Ciekawe rodzaje błędów programistycznych

*Krzysztof PIECUCH**

* doktorant, Wydział Matematyki i Informatyki, Uniwersytet Wrocławski

Mówi się, że nie ma ludzi nieomylnych. Zgubiony minus w równaniu może doprowadzić do szału niejednego matematyka czy fizyka. Podobnie jak zgubiony średnik, znak równości czy inna literówka potrafi sprawić, że informatycy poświęcają pisaniu programu więcej czasu, niż mieli zamiar. Dziś chciałbym porozmawiać o błędach w programowaniu. Ale nie takich zwykłych. W niniejszym artykule przedstawię kilka kategorii błędów, które zdarzają się bardzo rzadko, ale gdy już się zdarzą, wprawiają w zdumienie nawet doświadczonych programistów.

Moim ulubionym rodzajem błędu jest tak zwany *Heisenbug*. Gdy w naszym programie pojawia się błąd, próbujemy go znaleźć za pomocą specjalnych programów albo wypisując po kolei wartości pewnych zmiennych. Proces znajdowania błędów w oprogramowaniu nazywamy debugowaniem. Heisenbug to błąd, który znika, gdy próbujemy debugować nasz program. Nazwa pochodzi od zasady nieoznaczoności Heisenberga. Pierwszy raz spotkałem się z tym błędem na studiach, gdy pisałem projekt w języku C. Napisany przeze mnie program nie wypisywał oczekiwanych wyników. Aby znaleźć w nim błąd, wypisałem na ekran poszczególne kroki algorytmu. Gdy to zrobiłem, program – ku mojemu zaskoczeniu – wypisał poprawne wyniki. Okazało się, że program zawierał zmienną, która nie była zainicjowana (nie nadałem jej wartości na początku programu). Gdy uruchamiamy funkcję, jej zmienne umieszczane są na tak zwanym stosie. Jeśli zmienna nie jest inicjowana, ma wartość w pewnym sensie „losową”.

System operacyjny przydziela naszej zmiennej miejsce w pamięci RAM. W tym miejscu mogą znajdować się jakieś dane, które były wykorzystywane przez nas albo inny program. Jeśli zapomnimy nadać zmiennej wartość – będzie ona zawierać właśnie takie „śmieci”. No dobrze. Ale dlaczego program zaczął działać, gdy wyświetlałem dodatkowe informacje? Aby wyświetlić informację na ekran, użyłem specjalnej funkcji. Ona również ma swoje zmienne i one również zostają umieszczone na stosie. Gdy funkcja wypisująca zakończyła działanie, uruchamiałem swoją funkcję z niezainicjowaną zmienną. Okazało się, że funkcja wypisująca zerowała to miejsce, gdzie później trafiała moja nieszczęsna zmienna. Niezwykły zbieg okoliczności sprawił, że program zaczął działać poprawnie.

Kolejnym rodzajem błędu jest *Ghost in the Code*, czyli duch w programie. Jest to rodzaj błędu, który może pozostać nieodkryty przez wiele, wiele lat. Jak powszechnie wiadomo, programiści nie dowodzą poprawności swoich programów. Oni je testują. Może się zdarzyć, że pomimo wielu testów błąd w programie nadal pozostanie. Może się też zdarzyć, że informatycy nawet udowodnią poprawność swojego algorytmu, a błąd w jego implementacji nadal pozostanie. Tak było w przypadku bardzo popularnego algorytmu. *Wyszukiwanie binarne*, bo o nim mowa, polega na znajdowaniu zadanego elementu w posortowanej tablicy poprzez dzielenie tablicy na połowę i sprawdzanie środkowego elementu. Jeśli środkowy element jest mniejszy od naszej szukanej wartości, to musi się ona znajdować w prawej części tablicy, w przeciwnym przypadku – w lewej części. Algorytm został wymyślony w 1946 roku. Jak podaje Jon Bentley w swojej książce *Perełki oprogramowania* – pierwsza poprawna implementacja tego algorytmu została przedstawiona w 1962 roku. Książka została wydana w 1986 roku. Zawiera ona implementację tego algorytmu wraz z dowodem poprawności. Twórcy języka Java skorzystali z tej implementacji w swoim języku. Jakie było zdziwienie społeczności informatyków, gdy okazało się, że w 2006 roku algorytm wyszukiwania binarnego w Javie uległ awarii. Feralną linijką okazała się instrukcja licząca średnią arytmetyczną dwóch liczb. Informatycy korzystali z prostego wzoru $(l + r)/2$. Zastosowany wprost, na komputerach niekoniecznie działa poprawnie. Komputery 32-bitowe przeznaczają na zapisanie liczby naturalnej 32 bity. Umożliwia to komputerom zapisanie liczby z przedziału od 0 do $2^{32} - 1$. Jeżeli zarówno wartość zmiennej l, jak i r jest bliska prawemu krańcowi tego przedziału, może się okazać, że suma nie zmieści się w 32 bitach. Aby pozbyć się tego problemu, informatycy obecnie korzystają z następującego wzoru na średnią arytmetyczną: $(r - l)/2 + l$. Ponieważ algorytm gwarantuje, że wartość zmiennej r jest zawsze większa od zmiennej l, więc wszystkie operacje możemy wykonać na 32 bitowych zmiennych.

