# 1. Maszyna Turinga, gramatyki formalne i ONP

### *1.1.Maszyna Turinga*

Automat skończony składa się ze skończonego zbioru stanów i zbioru przejść ze stanu do stanu, zachodzących przy różnych symbolach wejściowych wybranych z pewnego alfabetu Σ. Dla każdego symbolu wejściowego istnieje dokładnie jedno przejście z każdego stanu (które może w szczególnym przypadku prowadzić do tego samego stanu) odpowiadające temu symbolowi. Jeden ze stanów, oznaczany zazwyczaj symbolem $q_0$, jest stanem początkowym, od którego automat rozpoczyna działanie. Niektóre ze stanów pełnią rolę końcowych.

Automat skończony przedstawiamy formalnie jako piątkę uporządkowaną $(Q, \Sigma, \delta, q_0, F)$, gdzie Q jest zbiorem stanów, Σ jest skończonym alfabetem wejściowym, $q_0$ należące do Q jest stanem początkowym, $F \subseteq Q$ jest zbiorem stanów końcowych, a δ jest funkcją odwzorowującą $Q \times \Sigma$ w Q. Tak więc δ przyporządkowuje każdemu stanowi q i każdemu symbolowi wejściowemu a nowy stan $\delta(q,a)$.

Maszyna Turinga jest automatem abstrakcyjnym, służącym do analizy algorytmów. Składa się ona z podzielonej na kratki nieskończenie długiej taśmy oraz głowicy przesuwającej się nad taśmą. Głowica ma możliwość czytania i zapisywania symboli na taśmę. Zdefiniujemy teraz podstawowe pojęcia używane do opisu maszyny Turinga.

Zbiór symboli $S = \{ s_i \}$ - zbiór symboli, które będą przetwarzane przez maszynę Turinga, przyjęty jako alfabet.

Napis - ciąg symboli alfabetu S.

Głowica - rozróżnia, czyta i zapisuje symbole alfabetu S. Może przesuwać się wzdłuż taśmy. Przyjmuje zawsze jeden z m stanów $q_1, \ldots, q_m$.

Stan maszyny - określa jednoznacznie stan głowicy $q_j$ i odczytany przez nią symbol $s_i$:

$S_{ij} = (s_i, q_j)$.

Ruch maszyny - opisuje akcję, jaką ma wykonać maszyna Turinga. Ruch maszyny jest reakcją maszyny na stan $S_{ij}$. Opisywany jest za pomocą trzech parametrów $R_{ij} = (s_k, K, q_l)$. $s_k$ jest symbolem, który ma być zapisany przez głowicę, K określa sposób przesunięcia taśmy, a $q_l$ charakteryzuje stan, do którego ma przejść głowica.

Po zapoznaniu się a powyższymi określeniami przejdźmy do opisu działania oraz programowania maszyny Turinga. Po pierwsze należy odnotować fakt, iż każdy ruch Rij jest związany jednoznacznie ze stanem maszyny Sij.

Możemy zatem zapisać następującą zależność:

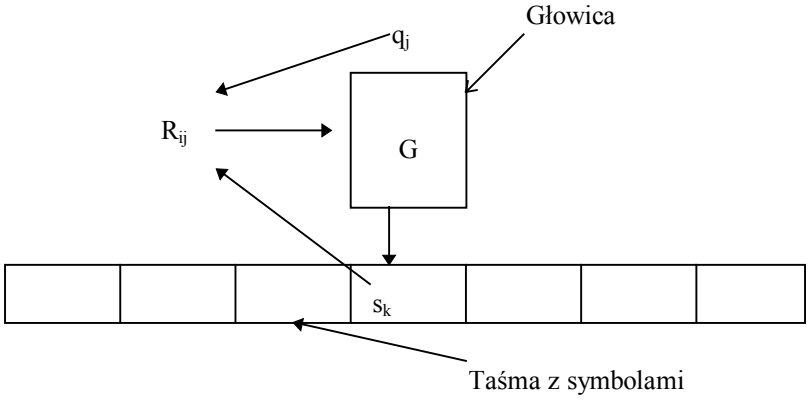
$R_{ij} = T( S_{ij} )$

Funkcję T nazywamy tablicą charakterystyczną maszyny Turinga. Nazwa ta związana jest z reprezentacją tej funkcji. Przedstawimy teraz przykładową tablicę charakterystyczną.

