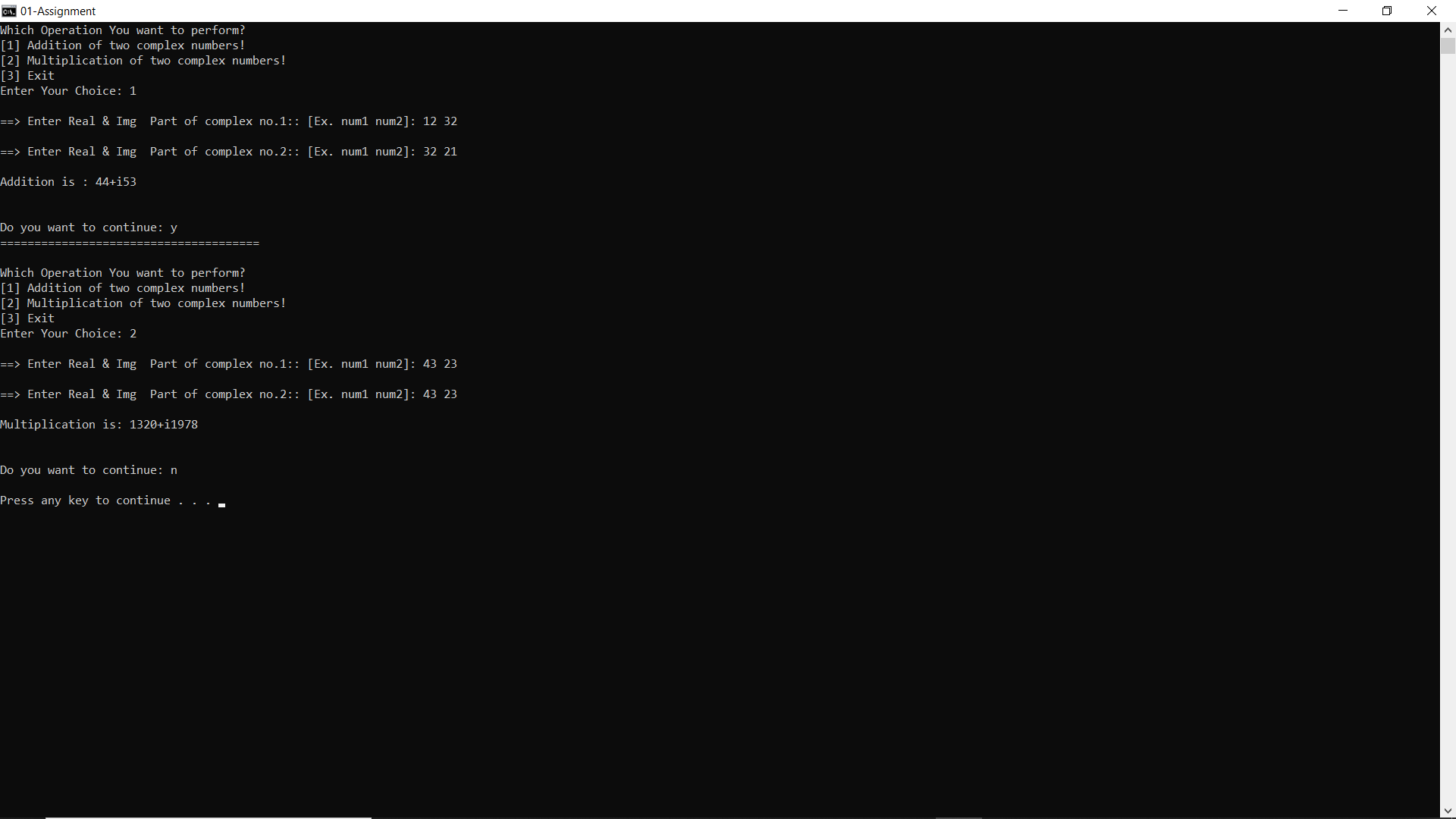
## Code -

|  |
| --- |
| /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Implement a class Complex which represents the Complex Number data type. Implement the following operations:  1. A constructor (including a default constructor which creates the complex number 0+0i).  2. Overloaded operator+ to add two complex numbers.  3. Overloaded operator\* to multiply two complex numbers.  4. Overloaded << and >> to print and read Complex Number  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  #include<iostream>  using namespace std;  class complex  {  float real, imag;  public:  complex(float x1, float x2) // parametrrised consructor  {  real = x1;  imag = x2;  }  complex() // default consructor #1  {  real = imag = 0;  }  friend istream &operator >> (istream &in,complex &t) //operator >> overloaded #4  {  in >> t.real >> t.imag;  return in;  }  complex operator + (complex t) //operator + overloaded #2  {  complex z;  z.real = this->real + t.real;  z.imag = this->imag + t.imag;  return z;  }  complex operator \* (complex t) //operator \* overloaded #3  {  complex z;  z.real = this->real \* t.real-this->imag \* t.imag;  z.imag = this->imag \* t.real+this->real \* t.imag;  return z;  }  friend ostream &operator << (ostream &op,complex &t) //operator << overloaded #4  {  op << t.real << "+i" << t.imag;  return op;  }  };  int main()  {  complex c1; // creates complex no 0+i0(default)  complex c2(3,5); // creates complex no 3+i5(parameterized)  complex c3,c4;  int c;  char ch;  do  {  cout << "======================================\n";  cout << endl;  cout << "Which Operation You want to perform?\n";  cout << "[1] Addition of two complex numbers!\n";  cout << "[2] Multiplication of two complex numbers!\n";  cout << "[3] Exit";  cout << endl;  cout << "Enter Your Choice: ";  cin >> c;  switch(c)  {  case 1:  cout << endl;  cout << "==> Enter Real & Img Part of complex no.1:: [Ex. num1 num2]: ";  cin >> c1;  cout << "\n==> Enter Real & Img Part of complex no.2:: [Ex. num1 num2]: ";  cin >> c2;  c3 = c1 + c2;  cout << endl;  cout << "Addition is : ";  cout << c3 << "\n";  cout << endl;  cout << endl;  break;  case 2:  cout << endl;  cout << "==> Enter Real & Img Part of complex no.1:: [Ex. num1 num2]: ";  cin >> c1;  cout << "\n==> Enter Real & Img Part of complex no.2:: [Ex. num1 num2]: ";  cin >> c2;  cout << endl;  cout << "Multiplication is: ";  c4 = c1 \* c2;  cout << c4 << "\n";  cout << endl;  cout << endl;  break;  case 3:  return 0;  }  cout << "Do you want to continue: ";  cin >> ch;  }while(ch == 'y' || ch == 'Y');  return 0;  } |

Roll No. SC55 - Shreyas Chavhan

===================================================

## Output -



===================================================

Roll No. SC55 - Shreyas Chavhan

Assignment No. 02

Roll No. SC55 - Shreyas Chavhan

===================================================

# Quadratic Class

## Problem Statement -

Implement a class Quadratic that represents two degree polynomials i.e.,

polynomials of type ax2+bx+c. Your class will require three data member corresponding

to a, b and c. Implement the following operations:

1.A constructor(including a default constructor which creates the 0 polynomial).

