**Program for data types, variables and basic program structure**

***Aim : To convert Temperature from degree Celsius to degree Fahrenheit using classes***

***Algorithm :***

1. *Set value of C.*
2. *Use the formula F=32+(9/5)\*C;*
3. *Print the value of degree Celsius*.

***Program :***

#include<iostream>

using namespace std;

class tempconv{

public :

void setcel(int c){

celc = c;

}

protected :

int celc;

};

class tempcon : public tempconv{

public :

float gettemp(){

return (9/5\*celc)+32;

}

};

int main()

{

tempcon t1;

float t;

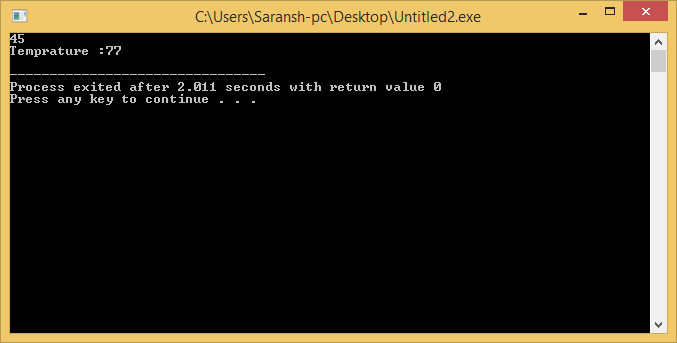
cin>>t;

t1.setcel(t);

cout<<"Temprature :"<<t1.gettemp()<<endl;

}

***Output :***



***Result :*** The program compiled and executed successfully.

**Program for data types, variables and basic program structure**

***Aim :To Add two complex numbers.***

***Algorithm :***

1. *Enter real part of First complex number.*
2. *Enter imaginary part of First complex number.*
3. *Enter real part of second complex number.*
4. *Enter Imaginary part of second complex number.*
5. *Add real part of first number with real part of second number.*
6. *Add imaginary part of first number with imaginary part of second number.*
7. *Print the result.*

***Program :***

#include <iostream>

using namespace std;

class complex

{

private:

int real;

int imag;

public:

void setcomplex( int r , int i)

{

real = r;

imag = i;

}

complex operator+(const complex& b)

{

complex c;

c.real = this->real + b.real;

c.imag = this->imag + b.imag;

return c;

}

void getcomplex(complex x)

{

cout<<x.real<<" + i"<<x.imag<<endl;

}

};

int main( )

{

complex c1, c2, c3;

int x,y;

cout<<"Enter the real and imaginary part of the first complex number\n";

cin>>x>>y;

c1.setcomplex(x,y);

cout<<"Enter the real and imaginary part of the Second complex number\n";

cin>>x>>y;

c2.setcomplex(x,y);

c3=c1+c2;

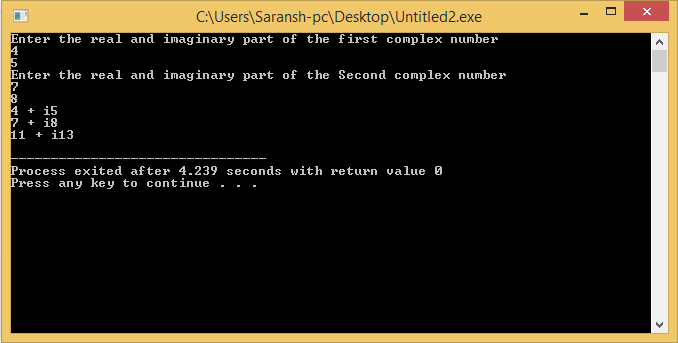
c1.getcomplex(c1);

c2.getcomplex(c2);

c3.getcomplex(c3);

}

***Output :***



***Result :*** The program Compiled and executed successfully.

**Control structures and loops**

***Aim : To calculate sum of number using class in C++;***

***Algorithm :***

1. *Get input x from keyboard till press 0;*
2. *Set sum=sum+x;*
3. *Print sum;*

***Program :***

#include<iostream>

using namespace std;

class sum{

int sum=0;

public :

int a;

int sum1(int x){

sum=sum+x;

}

int get(){

a=sum;

return a;

}

};

int main(){

int x;

sum s;

cout<<"Enter number \n";

do{

cin>>x;

s.sum1(x);

