DAY 4

(4) : Write a c++ program to create a class for a book with a constructor and a destructor

Sol : #include <iostream>

#include <string>

class Book {

private:

std::string title;

std::string author;

int year;

public:

Book(const std::string& title, const std::string& author, int year) : title(title), author(author), year(year) {

std::cout << "Book object created.\n";

}

~Book() {

std::cout << "Book object destroyed.\n";

}

void display() const {

std::cout << "Title: " << title << ", Author: " << author << ", Year: " << year << std::endl;

}

};

int main() {

Book book("The Catcher in the Rye", "J.D. Salinger", 1951);

book.display();

return 0;

}

O/P : Book object created.

Title: The Catcher in the Rye, Author: J.D. Salinger, Year: 1951

Book object destroyed.

(5) : Write a c++ program to create a class for student with a constructor and a destructor

Sol : #include <iostream>

#include <string>

class Student {

private:

std::string name;

int age;

public:

Student(const std::string& name, int age) : name(name), age(age) {

std::cout << "Student object created.\n";

}

~Student() {

std::cout << "Student object destroyed.\n";

}

void display() const {

std::cout << "Name: " << name << ", Age: " << age << std::endl;

}

};

int main() {

Student student("John Doe", 20);

student.display();

return 0;

}

O/P : Student object created.

Name: John Doe, Age: 20

Student object destroyed.

Question-12 : Problem Statement: Implement a simplified online banking system using C++. Create classes to represent bank accounts, customers, and transactions. Focus on:

Customer: Should include attributes such as name and customer ID. Methods might include registering an account.

BankAccount: Attributes include account number, balance, and the customer it belongs to. Include methods for deposit, withdrawal, and balance inquiry.

Transaction: Represents banking transactions, including the type (deposit, withdrawal), amount, and date.

Sol : #include <iostream>

using namespace std;

// class

class Bank {

private:

int acno;

char name[30];

long balance;

public:

void OpenAccount()

{

cout << "Enter Account Number: ";

cin >> acno;

cout << "Enter Name: ";

cin >> name;

cout << "Enter Balance: ";

cin >> balance;

}

void ShowAccount()

{

cout << "Account Number: " << acno << endl;

cout << "Name: " << name << endl;

cout << "Balance: " << balance << endl;

}

void Deposit()

{

long amt;

cout << "Enter Amount U want to deposit? ";

cin >> amt;

balance = balance + amt;

}

void Withdrawal()

{

long amt;

cout << "Enter Amount U want to withdraw? ";

cin >> amt;

if (amt <= balance)

balance = balance - amt;

else

cout << "Less Balance..." << endl;

}

int Search(int);

};

int Bank::Search(int a)

{

if (acno == a) {

ShowAccount();

return (1);

}

return (0);

}

// main code

int main()

{

Bank C[3];

int found = 0, a, ch, i;

for (i = 0; i <= 2; i++) {

C[i].OpenAccount();

}

do {

// display options

cout << "\n\n1:Display All\n2:By Account No\n3:Deposit\n4:Withdraw\n5:Exit" << endl;

// user input

cout << "Please input your choice: ";

cin >> ch;

switch (ch) {

case 1: // displating account info

for (i = 0; i <= 2; i++) {

C[i].ShowAccount();

}

break;

case 2: // searching the record

cout << "Account Number? ";

cin >> a;

for (i = 0; i <= 2; i++) {

found = C[i].Search(a);

if (found)

break;

}

if (!found)

cout << "Record Not Found" << endl;

break;

case 3: // deposit operation

cout << "Account Number To Deposit Amount? ";

cin >> a;

for (i = 0; i <= 2; i++) {

found = C[i].Search(a);

if (found) {

C[i].Deposit();

break;

}

}

if (!found)

cout << "Record Not Found" << endl;

break;

case 4: // withdraw operation

cout << "Account Number To Withdraw Amount? ";

cin >> a;

for (i = 0; i <= 2; i++) {

found = C[i].Search(a);

if (found) {

C[i].Withdrawal();

break;

}

}

if (!found)

cout << "Record Not Found" << endl;

break;

case 5: // exit

cout << "Have a nice day" << endl;

break;

default:

cout << "Wrong Option" << endl;

}

} while (ch != 5);

return 0;

}

O/P : Enter Account Number: 192110361

Enter Name: rakhi

Enter Balance: 2340

Enter Account Number: 192110356

Enter Name: vamshi

Enter Balance: 4765

Enter Account Number: 1921110363

Enter Name: shasha

Enter Balance: 4354

1:Display All

2:By Account No

3:Deposit

4:Withdraw

5:Exit

Please input your choice: 1

Account Number: 192110361

Name: rakhi

Balance: 2340

Account Number: 192110356

Name: vamshi

Balance: 4765

Account Number: 1921110363

Name: shasha

Balance: 4354

Question-13 : .Problem Statement: You are tasked with designing a portion of a video game that involves dynamic resource management, specifically focusing on the creation and deletion of game objects (e.g., enemies, items) during runtime. Each game object requires allocation of resources (like textures and sounds) when created and proper deallocation when it is no longer needed to avoid memory leaks. Implement a class GameObject that demonstrates the use of constructors for resource allocation and destructors for resource deallocation.

Sol : #include <iostream>

class Resource {

public:

Resource() {

std::cout << "Resource allocated" << std::endl;

}

~Resource() {

std::cout << "Resource deallocated" << std::endl;

}

};

class GameObject {

private:

Resource\* resource;

public:

GameObject() {

resource = new Resource();

std::cout << "GameObject created" << std::endl;

}

~GameObject() {

delete resource;

std::cout << "GameObject destroyed" << std::endl;

}

};

int main() {

GameObject\* obj = new GameObject();

delete obj;

return 0;

}

O/P : Resource allocated

GameObject created

Resource deallocated

GameObject destroyed