DSA 0136 OBJECT ORIENTED PROGRAMMING USING C++

EXCEPTION HANDLING PROGRAMS

1)Write a c++program for divide by zero

#include<iostream>

using namespace std;

double division(int a,int b)

{

if(b==0)

{

throw"division by zero";

}

return (a/b);

}

int main()

{

int x=50,y=0;

double z=0;

try

{

z=division(x,y);

cout<<z<<endl;

}

catch(const char\*msg)

{

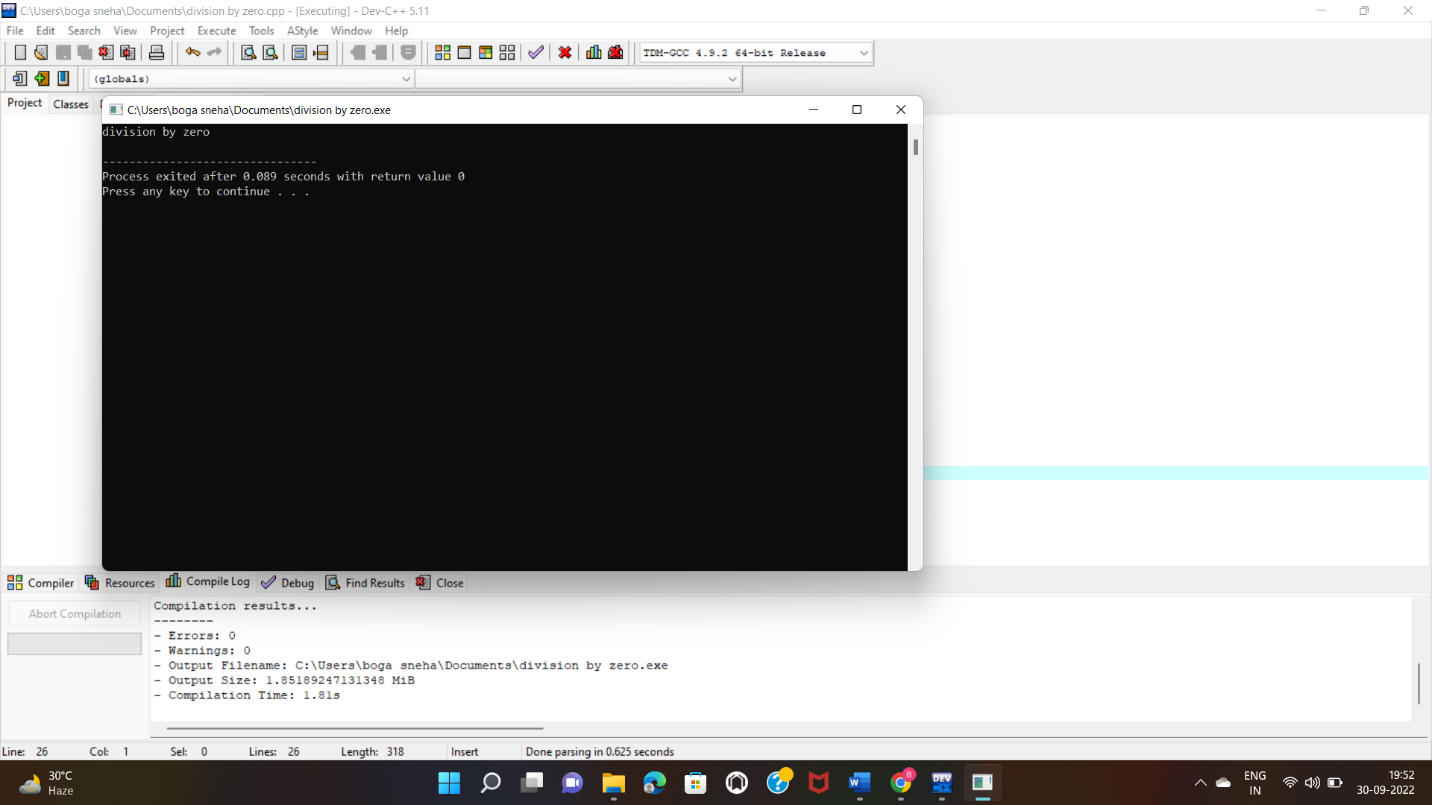
cerr<<msg<<endl;

