

**1. Write a Program to design a class having static member function named**

***showcount()*** ***which*** ***has*** ***the*** ***property*** ***of***  **displaying** **the** **number** **of** **objects**

**created of the class.**

#include<iostream>

using namespace std;

class test

{

int code;

static int count;

public:

void setcode(void)

{

code = ++count;

}

void showcode(void)

{

cout<<"object number:"<<code<<"\n";

}

static void showcount(void)

{

cout<<"count:"<<count<<"\n";

}

};

int test :: count;

int main()

{

test t1,t2;

t1.setcode();

t2.setcode();

test :: showcount();

test t3;

t3.setcode();

test :: showcount();

t1.showcode();

t2.showcode();

t3.showcode();

return 0;

}

Output

count:2

count:3

object number:1

object number:2

object number:3

**2. Write a Program using class to process Shopping List for a Departmental**

**Store. The list include details such as the Code No and Price of each item**

**and** **perform** **the** **operations** **like** **Adding,** **Deleting** **Items** **to** **the** **list** **and**

**Printing the Total value of a Order.**

#include<iostream>

#define cout print; //define count in ‘print’

using namespace std;

class ITEMS

{

int iCode[500];

float iPrice[500];

int count;

public:

void ICOUNT(void){count=0;}

void getItem(void);

void printSum(void);

void remove(void);

void displayItems(void);

};

void ITEMS :: getItem(void)

{

print<<"Enter item code";

cin>>iCode[count];

print<<"Enter Item cost";

cin>>iPrice[count];

count++;

}

void ITEMS :: printSum(void)

{

float sum=0;

for(int i=0;i<count;i++)

sum=sum+iPrice[i];

print<<"\n Total Value:"<<sum<<"\n";

}

void ITEMS :: remove(void)

{

int a;

print<<"Enter Item Code";

cin>>a;

for(int i=0;i<count;i++)

if(iCode[i] == a)

iPrice[i]=0;

}

void ITEMS :: displayItems(void)

{

print<<"\n Code Price\n";

for(int i=0;i<count;i++)

{

print<<"\n"<<iCode[i];

print<<" "<<iPrice[i];

}

print<<"\n";

}

int main()

{

ITEMS order;

order.ICOUNT();

int x;

do

{

print<<"\n You can do the following;"

<<"Enter appropriate number\n";

print<<"\n1 : Add an Item";

print<<"\n2 : Display Total Value";

print<<"\n3 : Delete an Item";

print<<"\n4 : Display all items";

print<<"\n5 : Quit";

print<<"\n\n What is your option?";

cin>>x;

switch(x)

{

case 1 : order.getItem();

break;

case 2 : order.printSum();

break;

case 3 : order.remove();

break;

case 4 : order.displayItems();

break;

default : cout<<"Error in input";

}

}while(x!=5);

return 0;

}

Output:

You can do the following; Enter appropriate number

1: Add an Item

2: Display Total Value

3: Delete an Item

4: Display all items

5: Quit

What is your option?

**3. Write a Program which creates & uses*array of object of a class.( for eg.***

**implementing the list of Managers of a Company having details such as Name,**

**Age, etc..).**

#include<iostream>

using namespace std;

class employee

{

char name[30];

float age;

public:

void getdata(void);

void putdata(void);

};

void employee :: getdata(void)

{

cout<<"Enter Name :";

cin>>name;

cout<<"Enter Age :";

cin>>age;

}

void employee :: putdata(void)

{

cout<<"Name :"<<name<<"\n";

cout<<"Age : "<<age<<"\n";

}

const int size=3;

int main()

{

employee manager[size];

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

{

cout<<"\nDetails of manager :-"<<i+1<<"\n";

manager[i].getdata();

}

cout<<"\n";

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

{

cout<<"\n Manager"<<j+1<<"\n";

manager[j].putdata();

}

return 0;

}

OUTPUT:

Details of manager :-1

Enter Name :suvadip

Enter Age :12

Details of manager :-2

Enter Name :rabi

Enter Age :17

Details of manager :-3

Enter Name :sourav

Enter Age :27

Manager1

Name :suvadip

Age : 12

Manager2

Name :rabi

Age : 17

Manager3

Name :sourav

Age : 27

**4. Write a Program to find Maximum out of Two Numbers using*friend function.***

***Note: Here one number is a member of one class and the other number is member***

***of some other class.***

#include<iostream>

using namespace std;

class ABC;

class XYZ

{

int x;

public:

void setvalue(int i)

{

x=i;

}

friend void max(XYZ, ABC);

};

class ABC

{

int a;

public:

void setvalue(int i)

{

a=i;

}

friend void max(XYZ, ABC);

};

void max (XYZ m, ABC n)

{

if(m.x>=n.a)

cout<<”maxumum :”<<m.x;

else

cout<<”maximum :”<<n.a;

}

int main()

{

ABC abc;

abc.setvalue(10);

XYZ xyz;

xyz.setvalue(20);

max(xyz,abc);

return 0;

}

OUTPUT

maximum :20

**5. Write a Program to swap private data members of classes named as*class\_1,***

***class\_2* using friend function.**

#include<iostream>

using namespace std;

class class\_2;

class class\_1

{

int value1;

public:

void indata(int a)

{

value1=a;

}

void display(void)

{

cout<<value1<<"\n";

}

friend void exchange(class\_1 &, class\_2 &);

};

class class\_2

{

int value2;

public:

void indata(int a)

{

value2=a;

}

void display(void)

{

cout<<value2<<"\n";

}

friend void exchange(class\_1 &, class\_2 &);

};

void exchange(class\_1 &x, class\_2 &y)

{

int temp = x.value1;x.value1 = y.value2;y.value2 = temp;

}

int main()

{

class\_1 C1;class\_2 C2;

C1.indata(100);

C2.indata(200);

cout<<"Values before exchange"<<"\n";

C1.display();

C2.display();

exchange(C1, C2);

cout<<"Values after exchange"<<"\n";

C1.display();

C2.display();

return 0;

}

Output:

Values before exchange

100

200

Values after exchange

200

100

**6. Write a Program to design a class complex to represent complex numbers.**

**The complex class shuold use an external function (use it as a friend**

**function) to add two complex numbers.The function should return an object of**

**type complex representing the sum of two complex numbers.**

#include<iostream>

using namespace std;

class complex

{

float x;

float y;

public:

void input(float real, float img)

{

x=real;

y=img;

}

friend complex sum(complex, complex);

void show(complex);

};

complex sum(complex c1, complex c2)

{

complex c3;

c3.x = c1.x + c2.x;

c3.y = c1.y + c2.y;

return (c3);

}

void complex :: show(complex c)

{

cout<<c.x<<"+j"<<c.y<<"\n";

}

int main()

{

complex A,B,C;

A.input(3.1, 5.65);

B.input(2.75, 1.2);

C=sum(A,B);

cout<<"A=";

A.show(A);

cout<<"B=";

B.show(B);

cout<<"C=";

C.show(C);

return 0;

}

Output:

A=3.1+j5.65

B=2.75+j1.2

C=5.85+j6.85

**7. Write a Program using*copy constructor* to copy data of an object to**

**another object.**

#include<iostream>

using namespace std;

class code

{

int id;

public:

code(){}

code(int a)

{

id = a;

}

code(code & x)

{

id = x.id;

}

void display(void)

{

cout<<id;

}

};

int main()

{

code A(100);

code B(A);

code C = A;

code D;

D = A;

cout<<"\n id of A:";

A.display();

cout<<"\n id of B:";

B.display();

cout<<"\n id of C:";

C.display();

cout<<"\n id of D:";

D.display();

return 0;

}

Output:

id of A:100

id of B:100

id of C:100

id of D:100

**8. Write a Program to allocate memory dynamically for an objects of a given**

**class using class’s constructor.**

#include<iostream>

#include<string.h>

using namespace std;

class String

{

char \*name;

int length;

public:

String()

{

length = 0;

name = new char[length +1];

}

String (char \*s)

{

length = strlen(s);

name= new char[length + 1];

strcpy(name, s);

}

void display(void)

{

cout<<name<<"\n";

}

void join(String &a, String &b);

};

void String :: join (String &a, String &b)

{

length = a.length + b.length;

delete name;

name = new char [length + 1];

strcpy(name,a.name);

strcat(name, b.name);

};

int main()

{

char \*first = "Rabi ";

String name1(first), name2("Biswarup "), name3("Aishi "),s1,s2;

s1.join(name1, name2);

s2.join(s1, name3);

name1.display();

name2.display();

name3.display();

s1.display();

s2.display();

return 0;

}

Output:

Rabi

Biswarup

Aishi

RabiBiswarup

RabiBiswarup Aishi

**9. Write a Program to design a class to represent a matrix. The class should**

**have the functionality to insert and retrieve the elements of the matrix.**

#include<iostream>

using namespace std;

class matrix

{

int \*\*p;

int d1,d2;

public:

matrix(int x, int y);

void get\_element(int i, int j, int value)

{

p[i][j]=value;

}

int & put\_element(int i, int j)

{

return p[i][j];

}

};

matrix ::matrix(int x, int y)

{

d1 = x;

d2 = y;

p = new int \*[d1];

for(int i = 0; i < d1; i++)

p[i] = new int[d2];

}

int main()