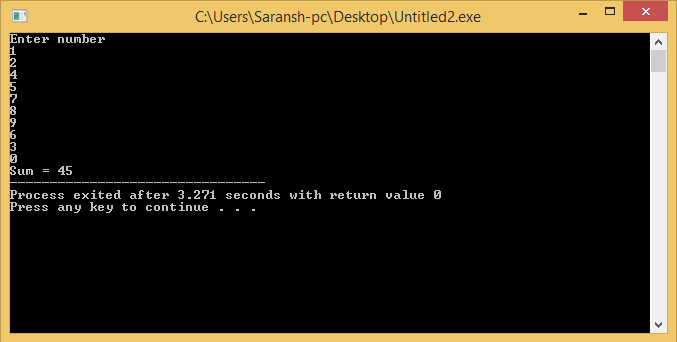
}

while(x!=0)

;cout<<"Sum = "<<s.get();

}

***Output :***



***Result*** : The program compiled and executed successfully.

**Control structures and loops**

***Aim : Factorial using classes and objects.***

***Algorithm :***

1. *Get value of x.*
2. *Pass value to fac(x).*
3. *Set i=1,*
4. *Repeat till i<=x;*
5. *Factorial = factorial\*I;*
6. *Return factorial.*

***Program :***

#include<iostream>

using namespace std;

class facto

{

private :

int factorial=1;

public :

int a;

int fac(int x)

{

int i;

for(i=1;i<=x;i++)

{

factorial = factorial\*i;

}

}

int get()

{

a=factorial;

return a;

}

};

int main()

{

facto f;

int x;

cout<<"Enter the number : ";

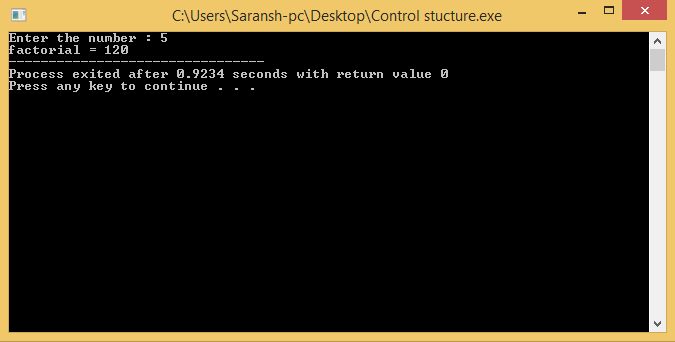
cin>>x;

f.fac(x);

cout<<"factorial = "<<f.get();

}

***Output :***



***Result :*** The program compiled and executed successfully.

**Control structures and loops**

***Aim : To calculate greatest of three numbers using if else.***

***Algorithm :***

1. *Get three numbers from keyboard.*
2. *Check the condition*
3. *If(a>=b&&a>=c)*
4. *Else If(b>=a&&b>=c)*
5. *Else If(c>=a&&c>=b)*
6. *For the true condition return the corresponding value.*

***Program :***

#include<iostream>

using namespace std;

class greatest {

public:

int g;

int get(){

g=great;

return g;

}

int great1(int a, int b, int c){

if(a>=b&&a>=c)

great = a;

else if(b>=a&&b>=c)

great=b;

else if(c>=a&&c>=b)

great = c;

}

private : int great;

};

int main(){

greatest g1;

int a,b,c;

cout<<"Enter three numbers \n";

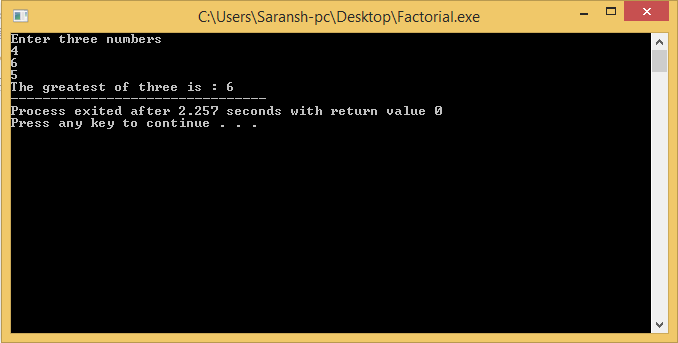
cin>>a>>b>>c;

g1.great1(a,b,c);

cout<<"The greatest of three is : "<<g1.get();