| T | $q_1$ | ... | $q_j$ | ... | $q_m$ |
|---|---|---|---|---|---|
| $s_1$ | $R_{11}$ | ... | $R_{1j}$ | ... | $R_{1m}$ |
| ... | ... | ... | ... | ... | ... |
| $s_i$ | $R_{i1}$ | ... | $R_{ij}$ | ... | $R_{im}$ |
| ... | ... | ... | ... | ... | ... |
| $s_n$ | $R_{n1}$ | ... | $R_{nj}$ | ... | $R_{nm}$ |

$R_{ij} = (s_k, K, q_l)$

Interpretacja tej tablicy jest następująca: Jeżeli będąc w stanie $q_j$ głowica odczytała symbol $s_i$, to należy zapisać na taśmie symbol $s_k$, przesunąć taśmę w kierunku określonym przez K, a głowica ma przejść do stanu $q_l$. Sytuację tę zilustrujemy rysunkiem.




**Rysunek 1.1-1**

### 1.1.1.Twierdzenie:

Każdy algorytm może być zrealizowany przez odpowiednio zaprogramowaną (za pomocą tablicy charakterystycznej) maszynę Turinga.

#### *1.1.1.1.Przykład*

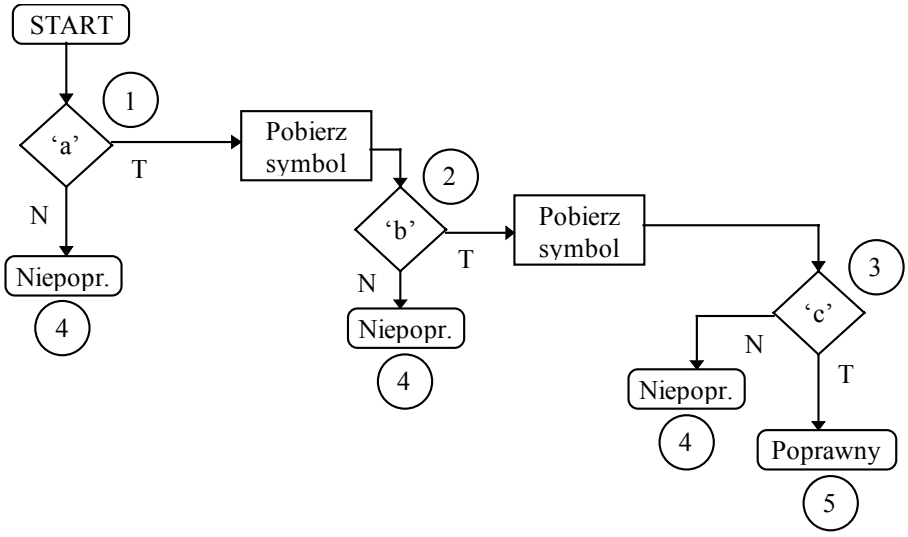
W zbiorze napisów trójliterowych utworzonych z liter a, b, c tylko napis abc jest poprawny. Podać algorytm rozpoznawania tego napisu.

##### 1.1.1.1.1.Rozwiązanie

Algorytm realizujący powyższe zadanie przedstawimy w postaci opisu słownego.

1. Pobierz symbol. Jeżeli jest nim a, to pobierz następny symbol i idź do 2, w przeciwnym przypadku idź do 4.
2. Jeżeli pobranym symbolem jest b, to pobierz następny symbol i przejdź do 3, jeżeli nie, to idź do 4.
3. Jeżeli pobranym symbolem jest c, idź do 5, w przeciwnym przypadku idź do 4.
4. Sygnalizuj nieprawidłowy napis.
5. Sygnalizuj napis poprawny.

Schemat blokowy algorytmu.




