Politechnika Poznańska, Poznań, 27 czerwca 2022
Artykuł przeglądowy

Ezoteryczne języki programowania
niezwykłe twory zwykłych programistów

Mateusz Szuda[1]

Streszczenie
Istnieje pewna grupa języków programowania zwana językami ezoterycznymi (ang. ***esolang***). *Są one tworzone w celu badania i demonstracji niekonwencjonalnych technik programistycznych oraz metod programowania. Nie są one natomiast przeznaczone do pisania rzeczywistych aplikacji.*[1] Ich twórcy skupiają się na dążeniu do stworzenia języka *dziwnego* - **Malbolge**, *minimalistycznego* - **Brainfuck**, *tematycznego* - **Chef**, czy stworzonego dla *zwykłego żartu* - **Ook!**. Tylko kilka z ogromnej, stale rosnącej bazy języków zostało wymienione, a w dalszej części dokumentu zostaną poruszone kwestie związane z ich genezą, zostaną opisane wymienione i inne języki oraz zgłębiony zostanie temat celów, jakimi kierowali się programiści podczas ich tworzenia. Dodatkowo zostanie przedstawiona semantyka *esolangów* okraszona przykładowymi programami napisanymi za ich pomocą.

Słowa kluczowe
ezoteryczne — języki — programowania — esolangs

[1] *student 3 roku Teleinformatyki, nr indeksu 144379, Politechnika Poznańska*

Spis treści

Wstęp

Opisując pojęcie *języka programowania*, większość osób wyobraża sobie je jako narzędzie, zbiór zasad określających dane działania. Kod taki jest następnie kompilowany *(np. C, C++)*, bądź interpretowany *(np. Python)* przez komputer, który wykonuje określone instrukcje zawarte w kodzie. Jednymi z najbardziej popularnych języków pozwalających nam na takie działania są: Python, Java, C, C++, JavaScript, C#, R, Go [3][1]. Są to języki proste w swoim użyciu, zawierające bogatą i szeroko opisaną składnię językową.
Nie każdy język natomiast nadaje się do przetwarzania pewnych problemów, z którymi mierzą się programiści. Program napisany w jednym języku może się wykonywać szybciej niż w drugim. To samo tyczy się objętości samych programów, ich kompilatorów, czy samego kodu źródłowego. Dodatkowo należy mieć też na uwadze fakt, iż jedne języki programowania są bardziej odpowiednie do rozwiązywania określonych zadań, konkretnych problemów, w porównaniu do innych tworów. Dobrym przykładem byłoby tutaj przywołanie grupy języków R, SQL i Matlab w opozycji do C, C++ i Javy. Pierwsza grupa to języki przeznaczenia specjalistycznego, wykorzystywane do wszelakich obliczeń statystycznych, czy algorytmicznych, reprezentacji graficznej wyników, czy wykonywania operacji bazodanowych. Druga grupa są to języki ogólnego przeznaczenia, które pozwalają na stworzenie wysokowydajnych aplikacji, istniejące od wielu lat. Ich niewątpliwą zaletą jest również fakt istnienia ogromnej bazy dokumentacji oraz aplikacji napisanych za pomocą tych języków. *W rzeczy samej, nawet znacząca część języka Python, jak i jego dokumentacji jest napisana w języku C z powodów wydaj-*

[1] wybór 8 najbardziej popularnych języków programowania w 2021 roku według serwisu IEEE spectrum: `https://spectrum.ieee.org/top-programming-languages-2021`

nościowych [3][2].

1. Czas na języki ezoteryczne

1.1 Pojęcie języka ezoterycznego

W przeciwieństwie do języków programowania tworzonych w celu produktywnego i efektywnego projektowania i budowania aplikacji, **języki ezoteryczne** tworzone są w innych celach: Mogą one reprezentować pewne idee **proof-of-concept**, demonstrować **minimalizację** składni języka, który będzie nadal używalny, zachowując jednocześnie **uniwersalność**. Mogą one pomóc w udowodnieniu pewnych konceptów matematycznych, czy ustaleniu granic analiz złożoności. Proces projektowania języków ezoterycznych można zdefiniować jako operacja artystyczna, będąca wynikiem, a wręcz ekspresją ludzkiego intelektu, dowcipu, czy zmysłu estetyki. Mogą być również stworzone w celu podjęcia się swego rodzaju rywalizacji między programistami, bądź wyzwania, czy to dla siebie, czy dla hipotetycznego użytkownika. Istnieje jeszcze kolejna grupa języków ezoterycznych - języki prześmiewcze, napisanych ku uciesze autorów, użytkowników, czy choćby czytelników ich specyfikacji[4]. Wszystkie wymienione właściwości,
jak i charakterystyki języków ezoterycznych wywołują chęć przyjrzenia się im dokładniej oraz poznania zamiarów ekscentrycznych programistów stojących za ich powstaniem.

1.2 Geneza powstania języków ezoterycznych

Za najstarszy znany język ezoteryczny uważa się ***INTERCAL*** stworzony w **1972** roku przez Donalda R. Woodsa oraz Jamesa M. Lyona.

1.2.1 Prehistoria języków ezoterycznych

W historii informatyki można natomiast doszukać się informacji o językach, które miały wspólną genezę ze współczesnymi językami ezoterycznymi. Co natomiast odróżniało je od nowoczesnych post-INTERCAL'owych języków ezoterycznych, tworzone one były w celach profesjonalnych i całkowicie użytkowych. Przypatrzmy się przykładowo językowi ***EXPLOR***:

> Jego główna idea polegała na wykonywaniu się kodu czysto probabilistycznie - każda poszczególna instrukcja ma pewne *prawdopodobieństwo*, które definiuje częstość jej wykonania. W dodatku podczas pisania kodu za jego pomocą nie ma możliwości użycia *zmiennych* - każda instrukcja bazuje na aktualnej wartości stanu wyświetlacza, bądź na innych instrukcjach.
>
> (!) Ciekawostka: języka tego używało wielu artystów, by móc generować obrazy używane do animacji swoich ekspresji, jak i również był nawet używany przez pewien czas jako narzędzie do nauki GRAFIKI KOMPUTEROWEJ zarówno artystów, jak i dzieci.

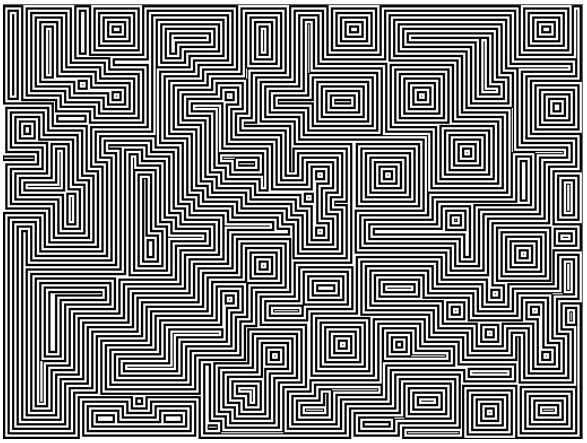
Przykładowy program napisany w języku EXPLOR wygląda tak:

Listing 1: EXPLOR code

```
1    dim     (1,1)     (512,384)
2    wbt     (1,1)     (abc,012,)
3    bxl     (1,1)
         ↪ 40(20,20,45,45,10,10,50,38)
         ↪ 1(a.)
4  ax axl    (1,1)     4,news,01,1(.01)
5    axl     (1,1)     4,news,12,1(.12)
6    axl     (1,1)     4,news,ab,1(.ab)
7    axl     (1,1)     4,news,bc,1(.bc)
8    xl      (1,1)     1(.012abac0)
9    camera  (1,1)     1
10 cm goto   (x,29,1)  ax
```