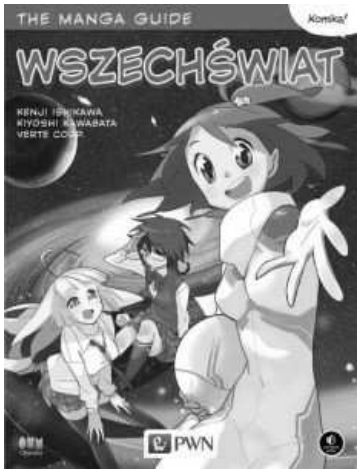
Ostatnim rodzajem błędu, jaki chciałem przedstawić, jest *Phase of the Moon*. Nazwa wzięła się z żartu o tym, że błąd występuje tak bardzo nieregularnie, że najwyraźniej musi zależeć od fazy księżyca. Jak wspomniałem wcześniej – informatycy zazwyczaj nie dowodzą poprawności swoich programów, oni je testują. Kłopot w tym, że czasami bardzo trudno przetestować wszystkie błędy w programach. Przykłady? Proszę bardzo. Do każdego programu zwyczajowo dołącza się dziennik zmian (tak zwane changelogi). Są to pliki tekstowe opisujące, jakie dodatkowe funkcje zostały dodane, albo jakie błędy zostały usunięte w kolejnych wersjach programu. Moim faworytem jest tutaj program *Radiorecorder Web GUI*. W jego changelogu możemy przeczytać – „Naprawiono błąd: od teraz można nagrywać także w październiku”. Najwyraźniej twórcy zapomnieli przetestować, czy ich program będzie działał w październiku, bo w sumie

kto normalny by to sprawdzał. Jaki błąd popełnili twórcy oprogramowania i dlaczego akurat październik – możemy się tylko domyślać. Inny program działał dobrze przez 362 dni w roku. Problemem były tylko środy, tylko we wrześniu i tylko po 9-tym dniu tego miesiąca. Program daty zapisywał po angielsku w formacie: „Wednesday, September 22 2008”, ale programista zarezerwował o jeden bajt za mało dla tego napisu. Dlatego problem pojawiał się tylko, gdy napis był najdłuższy w danym roku – a to zdarzało się tylko trzy razy w roku. Tylko wtedy, gdy nazwa tygodnia była najdłuższa, nazwa miesiąca była najdłuższa i numer dnia w miesiącu zajmował dwa bajty. Inne błędy związane z czasem dotyczą tego, w jaki sposób komputery go liczą. Niektóre systemy operacyjne umożliwiają mierzenie czasu za pomocą specjalnych funkcji. Trzeba na nie czasami bardzo uważać. Na przykład liczą one ilość milisekund, które minęły od czasu włączenia się systemu operacyjnego.

Licznik liczy modulo 2^{32}, dodanie jedynki do $2^{32} - 1$ i otrzymanie zera nazywamy przekręceniem się licznika.

