Software Technical Assessment

|  |
| --- |
| **Question 1**  Consider the following C++ program     * What are the final values of x, y and z? |

**Answer**

The final values for x, y and z are:

x = 9

y = 10

z = 11

|  |
| --- |
| **Question 2**  Consider the following two C++ programs:  **Program 1:**    **Program 2:**     * Explain in few words what is the difference between the two programs? * What will be the final value of the counter variable?   So the in the first program, the argument in **incrementFunction** is passed by value and in the second programe  the argument is passed by reference. When you pass an argument by reference, you are passing  a reference to the variable. Any changes will directly affect the original variable outside the function.  When you pass an argument by value to a function (like in the first program), a copy of the  argument's value is made and passed to the function. Any changes made to the parameter inside the  function do not affect the original value of the argument outside the function.  The first program will output  **counter value is 0**  The second:  **counter value is 1** |
| **Question 3**  We are in 2035, a car rental company provides several types of vehicles (Petrol, Electric and Hybrid) for rental and they offer discounts for Electric and Hybrid cars.   * The basic rental price (before any deduction) is calculated as follows: * The Price per day is fixed to 50$. * For Electric car, 50% discount on the price is offered. * For Hybrid car, 25% discount on the price is offered * Write a code which allows to calculate the rental price of several vehicles. This shal be done by performing the following: * Define a class “Car” with a constructor that initialize PricePerDay(x) and RentalDuration(y) * Defines three subclasses “PetrolCar” and “ElectricCar” and “HybridCar”. * Defines the method getRentalPrice() in the three subclasses, which calculates the rental price.   You code shall work with the following main() function:     * What will be the outputted values (petrolCar.getRentalPrice(), electricCar.getRentalPrice() and hybridCar.getRentalPrice())? |

**Answer**

#include<iostream>

// Base class for vehicles

class Car {

protected:

double \_dailyPrice;

unsigned int \_rentalTime; //gp: unsigned because we won't have negative rental time

public:

Car(double x, int y) : \_dailyPrice(x), \_rentalTime(y) {}

virtual double getRentalPrice() = 0;

};

// Class for the petrol car

class PetrolCar : public Car {

public:

PetrolCar(double x, int y) : Car(x, y) {}

double getRentalPrice() override {

return \_dailyPrice \* \_rentalTime;

}

};

// Class for the elctric car

class ElectricCar : public Car {

public:

ElectricCar(double x, int y) : Car(x, y) {}

double getRentalPrice() override {

return 0.5 \* \_dailyPrice \* \_rentalTime; // 50% discount for electric car...

}

};

class HybridCar : public Car {

public:

HybridCar(double x, int y) : Car(x, y) {}

double getRentalPrice() override {

return 0.75 \* \_dailyPrice \* \_rentalTime; // 25% discount for Hybrid car

}

};

int main() {

PetrolCar petrolCar(50, 7);

ElectricCar electricCar(50, 10);

HybridCar hybridCar(50, 5);

std::cout << "Petrol Car Rental Price: $" << petrolCar.getRentalPrice() << std::endl;

std::cout << "Electric Car Rental Price: $" << electricCar.getRentalPrice() << std::endl;

std::cout << "Hybrid Car Rental Price: $" << hybridCar.getRentalPrice() << std::endl;

return 0;

}

**OUTPUTTED VALUES**

Petrol Car Rental Price: $350

Electric Car Rental Price: $250

Hybrid Car Rental Price: $187.5

|  |
| --- |
| **Question 4**   * Perform a code review on the following C++ program. Note the errors and share your observations:      * Refactor the code and fix the errors you identified in 1). |

**Answer**

* In the main() function, ptAccount1 is declared as a pointer to an Account object but is not initialized. You need to allocate memory for ptAccount1 before using it, like with new, or malloc()
* Also in main() function, ptAccount1 and account2 are accessed with the -> operator, used to access pointers. But, account2 is not a pointer, it's an object. It should be accessed with the . operator.
* In my first point, if we fix the ptAccount1 variable initialization, we need to add a delete or a free to deallocate the memory.
* I would declare both variables for the accounts as objects, and use the same nomenclature for the var name.
* The addInterest function has an issue in interest calculation It multiplies the balance by (1 + rate), but this isn't correct for calculating compound interest, it should use a percentage. IIt should multiply by (1 + rate/100) to properly calculate interest.
* I would add another NEW line at the end of the program for esthetics. Use endl instead of "\n" with STD (optional but better IMO)

#include <iostream>

using namespace std;

class Account

{

private:

int balance;

public:

Account() : balance(0) {}

Account(int initial\_balance) : balance(initial\_balance) {}

int getBalance() const { return balance; }

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

void withdraw(int amount) { balance -= amount; }

// cast as int

void addInterest(double rate) { balance = ((int)(balance \* (1 + rate / 100))); }

};

int main()

{

Account account1;