Według reguł języka, składnia instrukcji wygląda następująco:

INSTRUKCJA PRAWDOPODOBIEŃSTWO ARGUMENTY



Rysunek 1. Obraz wygenerowany za pomocą przykładowego kodu zaprezentowanego na **Listingu 1**

1.2.2 Prehistoria języków ezoterycznych - cellular automata

Do języków ezoterycznych zaliczany jest również dział tzw. ***automatów komórkowych***. Mogą być one uznawane za swego rodzaju języki ezoteryczne, a przynajmniej w sensie ogólnym, jakim określamy całą tematyczną grupę.
Wiele konstruktów określanych jako *automaty komórkowe* jest jednocześnie nazywanymi ezoterycznymi. Oznacza to nic innego jak fakt, iż mogą one (w teorii) rozwiązyć wszelakie (możliwe do rozwiązania) operacje i problemy matematyczne, między innymi *wyliczanie liczby pi* i tak dalej. W związku z tym mogą być one uznawane za **zupełne w sensie Turinga**.

> ↪ Dowolny zestaw reguł i obiektów, który może być użyty do symulacji ***Uniwersalnej Maszyny Turinga*** jest uważany za *kompletny w sensie Turinga*. Wszystkie modele zupełne w sensie Turinga są sobie tożsame, w takim sensie, że korzystają one z tego samego zestawu funkcji do wyliczeń.[5]

Dzięki takiemu założeniu możemy przyjąć, że automaty: stworzony na początku lat 1950 - opublikowany w **1966** roku "29-cio stanowy automat"Von Neumanna oraz opubli-

[2] „Indeed, significant parts of Python itself and its libraries are written in C for performance reasons." https://spectrum.ieee.org/top-programming-languages-2021

kowana w **1970** roku "Gra w życie"Johna Conwaya są jednocześnie jednymi z pierwszych ***prehistorycznych języków ezoterycznych***!

2. Rodzaje języków ezoterycznych [2]

Istnieje wiele powodów przemawiających za stworzeniem języka ezoterycznego. Wiadomym już jest, że najważniejszą ich właściwością, odróżniającą je od zwykłych języków, jest fakt, iż nie są one przeznaczone do poważnego, profesjonalnego użytkowania i funckjonowania. Możemy natomiast wyróżnić kilka głównych przyczyn, którymi kierują się autorzy w trakcie ich konstruowania. Wiele języków zostanie tutaj wymienionych, a te podkreślone zostaną w późniejszej części pracy dogłębniej wytłumaczone:

- **Minimalizm** - popularnym wyborem podczas projektowania danego języka jest stworzenie zestawu jak najmniejszej liczby instrukcji - uproszczone w pewnych aspektach do granic możliwości. Języki tego rodzaju, jeśli są *Turing-zupełne*, określane są mianem *grzęzawiska Turinga*: "Strzeżcie się grzęzawiska Turinga, w którym wszystko jest możliwe, ale nic nie jest łatwe"[6] Do minimalistycznych języków zaliczyć można **brainfuck**, **OISC** oraz **Lazy K**.
- **Nowe koncepty** - wśród języków ezoterycznych bardzo popularnym podejściem w trakcie ich projektowania jest wynajdowanie nowych, alternatywnych sposobów na reprezentację kodu. Dobrymi przykładami tego rodzaju języków są **Befunge**, **Thue** oraz **Unlambda**.
- **Dziwność** - niektóre języki tworzone są po to, by uzyskać twór dziwny i trudny do pisania w nim kodu. Sztandarowym przykładem byłby tutaj język, uznawany za pionerski wśród języków ezoterycznych - **INTERCAL** - stworzony głównie z powodu, by być tak inny jak to możliwe od "normalnych"języków programowania, mimo wielu podobieństw między nim, a konwencjonalnymi językami. Drugim bardzo interesującym przypadkiem jest **Malbolge**, który został stworzony po to, by być niemal niemożliwym do użytkowania.
- **Tematyka** - są to języki bazujące na pewnym, wybranym przez twórcę temacie, niezwiązanym z informatyką. Mogą być to języki takie jak **ArnoldC** - bazujący na sławnych cytatach wypowiadanych przez Arnolda Schwarzeneggera; **Shakespeare** - pisany w formie utworów scenicznych tytułowego angielskiego pisarza; oraz **Chef**, którego kod wygląda jak przepisy kulinarne!
- **Zwięzłość** - wiele języków dąży do spełnenia pojęcia "im mniej kodu, tym lepiej". Tego rodzaju języki znane są jako *Golfing languages* i bardzo często używane są w konkurencji **code golf**, której celem jest rozwiązanie danych problemów programistycznych za pomocą jak najmniejszej liczby znaków/bajtów, jak to jest możliwe. Kategoria ta obfituje w takie języki jak **CJam**, **Pyth**, czy **GolfScript**.
- **Dowcip** - języki tworzone typowo dla swego rodzaju żartu. Tym niemniej, wiele z nich jest zupełnie funkcjonalnych w celach programistycznych. Są to między innymi **l33k**, **Ook!**, jak i również oba wywodzące się z *brainfuck'a* **Moanfuck** i **Fuckfuck**.
- **Nieczytelność** - zaprojektowane w taki sposób, by odczytywanie kodu było w jak największym stopniu utrudnione, w przeciwieństwie do bycia ciężkimi do pisania, czy zrozumienia. Doskonałym przykładem jest tutaj tytułowy **Unreadable**. Bardzo ciekawym przypadkiem jest również wpisujący się w te ramy **Whitespace**.

3. Właściwości wybranych języków

Spośród stale rosnącej grupy ezoterycznych języków programowania, skupimy się na kilku z nich: charakterystycznych w swoim rodzaju, humorystycznych i wysoce specyficznych w porównaniu do normalnych języków, używanych przez programistów na codzień.

3.1 brainfuck

Nieomyłkowo napisany z małej litery, sztandarowy i prawdopodobnie najbardziej rozpowszechniony język z klasy ezoterycznych. Stworzony w 1993 roku przez Urbana Müllera, Turing-kompletny język, podczas próby stworzenia możliwie najmniejszego kompilatora dla Amigi OS w wersji 2.0. Müllerowi udało się napisać 296-bajtowy kompilator, który wzorowany był na tym napisanym w języku *FALSE* zajmującym aż 1024 bajty! *"Intencją autora było stworzenie języka trudnego do czytania i zrozumienia, nawet dla osoby zaznajomionej z czytaniem tego typu programów."*[7] Całość operacji dostępnych w tym języku mieści się zaledwie w 8 komendach, każdej reprezentowanej przez dany znak przestankowy. Wszystkie pozostałe znaki są ignorowane, co można uznać za swego rodzaju komentarz do danego programu.

Komenda	Opis
>	Przesuń wskaźnik w prawo
<	Przesuń wskaźnik w lewo
+	Inkrementuj wartość komórki wskazywanej
-	Dekrementuj wartość komórki wskazywanej
.	Wyświetl znak o wartości z komórki
,	Pobierz znak i zapisz go do komórki
[	Skocz bezpośrednio za odpowiadający znak], o ile bieżąca komórka zawiera 0
]	Skocz do odpowiadającego znaku [

Tabela 1. Spis instrukcji języka **brainfuck**

Co czyni **brainfuck** interesującym, jest fakt, że ten *minimalny język* z tak *niecodzienną logiką instrukcji* może stać na równi z takimi językami jak **Python** i **C**, pomimo braku implementacji tak arbitralnych funkcji jak reprezentacja liczby 2, czy operacja mnożenia.

