1. a. False

b. False

c. True

d. True

e. False

1. Line 7: has a function that resembles a constructor hence making an error
2. Line 5: return type is absent
3. Line 6: constructor must not be const
4. Line 3: semi colon(;) instead of colon (:), line 8: the function has the same name as a constructor but has a return type, line 13: no semi colon to end the class declaration
5. They are
   1. 12
   2. 5
   3. 2
   4. 3
   5. bagType();
6. Answers in code:

bagType::bagType(string style, double length, double width, double height, double price):

style(style), l(length), w(width), h(height), price(price){}

// a

void bagType::set(string style, double length, double width, double height, double price){

    this->style = style;

    this->l = length;

    this->w = width;

    this->h = height;

    this->price = price;

}

//b

void bagType::print() const{

    cout << string(50, '-') << endl;

    cout << "Style: " << style << endl;

    cout << "Length: " << l << endl;

    cout << "Width: " << w << endl;

    cout << "Height: " << h << endl;

    cout << "Price: " << price << endl;

    cout << string(50, '-') << endl;

}

//c

bagType::bagType(){

    style = "";

    l = 0.0;

    w = 0.0;

    h = 0.0;

    price = 0.0;

}

string bagType::getStyle() const{

    return style;

}

double bagType::getPrice() const{

    return price;

}

int main (){

    bagType newBag;

    newBag.set("Tote", 12, 12, 12, 12);

    //d

    newBag.print();

    //e

    bagType tempBag("backPack", 15, 8, 20, 49.99);

    tempBag.print();

}

* 1. i. line 4

ii. line 5

iii. line 5

// b

employee::employee(){

    name = "";

    numOfServiceYears = 0;

    pay = 0.0;

}

// c

employee::employee(string n, int y, double p){

    name = n;

    numOfServiceYears = y;

    pay = p;

}

// d

employee::employee(int y, double p){

    name = "";

    numOfServiceYears = y;

    pay = p;

}

1. Accessors:
2. getNumOfServiceYears

ii. getPay

iii. print

Mutators:

1. setData

#include <iostream>

using namespace std;

class employee

{

public:

    employee();

    employee(string, int, double);

    employee(int, double);

    employee(string);

    void setData(string, int, double);

    void print() const;

    void updatePay(double x);

    int getNumOfServiceYears() const;

    double getPay() const;

private:

    string name;

    int numOfServiceYears;

    double pay;

};

employee::employee(){

    name = "";

    numOfServiceYears = 0;

    pay = 0.0;

}

employee::employee(string n, int y, double p){

    name = n;

    numOfServiceYears = y;

    pay = p;

}

employee::employee(int y, double p){

    name = "";

    numOfServiceYears = y;

    pay = p;

}

employee::employee(string n){

    name = n;

    numOfServiceYears = 0;

    pay = 0.0;

}

// a

void employee::setData(string n, int y, double p){

    name = n;

    numOfServiceYears = y;

    pay = p;

}

// b

void employee::print() const{

    cout << string(50, '-') << endl;

    cout << "Name: " << name << endl;

    cout << "Number of Service Years: " << numOfServiceYears << endl;

    cout << "Pay: " << pay << endl;

    cout << string(50, '-') << endl;

}

// c

void employee::updatePay(double x){

    this->pay = x;

}

// d

int employee::getNumOfServiceYears() const{

    return numOfServiceYears;

}

// e

double employee::getPay() const{

    return pay;

}

int main(){

    // f

    employee emp1;

    emp1.setData("John", 2, 2000);

    emp1.print();

    employee emp2("Jane");

    emp2.print();

}

* 1. 11
  2. 3
  3. One constructor. Yes it can.

Example

temporary::temporary(string desc, double f, double s){

    description = desc;

    first = f;

    second = s;

}

int main(){

    temporary temp;

}

#include <iostream>

#include <iomanip>

using namespace std;

double PI = 3.14;

class temporary

{

public:

    void set(string, double, double);

    void print();

    double manipulate();

    void get(string&, double&, double&);

    void setDescription(string);

    void setFirst(double);

    void setSecond(double);

    string getDescription() const;

    double getFirst()const;

    double getSecond()const;

    temporary(string = "", double = 0.0, double = 0.0);

private:

    string description;

    double first;

    double second;

};

// a

void temporary::set(string desc, double f, double s){

    description = desc;

    first = f;

    second = s;

}

// b

double temporary::manipulate(){

    if (this->description == "rectangle"){

        return this->first \* this->second;

