W poprzednim [rozdziale](https://eduinf.waw.pl/inf/alg/006_bin/0002.php) podaliśmy algorytm obliczania wartości liczby na podstawie [wzoru](https://eduinf.waw.pl/inf/alg/006_bin/0002.php#wzor):

https://eduinf.waw.pl/inf/alg/006_bin/images/0003.p1.gif

W zastosowaniach informatycznych korzysta się z innego rozwiązania, zwanego **schematem Hornera**. Właściwie schemat ten ma zastosowanie przy wyznaczaniu wartości wielomianu, lecz jeśli przyjrzymy się dokładnie powyższemu wzorowi, to na pewno zauważymy podobieństwo do wzoru na wartość wielomianu:

https://eduinf.waw.pl/inf/alg/006_bin/images/0003.p2.gif

Współczynniki *ai*dla*i* = 0,1,2,...,n-1odpowiadają wartościom cyfr *c*. Natomiast kolejne potęgi zmiennej *x* to oczywiście potęgi podstawy *p*. Schemat Hornera wyznaczymy dla 5-cio cyfrowej liczby (dla *n*-cyfrowej zasada jest identyczna, jednakże uczniowie gubią się w rachunkach). Liczba zapisana jest w systemie pozycyjnym o podstawie *p* ciągiem cyfr *c*4*c*3*c*2*c*1*c*0 i ma wartość:

https://eduinf.waw.pl/inf/alg/006_bin/images/0003.p3.gif

Ponieważ *p1* = *p* oraz *p0 = 1*, powyższy wzór można nieco uprościć i zapisać go w postaci:

https://eduinf.waw.pl/inf/alg/006_bin/images/0003.p4.gif

Wyprowadzamy przed nawias wspólny czynnik *p*:

https://eduinf.waw.pl/inf/alg/006_bin/images/0003.p5.gif

Zwróć uwagę, iż wyrażenie w nawiasie ma niższy stopień. Znów wyprowadzamy przed nawias wspólny czynnik *p*.

https://eduinf.waw.pl/inf/alg/006_bin/images/0003.p6.gif

I jeszcze raz:

https://eduinf.waw.pl/inf/alg/006_bin/images/0003.p7.gif

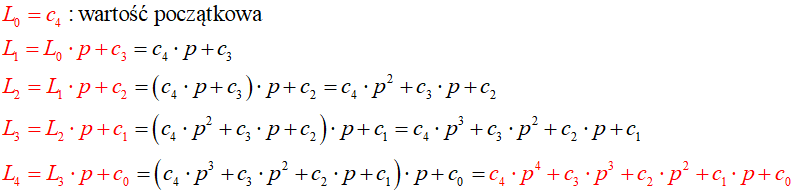
I po raz ostatni:

https://eduinf.waw.pl/inf/alg/006_bin/images/0003.p8.gif

Ze względu na przemienność operacji mnożenia otrzymany wzór możemy zapisać w postaci:

https://eduinf.waw.pl/inf/alg/006_bin/images/0003.p9.gif

Teraz wartość liczby obliczamy wyliczając wartości wyrażeń w kolejnych nawiasach:



Zwróć uwagę na sposób wyliczania wartości liczby. Wyraźnie widoczny jest pewien schemat postępowania. Najpierw za wartość liczby przyjmujemy *c*4. Następnie do wyczerpania pozostałych cyfr wykonujemy te same obliczenia: nową wartość otrzymujemy mnożąc poprzednią wartość przez podstawę systemu i dodając kolejną cyfrę. Rachunki kończymy po dodaniu ostatniej cyfry zapisu liczby.

Schemat ten nosi nazwę **schematu Hornera**.

