Q.1 Create class for Handling Exception for Student

Create StudentException class;

Create Student class with data members (rollno,name,age)

at the time of student creation check age if age<18 then throw exception

#include<iostream>

using namespace std;

class Student{

private:

int age,rollno;

string name;

public:

Student(int age,int rollno,string name){

this->age=age;

this->rollno=rollno;

this->name=name;

}

void display(){

cout<<rollno<<" "<<name<<" "<<age<<endl;

}

};

class StudentExp{

public:void Error(){

cout<<"Exception: Age is Less than 18"<<endl;

}

};

int main(){

int age;

cin>>age;

try{

if(age<18){

throw StudentExp();

}

Student s1(age,1,"Abc");

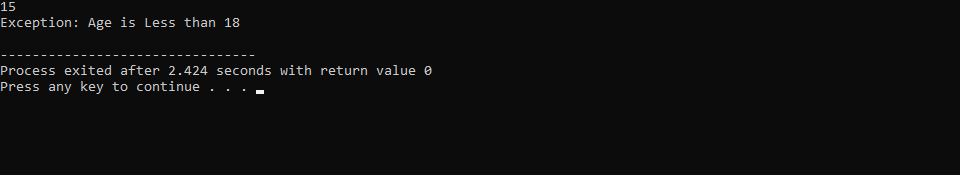
s1.display();

}catch(StudentExp e1){

e1.Error();

}

}



Q.2 Create Function Template and class template and use it

#include<iostream>

using namespace std;

template<class T> class Myclass{

private :

T val1;

T val2;

public :

Myclass(T val1,T val2){

this->val1=val1;

this->val2=val2;

}

void display(T &val1, T &val2){

T temp=val1;

val1=val2;

val2=temp;

}

};

int main(){

string a="Abhi",b="Hrushi";

Myclass<string> obj(a,b);

cout<<"Before Swapping"<<endl;

cout<<a<<" "<<b<<endl;

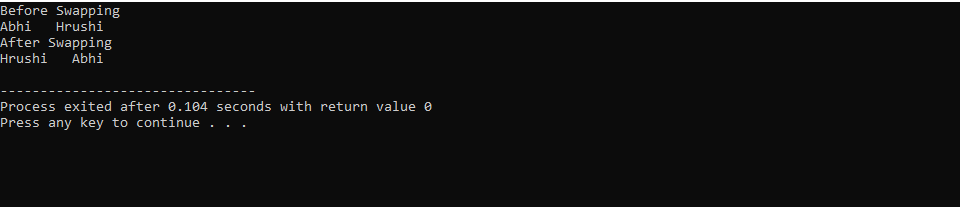
cout<<"After Swapping"<<endl;

obj.display(a,b);

cout<<a<<" "<<b<<endl;

}

**Output:**



Q.3 Create abstract class Fruit with abstract function taste

Create derived classes for Fruit:Apple,Mango,Orange

Implement abstract method in derived class ,add specific methods like juice(),jam(),pulp() in respective derived class(as discussed in lect);

create taster program.create fruit basket and invoke method

Use:dynamic\_cast,typeid

**Code:**

#include<iostream>

#include<typeinfo>

using namespace std;

class Fruit

{

public:

virtual void taste()=0;

};

class Mango:public Fruit

{

public:

void taste()

{

cout<<"Mango is sweet"<<endl;

}

void pulp()

{

cout<<"Pulp from mango"<<endl;

}

};

class Orange:public Fruit

{

public:

void taste()

{

cout<<"Orange is Sweet and sour"<<endl;

}

void juice()

{

cout<<"Juice from orange"<<endl;

}

};

class Apple:public Fruit

{

public:

void taste()

{

cout<<"Apple is Sweet"<<endl;

}

void jam()

{

cout<<"Jam from Apple"<<endl;

}

};

int main()

{

Fruit\*Basket[10];

int i=0,ch;

do

{

cout<<"1.Add mango 2.add orange 3.add apple 4.display 5. Exit"<<endl;

cin>>ch;

switch(ch)

{

case 1:{

Mango\* m1=new Mango();

Basket[i]=m1;

cout<<"Mango added at index "<<i<<endl;

i++;

break;

}

case 2:

{

Orange\*o1=new Orange();

Basket[i]=o1;

cout<<"Orange added at index "<<i<<endl;

i++;

break;

}

case 3:

{

Apple\*a1=new Apple();

Basket[i]=a1;

cout<<"Appple added at index "<<i<<endl;

i++;

break;

}

case 4:

{

cout<<"Fruites in the baskets are "<<endl;

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

{

Basket[j]->taste();

if(typeid(\*Basket[j])==typeid(Mango))

{

Mango \*m=dynamic\_cast<Mango \*>(Basket[j]);

m->pulp();

}

else if(typeid(\*Basket[j])==typeid(Orange))

{

Orange \*o=dynamic\_cast<Orange \*>(Basket[j]);

o->juice();

}

else if(typeid(\*Basket[j])==typeid(Apple))

{

dynamic\_cast<Apple \*>(Basket[j])->jam();

}

}

break;

}

default :

cout<<"Invalid Choice"<<endl;

break;

}

}while(ch!=5);

}