}

***Output :***



***Result :*** The program compiled and executed successfully.

**Inheritance**

***Aim : To implement Class inheritance in C++***

***Algorithm*** :

1. *Pass the value of height and width;*
2. *Calculate height \* width.*
3. *Return the product.*
4. *Print the result.*

***Program:***

#include <iostream>

using namespace std;

class Shape

{

public:

void setWidth(int w)

{

width = w;

}

void setHeight(int h)

{

height = h;

}

protected:

int width;

int height;

};

class PaintCost

{

public:

int getCost(int area)

{

return area \* 70;

}

};

class Rectangle: public Shape, public PaintCost

{

public:

int getArea()

{

return (width \* height);

}

};

int main(void)

{

Rectangle Rect;

int area;

Rect.setWidth(5);

Rect.setHeight(7);

area = Rect.getArea();

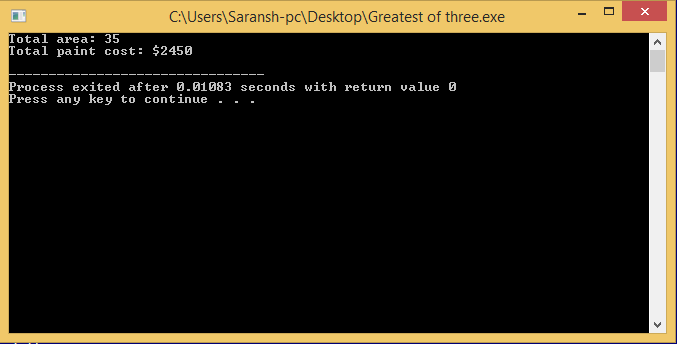
cout << "Total area: " << Rect.getArea() << endl;

cout << "Total paint cost: $" << Rect.getCost(area) << endl;

return 0;

}

***Output***:



***Result*** : The program compiled and executed successfully.

**Polymorphism**

***Aim : To calculate area of triangle and rectangle using polymorphism in C++.***

***Algorithm***:

1. *Set the value of width and height ;;*
2. *Pass both values in the respective functions Ex. Rec(a,b), and Tri(a,b).*
3. *Print the values of both.*

***Program*** :

#include <iostream>

using namespace std;

class Shape {

protected:

int width, height;

public:

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

{

width = a;

height = b;

}

virtual int area()

{

cout << "Parent class area :" <<endl;

return 0;

}

};

class Rectangle: public Shape{

public:

Rectangle( int a=0, int b=0):Shape(a, b) { }

int area ()

{

cout << "Rectangle class area :"<<width\*height <<endl;

return (width \* height);

}

};

class Triangle: public Shape{

public:

Triangle( int a=0, int b=0):Shape(a, b) { }

int area ()

{

cout << "Triangle class area :" <<width\*height/2<<endl;

return (width \* height / 2);

}

};

int main( )

{

Shape \*shape;

Rectangle rec(10,7);

Triangle tri(10,5);

shape = &rec;

shape->area();

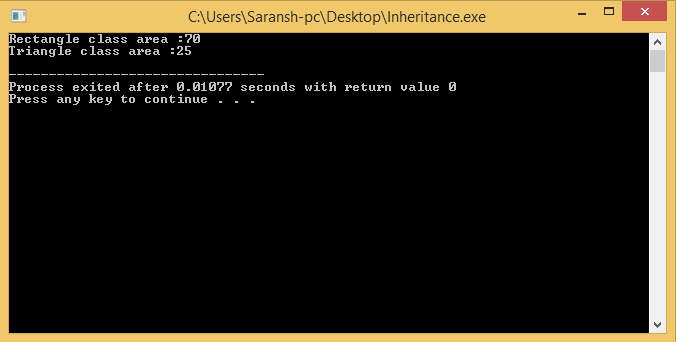
shape = &tri;

shape->area();

return 0;

}

***Output*** :



Result : The program compiled and executed successfully.

