Program 1

Design a class name Rectangle to represent a rectangle. The class contains:

1) Two double data fields named width and height that specify the width and height of the rectangle.

2) A no-arg constructor that creates a rectangle with width 1 and height 1.

3) A constructor that creates a default rectangle with the specified width and height.

4) The accessor and mutator functions for all the data fields.

5) A function named getArea() that returns the area of this rectangle.

6) A function named getPerimeter() that returns the perimeter.

Implement this class. Write a test program that creates two Rectangle objects. Assign width 4 and height 40 to the first object and width 3.5 and height 35.9 to the second. Display the properties of both objects and find their areas and perimeters.

Program code

#include <iostream>

using namespace std;

class Rectangle

{

public:

double width,height;

double getArea(){return width\*height;};

double getPerimeter(){return 2\*(width+height);}

Rectangle(double pw,double ph)

{

width=pw;

height=ph;

}

Rectangle(){width=1;height=1;};

};

int main()

{

Rectangle r1(4,40);

Rectangle r2(3.5,35.9);

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

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

return 0;

}

Program analysis

1. create a class , and define a default constructor.
2. Use new variables to pass the value of width and height

Program result

Program 2

Design a class Account that contains:

1) An int data field named id for the account.

2) A double data field name balance for the account.

3) A double data field named annualInterestRate that stores the current interest rate.

4) A no-arg constructor that creates a default account with id 0, balance 0 and annualInterestRate 0.

5) The accessor and mutator functions for id, balance and annualInterestRate.

6) A function named getMonthlyInterestRate() that returns the monthly interest rate.

7) A function named withDraw(amount) that withdraws a specified amount account.

8) A function named deposit(amount) that deposits a specified amount to the account.

 Implement the class. Write a test program that creates an Account object with an account id of 1122, a balance of 20000, and an annual interest rate of 4.5%. Use the withDraw function to withdraw 2500, use the deposit function to deposit 3000 and print the balance, the monthly interest.

Program code

#include <iostream>

using namespace std;

class Account

{

public:

int id;

double balance;

double annualInterestRate;

Account(){id=0;balance=0;annualInterestRate=0;}

Account(int pi,double pb,double pa){id=pi;balance=pb;annualInterestRate=pa;}

double getMonthlyInterestRate(){return annualInterestRate/12.0;}

void withDraw(double amount){balance=balance-amount;}

void deposit(double amount){balance=balance+amount;}

};

int main()

{

Account a1(1122,20000,0.045);

a1.withDraw(2500);

a1.deposit(3000);

cout<<a1.balance<<" "<<a1.balance \* a1.getMonthlyInterestRate()<<endl;

return 0;

}

Program analysis

1. using void function to change the value of the balance

Program result

Program 3

Design a class named StopWatch. The class contains:

1) Private data fields startTime and endTime with get functions.

2) A no-arg constructor that initializes startTime with the current time.

3) A function named start() that resets the startTime to current time.

4) A function named stop() that sets the endTime to current time.

5) A function named getElapsedTime() that returns the elapsed time for the stop watch in milliseconds.

Implement the class. Write a test program that measures the execution time of sorting 100000 numbers using selection sort.

Program code

#include <iostream>

#include <windows.h>

#include <WinBase.h>

#include <ctime>

using namespace std;

class Stopwatch

{

private:

int startTime,endTime;

public:

Stopwatch(){startTime=GetTickCount();}

void start(){startTime=GetTickCount();}

void stop(){endTime=GetTickCount();}

int getElapsedTime(){return (endTime-startTime);}

};

int main()

{

Stopwatch stop1;

stop1.start();

int i,j,temp;

int s[100000]={0};

for(i=0;i<100000;i++)

for(j=i+1;j<100000;j++)

{

if(s[i]>s[j])

{

temp=s[i];

s[i]=s[j];

s[j]=temp;

}

}

stop1.stop();

cout<<stop1.getElapsedTime()<<endl;

return 0;

}

Program analysis

1. using the GetTickCount() function to get the time of the system
2. call the function twice and put the selection sort between them
3. define the value as private, and define the function as public.

Program result

Program 4

Define the EvenNumber class for representing an even number. The class contains:

1) A data field value of the int type that represents the integer values stored in the object.

2) A no-arg constructor that creates an EvenNumber object for the value 0.

3) A constructor that constructs an EvenNumber object with the specified value.

4) A function named getValue() to return an int value for this object.

5) A function named getNext() to return an EvenNumber object that represents the next even number after the current even number in this object.

6) A function named getPrevious() to return an EvenNumber object that represents the previous number before the current even number in this object.

Implement the class. Write a test program that creates an EvenNumber object for value 16 and invokes the getNext() and getPrevious() functions to obtain and displays these numbers.

Program code

#include <iostream>

using namespace std;

class EvenNumber

{

public :

int value;

EvenNumber(){value=0;}

EvenNumber(int pv){value=pv;}

int getValue(){return value;}

int getNext(){return value+2;}

int getPrevious(){return value-2;}

};

int main()

{

EvenNumber e1(16);

cout<<e1.getPrevious()<<" "<<e1.getValue()<<" "<<e1.getNext()<<endl;;

return 0;

}

Program analysis

1.define two function and return the number before and later of the evennumber.

Program result