Artur Hermanowicz, Agnieszka Molga

Operatory i ich przeciążanie w aspekcie programowania przetwarzania danych wielowymiarowych = Operators and their Overloading in Terms of Programming Processing of Multi-Dimensional Data

Dydaktyka Informatyki 13, 159-164

2018

Dydaktyka Informatyki 13(2018) Uniwersytet Rzeszowski
ISSN 2083-3156; e-ISSN 2543-9847
DOI: 10.15584/di.2018.13.21
http://www.di.univ.rzeszow.pl

Artur HERMANOWICZ[1], **Agnieszka MOLGA**[2]

[1] *Dr, Uniwersytet Technologiczno-Humanistyczny w Radomiu, Wydział Informatyki i Matematyki, Katedra Informatyki, ul. Malczewskiego 22a, 26-600 Radom; e-mail: artur.hermanowicz@uthrad.pl*
[2] *Dr, Uniwersytet Technologiczno-Humanistyczny w Radomiu, Wydział Informatyki i Matematyki, Katedra Informatyki, ul. Malczewskiego 22a, 26-600 Radom; e-mail: agnieszka19216@wp.pl*

OPERATORY I ICH PRZECIĄŻANIE W ASPEKCIE PROGRAMOWANIA PRZETWARZANIA DANYCH WIELOWYMIAROWYCH

OPERATORS AND THEIR OVERLOADING IN TERMS OF PROGRAMMING PROCESSING OF MULTI-DIMENSIONAL DATA

Słowa kluczowe: operatory, przeciążanie operatorów, przetwarzanie danych wielowymiarowych, grafika 3D, sztuczna inteligencja.
Keywords: operators, operator overloading, processing of multi-dimensional data, 3D graphics, artificial intelligence.

Streszczenie

W pracy przedstawiono wybrane problemy dotyczące programowania przetwarzania danych wielowymiarowych. Szczególną uwagę zwrócono na zagadnienia związane z przeciążaniem operatorów w nowoczesnych obiektowych językach programowania.

Abstract

The paper presents selected problems regarding programming of multidimensional data processing. Particular attention was paid to issues related to overloading of operators in modern object-oriented programming languages.

Wstęp

Duży postęp technologiczny ludzkości, który można obserwować obecnie stwarza również nowe wyzwania. Łatwo zauważyć ogromny napływ nowych

informacji, co powoduje konieczność właściwego ich przetwarzania. Często dane te mają charakter wielowymiarowy.

Jednym z najważniejszych problemów przetwarzania dużej ilości wielowymiarowych danych jest wydajność. Rozwiązania, w których na wynik obliczeń trzeba czekać tygodniami, czy nawet godzinami rzadko są do przyjęcia. W pewnym stopniu problem wydajności jest redukowany rozwojem sprzętu. Komputery są coraz szybsze i problemy, które do niedawna leżały w strefie fantazji mogą być już rozwiązane. Niestety, zastosowanie nawet najlepszego komputera nie da tyle, co odpowiednio zoptymalizowany algorytm obliczeń.

Dziedziną, która szczególnie wymaga dużej wydajności obliczeń, a którą łatwo przeoczyć w aspekcie przetwarzania danych wielowymiarowych jest grafika komputerowa. Rynek gier komputerowych rozwija się błyskawicznie, a posiadanie przez nie grafiki trójwymiarowej jest już właściwie standardem. Warto zauważyć, że w praktyce programowania grafiki komputerowej wykorzystywane są współrzędne jednorodne, co daje już cztery wymiary. Do wygenerowania sceny wykorzystywane są obiekty składające się z tysięcy wierzchołków, a użytkownik oczekuje płynnej animacji w jak najwyższej rozdzielczości.