**Operator Overloading**

***Aim : To implement operator overloading to calculate sum of two complex number.***

***Algorithm***:

1. *Enter real part of First complex number.*
2. *Enter imaginary part of First complex number.*
3. *Enter real part of second complex number.*
4. *Enter Imaginary part of second complex number.*
5. *Add real part of first number with real part of second number.*
6. *Add imaginary part of first number with imaginary part of second number.*

*Print the result.*

***Program*** :

#include <iostream>

using namespace std;

class complex

{

private:

int real;

int imag;

public:

void setcomplex( int r , int i)

{

real = r;

imag = i;

}

complex operator+(const complex& b)

{

complex c;

c.real = this->real + b.real;

c.imag = this->imag + b.imag;

return c;

}

void getcomplex(complex x)

{

cout<<x.real<<" + i"<<x.imag<<endl;

}

};

int main( )

{

complex c1, c2, c3;

int x,y;

cout<<"Enter the real and imaginary part of the first complex number\n";

cin>>x>>y;

c1.setcomplex(x,y);

cout<<"Enter the real and imaginary part of the Second complex number\n";

cin>>x>>y;

c2.setcomplex(x,y);

c3=c1+c2;

c1.getcomplex(c1);

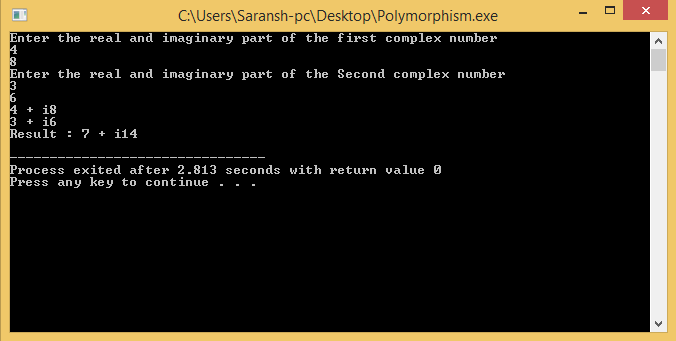
c2.getcomplex(c2);

cout<<"Result : ";

c3.getcomplex(c3);

}

***Output*** :



***Result*** : The program compiled and executed successfully.

**Inline Function**

***Aim : To implement inline function in C++***

***Algorithm*** :

1. *Create a inline function ;*
2. *Create a class box.*
3. *Create object.*
4. *Call inlne function using the object*.

***Program*** :

#include <iostream>

using namespace std;

inline int Max(int x, int y)

{

return (x > y)? x : y;

}

int main( )

{

cout << "Max (20,10): " << Max(20,10) << endl;

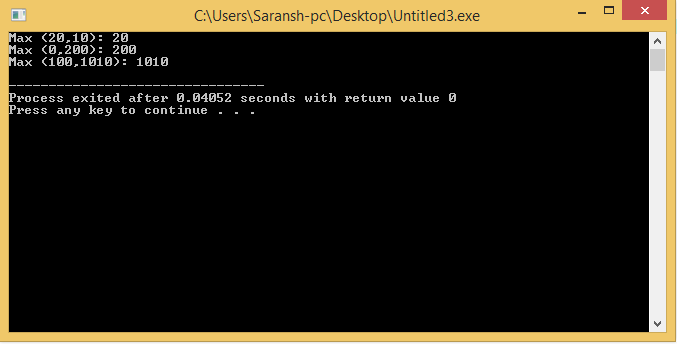
cout << "Max (0,200): " << Max(0,200) << endl;

cout << "Max (100,1010): " << Max(100,1010) << endl;

return 0;

}

***Output*** :



***Result*** : The program compiled and executed successfully.

**Friend Function**

***Aim : To write a program to implement Friend function in C++.***

***Algorithm*** ;

1. *Declare a friend function*
2. *Get width from keyboard.*
3. *Pass width to witdh() function.*
4. *Print the value.*

***Program*** *:*

#include <iostream>

using namespace std;

class Box

{

double width;

public:

friend void printWidth( Box box );

void setWidth( double wid );

};

void Box::setWidth( double wid )

{

width = wid;

}

void printWidth( Box box )

{

cout << "Width of box : " << box.width <<endl;

}

int main( )

{

float x;

Box box;

cout<<"Enter width : ";

cin>>x;

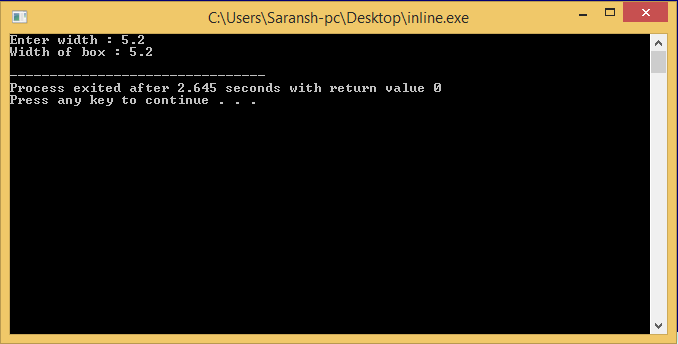
box.setWidth(x);

printWidth( box );

return 0;