2.Overloaded operator+ to add two polynomials of degree 2.

3.Overloaded << and >> to print and read polynomials.

To do this, you will need to decide what you want your input and output format to look like.

4.A function eval that computes the value of a polynomial for a given value of x.

5.A function that computes the two solutions of the equation ax 2 +bx+c=0

## Outcomes -

To understand operator overloading concept.

## Theory -

### Operator Overloading:

You can redefine or overload most of the built-in operators available in

C++. Thus a programmer can use operators with user-defined types as well.

Overloaded operators are functions with special names the keyword operator followed by the

symbol for the operator being defined. Like any other function, an overloaded operator has a

return type and a parameter list.

Box operator+(const Box&amp;);

declares the addition operator that can be used to add two Box objects and returns final Box

object.

Roll No. SC55 - Shreyas Chavhan

===================================================

### Overloadable/Non-overloadableOperators:

Following is the list of operators which can be overloaded:

+ - \* / % ^

&amp; | ~ ! , =

+= -= /= %= ^= &amp;=

|= \*= &lt;&lt;= &gt;&gt;= [ ] ( )

-&gt; -&gt;\* new new [ ] delete delete [ ]

Operator that are not overloaded are follows

scope operator - ::

sizeof

member selector - .

member pointer selector - \*

ternary operator - ?:

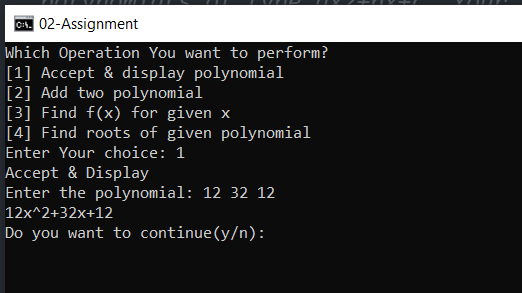
## Code -

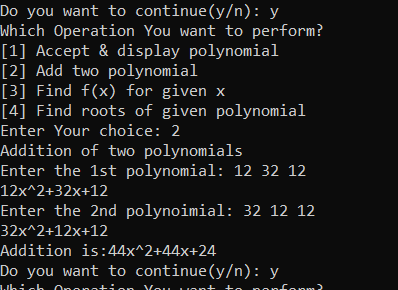
|  |
| --- |
| /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Implement a class Quadratic that represents two degree polynomials i.e.,  polynomials of type ax2+bx+c. Your class will require three data member corresponding  to a, b and c. Implement the following operations:  1.A constructor(including a default constructor which creates the 0 polynomial).  2.Overloaded operator+ to add two polynomials of degree 2.  3.Overloaded << and >> to print and read polynomials.  To do this, you will need to decide what you want your input and output format to look like.  4.A function eval that computes the value of a polynomial for a given value of x.  5.A function that computes the two solutions of the equation ax 2 +bx+c=0  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  #include<iostream>  #include<cmath>  using namespace std;  class Quadratic  {  int a,b,c;  public:  Quadratic() //default constructor  {  a = 0;  b = 0;  c = 0;  }  Quadratic(int x, int y, int z) //parameterized constructor  {  a = x; b = y; c = z;  }  friend istream &operator >> (istream &IN, Quadratic &T) //operator >> overloaded  {  IN >> T.a >> T.b >> T.c;  return IN;  }  friend ostream &operator << (ostream &OUT, Quadratic &T) //operator << overloaded  {  OUT << T.a << "x^2+" << T.b << "x+" << T.c;  return OUT;  }  Quadratic operator + (Quadratic T) //opeartor + overloaded  {  Quadratic R;  R.a = a + T.a;  R.b = b + T.b;  R.c = c + T.c;  return R;  }  void eval(Quadratic T,int x) //eval fn to evaluate the polynomial for given value of x  {  int z;  z = T.a \* x \* x + T.b \* x + T.c;  cout << "for x = " << x << ", f(" << x << ") = " << z << "\n";  }  void compute(Quadratic T) //compute fn to find roots of polynomial  {  float x, y1, y2;  x = T.b \* T.b-4 \* T.a \* T.c;  if(x > 0)  {  cout << "Roots are real & not equal\n";  y1 = (-T.b+sqrt(x)) / (2 \* T.a);  y2 = (-T.b-sqrt(x)) / (2 \* T.a);  cout << y1 << "\n";  cout << y2 << "\n";  }  else if(x == 0)  {  cout << "Roots are real & equal\n";  y1 = -T.a / (2\*T.a);  cout << y1 << "\n";  }  else if(x < 0)  {  cout << "complex Roots\n";  }  }  };  int main()  {  int x, ch;  char p;  Quadratic s1(5,6,10), s2, s3;  do  {  cout << "Which Operation You want to perform?\n";  cout << "[1] Accept & display polynomial\n";  cout << "[2] Add two polynomial\n";  cout << "[3] Find f(x) for given x\n";  cout << "[4] Find roots of given polynomial\n";  cout << "Enter Your choice: ";  cin >> ch;  switch(ch)  {  case 1:  cout << "Accept & Display\n";  cout << "Enter the polynomial: ";  cin >> s2;  cout << s2 << endl;  break;  case 2:  cout << "Addition of two polynomials\n";  cout << "Enter the 1st polynomial: ";  cin >> s1;  cout << s1 << endl;  cout << "Enter the 2nd polynoimial: ";  cin >> s2;  cout << s2 << endl;  s3 = s1 + s2;  cout << "Addition is:";  cout << s3 << endl;  break;  case 3:  cout << "Calculate f(X) for given x\n";  cout << "Enter the polynomial: ";  cin >> s1;  cout << "Enter the value for x = ";  cin >> x;  s2.eval(s1,x);  break;  case 4:  cout << "find roots of polynomial\n";  cout << "enter the polynomial: ";  cin >> s2;  s2.compute(s2);  break;  }  cout << "Do you want to continue(y/n): ";  cin >> p;  }while( p == 'y' || p == 'Y');  return 0;  } |

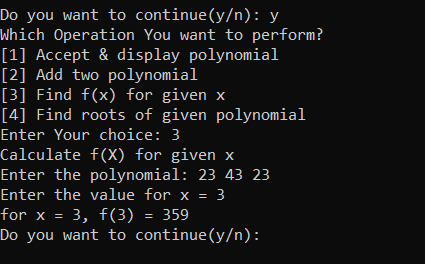
Roll No. SC55 - Shreyas Chavhan

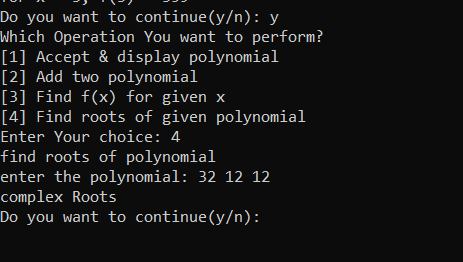
===================================================

## Output -









===================================================

Roll No. SC55 - Shreyas Chavhan

Assignment No. 03

Roll No. SC55 - Shreyas Chavhan

===================================================

# Calculator

## Problem Statement -

Write a C++ program create a calculator for an arithmetic operator (+, -, \*, /). The program

should take two operands from user and performs the operation on those two operands

depending upon the operator entered by user. Use a switch statement to select the operation.