{

int m, n;

cout<<"Enter size of matrix";

cin>>m>>n;

matrix A(m,n);

cout<<"Enter Matrix Element row by row:";

int i,j,value;

for(i=0;i<m;i++)

for(j=0;j<n;j++)

{

cin>>value;

A.get\_element(i,j,value);

}

cout<<"\n";

cout<<A.put\_element(1,2);

return 0;

}

Output:

Enter size of matrix 3 2

Enter Matrix Element row by row:1

2

2

3

3

5

13500752

**10. Write a program to design a class representing complex numbers and having**

**the functionality of performing addition & multiplication of two complex**

**numbers using operator overloading.**

#include<iostream>

using namespace std;

class complex

{

private:

float real,imag;

public:

complex( )

{

}

complex( float r, float i)

{

real = r;

imag = i;

}

void getdata( )

{

float r,

i;

cout << endl << "Enter real and imaginary part ";

cin >> r >> i;

real = r;

imag = i;

}

void setdata( float r, float i)

{

real = r;

imag = i;

}

void displaydata( )

{

cout << endl << "real = " << real;

cout<<endl<<"Imaginary = "<<imag;

}

complex operator +( complex c )

{

complex t;

t.real = real + c.real;

t.imag = imag + c.imag;

}

complex operator \*( complex c )

{

complex t;

t.real = real \* c.real - imag \* c.imag;

t.imag = real \* c.imag + c.real \* imag;

return t;

}

};

int main()

{

complex c1,c2( 1.2, -2.5 ),c3,c4;

c1.setdata( 2.0, 2.0 );

c3 = c1 + c2;

c3.displaydata( );

c4.getdata( );

complex c5 ( 2.5, 3.0 ),c6;

c6 = c4 \* c5;

c6.displaydata( );

complex c7;

c7 = c1 + c2 \* c3;

c7.displaydata( );

return 0;

}

OUTPUT

real = 9.64233e-039

Imaginary = -2.5

Enter real and imaginary part 56 8

real = 116

Imaginary = 188

real = 9.64233e-039

Imaginary = -3

**11. Write a Program to overload operators like \*, <<, >> using friend**

**function. The following overloaded operators should work for a class*v****ector*.

#include<iostream>

using namespace std;

const int size = 3;

class vector

{

int v[size];

public:

vector();

vector(int \*x);

friend vector operator \*(int a, vector b);

friend vector operator \*(vector b, int a);

friend istream & operator >>(istream &, vector &);

friend ostream & operator <<(ostream &, vector &);

};

vector ::vector()

{

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

v[i]=0;

}

vector :: vector(int \*x)

{

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

v[i] = x[i];

}

vector operator \*(int a, vector b)

{

vector c;

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

c.v[i] = a \* b.v[i];

return c;

}

vector operator \*(vector b, int a)

{

vector c;

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

c.v[i] = b.v[i] \* a;

return c;

}

istream & operator >> (istream &din, vector &b)

{

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

din>>b.v[i];

return(din);

}

ostream & operator << (ostream &dout, vector &b)

{

dout<<"("<<b.v [0];

for(int i=1; i<size; i++)

dout<<","<<b.v[i];

dout<<")";

return(dout);

}

int x[size] = {2,4,6};

int main()

{

vector m;

vector n = x;

cout<<"Enter Elements of vector m";

cin>>m;

cout<<"\n";

cout<<"m="<<m<<"\n";

vector p,q;

p = 2 \* m;

q = n \* 2;

cout<<"\n";

cout<<"p="<<p<<"\n";

cout<<"q="<<q<<"\n";

return 0;

}

OUTPUT:

Enter Elements of vector m

5 6 8

m=(5,6,8)

p=(10,12,16)

q=(4,8,12)

**12. Write a program for developing a matrix class which can handle integer**

**matrices of different dimensions. Also overload the operator for addition,**

**multiplication & comparison of matrices.**

#include<iostream>

#include<iomanip>

using namespace std;

class matrix

{

int maxrow, maxcol;

int \* ptr;

public:

matrix( int r, int c )

{

maxrow = r;

maxcol = c;

ptr = new int [r \* c];

}

void getmat( )

{

int i,j, mat\_off,temp;

cout << endl << "enter elements matrix:" << endl;

for( i = 0; i < maxrow; i++ )

{

for( j = 0; j < maxcol; j++ )

{

mat\_off = i \* maxcol + j;

cin >> ptr[ mat\_off ];

}

}

}

void printmat( )

{

int i, j, mat\_off;

for( i = 0; i < maxrow; i++ )

{

cout << endl;

for( j = 0; j < maxcol; j++ )

{

mat\_off = i \* maxcol + j;

cout << setw( 3 ) << ptr[ mat\_off ];

}

}

}

int delmat( )

{

matrix q ( maxrow - 1, maxcol - 1 );

int sign = 1, sum = 0, i, j,k,count;

int newsize,newpos,pos,order;

order = maxrow;

if( order == 1 )

{

return ( ptr[ 0 ] );

}

for( i = 0; i < order; i++, sign \*= -1 )

{

for( j = 1; j < order; j++ )

{

for( k = 0, count = 0; k < order;

k++ )

{

if( k == i )

continue;

pos = j \* order + k;

newpos = ( j - 1 ) \* ( order - 1 ) + count;

q.ptr[ newpos ] = ptr[ pos ];

count++;

}

}

sum = sum + ptr[ i ] \* sign \* q.delmat( );

}

return ( sum );

}

matrix operator +( matrix b )

{

matrix c ( maxrow, maxcol );

int i,j,mat\_off;

for( i = 0; i < maxrow; i++ )

{

for( j = 0; j < maxcol; j++ )

{

mat\_off = i \* maxcol + j;

c.ptr[ mat\_off ] = ptr[ mat\_off ] + b.ptr[ mat\_off ];

}

}

return ( c );

}

matrix operator \*( matrix b )

{

matrix c ( b.maxcol, maxrow );

int i,j,k,mat\_off1, mat\_off2, mat\_off3;

for( i = 0; i < c.maxrow; i++ )

{

for( j = 0; j < c.maxcol; j++ )

{

mat\_off3 - i \* c.maxcol + j;

c.ptr[ mat\_off3 ] = 0;

for( k = 0; k < b.maxrow; k++ )

{

mat\_off2 = k \* b.maxcol + j;

mat\_off1 = i \* maxcol + k;

c.ptr[mat\_off3]+=ptr[mat\_off1]\* b.ptr[mat\_off2 ];

}

}

}

return ( c );

}

int operator ==( matrix b )

{

int i,j, mat\_off;

if( maxrow != b.maxrow

|| maxcol != b.maxcol )

return ( 0 );

for( i = 0; i < maxrow; i++ )

{

for( j = 0; j < maxcol; j++ )

{

mat\_off = i \* maxcol + j;

if( ptr[ mat\_off ]

!= b.ptr[ mat\_off ] )

return ( 0 );

}

}

return ( 1 );

}

} ;

 main( )

{

int rowa, cola, rowb, colb;

cout << endl << "Enter dimensions of matrix A ";

cin >> rowa >> cola;

matrix a ( rowa, cola );

a.getmat( );

cout << endl << "Enter dimensions of matrix B";

cin >> rowb >> colb;

matrix b ( rowb, colb );

b.getmat( );

matrix c ( rowa, cola );

c = a + b;

cout << endl << "The sum of two matrics = ";

c.printmat( );

matrix d ( rowa, colb );

d = a \* b;

cout << endl << "The product of two matrics = ";

d.printmat( );

cout << endl << "Determinant of matrix a =" << a.delmat( );

if( a == b )

cout << endl << "a & b are equal";

else

cout << endl << "a & b are not equal";

return 0;

}

Output:

Enter dimensions of matrix A 2 2

enter elements matrix:

2 4

3 2

Enter dimensions of matrix B2 2

enter elements matrix:

3 4

7 8

The sum of two matrix’s =

5 8

10 10

**13. Write a program to overload*new/delete* operators in a class.**

#include<iostream.h>

#include<stdlib.h>

#include<string.h>

#include<new.h>

using namespace std;

const int MAX = 5;

const int FREE = 0;

const int OCCUPIED = 1;

void memwarning( )

{

cout << endl << "Free store has now gone empty";

exit( 1 );

}

class employee

{

private:

char name[ 20 ];

int age;

float sal;

public:

void \*operator new(size\_t bytes)

void operator delete( void \* q );

void setdata( char \* n, int a, float s );

void showdata( );

~employee( );

} ;

struct pool

{

employee obj;

int status;

} ;

int flag = 0;

struct pool \* p = NULL;

void \* employee::operator new( size\_t sz )

{

int i;

if( flag == 0 )

{

p = ( pool \* )malloc( sz \* MAX );

if( p == NULL )

memwarning( );

for( i = 0; i < MAX; i++ )

p[ i ].status = FREE;

flag = 1;

p[ 0 ].status = OCCUPIED;

return &p[ 0 ].obj;

}

else

{

for( i = 0; i < MAX; i++ )

{

if( p[ i ].status = FREE )

{

p[ i ].status = OCCUPIED;

return &p[ i ].obj;

}

}

memwarning( );

}

}

void employee::operator delete( void \* q )

{

if( q == NULL )

return;

for( int i = 0; i < MAX; i++)

