## Zajęcia 7

**Temat:** Łańcuchy znaków (proste)

**Czas trwania:** 2x45 min

**Cel zajęć:**

projektuje i programuje proste problemy z różnych dziedzin, stosuje przy tym: instrukcje wejścia/wyjścia, wyrażenia arytmetyczne i logiczne, instrukcje warunkowe, instrukcje iteracyjne, tablice, łańcuchy znaków, testuje poprawność programów dla różnych danych, posługuje się zintegrowanym środowiskiem programistycznym przy pisaniu, uruchamianiu i testowaniu programów;

**Efekty:**

- umie uruchomić potrzebne oprogramowanie,
- umie napisać program z wykorzystaniem tablic ii łańcuchów znaków w C++,
- potrafi zaimplementować własną arytmetykę dużych liczb.
- zna algorytm przeszukiwania tekstu,
- potrafi dokonać zamiany pomiędzy różnymi systemami liczbowymi,
- zna schemat Hornera

**Formy i metody pracy:** praca samodzielna,

| Zadania do wykonania na zajęciach | Treści programowe |
|---|---|
| 1. Brakująca cyfra | M.3, P.2.15, A.3.3 |
| 2. Ciąg monotoniczny | M.3, P.2.15, A.3.2 |
| 3. System 36 | M.3, P.2.15, A.3.1, A.3.2 |
| 4. James i potęgi dwójki | M.3, P.2.15, A.3.2 |

**Materiały do zajęć:**

https://www.main2.edu.pl/main2/courses/show/6/17/

**Zadania do wykonania w domu:**

**Statystyki (OIG):**

https://szkopul.edu.pl/problemset/problem/XAJ_VTHmix0ioDPo5ygaq1sc/site/?key=statement

**Sumujący Jaś (OIG):**

https://szkopul.edu.pl/problemset/problem/Tczhl-p0p4d8QI5QKSByWTME/site/?key=statement

# ZADANIA I ROZWIĄZANIA

## Zadanie 1. Brakująca cyfra

Dostępna pamięć: 64MB

Mamusia ciągle powtarzała Jankowi: ‘Odrabiając pracę domową z matematyki nie pij nad zeszytem mleka!’. Janek oczywiście nie słuchał mamy i mleko nad zeszytem pił. Aż do czasu, kiedy mleko rozlało się na zadanie domowe! Całe szczęście, że rozmyła się tylko jedna cyfra w całej liczbie! Janek pamięta, że każda liczba na kartce była podzielna przez 9. Twoim zadaniem jest odnalezienie brakującej cyfry. Jeśli warunek spełnia kilka cyfr, pomóż odnaleźć Jankowi najmniejszą z nich.

### Wejście

Pierwszy wiersz wejścia zawiera jedną liczbę k z pracy domowej($1 \le k \le 10^{1000}$). Zagubiona cyfra zastąpiona jest znakiem ‘X’. Możesz przyjąć, że dla 40% testów zachodzi warunek $k \le 10^{18}$.

### Wyjście

Na wyjściu wypisz brakującą cyfrę.

### Przykład

| Wejście | Wyjście |
|---|---|
| 23X54 | 4 |

### Rozwiązanie

Rozwiązanie zadania wymaga omówienia podstawowych informacji związanych z obsługą łańcuchów znaków (typ char, string, rzutowanie typów).

Zauważmy, że liczba jest podzielna przez 9, jeżeli suma jej cyfr jest podzielna przez 9. Naszym zadaniem jest zatem obliczenie sumy cyfr bez X i wypisanie brakującej wartości do 9. Wyjątek stanowią sytuacje, gdy suma jest równa 9. Wówczas, gdy X umieszczono na pierwszym miejscu, wypiszemy 9, w każdym innym wypadku 0.

```
wczytaj s
suma ← 0
dla i=0 do długość(s)-1 wykonaj
     jeżeli s[i] ≠ 'X'
          suma ← suma + s[i] - 48
brakująca_cyfra ← 9 - suma mod 9
jeżeli brakująca_cyfra = 9 i s[0] ≠'X'
     brakująca_cyfra ← 0
wypisz brakująca_cyfra
```

## Zadanie 2. Ciąg monotoniczny

Dostępna pamięć: 32MB

Ciąg liczbowy nazywamy monotonicznym jeżeli jest rosnący, albo malejący, albo stały.

