Jak wyszukiwać błędy niewidoczne dla kompilatora? (sheadovas/artykuly/programowanie/jak-wyszukiwac-bledy-niewidoczne-dla-kompilatora/)

Lut 19, 2015 / Programowanie (sheadovas/category/artykuly/programowanie/)

Wstęp

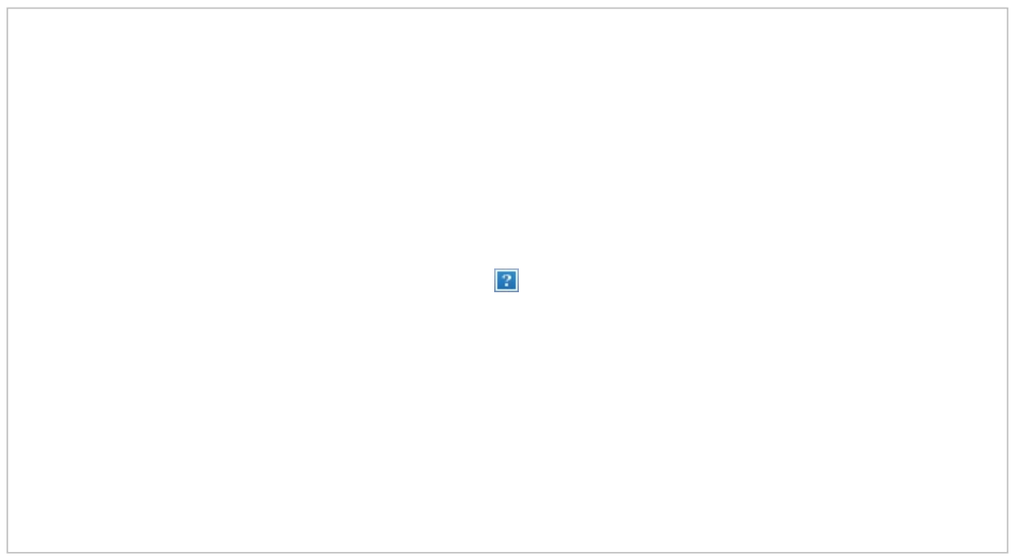
Witam cię w artykule poświęconym wyłapywaniu bądź co bądź trudniejszym do wychwycenia błędom, ponieważ one generowane są dopiero podczas działania programu, a nie jak w przypadku tych znanych głównie na początku programowania „syntax error'ów" i innych podobnych, które potrafi nam z dokładnością co do każdej linii wskazać kompilator.

Wbrew pozorom ten rodzaj błędów jest dość często spotykany, dopiero w nieco większych programach. Sposób na ich znajdywanie jest wiele, ja podam ci te, które prawdopodobnie wcześniej czy później byś odkrył, bo jakby nie było te sposoby są dość łatwe do wymyślenia ich samemu i są wręcz narzucające się.

Najbardziej podstawowe sytuacje

Błędy, o których mówimy najczęściej pojawiają się w przypadku wycieków pamięci, wskazanie na niewłaściwe miejsce w pamięci, czyli np. odwołanie się do 10 elementu tablicy, która jest 9-elementowa, czy nawet dzieleniu przez 0.

Najczęściej logi błędów są średnio czytelne dla użytkownika, znaczy mamy informację jaki był powód „wykrzaczenia" programu, ale czasami już nie mamy informacji o tym, która linia to spowodowała (zależy od debuggera). Jednak coraz nowsze narzędzia dają nam więcej informacji niż kiedyś i warto z nich skorzystać i tak na screenie poniżej mamy podświetloną linię, która spowodowała błąd, a także aktualne wartości używanych zmiennych. Log błędu daje nam także jasną informację, że spowodowaliśmy pęknięcie wszechświata (podzieliliśmy przez 0).



Lecz co w wypadku, gdy nie dostajemy jawnego błędu lub nie daje on nam większych wskazówek?

„Deleted code is debugged code”

Wtedy stosujemy się do podanego wyżej cytatu, czyli „usuwamy” fragmenty kody aż znajdziemy taki fragment, że po jego uruchomieniu program będzie działał poprawnie. Przez usuwanie kodu rozumiemy branie kolejnych fragmentów kodu w komentarze.

Na początku wyszukujemy podejrzany fragment kodu, który naszym zdaniem może powodować błędy, ja zazwyczaj idę od końca kodu (funkcji) i poruszam się (biorę kolejne elementy kodu w komentarze) do góry, dzięki czemu jestem w stanie wyłapać linię kodu która powoduje problemy. Następnie modyfikuję daną linię (fragment) dopóki nie pozbędę się problemu.

Zasada jest prosta i narzucająca, jednak dość często bardzo czasochłonna, bo nie zawsze znalezienie kodu, który może powodować błędy jest takie łatwe. W dodatku trzeba zadbać, żeby kod, który „usuniemy” a raczej jego brak nie spowodował błędów w dalszej części programu.

A co jeżeli okaże się, że w komentarzach jest cały kod i nic więcej nie pozostało do zakomentowania? Wtedy trzeba napisać kod od nowa ;)

Błędy „logiczne”

Teraz pomówimy o nieco innym typie błędów, które nazywam błędami logicznymi, czyli takimi, które w zasadzie nie są błędami (bo nie generują ich w żadnej postaci), a raczej wynikają z naszego błędnego rozumowania i kod nie działa tak jakbyśmy chcieli.

Nie powiem tutaj nic odkrywczego, a z racji że każda sytuacja jest nieco inna ciężko tutaj znaleźć jakiś przepis czy regułę, jednak zawsze w moim przypadku się sprawdza wyświetlanie wartości zmiennych w danym momencie działania programu. Pomaga to dość bardzo znalezienie źródła niedziałania czegoś.

Oczywiście nikt nie broni nam wymieszać tych dwóch sposobów na znalezienie problemów, z czego korzystam dość często.

Epilog

Prawdopodobnie sam wpadłeś na te sposoby wyszukiwania problemów/błędów, jednak jeżeli ci się ten artykuł przydał to bardzo się z tego powodu cieszę.

Jeżeli macie jakieś własne „tricki” na szybsze wyszukiwanie trudniejszych do zidentyfikowania błędów, a może chcecie podzielić się swoją opinią o tym wpisie to jak zwykle zapraszam do komentowania.