Wojciech Myszka

Laboratorium 10: „Maszyna stanów”

2016-05-07 09:05:39 +0200

1. Wprowadzenie

Laboratorium poświęcone jest operacjom na napisach (ciągach znaków).
Przypominam, że:

— ’a’ to stała typu **char** o „wartości“ znaku (litery) a małe;
— "Ala␣ma␣kota" to stała typu **char** ∗ (czyli tablica znakowa) zawierająca napis „Ala ma kota”;
— **char** z = ’A’; to deklaracja zmiennej o nazwie *zet* zainicjowanej wartością literą „A wielkie”;
— **char** napis[]="Ala␣ma␣kota"; to deklaracja tablicy znakowej o długości 12 elementów (jedenaście liter i specjalny znak, o kodzie ASCII 0) informujący o końcu napisu; zawartość tablicy **może być** dowolnie modyfikowana, ale nie możemy przekroczyć sumarycznej długości 11 liter (plus znak końca).
— **char** ∗ tekst = "Ala␣ma␣kota"; to deklaracja wskaźnika typu **char** ∗ i zainicjowanie go wartością adresu miejsca pamięci, w którym przechowywana jest **stała znakowa** o wartości „Ala ma kota”; W efekcie jest to też „tablica” o długości 11 znaków (plus znak końca), ale jej zawartość **nie może** być modyfikowana!
— pojawiający się czasami w napisach znak ’␣’ oznacza odstęp.
— na zmiennych znakowych można wykonywać normalne operacje porównywania: ’a’ < ’b’, tekst[0]==’A’, itd.

Funkcja sprawdzająca czy liczba jest odstępem moze wyglądac tak:

```
int czy_odstęp( char x)
{
    return x == ’␣’ || x == ’\n’ || x == ’\t’;
}
```

Funkcja jako odstępy traktuje spacje, znak nowej linii i znak tabulacji.

2. Zadanie do wykonania

Mamy dwa zadania do wyboru:
1. Funkcja sprawdzająca czy zadany ciąg liczb jest liczbą całkowitą.
2. Funkcja sprawdzająca czy zadany ciąg liczb jest liczbą rzeczywistą.

(A dokładniej mówiąc, czy liczba zostanie potraktowana właściwie jako dana wejściowa dla programu napisanego w C, albo czy będzie właściwą stałą odpowiedniego typu[1].)

3. Maszyna stanów

Żeby zadanie zrealizować trzeba najpierw zastanowić się jak wygląda (poprawna) liczba całkowita. Nie jest to skomplikowane:
— Na jej początku jest (być może) pewna liczba odstępów[2] (które nie mają wpływu na poprawność liczby); może je zignorować.
— Później jest (nieobowiązkowy) znak (+−). Po znaku musi wystąpić co najmniej jedna cyfra.
— Po pierwszej cyfrze może wystąpić zero lub więcej cyfr.

Zakładamy, że pierwszy odstęp (po cyfrach) kończy liczbę. Kończy ją również znak o kodzie ASCII 0 (zero).

Można sobie życie ułatwić (albo utrudnić) budując maszynę Turinga, która mając na taśmie napis przejrzy go i zdecyduje czy napis jest liczbą czy też nie.

Program nawiązuje do maszyny Turinga. Szerzej opisywana ona była na zajęciach z Technologii Informacyjnych.

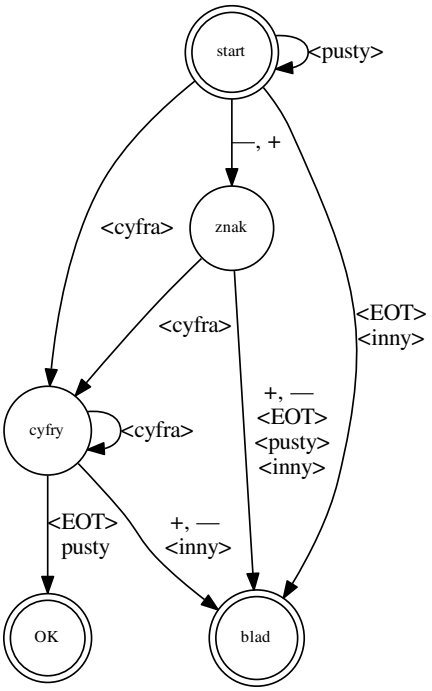
Możemy skonstruować maszynę Turinga realizującą algorytm sprawdzania czy napis jest liczbą, zgodnie z powyższymi zasadami. Wyglądać ona będzie tak jak na rysunku 1.

Zakładamy, że liczba zapisana jest na taśmie, a głowica znajduje się przed pierwszym znakiem napisu.

Na diagramie tym, podwójne okręgi oznaczają stan początkowy i stany końcowe. Opisy nad krawędziami (lab na prawo od nich) oznaczją „kategorie" znaków, które mogę się pojawić:
— +, − to (ewentualny) znak liczby;
— <pusty> oznacza tak zwany „biały znak" (czyli w najprostszym przypadku jest to odstęp, w sytuacjach bardziej skomplikowanych może to być nowa linia,

[1] Ale rozważamy wyłącznie typy podstawowe!

