UNIVERSITY OF THE WEST OF ENGLAND





Submitted by: - Albin Pokharel

Module: - Programming in C++

**Task 1: MCQ on Fundamentals of OOP in C++ [2X10 = 20 Marks]**

1. Which of the following is not a classification of class member functions?
2. inspector functions
3. detective functions
4. mutator functions
5. auxiliary function

ANS. Detective functions

1. A \_\_\_\_\_\_\_\_\_\_\_is called automatically each time an object is created.
2. compiler
3. builder
4. constructor
5. destructor

Ans . constructor

1. Which of the following statements is true ?
2. A friendfunction can declare itself to be a friend of a class.
3. A friendfunction can declare itself to be a friend of anotherunction.
4. Both of the above are true.
5. None of the above is true.

ANS: A friendfunction can declare itself to be a friend of a class.

4. A constructor is defined as Computer(int = 512, int = 2, double = 1000.00);. Which of the following is a legal statement that uses the constructor?

1. Computer myMachine(128);
2. Computer myMachine(1234.56);
3. Computer myMachine(256, 899.99);
4. All of the above are legal.

ANS: All of the above are legal.

5. A function that has been declared to be a friend of a class has access to the \_\_\_\_\_\_\_\_\_\_\_data in the class.

1. public
2. private
3. both of the above
4. none of the above

ANS: private

6. When an object goes out of scope, a(n) \_\_\_\_\_\_\_\_\_\_\_is called automatically.

1. destructor
2. constructor
3. overloaded function
4. operating system error message

ANS: destructor

7. Which of the following is a legal example of the way overloading is used in C++?

1. creating any two functions for a class
2. creating two constructors for a class
3. creating two destructors for a class
4. all of the above

ANS :all of the above

8. The primary advantage to overloading functions is \_\_\_\_\_\_\_\_\_\_\_.

1. you can use one function name for every operation you want to perform with a class
2. you can use one class name for many types of items
3. you can use one function name for similar operations, regardless of the data types involved
4. you do not have to write separate code for the bodies of functions that perform similar tasks

ANS: you can use one function name for similar operations, regardless of the data types involved

9. Whenever a class contains two constructors, the constructors are \_\_\_\_\_\_\_\_\_\_\_.

1. default constructors
2. destructors
3. overloaded
4. static

ANS: overloaded

10. A constructor has been defined as FlowerSeed(int = 7);. Which of the following constructors could coexist with the defined constructor without any possible ambiguity?

1. FlowerSeed(int);
2. FlowerSeed();
3. FlowerSeed(int = 10, int = 18);
4. FlowerSeed(int, double);

**ANS:** FlowerSeed();

**Define the Triangle class with members three integers, which correspond to the sides of a triangle (e.g., a, b, and c). Overload the != operator, so that we can compare the perimeters of two triangles (e.g., write a statement such as t1 != t2; to compare the perimeters of t1 and t2). Also, overload the [] operator, so that new values are assigned to the sides of the triangle (e.g., the statement t1[0] = 3; should make the side a of the t1 object equal to 3, while the statement t1[1] = 5; should make the side b equal to 5). Add appropriate functions in the class. Write a program that tests the functionality of the Triangle class.**

#include <iostream>

using namespace std;

class Triangle {

private:

int sides[3]; // sides of the triangle: sides[0] = a, sides[1] = b, sides[2] = c

public:

// Constructor

Triangle(int a = 0, int b = 0, int c = 0) {

sides[0] = a;

sides[1] = b;

sides[2] = c;

}

// Overload != operator to compare perimeters of two triangles

bool operator!=(const Triangle& other) const {

return perimeter() != other.perimeter();

}

// Overload [] operator to assign new values to sides of the triangle

int& operator[](int index) {

if (index >= 0 && index < 3)

return sides[index];

else

throw out\_of\_range("Invalid index");

}

// Calculate perimeter of the triangle

int perimeter() const {

return sides[0] + sides[1] + sides[2];

}

// Display sides of the triangle

void display() const {

cout << "Triangle sides: a = " << sides[0] << ", b = " << sides[1] << ", c = " << sides[2] << endl;

}

};

int main() {

Triangle t1(3, 4, 5);

Triangle t2(5, 12, 13);

Triangle t3;

t1.display();

t2.display();

t3.display();

if (t1 != t2)

cout << "Perimeters of t1 and t2 are not equal." << endl;

else

cout << "Perimeters of t1 and t2 are equal." << endl;

t1[0] = 6; // Change side 'a' of t1

t1[1] = 8; // Change side 'b' of t1

t1.display();

return 0;

}

2.Define the class Product with protected member the code of the product (e.g., code). From Product derive with public access the class Book with private member the title of the book (e.g., title). Each class should contain its own version of show(), which displays the member values. Define the function f() that takes as arguments two pointers to objects and displays their members. Add appropriate functions in the classes so that the following program works.

int main()

{

 Product prod(100); // The code becomes 100.

