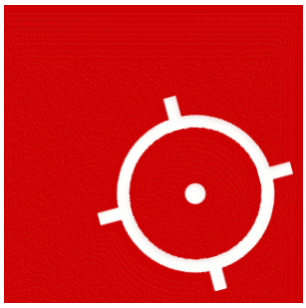

testerzy.pl

Błędy podczas tworzenia kodu źródłowego

Zagadnienie występowania błędów w kodzie źródłowym leży w gestii programistów i/lub testerów białoskrzynkowych. Wiedza z tego zakresu może jednak być przydatna nie tylko dla nich, ale również dla wszystkich niezajmujących się profesjonalnie sprawdzaniem kodu. Artykułu ten nie jest powiązany z konkretnym językiem programowania ani też językiem skryptowym. Przedstawia ogólne założenia sprawdzania kodu. Odnośniki na końcu mogą pomóc w zdobyciu bardziej szczegółowej wiedzy.

Informacje kompilatora.

Kompilator tłumaczy kod źródłowy na język maszynowy. Cała ta operacja kończy się zazwyczaj wylistowaniem błędów i/lub też ostrzeżeń. Są to informacje narzędzia, co do jakości naszego kodu.

Kiedy kompilator odmawia wykonania polecenia jest to znak, że stworzony przez nas program nie ma prawa działać. W znalezieniu błędu pomaga zazwyczaj wskazanie linii błędnego kodu oraz informacja o rodzaju błędu (często podany jako wartość liczbowa). W większości przypadków nasz błąd polega na drobnych niedociągnięciach typu brak znaku końca linii. Częstym przypadkiem jest to, że błąd znajduje się w linii poprzedzającej tą, w której według narzędzia jest błąd. Należy, więc od niej zacząć szukanie błędu.

Różnice między kompilatorami prowadzą do tego, że co w jednym z nich jest poważnym błędem w innym będzie już tylko ostrzeżeniem.

Ostrzeżenia pojawiają się, gdy nasz kod jest wykonalny, ale kompilator dostrzega np. błędy składni, niezdefiniowaną zmienną itp. Ich naprawienie jest konieczne by uchronić się przed poważniejszymi kłopotami w przyszłości, które nie muszą wcale być łatwe do znalezienia. Ważne jest, aby zaczynać od początku listy ostrzeżeń. Informacja o nie zadeklarowanej zmiennej będzie się pojawiać za każdym razem, gdy będzie ona wywoływana w kodzie. Naprawienie tej pomyłki na początku skróci listę ostrzeżeń. Dużym błędem początkujących programistów jest zostawianie naprawy trudniejszych do zlokalizowania błędów późniejszym czasie. Może to tylko uruchomić kaskadę narastającej ilości błędów w przyszłości.

Informacje linkera

Większość kompilatorów jednocześnie również linkuje ze sobą pliki z kodem i bibliotekami jednak nie wszystkie. Dlatego też wyróżniamy tu osobo proces łączenia.

Informacja linkera świadczy o tym, że nasz kod może zostać skompilowany, ale nastąpiły problemy z ze znalezieniem pewnych funkcji lub bibliotek. Przykładem takiego błędu może być wezwanie funkcji (lub metody) za pomocą błędnej nazwy lub też uwzględnienie biblioteki, która nie zostaje następnie "dokompilowana" przy użyciu flag linkera.

Błędy podczas wykonywania programu

Tego typu błędy dostrzegamy podczas wykonywania programu. Dzielimy je na:

1) błąd krytyczny – występuje, gdy wykonanie programu zostaje przerwane np. gdy niespodziewanie program chce podzielić przez zero lub, gdy próbuje uzyskać dostęp do obszarów pamięci dla niego niedozwolonych.
2) Błąd logiczny – kiedy program nie robi tego, co powinien. Przykład: niekończąca się pętla.

Znalezienie tego typu błędów jest trudniejsze i wymaga już użycia programów wspierających znajdywanie błędów (debugger). Pomocne może również być użycie przed kompilacją narzędzia sprawdzającego dzielenie przez zero czy nieskończone pętle. W języku C jest to np. Lint.

Błędy użytkowania programu

Mimo opinii programistów, że tego typy błędów są błędami użytkownika powinniśmy pamiętać, że to twórca kodu powinien nim zapobiec. Oczywiste jest, że jeśli użytkownik zostanie poproszony o podanie liczby z konkretnego przedziału to na 50% poda wartość z poza niego. Znany jest przykład pewnego profesora informatyki, który pierwszą rzecz, jaką robił po uruchomieniu programu to naciskał wszystkie klawisz naraz. Efekt: 80% ocen niedostatecznych.

Wyjątki

Błędy występujące niezależnie od kodu zazwyczaj wykrywane przez sprzęt lub system operacyjny.

Przeczytaj więcej zanim zaczniesz analizę kodu:

C/C++
http://oopweb.com/CPP/Documents/DebugCPP/VolumeFrames.html
http://www.cs.princeton.edu/courses/archive/spring03/cs217/lectures/Robust.pdf

Python
http://www.python.org/doc/current/lib/

Java
http://java.sun.com/j2se/1.4.2/docs/index.html

Lint
http://en.wikipedia.org/wiki/Lint_programming_tool

Kompilatory i linkery
http://www.cs.bu.edu/teaching/cpp/debugging/errors/

Analiza kodu ogólnie
http://www.cs.princeton.edu/~bwk/testing.html