[2] Jako odstępy uważać będziemy dowolne „znaki białe" (to znaczy takie, które nie zostawiają na papierze żadnego czarnego śladu). Będą to odstęp, znak tabulacji (\t), znak nowej linii (\n) i być może jeszcze jakieś...



Rysunek 1. Diagram (uproszczonej) maszyny Turinga sprawdzający czy napis jest liczbą

znak poziomej lub pionowej tabulacji, wysów strony). Wszystkie takie znaki **przed** rozpoczęciem cyfry mogą być po cichu zignorowane;

— <cyfra> to dowolna cyfra z zakresu od 0 do 9;

— <EOT> to umowny (symboliczny) zapis znaku o kodzie ASCII zero, oznaczający w języku C koniec napisu;

— <inny> to (ponownie umowny) zapis oanaczajacy znaki, które nie zostały jeszcze rozpatrzone.

Nie gwarantuję, oczywiście, że przedstawione schematy są poprawne.

Każda ścieżka od **start** od **OK** powinna opisywać poprawną postać liczby.

Można opisać podobne zasady opisujące wygląd liczby rzeczywistej:

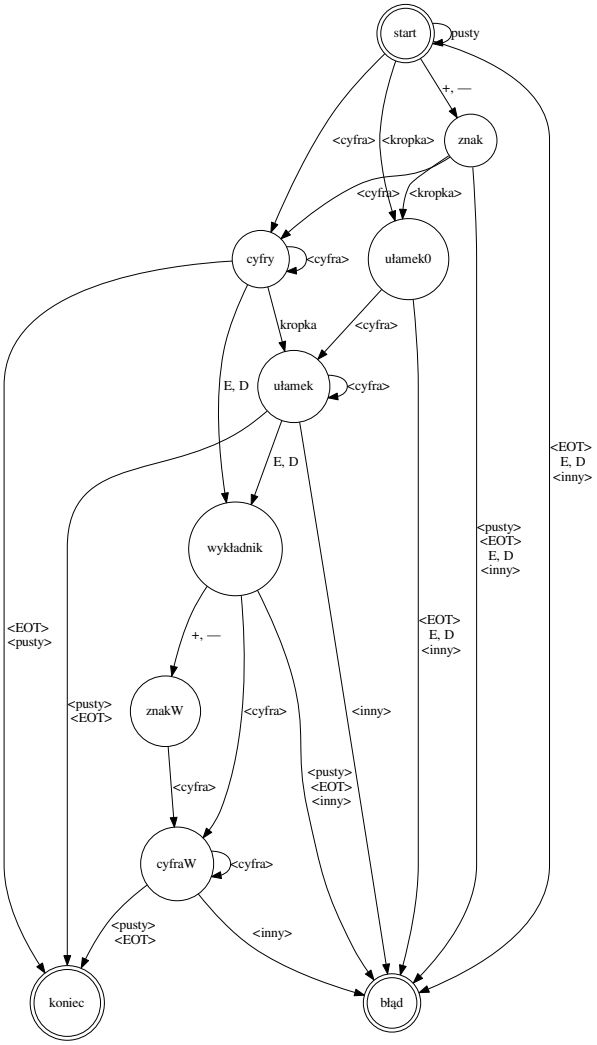
— ignorujemy znaki puste na początku;

— później może być +/–/cyfra,

— pomiędzy cyframi może pojawić się kropka (lub przecinek) dziesiętna,

— po cyfrach może pojawić się litera e (lub E) a po niej plus albo minus albo odstęp, a później cyfra/cyfry wykładnika.

Uproszczony (choć niewolny zapewne od błędów) diagram maszyny Turinga sprawdzającej czy napis jest liczbą rzeczywistą przedstawia rysunek 2.



Rysunek 2. (Uproszczony) diagram przejść maszyny Turinga sprawdzającej czy napis jest liczbą rzeczywistą

Wybieramy jedno z tych zadań i programujemy. Wszystkie czynności sprawdzające powinna wykonywać funkcja (na potrzeby instrukcji nazwę ją czy, wywoływana w sposób następujący:

```
int czy(char napis[ ]);
```

na przykład:

```
if(czy("+123") printf("Liczba!\n");
```

Funkcja zwraca wartość jeden gdy napis jest liczbą, zero w przeciwnym razie.

Można próbować utrudnić/ułatwić sobie życie programując szereg funkcji pomocniczych sprawdzających czy:
— znak jest cyfrą (**int** czy_cyfra(**char**);)
— jest znakiem (+ −) **int** czy_znak(**char**);
— odstępem,
— ...

4. Uwagi

Uwaga 1: Można zaprogramować funkcję bardziej rozbudowaną, która będzie poprawnie rozpoznawała jako liczby stałe dwójkowe, ósemkowe czy szesnastkowe...

Uwaga 2: Na koniec zajęc należy wysłać na adres e-mail prowadzacego zadanie w takiej postaci do jakiej uda się je doprowadzić...

Uwaga 3: Można zaprogramować oba!

Uwaga 4: Można pokusić się o rozpoznawanie liczb całkowitych zapisanych ósemkowo (zaczynają się od cyfry 0), później mogą być tylko cyfry z zakresu 0–7 i/lub szesnastkowo.

5. Wersja PDF tego dokumentu...

...pod adresem.

Wersja: 50 z drobnymi modyfikacjami! data ostatniej modyfikacji 2016-05-07 09:05:39 +0200