Finally, display the result.

Some sample interaction with the program might look like this:

Enter first number, operator, second number: 10 / 3

Answer = 3.333333

Do another (y/n)? y

Enter first number, operator, second number: 12 + 100

Answer = 112

Do another (y/n)? n

## Theory -

## Constructors are special class functions which perform initialization of every object. The Compiler calls the Constructor whenever an object is created. Constructors initialize values to object members after storage is allocated to the object.

## Whereas, Destructor on the other hand is used to destroy the class object.

## 

Types of Constructors in C++

## Constructors are of three types:

## Default Constructor -

Default Constructors

## Default constructor is the constructor which doesn't take any argument. It has no parameter.

Syntax:

class\_name(parameter1, parameter2, …….){

// constructor definition

}

A default constructor is so important for initialization of object members, that even if we do not define a constructor explicitly, the compiler will provide a default constructor implicitly.

## 

1. Parameterized Constructor -

## These are the constructors with parameters. Using this Constructor you can provide different values to data members of different objects, by passing the appropriate values as argument.

1. Copy Constructor -

## These are special types of Constructors which takes an object as argument, and is used to copy values of data members of one object into another object

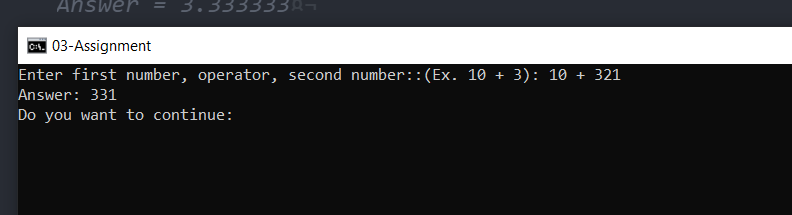
## Code -

|  |
| --- |
| /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Write a C++ program create a calculator for an arithmetic operator (+, -, \*, /). The program  should take two operands from user and performs the operation on those two operands  depending upon the operator entered by user. Use a switch statement to select the operation.  Finally, display the result.  Some sample interaction with the program might look like this:  Enter first number, operator, second number: 10 / 3  Answer = 3.333333  Do another (y/n)? y  Enter first number, operator, second number: 12 + 100  Answer = 112  Do another (y/n)? n  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  #include<iostream>  using namespace std;  int main()  {  char c;  float n1,n2;  char ch;  do  {  cout<<"Enter first number, operator, second number::(Ex. 10 + 3): ";  cin>> n1 >> c >> n2;  switch (c)  {  case'+':  cout<<"Answer: " << n1+n2<<"\n";  break;  case'-':  cout<<"Answer: " << n1-n2<<"\n";  break;  case'\*':  cout<<"Answer: " <<n1\*n2<<"\n";  break;  case'/':  cout<<"Answer: " <<n1/n2<<"\n";  break;  default: cout<<"Enter a valid choice";  }  cout << "Do you want to continue: ";  cin >> ch;  }while(ch == 'y' || ch == 'Y');  return 0;  } |

Roll No. SC55 - Shreyas Chavhan

===================================================

## Output -



===================================================

Roll No. SC55 - Shreyas Chavhan

Assignment No. 04

Roll No. SC55 - Shreyas Chavhan

===================================================

# Student Information System

## Problem Statement -

Develop an object oriented program in C++ to create a database of student information system containing the following information: Name, Roll number, Class, division,

Date of Birth, Blood group, Contact address, telephone number, driving license no. etc Construct the database with suitable member functions for initializing and destroying the

data with constructor, default constructor, Copy constructor, destructor, static member functions, friend class, this pointer, inline function.Code and dynamic memory allocation

operators-new and delete.

## Theory -

A special method of the class that will be automatically invoked when an instance of

the class is created is called as constructor. Following are the most useful features of constructor.

1) Constructor is used for Initializing the values to the data members of the Class.

2) Constructor is that whose name is same as name of class.

3) Constructor gets Automatically called when an object of class is created.

4) Constructors never have a Return Type even void.

5) Constructor is of Default, Parameterized and Copy Constructors.

The various types of Constructor are as follows:-

Constructors can be classified into 3 types

1. Default Constructor

2. Parameterized Constructor

3. Copy Constructor

1. Default Constructor:- Default Constructor is also called as Empty Constructor which has

no arguments and It is Automatically called when we creates the object of class but

Remember name of Constructor is same as name of class and Constructor never declared

with the help of Return Type.

2. Parameterized Constructor: - This is another type constructor which has some

Arguments and same name as class name but it uses some Arguments So For this We have

to create object of Class by passing some Arguments at the time of creating object with the

name of class. When we pass some Arguments to the Constructor then this will

automatically pass the Arguments to the Constructor and the values will retrieve by the

Respective Data Members of the Class.

3. Copy Constructor: - This is also another type of Constructor. In this Constructor we pass

the object of class into the Another Object of Same Class. As name Suggests you Copy,

means Copy the values of one Object into the another Object of Class .This is used for

Copying the values of class object into an another object of class So we call them as Copy

Constructor and For Copying the values We have to pass the name of object whose values

we wants to Copying and When we are using or passing an Object to a Constructor then

we must have to use the &amp; Ampersand or Address Operator.

Destructor: As we know that Constructor is that which is used for Assigning Some Values to data

Members and for Assigning Some Values this May also used Some Memory so that to free up the

Memory which is Allocated by Constructor, destructor is used which gets Automatically Called at

the End of Program and we doesn’t have to Explicitly Call a Destructor and Destructor Cant be

Parameterized or a Copy This can be only one Means Default Destructor which Have no

Arguments. For Declaring a destructor we have to use ~tiled Symbol in front of Destructor.

Static members

A class can contain static members, either data or functions.

A static member variable has following properties:

It is initialized to zero when the first object of its class is created. No other initialization is

permitted.

Only one copy of that member is created for the entire class and is shared by all the

objects of that class.

It is the visible only within the class but its lifetime is the entire program.

Static data members of a class are also known as &quot;class variables&quot;, because there is only one

unique value for all the objects of that same class. Their content is not different from one object

static members have the same properties as global variables but they enjoy class scope. For that

reason, and to avoid them to be declared several times, we can only include the prototype (its

declaration) in the class declaration but not its definition (its initialization). In order to initialize a

static data-member we must include a formal definition outside the class, in the global scope of

this class to another. Because it is a unique variable value for all the objects of the same class, it

can be referred to as a member of any object of that class or even directly by the class name (of

course this is only valid for static members.