**Rysunek 1.1-2**

Przedstawimy teraz zapis tego algorytmu w postaci tablicy charakterystycznej maszyny Turinga. W tym celu określimy 5 stanów maszyny odpowiadających kolejnym krokom algorytmu. Zgodnie z przyjętym sposobem opisu tabeli stany te zaznaczymy w najwyższym wierszu tabeli. W pierwszej kolumnie umieszczone zostaną wszystkie możliwe symbole, czyli litery a, b, c. Ponieważ algorytm nie wymaga zapisywania żadnych symboli na taśmę i jednocześnie nie występuje konieczność zmiany kierunku przesuwania taśmy, do opisu ruchu maszyny użyjemy następujących symboli:

P - pobierz następny symbol,

N - nie pobieraj niczego,

$q_n$ - przejdź do stanu n,

sam ruch charakteryzując tylko dwoma parametrami:

$R_{ij} = (K, q_l)$

Oto tabela charakterystyczna.

| | $q_1$ | $q_2$ | $q_3$ | $q_4$ | $q_5$ |
|---|---|---|---|---|---|
| a | $Pq_2$ | $Nq_4$ | $Nq_4$ | $Nq_4$ | $Nq_5$ |
| b | $Nq_4$ | $Pq_3$ | $Nq_4$ | $Nq_4$ | $Nq_5$ |
| c | $Nq_4$ | $Nq_4$ | $Nq_5$ | $Nq_4$ | $Nq_5$ |

Zauważmy, że stany $q_4$ i $q_5$ są stanami, w których zatrzyma się maszyna po wykonaniu algorytmu. Warto zwrócić tutaj uwagę na semantykę tych stanów: zatrzymanie się maszyny w stanie $q_4$ oznacza, że wprowadzony ciąg znaków nie był poprawny, a zatrzymanie w stanie $q_5$ oznacza że był.

#### *1.1.1.2.Przykład*

Zaprojektować maszynę Turinga obliczającą sumę dowolnej liczby zapisanej w systemie dziesiętnym i liczby 1.

##### 1.1.1.2.1.Rozwiązanie

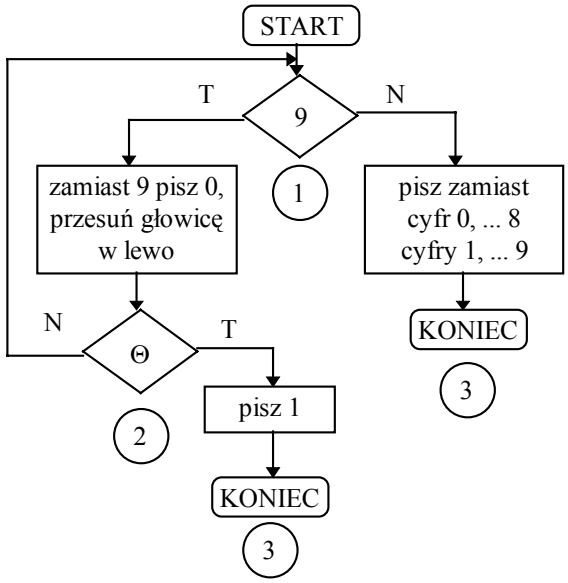
Aby do dowolnej liczby dodać 1, należy przeanalizować jej ostatnią cyfrę. Jeżeli jest nią któraś z cyfr 0-8, to dodanie jedynki sprowadzi się do zapisania w tym miejscu danej cyfry zwiększonej o jeden, tzn. Jednej z cyfr 1-9. Jeśli ostatnią cyfrą jest cyfra 9, to należy w jej miejscu zapisać cyfrę 0, a następnie dodać 1 do cyfry przedostatniej lub napisać jedynkę, jeżeli cyfra ta nie występuje. A oto zapis tego algorytmu w języku naturalnym (zaczynamy analizę liczby od jej ostatniej cyfry - maszyna Turinga powinna być wstępnie ustawiona właśnie na tej cyfrze):

1. Jeżeli cyfra z zakresu 0-8 pisz odpowiednio 1-9 i idź do 3, w przeciwnym przypadku pisz 0, przesuń głowicę w lewo i idź do 2.
2. Jeżeli miejsce puste pisz 1 i idź do 3, jeśli nie idź do 1.
3. Sygnalizuj koniec.