3.1.1 brainfuck - logika działania

Sam język skonstruowany jest z tablicy bajtowej o rozmiarze conajmniej 30000 bajtów, wszystkich zainicjowanych zerami. Wskaźnik jest przypisywany do pierwszego bajtu całego zbioru.

3.1.2 brainfuck - dodawanie

Listing 2: Dodawanie w brainfucku

```
1  ,>++++++[<--------->-]
2  ,[<+>-]<.
```

Podczas operacji arytmetycznych między obiema liczbami, należy pamięć, że ich reprezentacja binarna zapisana jest w formie standardu **ASCII**[8]. Oznacza to nic innego jak potrzebę zmniejszenia jednej z liczb o 48 - jest to wartość zapisu liczby *"0"* w standardzie ASCII. Program wykonuje tą operację na pierwszej pobranej liczbie. Następnie druga wartość pobrana jest dodawana do już zmniejszonej pierwszej liczby. Na koniec wynik wypisywany jest na wyjście.
Prześledźmy teraz po kolei kod zaprezentowany na **listingu 2**:

- **linia 1** - Najpierw pobieramy znak z wejścia (**,**) i zapisujemy go pod adresem aktualnie wskazywanej komórki (z uwagi na to, że jest to początek programu - będzie to komórka o adresie 0). Następnie przechodzimy do następnej komórki (**>**) i inkrementujemy jej wartość 6 razy (**+**) - będzie to mnożnik wykorzystywamy do zmniejszenia wartości zapisanej w komórce pierwszej. Otwieramy pętlę ([- pętla będzie sprawdzać wartość zapisaną w komórce wskaźnika - 6) i napotykamy znak przejścia do poprzedniej komórki (**<**). Teraz dekrementujemy wartość komórki 0 (**-**) 8-krotnie, aż przejdziemy do komórki prawej i raz dekrementujemy jej wartość (jest to jednocześnie komórka iteracyjna pętli). Pętla ta będzie wykonywać się, dopóki nie napotka znaku końca pętli (]) - sumarycznie będzie to 6 przejść pętli, w trakcie których liczba zapisana w komórce 0 zostanie zmniejszona o 8, co da wynikowo *wartość pomniejszenia o 48*.

- **linia 2** - Po operacjach z *linii 1* wskaźnik wskazuje na komórkę 1, do której przypisywana jest wartość z wejścia. Następnie w pętli program przechodzi do komórki 0, którą inkrementuje tyle razy, jaka była wartość liczby drugiej w standardzie ASCII. Po skończeniu wykonywania się pętli, wartość w komórce 0 jest podawana na wyjście programu - dodawanie zostało zakończone.

3.1.3 brainfuck - program Witaj PP!

Alternatywna wersja jednego z najbardziej popularnych programów na świecie - **Hello World!** Kod bez odpowiedniego komentarza jest bardzo trudny do zrozumienia:

Listing 3: Witaj PP! w brainfucku bez komentarza

```
1  ++++++++++[>++++++++>++++++++++
       ↪ >++++++++++>++++++++++
       ↪ >+++>+<<<<<<-]
2  >+++++++.>+++++.>++++++.>---.<<+.
3  >>>++.<<<<-------..>>>>+.>.
```

Jak widać, kod ten nawet dla doświadczonych czytelników *brainfuck'a* może sprawiać problemy w odczytaniu. Na **listingu 4** został on podzielony na fragmenty wraz z odpowiednim komentarzem ilustrującym dane działanie:

Listing 4: Program Witaj PP! w brainfucku

```
1  +++++ +++++   inicjalizacja licznika
       ↪  (adres #0) do 10
2  [             wykorzystujemy petle,
       ↪ by ustalic w 6 kolejnych
       ↪ komorkach wartosci
       ↪ 80/100/110/100/30/10
3  > +++++ +++       dodaj 8 do komorki
       ↪  #1
4  > +++++ +++++     dodaj 10 do
       ↪ komorki #2
5  > +++++ +++++ +   dodaj 11 do
       ↪ komorki #3
6  > +++++ +++++     dodaj 10 do
       ↪ komorki #4
7  > +++             dodaj 3 do komorki
       ↪  #5
8  > +               dodaj 1 do komorki
       ↪  #6
9  <<<<<<-          dekrementuj licznik
       ↪ (komorka #0)
10 ]
11 > +++++ ++ .  wyswietl 'W'
12 > +++++ .     wyswietl 'i'
13 > +++++ + .   wyswietl 't'
14 > --- .       wyswietl 'a'
15 << + .         wyswietl 'j'
16 >>> ++ .       wyswietl '␣'
17 <<<< ----- -- .   wyswietl 'P'
18 .              wyswietl 'P'
19 >>>> + .       wyswietl '!'
20 > .            wyswietl '\n'
```

Programiści brainfucka bardzo często rywalizują ze sobą w rozwiązywaniu pewnych problemów programistycznych w celu osiągnięcia jak najmniejszej wielkości tworzonych przez nich programów.

> Przykładowo program "Hello World!"może zostać jednocześnie zapisany używając 106 instrukcji (znaków). Natomiast aktualnie najmniejszy napisany w formie "Hello, World!"(warto zwrócić uwagę na dodatkowy **przecinek**) zawarty jest na zaledwie 72 bajtach. Można jeszcze doliczyć wariancję pisaną małymi literami "hello, world!", która pozwala na zapisanie kodu o wielkości 70 bajtów.[9]

3.2 Fuckfuck & Moanfuck - dwa obsceniczne klony brainfucka

Oba języki są swoistymi klonami brainfucka przedstawionego w poprzednim podpunkcie. Klony w tym znaczeniu są niczym innymi jak przedstawieniem, zmianą nazwy instrukcji z jednoznakowej reprezentacji znaków przestankowych na obsceniczne, sprośne słowa używane w języku angielskim:

- **Fuckfuck** - używa popularnych bluźnierstw;
- **Moanfuck** - używa wyrażeń używanych przez ludzi w trakcie stosunku płciowego;

3.2.1 Fuckfuck

Instrukcje nie są *case-sensitive*[3], ani na występowanie białych znaków. Do komentarzy można użyć stylu znanego z języka C: /* */. Każda instrukcja zbudowana jest z 4 liter i oprócz bezpośredniego przedstawiania instrukcji, pozwala na "cenzurowanie"środkowych liter w celu "wyczyszczenia"tekstu. I tak słowa-komendy odpowiadające instrukcjom w brainfucku:

Fuckfuck	brainfuck
fuck	>
shag	<
boob	+
tits	-
cock	.
knob	,
arse	[
butt	]

Tabela 2. Spis instrukcji języka **fuckfuck** i odpowiadające im instrukcje **brainfucka**

można zapisać w dowolnej formie, ale należy pamiętać, że **pierwsza** i **ostatnia** litera danego słowa muszą zgadzać się ze specyfikacją zamieszczoną w **Tabeli 2**. Przykładowo wyrażenie *"fuck"* może zostać zamienione na *"fork"*, *fkfk*, czy *f**k*, ale *"f***"* nie będzie już formą poprawną.[10]

Dodatkową instrukcją jaka pojawiła się w tym języku jest wykrzyknik (**!**), który sprawia, że komenda zapisana bezpośrednio przed nim jest powtarzana tyle razy, ile razy zostanie on użyty - pozwala to na zmniejszenie objętości kodu i brak potrzeby pisania tego samego słowa wielokrotnie. Tak na przykład zapis:

Listing 5: Fuckfuck - inkrementacja z !

```
boob!!!!!!!!!
```

oznacza, że wartość aktualnej komórki zostanie zainkrementowana 10-krotnie.

