```
#include <iostream>

// Enumerated data type for Rating
enum Rating { Present, Absent, Leave, Withdrawn };

// Data structure for Phone
struct Phone {
    int areaCode;
    int exchange;
    int phoneNumber;
};

// Data structure for ClassAttendance
struct ClassAttendance {
    char className[50];
    Phone phone;
    Rating rating;
};

// Function to set up ClassAttendance with user input
void SetUpAttendance(ClassAttendance &attendance) {
    std::cout << "Enter class name: ";
    std::cin.ignore(); // Ignore previous newline character
    std::cin.getline(attendance.className, 50);

    std::cout << "Enter area code: ";
    std::cin >> attendance.phone.areaCode;

    std::cout << "Enter exchange: ";
    std::cin >> attendance.phone.exchange;

    std::cout << "Enter phone number: ";
    std::cin >> attendance.phone.phoneNumber;

    int ratingValue;
    std::cout << "Enter rating (0 for Present, 1 for Absent, 2 for Leave, 3 for
Withdrawn): ";
    std::cin >> ratingValue;
    attendance.rating = static_cast<Rating>(ratingValue);
}

// Function to set up ClassAttendance with default punctual student information
ClassAttendance SetUpAttendance() {
    ClassAttendance attendance;
    // Default student information
    strcpy(attendance.className, "Punctual Student");
    attendance.phone.areaCode = 123;
    attendance.phone.exchange = 456;
    attendance.phone.phoneNumber = 7890;
    attendance.rating = Present;
```

```
    return attendance;
}

// Function to write out ClassAttendance information
void WriteItOut(const ClassAttendance &attendance) {
    std::cout << "Class Name: " << attendance.className << std::endl;
    std::cout << "Phone: " << "(" << attendance.phone.areaCode << ") "
              << attendance.phone.exchange << "-" << attendance.phone.phoneNumber <<
std::endl;
    std::cout << "Rating: ";

    switch (attendance.rating) {
        case Present:
            std::cout << "Present";
            break;
        case Absent:
            std::cout << "Absent";
            break;
        case Leave:
            std::cout << "Leave";
            break;
        case Withdrawn:
            std::cout << "Withdrawn";
            break;
    }

    std::cout << std::endl << std::endl;
}

int main() {
    ClassAttendance attendance1, attendance2;

    // Call the version of SetUpAttendance with reference parameter
    SetUpAttendance(attendance1);

    // Call the version of SetUpAttendance without input parameters
    attendance2 = SetUpAttendance();

    // Call WriteItOut function for both ClassAttendance variables
    WriteItOut(attendance1);
    WriteItOut(attendance2);

    return 0;
}
```