

Kompilacja przez interpretację: metaprogramowanie w C₌₌-1

1st Adam Grabski

Wydział Elektroniki i Technik Informatycznych

Politechnika Warszawska

Warszawa, Polska

adam.grabski.stud@pw.edu.pl

Abstract—Statyczne metaprogramowanie to technika programistyczna umożliwiająca analizę, modyfikację lub generację kodu w czasie kompilacji. Współczesne, niskopoziomowe języki programowania wprowadzają wsparcie dla tych mechanizmów, w ograniczonym zakresie. W ramach tej pracy, zaproponowano nowy język programowania: C₌₌-1, zaprojektowany od początku z myślą o metaprogramowaniu, oraz nowe podejście do konstrukcji kompilatora: kompilacja przez interpretację.

Index Terms—Metaprogramowanie, język programowania, kompilator, kompilacja

I. WSTĘP

Dwa trendy w rozwoju współczesnych języków programowania niskiego poziomu to wykonanie kodu w czasie kompilacji oraz wsparcie dla statycznego metaprogramowania. Mechanizmy takie jak system makr w Rust oraz szablony w C++ umożliwiają generowanie kodu oraz jego analizę w pewnym ograniczonym zakresie. Dają one niewielki wgląd w strukturę programu oraz wykorzystują składnię odrębną od reszty języka, co utrudnia ich efektywne wykorzystanie.

Te języki, umożliwiają też wykonywanie kodu w czasie kompilacji, jednak tutaj programista także spotyka się ze znacznymi ograniczeniami. C++ wprowadził tę możliwość jako pierwszy, tworząc funkcje `constexpr` w C++11. Na początku dawały dostęp wyłącznie do najbardziej podstawowych elementów języka, jednak z czasem te ograniczenia były znoszone. W C++20, funkcje `constexpr` mają już tylko jedno, poważne ograniczenie: nie mogą zwracać dynamicznie alokowanej pamięci. Rust podąża śladami C++, wprowadzając funkcję `const`.

Celem C₌₌-1 jest umożliwienie wykonywania oraz dowolnego modyfikowania kodu w czasie kompilacji. Osiągnięto to poprzez stworzenie interpretera tego języka a następnie udostępnienie mu programu jako modyfikowalnej struktury danych. W ten sposób, dowolny fragment kodu może być wykonany w czasie kompilacji oraz użytkownik może dowolnie modyfikować swój program, używając normalnej składni.

II. STRUKTURA JĘZYKA

Język C₌₌-1 bazuje na C++, z elementami składni Rust. Program może składać się z przestrzeni nazw, funkcji oraz typów takich jak klasy, typy enumeracyjne oraz interfejsy. Tak jak C++, C₌₌-1 jest językiem programowania ogólnego przeznaczenia, ze wsparciem dla programowania obiektowego.

A. System typów

W przeciwieństwie do C++, C₌₌-1 nie pozwala na dziedziczenie po klasach. Zamiast tego, wprowadza koncepcję interfejsu z języków takich jak C#. Interfejs może zawierać wyłącznie nagłówki metod, które implementująca klasa musi zdefiniować. Klasy i interfejsy mogą dziedziczyć po dowolnej ilości interfejsów.

III. PROCES KOMPILACJI

A. Reprezentacja pośrednia

B. Interpreter

C. Kompilator

ACKNOWLEDGMENT

The preferred spelling of the word “acknowledgment” in America is without an “e” after the “g”. Avoid the stilted expression “one of us (R. B. G.) thanks ...”. Instead, try “R. B. G. thanks...”. Put sponsor acknowledgments in the unnumbered footnote on the first page.

REFERENCES

Please number citations consecutively within brackets [1]. The sentence punctuation follows the bracket [2]. Refer simply to the reference number, as in [3]—do not use “Ref. [3]” or “reference [3]” except at the beginning of a sentence: “Reference [3] was the first ...”

Number footnotes separately in superscripts. Place the actual footnote at the bottom of the column in which it was cited. Do not put footnotes in the abstract or reference list. Use letters for table footnotes.

Unless there are six authors or more give all authors’ names; do not use “et al.”. Papers that have not been published, even if they have been submitted for publication, should be cited as “unpublished” [4]. Papers that have been accepted for publication should be cited as “in press” [5]. Capitalize only the first word in a paper title, except for proper nouns and element symbols.

For papers published in translation journals, please give the English citation first, followed by the original foreign-language citation [6].

REFERENCES

- [1] G. Eason, B. Noble, and I. N. Sneddon, "On certain integrals of Lipschitz-Hankel type involving products of Bessel functions," *Phil. Trans. Roy. Soc. London*, vol. A247, pp. 529–551, April 1955.
- [2] J. Clerk Maxwell, *A Treatise on Electricity and Magnetism*, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.
- [3] I. S. Jacobs and C. P. Bean, "Fine particles, thin films and exchange anisotropy," in *Magnetism*, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.
- [4] K. Elissa, "Title of paper if known," unpublished.
- [5] R. Nicole, "Title of paper with only first word capitalized," *J. Name Stand. Abbrev.*, in press.
- [6] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, "Electron spectroscopy studies on magneto-optical media and plastic substrate interface," *IEEE Transl. J. Magn. Japan*, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetism Japan, p. 301, 1982].
- [7] M. Young, *The Technical Writer's Handbook*. Mill Valley, CA: University Science, 1989.

IEEE conference templates contain guidance text for composing and formatting conference papers. Please ensure that all template text is removed from your conference paper prior to submission to the conference. Failure to remove the template text from your paper may result in your paper not being published.