Program ***Witaj PP!*** natomiast wygląda następująco:

Listing 6: *Witaj PP!* w fuckfucku

```
boob!!!!!!!!! arse fuck boob!!!!!!!
fuck boob!!!!!!!!! fuck boob
    ↪ !!!!!!!!!!
fuck boob!!!!!!!!! fuck boob!! fuck
boob shag!!!!! tits butt fuck
boob!!!!!! cock fuck boob!!!! cock
fuck boob!!!!! cock fuck tits!! cock
shag! boob cock fuck!! boob! cock
shag!!! tits!!!!!! cock! fuck!!!
boob cock fuck cock
```

Jednocześnie zgodnie z możliwością cenzurowania słów, "odkażony"program będzie wyglądał tak:

Listing 7: *Witaj PP* ocenzurowane

```
barb!!!!!!!!! able folk barb!!!!!!!
folk barb!!!!!!!!! folk barb
    ↪ !!!!!!!!!!
folk barb!!!!!!!!! folk barb!! folk
barb sing!!!!! teas bait folk
barb!!!!!! cask folk barb!!!! cask
folk barb!!!!! cask folk teas!! cask
sing! barb cask folk!! barb! cask
sing!!! teas!!!!!! cask! folk!!!
barb cask folk cask
```

3.2.2 Moanfuck

Zaimplementowany przez użytkownika *Razetime*[4]. W skład tego języka wchodzą następujące instrukcje[11]:

Moanfuck	brainfuck
YES	>
FUCK	<
AH	+
OH	-
YEAH	.
MORE	,
AHH	[
OOH	]
BABY	# (breakpoint debugowania)

Tabela 3. Spis instrukcji języka **moanfuck** i odpowiadające im instrukcje **brainfucka**

Wszystkie komendy muszą być oddzielone spacjami w momencie parsowania programu. Jednocześnie, podobnie jak miało to miejsce w *fuckfucku*, komendy są **case-insensitive**. Program ***Witaj PP!*** wygląda następująco:

[3] *case-sensitive* - podatne na zmiany wielkości liter

[4] https://esolangs.org/wiki/User:Razetime

Listing 8: *Witaj PP!* w moanfucku

```
AH AH AH AH AH AH AH AH AH AH AHH
YES AH AH AH AH AH AH AH AH YES
AH AH AH AH AH AH AH AH AH AH YES
AH AH AH AH AH AH AH AH AH AH AH
YES AH AH AH AH AH AH AH AH AH AH
YES AH AH AH YES AH FUCK FUCK
FUCK FUCK FUCK FUCK OH OOH YES
AH AH AH AH AH AH AH YEAH YES
AH AH AH AH AH YEAH YES AH AH AH
AH AH AH YEAH YES OH OH OH YEAH
FUCK FUCK AH YES YES YES AH AH
YEAH FUCK FUCK FUCK FUCK OH OH
OH OH OH OH OH YEAH YEAH YES YES
YES YES AH YEAH YES
```

3.3 Ook! - jeszcze jeden bazujący na brainfucku

Ten język również bazuje na liście intrukcji z języka *brainfuck*, ale znaczącą różnicą jest zmiana znaczenia słów na bardziej przyjazne Orangutanom[12]:

Ook!	brainfuck
Ook. Ook?	>
Ook? Ook.	<
Ook. Ook.	+
Ook! Ook!	-
Ook! Ook.	.
Ook. Ook!	,
Ook! Ook?	[
Ook? Ook!	]

Tabela 4. Spis instrukcji języka **Ook!** i odpowiadające im instrukcje **brainfucka**

Program ***Hello, world!*** napisany w "orangutanim"języku prezentuje się tak:

```
Ook. Ook? Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook.
Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook! Ook? Ook? Ook.
Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook.
Ook. Ook. Ook. Ook. Ook. Ook? Ook! Ook! Ook? Ook! Ook? Ook.
Ook! Ook. Ook. Ook? Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook.
Ook. Ook. Ook. Ook. Ook. Ook. Ook! Ook? Ook? Ook. Ook. Ook.
Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook? Ook! Ook! Ook? Ook!
Ook? Ook. Ook. Ook. Ook! Ook. Ook. Ook. Ook. Ook. Ook. Ook.
Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook! Ook. Ook! Ook.
Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook! Ook. Ook. Ook? Ook. Ook?
Ook. Ook? Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook.
Ook. Ook. Ook. Ook. Ook. Ook. Ook! Ook? Ook? Ook. Ook. Ook.
Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook? Ook! Ook! Ook? Ook!
Ook? Ook. Ook! Ook. Ook. Ook? Ook. Ook? Ook. Ook? Ook. Ook.
Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook.
Ook. Ook. Ook. Ook. Ook. Ook. Ook! Ook? Ook? Ook. Ook. Ook.
Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook.
Ook. Ook. Ook. Ook. Ook. Ook? Ook! Ook! Ook? Ook! Ook? Ook.
Ook! Ook! Ook! Ook! Ook! Ook! Ook! Ook. Ook? Ook. Ook? Ook.
Ook? Ook. Ook? Ook. Ook! Ook. Ook. Ook. Ook. Ook. Ook. Ook.
Ook! Ook. Ook! Ook! Ook! Ook! Ook! Ook! Ook! Ook! Ook! Ook!
Ook! Ook! Ook! Ook. Ook! Ook! Ook! Ook! Ook! Ook! Ook! Ook!
Ook! Ook! Ook! Ook! Ook! Ook! Ook! Ook! Ook. Ook. Ook?
Ook. Ook? Ook. Ook. Ook! Ook.
```

Rysunek 2. ***Hello, World!*** w języku Ook!

3.4 Befunge - program w dwóch wymiarach

Befunge jest językiem dwuwymiarowym, stworzonym w 1993 roku przez Chrisa Presseya[5], którego celem było stworzenie języka tak trudnego do kompilowania jak to możliwe. Kod każdego programu jest rozłożony na dwuwymiarowej siatce instrukcji, a podczas wykonywania program może przemieszczać się w każdym kierunku tej dwuwymiarowej płaszczyzny. Prostokątna siatka złożona jest ze znaków zakodowanych w reprezentacji **ASCII** - każdego reprezentującego daną instrukcję. Wskaźnik instrukcji zaczyna zawsze z ustalonej lokalizacji - lewy górny narożnik pola instrukcji; i domyślnie przechodzi w ustalonym kierunku - w prawo; jeśli napotka on instrukcje, to są one wykonywane.

3.4.1 Etymologia nazwy Befunge

Nazwa *befunge* wywodzi się z błędu typograficznego, polegającego na niepoprawnym zapisaniu słowa *before* przez Curtisa Colemana o 4 nad ranem na jednym z systemów chatowych BBS[13]...