 Book b("C++", 200); /\* The code becomes 200 and the title becomes C++.

The constructor of the Book should call the constructor of Product. \*/

 ... /\* Declare a pointer that points to prod, another one that points

to b and then call f(). \*/

 return 0;

}

void f(???? \*p1, ???? \*p2) /\* Find the type of the pointers, it must be the

same. \*/

{

 p1->show(); /\* It should display the code of the Product object, that is,

100. \*/

 p2->show(); /\* It should display the code and the title of the Book

object, that is, 200 and C++. \*/

}

**#include <iostream>**

**#include <string>**

**using namespace std;**

**class Product {**

**protected:**

**int code;**

**public:**

**Product(int c) : code(c) {}**

**virtual void show() {**

**cout << "Code: " << code << endl;**

**}**

**};**

**class Book : public Product {**

**private:**

**string title;**

**public:**

**Book(const string& t, int c) : Product(c), title(t) {}**

**void show() override {**

**cout << "Code: " << code << " | Title: " << title << endl;**

**}**

**};**

**void f(Product\* p1, Product\* p2) {**

**p1->show();**

**p2->show();**

**}**

**int main() {**

**Product prod(100);**

**Book b("C++", 200);**

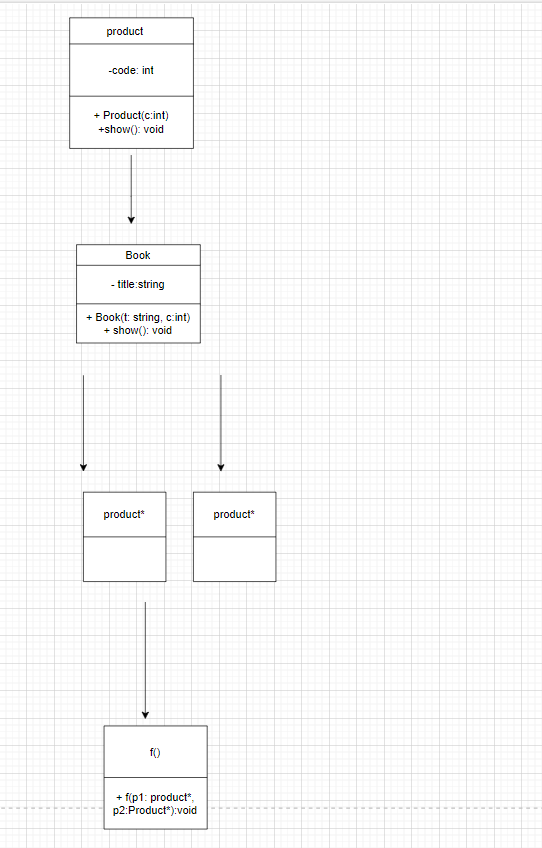
**Product\* p1 = &prod;**

**Product\* p2 = &b;**

**f(p1, p2);**

**return 0;**

**}**

****

**Add appropriate functions in the classes so that the following program works.**

class A

{

private:

 char \*s;

 ...

};

class B : public A

{

private:

 char \*s;

 ...

};

int main()

{

 B b1("abc"); /\* The string should be copied into the b1.s member and

in the s member of the A sub-object. \*/

 B b2(b1);

 B b3; // The s pointers should be initialized to nullptr.

 b3 = b2;

 b3.show(); /\* The program should display the strings that the two s

pointers point to, that is, to display abc twice. \*/

 return 0;

}

ANSWER

#include <iostream>

#include <cstring>

using namespace std;

class A

{

private:

char \*s;

public:

A(const char \*str = nullptr)

{

if (str)

{

s = new char[strlen(str) + 1];

strcpy(s, str);

}

else

{

s = nullptr;

}

}

A(const A &other)

{

if (other.s)

{

s = new char[strlen(other.s) + 1];

strcpy(s, other.s);

}

else

{

s = nullptr;

}

}

A &operator=(const A &other)

{

if (this != &other)

{

if (s)

delete[] s;

if (other.s)

{

s = new char[strlen(other.s) + 1];

strcpy(s, other.s);

}

else

{

s = nullptr;

}

}

return \*this;

}

~A()

{

if (s)

delete[] s;

}

void show()

{

if (s)

cout << s << endl;

}

};

class B : public A

{

private:

char \*s;

public:

B(const char \*str) : A(str)

{

if (str)

{

s = new char[strlen(str) + 1];

strcpy(s, str);