{

if( q == &p[ i ].obj )

{

p[ i ].status = FREE;

strcpy( p[ i ].obj.name, "" );

p[ i ].obj.age = 0;

p[ i ].obj.sal = 0.0;

}

}

}

void employee::setdata( char \* n, int a, float s )

{

strcpy( name, n );

age = a;

sal = s;

}

void employee::showdata( )

{

cout << endl << name << "\t" << age << "\t" << sal;

}

employee::~employee( )

{

cout << endl << "reached destructor";

free( p );

}

int main( )

{

void memwarning( );

set\_new\_handler( memwarning );

employee \* e1,\*e2,\*e3,\*e4,\*e5,\*e6;

e1 = new employee;

e1->setdata( "ajay", 23, 4500.50 );

e2 = new employee;

e2->setdata( "amol", 25, 5500.50 );

e3 = new employee;

e3->setdata( "anil", 26, 3500.50 );

e4 = new employee;

e4->setdata( "anuj", 30, 6500.50 );

e5 = new employee;

e5->setdata( "atul", 23, 4200.50 );

e1->showdata( );

e2->showdata( );

e3->showdata( );

e4->showdata( );

e5->showdata( );

delete e4;

delete e5;

e4->showdata( );

e5->showdata( );

e4 = new employee;

e5 = new employee;

e6 = new employee;

cout << endl << "Done!!";

return 0;

}

Output:

Enter dimensions of matrix A 2 1

enter elements matrix:

1 3

Enter dimensions of matrix B1 1

enter elements matrix:

2 1

The sum of two matrix’s =

3

3

**14. Write a program in C++ to highlight the difference between overloaded**

***assignment operator* and*copy constructor*.**

#include<iostream.h>

using namespace std;

class circle

{

private:

int radius;

float x, y;

public:

circle( )

{

}

circle( int rr, float xx, float yy )

{

radius = rr;

x = xx;

y = yy;

}

circle operator =( circle & c )

{

cout << endl << "Assignment operator invoked";

radiius = c.radius;

x = c.x;

y = c.y;

return circle( radius, x, y );

}

circle( circle & c )

{

cout << endl << "copy constructor invoked";

radius = c.radius;

x = c.x;

y = c.y;

}

void showdata( )

{

cout << endl << "Radius = " << radius;

cout << endl << "X-Coordinate=" << x;

cout << endl << "Y-Coordinate=" << y;

}

} ;

void main( )

{

circle c1 ( 10, 2.5, 2.5 );

circle c2,c4;

c4 = c2 = c1;

circle c3 = c1;

c1.showdata( );

c2.showdata( );

c3.showdata( );

c4.showdata( );

}

**15. Write a Program illustrating how the constructors are implemented and the**

**order in which they are called when the classes are inherited. Use three**

**classes named*alpha, beta, gamma such that alpha,beta are base class and***

***gamma is derived class inheriting alpha & beta***

#include<iostream>

using namespace std;

class alpha

{

int x;

public:

alpha(int i)

{

x = i;

cout<<"alpha initialized\n";

}

void show\_x(void)

{

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

}

};

class beta

{

float y;

public:

beta(float j)

{

y=j;

cout<<"beta initialized\n";

}

void show\_y(void)

{

cout<<"y= "<<y<<"\n";

}

};

class gamma : public beta, public alpha

{

int m,n;

public:

gamma(int a, float b, int c, int d):

alpha(a), beta(b)

{

m = c; n = d;

cout<<"gamma initialized\n";

}

void show\_mn(void){

cout<<"m="<<m<<"\n";

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

}

};

int main()

{

gamma g(5, 10.75, 20, 30);

g.show\_x();

g.show\_y();

g.show\_mn();

return 0;

}

Output:

beta initialized

alpha initialized

gamma initialized

x=5

y= 10.75

m=20

n=30

**16. Write a Program to design a stuent class representing student roll no.**

**and a test class (derived class of student) representing the scores of the**

**student** **in** **various** **subjects** **and** **sports** **class** **representing** **the** **score** **in**

**sports.** **The** **sports** **and** **test** **class** **should** **be** **inherited** **by** **a** **result** **class**

**having the functionality to add the scores and display the final result for a**

**student.**

#include<iostream>

using namespace std;

class student

{

protected:

int roll\_number;

public:

void get\_number(int a)

{

roll\_number = a;

}

void put\_number(void)

{

cout<<"Roll No:"<<roll\_number<<"\n";

}

};

class test : public student

{

protected:

float part1, part2;

public:

void get\_marks(float x, float y)

{

part1 = x;

part2 = y;

}

void put\_marks(void)

{

cout<<"Marks obtained"<<"\n"

<<"part1 ="<<part1<<"\n"

<<"part2 ="<<part2<<"\n";

}

};

class sports

{

protected:

float score;

public:

void get\_score(float s)

{

score = s;

}

void put\_score(void)

{

cout<<"Sports wt:"<<score<<"\n\n";

}

};

class result : public test, public sports

{

float total;

public:

void display(void);

};

void result ::display(void)

{

total = part1 + part2 + score;

put\_number();

put\_marks();

put\_score();

cout<<"Total Score:"<<total<<"\n";

}

int main()

{

result student\_1;

student\_1.get\_number (9);

student\_1.get\_marks (6.5, 6.9);

student\_1.get\_score (6.0);

student\_1.display ();

return 0;

}

Output:

Roll No:9

Marks obtained

part1 =6.5

part2 =6.9

Sports wt:6

Total Score:19.4

**17. Write a program to maintain the records of person with details*(Name and***

***Age)* and find the eldest among them. The program must use*this pointer* to**

**return the result.**

#include<iostream>

#include<string>

using namespace std;

class person

{

char name[20];

float age;

public:

person(char \*s, float a)

{

strcpy(name, s);

age = a;

}

person & person :: greater(person & x)

{

if(x.age >= age)

return x;

else

return \*this;

}

void display(void)

{

cout<<"Name:"<<name<<"\n"

<<"Age: "<<age<<"\n";

}

};

int main()

{

person p1("John", 37.50),

p2("Ahmed",29.0),

p3("Hebber", 40.5);

person p = p1.greater (p3);

cout<<"Elder Person is:\n";

p.display();

p = p1.greater (p2);

cout<<"Elder Person is:\n";

p.display();

return 0;

}

**18. Write a Program to illustrate the use of pointers to objects whch are**

**related by inheritance.**

#include<iostream>

using namespace std;

class BC

{

public:

int b;

void show()

{

cout<<"b="<<b<<"\n";

}

};

class DC : public BC

{

public:

int d;

void show()

{

cout<<"b="<<b<<"\n"

<<"d="<<d<<"\n";

}

};

int main()

{

BC \*bptr;

BC base;

bptr = &base;

bptr->b = 100;

cout<<"bptr points to base object\n";

bptr->show ();

DC derived;

bptr = &derived;

bptr->b = 200;

cout<<"bptr now points to derived object\n";

bptr->show ();

DC \*dptr;

dptr = &derived;

dptr->d = 300;

cout<<"dptr is derived type pointer\n";

dptr->show ();

cout<<"Using ((DC \*)bptr)\n";

((DC \*)bptr)->d = 400;

((DC \*)bptr)->show ();

return 0;

}

Output:

bptr points to base object

b=100

bptr now points to derived object

b=200

dptr is derived type pointer

b=200

d=300

Using ((DC \*)bptr)

b=200

d=400

**19. Write a program illustrating the use of virtual functions in class.**

#include<iostream>

using namespace std;

class Base

{

public:

void display()

{

cout<<"\n Display Base";

}

virtual void show()

{

cout<<"\n Show Base:";

}

};

class Derived : public Base

{

public:

void display()

{

cout<<"\n Display Derived";

}

void show()

{

cout<<"\n Show Derived";

}

};

int main()

{

Base B;

Derived D;

Base \*bptr;

cout<<"\n bptr points to Base\n";

bptr = &B;

bptr ->display ();

bptr ->show ();

cout<<"\n\n bptr points to derived\n";

bptr = &D;

bptr ->display ();

bptr ->show ();

return 0;

}

Output:

bptr points to Base

Display Base

Show Base:

bptr points to derived

Display Base

Show Derived

**20. Write a program to design a class representing the information regarding**

**digital library (books, tape: book & tape should be separate classes having**

**the base class as media ). The class should have the** **functionality for**

**adding new item, issuing, deposit etc. the program should use the runtime**

**polymorphism.**

#include<iostream>

#include<string.h>

using namespace std;

class media

{

protected:

char title[50];

float price;

public:

media(char \*s, float a)

{

strcpy(title, s);

price = a;

}

virtual void display(){}

};

class book : public media

{

int pages;

public:

book(char \*s, float a, int p) : media(s,a)

{

pages = p;

}

void display();

};

class tape : public media

{

float time;

public:

tape(char \* s, float a, float t):media(s,a)

{

time =t;

}

void display();

};

void book ::display()

{

cout<<"\n Title:"<<title;

cout<<"\n Pages:"<<pages;

cout<<"\n Price:"<<price;

}

void tape ::display ()

{

cout<<"\n Title:"<<title;

cout<<"\n Play Time:"<<time<<"mins";

cout<<"\n Price:"<<price;

}

int main()

