

Podstawy programowania w Nim

Namysław Tatarynowicz

n.tatarynowicz@gmail.com

31 maja 2023

Spis treści

1	Wprowadzenie do języka Nim	2
1.1	Słowo wstępu	2
1.2	Instalacja środowiska	2
1.3	Kompilacja	2
1.4	Pierwszy program	3
2	Zmienne i typy danych	6
2.1	Zmienne	6
2.2	Typy danych	6
3	Instrukcje warunkowe	7
4	Pętle	8
4.1	Pętla for	8
4.2	Pętla while	8
4.3	Instrukcja break	8
4.4	Instrukcja continue	8

1 Wprowadzenie do języka Nim

1.1 Słowo wstępu

Nim jest językiem programowania o składni zbliżonej do składni języka Python, ale kompilowanym. Dzięki temu umożliwia tworzenie programów o podobnej wydajności jak programy napisane w języku C. Nim do kompilacji kodu wykorzystuje jedną z dostępnych bibliotek języka C. Zatem pisząc program, Nim tworzy najpierw plik C, który następnie jest kompilowany.

Najnowszą wersję Nima znajdziemy na stronie <https://nim-lang.org/>. W momencie tworzenia tego dokumentu, na stronie twórców, Nim jest dostępny w wersji 1.6.12.

Znajduje się tam również środowisko, dzięki któremu będziemy mogli skompilować i uruchomić nasz kod on-line oraz wygenerować odnośnik i go udostępnić: <https://play.nim-lang.org/>.

Opracowując ten dokument, będę korzystał z następującego oprogramowania:

```
system operacyjny: antiX 22
język              : Nim 1.6.12
kompilator         : glibc 2.28
edytor kodu        : Geany 1.33
```

1.2 Instalacja środowiska

Instalacja...

1.3 Kompilacja

Najszybszym sposobem skompilowania przygotowanego pliku jest wykonanie polecenia:

```
nim c nazwa_pliku.nim
```

Po poprawnym skompilowaniu otrzymamy wykonywalny plik o takiej samej nazwie.

Ogólna składnia kompilacji programu jest następująca:

```
nim polecenie [opcje] [nazwa_pliku] [argumenty]
```

polecenie może przyjmować następujące wartości:

compile, c	kompiluje projekt z wykorzystaniem domyślnego generatora kodu C
r	
doc	generuje dokumentację dla pliku wejściowego

opcje mogą przyjmować następujące wartości:

-p, --path:PATH	dodaj ścieżkę do ścieżek wyszukiwania
-d, --define:SYMBOL(:VAL)	define a conditional symbol (Optionally: Define the value for that symbol, see: "compile time define pragmas")
-u, --undef:SYMBOL	undefine a conditional symbol
-f, --forceBuild:on off	wymuś przebudowę wszystkich modułów
--stackTrace:on off	włącz/wyłącz ślad stosu
--threads:on off	włącz/wyłącz obsługę wielowątkowości
-x, --checks:on off	włącz/wyłącz wszystkie kontrole środowiska uruchomieniowego
-a, --assertions:on off	włącz/wyłącz asercje
--opt:none speed size	nie optymalizuj, optymalizuj pod względem szybkości lub czasu
--debugger:native	użyj natywnego debugera (gdb)
--app:console gui lib staticlib	wygeneruj aplikację konsolową, graficzną, DLL lub bibliotekę statyczną
-r, --run	uruchom skompilowany program z podanymi argumentami
--eval:cmd	evaluate nim code directly; e.g.: nim -eval:"echo 1" defaults to e (nimscrip) but customizable: nim r -eval:'for a in stdin.lines: echo a'
--fullhelp	pokaż pełny podręcznik pomocy
-h, --help	pokaż podręcznik pomocy
-v, --version	pokaż szczegóły dot. zainstalowanej wersji

1.4 Pierwszy program

Do wyświetlania tekstu na ekranie służy polecenie `echo`. Konsekwencją jego działania, poza wyświetlaniem tekstu, jest przejście kursora do nowej linii.

```
echo "Hello world"
```

```
Hello world
```

```
-
```

Istnieją różne sposoby wyświetlania tekstu w konsoli.

```
echo ":)"
"Hej!".echo
"Hej! Hej!".echo()
"Hej! ".echo "Ho!"
echo("Hello world")
("Hello world!").echo
```

```
:)
Hej!
Hej! Hej!
Hej! Ho!
Hello world
Hello world!
-
```

Jeśli chcemy, aby po wyświetleniu tekstu kursor pozostał w tej samej linii, skorzystamy z `stdout.write`. Możemy i tu wymusić przejście do nowej linii używając `\n`

```
stdout.write "Hello world. "
stdout.write "Bye!\n"
```

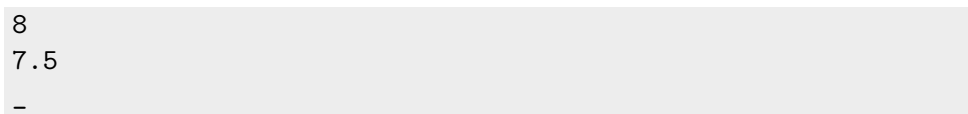
```
Hello world. Bye!
```

```
-
```

Można oczywiście wyświetlać liczby lub wyniki operacji arytmetycznych. Liczby zmiennoprzecinkowe mają oddzielną część całkowitą od dziesiętnej kropką.

```
echo 20
echo 3.14
echo 5+5
echo 6-2
echo 2*4
echo 15/2
```

```
20
3.14
10
4
```



2 Zmienne i typy danych

2.1 Zmienne

2.2 Typy danych

3 Instrukcje warunkowe

4 Pętle

4.1 Pętla for

4.2 Pętla while

4.3 Instrukcja break

4.4 Instrukcja continue