}

***Output*** :



***Result*** : The program compiled and executed successfully.

**Virtual function**

***Aim : To implement virtual function in C++.***

***Algorithm***:

1. *Set the value of width and height ;;*
2. *Pass both values in the respective functions Ex. Rec(a,b), and Tri(a,b).*
3. *Print the values of both.*

***Program*** :

#include <iostream>

using namespace std;

class Shape {

protected:

int width, height;

public:

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

{

width = a;

height = b;

}

virtual int area()

{

cout << "Parent class area :" <<endl;

return 0;

}

};

class Rectangle: public Shape{

public:

Rectangle( int a=0, int b=0):Shape(a, b) { }

int area ()

{

cout << "Rectangle class area :"<<width\*height <<endl;

return (width \* height);

}

};

class Triangle: public Shape{

public:

Triangle( int a=0, int b=0):Shape(a, b) { }

int area ()

{

cout << "Triangle class area :" <<width\*height/2<<endl;

return (width \* height / 2);

}

};

int main( )

{

Shape \*shape;

Rectangle rec(10,7);

Triangle tri(10,5);

shape = &rec;

shape->area();

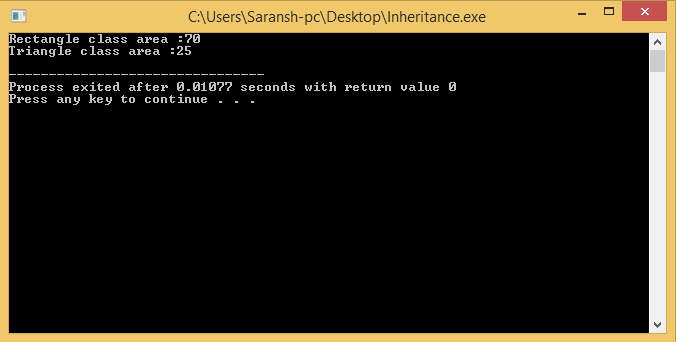
shape = &tri;

shape->area();

return 0;

}

***Output*** :



Result : The program compiled and executed successfully.

**File Manupulation**

***Aim : To implement file handeling using classes in C++.***

***Algotithm*** :

1. *Create a file using ofstream pointer.*
2. *Scan data from keyboard and write to file.*
3. *Now open file in read mode using ifstream.*
4. *Read file and write on screen.*

***Program*** :

#include<iostream>

#include<string.h>

#include<fstream>

using namespace std;

class employee

{

int ENO;

string ENAME;

public :

void GETIT()

{

ofstream f;

f.open("data.txt");

cout<<"Enter the ID Number\n";

cin>>ENO;

fflush(stdin);

f<<ENO<<endl;

cout<<"Enter the Employee name\n";

getline(cin,ENAME);

f<<ENAME<<endl;

f.close();

}

void SHOWIT()

{

string x;

ifstream f;

f.open("data.txt");

f>>x;

cout<<"Employee number : "<<x<<endl;

f>>x;

cout<<"Employee Name : "<<x;

}

};

int main()

{

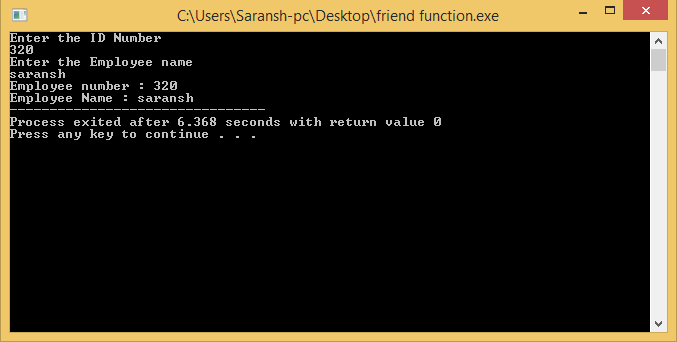
employee e;

e.GETIT();

e.SHOWIT();