**Przykład**:

Obliczyć za pomocą schematu Hornera wartość liczby piątkowej 4223213(5).

|  |
| --- |
| L0 = 4 L1 = 4 · 5 + 2 = 22 L2 = 22 · 5 + 2 = 112 L3 = 112 · 5 + 3 = 563 L4 = 563 · 5 + 2 = 2817 L5 = 2817 · 5 + 1 = 14086 L6 = 14086 · 5 + 3 = **70433** - koniec, ponieważ wyczerpaliśmy wszystkie cyfry  4223213(5) = 70433(10). |

**Zalety schematu Hornera**

Po co to wszystko jest potrzebne? Po pierwsze oszczędność w mnożeniu. Sprawdźmy. Dla pięciu cyfr musimy wykonać następujące rachunki przy zastosowaniu standardowego wzoru:

https://eduinf.waw.pl/inf/alg/006_bin/images/0003.p11.gif

Daje to w sumie 10 mnożeń i 4 dodawania. Ten sam rachunek schematem Hornera prowadzi do wykonania 4 mnożeń i 4 dodawań. Mniej mnożeń oznacza większą efektywność algorytmu Hornera, ponieważ mnożenie zajmuje procesorowi komputera więcej czasu od dodawania. Drugą zaletą jest sposób przetwarzania cyfr. Bierzemy je kolejno jedna po drugiej z ciągu wejściowego aż do napotkania końca zapisu. Ponieważ taka kolejność cyfr jest zwykle zgodna z kolejnością ich przechowywania w łańcuchu tekstowym, zatem sposób ten daje nam kolejne przyspieszenie i uproszczenie działania algorytmu (w poprzednim algorytmie cyfry przetwarzaliśmy w kierunku odwrotnym poczynając od ostatniej w zapisie).

## Algorytm

### Specyfikacja problemu

#### Dane wejściowe

|  |
| --- |
| *p*- podstawa systemu pozycyjnego zapisu liczby *p* ∈ N,   *p* ∈ {2,3,...,10} |
| *s*- tekst zawierający ciąg znaków ASCII przedstawiających poprawny zapis liczby. |

#### Dane wyjściowe

Liczba *L* będąca wartością liczby o podstawie*p* i zapisanej w postaci ciągu znaków *s. L* ∈ N + {0}

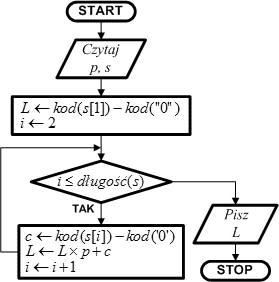
#### Zmienne pomocnicze i funkcje

|  |  |
| --- | --- |
| *i* | - numery pozycji znaków w *s, i* ∈ N |
| *c* | - przechowuje wartość cyfry, *c* ∈ N + {0} |
| kod(*znak*) | - funkcja zwraca kod ASCII znaku |
| długość(*tekst*) | - zwraca liczbę znaków zawartych w tekście |

### Lista kroków

|  |  |
| --- | --- |
| K01: | **Czytaj** *p* i *s* |
| K02: | *L* ← kod(*s*[1]) - kod("0") |
| K03: | **Dla** *i* = 2,3,...,długość(*s*):     **wykonuj kroki** K04...K05 |
| K04: | *c* ← kod(*s*[*i*]) - kod("0") |
| K05: | *L* ← *L* · *p* + *c* |
| K06: | **Pisz** *L* |
| K07: | **Zakończ** |

### Schemat blokowy



Odczytujemy podstawę *p* systemu liczbowego, w którym zapisana jest liczba. Podstawa musi należeć do zakresu od 2 do 10. Następnie odczytujemy ciąg znaków *s* reprezentujących cyfry. W zmiennych łańcuchowych pozycje znaków są numerowane od 1 (w C++, Pythonie i JavaScript od 0) począwszy od strony lewej do prawej.

Po odczytaniu danych wejściowych inicjujemy zmienne robocze. Początkowa wartość liczby *L* ustawiana jest na wartość pierwszej cyfry zapisu. Zmienna i steruje pętlą iteracyjną. Wprowadzamy do niej indeks drugiego znaku odczytanego łańcucha *s* i rozpoczynamy pętlę.

Pętla wykonuje się o jeden raz mniej niż liczba znaków w łańcuchu *s*. W pętli wyznaczamy wartość kolejnej cyfry w *c*.  Za nową wartość liczby L przyjmujemy poprzednią wartość pomnożoną przez *p* i zwiększoną o wartość cyfry *c* (schemat Hornera). Zwiększamy indeks i przechodzimy na początek pętli.

Po zakończeniu pętli w zmiennej *L* mamy wartość liczby. Wypisujemy ją i kończymy algorytm.