תרגיל בית 3- חלק יבש

חלק א':

הסבר לשגיאות ותיקונן:

שגיאה ראשונה: השגיאה מתרחשת מכיוון שמנסים לשרשר לאופרטור הפלט את

האות שהיא למעשה שדה בclass B. ההנחה השגויה היא שאופרטור פלט מקבל this אך זה לא המקרה כי הפונקציה הוגדרה כfriend .

תיקון השגיאה: out<< :B: : << b.n;

שגיאה שניה: יש ניסיון השוואה בין שני איברי const B& אבל לא קיים אופרטור השוואה המקבל שני איברים כאלו. האופרטור שהוגדר בשורה 16 מקבל const B& ו- *this שאינו const.

תיקון השגיאה: **bool operator < (const B& rhs) const;**

עתה הפונקציה מקבלת שני איברים שהם const ולכן ההשוואה תוכל להתבצע.

שגיאה שלישית: יש ניסיון לבצע חיבור בין אובייקטים המועברים by value בעוד אופרטור חיבור הוגדר כאופרטור המקבל איברים by reference.

תיקון: B operator + (B b){

כעת האופרטור מקבל את האובייקט by value , תיהיה קריאה לcopy constructor, לפי הגדרת המחלקה הבנאי הדיפולטיבי יספיק (כי השדות מכילים טיפוסים פשוטים), והאובייקט יועבר by value כפי שנדרש.

סעיף ב'

שורת קוד: A* pa = new B();

שרשרת קריאות:

B::B() ;

A::A() ;

הסבר: יצירת אובייקט חדש מסוג B , מכיוון שלא הוגדר בנאי דיפולטיבי, נעשה שימוש בבנאי דיפולטיבי שהקומפיילר יגדיר עבור עצם זה.

מכיוון שמדובר בירושה, הבנאי של B יקרא תחילה לבנאי של מחלקת האב A (יאתחל את השדות שעוברים בירושה).

הבנאי של B ימשיך באתחול השדות האחרים של B (כאן אין כאלו) ואז בביצוע הפקודות בתוך ה scope שלו (ריק במקרה זה).

שורת קוד: cout << "applying function f:" << endl;

שרשרת קריאות: אין, הדפסה בלבד של השורה:

applying function f

שורת קוד: f(*pa).type();

שרשרת קריאות:

A::A(const A& a);

cout<<" A copy ctor"<< endl;

a.type;

A::type();

cout<< "This is A"<<endl;

A::A(const A& a);

cout<<" A copy ctor"<< endl

A::virtual~A();

cout<< "A dtor"<<endl;

הסבר: הפונקציה f מקבלת אובייקט מסוג A -by value ולכן יש קריאה לבנאי ההעתקה של A.

בקוד של בנאי ההעתקה יש קריאה לשורת ההדפסה שתוציא פלט:

" A copy ctor".

לאחר מכן מגוף,הפונקציה f תקרא הפונקציה type() של מחלקה A כי a מטיפוס זה . והיא תדפיס את הפלט:

This is A

לאחר מכן הפונקציה f מחזירה עותק של a ולכן יש קריאה לבנאי ההעתקה של A. הפונקציה רצה כפי שתואר ומדפיסה לאחר ההעתקה:

“ A copy ctor”

ה scope של הפונקציה נגמר ולכן כל הפרמטרים הלוקאליים יושמדו. לשם כך נקרא ההורס של

האובייקט a. לאחר שמשמיד את העותק הלוקאלי של A שיצר, הוא מדפיס :

. A dtor

סיימנו את שרשרת הקריאות עבור שורה זו.

שורת קוד: cout<< “applying function g: ”<<endl

שרשרת קריאות: אין, הדפסה בלבד של השורה :

.applying function g

שורת קוד: g(*pa).type();

שרשרת קריאות:

a.type();

B: : type() ;

cout<< “This is B” << endl;

B: : type();

cout<< “This is B” << endl;

הסבר: הפונקציה g מקבלת איבר מסוג רפרנס ל a ומאחר וpa מצביע על אובייקט מסוג B אז הרפרנס הנדרש מתקבל.

מכיוון שהוגדר override לפונקציה הוירטואלית type , והאובייקט שהועבר הוא מטיפוס B אז הקריאה תיהיה ל type של B.

פונקציה זו תדפיס :

This is B

לאחר מכן חוזרים לפונקציה g שתחזיר את a כרפרנס (לא מחזירה עותק לכן אין קריאה לבנאי העתקה).

עתה, לאחר ש-g(*pa) הופעלה והחזירה את B המתודה type תופעל עליו. (בהתאם לשורת הקוד g(*pa).type().

בהתאם לכך תקרא הפונקציה type של B.

פונקציה זו תדפיס :

This is B

שורת קוד: delete pa;

שרשרת קריאות:

B: : ~B();

cout << ״B dtor״ << endl;

A: : virtual~A();

cout<< “A dtor”<<endl;

הסבר: כאן ניתנת פקודה לשחרר מצביע שהוקצה ידנית. Pa מצביע לB (מחלקת הבן).

ההורס של B יקרא קודם והקוד שבתוך ה scope שלו יתבצע תחילה, ואחר כך יהרוס תא השדות של B (לא קיימים כאלו מלבד השדות שירש מA).לאחר מכן השדות שירש ממחלקת האב יהרסו ולשם כך יקרא ההורס של A.

ה scope של B ריק, מלבד פקודת הדפסה שתבוצע ויודפס למסך :

B dtor

לאחר מכן נקרא ההורס של A שיפעל באותו האופן (scope ריק) מלבד פקודת ההדפסה שתדפיס :

A dtor

שורת קוד: return 0;

סיום התוכנית.

פלט התוכנית:

applying function f:

A copy ctor

This is A

A copy ctor

This is A

A dtor

applying function g:

This is B

This is B

B dtor

A dtor

סעיף ג'

```
#include <iostream>
# include <vector>

class Road {
public:
    double length();
    int speed();
};

//part A//

class Car {
public:
    virtual const double getFuelConsumption(const int speed) const = 0;
};

//part B//

double getPetrol(std::vector<Road> path, const Car& car) {
    double petrol_consumed, total_consumed = 0;
    for (int i = 0; i < path.size(); i++) {
        int path_speed = path[i].speed();
        int path_len = path[i].length();
        petrol_consumed = car.getFuelConsumption(path_speed);
        total_consumed += (path_len / petrol_consumed);
    }
    return total_consumed;
}
```