    }

    else if (this->description == "circle"){

        return PI \* this->first \* this->first;

    }

    else if (this->description == "sphere"){

        return 4/3 \* PI \* this->first \* this->first \* this->first;

    }

    else if (this->description == "cylinder"){

        return PI \* this->first \* this->first \* this->second;

    }

    else{

        return -1;

    }

}

// c

void temporary::print(){

    if (this->description == "rectangle"){

        cout << "rectangle: " << "length = " << this->first << ", width = "<< this->second << ", area = "<<this->manipulate() << endl;

    }

    else if (this->description == "circle"){

        cout << "circle: " << "radius = " << this->first << ", area = "<<this->manipulate() << endl;

    }

    else if (this->description == "sphere"){

        cout << "sphere: " << "radius = " << this->first << ", volume = "<<this->manipulate() << endl;

    }

    else if (this->description == "cylinder"){

        cout << "cylinder: " << "radius = " << this->first << ", height = "<< this->second << ", volume = "<<this->manipulate() << endl;

        }

    else{

        cout << "Invalid description" << endl;

    }

}

// d

temporary::temporary(string desc, double f, double s){

    set(desc, f, s);

}

// e

void temporary::get(string& desc, double& f, double& s){

    cout << string(50, '-') << endl;

    cout << "Description: " << this->description << endl;

    cout << "First: " << this->first << endl;

    cout << "Second: " << this->second << endl;

    cout << string(50, '-') << endl;

}

void temporary::setFirst(double f){

    this->first = f;

}

void temporary::setSecond(double s){

    this->first = s;

}

void temporary::setDescription(string desc){

    this->description = desc;

}

string temporary::getDescription() const{

    return this->description;

}

double temporary::getFirst() const{

    return this->first;

}

double temporary::getSecond() const {

    return this->second;

}

int main(){

    temporary object1;

    temporary object2("rectangle", 8.5, 5);

    temporary object3("circle", 6, 0);

    temporary object4("cylinder", 6, 3.5);

    cout << fixed << showpoint << setprecision(2);

    object1.print();

    object2.print();

    object3.print();

    object4.print();

    object1.set("sphere", 4.5, 0);

    object1.print();

}

1. I. An object is created with default values

II. A rectangle with length 3.0 and width 5.0 is created

III. A circle with radius 6.5 is created

IV. A cylinder with radius 6.0 and height 3.5 is created

1. The output would be

Invalid description

rectangle: length = 8.50, width = 5.00, area = 42.50

circle: radius = 6.00, area = 113.04

cylinder: radius = 6.00, height = 3.50, volume = 395.64

sphere: radius = 4.50, volume = 286.13

* 1. Object construction and destruction
  2. String Representations
  3. Attribute Access
  4. Arithmetic Operations
  5. Comparison Operations
  6. Container Operations
  7. Hashing

1. Difference between struct and class in c++

|  |  |
| --- | --- |
| Struct | Class |
| By default all members are public. | By default all members are private. |

#include <iostream>

using namespace std;

class testClass

{

public:

    int sum();

    void print() const;

    testClass();

    testClass(int a, int b);

private:

    int x;

    int y;

};

// a

testClass::testClass(){

    x = 0;

    y = 0;

}

testClass::testClass(int a, int b){

    x = a;

    y = b;

}

int testClass::sum(){

    return x + y;

}

void testClass::print() const{

    cout << "x = " << x << endl;

    cout << "y = " << y << endl;

}

// b

int main(){

    testClass object1;

    testClass object2(5, 10);

    cout << "object1: " << object1.sum() << endl;

    cout << "object2: " << object2.sum() << endl;

    object1.print();

    object2.print();

}

1. The output is

00:00:00

23:13:00

06:59:39

07:00:39

The two times are different

int main() {

    // a

    personType student("Buddy", "Aurora");

    // b

    student.print();

    // c

    student.setName("Susan", "Gilbert");

}

1. In object-oriented programming, using both public and private members in a class is fundamental for achieving **encapsulation**, one of the core principles of OOP
2. A **constructor** is a special member function in a class that is automatically invoked when a new object of that class is created. Its primary purpose is to initialize the object, ensuring that the object's data members are set to valid, predictable values before the object is used.
3. C. ~
4. A **destructor** is a special member function in a class that is automatically invoked when an object is destroyed or goes out of scope. Its main purpose is to perform cleanup tasks, such as releasing resources that the object may have acquired during its lifetime.