Problem 12: Złożony kod

Punkty: 30

Autor: Matt Hussey, Ampthill, Reddings Wood, Wielka Brytania

Wprowadzenie

W ramach zawodów Code Quest głównym celem jest uzyskanie prawidłowego rozwiązania. Nie jest istotne, czy wasz kod jest czysty i wydajny (w większości przypadków); jeśli wynik jest prawidłowy, dostajecie punkty.

Jednak w praktyce programy komputerowe muszą być tworzone z myślą o dalszych pracach. Pisanie czystego, uporządkowanego kodu i opatrywanie go szczegółową dokumentacją pozwala innym zrozumieć, co dany kod wykonuje, co z kolei ułatwia programistom usuwanie ewentualnych błędów (bugów) po ich wykryciu. Z kolei kod zbyt skomplikowany jest trudny w utrzymaniu, ponieważ potrzeba więcej czasu, by zrozumieć, jakie jest jego zadanie.

Opis problemu

Kod programu można analizować pod kątem jego złożoności. W tym celu używa się dwóch miar, „złożoności cyklomatycznej" i „głębokości zagnieżdżenia". Wasz zespół pracuje z inżynierami systemów Lockheed Martin nad przeanalizowaniem fragmentu kodu w celu ustalenia, czy mieści się on w ramach określonych tymi miarami.

Złożoność cyklomatyczna stanowi miarę tego, jak wieloma ścieżkami można przejść przez dany fragment kodu. Rozpatrzmy poniższy pseudo-kod:

```
Print “Hello”
Print “World”
Print “This is not complex”
```

Ten fragment kodu zawiera trzy zdania, ale można je wykonać w tylko jeden sposób; program nie oferuje żadnej ścieżki alternatywnej. Dlatego złożoność cyklomatyczna kodu wynosi 1.

Teraz rozpatrzmy fragment na następnej stronie:

```
Print “I have a number”
If num > 2
{
     Print “It’s more than 2”
}
Else
{
     Print “It’s 2 or less”
}
```

Tutaj pierwsze polecenie wyświetlania będzie wykonywane zawsze. Ale później program może wykonać jedno z dwóch pozostałych poleceń, w zależności od wartości „num”. W ten sposób stworzono dwie ścieżki przechodzenia fragmentu kodu, co daje złożoność cyklomatyczną równą 2.

Głębokość zagnieżdżenia wskazuje maksymalną liczbę poleceń zagnieżdżonych we fragmencie kodu. Pierwszy fragment powyżej zawiera jedynie wyświetlany tekst; nie zastosowano zagnieżdżania. Jego głębokość zagnieżdżenia wynosi 0. Drugi fragment korzystał z instrukcji warunkowych IF i ELSE. Każde z nich zawierało zagnieżdżone polecenie. Ten fragment ma głębokość zagnieżdżenia równą 1. Poniższy ma głębokość zagnieżdżenia równą 3.

```
If num > 2
{
     If num > 3
     {
          If num > 4
          {
               Print “It’s really big”
          }
          Else
          {
               Print “It’s pretty big”
          }
     }
     Else
     {
          Print “It’s kinda big”
     }
}
Else
{
     Print “It’s not big”
}
```

Pierwsze dwa polecenia wyświetlające tekst są zagnieżdżone w trzech warstwach, co daje głębokość zagnieżdżania równą 3. Choć kod zawiera inne zagnieżdżone polecenia, żadne nie schodzi jeszcze niżej, zatem nie zwiększają głębokości zagnieżdżenia. Ten fragment kodu ma złożoność cyklomatyczną równą

4, ponieważ można go przejść czterema ścieżkami: pierwsza, gdy num wynosi 2 lub mniej, druga, gdy num wynosi 3, trzecia, gdy num wynosi 4 i czwarta, gdy num jest większe od 4.

Przykładowe dane wejściowe

Pierwszy wiersz danych wejściowych programu, otrzymanych przez standardowy kanał wejściowy, będzie zawierać dodatnią liczbę całkowitą oznaczającą liczbę przypadków testowych. Każdy przypadek testowy będzie zawierać:

- Wiersz zawierający trzy liczby całkowite oddzielone spacjami:
    - L, dodatnią liczbę całkowitą odpowiadającą liczbie wierszy kodu poddawanych analizie
    - C, dodatnią liczbę całkowitą odpowiadającą najwyższej dopuszczalnej złożoności cyklomatycznej dla fragmentu kodu
    - N, nieujemną liczbę całkowitą odpowiadającą najwyższej dopuszczalnej głębokości zagnieżdżenia dla fragmentu kodu
- L wierszy zawierających fragment pseudo-kodu, które będą zawierać szereg poleceń wyświetlania (PRINT) i instrukcji warunkowych (IF i ELSE). Wiersze nie będą zawierać wcięć; powyższy przykład opatrzono wcięciami jedynie w celu zademonstrowania stojących za nimi konceptów.
    - Polecenia wyświetlania (PRINT) są w jednym wierszu i zaczynają się od słowa „Print", po którym następuje łańcuch tekstowy w cudzysłowach (").
    - Instrukcje warunkowe IF mogą mieścić się w kilku wierszach. Pierwszy wiersz zaczyna się od słowa „If", po którym następuje warunek. Warunek zawiera zmienną (pisaną wyłącznie małymi literami) i liczbę całkowitą, które porównuje się ze sobą operatorami >, <, >=, <=, == i !=. Kolejny wiersz zawiera lewostronny nawias klamrowy ({). Późniejszy wiersz będzie zawierać domykający, prawostronny nawias klamrowy (}), który kończy instrukcję warunkową.
    - Instrukcje ELSE mogą mieścić się w kilku wierszach i występują jedynie zaraz po instrukcjach IF. Pierwszy wiersz takiej instrukcji zawiera słowo „Else". Kolejny wiersz zawiera lewostronny nawias klamrowy ({). Późniejszy wiersz będzie zawierać domykający, prawostronny nawias klamrowy (}), który kończy instrukcję warunkową.

```
2
9 2 1
Print “I have a number”
If num > 2
{
Print “It’s more than 2”
}
Else
{
Print “It’s 2 or less”
}
11 2 2
If num > 3
{
If num > 4
{
Print “It’s really big”
}
Else
{
Print “It’s pretty big”
}
}
```

Przykładowe dane wyjściowe

W każdym przypadku testowym wasz program musi wyświetlić pojedynczy wiersz zawierający poniższe wartości, oddzielone spacjami:

- Liczbę całkowitą reprezentującą złożoność cyklomatyczną fragmentu kodu
- Liczbę całkowitą reprezentującą głębokość zagnieżdżenia fragmentu kodu
- Jeśli obie te miary są co najwyżej równe wartościom granicznym (odpowiednio C i N), to program ma wyświetlić słowo „PASS”, w przeciwnym razie ma wyświetlić „FAIL”.

```
2 1 PASS
3 2 FAIL
```