Zawiera w sobie słowo "Funge", oznaczające kategorię języków, w których programy reprezentowane są w przestrzeniach metrycznych z układem współrzędnych używanym do położenia instrukcji - czyli dokładnie tak, jak dzieje się to w tym języku.
Przykładowo nieskończona pętla w 2D wyglądać może tak:

Listing 9: Nieskończona pętla w Befunge

```
>v
^<
```

A prosty program ***Witaj PP!*** tak:

Listing 10: *Witaj PP!* w Befunge

```
"!PP␣jatiW">:v
             |,<
             @
```

Program ten można rozbić na poszczególne części:

1. znakiem **"** rozpoczynamy pobieranie ciągu znaków (zapisujemy na stos) do następnego znaku **"**
2. przechodzimy w prawo **>** i pobieramy znak ze stosu **:**
3. przechodzimy na dół (kolejno **v <**) i wypisujemy znak w reprezentacji ASCII dzięki **,**
4. napotykamy instrukcję IF pionową (|) - jeśli wartość pobrana ze stosu to 0, zejdź na **dół**, jeśli nie, to idź na **górę**
5. jeśli program doszedł do @ to kończy wykonywanie;

3.5 INTERCAL - pionierski język ezoteryczny

INTERCAL jako język programowania został stworzony 26 maja 1972 przez Donalda R. Woodsa oraz Jamesa M. Lyona

[5] https://esolangs.org/wiki/Chris_Pressey

w uniwersytecie Princeton. Jego twórcy mieli jedną aspirację - stworzyć język, który nie ma nic wspólnego z żadnym innym ważnym i popularnym językiem programowania. Mieli oni na myśli takie języki jak *FORTRAN*, *BASIC*, *COBOL*, *ALGOL* i tym podobne. Pełna nazwa języka to *"Compiler Language With No Pronounceable Acronym"*, która z wiadomych powodów została skrócona do **"INTERCAL"**[14].

Pisanie jakiegokolwiek programu w tym języku może być bardzo uciążliwe dla potencjalnego programisty. Nawet tak prosty program jak ***Hello, World!*** może przysporzyć wiele problemów:

Listing 11: *Hello, World!* w INTERCALu

```
1  DO ,1 <- #13
2  PLEASE DO ,1 SUB #1 <- #238
3  DO ,1 SUB #2 <- #108
4  DO ,1 SUB #3 <- #112
5  DO ,1 SUB #4 <- #0
6  DO ,1 SUB #5 <- #64
7  DO ,1 SUB #6 <- #194
8  PLEASE DO ,1 SUB #7 <- #48
9  DO ,1 SUB #8 <- #26
10 DO ,1 SUB #9 <- #244
11 PLEASE DO ,1 SUB #10 <- #168
12 DO ,1 SUB #11 <- #24
13 DO ,1 SUB #12 <- #16
14 DO ,1 SUB #13 <- #162
15 PLEASE READ OUT ,1
16 PLEASE GIVE UP
```

Na pierwszy rzut oka nie wiadomo, co dokładnie program ten wykonuje. Istotny tu jest fakt, że komenda **READ OUT** wypisuje zawartość tablicy, która została akurat zapisana w punkcie **,1** (linia 15). Znak przecinka **,** mówi kompilatorowi, że ma do czynienia z tablicą 16-bitowych liczb całkowitych. Wartości tablicy są wypisywane na wyjście pojedynczo, od lewej do prawej. By móc zdecydować, który znak należy wypisać, bazując na i-tej pozycji tablicy, kompilator przeprowadza operacje według danego algorytmu[15]:

1. odwróć binarnie kod ASCII poprzednio wyświetlonego znaku (dla pierwszego znaku będzie to 0)
2. pobierz i-ty element tablicy
3. odejmij wartość pobranego elementu (2.) od odwróconego binarnie znaku (1.)
4. odwróć binarnie otrzymaną wartość (3.) uzyskując zapisany w reprezentacji ASCII znak do wyświetlenia na i-tej pozycji;

3.6 Malbolge - diabelski język

Malbolge jest znany jako jeden z najbardziej ezoterycznych językó programowania. Jest też również często potocznie nazywany *"Językiem programowania pochodzącym z PIEKŁA"*. Jego nazwa wywodzi się z *Malebolge*, nazwy ósmego kręgu piekła w *Boskiej Komedii* Dantego[6], zarezerwowanego dla sprawców przestępstw i oszustów. Trudność użytkowania tego języka wywodzi się z [16]:

- restrykcyjnych instrukcji;
- podmian instrukcji w trakcie wykonywania;
- restrykcji wczytywanych danych;

Najłatwiej można to zauważyć, analizując kod programu ***Hello, World!***[7]:

Listing 12: *Hello, World!* w Malbolge

```
1  (=<‘#9]~6ZY327Uv4-QsqpMn&+Ij”’E%e{Ab
       ↪ ~w=_:]Kw%o44Uqp0/Q?xNvL:‘H%c#
       ↪ DD2^WV>gY;dts76qKJImZkj
```

Jeśli dana instrukcja nie mieści się w przedziale 33-126 (wartości binarnej danego znaku) to wykonywanie się programu jest kończone. W innym przypadku, by sprecyzować dokładnie, jaka instrukcja zostanie wykonana, wartość wskazywana przez dany rejestr C jest dodawana do siebie, a następnie wynik ten jest dzielony przez 94 w celu otrzymania reszty z dzielenia. Malbolge składa się z 8 instrukcji, każdej przypisanej do innej wartości reszty z dzielenia danego znaku:

(C+[C])%94	Opis
4	Ustawia wskaźnik kodu na wartość wskazywaną przez bieżący wskaźnik danych
5	Wyświetla znak w formacie ASCII (wartość A%256)
23	Pobiera znak i zapisuje go do rejestru A
39	Obraca liczbę o najmniej znaczący bit (np. 0002111112 staje się 2000211111). Zapisuje wynik w A i wskaźniku danych
40	Ustaw wskaźnik danych na wartość wskazywaną przez niego
62	Wykonaj *szaloną* operację ze wskaźnikiem [d] oraz rejestrem A
68	Nic nie rób
81	Zakończ program
jakakolwiek inna wartość	Patrz **wartość 68**

Tabela 5. Spis instrukcji języka **Malbolge**

W **Tabeli 5** (**wartość 62**) wspomniano o tak zwanej *szalonej operacji*:

[6] https://wolnelektury.pl/katalog/lektura/boska-komedia-pieklo/

[7] https://gist.github.com/kspalaiologos/a1fe6913aaff8edea515b4af385368fe

		A		
		0	***1***	***2***
[D]	***0***	1	0	0
	1	1	0	2
	2	2	2	1

Tabela 6. Tabela *szalonej operacji* języka **Malbolge**

Polega ona na zamianie bitów rejestru A oraz wskaźnika [D], a następnie wpisania do obu rejestrów wyniku operacji, np. instrukcja (zaprezentowana za pomocą *crz* dla lepszej czytelności) **crz 0001112220, 0120120120** da wynik 1001022211.