}

return 0;

}

Output:



2)using namespace std;

#include<iostream>

int main()

{

int x=-1;

try

{

if(x<0)

{

throw x;

cout<<"x is too small";

}

}

catch (int x)

{

cout<<"\nx is negative value";

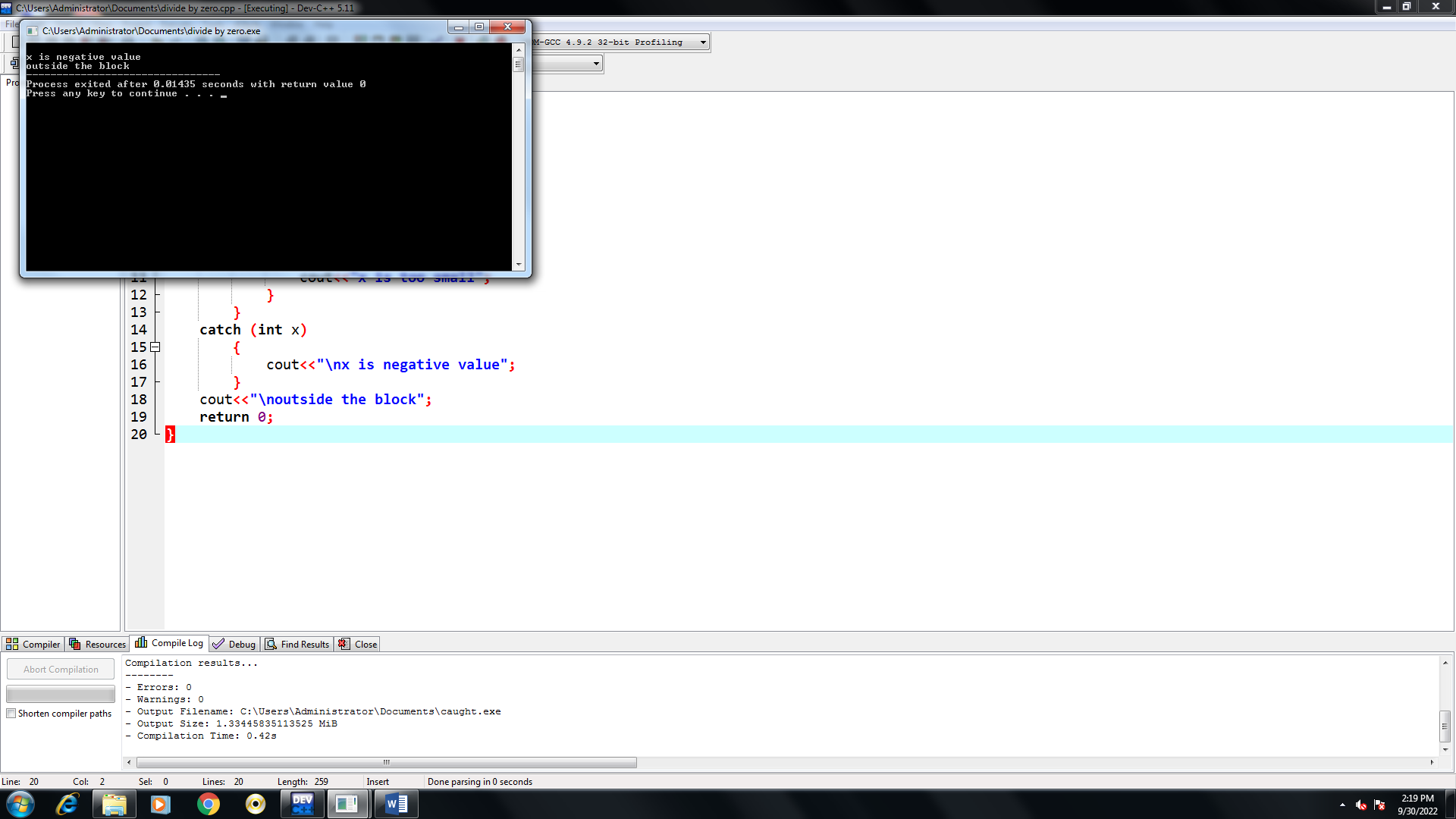
}

cout<<"\noutside the block";

return 0;

}

Output:



3) using namespace std;

#include<iostream>

int main()