}

***Output*** :



***Result*** : The program compiled and executed successfully.

***Manipulating pointers***

***Aim : To implement pointer airthmatics in C++.***

***Algorithm :***

1. *Declare a pointer type ptr.*
2. *Declare an array.*
3. *Assign pointer to array.*
4. *Increment pointer and print the value.*
5. *Decrement pointer and print the value.*

***Program*** :

#include <iostream>

using namespace std;

const int MAX = 3;

int main ()

{

int var[MAX] = {10, 100, 200};

int \*ptr;

ptr = &var[MAX-1];

cout<<"Decrementation using pointer.\n";

for (int i = MAX; i > 0; i--)

{

cout << "Address of var[" << i << "] = ";

cout << ptr << endl;

cout << "Value of var[" << i << "] = ";

cout << \*ptr << endl;

// point to the previous location

ptr--;

}

cout<<"\n\nIncrementation using pointer\n";

ptr = var;

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

{

cout << "Address of var[" << i << "] = ";

cout << ptr << endl;

cout << "Value of var[" << i << "] = ";

cout << \*ptr << endl;

// point to the next location

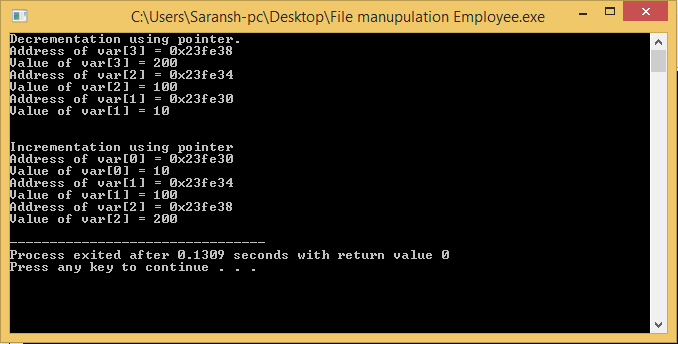
ptr++;

}

return 0;

}

***Output*** :



***Result*** : The program was compiled and executed successfully.

**Templates**

***Aim*** : To implement Template using C++.

***Algorithm*** :

1. *Declare a template.*
2. *template <class type> ret-type func-name(parameter list)*

*{*

*// body of function*

*}*

1. *Print the values.*

***Program*** :

#include <iostream>

#include <string>

using namespace std;

template <typename T>

inline T const& Max (T const& a, T const& b)

{

return a < b ? b:a;

}

int main ()

{

int i = 39;

int j = 20;

cout << "Max(i, j): " << Max(i, j) << endl;

double f1 = 13.5;

double f2 = 20.7;

cout << "Max(f1, f2): " << Max(f1, f2) << endl;

string s1 = "Hello";

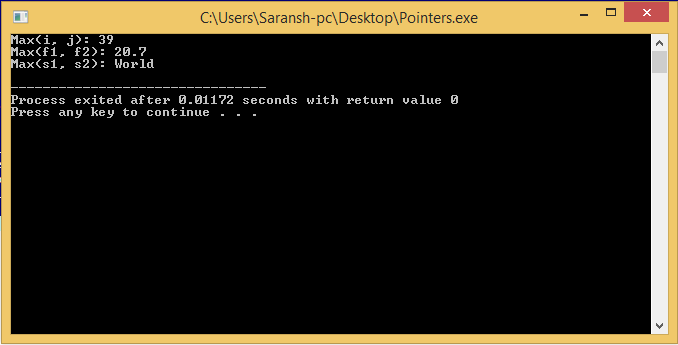
string s2 = "World";

cout << "Max(s1, s2): " << Max(s1, s2) << endl;

return 0;

}

***Output*** :



***Result*** : The program was compiled and executed successfully.

**Exceptions handling**

***Aim : To implement Exception Handling in C++.***

***Comcept*** :

*An exception is a problem that arises during the execution of a program. A C++ exception is a response to an exceptional circumstance that arises while a program is running, such as an attempt to divide by zero.*

***Program*** :

#include <iostream>

using namespace std;

double division(int a, int b)

{

if( b == 0 )

{

throw "Division by zero condition!";

}

return (a/b);

}

int main ()

{

int x = 50;

int y = 0;

double z = 0;

try {

z = division(x, y);

cout << z << endl;