W naszym zadaniu dany jest ciąg liter. Jakie litery należy w nim zmienić, aby nasz ciąg był stały, zastępując przy tym jak najmniej znaków? Jeżeli jest kilka takich znaków, które można pozostawić, nie zmieniaj alfabetycznie największej z nich. Możesz założyć, że zawsze będzie co najmniej jedna litera do zmiany.

### Wejście

W pierwszej linii wejścia znajduje się ciąg co najmniej dwóch liter (wyłącznie małe litery alfabetu łacińskiego) nie dłuższy niż $10^6$ elementów.

### Wejście

Na wyjściu w pierwszym wierszu wypisz alfabetycznie wszystkie litery do zmiany.

### Przykład

| Wejście | Wyjście |
|---|---|
| `kajak` | `aaj` |

Wyjaśnienie: Należy pozostawić litery ‘k’.

### Rozwiązanie

Rozwiązanie zadania wymaga omówienia podstawowych informacji związanych z obsługą łańcuchów znaków (typ char, string, rzutowanie typów).

W zadaniu należy zliczyć wystąpienia liter, odszukać ostatnią najczęściej występującą i wypisać każdą literę zgodnie z liczbą jej wystąpień (patrz algorytm: sortowanie przez zliczanie).

```
wczytaj s
dla i=0 do długość(s) wykonaj
     ile_wystąpień[ s[i] ] ← ile_ wystąpień[ s[i] ] + 1
najczęściej_wyst ← 0
dla i=’a’ do ’z’ wykonaj
     jeżeli ile_wystąpień[i] ≥ ile_wystąpień[ najczęściej_wyst ]
          najczęściej_wyst ← i
dla i=’a’ do ’z’ wykonaj
     jeżeli najczęściej_wyst ≠ i
          dla j=0 do ile_wystąpień[i]
               wypisz znak i
```

## Zadanie 3. System 36

Dostępna pamięć: 32MB

Bitek sprawdza przydatność obliczeń w różnych systemach obliczeń. Stwierdził, że korzystając z cyfr i dużych liter alfabetu łacińskiego może korzystać nawet z systemu trzydziestoszóstkowego! Próbuje teraz napisać program, który szybko przeliczałby wartości pomiędzy dowolnymi systemami liczbowymi.

### Wejście

Pierwszy wiersz zawiera trzy liczby: X, Y i Z, gdzie X jest liczbą zapisaną w systemie o podstawie Y (wartość X nie przekracza $10^{15}$ w systemie dziesiętnym). Z jest podstawą systemu, na który należy zamienić liczbę X ($1 < Y, Z < 37$).

### Wyjście

Jedna liczba całkowita - zapis X w systemie o podstawie Z.

### Przykład

| Wejście | Wyjście |
|---|---|
| 37826876 10 36 | MIREK |

### Rozwiązanie

Rozwiązanie zadania wymaga omówienia podstawowych informacji związanych z zamianą liczb pomiędzy poszczególnymi systematami pozycyjnymi oraz schematu Hornera.

Najprostszym rozwiązaniem jest zamiana wskazanego tekstu zapisanego w systemie o podstawie Y na jego reprezentacją w systemie dziesiętnym (z użyciem schematu Hornera).

```
wczytaj X, Y, Z
dziesiętnie ← 0
dla i=0 do długość(X) wykonaj
      dziesiętnie ← dziesiętnie ·Y + s[i]-48
```

Następnie należy zamienić otrzymaną wartość na system o podstawie Z (zmienna tekstowa `wynik`). Zakładamy, że instrukcja `(znak) wartość` rzutuje wartość całkowitoliczbową na jej znak w kodzie ASCII. Łączenie tekstów pozwala na umieszczanie nowo obliczonych cyfr na początku zapisu liczby.

```
wynik ← ""
dopóki dziesiętnie ≠ 0 wykonuj
     jeżeli dziesiętnie mod Z < 10
          c ← (znak) (dziesiętnie mod Z + '0')
     w przeciwnym wypadku
          c ← (znak) (dziesiętnie mod Z + 'A')
     wynik ← c + wynik
     dziesiętnie ← dziesiętnie div Z
wypisz wynik
```