Jeśli zmienna, w której ta wartość jest przechowywana, jest 32 bitowa to przekręci się ona po 49 dniach działania non-stop systemu. Mało osób zostawia komputer włączony na tak długo. Z pewnością też nie robią tego wszyscy testerzy oprogramowania. Dlaczego warto mówić o takich rzeczach? Dlatego, że może to mieć kolosalne konsekwencje. Na przykład z powodu błędów w liczeniu czasu, 25 lutego 1991 roku zawiódł amerykański system obrony przeciwlotniczej PATRIOT. Tego dnia iracka rakieta typu SCUD trafiła w barak w Dhahranie (Arabia Saudyjska), zabijając 28 amerykańskich żołnierzy.

Warto również pamiętać, jaki strach wywołał tak zwany problem roku 2000. Problem ten związany był z faktem, że w pierwszych komputerach rok zapisywano, używając jedynie ostatnich dwóch cyfr co czasem powodowało pomylenie roku 1900 z rokiem 2000. Do dzisiaj inni naukowcy śmieją się z nas, że informatyka to jedyna dziedzina nauki, która nie przewidziała nadejścia XXI wieku.

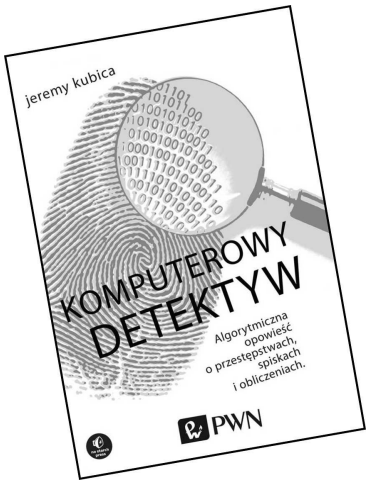


The Manga Guide *Wszechświat*

Bardzo lubię komiksy i bardzo się ucieszyłam, jak tata dał mi nowy komiks do przeczytania. Tym większa była moja radość, gdy okazało się, że jest to manga. Akcja komiksu jest wciągająca. Zainteresowały mnie japońskie legendy, jak, na przykład, legenda o zbieraczu bambusów. Bohaterowie komiksu w ciekawy sposób dyskutują na temat odkryć naukowych, np. o tym, czy Słońce krąży wokół Ziemi, czy Ziemia wokół Słońca, albo jak Eratostenes doszedł do przekonania, że Ziemia jest kulista. Bardzo przyjemna czcionka pozwala szybko czytać.

Kiedy mój tata zaczął przeglądać książkę, to też długo nie mógł się od niej oderwać. Bardziej od komiksów interesowały go teksty naukowe wplecione między nimi. Jego zdaniem dobór tematów jest bardzo dobry. Są opisane bardzo przystępnie, przejrzyście, w sposób przemawiający do wyobraźni. Jak powiedział, nie znalazł niczego, do czego mógłby się przyczepić i nawet sam się czegoś ciekawego dowiedział.

Zosia CHARZYŃSKA, 10 lat



Szukajcie (mądrze), a znajdziecie (szybko)

Cyberprzestępcy, strzeżcie się! Najmniejszy Wasz ślad nie ukryje się bowiem przed Frankiem Bieżącym, prywatnym detektywem, specjalizującym się w algorytmach wyszukiwania. Nic zatem dziwnego, że po kradzieży dokumentów z posterunku policji jej kapitan zwraca się właśnie do Bieżącego z prośbą o pomoc w ujęciu sprawców. O jego przygodach na drodze do wyjaśnienia zagadkowego rabunku możecie przeczytać na stronach *Komputerowego Detektywa* autorstwa Jeremy'ego Kubicy. Śledząc zmagania detektywa z przeciwnościami losu (często w uroczy sposób absurdalnymi), Czytelnik mimochodem opanuje podstawowe algorytmy wyszukiwania oraz związane z nimi struktury danych (dowie się, na przykład, dlaczego lepiej mieć kolejkę brudnych naczyń niż ich stos). Książka lekka, nietrudna i pouczająca – warto ją wyszukać (dowolnym algorytmem) w pobliskiej księgarni!

Ł. R.