3.7 ArnoldC - terminator wśród ezoteryków

Język bazujący na sławnych, tak zwanych "one-linerach", Arnolda Schwarzeneggera[8], stworzony przez fińskiego programistę Lauri Hartikkego[9] w 2013 roku. Składnia samego języka jest znacznie rozbudowana w porównaniu do wcześniejszych przykładów, natomiast jednymi z ciekawszych i bardziej zabawnych kwestii, a raczej instrukcji tego języka są:

Instrukcja	Znaczenie
@I LIED	False
@NO PROBLEMO	True
BECAUSE I'M GOING TO SAY PLEASE	If
BULLSHIT	Else
YOU HAVE NO RESPECT FOR LOGIC	EndIf
STICK AROUND	While
CHILL	EndWhile
HE HAD TO SPLIT	/
YOU ARE NOT YOU YOU ARE ME	==
LISTEN TO ME VERY CAREFULLY	DeclareMethod
I'LL BE BACK	Return
HASTA LA VISTA, BABY	EndMethodDeclaration
DO IT NOW	CallMethod
YOU SET US UP	SetInitialValue
IT'S SHOWTIME	BeginMain
YOU HAVE BEEN TERMINATED	EndMain
TALK TO THE HAND	Print
I WANT TO ASK YOU A BUNCH OF QUESTIONS AND I WANT TO HAVE THEM ANSWERED IMMEDIATELY	ReadInteger
WHAT THE FUCK DID I DO WRONG	ParseError

Tabela 7. Tabela wybranych instrukcji języka **ArnoldC**

[8]Większość cytatów można znaleźć tutaj: https://youtu.be/ybJWKZB0Erk

[9]PROFIL: https://github.com/lhartikk; PROJEKT: https://github.com/lhartikk/ArnoldC

Spis wszystkich instrukcji możliwy jest to odnalezienia na stronie projektu na profilu GitHub twórcy[9].

Bardzo prosty do napisania jest również klasyczny program ***Witaj PP!***:

Listing 13: *Witaj PP!* w ArnoldC

```
1 IT'S␣SHOWTIME
2 TALK␣TO␣THE␣HAND␣"Witaj␣PP!"
3 YOU␣HAVE␣BEEN␣TERMINATED
```

Istotnym jest, że każda kolejna komenda musi zostać zapisana od nowej linii, zatem kod napisany w taki sposób nie zadziała[17]:

Listing 14: Niepoprawny kod w ArnoldC

```
1 GET TO THE CHOPPER x HERE IS MY
      ↪ INVITATION y KNOCK KNOCK z
      ↪ ENOUGH TALK
```

3.8 Chef - książka kucharska dla programistów

Język stworzony przez Davida Morgana-Mara w 2002 roku. Według opisu ze strony głównej języka "**Chef** to język programowania, w którym pisane programy wyglądają jak przepisy kulinarne"[10].
Zasady pisania w Chefie są następujące[18]:

- Program nie tylko powinien generować poprawne wartości wynikowe, ale powinien być również łatwy do przyrządzenia i smaczny!
- Przepisy mogą być dostępne dla kucharzy dysponujących różnym budżetem;
- Jednostki w przepisach muszą być zdefiniowane w systemie metrycznych, ale dopuszczalne jest użycie tradycyjnych miar kucharskich, takich jak *szklanki* (ang. cups), czy *łyżki stołowe* (ang. tablespoons).

3.8.1 Koncepty zawarte w języku

- **Ingredients** - Składniki przechowują indywidualne wartości używanych danych. Płynne składniki są interpretowane jako znaki Unicode, a suche, bądź niesprecyzowane składniki są wyświetlane jako liczby.
- **Mixing Bowls and Baking Dishes** - składniki w tej strukturze są uporządkowane w danej kolejności - struktura działająca na zasadzie stosu "naleśników amerykańskich"[11] z pewnymi restrykcjami:
  - nowe **składniki** są wkładane na górę stosu
  - usuwanie **składników** polega na ściąganiu składnika z góry stosu

[10]"Chef is a programming language in which programs look like recipes.", https://www.dangermouse.net/esoteric/chef.html, uzyskano dostęp 26.06.2022

[11]"like a stack of pancakes", https://www.dangermouse.net/esoteric/chef.html, uzyskano dostęp 26.06.2022

– jeśli zmieni się wartość **składnika**, to nie wpływa ona na wartość zapisaną w **Mixing Bowl**, czy **Baking Dish**
– wartości zapisane na stosie utrzymują swoje *mokre* lub *suche* właściwości;

3.8.2 Elementy składni

Następujące elementy pojawiają się w przepisach, niektóre z nich są opcjonalne. Każdy pojedynczy element musi być przedstawiony w ukazanej kolejności, z pustą linią pomiędzy każdym elementem:

1. **Tytuł przepisu** - opisuje pokrótce działanie programu; zawsze w pierwszej linii programu, zakończony kropką;
2. **Komentarze** - *opcjonalne*, pisanie w formie paragrafu po tytule;
3. **Lista składników** używanych przez program. *Opcjonalna*. Składnia wygląda następująco:
   INGREDIENTS.
   [WARTOŚĆ] [[TYP-MIARY] MIARA] NAZWA
   [KOLEJNE SKŁADNIKI]
   - **g | kg | pinch[es]** - *suche* miary
   - **ml | l | dash[es]** - *mokre* miary
   - **cup[s] | teaspoon[s] | tablespoon[s]** - miary obu rodzajów

   Opcjonalnie można określić jeszcze TYP-MIARY: **heaped | level**, określa on miary *suche*; Wartości składników, których nazwy się powtarzają, są nadpisywane;
4. **Czas przyrządzenia** jest *opcjonalny*. Czas jest liczbą.
   COOKING TIME: *czas* (*godzin[y]* | *minut[y]*).
5. **Temperatura piekarnika** jest *opcjonalna*. Niektóre przepisy wymagają pieczenia.
   PRE-HEAT OVEN TO *temp* DEGREES CELSIUS.
6. **Metody** zawierają faktyczne instrukcje dotyczące przepisu. Każda metoda zapisana jest w danej sentencji. Ignorowane są znaki nowego wiersza.
   METHOD.
   składnia danej metody
7. **Serwowanie**, ostatnie twierdzenie w przepisie.
   SERVES *liczba-gości*
   - wyświetla zawartość **Baking Dish**, jednocześnie usuwając ze stosu jego składniki, dopóki nie będzie puste. Zaczynając od naczynia pierwszego, przechodzi do kolejnych, dopóki wszystkie naczynia nie zostaną wyświetlone.

   Jest to wartość *opcjonalna*, ale wymagana, jeśli przepis ma wyświetlić cokolwiek.

Program ***Hello, world!*** napisany w tym języku wygląda bardzo smakowicie[12]:

Listing 15: *Hello, world!* w Chef

```
1  Hello World Souffle.
2
3  This recipe prints the immortal
       ↪ words "Hello world!", in a
       ↪ basically brute force way. It
       ↪  also makes a lot of food for
       ↪  one person.
4
5  Ingredients.
6  72 g haricot beans
7  101 eggs
8  108 g lard
9  111 cups oil
10 32 zucchinis
11 119 ml water
12 114 g red salmon
13 100 g dijon mustard
14 33 potatoes
15
16 Method.
17 Put potatoes into the mixing bowl.
       ↪ Put dijon mustard into the
       ↪ mixing bowl. Put lard into
       ↪ the mixing bowl. Put red
       ↪ salmon into the mixing bowl.
       ↪ Put oil into the mixing bowl.
       ↪  Put water into the mixing
       ↪ bowl. Put zucchinis into the
       ↪ mixing bowl. Put oil into the
       ↪  mixing bowl. Put lard into
       ↪ the mixing bowl. Put lard
       ↪ into the mixing bowl. Put
       ↪ eggs into the mixing bowl.
       ↪ Put haricot beans into the
       ↪ mixing bowl. Liquefy contents
       ↪  of the mixing bowl. Pour
       ↪ contents of the mixing bowl
       ↪ into the baking dish.
18
19 Serves 1.
```