Bardzo duży rozwój można zauważyć również w zakresie zastosowań sztucznej inteligencji. Zagadnienia związane z grupowaniem danych, algorytmami genetycznymi, czy też zastosowaniami sztucznych sieci neuronowych mają bardzo duże zapotrzebowanie na wydajność. Zastosowania związane z technikami biometrycznymi, jak rozpoznawanie głosu czy twarzy, wymagają przetwarzania bardzo dużej liczby wielowymiarowych wektorów. Powoduje to duże zapotrzebowanie na moc obliczeniową. Uwierzytelniając swą tożsamość podczas np. rozmowy telefonicznej z bankiem użytkownik oczekiwał jednak będzie wykonania tej operacji w czasie rzeczywistym.

Programista tworzący aplikację służącą do przetwarzania wielowymiarowych danych staje przed wyborem w narzędzia, przy pomocy którego swe zadanie wykona. Istnieje wiele bardzo dobrych środowisk i bibliotek, w których zaimplementowano sprawdzone i zoptymalizowane metody. Bardzo dobrym środowiskiem, wykorzystywanym często do naukowych eksperymentów jest MATLAB[1]. Innym rozwiązaniem jest wybór jednego z nowoczesnych języków programowania. Ze względu na zalety programowania obiektowego dobrym wyborem wydają się być języki programowania, tj. C++[2], C#[3] czy Java[4].

Duży wzrost zapotrzebowania na oprogramowanie służące do przetwarzania danych wielowymiarowych implikuje również wzrastającą potrzebę przygoto-

[1] https://www.mathworks.com.
[2] B. Stroustrup, *Język C++*, Warszawa 1995.
[3] A. Troelsen, Ph. Japikse, *Język C# 6.0 i platforma .NET 4.6*, Warszawa 2017.
[4] C. Horstmann, G. Cornell, *JAVA. Podstawy*, wyd. 9, Gliwice 2014.

wania nowego pokolenia informatyków, a w szczególności programistów, do sprostania temu zadaniu. Proces dydaktyczny powinien zostać uzupełniony o wiedzę i umiejętności pozwalające przyszłemu autorowi oprogramowania świadomie i w pełni wykorzystać dostępne narzędzia. Włączenie poruszanych w dalszej części aspektów programowania do procesu dydaktycznego powinno zaowocować w przyszłości wymiernymi korzyściami w postaci bardziej wydajnego oprogramowania.

Operatory i ich przeciążanie

Jakkolwiek skomplikowany algorytm przetwarzania danych by nie był ostateczne operacje sprowadzają się do pewnych prostych obliczeń, zwykle z wykorzystaniem operatorów. Stosując nawet predefiniowane w danym języku operatory ciężko ustrzec się błędów. Często jest tego przyczyną zaniedbanie programisty i brak znajomości specyfikacji danego języka. Typowym przykładem jest tu operator złożonego przypisania (ang. *compound assignment operator*). Operator ten jest głównie postrzegany jako wygoda dla programisty, jednak ma on znacznie większe znaczenie. Bazując na przykładzie operatora dodawania, zamiast instrukcji:

X=X+Y

można napisać krócej:

X+=Y,

co w obydwu przykładach oznacza zwiększenie wartości zmiennej X o wartość zmiennej Y. Istotną różnicą jest jednak to, że w drugim przypadku adres zmiennej X jest wyznaczany tylko raz, co powoduje wzrost wydajności. Niestety, rzadko pamięta się o tym, że kolejną różnicą jest obliczenie wartości po prawej stronie przed wykonaniem właściwego operatora w przypadku złożonego przypisania, co stanowi częstą przyczynę błędów w przypadku obliczeń na liczbach całkowitych.

Ważnym aspektem tworzenia programu jest jego czytelność. Ma to tym większe znaczenie im większy projekt powstaje, a staje się krytycznym czynnikiem przy pracy zespołowej. Znakomitą pomocą może tu być zastosowanie tzw. przeciążania operatorów (ang. *operator overloading*). Możliwość definiowania nowych znaczeń dla operatorów stwarza możliwość uczynienia kodu bardziej przejrzystym i zwartym, niesie jednak ryzyko popełnienia błędów. Nie jest nowa technika, albowiem dostępna była już w języku ALGOL. Zmiany w obecnych językach programowania zmuszają do przyjrzenia się tej sprawie uważniej.