A static member function has following properties :

4. A static function can have access to only other static members (fun or var) declared in the

same class

5. A static function can be called using the class name instead of its object name

Class\_name :: function\_name;

Static member functions are considered to have class scope. In contrast to non static member

functions, these functions have no implicit this argument; therefore, they can use only static data

members, enumerators, or nested types directly. Static member functions can be accessed without

using an object of the corresponding class type.

The following restrictions apply to such static functions:

They cannot access non static class member data using the member-selection operators

(. or –&gt;).

They cannot be declared as virtual.

They cannot have the same name as a non static function that has the same argument

types.

Ex. Shall we give the example………………………

Friend functions:

In principle, private and protected members of a class cannot be accessed from outside the same

class in which they are declared. However, this rule does not affect friends. Friends are functions

or classes declared as such. If we want to declare an external function as friend of a class, thus

allowing this function to have access to the private and protected members of this class, we do it

by declaring a prototype of this external function within the class, and preceding it with the

keyword friend.

Properties of friend function:

It is not in the scope of the class to which it has been declared as friend.

Since it is not in the scope of the class , it cannot be called using the object of that class

It can be invoked like a normal function w/o the help of any object.

It can be declared in private or in the public part of the class.

Unlike member functions, it cannot access the member names directly and has to use an

object name and dot operator with each member name.

Friend classes

Just as we have the possibility to define a friend function, we can also define a class as friend of

another one, granting that second class access to the protected and private members of the first

one.

Pointers:

A pointer is a derived data type that refers to another data variable by storing the variables

memory address rather than data.

Declaration of pointer variable is in the following form:

Data\_type \* ptr\_var;

Eg. int \* ptr;

Here ptr is a pointer variable and points to an integer data type.

We can initialize pointer variable as follows

int p, \* ptr ; // declaration

ptr = &amp; a ; // initialization

Pointers to objects:

Consider the following example

item P ; // where item is class &amp; P is object

Similarly, we can define a pointer item\_ptr of type item as follows

item \*it\_ptr ;

Object pointers are useful in creating objects at runtime. We can also

access public members of the class using pointers.

Ex. item X;

item \*ptr = &amp;X;

the pointer „ptr „is initialized with address of X.

we can access the member functions and data using pointers as

follows

ptr getdata();

ptr show();

this pointer:

C++ uses a unique keyword called this to represent an object that invokes a member function. this

is a pointer that points to the object for which this function was called. This unique pointer is

automatically passed to a member function when it is called.

Important notes on this pointer:

this pointer stores the address of the class instance, to enable pointer access of the

members to the member functions of the class.

this pointer is not counted for calculating the size of the object.

this pointers are not accessible for static member functions.

this pointers are not modifiable.

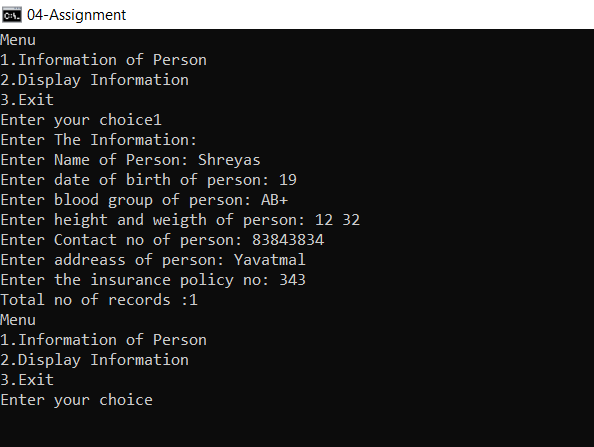
## Code -

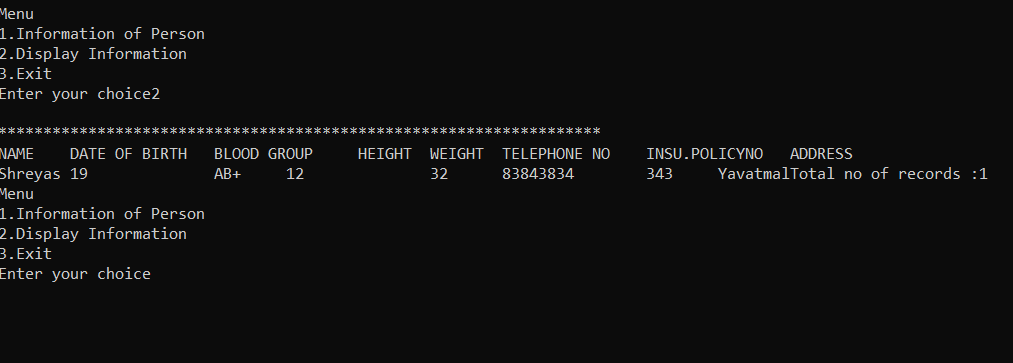
|  |
| --- |
| #include <iostream>  #include <string>  #include <cstring>  using namespace std;  class person  {  private:  char name[40], date\_of\_birth[15], blood\_group[15];  int h, w;  public:  static int count;  friend class personal;  person()  {  char \*name = new char[40];  char \*date\_of\_birth = new char[80];  char \*blood\_group = new char[15];  h = w = 0;  }  static void recordcount()  {  cout << "Total no of records :" << count << endl;  }  };  class personal  {  private:  char add[70], telephone[15], policy\_no[10];  public:  personal()  {  strcpy(add, "");  strcpy(telephone, "");  strcpy(policy\_no, "");  }  void getdata(person \*obj);  void displaydata(person \*obj);  friend class person;  };  int person::count = 0;  void personal::getdata(person \*obj)  {  cout << "Enter Name of Person: ";  cin >> obj -> name;  cout << "Enter date of birth of person: ";  cin >> obj -> date\_of\_birth;  cout << "Enter blood group of person: ";  cin >> obj -> blood\_group;  cout << "Enter height and weigth of person: ";  cin >> obj -> h >> obj -> w;  cout << "Enter Contact no of person: ";  cin >> this -> telephone;  cout << "Enter addreass of person: ";  cin >> this -> add;  cout << "Enter the insurance policy no: ";  cin >> this -> policy\_no;  obj->count++;  }  void personal :: displaydata(person \*obj)  {  cout << obj -> name << "\t"  << obj -> date\_of\_birth << "\t\t"  << obj -> blood\_group << "\t"  << obj -> h << "\t\t"  << obj -> w << "\t"  << this -> telephone << "\t"  << this -> policy\_no << "\t"  << this -> add;  }  int main()  {  personal \*p1[30];  person \*p2[30];  int n = 0, ch, i;  do  {  cout << "Menu" << endl;  cout << "1.Information of Person" << endl;  cout << "2.Display Information" << endl;  cout << "3.Exit" << endl;  cout << "Enter your choice";  cin >> ch;  switch (ch)  {  case 1:  cout << "Enter The Information: " << endl;  p1[n] = new personal;  p2[n] = new person;  p1[n]->getdata(p2[n]);  n++;  person::recordcount();  break;  case 2:  cout << " ";  cout << endl << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;  cout << "NAME"  << "\t"  << "DATE OF BIRTH"  << "\t"  << "BLOOD GROUP"  << "\t"  << "HEIGHT"  << "\t"  << "WEIGHT"  << "\t"  << "TELEPHONE NO"  << "\t"  << "INSU.POLICYNO"  << "\t"  << "ADDRESS ";  cout << endl;  for (i = 0; i < n; i++)  {  p1[i]->displaydata(p2[i]);  }  person::recordcount();  break;  }  } while (ch != 4);  return 0;  } |