{

char \* title = new char[30];

float price, time;

int pages;

cout<<"\n Enter Book Details \n";

cout<<"\n Title:";

cin>>title;

cout<<"\n Price:";

cin>>price;

cout<<"\n Pages:";

cin>>pages;

book book1(title, price, pages);

cout<<"\n Enter Tape Details";

cout<<"\n Title:";

cin>>title;

cout<<"\n Price:";

cin>>price;

cout<<"\n Play Times(mins):";

cin>>time;

tape tape1(title, price, time);

media\* list[2];

list[0] = &book1;

list[1] = &tape1;

cout<<"\n Media Details";

cout<<"\n............. Book.....";

list[0]->display ();

cout<<"\n............. Tape.....";

list[1]->display ();

return 0;

}

Output:

bptr points to Base

Display Base

Show Base:

bptr points to derived

Display Base

Show Derived

**21. write a program to show conversion from string to int and vice-versa.**

#include<iostream>

#include<stdlib.h>

#include<string.h>

using namespace std;

class strings

{

private:

char str[ 20 ];

public:

strings( )

{

str[ 0 ] = '\0';

}

strings( char \* s )

{

strcpy( str, s );

}

strings( int a )

{

itoa( a, str, 10 );

}

operator int( )

{

int i= 0,l,ss = 0,k = 1;

l = strlen( str ) - 1;

while( l >= 0 )

{

ss = ss + ( str[ l ] - 48 ) \* k;

l--;

k \*= 10;

}

return ( ss );

}

void displaydata( )

{

cout << str;

}

} ;

int main( )

{

strings s1 = 123;

cout << endl << "s1=";

s1.displaydata( );

s1 = 150;

cout << endl << "s1=";

s1.displaydata( );

strings s2 ( "123" );

int i = int( s2 );

cout << endl << "i=" << i;

strings s3 ( "456" );

i = s3;

cout << endl << "i=" << i;

return 0;

}

OUTPUT

s1=123

s1=150

i=123

i=456

**22. Write a program showing data conversion between objects of different**

**classes.**

#include<iostream.h>

#include<stdlib.h>

#include<string.h>

using namespace std;

class date

{

private:

char dt[ 9 ];

public:

date( )

{

dt[ 0 ] = '\0';

}

date( char \* s )

{

strcpy( dt, s );

}

void displaydata( )

{

cout << dt;

}

} ;

class dmy

{

private:

int day,

mth,

yr;

public:

dmy( )

{

day = mth = yr = 0;

}

dmy( int d, int m, int y )

{

day = d;

mth = m;

yr = y;

} ;

operator date( )

{

char temp[ 3 ], str[ 9 ];

itoa( day, str, 10 );

strcat( str, "/" );

itoa( mth, temp, 10 );

strcat( str, temp );

strcat( str, "/" );

itoa( yr, temp, 10 );

strcat( str, temp );

return ( date( str ) );

}

void displaydata( )

{

cout << day << "\t" << mth << "\t" << yr;

}

} ;

void main( )

{

date d1;

dmy d2 ( 17, 11, 94 );

d1 = d2;

cout<,endl<<"d1=";

d1.displaydata( );

cout << endl << "d2=";

d2.displaydata( );

}

Output:

d1=17/11/94

d2=17 11 94

**23.** **Write** **a** **program** **showing** **data** **conversion** **between** **objects** **of** **different**

**classes and conversion routine should reside in destination class.**

#include<iostream>

#include<string.h>

#include<stdlib.h>

using namespace std;

class dmy

{

int day,

mth,

yr;

public:

dmy( )

{

day = mth, yr = 0;

}

dmy( int d, int m, int y )

{

day = d;

mth = m;

yr = y;

}

int getday( )

{

return ( day );

}

int getmth( )

{

return ( mth );

}

int getyr( )

{

return ( yr );

}

void displaydata( )

{

cout << day << "\t" << mth << "\t" << yr;

}

} ;

class date

{

private:

char dt[ 9 ];

public:

date( )

{

dt[ 0 ] = '\0';

}

date( char \* s )

{

strcpy( dt, s );

}

void displaydata( )

{

cout << dt;

}

date( dmy t )

{

int d = t.getday( );

int m = t.getmth( );

int y = t.getyr( );

char temp[ 3 ];

itoa( d, dt, 10 );

strcat( dt, "\t" );

itoa( m, temp, 10 );

strcat( dt, temp );

strcat( dt, "/" );

itoa( y, temp, 10 );

strcat( dt, temp );

}

} ;

int main( )

{

date d1;

dmy d2 ( 17, 11, 94 );

d1 = d2;

cout << endl << "d1=";

d1.displaydata( );

cout << endl << "d2=";

d2.displaydata( );

return 0;

}

OUTPUT

d1=17 11/94

d2=17 11 94

**24. Write a program to implement I/O operations on characters. I/O operations**

**includes inputing a string, Calculating length of the string, Storing the**

**String in a file, fetching the stored characters from it, etc.**

#include<iostream>

#include<fstream.h>

#include<string.h>

using namespace std;

int main()

{

char string[80];

cout<<"Enter a String \n";

cin>>string;

int len = strlen(string);

fstream file;

file.open("TEXT", ios::in | ios::out);

for(int i=0;i<len;i++)

file.put(string[i]);

file.seekg(0);

char ch;

while(file)

{

file.get(ch);

cout<<ch;

}

return 0;

}

OUTPUT

Suvadip

Suvadip

**25. Write a program to copy the contents of one file to another.**

#include<iostream>

#include<fstream>

using namespace std;

int main( )

{

char source[ 67 ],target[ 67 ];

char ch;

cout << endl << "Enter source filename :";

cin >> source;

cout << endl << "Enter target filename :";

cin >> target;

ifstream infile ( source );

ofstream outfile ( target );

while( infile )

{

infile.get( ch );

outfile.put( ch );

}

return 0;

}

OUTPUT

Enter source filename :TEXT

Enter target filename :suva

**26. Write a program to perform read/write binary I/O operation on a file**

**(i.e. write the object of a structure/class to file).**

#include<fstream>

#include<iostream>

using namespace std;

void main( )

{

struct employee

{

char name[ 20 ];

int age;

float basic;

float gross;

} ;

employee e;

char ch = 'Y';

ofstream outfile;

outfile.open( "EMPLOYEE.DAT", ios::out | ios::binary );

while( ch == 'Y' )

{

cout << endl << "Enter a record";

cin >> e.name >> e.age >> e.basic >> e.gross;

outfile.write( ( char \* )&e, sizeof( e ) );

cout << endl << "Add Another Y/N";

cin >> ch;

}

outfile.close( );

ifstream infile;

infile.open( "EMPLOYEE.DAT", ios::in | ios::binary );

while( infile.read( ( char \* )&e, sizeof( e ) ) )

{

cout << endl << e.name << "\t" << e.age << "\t" << e.basic << "\t"

<< e.gross;

}

}

Output:

Enter a record3

7

4

1

Add Another Y/Nn

3 7 4 1

**27. Write a program to maintain a elementary database of employees using**

**files.**

#include<fstream>

#include<conio.h>

#include<stdlib.h>

#include<stdio.h>

#include<string.h>

#include<iomanip>

#include<iostream>

using namespace std;

class group

{

private:

struct person

{

char flag;

char empcode[ 5 ];

char name[ 40 ];

int age;

float sal;

} p;

fstream file;

public:

group( );

void addrec( );

void listrec( );

void modirec( );

void delrec( );

void recallrec( );

void packrec( );

void exit( );

} ;

int main( )

{

char choice;

group g;

do

{

clrscr( );

gotoxy( 30, 10 );

cout << "1. Add records";

gotoxy( 30, 11 );

cout << "2. List records";

gotoxy( 30, 12 );

cout << "3. Modify records";

gotoxy( 30, 13 );

cout << "4. Delete records";

gotoxy( 30, 14 );

cout << "5. Recall records";

gotoxy( 30, 15 );

cout << "6. Pack records";

gotoxy( 30, 16 );

cout << "0. Exit";

gotoxy( 30, 18 );

cout << "Your Choice ? ";

cin >> choice;

clrscr( );

switch( choice )

{

case '1':

g.addrec( );

break;

case '2':

g.listrec( );

break;

case '3':

g.modirec( );

break;

case '4':

g.delrec( );

break;

case '5':

g.recallrec( );

break;

case '6':

g.packrec( );

break;

case '0':

g.exit( );

break;

}

} while( choice != 0 );

return 0;

}

void group::group( )

{

file.open( "emp.dat", ios::binary || ios::in || ios::out );

if( !file )

{

cout << endl << "Unable to open file";

exit( );

}

}

void group::addrec( )

{

char ch;

file.seekp( 0L, ios::end );

do

{

cout << endl << "Enter emp code, name, age & salary" << endl;

cin >> p.empcode >> p.name >> p.age >> p.sal;

p.flag = '';

file.write( ( char \* )&p, sizeof( p ) );

cout << "Add another record? (Y/N)";

cin >> ch;

} while( ch == 'Y' || ch == 'Y' );

}

void group::listrec( )

{

int j = 0,a;

file.seekg( 0L, ios::beg );

while( file.read( ( char \* )&p, sizeof( p ) ) )

{

if( p.flag != '\*' )