3.9 Unreadable - aż za bardzo nieczytelny

Unreadable jest językiem ezoterycznym powstałym dzięki użytkownikowi *TehZ*[13]. Jest to jeden z języków, tak jak INTERCAL, stworzonych po to, by być jak najmniej czytelne i zrozumiałe dla programisty i czytelnika. Ten język jest natomiast trudniejszy od poprzedników w takim sensie, że bazuje on na dwóch bliźniaczych znakach, których zapis za pomocą wielu zestawów czcionek jest nie odróżnienia dla człowieka - rozróżnić je można tylko za pomocą reprezentacji binarnej. Są to znaki apostrofu oraz cudzysłowu:

' "

Pisząc programy w tym języku nie wolno używać spacji, ani znaków nowego wiersza. Co ciekawe, **Unreadable**

[12] https://www.dangermouse.net/esoteric/chef_hello.html, uzyskano dostęp 26.06.2022

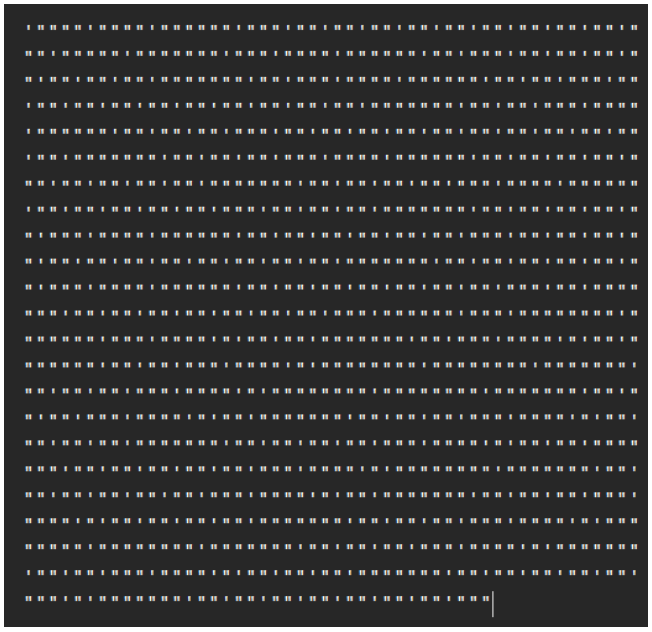
[13] https://esolangs.org/wiki/User:TehZ

jest językiem Turing-kompletnym, a jego składnia zawiera zaledwie 10 instrukcji[19]:

Instrukcja	Opis
’"X	Wyświetl X jako znak Unicode i zwróć wartość X
’""X	Zwróć X+1
’"""	Zwróć 1
’"""XY	Wykonaj X i Y, zwróć wartość zwracaną przez Y
’""""XY	Wykonaj Y, gdy X != 0 i zwróć ostatni zapisany wynik
’"""""XY	Przypisz wartość X do Y i zwróć Y
’""""""X	Zwróć wartość X lub 0, jeśli zmienna nigdy nie została zainicjowana
’"""""""X	Zwróć X-1
’""""""""XYZ	Jeśli X != 0: Wykonuj Y, w innym przypadku wykonuj Z. Zwróć wartość $\frac{Y}{Z}$
’"""""""""	Zwróć pojedynczy znak Unicode wczytywany z wejścia lub -1 - jeśli bufor wejścia został wyczerpany

Tabela 8. Spis instrukcji języka **Unreadable**

Zgodnie z opisem tego języka, każdy program jest bardzo nieczytelny dla zwykłego użytkownika, a dobrym przykładem jest zwykły program wyświetlający napis ***Hello World***[14]:



Rysunek 3. ***Hello World*** w języku Unreadable

3.10 Whitespace - nieczytelny, ale potężny

"Niematerialność jest powtarzającym się motywem języków ezoterycznych, być może poprzez fakt, iż języki są praktycznie niczym - zestawem zasad, bez żadnej szczególnej implementacji."[20]

Najlepszym przykładem takich języków jest **Whitespace**, stworzony w 2003 roku przez Edwina Bradiego[15] oraz Chrisa Morrisa, w pełni funkcjonalny język kodowany za pomocą JEDYNIE 3 znaków białych: *spacji*, *tabulatora*, *znaku nowego wiersza - Entera*.

Dzięki takim właściwościom plik zawierający program napisany w tym języku, może sprawiać wrażenie pustego. (!) Co ciekawe, nawet Bjarne Stroustroup - twórca języka C++ - zasugerował dodanie możliwości przeciążania znaków białych, co byłoby równoważne z możliwością przypisywania danych akcji, które mogłyby być wykonywane przy użyciu tych znaków. Przykładowo mnożenie dwóch liczb bez użycia znaku mnożenia (*).

3.10.1 Składnia języka Whitespace

Przed każdą komendą należy wpisać tak zwane Parametry Modyfikacji Instrukcji (ang. **IMP** - Instruction Modification Parameters). Składnia jednej pełnej instrukcji wygląda następująco:

IMP KOMENDA PARAMETRY

- parametry zakończone są znakiem nowego wiersza.

IMP	Rodzaj komendy
[Tab][LF]	I/O
[Space]	Manipulacja na stosie
[Tab][Space]	Arytmetyka
[LF]	Kontrola przepływu
[Tab][Tab]	Dostęp do elementów na stosie

Tabela 9. Spis instrukcji IMP języka **Whitespace**

Podział na komendy i ich reprezentacje w języku:

- **Liczby** - by móc wprowadzić liczbę, najpierw należy określić jej znak
  - [SPACE] - dodatnia
  - [TAB] - ujemna

  Następnie należy wprowadzić liczbę w postaci binarnej
  - [SPACE] - 0
  - [TAB] - 1
- **Manipulacje wejścia/wyjścia**

Komenda	Opis
[Tab][Space]	Wczytaj znak i umieść go na szczycie stosu
[Tab][Tab]	Wczytaj liczbę i umieść ją na szczycie stosu
[Space][Space]	Wyświetl znak ze szczytu stosu
[Space][Tab]	Wyświetl liczbę ze szczytu stosu

Tabela 10. Spis komend I/O języka **Whitespace**

[14] https://esolangs.org/wiki/Talk:Unreadable, uzyskano dostęp 26.06.2022

[15] https://esolangs.org/wiki/Edwin_Brady

- **Manipulacja stosu**

Komenda	Opis
[Space]	Wypchnij liczbę na szczyt stosu (parametr **LICZBA**)
[LF][Space]	Zduplikuj element na szczycie stosu
[LF][Tab]	Zamień miejscami dwa szczytowe elementy stosu
[LF][LF]	Odrzuć element ze szczytu stosu
[Tab][Space]	Kopiuj *n-ty* (n - parametr) element stosu na szczyt
[Tab][LF]	Usuń *n*-elementów (n - parametr) ze stosu, zachowując ten na szczycie

Tabela 11. Spis komend manipulacji stosem języka **Whitespace**

- **Arytmetyka** - opeacje na dwóch szczytowych elementach, wynik operacji nadpisuje je.

