

***Assignment Submitted on the Topic:***

Mid Exam.

***Subject:***

OOP.

***Class:***

BSCS 2A.

***Sap id:***

45714.

***Submitted By:***

Ayesha Imran.

***Submitted To:***

Sir Shahzad Sameer.

Question#1:

Find the errors (if any) or give output of the give code:

1.

class base{

private:

int a;

protected:

int b;

public:

void fun(){

cout<<"this works";

}

};

class child: private base{

public:

void fun1(){

base::fun();

base::b=0;

}

};

int main()

{

child c1;

c1.fun();

}

Output:

After running the code, there are 2 errors in the code:

1. 'void base::fun()' is inaccessible within this context.
2. 'base' is not an accessible base of 'child'.

2.

class Tableofcontents{

private:

list <string> items;

public:

int items;

Tableofcontents(){

cout<<"Table of contents is shown\n";

}

void addItem(string item){

items.push\_back(item);

}

};

class Book{

public:

Tableofcontents toc;

list <string> sections;

list <string> chapters;

Book(){

toc=Tableofcontents;

}

};

int main()

{

Book book1=Book();

getchar();

return 0;

}

Output:

“Table of contents” will be shown 2 times.

3.

class A{

int a;

public:

A(int i){

a=i;

}

void assign(int i){

a=i;

}

int return\_value(){

return a;

}

};

int main(int argc, char const \*argv[])

{

A obj;

obj.assign(5);

cout<<obj.return\_value();

}

Output:

After running the code, there is an error in the code:

no matching function for call to 'A::A()'.

4.

class A{

int a;

public:

A(){

cout<<"A's constructor is called";

}

};

class B{

static A a;

public:

B(){

cout<<"B's constructor is called";

}

static A get(){

return a;

}

};

A B::a;

int main(int argc, char const \*argv[])

{

B b;

A a1=b.get();

A a2=b.get();

A a3=b.get();

}

Note:

In the following code how many times the string “A’s constructor called” will be printed?

Output:

In the following code, the string “A’s constructor called” will be printed 1 time.

Question#2:

Give the short answers of the following questions:

1. What is the purpose of access modifiers in OOP languages?

Access modifiers (or access specifiers) are keywords in object-oriented languages that set the accessibility of classes, methods, and other members. Access modifiers are a specific part of programming language syntax used to facilitate the encapsulation of components.

1. If we want to access private members of a class in the child class what do we need to change?

If we want to access private members of a class in the child class, we can use the protected access modifier instead of the private access modifier when declaring those members in the parent class.

1. Determine the accessibility of the functions and data members in the following scenarios:

|  |  |  |
| --- | --- | --- |
| Scenario | Accessible | Not accessible |
| A private data member is declared in a class in accessible by its object in the main function. |  | No |
| A protected function defined in parent class by the functions of the child class. | Yes |  |
| A public data member of the parent class by the object of child class. | Yes |  |

Question#3:

There are five errors (Syntax & Logical) in the code given below. Identify error lines and write correct code.

class B1{

public:

i;

int j;

void g(int){}

};

class B2{

public:

int j;

void g(){}

};

class D:public B1,class public B2{

public:

int i;

};

int main(int argc, char const \*argv[])

{

D dobj;

D \*dptr=&dobj;

dptr->i=5;

dptr->j=10;

dobj.g();

}

Output:

Following are the errors:

1. On line number 7: 'i' does not name a type
2. On line number 13: expected class-name before 'class'
3. On line number 13: expected '{' before 'class'
4. On line number 13: expected identifier before 'public'
5. On line number 13: expected unqualified-id before 'public'
6. On line number 18: aggregate 'D dobj' has incomplete type and cannot be defined
7. On line number 20: invalid use of incomplete type 'class D'
8. On line number 21: invalid use of incomplete type 'class D'

Question#4:

You have to develop a game that has multiple characters. These characters share some common properties like id, name, maximum power and strength. There are other properties as well that they have of their own like Doremon has properties like a list of name gadgets and the name of partner, Benten has the watch name, a list of name powers and total charge of the watch. There are also some common actions that they can perform like walk, jump and eat. Doremon can show gadgets, launch attack and fly. Benten can perform the actions like rotate the watch, fight and drive. Implement the game using inheritance in C++.

Solution:

#include <iostream>

#include <string>

using namespace std;

class Character {

protected:

int id;

string name;

int maxPower;

int strength;

public:

Character(int id, string name, int maxPower, int strength) :

id(id), name(name), maxPower(maxPower), strength(strength) {}

void walk() {

cout << name << " is walking." << endl;

}

void jump() {

cout << name << " is jumping." << endl;

}

void eat() {

cout << name << " is eating." << endl;

}

};

class Doremon : public Character {

private:

string gadgets[4];

string partner;

public:

Doremon(int id, string name, int maxPower, int strength, string gadget1, string gadget2, string gadget3, string gadget4, string partner) :

Character(id, name, maxPower, strength), partner(partner) {

gadgets[0] = gadget1;

gadgets[1] = gadget2;

gadgets[2] = gadget3;

gadgets[3] = gadget4;

}

void showGadgets() {

cout << name << " is showing gadgets." << endl;

for (int i = 0; i < 4; i++) {

cout << "- " << gadgets[i] << endl;

}

}

void launchAttack() {

cout << name << " is launching an attack." << endl;

}

void fly() {

cout << name << " is flying." << endl;

}

};

class Benten : public Character {

private:

string watchName;

string powers[4];

int watchCharge;

public:

Benten(int id, string name, int maxPower, int strength, string watchName, string power1, string power2, string power3, string power4, int watchCharge) :

Character(id, name, maxPower, strength), watchName(watchName), watchCharge(watchCharge) {

powers[0] = power1;

powers[1] = power2;

powers[2] = power3;

powers[3] = power4;

}

void rotateWatch() {

cout << name << " is rotating the watch." << endl;

}

void fight() {

cout << name << " is fighting." << endl;

}

void drive() {

cout << name << " is driving." << endl;

}

};

int main() {

Doremon doremon(1, "Doremon", 100, 50, "Anywhere Door", "Time Machine", "Take-copter", "Bamboo-copter", "Nobita");

Benten benten(2, "Benten", 150, 75, "Omnitrix", "Heatblast", "Wildvine", "XLR8", "Diamondhead", 80);

doremon.walk();

doremon.showGadgets();

doremon.launchAttack();

doremon.fly();

benten.jump();

benten.rotateWatch();

benten.fight();

benten.drive();

return 0;

}