{

cout <<endl << "Record#" << j++ << setw( 6 )<< p.empcode

<<setw(20)<<p.name<<setw(4<<p.age<<setw(9)<< p.sal;

}

file.clear( );

cout << endl << "Press any key......";

getch( );

}

void group::modirec( )

{

char code[ 5 ];

int count = 0;

long int pos;

cout << "Enter employee code: ";

cin >> code;

file.seekg( 0L, ios::beg );

while( file.read( ( char \* )&p, sizeof( p ) ) )

{

if( strcmp( p.empcode, code ) == 0 )

{

cout << endl << "Enter new record" << endl;

cin >> p.empcode >> p.name >> p.age;

p.flag = '';

pos = count \* sizeof( p );

file.seekp( pos, ios::beg );

file.write( ( char \* )&p, sizeof( p ) );

return;

}

count++;

}

cout << endl << "No employee in file with code = " << code;

cout << endl << "Press any key .....";

getch( );

file.clear( );

}

void group::delrec( )

{

char code[ 5 ];

long int pos;

int count = 0;

cout << "Enter employee code : ";

cin >> code;

file.seekg( 0L, ios::beg );

while( file.read( ( char \* )&p, sizeof( p ) ) )

{

if( strcmp( p.empcode, code ) == 0 )

{

p.flag = '\*';

pos = count \* sizeof( p );

file.seekp( pos, ios::beg );

file.write( ( char \* )&p, sizeof( p ) );

return;

}

count++;

}

cout << endl << "No employee in file with code = " << code;

cout<<endl<<Press any key ....";

getch( );

file.clear( );

}

void group.recallrec()

{

char code[ 5 ];

long int pos;

int count = 0;

cout << "Enter employee code: ";

cin >> code;

file.seekg( 0L, ios::beg );

while( file.read( ( char \* )&p, sizeof( p ) ) )

{

if( strcmp( p.empcode, code ) == 0 )

{

p.flag = '';

pos = count \* sizeof( p );

file.seekp( pos, ios::beg );

file.write( ( char \* )&p, sizeof( p ) );

return;

}

count++;

}

cout << endl << "No employee in file with code = " << code;

cout << endl << "Press any key ....";

file.clear( );

void group::packrec( )

{

ofstream outfile;

outfile.open( "TEMP", ios::out );

file.seekg( 0, ios::beg );

while( file.read( ( char \* )&p, sizeof( p ) ) )

{

if( p.flag != '\*' )

outfile.write((char \*)&p,sizeof(p)));

}

outfile.close( ); file.close( );

remove( "EMP.dat" );

rename( "TEMP", "TEMP.dat" );

file.open( "EMP.dat", ios::binary | ios::in | ios::out | ios::nocreate );

}

void group::exit( )

{

file.close( );

}

OUTPUT

1. Add records";

2. List records";

3. Modify records";

4. Delete records";

5. Recall records";

6. Pack records";

0. Exit";

**28. Write a Program for reading and writing data to and from the file using**

**command line arguments.**

#include<iostream>

#include<fstream.h>

#include<stdlib.h>

using namespace std;

int main(int argc, char \*argv[])

{

int number[9] = {11,22,33,44,55,66,77,88,99};

if(argc!=3)

{

cout<<"argc="<<argc<<"\n";

cout<<"Error in arguments\n";

exit(1);

}

ofstream fout1, fout2;

fout1.open(argv[1]);

if(fout1.fail())

{

cout<<"Could not open the file:"

<<argv[1]<<"\n";

exit(1);

}

fout2.open(argv[2]);

if(fout2.fail())

{

cout<<"Could not open the file:"

<<argv[2]<<"\n";

exit(1);

}

for(int i=0; i<9; i++)

{

if(number[i] % 2 == 0)

fout2<<number[i]<<" ";

else

fout1<<number[i]<<" ";

}

fout1.close();

fout2.close();

ifstream fin;

char ch;

for(i=1; i<argc; i++)

{

fin.open(argv[i]);

cout<<"Contents of "<<argv[i]<<"\n";

do

{

fin.get(ch);

cout<<ch;

}while(fin);

cout<<"\n\n";

fin.close();

}

return 0;

}

OUTPUT

argc=1

Error in arguments

**29. Write a program showing implementation of stack class having the**

**functionality of push, pop operations.**

#include<iostream>

#define MAX 10

using namespace std;

class stack

{

private:

int arr[ MAX ], top;

public:

stack( )

{

top = -1;

}

void push( int item )

{

if( top == MAX - 1 )

{

cout << endl << "Stack is full";

return;

}

top++;

arr[ top ] = item;

}

int pop( )

{

if( top == -1 )

{

cout << endl << "Stack is empty";

return NULL;

}

int data = arr[ top ];

top--;

return data;

}

} ;

int main( )

{

stack s;

s.push( 11 );

s.push( 12 );

s.push( 13 );

s.push( 14 );

s.push( 15 );

s.push( 16 );

s.push( 17 );

s.push( 18 );

s.push( 19 );

s.push( 20 );

s.push( 21 );

int i = s.pop( ); cout << endl << "Item popped=" << i;

i = s.pop( ); cout << endl << "Item popped=" << i;

i = s.pop( ); cout << endl << "Item popped=" << i;

i = s.pop( ); cout << endl << "Item popped=" << i;

return 0;

}

OUTPUT

Stack is full

Item popped=20

Item popped=19

Item popped=18

Item popped=17

**30. Write program to implement a queue class with requried operations/**

**functions.**

#include<iostream.h>

#define MAX 10

using namespace std;

class queue

{

private:

int arr[ MAX ];

int front,

rear;

public:

queue( )

{

front = -1;

rear = -1;

}

void addq( )

{

int item;

if( rear == MAX - 1 )

{

cout << endl << "Queue is full";

return;

}

rear++;

arr[ rear ] = item;

if( front == -1 )

front = 0;

}

int delq( )

{

int data;

if( front == -1 )

{

cout << endl << "Queue is empty";

return NULL;

}

data = arr[ front ];

if( front == rear )

front = rear = -1;

else

front++;

return data;

}

} ;

void main( )

{

queue a;

a.addq( 11 );

a.addq( 12 );

a.addq( 13 );

a.addq( 14 );

a.addq( 15 );

a.addq( 16 );

a.addq( 17 );

a.addq( 18 );

a.addq( 19 );

a.addq( 20 );

a.addq( 21 );

int i = a.delq( );

cout << endl << "Item deleted=" << i;

i = a.delq( );

cout << endl << "Item deleted=" << i;

i = a.delq( );

cout << endl << "Item deleted=" << i;

}

OUTPUT

Queue is full

Item deleted=52

Item deleted=2

Item deleted=3043328

**31. Write a program to implement circular queue class with required**

**operations/ functions.**

#include<iostream.h>

#define MAX 10

using namespace std;

class queue

{

private:

int arr[ MAX ];

int front,

rear;

public:

queue( )

{

front = -1;

rear = -1;

}

void addq( int item )

{

if( ( rear == MAX - 1 && front == 0 )

|| ( rear + 1 == front ) )

{

cout << endl << "Queue is full";

return;

}

if( rear == MAX - 1 )

rear = 0;

else

rear = rear + 1;

arr[ rear ] = item;

if( front == -1 )

front = 0;

}

int delq( )

{

int data;

if( front == -1 )

{

cout << endl << "Queue is empty";

return NULL;

}

else

{

data = arr[ front ];

if( front == rear )

{

front = -1;

rear = -1;

}

else

{

if( front == MAX - 1 )

front = 0;

else

front = front + 1;

}

return data;

}

}

} ;

int main( )

{

queue a;

a.addq( 11 );

a.addq( 12 );

a.addq( 13 );

a.addq( 14 );

a.addq( 15 );

a.addq( 16 );

a.addq( 17 );

a.addq( 18 );

a.addq( 19 );

a.addq( 20 );

a.addq( 21 );

int i = a.delq( );

cout << endl << "Item deleted=" << i;

i = a.delq( );

cout << endl << "Item deleted=" << i;

i = a.delq( );

cout << endl << "Item deleted=" << i;

return 0;

}

**32. Write a program implementing linked list as a class. Also Perform some**

**required operations like inserting, deleting nodes & display the contents of**

**entire linked list.**

#include<iostream>

using namespace std;

class linklist

{

struct node

{

int data;

node \*link;

}\*p;

public:

linklist( );

void append( int num );

void addatbeg( int num );

void addafter( int c, int num );

void del( int num );

void display( );

int counts( );

~linklist( );

} ;

linklist::linklist( )

{

p = NULL;

}

void linklist::append( int num )

{

node \*q,\*t;

if( p == NULL )

{

p = new node;

p->data = num;

p->link = NULL;

}

else

{

q = p;

while( q->link != NULL )

q = q->link;

t = new node;

t->data = num;

t->link = NULL;

q->link = t;

}

}

void linklist::addatbeg( int num )

{

node \*q;

q = new node;

q->data = num;

q->link = p;

p = q;

}

void linklist::addafter( int c, int num )

{

node \*q,\*t;

int i;

for( i = 0, q = p; i < c; i++ )

{

q = q->link;

if( q = NULL )

{

cout << endl << "There are less than " << c << "element";

return;

}

}

t = new node;

t->data = num;

t->link = q->link;

q->link = t;

}

void linklist::del( int num )