{

try

{

throw 10;

}

catch (char \*exp)

{

cout<<"caught";

}

catch (int a)

{

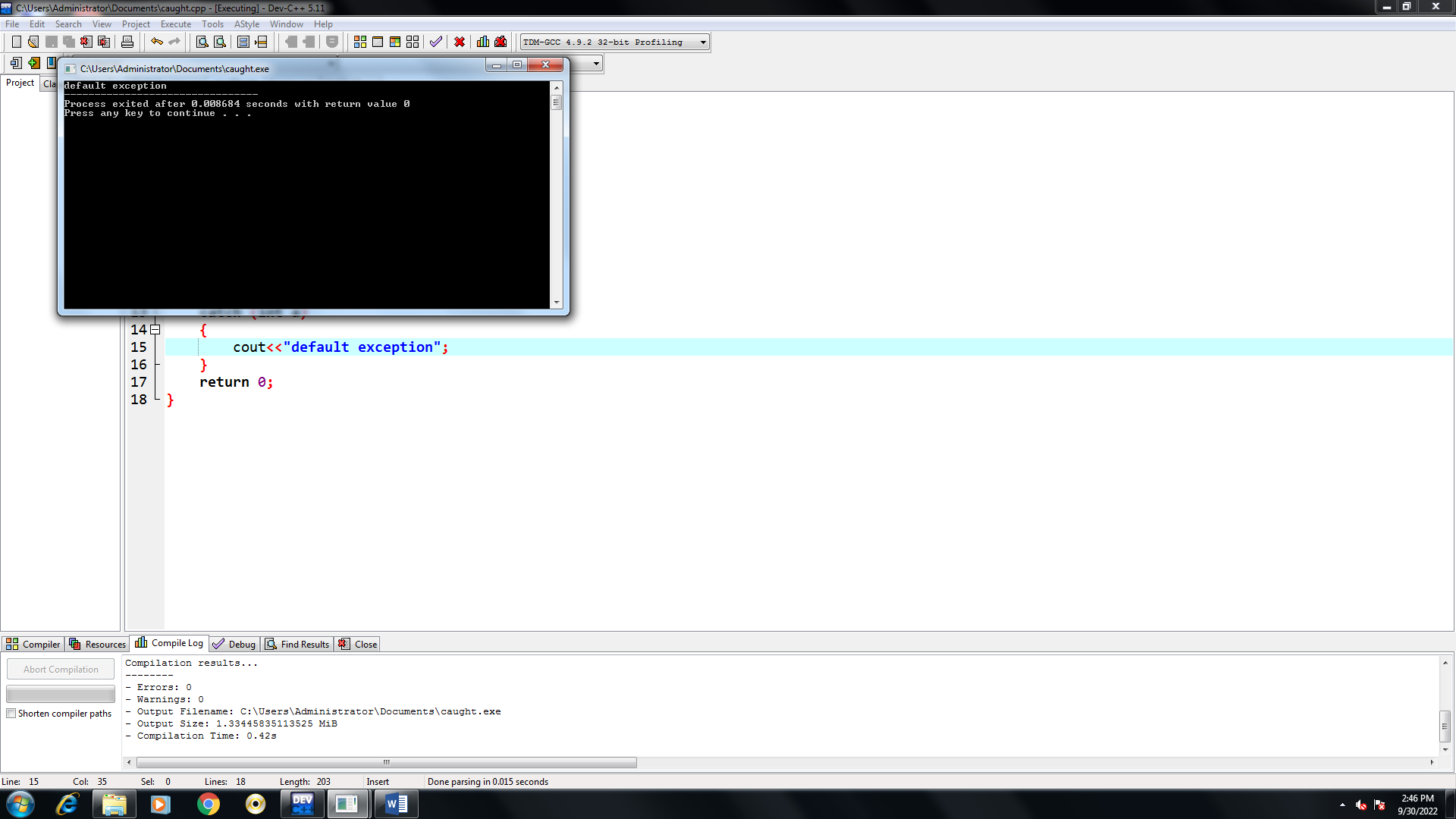
cout<<"default exception";

}

return 0;

}

Output:



3)using namespace std;

#include<iostream>

#include<exception>

struct myexception:public exception

{

const char\*what() const throw()

{

return "c++exception";

}

};

int main()

{

try

{

throw myexception();