Najprościej sytuacja z przeciążaniem prezentuje się w języku Java. Twórcy tego narzędzia zrezygnowali z tej opcji jako potencjalnie niebezpiecznej ze względu na możliwość popełniania błędów. Istotna różnica jest pomiędzy językami C++ i C#. W języku C++ programista ma możliwość oddzielnego zdefi-

niowania operatorów zarówno podstawowych, jak i złożonego przypisania. W języku C# możliwe jest zdefiniowanie jedynie operatorów podstawowych. Operator złożonego przypisania tworzony jest automatycznie na podstawie zdefiniowanego przez programistę odpowiednika podstawowego[5].

W celu przeanalizowania wspomnianych różnic zdefiniowane zostały przykładowe operatory: dodawania w C++ (rys. 1), złożonego przypisania w C++ (rys. 2) oraz dodawania w C# (rys. 3). Założono, że klasa dla której definiowane są operatory ma identyfikator Klasa oraz pozostawiono tylko najistotniejsze elementy kodu.

```
Klasa operator+(const Klasa& a, const Klasa& b)
   {
     //konieczne obliczenia
     return Klasa(/* parametry */);
   }
```

Rys. 1. Schemat przeciążania operatora + w języku C++

Źródło: opracowanie własne.

```
Klasa& operator+=(const Klasa& a)
   {
     //konieczne obliczenia
     return *this;
   }
```

Rys. 2. Schemat przeciążania operatora += w języku C++

Źródło: opracowanie własne.

```
Klasa operator+(Klasa a, Klasa b)
   {
     //konieczne obliczenia
     return new Klasa(/* parametry */)
   }
```

Rys. 3. Schemat przeciążania operatora + w języku C#

Źródło: opracowanie własne.

Problem wydajności

Problem wydajności obliczeń dla pojedynczych wyrażeń teoretycznie można uznać za nieistotny. Zupełnie inaczej sytuacja ta prezentuje się, gdy obliczenia te wykonywane są wielokrotnie dla dużych zbiorów wielowymiarowych danych.

[5] C# Language Specification Version 5.0, Microsoft Corporation, 2013.

Jak już wspomniano, samo zastosowanie operatora złożonego przypisania może prowadzić do wzrostu wydajności programu. Jest to jednak prawdą jedynie dla predefiniowanych operatorów dla typów prostych. W przypadku złożonych typów definiowanych przez programistę oczywiście istotne są jego umiejętności. Te jednak również nie pomogą ze względu na specyfikę różnych języków programowania.

Należy mieć na względzie, że każda dodatkowa operacja powoduje spadek wydajności. Stąd dla przykładu samo przeciążenie operatora dodawania dla dwu wektorów zamiast wykonania tej operacji składowa po składowej potencjalnie obniża wydajność programu. Nie musi się tak stać dzięki bardzo zaawansowanym metodom optymalizacji wbudowanym w obecne kompilatory. Jest to jednak ewentualna cena za zwiększenie czytelności kodu.

Widoczna jest jednak różnica ze względu na specyfikację odpowiednich języków programowania. Pozornie różnica pomiędzy definicją tego samego operatora w języku C++ (rys. 1) i C# (rys. 3) jest niewielka. Jednakże w pierwszym przypadku wynikiem działania będzie jedynie powstanie chwilowego obiektu statycznego, gdy w drugim przypadku jest to już utworzenie nowego dynamicznego obiektu, co wiąże się z operacją przydziału pamięci i niestety wpływa na mniejszą wydajność.