{

node \*q,\*r;

q = p;

if( q->data == num )

{

p = q->link;

delete q;

return;

}

r = q;

while( q != NULL )

{

if( q->data == num )

{

r->link = q->link;

delete q;

return;

}

r = q;

q = q->link;

}

cout << endl << "Element" << num << "not found";

}

void linklist::display( )

{

node \* q;

cout << endl;

for( q = p; q->link != NULL; q = q->link )

{

cout << endl << q->data;

}

}

int linklist::counts( )

{

node \*q;

int c = 0;

for( q = p; q != NULL; q = q->link )

c++;

return (c);

}

linklist::~linklist( )

{

node \*q;

if( p == NULL )

return;

while( p != NULL )

{

q = p->link;

delete p;

p = q;

}

}

int main( )

{

linklist ll;

cout << endl << "No. of elements in linked list= " << ll.counts( );

ll.append( 11 );

ll.append( 22 );

ll.append( 33 );

ll.append( 44 );

ll.append( 55 );

ll.addatbeg( 100 );

ll.addatbeg( 200 );

ll.addatbeg( 300 );

ll.addafter( 3, 333 );

ll.addafter( 6, 444 );

ll.display( );

cout << endl << "No. of element in linked list =" << ll.counts( );

ll.del( 300 );

ll.del( 66 );

ll.del( 0 );

ll.display( );

cout << endl << "No. of element in linked list =" << ll.counts( );

return 0;

}

OUTPUT

No. of elements in linked list= 0

**33. Write a program implementing stack & its operations using dynamic memory**

**allocation.**

#include<iostream>

using namespace std;

struct node

{

int data;

node \* link;

};

class stack

{

private:

node \*top;

public:

stack( )

{

top = NULL;

}

void push( int item )

{

node \*temp;

temp = new node;

if( temp = NULL )

cout << endl << "Stack is full";

temp->data = item;

temp->link = top;

top = temp;

}

int pop( )

{

if( top == NULL )

{

cout << endl << "Stack is empty";

return NULL;

}

node \*temp;

int item;

temp = top;

item = temp->data;

top = top->link;

delete temp;

return item;

}

~stack( )

{

if( top == NULL )

return;

node \*temp;

while( top != NULL )

{

temp = top;

top = top->link;

delete temp;

}

}

};

int main( )

{

stack s;

int i = s.pop( );

cout << endl << "Item popped=" << i;

i = s.pop( );

cout << endl << "Item popped=" << i;

i = s.pop( );

cout << endl << "Item popped=" << i;

return 0;

}

OUTPUT

Stack is empty

**36. Write a program implementing Queue stack & its operations using dynamic**

**memory allocation.**

#include<iostream>

using namespace std;

struct node

{

int data;

node \* link;

} ;

class queue

{

private:

node \* front,

\* rear;

public:

queue( )

{

front = rear = NULL;

}

void addq( int item )

{

node \* temp;

temp = new node;

if( temp == NULL )

cout << endl << "Queue is full";

temp->data = item;

temp->link = NULL;

if( front == NULL )

{

rear = front = temp;

return;

}

rear->link = temp;

rear = rear->link;

}

int delq( )

{

if( front == NULL )

{

cout << endl << "queue is empty";

return NULL;

}

node \* temp;

int item;

item = front->data;

temp = front;

front = front->link;

delete temp;

return item;

}

~queue( )

{

if( front == NULL )

return;

node \* temp;

while( front != NULL )

{

temp = front;

front = front->link;

delete temp;

}

}

} ;

void main( )

{

queue a;

a.addq( 11 );

a.addq( 12 );

a.addq( 13 );

a.addq( 14 );

a.addq( 15 );

a.addq( 16 );

a.addq( 17 );

int i = a.delq( );

cout << endl << "Item extracted=" << i;

i = a.delq( );

cout << endl << "Item extracted=" << i;

i = a.delq( );

cout << endl << "Item extracted=" << i;

}

Output:

Item extracted=11

Item extracted=12

Item extracted=13

**35. Write a program to implement Binary search tree using class and traverse**

**the tree using any traversal scheme. In addition to it the class must have**

**capability to copy the contents from one tree to another and compare the**

**contents of two binary trees.**

#include<iostream>

#define TRUE 1

#define FALSE 0

using namespace std;

class tree

{

private:

struct node

{

node \*l;

int data;

node \*r;

}\*p;

public:

tree( );

void searchs( int n, int &found,node \*parent );

void inserts( int n );

void traverse( );

int in( node \*q );

void pre( node \*q );

void post( node \*q );

int operator ==( tree t );

int compare( node \*pp, node \*qq );

void operator =( tree t );

node \*copys( node \*q );

};

tree::tree( )

{

p = NULL;

}

int tree::searchs( int n, int &found, node \*parent )

{

node \*q;

found = FALSE;

parent = TRUE;

if( p == NULL )

return;

q = p;

while( q != NULL )

{

if( q->data == n )

{

found = TRUE;

return;

}

if( q->data > n )

{

parent = q;

q = q->l;

}

else

{

parent = q;

q = q->r;

}

}

}

void tree::inserts( int n )

{

int found;

node \*t,

\*parent;

searchs( n, found, parent );

if( found == TRUE )

cout << endl << "Such a node already exist";

else

{

t = new node;

t->data = n;

t->l = NULL;

t->r = NULL;

if( parent == NULL )

p = t;

else

parent->data > n?parent->l:parent->r = t;

}

}

void tree::traverse( )

{

int choice;

cout << endl << "q.Inorder" << endl << "2. Preorder" << endl<< "3. Postorder" << endl << "4. Your choice ";

cin >> choice;

switch( choice )

{

case 1:

in( p );

break;

case 2:

pre( p );

break;

case 3:

post( p );

break;

}

}

void tree::in( node \*q )

{

if( q != NULL )

{

in( q->l );

cout << "\t" << q->data;

in( q->r );

}

}

void tree::pre( node \*q )

{

if( q != NULL )

{

cout << "\t" << q->data;

pre( q->l );

pre( q->r );

}

}

void tree::post( node \*q )

{

if( q != NULL )

{

post( q->l );

post( q->r );

cout << "\t" << q->data;

}

}

int tree::operator ==( tree t )

{

int flag;

flag = compare( p, t.p );

return ( flag );

}

int tree::compare( node \*pp, node \*qq )

{

static int flag;

if( ( pp == NULL ) && ( q != NULL ) )

{

if( ( pp != NULL ) && ( qq != NULL ) )

{

if( pp->data != qq->data )

flag = FALSE;

else

{

compare( pp->l, qq->l );

compare( qq->r, qq->r );

}

}

}

return ( flag );

}

void tree::operator =( tree t )

{

p = copys( t.p );

}

tree::node \*tree::copys( node \*q )

{

if( q != NULL )

{

t = new node;

t->data = q->data;

t->l = copys( q->l );

t->r = copys( q->r );

return ( t );

}

else

return ( NULL );

}

void main( )

{

tree tt,ss;

int i,

num;

for( i = 0; i <= 6; i++ )

{

cout << endl << "Enter the data for the node to be inserted";

cin >> num;

tt.inserts( num );

}

tt.traverse( );

ss = tt;

ss.traverse( );

if( ss == tt )

cout << endl << "Trees are equal";

else

cout << endl << "Trees are not equal";

}

**36. Write a program to implement the exception handling with multiple*catch***

**statements.**

#include<iostream>

using namespace std;

void test(int x)

{

try

{

if(x==1)

throw x;

else

if(x==0)

throw 'x';

else

if(x==-1)

throw 1.0;

cout<<"End of try-black\n";

}

catch(char c)

{

cout<<"Caught a Character\n";

}

catch(int c)

{

cout<<"Caught an Integer\n";

}

catch(double c)

{

cout<<"Caught a Double\n";

}

cout<<"End of try-catch system\n";

}

int main()

{

cout<<"Testing Multiple Catches\n";

cout<<"x==1\n";

test(1);

cout<<"x==0\n";

test(0);

cout<<"x==2\n";

test(2);

return 0;

}

Output:

Testing Multiple Catches

x==1

Caught an Integer

End of try-catch system

x==0

Caught a Character

End of try-catch system

x==2

End of try-black

End of try-catch system

**37. Write a program to implement the exception handling with rethrowing in**

**exception.**

#include<iostream>

using namespace std;

void divide(double x, double y)

{

cout<<"Inside Function\n";

try

{

if(y==0.0)

throw y;

else

cout<<"Division ="<<x/y<<"\n";

}

catch(double)

{

cout<<"Caught double inside function\n";

throw;

}

cout<<"End of Function\n";

}

int main()

{

cout<<"Inside Main\n";

try

{

divide(10.5,2.0);

divide(20.0,0.0);

}

catch(double)

{

cout<<"Caught double inside main\n";

}

cout<<"End of Main\n";

return 0;

}

Output:

Inside Main

Inside Function

Division =5.25

End of Function

Inside Function

Caught double inside function

Caught double inside main

End of Main

**38. Write a program to implement the exception handling with the**

**functionality of testing the*throw* restrictions.**

#include<iostream>

using namespace std;

void test(int x) throw(int, double)

{

if(x==0)

throw 'x';

else

if(x == 1)

throw x;

else

if(x == -1)

throw 1.0;

cout<<"End of Function Block\n";