Prześledźmy dla przykładu dodanie jedynki do liczby 89:

| Stan maszyny | $q_1$ | $q_2$ | $q_1$ | $q_3$ |
|---|---|---|---|---|
| Stan taśmy | 89 | 80 | 80 | 90 |
| | _↑ | ↑_ | ↑_ | ↑_ |

Każda kolumna tabeli ilustruje jeden krok algorytmu. Strzałki pionowe oznaczają pozycję głowicy maszyny dla danego stanu (np. dla pierwszej kolumny - maszyna znajduje się w stanie $q_1$, głowica ustawiona jest na symbolu '9'). A oto schemat blokowy oraz tablica charakterystyczna:




**Rysunek 1.1-3**

| | $q_1$ | $q_2$ | $q_3$ |
|---|---|---|---|
| 9 | $0Lq_2$ | $q_1$ | $q_3$ |
| Θ | - | $1q_3$ | $q_3$ |
| 0 | $1q_3$ | $q_1$ | $q_3$ |
| 1 | $2q_3$ | $q_1$ | $q_3$ |
| ... | ... | ... | ... |
| 8 | $9q_3$ | $q_1$ | $q_3$ |

Podobnie jak w poprzednim przykładzie w opisie ruchów maszyny pominięto nie występujące w danej sytuacji czynności. Poniżej przedstawimy przykład działania maszyny dla liczby 199.

| stan maszyny | $q_1$ | $q_2$ | $q_1$ | $q_2$ | $q_1$ | $q_3$ |
|---|---|---|---|---|---|---|
| stan taśmy | 199 | 190 | 190 | 100 | 100 | 200 |
| | __↑ | _↑_ | _↑_ | ↑__ | ↑__ | ↑__ |

### *1.2.Języki i gramatyki formalne*

Zapoznamy się z podstawami lingwistyki formalnej oraz przykładami jej zastosowania. Podamy systemy opisu języków formalnych oraz omówimy Odwrotną Notację Polską a także zaprezentujemy algorytm konwersji wyrażeń arytmetycznych na ONP.

Aby uświadomić sobie, czym są języki formalne, posłużymy się porównaniem z językiem naturalnym który powstał jako środek porozumiewania się ludzi ze sobą. Na świecie znanych jest bardzo wiele języków. Do cech charakterystycznych języka

naturalnego można zaliczyć dużą swobodę konstruowania zdań (brak ścisłych reguł gramatycznych), dużą ilość wyjątków, etc. Nic więc dziwnego, że rozwój nauk ścisłych wymusił opracowywanie systemów formalnych umożliwiających ścisły i jednoznaczny opis tworzonych konstrukcji. Systemy te noszą nazwę języków formalnych. Za twórcę lingwistyki formalnej uważa się Chomsky'ego.

Uściślijmy pojęcie języka formalnego. Składają się na niego następujące elementy: alfabet, słowo, język, gramatyka

alfabet - dowolny zbiór symboli, z których będą zestawiane słowa języka,

słowo - uporządkowany ciąg symboli alfabetu,

język - zbiór wszystkich możliwych słów,

gramatyka - zbiór reguł umożliwiających generowanie słów danego języka jak też rozróżnienie słów poprawnie zbudowanych od zbudowanych niepoprawnie.

#### *1.2.1.1.Przykład*

Alfabet - {a, b}

Słowo - np. aaaab, bbbb, a, b

Gramatyka -

Reguła 1: a jest poprawne (tzn. słowo złożone z pojedynczego symbolu a jest poprawne).

Reguła 2: Jeżeli $\alpha$ jest poprawne, to a$\alpha$b też jest poprawne.

Przykłady poprawnych słów:

a - na podstawie Reguły 1.

aab - na podstawie Reguły 1 i Reguły 2.

aaabbb - na podstawie Reguły 1 i dwukrotne zastosowanie Reguły 2.

### 1.2.2.Gramatyki Chomsky'ego

Gramatykę Chomsky'ego określa związek 4 elementów:

$G = < V, \Sigma, P, \sigma >$

Znaczenie poszczególnych symboli jest następujące:

V - alfabet terminalny: określa zbiór symboli, z których będą budowane słowa generowane przez gramatykę G.