Znacznie gorzej dla języka C# sytuacja prezentuje się w przypadku operatora złożonego przypisania. W języku C++ istnieje możliwość zdefiniowania tego operatora (rys. 2) niezależnie od operatora podstawowego (rys. 1). W efekcie możliwe jest takie zdefiniowanie operatora, że podczas jego działania nie powstanie żaden nowy obiekt. Jest to zgodne ze specyfikacją języka i ideą przyświecającą powstaniu operatora złożonego przypisania. Żaden nowy byt nie musi powstać w takiej operacji, modyfikowana jest jedynie zawartość istniejącego obiektu. W języku C# natomiast programista otrzymuje automatycznie odpowiedni operator złożonego przypisania o ile zdefiniuje operator podstawowy. Jest to oczywista oszczędność czasu pracy. Ceną za nią jest jednak wydajność programu. Automatycznie utworzony operator wciąż tworzy nowy obiekt, co w przypadku języka C++ nie ma miejsca. Działa to na niekorzyść programu napisanego w języku C# i obniża jego wydajność względem analogicznego programu napisanego w języku C++. Jak wykazano[6], w przypadku operatora podstawowego może to prowadzić nawet do dwukrotnego spadku wydajności, a w przypadku złożonego operatora przypisania prawie trzykrotnego.

[6] A. Hermanowicz, *Techniki programowania i implementacja aplikacji sztucznej inteligencji* [w:] *Informatyka w dobie XXI wieku. Technologie informatyczne i ich zastosowania w nauce, technice i edukacji*, red. A. Jastriebow, M. Raczyńska, J. Wołoszyn, Radom 2013.

Zakończenie

Intensywny rozwój technologiczny cywilizacji niesie ze sobą wzrastający gwałtownie wzrost produkowanych danych i konieczności ich przetwarzania. Można również zaobserwować wciąż postępujący rozwój zaplecza sprzętowego, komputery mają coraz większą wydajność i więcej pamięci. Postęp w dziedzinie sprzętu nie jest jednak w stanie dorównać narastającemu zapotrzebowaniu na przetwarzania coraz obszerniejszych zbiorów danych począwszy od rozrywki – gier komputerowych, a kończąc na zaawansowanych projektach naukowych z dziedziny sztucznej inteligencji.

Rozwija się również zaplecze umożliwiające tworzenie narzędzi umożliwiających radzenie sobie z tym problemem. Zaobserwować można różne ścieżki tego rozwoju. W języku C++ dostarczono narzędzie w postaci możliwości przeciążania operatorów, co umożliwia tworzenie bardziej zwartego i przejrzystego kodu. W nowszych językach powstałych zresztą na jego bazie przyjęto nieco odmienne rozwiązania. W języku Java całkowicie zrezygnowano z tej opcji, a w języku C# przyjęto rozwiązanie pośrednie, co niestety może wprowadzić w pułapkę mniej doświadczonych programistów.

Zdaniem autorów niniejszej pracy przedyskutowane w niej zagadnienia dotyczące różnic w poszczególnych narzędziach programistycznych powinny stanowić uzupełnienie w procesie dydaktycznym przyszłego pokolenia informatyków w ramach zajęć dotyczących programowania. Świadome wykorzystanie przez nich narzędzi programistycznych zaowocuje wydajniejszym oprogramowaniem, które będzie powstawać w przyszłości.

Bibliografia

Hermanowicz A., *Techniki programowania i implementacja aplikacji sztucznej inteligencji* [w:] *Informatyka w dobie XXI wieku. Technologie informatyczne i ich zastosowania w nauce, technice i edukacji*, red. A. Jastriebow, M. Raczyńska, J. Wołoszyn, Radom 2013.

Horstmann C., Cornell G., *JAVA. Podstawy*, wyd. 9, Helion, Gliwice 2014.

Stroustrup B., *Język C++*, WNT, Warszawa 1995.

Troelsen A., Japikse Ph., *Język C# 6.0 i platforma .NET 4.6*, PWN, Warszawa 2017.

Netografia

C# Language Specification Version 5.0, Microsoft Corporation, 2013.

https://www.mathworks.com.