Roll No. SC55 - Shreyas Chavhan

===================================================

## Output -





===================================================

Roll No. SC55 - Shreyas Chavhan

Assignment No. 05

Roll No. SC55 - Shreyas Chavhan

===================================================

# Virtual Function (Polymorphism)

## Problem Statement -

Write C++ Program with base class convert declares two variables, val1

and val2, which hold the initial and converted values, respectively. It also

defines the functions getinit( ) and getconv( ), which return the initial value

and the converted value. Int Val1 = 5 Explicitly convert the int to float

These elements of convert are fixed and applicable to all derived classes that

will inherit convert. However, the function that will actually perform the

conversion, compute ( ), is a pure virtual function that must be defined by

the classes derived from convert. The specific nature of compute ( ) will be

determined by what type of conversion is taking place.

Baseclass getinit() getcnv()

## Theory -

Polymorphism in C++

The word polymorphism means having many forms. Typically, polymorphism occurs when there

is a hierarchy of classes and they are related by inheritance.

C++ polymorphism means that a call to a member function will cause a different function to be

executed depending on the type of object that invokes the function.

Virtual Function:

A virtual function is a function in a base class that is declared using the keyword virtual.

Defining in a base class a virtual function, with another version in a derived class, signals to the

compiler that we don&#39;t want static linkage for this function.

Late Binding

In Late Binding function call is resolved at runtime. Hence, now compiler determines the type of

object at runtime, and then binds the function call. Late Binding is also called Dynamic Binding

or Runtime Binding.

What we do want is the selection of the function to be called at any given point in the program to

be based on the kind of object for which it is called. This sort of operation is referred to as

dynamic linkage, or late binding.

Problem without Virtual Keyword

class Base

{

public:

void show()

{

cout &lt;&lt; &quot;Base class&quot;;

}

};

class Derived:public Base

{

public:

void show()

{

cout &lt;&lt; &quot;Derived Class&quot;;

}

}

int main()

{

Base\* b; //Base class pointer

Derived d; //Derived class object

b = &amp;d;

b-&gt;show(); //Early Binding Ocuurs

}

Output : Base class

When we use Base class&#39;s pointer to hold Derived class&#39;s object, base class pointer or reference

will always call the base version of the function

Using Virtual Keyword

We can make base class&#39;s methods virtual by using virtual keyword while declaring them. Virtual

keyword will lead to Late Binding of that method.

class Base

{

public:

virtual void show()

{

cout &lt;&lt; &quot;Base class&quot;;

}

};

class Derived:public Base

{

public:

void show()

{

cout &lt;&lt; &quot;Derived Class&quot;;

}

}

int main()

{

Base\* b; //Base class pointer

Derived d; //Derived class object

b = &amp;d;

b-&gt;show(); //Late Binding Ocuurs

}

Output : Derived class

On using Virtual keyword with Base class&#39;s function, Late Binding takes place and the derived

version of function will be called, because base class pointer pointes to Derived class object.

Pure Virtual Functions:

It&#39;s possible that you&#39;d want to include a virtual function in a base class so that it may be redefined

in a derived class to suit the objects of that class, but that there is no meaningful definition you

could give for the function in the base class.

class Shape

{

protected:

int width, height;

public:

Shape( int a=0, int b=0)

{

width = a;

height = b;

}

// pure virtual function

virtual int area() = 0;

};

The = 0 tells the compiler that the function has no body and above virtual function will be called

pure virtual function.

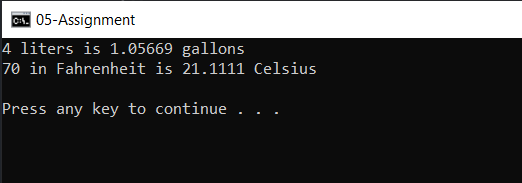
## Code -

|  |
| --- |
| /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Write C++ Program with base class convert declares two variables, val1  and val2, which hold the initial and converted values, respectively. It also  defines the functions getinit( ) and getconv( ), which return the initial value  and the converted value. Int Val1 = 5 Explicitly convert the int to float  These elements of convert are fixed and applicable to all derived classes that  will inherit convert. However, the function that will actually perform the  conversion, compute ( ), is a pure virtual function that must be defined by  the classes derived from convert. The specific nature of compute ( ) will be  determined by what type of conversion is taking place.  Baseclass getinit() getcnv()  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  #include <iostream>  using namespace std;  class convert  {  protected:  double val1;  double val2;  public:  convert(double i)  {  val1 = i;  }  double getconv()  {  return val2;  }  double getinit()  {  return val1;  }  virtual void compute() = 0;  };  class l\_to\_g : public convert  {  public:  l\_to\_g(double i) : convert(i)  {  }  void compute()  {  val2 = val1 / 3.7854;  }  };  // Fahrenheit to Celsius  class f\_to\_c : public convert  {  public:  f\_to\_c(double i) : convert(i)  {  }  void compute()  {  val2 = (val1 - 32) / 1.8;  }  };  int main()  {  convert \*p;  l\_to\_g lgob(4);  f\_to\_c fcob(70);  p = &lgob;  cout << p->getinit() << " liters is ";  p->compute();  cout << p->getconv() << " gallons\n";  p = &fcob;  cout << p->getinit() << " in Fahrenheit is ";  p->compute();  cout << p->getconv() << " Celsius\n";  return 0;  } |

Roll No. SC55 - Shreyas Chavhan

===================================================

## Output -



===================================================

Roll No. SC55 - Shreyas Chavhan

Assignment No. 06

Roll No. SC55 - Shreyas Chavhan

===================================================

# Implement Employee Database, Bio-data

## Problem Statement -

Create employee bio-data using following classes

i) Personal record

ii)) Professional record

iii) Academic record

Assume appropriate data members and member functions to accept required data & print bio-data.

Create bio-data using multiple inheritance using C++.

## Theory -

In C++, it is possible to inherit attributes and methods from one class to another. We group the "inheritance concept" into two categories:

* derived class (child) - the class that inherits from another class
* base class (parent) - the class being inherited from

To inherit from a class, use the : symbol.