}

catch(myexception & e)

{

std::cout<<"my exception caught"<<std::endl;

std::cout<<e.what()<<std::endl;

}

catch(std::exception &e)

{

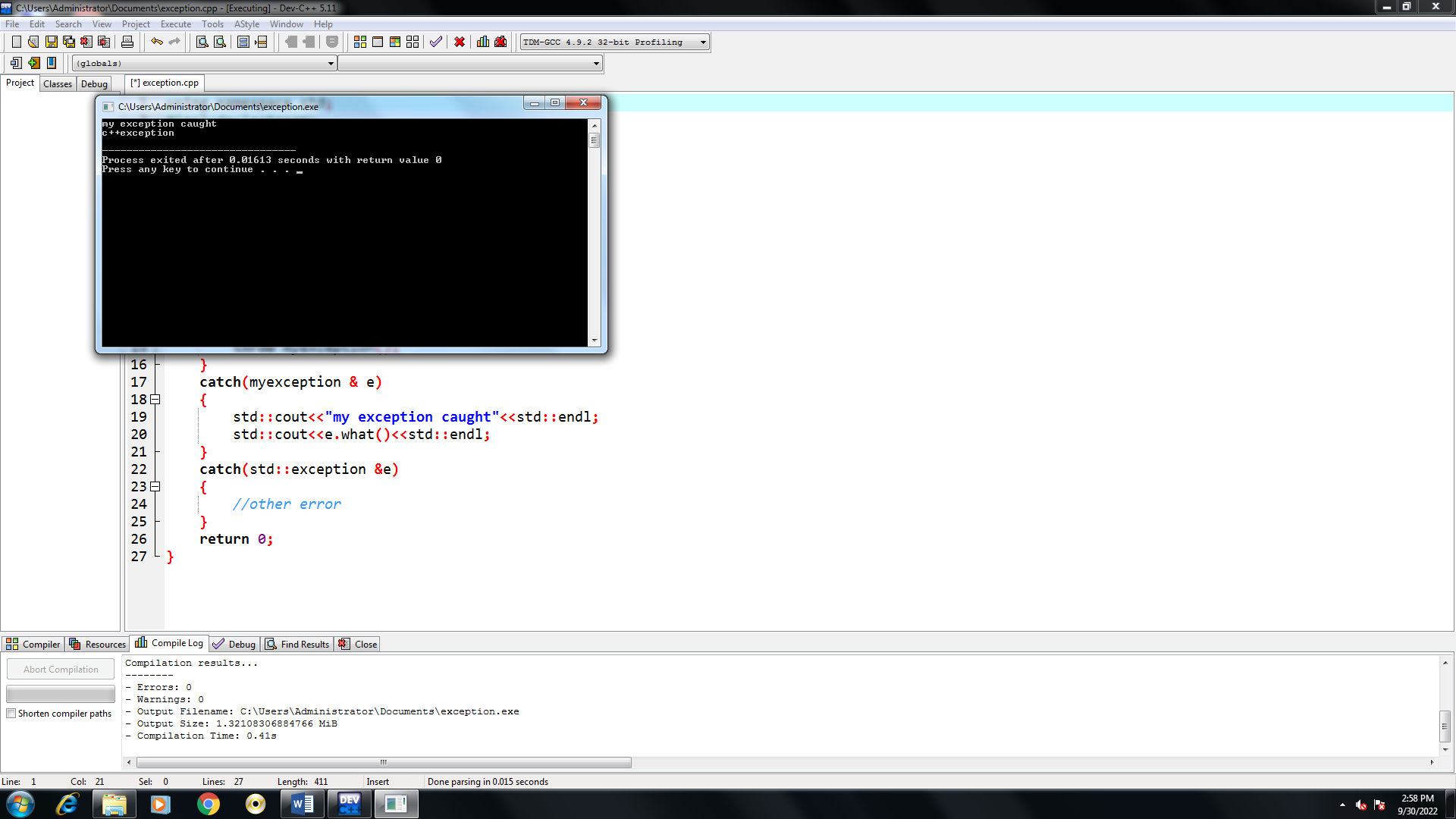
//other error

}

return 0;

}

Output:



4)FUNCTION OVERLOADING

#include<iostream>

using namespace std;

class A

{

public:

void getdata()

{

cout<<"\nA";

}

};

class B:public A

{

public:

void getdata()