$\Sigma$ - alfabet metajęzyka: zbiór symboli pomocniczych używanych przy określaniu reguł opisujących konstruowanie słów poprawnych w sensie gramatyki G.

P - lista produkcji: zestaw reguł.

$\sigma$ - głowa (aksjomat) języka: podaje z jakiej konstrukcji można wyprowadzić wszystkie generowane przez gramatykę G napisy.

Język generowany przez gramatykę G jest to zbiór wszystkich możliwych słów powstałych na bazie listy produkcji i wyprowadzonych z głowy języka.

$L(G) = \{ x : x \in Z \wedge \sigma \Rightarrow x \}$,

gdzie Z jest zbiorem wszystkich możliwych słów, zaś zapis $\Rightarrow$ oznacza proces wywodu słowa x z głowy $\sigma$ na podstawie listy produkcji P.

#### *1.2.2.1.Przykład*

Rozważmy gramatykę generującą proste wyrażenia arytmetyczne. Poniżej przedstawiono poszczególne elementy składowe gramatyki.

Alfabet terminalny V: $V = \{ a, b, d, +, *, (, ) \}$

Alfabet metajęzyka $\Sigma$: $\Sigma = \{ W, S, C \}$

Semantyka powyższych symboli jest następująca:

W - wyrażenie,

S - składnik,

C - czynnik.

Głowa języka $\sigma$: $\sigma = W$

lista produkcji P:

$P = \{$

$W \rightarrow S$

$W \rightarrow W + S$

$W \rightarrow S + W$

$S \rightarrow C$

$S \rightarrow C * S$

$S \rightarrow S * C$

$C \rightarrow a$

$C \rightarrow b$

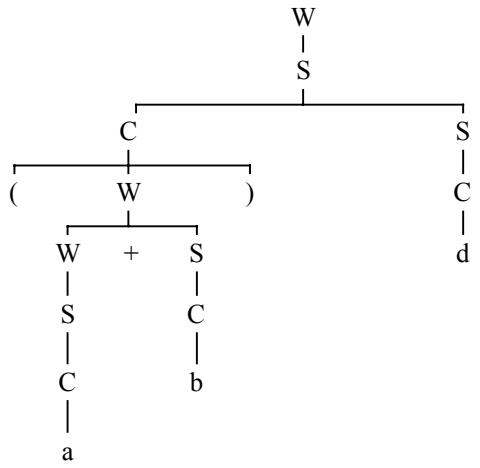
$C \rightarrow d$

$C \rightarrow ( W ) \}$

Zastanówmy się teraz nad następującym problemem: W jaki sposób można wykazać, że wyrażenie ( a + b ) * d jest prostym wyrażeniem arytmetycznym. Spróbujemy tego dokonać wyprowadzając napis ( a + b ) * d z głowy języka na podstawie listy produkcji. Wyprowadzenie to może mieć następującą postać:

W
S
C * S
( W ) * S
( W + S ) * C
( S + S ) * C
( C + C ) * C
( a + b ) * d

Jak widać, udało się wyprowadzić zadany napis. Jest on więc prostym wyrażeniem arytmetycznym. Przedstawimy jeszcze jeden sposób dowodzenia poprawności napisu - drzewo wywodu.




**Rysunek 1.2-1**

### 1.2.3.Notacja Backusa

Notacja Backusa jest jednym ze sposobów prezentowania listy produkcji. Składa się ona z szeregu symboli o określonej semantyce, ułatwiających zdefiniowanie gramatyki. Symbole metajęzyka w notacji Backusa zapisujemy przy użyciu pary nawiasów < >. Notacja posługuje się ponadto symbolem ::= rozumianym jako “jest zdefiniowane jako” oraz symbolem | oznaczającym alternatywę. W celu zaprezentowania sposobu posługiwania się notacją Backusa zdefiniujemy za jej pomocą poprzednio opisaną gramatykę generującą proste wyrażenia. Oto lista produkcji tej gramatyki zapisana przy użyciu notacji Backusa:

<W> ::= <S> | <W> + <S> | <S> + <W>
<S> ::= <C> | <C> * <S> | <S> * <C>
<C> ::= ( <W> ) | a | b | d

Należy zwrócić uwagę na fakt, że symbole zapisane bez nawiasów ostrych < > i nie będące symbolami używanymi przez notację Backusa są symbolami terminalnymi.