In the example below, the Car class (child) inherits the attributes and methods from the Vehicle class (parent):

### Example

// Base class

class Vehicle {

public:

string brand = "Ford";

void honk() {

cout << "Tuut, tuut! \n" ;

}

};

// Derived class

class Car: public Vehicle {

public:

string model = "Mustang";

};

int main() {

Car myCar;

myCar.honk();

cout << myCar.brand + " " + myCar.model;

return 0;

}

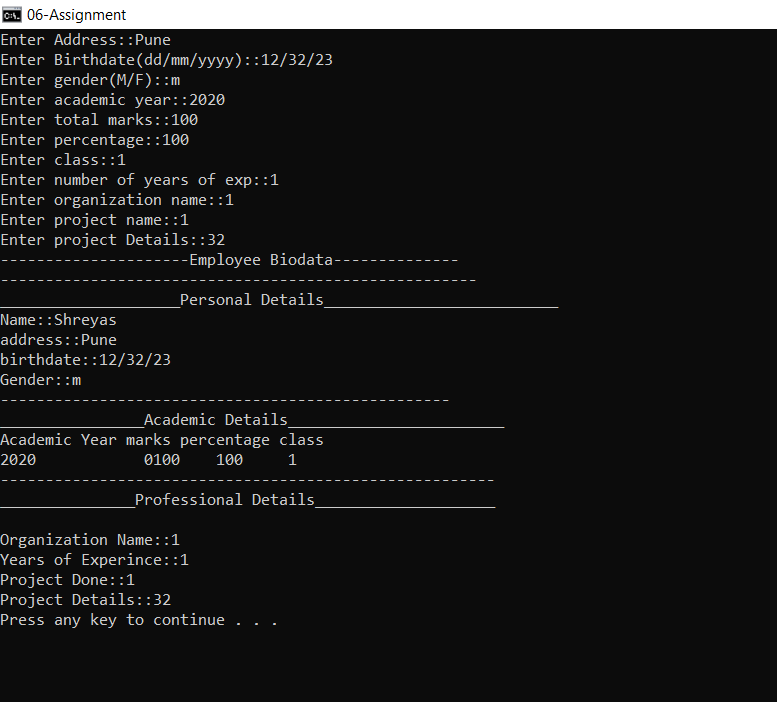
## Code -

|  |
| --- |
| /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Create employee bio-data using following classes  i) Personal record  ii))Professional record  iii) Academic record  Assume appropriate data members and member function to accept required data & print bio-data.  Create bio-data using multiple inheritance using C++.  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  #include <iostream>  using namespace std;  class personal  {  protected:  char name[50];  char address[50];  char birthdate[50];  char gender;  public:  void get\_personal();  };  class professional  {  protected:  int noofyearsexp;  char orgname[50];  char projname[50];  char projdetails[50];  public:  void get\_professional();  };  class academic  {  protected:  int year;  int marks;  int percentage;  char Class[50];  public:  void get\_academic();  };  class biodata : public personal, public academic, public professional  {  public:  void display();  };  void personal::get\_personal()  {  cout << "Enter name::";  cin >> name;  cout << "Enter Address::";  cin >> address;  cout << "Enter Birthdate(dd/mm/yyyy)::";  cin >> birthdate;  cout << "Enter gender(M/F)::";  cin >> gender;  }  void professional::get\_professional()  {  cout << "Enter number of years of exp::";  cin >> noofyearsexp;  cout << "Enter organization name::";  cin >> orgname;  cout << "Enter project name::";  cin >> projname;  cout << "Enter project Details::";  cin >> projdetails;  }  void academic::get\_academic()  {  cout << "Enter academic year::";  cin >> year;  cout << "Enter total marks::";  cin >> marks;  cout << "Enter percentage::";  cin >> percentage;  cout << "Enter class::";  cin >> Class;  }  void biodata::display()  {  cout << "---------------------Employee Biodata--------------" << endl;  cout << "-----------------------------------------------------" << endl;  cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Personal Details\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;  cout << "Name::" << name << endl;  cout << "address::" << address << endl;  cout << "birthdate::" << birthdate << endl;  cout << "Gender::" << gender << endl;  cout << "--------------------------------------------------" << endl;  cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Academic Details\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;  cout << "Academic Year "  << "marks "  << "percentage "  << "class " << endl;  cout << year << "\t\t0" << marks << "\t" << percentage << "\t" << Class << endl;  cout << "-------------------------------------------------------" << endl;  cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Professional Details\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;  cout << "\nOrganization Name::" << orgname;  cout << "\nYears of Experince::" << noofyearsexp;  cout << "\nProject Done::" << projname;  cout << "\nProject Details::" << projdetails;  }  int main()  {  biodata b;  b.get\_personal();  b.get\_academic();  b.get\_professional();  b.display();  } |

Roll No. SC55 - Shreyas Chavhan

===================================================

## Output -



===================================================

Roll No. SC55 - Shreyas Chavhan

Assignment No. 07

Roll No. SC55 - Shreyas Chavhan

===================================================

# Exception Handling

## Problem Statement -

Create user defined exception to check the following conditions and throw the

exception if the criterion does not meet.

a. User has age between 18 and 55.

b. User stays has income between Rs. 50,000 - Rs. 1,00,000 per month

c. User stays in Pune/Mumbai/Banglore/Chennai

d. User has 4-wheeler

Accept age, Income, City, Vehicle from the user and check for the conditions

mentioned above. If any of the condition not met then throw the expception;

## Theory -

C++ Exceptions

When executing C++ code, different errors can occur: coding errors made by the programmer, errors due to wrong input, or other unforeseeable things.

When an error occurs, C++ will normally stop and generate an error message. The technical term for this is: C++ will throw an exception (throw an error).

C++ try and catch

Exception handling in C++ consist of three keywords: try, throw and catch:

The try statement allows you to define a block of code to be tested for errors while it is being executed.

The throw keyword throws an exception when a problem is detected, which lets us create a custom error.

The catch statement allows you to define a block of code to be executed, if an error occurs in the try block.

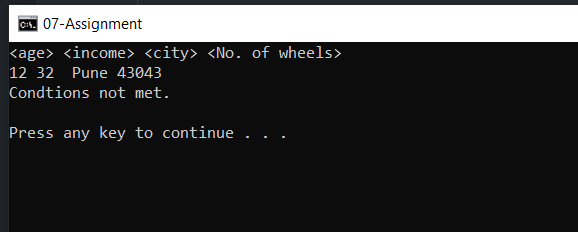
## Code -

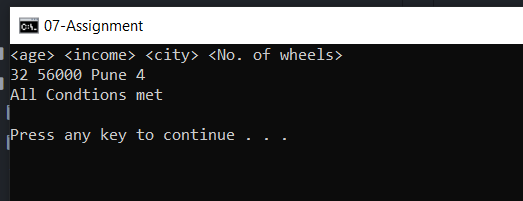
|  |
| --- |
| /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Assignment no. 7 -  Create user defined exception to check the following conditions and throw the  exception if the criterion does not meet.  a. User has age between 18 and 55.  b. User stays has income between Rs. 50,000 - Rs. 1,00,000 per month  c. User stays in Pune/Mumbai/Banglore/Chennai  d. User has 4-wheeler  Accept age, Income, City, Vehicle from the user and check for the conditions  mentioned above. If any of the condition not met then throw the expception;  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  #include <iostream>  using namespace std;  int main(){  int age, income, vehicle\_wheels;  string city;  cout << "<age> <income> <city> <No. of wheels>" << endl;  cin >> age >> income >> city >> vehicle\_wheels;  try{  if(age > 18 && age < 55 && income > 50000 && income < 100000 && vehicle\_wheels == 4 && (city == "Pune" || city == "Mumbai" || city == "Banglore" || city == "Chennai")){  cout << "All Condtions met" << endl;  }  else{  throw age;  }  }  catch(int age){  cout << "Condtions not met." << endl;  }  return 0;  } |

Roll No. SC55 - Shreyas Chavhan

===================================================

## Output -





===================================================

Roll No. SC55 - Shreyas Chavhan

Assignment No. 08

Roll No. SC55 - Shreyas Chavhan

===================================================

# File Handling

## Problem Statement -

Write a menu driven program that will create a data file containing the list of

telephone numbers in the following form

John 23456

Ahmed 9876

........................