}catch (const char\* msg) {

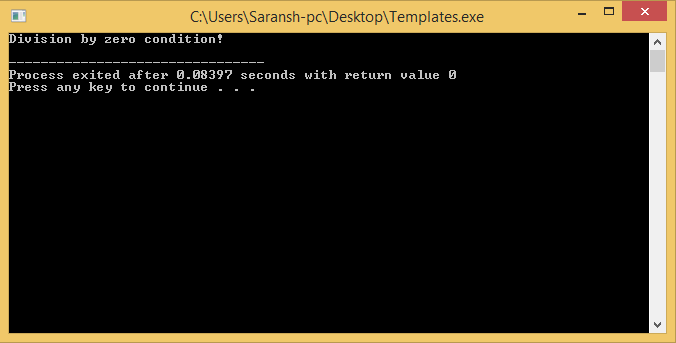
cerr << msg << endl;

}

return 0;

}

***Output*** :



***Result*** : The program was compiled and executed successfully.

**STL for Sequential containers and iterators**

***Aim: To implement C++ programs to construct a STL for Sequential containers and iterators***

***Algorithm*** :

1. *Insert element*
2. *If element already present ,*
3. *Skip it.*
4. *Else*
5. *Insert the element into the list.*
6. *Exit*

***Program*** :

#include <iostream>

#include <iterator>

#include <set>

using namespace std;

int main(void)

{

set<int> intset;

for(int i = 0; i < 2; i++) /\* Even if 2 times inserted, the set will accept only one set of values not duplicate ones \*/

for(int j = 0; j < 24; j++)

intset.insert(j);

cout <<"Set Size "<<intset.size()<<endl;

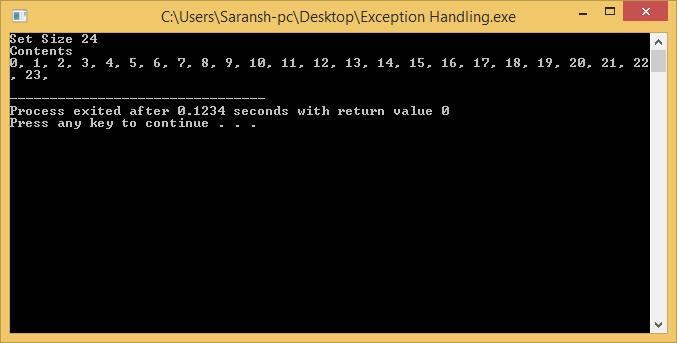
cout <<"Contents"<<endl;

copy(intset.begin(),intset.end(),ostream\_iterator<int>(cout, ", "));

cout<<"\n";

}

***Output*** :



***Result*** : The program compiled and executed successfully.

**STL for Associative containers**

***Aim: To implement C++ programs to construct a STL for Associative containers***

***Algorithm:***

1. *Create employee with different number and names.*
2. *Repeat step 3 till ii !=emlpoyees.end();*
3. *Print the employees details.*
4. *Exit().*

***Program*** :

#include <string.h>

#include <iostream>

#include <map>

#include <utility>

using namespace std;

int main()

{

map<int, string> Employees;

// 1) Assignment using array index notation

Employees[5234] = "Mike C.";

Employees[3374] = "Charlie M.";

Employees[1923] = "David D.";

Employees[7582] = "John A.";

Employees[5328] = "Peter Q.";

cout << "Employees[3374]=" << Employees[3374] << endl << endl;

cout << "Map size: " << Employees.size() << endl;

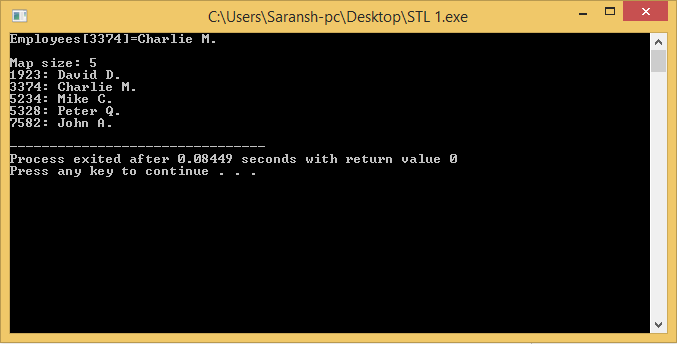
for( map<int,string>::iterator ii=Employees.begin(); ii!=Employees.end(); ++ii){

cout << (\*ii).first << ": " << (\*ii).second << endl;

}

}

***Output :***



***Result*** : The program compiled and executed successfully.