Account account2(3000);

account1.deposit(100);

account2.withdraw(1000);

account1.addInterest(0.3);

cout << account1.getBalance() << endl;

cout << account2.getBalance()<< endl << endl ;

return 0;

}

|  |
| --- |
| **Question 5**   * The following code shows an excerpt of a method used for reading a configuration from an XML file. Explain in your words why this is not considered clean code and how can this be improved. |

* **Complex Conditional Statements**: The conditional statements used for attribute comparison are overly complex and difficult to understand at a glance. It uses ternary operators within the assignment statements, making the code harder to read and prone to errors.
* **Inefficient Looping**: The loop iterates over XML elements but doesn't seem to update the iterator xParam, which may result in an infinite loop. Additionally, the loop body doesn't seem to perform any meaningful actions on xParam.
* **Inconsistent Naming**: Variable names like xConfiguration, xConfig, and xParam are not descriptive enough, making it difficult to understand their purpose without examining the code in detail.
* **Lack of Comments**: The code lacks comments to explain the purpose and logic behind certain operations, making it challenging for other devs to understand its functionality.

To improve the code, i would do these points:

* Use variable names that accurately describe their purpose and improve readability.
* Simplify Conditional Statements; Replace complex conditional statements with more readable and straightforward comparisons.
* Implement Proper Error Handling: Add try-catch blocks to handle potential exceptions when accessing XML elements, I dont see the catch statement now.
* Clarify Looping Logic: Ensure that loop iterators are updated correctly, and the loop body performs meaningful operations on each iteration.
* Add Comments: Include comments to explain the purpose and logic of the code, especially for complex or non-intuitive sections.

|  |
| --- |
| **Question 6**  Consider the following technical description of an audio device, called the “Passenger Call Unit (PCU)” device:  The “PCU” device provides communication between passengers and the driver in case of emergency in the train. To initiate the communication, the Push button must be pressed by the passenger. When this happens, an audio tone starts being played on the speaker and the “Wait/Listen” Indicator turns ON. This notifies the passenger that the call has been initiated, he can therefore wait until the call is answered by the driver of the train.  When this call is answered by the driver, the audio tone stops being played on the speaker. In addition, the “Talk” indicator turns ON and the “Wait/Listen” Indicator turns OFF. This indicates to the passenger that he can now speak with the driver. Audio from passenger is captured by the Microphone.  When the driver is speaking, the “Talk” indicator turns OFF and the “Wait/Listen” Indicator turns ON in order to indicate to the passenger that he should listen, in this case, the audio coming from the driver can be heard over the speaker.  The call can only be terminated by the driver, when this happens, both “Talk” and “Wait/Listen” Indicators turn OFF.     * Create the state machine diagram describing the states of the “PCU” device. Please describe each of the state and the transitions. * Write a code (C or C++) allowing to implement this state machine.     Normal: In Idle Mode.  Press button: an audio tone starts: Call Init  Driver answers the call; the audio tone stops. "Talkk" indicator turns ON. Call Active  Driver Ends call: Indicators turn OFF and this is the Call Ended State.  CODE  #include <iostream>  #include <string>  using namespace std;  // Enum for device states, I write them as enums and not class enum because I may need to cast them to int for output.  enum State {  Idle = 0,  CallInit = 1,  CallActive = 2,  CallEnded = 3  };  // Enum for device events  enum Event {  PressButton = 0,  AnswerCall = 1,  EndCall = 2  };  // Function prototypes  void updateState(State& currentState, Event event);  int main() {  State currentState = State::Idle;  cout << "State Machine Testv 1.0" << endl << endl;  // run the state machine  while (true) {  // get my inpt  string input;  cout << "Enter event (PressButton (press 'P'), AnswerCall (press 'A'), EndCall (press 'E')): ";  cin >> input;  Event event;  // check for entry  if (input == "P") {  event = Event::PressButton;  }  else if (input == "A") {  event = Event::AnswerCall;  }  else if (input == "E") {  event = Event::EndCall;  }  else {  cout << "Invalid entry, try using 'P', 'A' or 'E'. Please try again." << endl;  continue;  }  // update the new state  updateState(currentState, event);  if (currentState == State::CallEnded) {  cout << "Call ended. Exiting." << endl;  break;  }  }  return 0;  }  void updateState(State& currentState, Event event) {  switch (currentState) {  case State::Idle:  if (event == Event::PressButton) {  cout << "Call initiated. Playing audio tone. Current State is " << currentState << endl;  currentState = State::CallInit;  }  break;  case State::CallInit:  if (event == Event::AnswerCall) {  cout << "Call answered by driver. Call active. Current State is " << currentState << endl;  currentState = State::CallActive;  }  break;  case State::CallActive:  if (event == Event::EndCall) {  cout << "Call ended by driver. Call terminated. Current State is " << currentState << endl;  currentState = State::CallEnded;  }  break;  case State::CallEnded:  // No transitions from CallEnded state  cout << "Call Ended!" << endl;  break;  }  }  I have compiled my code and it works: |