Use a class object to store each set of data, access the file created and implement

the following tasks

I. Determine the telephone number of specified person

II. Determine the name if telephone number is known

III. Update the telephone number, whenever there is a change.

## Theory -

Stream:

A stream is a sequence of bytes. It acts as source from which the input data can be

obtained or as a destination to which the output data can be sent.

1. Input Stream

Input Streams are used to hold input from a data producer, such as a keyboard, a

file, or a network. The source stream that provides data to the program is called

the input stream. A program extracts the bytes from the input stream. In most

cases the standard input device is the keyboard. With the cin and “extraction”

operator ( &gt;&gt;) it is possible to read input from the keyboard.

2. Output Stream

Output Streams are used to hold output for a particular data consumer, such as a

monitor, a file, or a printer. The destination stream that receives data from the

program is called the output stream. A program inserts the bytes into an output

stream. By default, the standard output of a program points at the screen. So

with the cout operator and the “insertion” operator (&lt;&lt;) you can print a message

onto the screen.

iostream standard library provides cin and cout methods for reading from standard

input and writing to standard output respectively.

file handling provides three new data types:

Data Type Description

ofstream -

This data type represents the output file stream and is used to

create files and to write information to files.

ifstream -

This data type represents the input file stream and is used to read

information from files.

fstream -

This data type represents the file stream generally, and has the

capabilities of both ofstream and ifstream which means it can

create files, write information to files, and read information from

files.

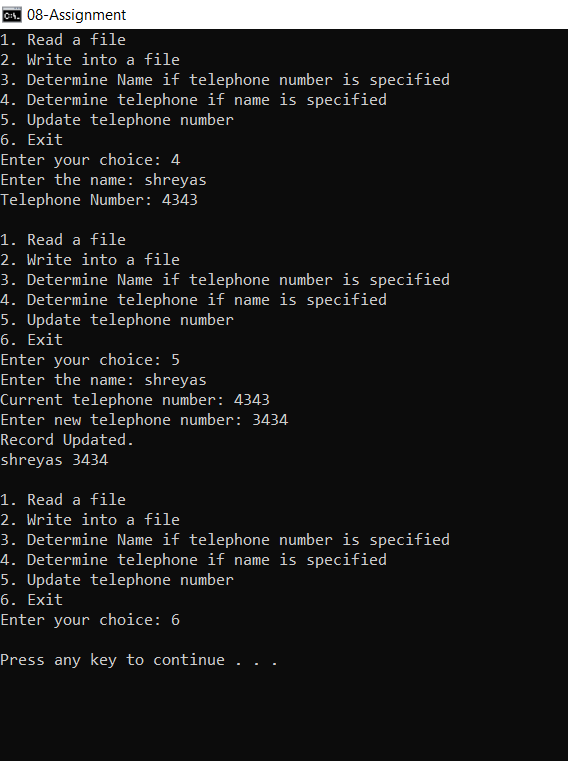
## Code -

|  |
| --- |
| /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Assignment No. 8 -  Write a menu driven program that will create a data file containing the list of  telephone numbers in the following form  John 23456  Ahmed 9876  ........................  Use a class object to store each set of data, access the file created and implement  the follwoing tasks  I. Determine the telephone number of specified person  II. Determine the name if telephone number is known  III. Update the telephone number, whenever there is a change.  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  #include <iostream>  #include <fstream>  #include <string.h>  using namespace std;  class PhoneBook{  public:  long int telephone;  string name;  void input(){  cout << "<Name> <telephone>" << endl;  cin >> name >> telephone;  }  void output(){  cout << name << " " << telephone << endl;  }  };  int main(){  PhoneBook record;  long int telephone\_number, choice, pos, offset, i;  string name;  fstream fp;  ifstream in;  ofstream out;  while(true){  cout << endl;  cout << "1. Read a file" << endl;  cout << "2. Write into a file" << endl;  cout << "3. Determine Name if telephone number is specified" << endl;  cout << "4. Determine telephone if name is specified" << endl;  cout << "5. Update telephone number" << endl;  cout << "6. Exit" << endl;  cout << "Enter your choice: ";  cin >> choice;  switch (choice){  case 1:  in.open("PhoneBook.txt", ios::in | ios::binary);  cout << "The contents of file are: " << endl;  while(in.read((char\*)&record, sizeof(record))){  record.output();  }  in.close();  break;  case 2:  record.input();  char ch;  cin.get(ch);  out.open("PhoneBook.txt", ios::out | ios::app | ios::binary);  out.write((char\*)&record, sizeof(record));  out.close();  break;  case 3:  cout << "Enter the phone number: ";  cin >> telephone\_number;  fp.open("PhoneBook.txt", ios::ate | ios::in | ios::out | ios::binary);  fp.seekg(0, ios::beg);  pos = -1;  i = 0;  while(fp.read((char\*)&record, sizeof(record))){  if(telephone\_number == record.telephone){  pos = i;  break;  }  i++;  }  offset = pos \* sizeof(record);  fp.seekp(offset);  fp.read((char\*)&record, sizeof(record));  cout << "Name: " << record.name << endl;  fp.close();  break;  case 4:  cout << "Enter the name: ";  cin >> name;  fp.open("PhoneBook.txt", ios::ate | ios::in | ios::out | ios::binary);  fp.seekg(0, ios::beg);  pos = -1;  i = 0;  while(fp.read((char\*)&record, sizeof(record))){  if(!name.compare(record.name)){  pos = i;  break;  }  i++;  }  offset = pos \* sizeof(record);  fp.seekp(offset);  fp.read((char\*)&record, sizeof(record));  cout << "Telephone Number: " << record.telephone << endl;  fp.close();  break;  case 5:  cout << "Enter the name: ";  cin >> name;  fp.open("PhoneBook.txt", ios::ate | ios::in | ios::out | ios::binary);  fp.seekg(0, ios::beg);  pos = -1;  i = 0;  while(fp.read((char\*)&record, sizeof(record))){  if(!name.compare(record.name)){  pos = i;  break;  }  i++;  }  offset = pos \* sizeof(record);  fp.seekp(offset);  cout << "Current telephone number: " << record.telephone << endl;  cout << "Enter new telephone number: ";  cin >> telephone\_number;  record.telephone = telephone\_number;  fp.write((char\*)&record, sizeof(record)) << flush;  cout << "Record Updated." << endl;  fp.seekg(0);  while(fp.read((char\*)&record, sizeof(record))){  record.output();  }  fp.close();  break;  case 6:  return 0;  default:  cout << "Invalid Choice" << endl;  }  }  } |

Roll No. SC55 - Shreyas Chavhan

===================================================

## Output -



===================================================

Roll No. SC55 - Shreyas Chavhan

Assignment No. 09

Roll No. SC55 - Shreyas Chavhan

===================================================

# File Handling in C++

## Problem Statement -

Assignment No. 9 -

Write a C++ program that creates an output file, writes

information on it, closes the file and open it again as an input

file and read the information from the file.