Komenda	Opis
[Space][Space]	Dodawanie
[Space][Tab]	Odejmowanie
[Space][LF]	Mnożenie
[Tab][Space]	Dzielenie całkowite
[Tab][Tab]	Dzielenie modulo

Tabela 12. Spis komend arytmetycznych języka **Whitespace**

- **Kontrola przepływu**

Komenda	Opis
[Space][Space]	Znacznik lokalizacji odwołania w programie (parametr LABEL)
[Space][Tab]	Przywołaj podprogram (parametr LABEL)
[Space][LF]	Skocz do znacznika (parametr LABEL)
[Tab][Space]	Skocz do znacznika, jeśli szczytowy element == 0 (parametr LABEL)
[Tab][Tab]	Skocz do znacznika, jeśli szczytowy element jest ujemny (parametr LABEL)
[Tab][LF]	Zakończ funkcję i wróć do miejsca jej wywołania
[LF][LF]	Zakończ program

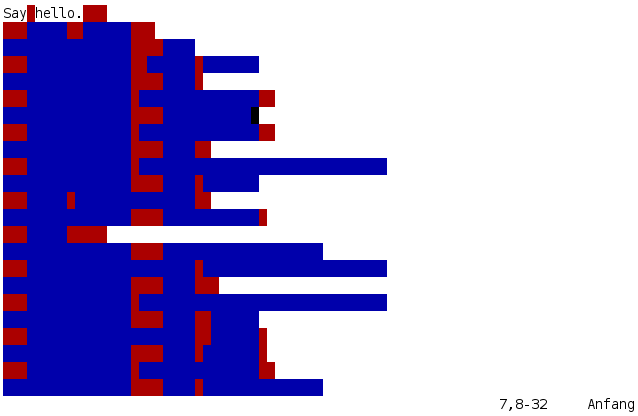
Tabela 13. Spis komend kontroli przepływu języka **Whitespace**

- **Dostęp do elementów na stosie**

Komenda	Opis
[Space]	Wrzuć wartość na stos
[Tab]	Pobierz wartość ze stosu

Tabela 14. Spis komend dostępu do stosu języka **Whitespace**

Przykładowy program z kolorowaną składnią wyświetlający napis ***Hello World***[16]:



Rysunek 4. Program wyświetlający napis ***Hello World*** w języku Whitespace

- kolorem niebieskim zaznaczono znaki tabulatora [TAB]
- kolorem czerwonym zaznaczono znaki spacji [SPACE]

4. Podsumowanie i wnioski

Języki ezoteryczne na pierwszy rzut oka wydają się być bezsensownymi tworami, czasem ciekawymi i intrygującymi, a innym razem frustrującymi zbiorami z pozoru bezsensownych układów znaków, czy bluźnierczych wyrazów.

Przyglądając im się coraz uważniej, można natomiast odkryć, jak fascynująca jest nauka i komputer, którym ogromna rzesza ludzi na świecie posługuje się codziennie.

Jednym z większych problemów, na jakie wskazują języki ezoteryczne jest ilość komend i jednocześnie wymagana pojemność pamięci, jaka jest potrzebna, by móc przetworzyć ten sam problem programistyczny. Spośród 11 wymienionych i opisanych języków można odnaleźć te, które potrzebują zaledwie **kilku/kilkunastu bajtów**, by móc wykonać klasyczny program wyświetlający napis *"Hello, World!"*.

Jednocześnie możliwym jest stworzenie takiego programu w innych językach, który jest nie tylko w zupełności nieczytelny dla programisty, ale jednocześnie zajmuje ogromną ilość miejsca. Zalet takich języków należy doszukiwać się natomiast w składni użytej do prezentacji danego problemu. Na przykładzie języka **Whitespace** można wykazać, że konstruując odpowiednio język kompilatora, można osiągnąć twór, który zawiera w sobie tylko 3 znaki, pozwalający jednocześnie w pełni na użytkowanie go w taki sposób jak inne popularne języki programowania. Jest to

[16] https://commons.wikimedia.org/wiki/File:Whitespace_in_vim2.png, uzyskano dostęp 26.06.2022

również bardzo ważny dowód, ukazujący uniwersalność komputerów użytkowanych we wszelakiej postaci.

Twórca danego języka ezoterycznego może stworzyć zestaw reguł i reprezentacji składni, który nie będzie w niczym odstawał od zwykłego tekstu beletrystycznego, dramatu, czy przepisu na smaczny suflet czekoladowy, a przy tym program w nim napisany będzie całkowicie wykonywalny.

Fascynującym we wszystkich tych pracach jest fakt, że każdy z języków został napisany i przygotowany z pasją i chęcią stworzenia czegoś nowego, innego, dziwnego i trudnego do zrozumienia, ale jednocześnie przemyślanego i dobrze skonstruowanego.

Dzięki takiemu podejściu można wysnuć wniosek, że za pomocą komputera można przedstawić praktycznie wszystko w formie wszelakiej, ale pozwalające uzyskać ten sam wynik końcowy - wystarczy tylko cierpliwość, zaangażowanie, pomysł i dużo wyobraźni.

Literatura

[1] Ezoteryczny język programowania. https://pl.wikipedia.org/wiki/Ezoteryczny_j%C4%99zyk_programowania, uzyskano dostęp 22.06.2022.

[2] Esoteric programming language. 2022. https://esolangs.org/wiki/Esoteric_programming_language#External_resources, uzyskano dostęp 25.06.2022.

[3] Stephen Cass. Top programming languages 2021, 2021. https://spectrum.ieee.org/top-programming-languages-2021, uzyskano dostęp 22.06.2022.

[4] Sebastian Morr. Esoteric programming languages. 2015.

[5] Adrian De Wynter. Turing Completeness and Sid Meier's Civilization. *IEEE Transactions on Games*, pages 1–1, 2022.

[6] Alan J. Perlis. Epigrams on programming. *ACM SIGPLAN Notices*, 17(9), 1982.

[7] Simon Mathis. brainfuck-programming language. 2011.

[8] Ascii code - the extended ascii table. https://www.ascii-code.com, uzyskano dostęp 25.06.2022.

[9] Shortest possible "hello, world!"program written in brainfuck. https://codegolf.stackexchange.com/a/163590/59487, uzyskano dostęp 26.06.2022.

[10] Fuckfuck. 2014. https://esolangs.org/wiki/Fuckfuck, uzyskano dostęp 26.06.2022.

[11] Moanfuck. 2020. https://esolangs.org/wiki/Moanfuck, uzyskano dostęp 26.06.2022.

[12] Ook! 2022. https://esolangs.org/wiki/Ook!, uzyskano dostęp 26.06.2022.

[13] Befunge. 2022. https://esolangs.org/wiki/Befunge, uzyskano dostęp 26.06.2022.

[14] Donald R Woods and James M Lyon. The intercal programming language reference manual, 1973.

[15] http://progopedia.com/users/Nickolas/ Nickolas. Hello, world! in intercal, 2011. http://progopedia.com/example/hello-world/257/, uzyskano dostęp 26.06.2022.

[16] Masahiko Sakai. Introduction to esoteric language malbolge. In *Japan-Vietnam Workshop on Software Engineering*, pages 15–19, 2010.

[17] Arnoldc. 2020. https://esolangs.org/wiki/ArnoldC, uzyskano dostęp 26.06.2022.

[18] David Morgan-Mar. Chef, 2022. https://www.dangermouse.net/esoteric/chef.html, uzyskano dostęp 26.06.2022.

[19] Unreadable. 2018. https://esolangs.org/wiki/Unreadable, uzyskano dostęp 26.06.2022.

[20] Daniel Temkin. Language without code: intentionally unusable, uncomputable, or conceptual programming languages. *Journal of Science and Technology of the Arts*, 9(3):83–91, Sep. 2017.