### 1.2.4.Beznawiasowe algebry Łukasiewicza

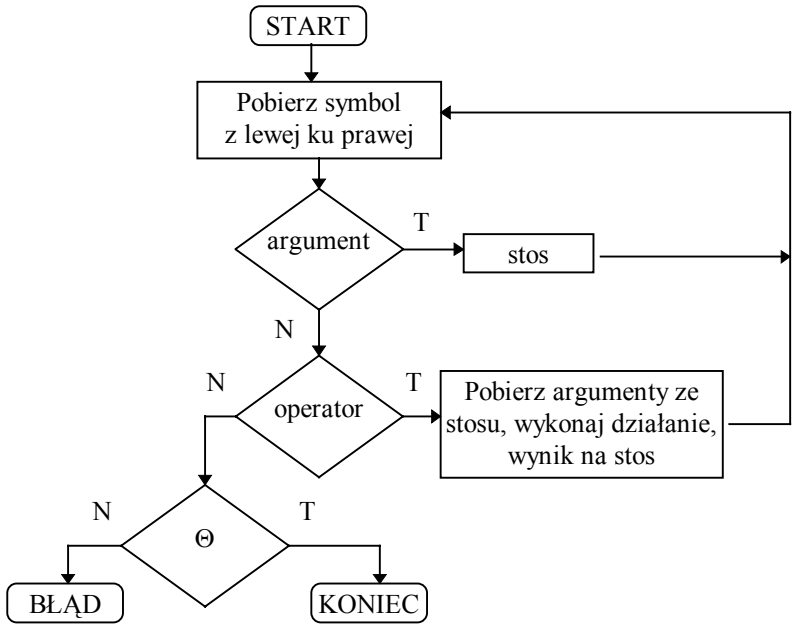
Notacją Backusa posłużymy się teraz do zdefiniowania beznawiasowej algebry Łukasiewicza, zwanej Odwrotną Notację Polską. Jest to przykład notacji umożliwiającej zapisanie dowolnego wyrażenia arytmetycznego bez użycia nawiasów zmieniających kolejność wykonywania działań. Ze względu na łatwość obliczania wyrażeń zapisanych w ONP przy użyciu stosu znalazła ona szerokie zastosowanie w arytmetyce komputerów. Przedstawimy teraz gramatykę pozwalającą na generowanie napisów w ONP oraz zaprezentujemy algorytm dokonujący zamiany wyrażeń zapisanych w arytmetyce nawiasowej na ONP.

Gramatyka prostych wyrażeń arytmetycznych w ONP
V = { a, b, d, +, * }
P:
<W> ::= <Z> <Z> <O> | <Z>
<Z> ::= <X> <X> <O > | <X>
<X> ::= a | b | d
<O> ::= + | *

A oto przykład prostego wyrażenia arytmetycznego i odpowiadający mu zapis w ONP: ( a + b ) * d ≡ a b + d *

Należy zaznaczyć, że w podanych tutaj definicjach gramatyk zbiór symboli terminalnych został ograniczony jedynie w celu zwiększenia czytelności. Nic nie stoi na przeszkodzie, aby zdefiniować gramatykę ONP na podstawie pełnego zestawu symboli stosowanego w matematyce.

Spróbujemy teraz opracować algorytm obliczający wartość wyrażenia zapisanego w ONP. Jak już zaznaczono, bardzo wygodne będzie się posłużenie się stosem jako strukturą przechowującą wyniki pośrednie. Dodatkową zaletą tej struktury jest łatwość, z jaką daje się ona zaimplementować w systemie komputerowym. Nasz algorytm będzie wyglądał następująco: będziemy pobierali po kolei symbole wyrażenia od lewej strony do prawej; jeżeli symbol będzie liczbą (zmienną), to odłożymy go na stos; jeżeli będzie opisem operacji - pobieramy elementy ze stosu, wykonujemy tę operację, a wynik przesyłamy z powrotem na stos. Czynności te wykonujemy aż do wyczerpania się danych wejściowych. Na stosie znajdziemy wtedy obliczaną wartość wyrażenia. Na rysunku przedstawimy teraz schemat blokowy realizujący tę ideę, a później przykład jego zastosowania.




