Program 1

Design a class named MyInteger. The class contains the following:

Implement the class. Write a client program that tests all functions in the class.

Program code

#include<iostream>

using namespace std;

class Myinteger

{

private:

int value;

public:

Myinteger(){value=0;}

Myinteger(int pv){value=pv;};

int getv() const

{return value;}

int isEven() const

{

if (value%2==0)

return 1;

else

return 0;

}

int isOdd() const

{

if (value%2==1)

return 1;

else

return 0;

}

int isPrime() const

{

int check=1;

for(int i=2;i\*i<=value;i++)

{

if(value % i ==0)

{

check=0;

break;

}

}

return check;

}

static int isEven(int pv)

{

if (pv%2==0)

return 1;

else

return 0;

}

static int isOdd(int pv)

{

if (pv%2==1)

return 1;

else

return 0;

}

static int isPrime(int pv)

{

int check=1;

for(int i=2;i\*i<=pv;i++)

{

if(pv % i ==0)

{

check=0;

break;

}

}

return check;

}

static isEven(const Myinteger& m1)

{

if (m1.value %2==0)

return 1;

else

return 0;

}

static int isOdd(const Myinteger& m1)

{

if (m1.value%2==1)

return 1;

else

return 0;

}

static int isPrime(const Myinteger& m1)

{

int check=1;

for(int i=2;i\*i<=m1.value;i++)

{

if(m1.value % i ==0)

{

check=0;

break;

}

}

return check;

}

bool equals(int pv)

{

if(value==pv)

return true;

else

return false;

}

bool equals( const Myinteger& m1)

{

if(value==m1.value)

return true;

else

return false;

}

static int parseInt(const string& s1)

{

return atoi(s1.c\_str());

}

};

int main()

{

Myinteger i(12);

cout<<i.getv()<<endl;

cout<<i.isOdd()<<endl;

cout<<i.isEven()<<endl;

cout<<i.isPrime()<<endl;

cout<<i.isOdd(29)<<endl;

cout<<i.isEven(29)<<endl;

cout<<i.isPrime(29)<<endl;

Myinteger ii(36);

cout<<i.isOdd(ii)<<endl;

cout<<i.isEven(ii)<<endl;

cout<<i.isPrime(ii)<<endl;

cout<<i.equals(29)<<endl;

cout<<i.equals(ii)<<endl;

cout<<i.parseInt("41431012")<<endl;

return 0;

}

Program analysis

1. from the program, we know that the parameter of the member function can be defaulted, int or object

Program result

Program 2

An n-sided regular polygon has n side of the same length, and all its angles have the same degree, i.e., the polygon is both equilateral and equiangular. Design a class named RegularPoloygon that contains the following:

Implement the class. Write a test program that creates three RegularPolygon objects, using the no-arg constructor, using RegularPolygon(6,4) and using RegularPolygon(10,4,5.6,7.8). For each object, display its perimeter an area.

Program code

#include<iostream>

#include<math.h>

using namespace std;

class RegularPoloygon

{

private:

int n;

double side;

double x,y;

public:

RegularPoloygon()

{

n=3;

side=1;

x=0;y=0;

}

RegularPoloygon(int pn,double ps)

{

n=pn;

side=ps;

x=0;y=0;

}

RegularPoloygon(int pn,double ps,double px, double py)

{

n=pn;

side=ps;

x=px;y=py;

}

int getn() const {return n;}

int gets() const {return side;}

int getx() const {return x;}

int gety() const {return y;}

void setn(int pn){n=pn;}

void sets(int ps){side=ps;}

void setx(int px){x=px;}

void sety(int py){y=py;}

double getPerimeter() const

{

return n\*side;

}

double getArea() const

{

return 0.5\*n\*side\*side\*sin(2\*3.14159265358/n);

}

};

int main()

{

RegularPoloygon r1;

RegularPoloygon r2(6,4);

RegularPoloygon r3(10,4,5.6,7.8);

cout<<r1.getPerimeter()<<" "<<r1.getArea()<<endl;

cout<<r2.getPerimeter()<<" "<<r2.getArea()<<endl;

cout<<r3.getPerimeter()<<" "<<r3.getArea()<<endl;

return 0;

}

Program analysis

1. we need the constant accessor functions and mutator functions for all the fields to access private member data.
2. The parameter of a constructor can be variable.

Program result

Program 3

Write a program that displays all the prime numbers less than 120 in decreasing order. Use the StackOfIntegers class to store the prime number (e.g., 2,3,5) and retrieve and display them in reverse order.

Program code

#include<iostream>

using namespace std;

class StackOfIntegers

{

private:

int prime[120];

int size;

public:

StackOfIntegers(){};

void setsize(int ps)

{

size=ps;

}

int getsize()

{

return size;

}

void push(int value)

{

prime[size]=value;

size++;

}

void pull()

{

cout<<prime[size-1]<<endl;

size--;

}

};

int main()

{

StackOfIntegers s1;

s1.setsize(0);

bool check;

int i,j;

for(i=2;i<=120;i++)

{

check=true;

for( j=2;j\*j<=i;j++)

{

if(i % j ==0)

{

check=false;

}

}

if (check==true)

s1.push(i);

}

for(i=s1.getsize();i>0;i--)

s1.pull();

return 0;

}

Program analysis

Since we want to create a stack, that means we have to store the data in it, and then get it from the stack.

Program result

Program 4

Design a class named Location for locating a maximal value and its location in a two-dimensional array. The class contains public data fields row, column and maxValue that store the maximal value and its indices in a two-dimensional array with row and column as int type and maxValue as double type.

Write the following function that returns the location of the largest element in a two-dimensional array. Assume that the column size is fixed.

const int ROW\_SIZE=3;

const int COLUMN\_SIZE=4;

Location locateLargest(const double a[][COLUMN\_SIZE]);

The return value is an instance of Location. Write a test program that prompts the user to enter a two-dimensional array and displays the location of the largest element in the array.

Program code

#include<iostream>

using namespace std;

const int ROW\_SIZE=3;

const int COLUMN\_SIZE=4;

class Location

{

public:

int row,column;

double maxValue;

Location(){};

Location(const double a[ROW\_SIZE][COLUMN\_SIZE])

{

int i,j;

maxValue=a[0][0];

for(i=0;i<ROW\_SIZE;i++)

for(j=0;j<COLUMN\_SIZE;j++)

{

if(a[i][j]>=maxValue)

{

maxValue=a[i][j];

row=i;

column=j;

}

}

}

};

int main()

{

double a[ROW\_SIZE][COLUMN\_SIZE];

int i,j;

for(i=0;i<ROW\_SIZE;i++)

for(j=0;j<COLUMN\_SIZE;j++)

cin>>a[i][j];

Location l1(a);

cout<<l1.row+1<<" "<<l1.column+1<<endl;

return 0;

}

Program analysis

Use the two-dimension array as a perimeter , then use a double loop to find the location and the max value.

Program result