}

else

{

s = nullptr;

}

}

B(const B &other) : A(other)

{

if (other.s)

{

s = new char[strlen(other.s) + 1];

strcpy(s, other.s);

}

else

{

s = nullptr;

}

}

B &operator=(const B &other)

{

if (this != &other)

{

A::operator=(other);

if (s)

delete[] s;

if (other.s)

{

s = new char[strlen(other.s) + 1];

strcpy(s, other.s);

}

else

{

s = nullptr;

}

}

return \*this;

}

~B()

{

if (s)

delete[] s;

}

void show()

{

A::show();

if (s)

cout << s << endl;

}

};

int main()

{

B b1("abc");

B b2(b1);

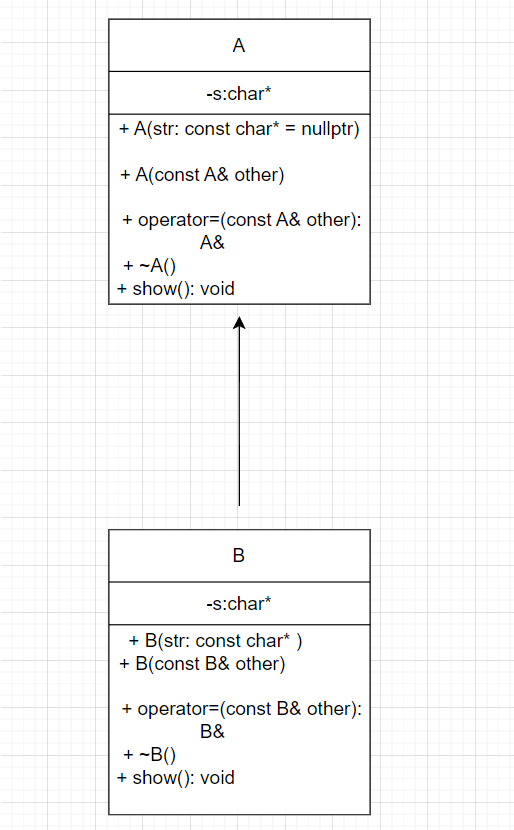
B b3(nullptr); // Explicitly pass nullptr to the constructor

b3 = b2;

b3.show();

return 0;

}



**Task 3: Basics of File Handling**

Write a program that reads the titles of 10 books (use an array of 100 characters) and writes them in a binary file selected by the user. First write the size of the title, then the title. Next, the program should read a title and display a message to indicate if it is contained in the file or not.

ANSWER

#include <iostream>

#include <fstream>

#include <cstring>

using namespace std;

const int MAX\_TITLE\_LENGTH = 100;

const int NUM\_BOOKS = 10;

// Function to write book titles to a binary file

void writeTitlesToFile(const char \*filename, const char titles[][MAX\_TITLE\_LENGTH], int numBooks) {

ofstream outFile(filename, ios::binary);

if (!outFile) {

cout << "Error opening file for writing." << endl;

return;

}

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

int titleLength = strlen(titles[i]);

outFile.write(reinterpret\_cast<const char \*>(&titleLength), sizeof(int));

outFile.write(titles[i], titleLength);

}

outFile.close();

}

// Function to search for a title in the binary file

bool searchTitleInFile(const char \*filename, const char \*searchTitle) {

ifstream inFile(filename, ios::binary);

if (!inFile) {

cout << "Error opening file for reading." << endl;

return false;

}

int titleLength;

char title[MAX\_TITLE\_LENGTH];

while (inFile.read(reinterpret\_cast<char \*>(&titleLength), sizeof(int))) {

inFile.read(title, titleLength);

if (titleLength == strlen(searchTitle) && strncmp(title, searchTitle, titleLength) == 0) {

inFile.close();

return true;

}

}

inFile.close();

return false;

}

int main() {

const char filename[] = "book\_titles.bin";

char titles[NUM\_BOOKS][MAX\_TITLE\_LENGTH];

cout << "Enter the titles of " << NUM\_BOOKS << " books:" << endl;

for (int i = 0; i < NUM\_BOOKS; ++i) {

cout << "Title " << (i + 1) << ": ";

cin.getline(titles[i], MAX\_TITLE\_LENGTH);

}

writeTitlesToFile(filename, titles, NUM\_BOOKS);

cout << "Titles written to file: " << filename << endl;

char searchTitle[MAX\_TITLE\_LENGTH];

cout << "Enter a title to search for: ";

cin.getline(searchTitle, MAX\_TITLE\_LENGTH);

if (searchTitleInFile(filename, searchTitle)) {

cout << "Title found in the file." << endl;

} else {

cout << "Title not found in the file." << endl;

}

return 0;

}