**Rysunek 1.2-2**

#### *1.2.4.1.Przykład*

Obliczyć wartość wyrażenia: 2 3 4 5 + * +

W celu zilustrowania sposobu obliczania wyrażenia posłużymy się tabelką. Przedstawimy w niej symbole pojawiające się kolejno na wejściu oraz zawartość stosu po każdym obiegu pętli głównej algorytmu obliczającego wyrażenie.

| wejście | stos |
|---|---|
| 2 | 2 |
| 3 | 2 3 |
| 4 | 2 3 4 |
| 5 | 2 3 4 5 |
| + | 2 3 9 |
| * | 2 27 |
| + | 29 |

### 1.2.5.Translacja

W praktyce informatycznej często zachodzi konieczność tłumaczenia tekstu zapisanego w jednym języku formalnym na inny język. Jako przykład można przytoczyć przetworzenie tekstu zapisanego w języku wysokiego poziomu na kod maszynowy. Proces tłumaczenia nosi nazwę translacji. Podamy teraz definicję terminu translacja.

### 1.2.6.Definicja:

Translacja to przekład tekstów zredagowanych w jednym języku zwanym źródłowym na równoważny semantycznie tekst w innym języku zwanym wynikowym.

#### *1.2.6.1.Przykład*

Opracować algorytm dokonujący translacji prostych wyrażeń arytmetycznych zapisanych w konwencji nawiasowej na proste wyrażenia zapisane w ONP.

##### 1.2.6.1.1.Rozwiązanie

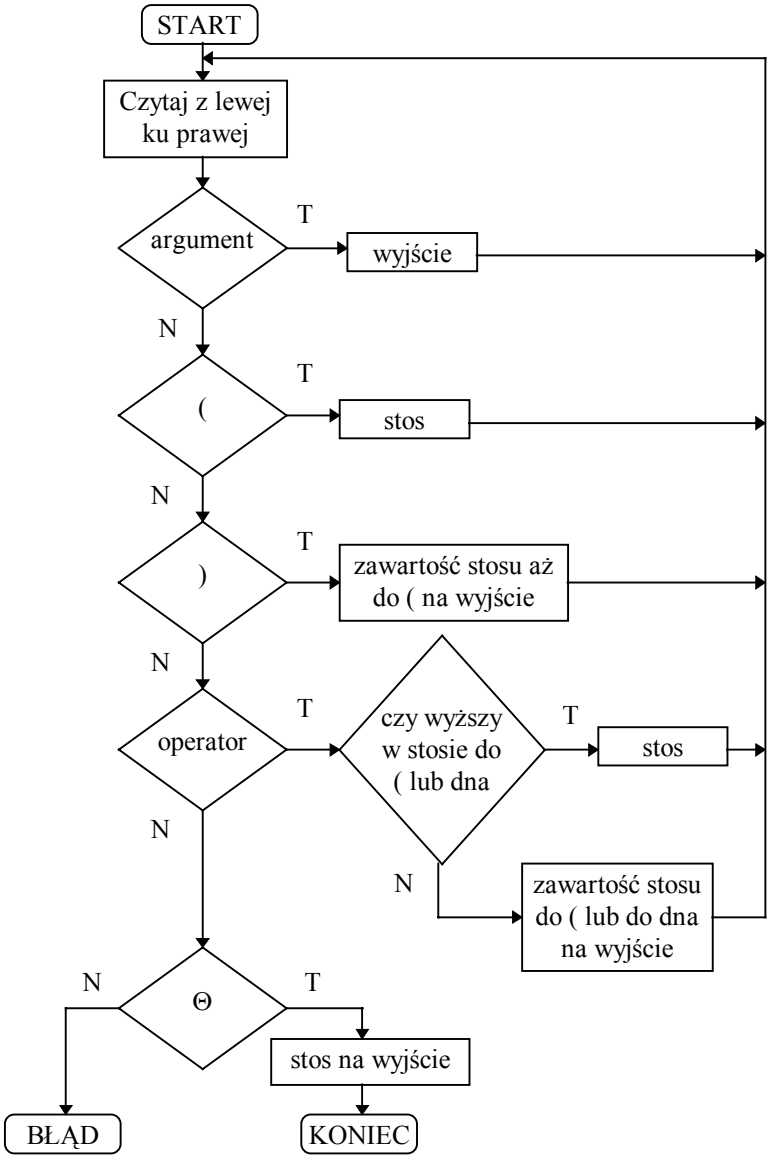
Zagadnieniami kluczowymi w trakcie dokonywania konwersji (translacji) wyrażeń z notacji nawiasowej na ONP są: priorytety operacji arytmetycznych i konwersja wyrażeń w nawiasach. W tym celu należy stworzyć tabelę priorytetów używanych działań arytmetycznych oraz zręcznie posłużyć się stosem. Oto przykładowa tabela priorytetów dla naszego zagadnienia. Przyjmujemy tu założenie, że im cyfra określająca priorytet jest niższa, tym priorytet jest wyższy.

