Odp. Na Apel Twórcy j. C++ Bjarne Stroustrup

autor:	Jacek Marcin Jaworski
pseudonim:	Energo Koder Atlant
pomocnicy autora:	BRAK
miejsce:	Pruszcz Gd.
utworzono:	2025-03-24, pon.
wersja: 2121 z dnia:	2026-06-28
program składu:	Libre Office Writer
sys. op.:	Kubuntu
źródło:	energokod.gda.pl

Ten dok. w wer. PDF jest podpisany cyfrowo wolnym prog. GNU gpg dostępnym bezpłatnie ze s. www.gnupg.org/download. Instrukcja w j. pol. jak się posługiwać prog. GNU gpg w sys. Linuks z rodz. Debian/Ubuntu znajduje się w całkowicie bezpłatnym dok. PDF Konf. i Zabezp. Sys. Op. z Rodz. Ubuntu w roz. "Podpisywanie Dok. PDF".

Spis treści

Wstęp

Oto moja odp. na apel autora j. C++ Bjarne Stroustrup z Danii, w którym wzywa społeczność C++ do obrony tego j. prog. O jego apelu dowiedziałem się z art. [tworca-c++-prosi-o-pomoc].

1.1 Skróty

ADA	https://pl.wikipedia.org/wiki/Ada_(j%C4%99zyk_programowania)
art.	artykuł
b.	bardzo
d.	dzień
dok.	dokument
f.	funkcja
j.	język
kl.	klasa
kol.	kolejny
odp.	odpowiedź
org.	organizacja
ost.	ostatni
p.	punkt
pow.	powyższy
roz.	rozdział
s.	strona
spr.	sprawdzenie
sys. op.	system operacyjny
tab.	tablica
testy aut.	testy automatyczne
tyg.	tygodnia
wer.	wersja
wew.	wewnętrzna
wył.	wyłącznie
zad.	zadanie
zaw.	zawartość

Skrótów 4literowych i dłuższych nie tłumaczę, bo uważam je za oczywiste.

2 Rozw. Krok 1: Podstawa Wysokiej Jakości Prog. To Dok. Tech. i Testy Aut.

Moim zdaniem nie ma "cudownego sposobu" na eliminację błędów w kodach prog. pisanych przez ludzi. Można za to wskazać sposoby ich wykrywania i poprawiania.

Najważniejszymi narzędziami programisty pomocnymi w prog. wysokiej jakości kodu są: specyfikacja techniczna i testy automatyczne. Moim zdaniem wszelkie problemy z atakami typu "buffer-overflow" wynikają albo z nieprecyzyjnej specyfikacji technicznej, albo ze zbyt słabych testów aut. I nie ma to związku z żadnym j. prog. Po prostu

niedbałe prog. bez ich dopracowania i tak będą niedbałe i dziurawe. Dlatego może się okazać konieczne wydłużenie procesów projektowania i testowania oprogramowania, tak by nadać mu wysoką jakość. Na pewno należy dopracowywać wszelkie formaty i protokoły stosowane w sieci Internet.

3 Rozw. Krok 2: Zmiana Org. Pracy Nad Oprogramowaniem

Potrzebna jest metodyczne podejście do testowania. W tym celu trzeba dynamicznie przydzielać zadania mające na celu pełne przetestowanie kodu proj. Dlatego proponuję nowe rozw. org., cytat z mojej [arch-prog-nieup], roz. „Sposób wykrywania błędów”:

„5. Podczas testów jednostkowych:

W ramach testów aut. testujemy: 1) Wart. typowe, 2) Skrajne i 3) Zabronione w dok. technicznej.

W czwartek po południu tester aut. podaje zad. testowe do realizacji w pią. Co pią. wszyscy programiści uzupełniają testy aut. i wzmacniają spr. niezmienników i testy kontraktowe. Tak samo w ost. tyg. (ost. 5 d. roboczych) każdego mieś.”

4 Rozw. Krok 3: Zmiany w Kodowaniu: Konieczne Jest Spr. Pram. f. operator[] (long long i)

Obecna polityka polegająca na nie sprawdzaniu param. f. std::vector::operaotr[](long long i) to po prostu skrajna nie odpowiedzialność. Jeśli chodzi o spadek wydajności prog., to on i ma on miejsce również w innych sytuacjach: np. Position-independent code [pic], albo przy f. wirt., albo przy samym dostępie do kl. kontenerowych za pomocą f. iteratorów, a nawet same f. operator stanowią narzut w porównaniu do dostępu do prostych tab. (trzeba jednak zaznaczyć, że w przypadku iteratorów i f. operator mogą to być też szybkie f. wstawiane). Rzucanie wyjątku gdy indeks Jest większy od wielk. Wektora jest najbardziej naturalne. To powinno być standardowo włączone w kl. std::vector.

5 Rozw. Krok 4: Fuzzery Sterowane SI Do Testów Aut.

Fuzzery je rozumiem jako prog. Które dostają prawidłowy pakiet danych wej. i próbują znaleźć dziurę stopniowo psując ten pakiet wej. Mi się wydaje, że można je znacznie zoptymalizować sterując inteligentnie. Okazuje się, że istnieje już taki fuzzer: „American Fuzzy Lop” dostępny https://github.com/AFLplusplus/AFLplusplus. Ja go jeszcze nie używałem, ale wydaje się to b. dobrym p. wyjścia do inteligentnych testów aut.

