Provkonstruktion

Syfte

Syftet med detta prov är att bedöma elevernas förståelse av felhantering och testning inom programmering. Eleverna ska kunna identifiera olika typer av fel, tillämpa felsökningstekniker och förstå vikten av enhetstester för att säkerställa kodens kvalitet.

Koppling till styrdokument

Centralt innehåll

”Eleven ska kunna identifiera och lösa fel i programmering samt bedriva tester av programkod för att säkerställa funktionalitet och kvalitet.”

Kunskapskrav

Eleven ska kunna:

- identifiera olika typer av fel och deras orsaker.
- tillämpa lämpliga felsökningstekniker.
- sätta upp och genomföra enhetstester för att säkerställa kodens funktionalitet.

Prov

Faktafrågor

1. Vad är ett syntaxfel?
    - A) Ett fel i programmet som uppstår under körning.
    - **B) Ett fel i skrivsyntaxen av koden.**
    - C) Ett fel som gör att programmet ger felaktiga resultat.
    - D) Ett fel som orsakas av ogiltig indata.
2. Vilken teknik kan användas för att isolera fel i koden?
    - A) Kodkomprimering.
    - **B) Använda debug-verktyg.**
    - C) Modifiera koden utan att testa.
    - D) Ignorera felmeddelanden.
3. Vad är ett logiskt fel?
    - A) Ett fel som leder till programkrasch.
    - **B) Ett fel som ger ett korrekt körande program men felaktiga resultat.**
    - C) Ett fel som kommer från syntaxen i koden.

    - D) Ett fel som orsakas av användarens ogiltiga indata.
4. Hur kan du använda loggar för felsökning?
    - A) Genom att stänga av alla varningar.
    - B) Genom att installera ny programvara.
    - **C) Genom att skriva ut meddelanden för att spåra flödet i programmet.**
    - D) Genom att ignorera felmeddelanden.
5. Vad innebär enhetstestning?
    - **A) Testning av enskilda komponenter eller funktioner i programmet.**
    - B) Testning av hela programmet på en gång.
    - C) Testning av användargränssnitt.
    - D) Testning av installationsproceduren.
6. Vilken typ av tester bör göras vid gränsvärden?
    - A) Tester av syntaxfel.
    - **B) Gränsvärdestester.**
    - C) Logiska tester.
    - D) Test av användarinteraktion.
7. Varför är det viktigt att förstå felmeddelanden?
    - A) För att de är en del av programstrukturen.
    - **B) För att de hjälper till att identifiera och åtgärda fel i koden.**
    - C) För att de kan ignoreras under utvecklingen.
    - D) För att de ger användaren feedback.
8. Vilka är de två huvudtyperna av fel i program?
    - A) Runtime-fel och användarfel.
    - **B) Syntaxfel och runtime-fel.**
    - C) Logiska fel och syntaxfel.
    - D) Logiska fel och konfigurationsfel.
9. Vad är syftet med att inta loggar i programmet?
    - A) För att minska programmet.
    - **B) För att hjälpa vid felsökning och ge insikt i programflödet.**
    - C) För att öka programmets hastighet.
    - D) För att förbättra grafik.
10. Vad syftar gränsvärdestester till att kontrollera?
    - A) Att programmet körs snabbt.
    - **B) Att koden hanterar extrema värden korrekt.**
    - C) Att inga logiska fel finns.
    - D) Att användargränssnittet är lättförståeligt.
11. Vad bör göras först vid felsökning?
    - **A) Analysera och förstå felmeddelandet.**
    - B) Börja skriva ny kod.
    - C) Avinstallera programmet.
    - D) Ignorera felet.

12. Vilken typ av fel är ett runtime-fel?
    - **A) Ett fel som uppstår under körning av programmet.**
    - B) Ett fel i skrivsyntaxen av koden.
    - C) Ett fel som uppstår vid kompilering.
    - D) Ett fel som ger felaktigt resultat.
13. Vad innebär det att implementera testning i koden?
    - A) Att man undviker att skriva tester.
    - B) Att man inte behöver felsöka.
    - **C) Att man förbereder koden för kontinuerlig testning och integration.**
    - D) Att man kan ignorera extrakrav.
14. Vad bör göras när fel upptäckts i koden?
    - A) Utelämna dem.
    - **B) Åtgärda dem och testköra koden igen.**
    - C) Skriva om hela koden.
    - D) Kontrollera om de påverkar användargränssnittet.
15. Varför är kontinuerlig integration viktigt i programutveckling?
    - A) För att man vill minimera koden.
    - **B) För att snabbt få feedback på gjorda ändringar och minska risken för fel.**
    - C) För att öka programmets komplexitet.
    - D) För att uppdatera användargränssnittet.

Resonerande frågor

1. Beskriv skillnaden mellan syntaxfel och runtime-fel.

   Syftet med denna fråga är att låta eleverna visa sin förståelse för olika feltyper och deras inverkan på programmet.

2. Vad är en av de mest effektiva felsökningsteknikerna och varför?

   Genom denna fråga kan eleverna resonera kring strategier och visa förmåga att sammanfatta lämpliga metoder.

3. Vilken roll spelar enhetstestning i programutvecklingscykeln?

   Denna fråga ger eleverna möjlighet att diskutera vikten av testning och dess påverkan på programkvalitet.

4. Hur kan man inkludera felsökning och testning i sitt dagliga arbetsflöde som programmerare?

   Eleverna får analysera och resonera kring praktiska tillämpningar av lärdomarna från lektionen.

5. Diskutera hur felhantering kan påverka användarupplevelsen av en

applikation.

Denna fråga uppmanar eleverna att tänka på användarperspektiv och betydelsen av korrekta felhanteringsmetoder.

6. Ge exempel på hur kontinuerlig integration kan bidra till att förbättra kodkvaliteten.

   Genom att diskutera detta kan eleverna visa förståelse för moderna utvecklingsmetoder och deras betydelse.

7. Vilka etiska aspekter räknar du med att programmeraren måste beakta i samband med felhantering?

   Följdfrågan ger eleverna möjlighet att resonera om ansvar och hållbar utveckling inom programmering.

8. Hur skulle du förklara vikten av felhantering och testning för någon som just börjat programmera?

   Denna fråga låter eleverna sammanfatta sina kunskaper på ett pedagogiskt sätt.

Bedömning

Faktafrågor: 1 poäng per rätt svar, totalt 15 poäng. Resonerande frågor: 2 poäng per rätt svar, totalt 16 poäng. Minst 8 poäng krävs för betyg E, 12 poäng för betyg C (varav minst 3 poäng från resonerande frågor) och 18 poäng för betyg A (varav minst 5 poäng från resonerande frågor).

Tags: Prov