}

int main()

{

try

{

cout<<"Testting Throw Restrictions\n";

cout<<"x == 0\n";

test(0);

cout<<"x == 1\n";

test(1);

cout<<"x == -1\n";

test(-1);

cout<<"x == 2\n";

test(2);

}

catch(char c)

{

cout<<"Caught a Character\n";

}

catch(int m)

{

cout<<"Caught an Integer\n";

}

catch(double d)

{

cout<<"Caught a Double\n";

}

cout<<"End of Try-catch system\n";

return 0;

}

Output:

Testting Throw Restrictions

x == 0

terminate called after throwing an instance of 'char'

**39. Write a function template that will sort an array of implicit types like**

**int,float,char etc. it can also sort user-defined objects like strings &**

**date. The necessary classes contains overloading of operators.**

#include <iostream>

#include <string.h>

using namespace std;

class mystring

{

private:

enum

{

sz = 100 // < >

} ;

char str[ sz ];

public:

mystring( char \* s = "" )

{

strcpy( str, s );

}

int operator <( mystring ss )

{

if( strcmp( str, ss.str ) <= 0 )

return 1;

else

return 0;

}

int operator <=( mystring ss )

{

if( strcmp( str, ss.str ) <= 0 )

return 1;

else

return 0;

}

int operator >( mystring ss )

{

if( strcmp( str, ss.str ) > 0 )

return 1;

else

return 0;

}

friend ostream & operator <<( ostream & o,mystring & dd );

} ;

ostream operator <<( ostream & o, mystring & ss )

{

o << ss.str;

return o;

}

class date

{

private:

int day,

mth,

yr;

public:

date( int d = 0, int m = 0, int y = 0 )

{

day = d;

mth = m;

yr = y;

}

int operator <( date dt )

{

if( yr < dt.yr )

return 1;

if( yr == dt.yr && mth < dt.mth )

return 1;

if( yr == dt.yr && mth == dt.mth && day = dt.day )

return 1;

return 0;

}

class date

{

private:

int day, mth, yr;

public:

date( int d = 0, int m = 0, int y = 0 )

{

day = d;

mth = m;

yr = y;

}

int operator <( date dt )

{

if( yr < dt.yr )

return 1;

if( yr == dt.yr && mth < dt.mth )

return 1;

if( yr == dt.yr && mth == dt.mth

&& day < dt.day )

return 1;

return 0;

}

int operator <=( date dt )

{

if( yr <= dt.yr )

return 1;

if( yr == dt.yr && mth <= dt.mth )

return 1;

if( yr == dt.yr && mth == dt.mth

&& day <= dt.yr )

return 1;

return 0;

}

int operator >( date dt )

{

if( yr > dt.yr )

return 1;

if( yr == dt.yr && mth > dt.mth )

return 1;

if( yr == dt.yr && mth == dt.mth

&& day > dt.day )

return 1;

return 0;

}

friend ostream & operator <<( ostream & o, date & dd );

} ;

ostream & operator <<( ostream & o, date & dd )

{

o << dd.day << "\t" << dd.mth << "\t" << dd.yr;

return 0;

}

template<class T> void quick( T \* n, int low, int high )

{

int pos;

if( low < high )

{

pos = split( n, low, high );

quick( n, low, pos - 1 );

quick( n, pos + 1, high );

}

}

template<class T> int split( T \* n, int low, int high )

{

int pos,

left,

right;

T item, t;

item = n[ low ];

left = low;

right = high;

while( left < right )

{

while( n[ right ] > item )

right = right - 1;

while( ( left < right )

&& ( n[ left ] <= item ) )

left = left + 1;

if( left < right )

{

t = n[ left ];

n[ left ] = n[ right ];

n[ right ] = t;

}

}

pos = right;

t = n[ low ];

n[ low ] = n[ pos ];

n[ pos ] = t;

return pos;

}

void main( )

{

float num[]={5.4f,3.23f,2.15f,1.09f,34.66f,23.3452f};

int arr[]={-12,23,14,0,245,78,66,-9};

date dtarr[]={date(17,11,62),date(23,12,65),date(12,12,78)

,date(23,1,69)};

mystring strarr[]={mystring("Kamal"),mystring("Anuj"),

mystring("Sachin"),mystring("Anil")};

int i;

cout << endl << endl;

quick( num, 0, 5 );

for( i = 0; i <= 5; i++ )

cout << num[ i ] << endl;

cout << endl << endl;

quick( arr, 0, 7 );

for( i = 0; i <= 7; i++ )

cout << arr[ i ] << endl;

cout << endl << endl;

quick( dtarr, 0, 3 );

for( i = 0; i <= 3; i++ )

cout << dtarr[ i ] << endl;

cout << endl << endl;

quick( strarr, 0, 3 );

for( i = 0; i <= 3; i++ )

cout << strarr[ i ] << endl;

}

**40. Write a program implementing stack and it’s operations using template**

**class.**

#include<iostream>

using namespace std;

const int MAX = 10;

template<class T>class stack

{

private:

T stk[ MAX ];

int top;

public:

stack( )

{

top = -1;

}

void push( T data )

{

if( top == MAX - 1 )

cout << endl << "Stack is full";

else

{

top++;

stk[ top ] = data;

}

}

T pop( )

{

if( top == -1 )

{

cout << endl << "Stack is empty";

return NULL;

}

else

{

T data = stk[ top ];

top--;

return data;

}

}

};

class complex

{

private:

float real,imag;

public:

complex( float r = 0.0, float i = 0.0 )

{

real = r;

imag = i;

}

friend ostream & operator <<( ostream &o,complex &c );

};

ostream & operator <<( ostream &o, complex &c )

{

o << c.real << "\t" << c.imag;

return o;

}

int main( )

{

stack< int >s1;

s1.push( 10 );

s1.push( 20 );

s1.push( 30 );

cout << endl << s1.pop( );

cout << endl << s1.pop( );

cout << endl << s1.pop( );

stack< float >s2;

s2.push( 3.14 );

s2.push( 6.28 );

s2.push( 8.98 );

cout << endl << s2.pop( );

cout << endl << s2.pop( );

cout << endl << s2.pop( );

complex c1 ( 1.5, 2.5 ),

c2 ( 3.5, 4.5 ),

c3 ( -1.5, -0.6 );

stack< complex >s3;

s3.push( c1 );

s3.push( c2 );

s3.push( c3 );

s3.pop( );

s3.pop( );

s3.pop( );

return 0;

}

OUTPUT

30

20

10

8.98

6.28

3.14

**41. Write a program implementing linked list & some required operations on it**

**using class template.**

#include<string.h>

#include<iostream>

using namespace std;

class emp

{

private:

char name[ 20 ];

int age;

float sal;

public:

emp( char \*n = "", int a = 0, float s = 0.0 )

{

strcpy( name, n );

age = a;

sal = s;

}

friend ostream &operator <<( ostream &s, emp &e );

};

/\*ostream operator <<( ostream &s, emp &e )

{

cout << e.name << "\t" << e.age << "\t" << e.sal;

return s;

}\*/

template<class T>class linklist

{

private:

struct node

{

T data;

node \*link;

}\*p;

public:

linklist( );

~linklist( );

void append( T );

void addatbeg( T );

void addafter( int, T );

void del( int );

void display( );

int count( );

} ;

template<class T> linklist< T >::linklist( )

{

p = NULL;

}

template<class T> linklist< T >::~linklist( )

{

node \*t;

while( p != NULL )

{

t = p;

p = p->link;

delete t;

}

}

template<class T> void linklist< T >::append( T num )

{

node \*q,\*t;

if( p == NULL )

{

p = new node;

p->data = num;

p->link = NULL;

}

else

{

q = p;

while( q->link != NULL )

q = q->link;

t = new node;

t->data = num;

t->link = NULL;

q->link = t;

}

}

template<class T> void linklist< T >::addatbeg( T num )

{

node \*q;

q = new node;

q->data = num;

q->link = p;

p = q;

}

template<class T> void linklist< T >::addafter( int c,T num )

{

node \*q,\*t;

int i;

for( i = q,q = p; i <= c; i++ )

{

q = q->link;

if( q == NULL )

{

cout << endl << "There are less than" << c << "element";

return;

}

}

t = new node;

t->data = num;

t->link = q->link;

q->link = t;

}

template<class T> void linklist< T >::del( int n )

{

node \*q,\*r;

int i = 1;

q = p;

if( n == 1 )

{

p = q->link;

delete q;

return;

}

r = q;

while( q != NULL )

{

if( i == n )

{

r->link = q->link;

delete q;

return;

}

r = q;

q = q->link;

i++;

}

cout << endl << "Element" << n << "not found";

}

template<class T> void linklist< T >::display( )

{

node \*q;

cout << endl;

for( q = p; q != NULL; q = q->link )

cout << q->data << endl;

}

template<class T> int linklist< T >::count( )

{

node \*q;

int c = 0;

for( q = p; q != NULL; q = q->link )

c++;

return ( c );

}

int main( )