6 Rozw. Krok 5: Zmiany w Procesorach

Dostęp do wszystkich tab. powinien się dokonywać za pomocą nowej instrukcji asemblerowej z 2 parametrami: 1) pocz. tab. i 2) indeks w tab. Wtedy wykorzystując to, że przed każdą tab. jest już umieszczona stała z jej wielk. można sprzętowo spr. czy nie przekroczono rozmiaru tab. W takiej sytuacji powinien następować wyjątek procesora i przekazanie kontroli do procedury obsługi tego wyjątku. Działać to ma podobnie do PIC (adresy względne w programach) i wyjątków koprocesora (przy jego braku do jego programowej emulacji – tak było w przypadku procesorów Intel 386 bez opcjonalnego koprocesora matematycznego i w przypadku tanich procesorów Intel 486SX z wył. uszkodzonym koprocesorem matematycznym). Te wyjątki wynikające z braku koprocesora wykorzystywały wczesne wer. rdzenia sys. op. Linuks (w ten sposób działała programowa emulacja koprocesora matematycznego).

7 Nonsensowna Argumentacja w Art. [tworca-c++-prosi-o-pomoc]

cytat z art. [tworca-c++-prosi-o-pomoc]:

"Presja ze strony regulatorów
Nie bez znaczenia są także zmiany regulacyjne. Amerykańska CISA opublikowała w październiku ubiegłego roku raport zalecający, by do 1 stycznia 2026r. producenci mieli plan eliminacji luk pamięciowych lub przeszli na języki bezpieczne dla pamięci. Stroustrup uznał to za "realne zagrożenie" dla C++.
Rząd USA proponuje następujące języki:
Rust
Go
C#
Java
Swift
JavaScript
Ruby"

Pow argumentacja jest nonsensowna z tego powodu, że na tej liście tylko Rust i Go są j. prog. a reszta to j. skryptowe. Obie grupy j. są nieporównywalne i mają inne zastosowania.

8 Podsumowanie

Moja końcowa myśl jest taka, że wygląda na to że trafne są moje wcześniejsze przewidywania, które opublikowa-

łem w [f-35-zasady-kodowania] i że faktycznie rząd SZAP/USONA chce wycofać z rynku cywilnego j. C++ i że chcą zrobić z niego kol. "czarny proj.". Bo na przekór pow. propagandzie SZAP/USONA uruchamia kol. proj. militarne oparte o prog. kodowane w j. C++ (np. patrz art. w Nowej Technice Wojskowej nr 2024/11, [ntw-2024-11] cytat s. 45: *„I tak zmiany wprowadzone w pierwszym z wariantów pocisku JASSM-ER [prawdopodobnie chodzi o AGM158 B2 – przyp. JMJ] dotyczyć mają przede wszystkim wymiany komponentów teraz trudnodostępnych, wychodzących z produkcji czy też po prostu przestarzałych (według obecnych standardów). Systemy komputerowe pocisku otrzymały ponadto oprogramowanie, stworzone w języku C++, które zastąpiło wcześniej stosowane, napisane w języku ADA. Pocisk w wariancie AGM-158B-2 wyposażono również w zmodernizowany komputer pokładowy (Missile Control Unit, MCU), dysponujący zwiększoną mocą obliczeniową.”*).

Znamy to z przeszłości: wcześniej tak było np. z sys. op. Plan9, który był stworzony przez twórców sys. op. Unix. Sys. op. Plan9 był genialny i przełomowy, jednak został porzucony. Co było kol. proj. twórców sys. op. UNIX? Pozostaje tajemnicą bez odp. Jednak ja się domyślam: następca sys. op. Plan9 należał do "czarnych proj." czyli tajnych i to zazwyczaj na zawsze.

Innym przykładem wycofywania rozw. z cywilnego rynku są dżojstiki używane przez domowych graczy komputerowych w latach 80 i 90 XXw. Teraz dżojstiki są używane wyłącznie w wojsku. Natomiast obecnie w domach do gier komputerowych używa się nieergonomicznych padów.

9 Bibliografia

Bibliografia

tworca-c++-prosi-o-pomoc: Paweł Czajkowski, Twórca C++ prosi o pomoc. Przyszłość języka zagrożona przez rząd USA i Rust, 2025, https://ithardware.pl/aktualnosci/tworca_c_jezyk_zagrozony_usa_rust-39456.html

arch-prog-nieup: Jacek Marcin Jaworski, Arch. Prog. Nieuprzywilejowanych, 2026, https://energokod.gda.pl/monografie/Arch.%20Prog.%20Nieuprzywilejowanych.pdf

pic: , Position-independent code, 2025, https://en.wikipedia.org/wiki/Position-independent_code

f-35-zasady-kodowania: Jacek Marcin Jaworski, F-35 - Zasady Kodowania - komentarz, 2026, https://energokod.gda.pl/komentarze-do-ksiazek/2025-01-19%20F-35%20-%20Zasady%20Kodownia%20-%20komentarz.pdf

ntw-2024-11: Michał Gajzler, JASSM niejedno ma imię, czyli AGM-158B-2, B-3, D oraz LRASM i JASSM-XR, 2024