## Zadanie 4. James i potęgi dwójki

Dostępna pamięć: 64MB

To był już prawie koniec misji. James Blond rozejrzał się uważnie po pokoju. Pomiędzy porozrzucanymi kartkami znajdowała się TA JEDNA, poszukiwana przez wszystkich. „Jak mogli jej nie zauważyć?”- zapytał sam siebie. Schylił się i wszystko zrozumiał. Na przedartej stronie widać było krótkie ciągi liczb. Ich końcowe cyfry musiały znajdować się na brakującej większej części. W panice zaczął przegarniać papiery. „Nie ma! Nie ma!” – krzyczał do siebie bezgłośnie. Usiadł bezsilnie wpatrując się ciągi znaków. I wtedy… przypomniał sobie ostatnie słowa Profesora: „Tylko potęgi dwójki!”. A więc o to mu chodziło! Każda z liczb to potęga dwójki! Wystarczy znaleźć najmniejszą potęgę dwójki, której początkowe cyfry widział na kartce! James wyjął swój smartphone, włączył aplikację kalkulatora w widoku programisty i… Nie mógł uwierzyć! Android się zawiesił! Czy coś mu jeszcze może pomóc?

### Zadanie

Napisz program, który dla zadanej dodatniej liczby całkowitej n wyznaczy najmniejszy wykładnik X taki, że pierwsze cyfry liczby $2^X$ zgadzają się z zadaną liczbą. Pamiętaj, że oderwano większą część kartki, więc na pewno brakuje ponad połowy cyfr.

### Wejście

W pierwszym wierszu wejścia znajduje się jedna liczba całkowita bez znaku n ($1 \le n < 10^{50}$).

### Wyjście

Dla podanej n wyznacz wykładnik X taki, że pierwsze cyfry liczby $2^X$ zgadzają się z zadaną liczbą. Możesz założyć, że wynik zawsze będzie istniał i $2^X \le 10^{100}$. Możesz założyć, że dla 40% testów zachodzi warunek $2^X < 10^{18}$.

### Przykład

| Wejście<br>1<br>Wyjście<br>7 | Wejście<br>5<br>Wyjście<br>9 | Wejście<br>16<br>Wyjście<br>14 |
|---|---|---|

### Rozwiązanie

Rozwiązanie zadania wymaga zaimplementowania własnej arytmetyki dużych liczb (dodawania), rzutowania typów i przeszukiwania tekstu.

W zadaniu liczymy kolejne potęgi dwójki. Będziemy więc po prostu dodawać je do siebie i naiwnie sprawdzać zgodność kolejnych znaków, stąd liczbę n wczytujemy jako tekst. Zauważmy, że najmniejszą potęgą, którą możemy wypisać, jest $2^7$ (musi to być liczba co najmniej 3-cyfrowa w zapisie dziesiętnym). Wygodnie będzie więc zacząć od $2^6$. Zakładamy, że instrukcja `(znak) wartość` rzutuje wartość całkowitoliczbową na jej znak w kodzie ASCII.

```
wczytaj n
potęga ← ”64”
X ← 6
wartownik ← 0
```

```
dopóki wartownik = 0 wykonuj
      // obliczamy nową potęgę
      // dodając do siebie cyfry począwszy od ostatniej
      X ← X + 1
      c ← 0 // kolejne wyznaczane cyfry
      dla i=długość(n)-1 do 0
            c ← 2 ·(potęga[i]-48) + c
            potęga [i] ← (znak) (c mod 10 + ’0’)
            c  ← c div 10
      // po dodaniu wszystkich cyfr został nam bit przeniesienia
      jeżeli c > 0
            potęga ← ’1’ + potęga
      //sprawdzamy zgodność początkowych cyfr
      jeżeli długość(potęga) > 2 ·długość(n)
            wartownik ← 1
            dla i=0 do długość(n)-1 wykonuj
                  jeżeli n[i] ≠ potęga[i]
                        wartownik ← 0
wypisz X
```