## Theory -

Stream:

A stream is a sequence of bytes. It acts as source from which the input data can be

obtained or as a destination to which the output data can be sent.

1. Input Stream

Input Streams are used to hold input from a data producer, such as a keyboard, a

file, or a network. The source stream that provides data to the program is called

the input stream. A program extracts the bytes from the input stream. In most

cases the standard input device is the keyboard. With the cin and “extraction”

operator ( &gt;&gt;) it is possible to read input from the keyboard.

2. Output Stream

Output Streams are used to hold output for a particular data consumer, such as a

monitor, a file, or a printer. The destination stream that receives data from the

program is called the output stream. A program inserts the bytes into an output

stream. By default, the standard output of a program points at the screen. So

with the cout operator and the “insertion” operator (&lt;&lt;) you can print a message

onto the screen.

iostream standard library provides cin and cout methods for reading from standard

input and writing to standard output respectively.

file handling provides three new data types:

Data Type Description

ofstream -

This data type represents the output file stream and is used to

create files and to write information to files.

ifstream -

This data type represents the input file stream and is used to read

information from files.

fstream -

This data type represents the file stream generally, and has the

capabilities of both ofstream and ifstream which means it can

create files, write information to files, and read information from

files.

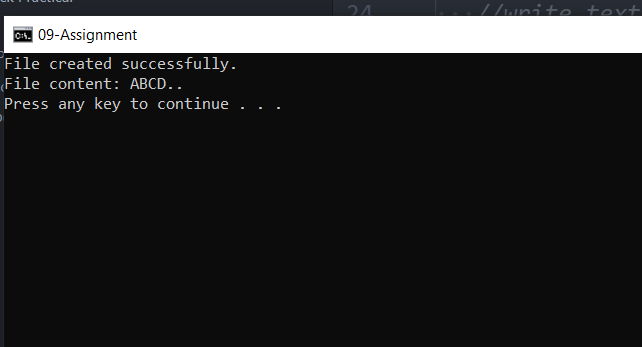
## Code -

|  |
| --- |
| /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Assignment No. 9 -  Write a C++ program that creates an output file, writes  information on it, closes the file and open it again as an input  file and read the information from the file.  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  #include <iostream>  #include <fstream>  using namespace std;  int main()  {  fstream file;  file.open("sample.txt",ios::out);  if(!file)  {  cout<<"Error in creating file!!!"<<endl;  return 0;  }  cout<<"File created successfully."<<endl;  //write text into file  file<<"ABCD.";  //closing the file  file.close();  //again open file in read mode  file.open("sample.txt",ios::in);  if(!file)  {  cout<<"Error in opening file!!!"<<endl;  return 0;  }  //read untill end of file is not found.  char ch; //to read single character  cout<<"File content: ";  while(!file.eof())  {  file>>ch; //read single character from file  cout<<ch;  }  file.close(); //close file  return 0;  } |

Roll No. SC55 - Shreyas Chavhan

===================================================

## Output -



===================================================

Roll No. SC55 - Shreyas Chavhan

Assignment No. 10

Roll No. SC55 - Shreyas Chavhan

===================================================

# Selection Sort Using Template

## Problem Statement -

Write a function template selection sort. Write a program that

inputs, sorts and outputs an integer array and a float array.

## Theory -

Algorithm Selection Sort:

Selection(A, N)

Step 1 − Set MIN to location 0

Step 2 − Search the minimum element in the list

Step 3 − Swap with value at location MIN

Step 4 − Increment MIN to point to next element

Step 5 − Repeat until list is sorted

TEMPLATES -

A template is a simple and yet very powerful tool in C++. The simple idea is to pass data type as a parameter so that we don’t need to write the same code for different data types. For example, a software company may need sort() for different data types. Rather than writing and maintaining the multiple codes, we can write one sort() and pass data type as a parameter.

C++ adds two new keywords to support templates: *‘template’* and *‘typename’*. The second keyword can always be replaced by the keyword ‘class’.

**How templates work?**

Templates are expanded at compile time. This is like macros. The difference is, the compiler does type checking before template expansion. The idea is simple, source code contains only function/class, but compiled code may contain multiple copies of the same function/class.

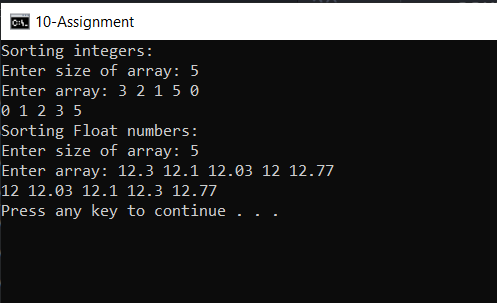
## Code -

|  |
| --- |
| /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Assignment No. 10 -  Write a function template selection sort. Write a program that  inputs, sorts and outputs an integer array and a float array.  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  #include <bits/stdc++.h>  using namespace std;  template <class T>  void selection\_sort(T arr[], int n){  T temp;  for(int i = 0; i < n; i++){  for(int j = i + 1; j < n; j++){  if(arr[i] > arr[j]){  temp = arr[i];  arr[i] = arr[j];  arr[j] = temp;  }  }  }  for(int i = 0; i < n; i++){  cout << arr[i] << " ";  }  }  int main(){  cout << "Sorting integers: " << endl;  int n;  cout << "Enter size of array: ";  cin >> n;  int int\_arr[n];  cout << "Enter array: ";  for(int i = 0; i < n; i++){  cin >> int\_arr[i];  }  selection\_sort(int\_arr, n);  cout << endl;  cout << "Sorting Float numbers: " << endl;  int m;  cout << "Enter size of array: ";  cin >> m;  float float\_arr[m];  cout << "Enter array: ";  for(int i = 0; i < m; i++){  cin >> float\_arr[i];  }  selection\_sort(float\_arr, m);  return 0;  } |

Roll No. SC55 - Shreyas Chavhan

===================================================

## Output -



===================================================

Roll No. SC55 - Shreyas Chavhan