| operacja | priorytet |
|---|---|
| * | 1 |
| + | 2 |

W konwencji nawiasowej priorytet działania oraz nawiasy wpływają na kolejność, w jakiej wyrażenie jest obliczane. Przykładowo, aby wyznaczyć wartość wyrażenia a+b*d obliczamy najpierw iloczyn b*d, a potem dodajemy a. W ONP kolejność wykonywania działań określona jest jedynie poprzez miejsce, w którym operator znajduje się w napisie. Aby dokonać konwersji uwzględniając te założenia, będziemy pobierać kolejne znaki wyrażenia, które ma zostać przekształcone znak po znaku, począwszy od lewej strony. Dalsze postępowanie zależeć będzie od symbolu. I tak jeżeli pobrany znak jest:

- argumentem, to należy przesłać go na wyjście,
- nawiasem otwierającym “(“, to należy położyć go na stos,
- nawiasem zamykającym “)”, to zawartość stosu, aż do napotkanego znaku “(“ należy przesłać na wyjście i usunąć nawias “(“,
- operatorem, to przeglądamy zawartość stosu w poszukiwaniu operatora o wyższym priorytecie: jeśli taki znajdziemy, to przesyłamy zawartość stosu na wyjście, jeśli nie, operator odkładamy na stos. Należy dodać, że stos przeszukujemy do napotkania nawiasu “(“ lub do końca stosu, jeżeli nawias nie występuje,
- znacznikiem końca wprowadzanego napisu, to kopiujemy zawartość stosu na wyjście, w przeciwnym wypadku sygnalizujemy błąd.

Przetwarzanie kończymy w momencie napotkania znacznika końca napisu wejściowego. Nawiasów “(“ i “)” oczywiście nie kopiujemy na wyjście. Narysujemy teraz schemat blokowy algorytmu realizującego przedstawione idee.




**Rysunek 1.2-3**

Spróbujemy na podstawie tego algorytmu dokonać konwersji wyrażenia zapisanego w arytmetyce nawiasowej na ONP.

#### *1.2.6.2.Przykład*

Dokonać konwersji wyrażenia (((a))+b+d)*(d+e) na ONP.

##### 1.2.6.2.1.Rozwiązanie

Całą konwersję zilustrujemy tabelką. Podamy w niej znaki, które kolejno będą pojawiać na wejściu, zawartość stosu oraz to, co będzie wygenerowane na wyjściu.

| Wejście | stos | wyjście |
|---|---|---|
| ( | ( | |
| ( | ( ( | |
| ( | ( ( ( | |
| a | ( ( ( | a |
| ) | ( ( | |
| ) | ( | |
| + | ( + | |
| b | ( + | b |
| + | ( + + | |
| d | ( + + | d |
| ) | | + + |
| * | * | |
| ( | * ( | |
| d | * ( | d |
| + | * ( + | |
| e | * ( + | e |
| ) | * | + |
| | | * |