{

linklist< int > l1;

cout << endl << "No. of elements in linked list = " << l1.count( );

l1.append( 11 );

l1.append( 22 );

l1.append( 33 );

l1.append( 44 );

l1.append( 55 );

l1.append( 66 );

l1.addatbeg( 100 );

l1.addatbeg( 200 );

l1.addafter( 3, 333 );

l1.addafter( 4, 444 );

l1.display( );

cout << endl << "No. of elements in linked list=" << l1.count( );

l1.del( 200 );

l1.del( 66 );

l1.del( 0 );

l1.del( 333 );

l1.display( );

cout << endl << "no. of elements in linked list = " << l1.count( );

linklist< emp > l2;

cout << endl << "No. of elements in linked list = " << l2.count( );

emp e1 ( "Sanjay", 23, 1100.00 );

emp e2 ( "Rahul", 33, 3500.00 );

emp e3 ( "Rakesh", 24, 2400.00 );

emp e4 ( "Sanket", 25, 2500.00 );

emp e5 ( "Sandeep", 26, 2600.00 );

l2.append( e1 );

l2.append( e2 );

l2.append( e3 );

l2.append( e4 );

l2.append( e5 );

l2.display( );

l2.del( 3 );

l2.display( );

cout << endl << "No. of elements in linked list = " << l2.count( );

l2.addatbeg( e5 );

l2.display( );

l2.addafter( 3, e1 );

l2.display( );

cout << endl << "No. of elements in linked list = " << l2.count( );

return 0;

}

**42. Write a program using mouse service routine (0x33 interrupt). The program**

**should track all mouse activities.**

#include<iostream>

using namespace std;

class mouse

{

private:

union REGS i,

o;

public:

mosue( )

{

initmouse( );

showmouseptr( );

}

void initmouse( )

{

i.x.ax = 0;

int86( 0x33, &i, &o );

}

void showmouseptr( )

{

i.x.ax = 1;

int86( 0x33, \*i, &o );

}

void hidemouseptr( )

{

i.x.ax = 2;

int86( 0x33, &i, &o );

}

void getmousepos( int & button, int & x, int & y )

{

i.x.ax = 3;

int86( 0x33, &i, &o );

button = o.x.bx;

x = o.x.cx;

y = o.x.dx;

}

void restrictmouseptr( int x1, int y1, int x2, int y2 )

{

i.x.ax = 7;

i.x.cx = x1;

i.x.dx = x2;

int86( 0x33, &i, &o );

i.x.ax = 8;

i.x.cx = y1;

i.x.dx = y2;

int86( 0x33, &i, &o );

}

} ;

#include <iostream.h>

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <graphics.h>

#include <conio.h>

#include <dos.h>

#include "mouse.cpp"

#include <fstream.h>

class shapes

{

public:

virtual void draw( )

{

}

virtual void save( ofstream & ft )

{

}

virtual void open( ifstream & fs )

{

}

} ;

class myline:public shapes

{

private:

int sx,

sy,

ex,

ey,

color;

public:

myline( )

{

}

myline( int x1, int y1, int x2, int y2,int clr )

{

sx = x1;

sy = y1;

ex = x2;

ey = y2;

color = clr;

}

void draw( )

{

setcolor( color );

moveto( sx, sy );

lineto( ex, ey );

}

void save( ofstream & ft )

{

ft << "R" << "\n";

ft <<sx<<""<<sy<<""<<ex<<""<< ey << "" << color << "\n";

}

void open( ifstream & fs )

{

fs >> sx >> sy >> ex >> ey >> color;

}

} ;

class myrectangle:public shapes

{

private:

int sx,

sy,

ex,

ey,

color;

public:

myrectangle( )

{

}

myrectangle( int x1, int y1, int x2, int y2,int clr )

{

sx = x1;

sy = y1;

ex = x2;

ey = y2;

color = clr;

}

void draw( )

{

setcolor( color );

rectangle( sx, sy, ex, ey );

}

void save( ofstream & ft )

{

ft << "R" << "\n";

ft <<sx<<""<<sy<<""<< ex << "" << ey << "" << color << endl;

}

void open( ifstream & fs )

{

fs >> sx >> sy >> ex >> ey >> color;

}

} ;

class mycircle:public shapes

{

private:

int sx,

radius,

color;

public:

mycircle( )

{

}

mycircle( int x1, int y1, int r, int clr )

{

sx = x1;

sy = y1;

radius = r;

color = clr;

}

void draw( )

{

setcolor( color );

circle( sx, sy, radius );

}

void save( ofstream & ft )

{

ft << "C" << "\n";

ft << sx << "" << sy << "" << radius << "" << color << endl;

}

void open( ifstream & fs )

{

fs >> sx >> sy >> radius >> color;

}

} ;

struct node

{

void \* obj;

node \* link;

} ;

class objarray

{

private:

node \* head;

public:

objarray( )

{

head = NULL;

}

void add( void \* o )

{

node \* temp = new node;

temp->obj = o;

temp->link = NULL;

if( head == NULL )

head = temp;

else

{

node \* q;

q = head;

while( q->link != NULL )

q = q->link;

q->link = temp;

}

}

void \* getobj( int i )

{

node \* q;

q = head;

int n;

for( n = 1; n < i; n++ )

{

q = q->link;

}

return ( q->obj );

}

int getcount( )

{

int n = 0;

node \* q;

q = head;

while( q != NULL )

{

q = q->link;

n++;

}

return n;

}

~objarray( )

{

node \* q;

q = head;

while( q != NULL )

{

head = head->link;

delete q;

q = head;

}

}

} ;

void mainscreen( )

{

clearddevice( );

rectangle( 0, 0, 639, 479 );

line( 0, 30, 640, 30 );

char \*names[]={"Clear","Open","Save","Line","Rect","Circ",

"Exit"};

int x, i;

for( x = 5, i = 0; x <= 7 \* 90; x += 90, i++ )

{

setcolor( WHITE );

rectangle( x, 5, x + 70, 25 );

floodfill( x + 1, 6, WHITE );

settextstyle( 1, 0, 3 );

setcolor( BLACK );

outtextxy( x + 10, 0, names[ i ] );

}

}

void main( )

{

ifstream fs;

ofstream ft;

int gd = DETECT, gm;

initgraph( &gd, &gm, "c:\\tc\\bgi" );

mainscreen( );

setviewport( 1, 31, 638, 478, 1 );

mouse m;

int button,

x,

y,

flag = 0;

int strptx,

strpty,

endptx,

endpty;

objarray arrl

while( 1 )

{

button = 0;

m.getmousepos( button, x, y );

if( ( button & q ) == 1 ) &&(flag==0))

{

for( t = 5, i = 0; t <= 7 \* 90; t += 90, i++ )

{

if( x >= t && x <= t + 70 && y >= 5&& y <= 25 )

{

index = i;

flag = 1;

break;

}

}

}

int cirnum = random( 16 );

int sx = random( 638 );

int sy = random( 478 );

int ex = random( 638 );

int ey = random( 478 );

int r = random( 200 );

switch( index )

{

case 0:

m.getmousepos( button, x, y );

if( ( ( button & 1 ) == 0 ) && ( flag == 1 ) )

{

clearviewport( );

flag = 0;

}

break;

case 1:

m.getmousepos( button, x, y );

if( ( ( button & 1 ) == 0 ) && ( flag == 1 ) )

{

fs.open( "output.txt", ios::in );

shapres \* ptr;

char a[ 2 ];

while( fs )

{

fs >> a;

if( strcmp( a, "L" ) == 0 )

{

myline \* l = new myline( );

l->open( fs );

arr.add( l );

}

if( strcmp( a, "R" ) == 0 )

{

myrectangle \* r = new myrectangle( );

c->open( fs );

arr.add( c );

}

}

fs.close( );

int count = arr.getcount( );

for( int i = 1; i <= count; i++ )

{

ptr = ( shapres \* )arr.getobj( i );

ptr->draw( );

}

flag = 0;

}

break;

case 2:

m.getmousepos( button, x, y );

if( ( ( button & 1 ) == 0 ) && ( flag == 1 ) )

{

ft.open( "output.txt", ios::out );

int count = arr.getcount( );

shapres \* ptr;

for( i = 1; i <= count; i++ )

{

ptr = ( shapres \* )arr.getobj( i );

ptr->save( ft );

}

ft.close( );

flag = 0;

}

break;

case 3:

m.getmousepos( button, x, y );

if( ( ( button & 1 ) == 0 )&& ( flag == 1 ) )

{

setcolor( clrnum );

moveto( sx, sy );

lineto( ex, ey );

myline \* l = new myline

( sx, sy,

ex, ey,

clrnum

);

{

if( l == NULL )

exit( 1 );

arr.add( l );

flag = 0;

}

break;

case 4:

m.getmousepos( button, x, y );

if( ( ( button & 1 ) == 0 ) && ( flag == 1 ) )

{

setcolor( clrnum );

rectangle( sx, sy, ex, ey );

myrectangle \* r = new myrectangle( sx, sy,

ex, ey, clrnum );

if( r == NULL )

exit( 1 );

arr.add( r );

flag = 0;

}

break;

case 5:

m.getmousepos( button, x, y );

if( ( ( button & 1 ) == 0 ) && ( flag == 1 ) )

{

setcolor( clrnum );

circle( sx, sy, r );

mycircle \*c = new mycircle(sx,sy,r,clrnum );

if( c == NULL )

exit( 1 );

arr.add( c );

flag = 0;

}

break;

}

if( index == 6 )

break;

}

closegraph( );

restorecrtmode( );

}