{

cout<<"\nB";

}

};

int main()

{

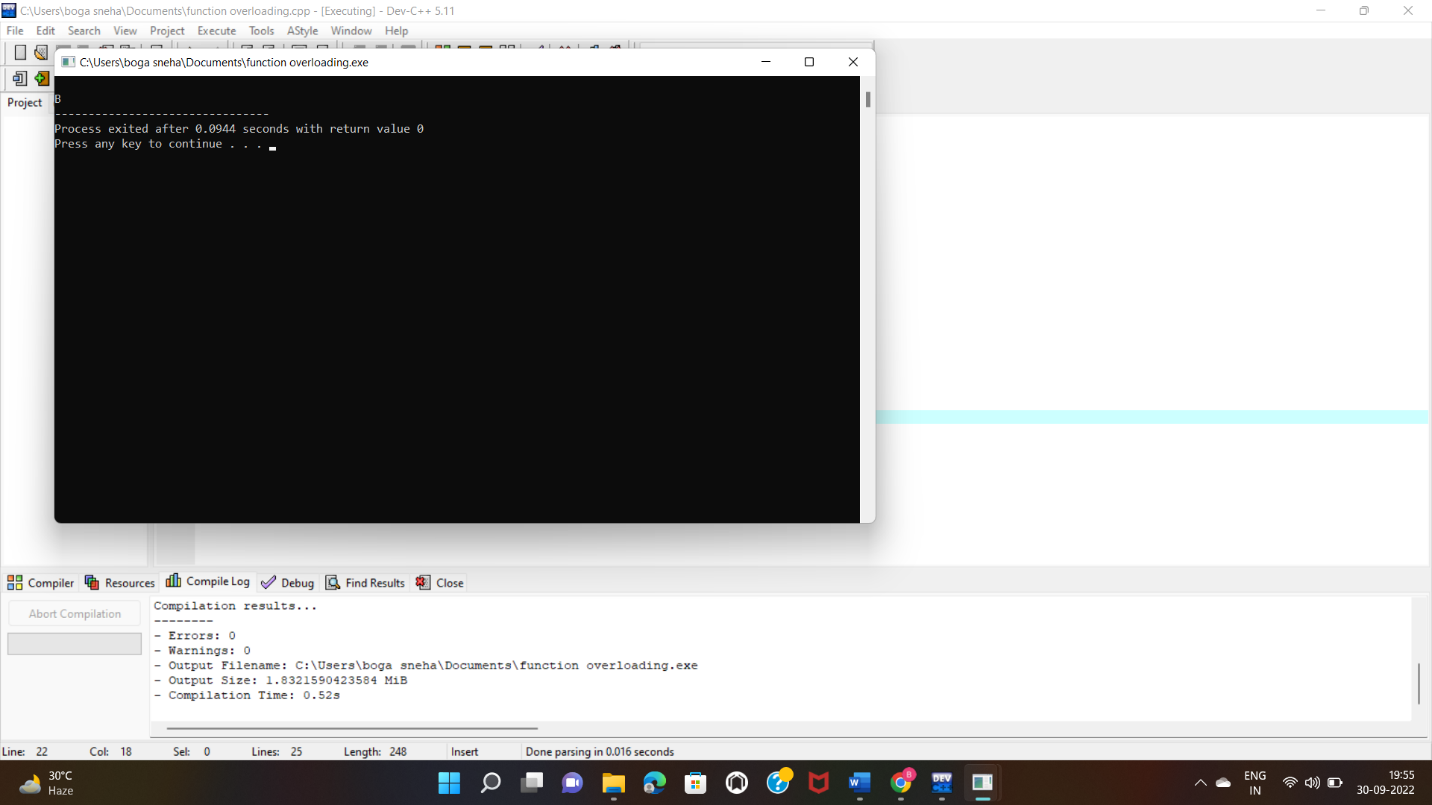
B b;

b.getdata() ;

return 0;

}

Output:



5)VIRTUAL FUNCTION

#include<iostream>

using namespace std;

class A

{

public:

virtual void getdata()

{

cout<<"\nA";

}

};

class B:public A

{

public:

void getdata()

{

cout<<"\nB";

}

};

int main()

{

A \*b;

B a;

b=&a;

b->getdata() ;

return 0;

}

OUTPUT:

