OBJECT ORIENTED PROGRAMMING WITH -C++

30/09/2022

1.program about polymorphism

PROGRAM:

#include<iostream>

using namespace std;

int l,b,h;

class area

{

public:

virtual void g()

{

cout<<"area is:"<<l\*b<<endl;

}

};

class volume:public area

{

public:

void g(){

cout<<"enter l,b,h:";

cin>>l>>b>>h;

cout<<"volume is:"<<l\*b\*h<<endl;

}

};

main()

{

area \*a;

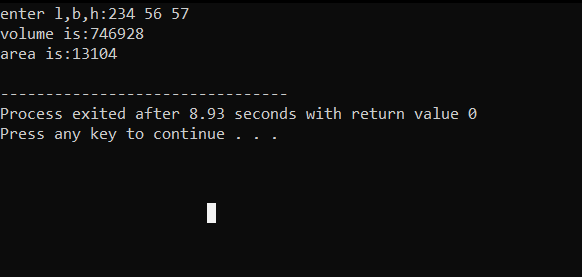
volume v;

a=&v;

a->g();

area u;

u.g();

OUTPUT: 

2.program about hierarchial inheritance bank

PROGRAM:

#include<iostream>

using namespace std;

char name[20];

int x,y,z;

class bank

{

public:

virtual void getrateofinterest()

{

cout<<"=== Bank Details ===\n";

cout<<"Enter your name = ";

cin>>name;

cout<<"Enter the account number of SBI = ";

cin>>x;

cout<<"Enter the account number of AXIS = ";

cin>>y;

cout<<"Enter the account number of ICICI = ";

cin>>z;

}

};

class sbi: public bank

{

public:

float h;

void getrateofinterest()

{

cout<<"\n\n=== SBI ===\n\n";

cout<<"Customer name = "<<name<<"\n";

cout<<"Account number = "<<x<<"\n";

cout<<"Balance = "<<"\n";

cin>>h;

cout<<"Rate of Interest = "<<h\*0.15;

}

};

class icici:public bank

{

public:

float i;

void getrateofinterest()

{

cout<<"\n\n=== ICICI ===\n\n";

cout<<"Customer name = "<<name<<"\n";

cout<<"Account number = "<<z<<"\n";

cout<<"Balance = "<<"\n";

cin>>i;

cout<<"Rate of Interest = "<<i\*0.10;

}

};

class axis:public bank

{

public:

float j;

void getrateofinterest()

{

cout<<"\n\n=== AXIS ===\n\n";

cout<<"Customer name = "<<name<<"\n";

cout<<"Account number = "<<y<<"\n";

cout<<"Balance = "<<"\n";

cin>>j;

cout<<"Rate of Interest = "<<j\*0.05;

}

};

int main()

{

bank b;

sbi s;

icici i;

axis a;

bank \*m;

m=&b;

m->getrateofinterest();

m=&s;

m->getrateofinterest();

m=&i;

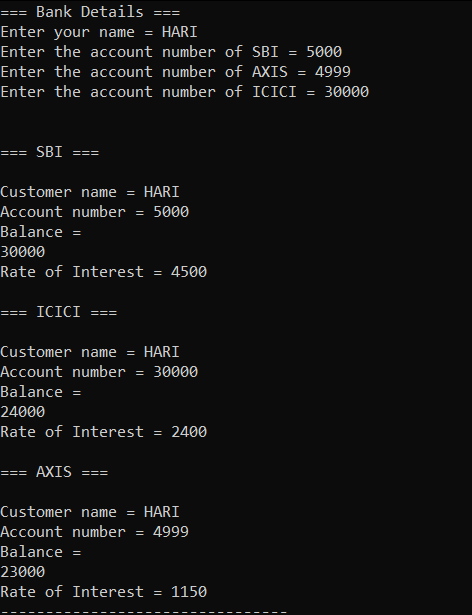
m->getrateofinterest();

m=&a;

m->getrateofinterest();

}

OUTPUT:



3.PROGRAM ABOUT EXCEPTION HANDLING

PROGAM:

#include<iostream>

using namespace std;

double division(int a,int b)

{

if(b == 0)

{

throw "division by zero";

}

return (a/b);

}

main()

{

int x = 50,y = 5;

double z = 0;

try

{

z = division(x,y);

cout<<z<<endl;

}

catch (const char\*msg)

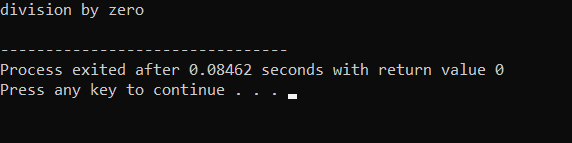
{

cerr<<msg<<endl;

}

}

OUTPUT:



4.PROGRAM ABOUT EXCEPTION FUNCTION

PROGRAM:

#include<iostream>

#include<exception>

using namespace std;

struct myexception:public exception

{

const char \*what() const throw()

{

return"c++ exception";

}

};

main()

{

try

{

throw myexception();

}

catch(myexception& e)

{

std::cout<<"myexception caught"<<std::endl;

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

}

catch(std::exception e)

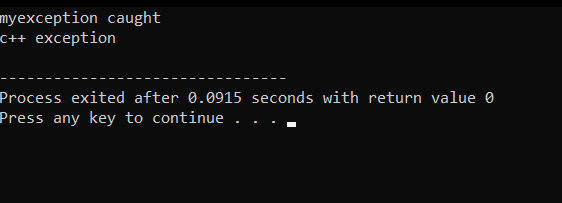
{

// other errors

}

}

OUTPUT:



5.program about the function overriding concept.

PROGRAM:

#include<iostream>

using namespace std;

class A

{

public:

virtual void getdata()

{

cout<<"A";

}

};

class B:public A

{

public:

void getdata()

{

cout<<"B";

}

};

main()

{

A \*b;

B a;

b=&a;

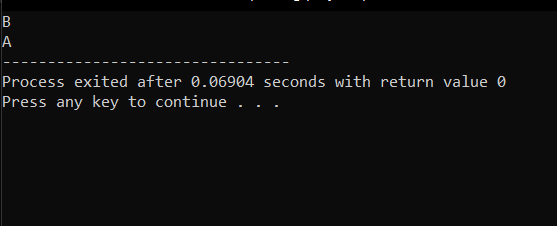
b->getdata();

A c;

c.getdata();

}

OUTPUT:



6.program to print BB in polymorphism

PROGRAM:

#include<iostream>

using namespace std;

class A

{

public:

virtual void getdata()

{

cout<<"A";

}

};

class B:public A

{

public:

void getdata()

{

cout<<"B"<<endl;

}

};

main()

{

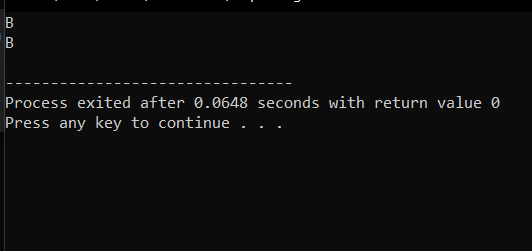
B b;

b.getdata();

b.getdata();

}

OUTPUT:



7.program about the