Najmniejsza liczba (rozwiązanie)

Autor zadania: **Karol Pokorski**
Opracowanie: **Paweł Anikiel, Michał Woźny**
Opis rozwiązania: **Bartosz Kostka**

Olimpiada Informatyczna Juniorów

W zadaniu tym mamy dane trzy cyfry X, Y i Z i naszym zadaniem jest z tych cyfr stworzyć najmniejszą możliwą liczbę trzycyfrową. Musimy jednak zwrócić uwagę na to, że liczba ta nie może zaczynać się zerami (zawierać zer wiodących).

Rozwiążmy najpierw prostszą wersję, w której nie mamy zer (za taką wersję zadania można było zdobyć 49 punktów). Mamy zatem $X,Y,Z \geq 1$. Dość naturalne wydaje się, że aby otrzymać najmniejszą liczbę, wystarczy te liczby posortować niemalejąco. Rzeczywiście: na początku musi stać najmniejsza cyfra (bo postawienie tam innej na pewno dałoby gorszy wynik), a druga najmniejsza z tego samego powodu musi stać w środku.

Możemy zatem po prostu sprawdzić kolejność cyfr X, Y, Z za pomocą kilku instrukcji warunkowych i wypisać odpowiedni wynik. Prowadzi to jednak do dość długiego kodu – zaznaczmy w tym miejscu, że na dłuższą metę warto unikać takiego stylu programowania, jako że jest trudno czytelny i podatny na błędy.

lic_bez_zer.cpp

```cpp
#include <iostream>

using namespace std;

int main() {
  // Deklarujemy zmienne.
  int x, y, z;
  // Wczytujemy cyfry z wejścia.
  cin >> x >> y >> z;
  // Sprawdzamy wszystkie możliwe kolejności cyfr
  // i wypisujemy odpowiedni wynik dla tej kolejności.
  if (x <= y and y <= z) cout << x << y << z << "\n";
  else if (x <= z and z <= y) cout << x << z << y << "\n";
  else if (y <= x and x <= z) cout << y << x << z << "\n";
  else if (y <= z and z <= x) cout << y << z << x << "\n";
  else if (z <= x and x <= y) cout << z << x << y << "\n";
  else cout << z << y << x << "\n";
}
```

lic_bez_zer.py

```python
def main():
  # Wczytujemy cyfry z wejścia.
  (x, y, z) = tuple(map(int, input().split()))
  # Sprawdzamy wszystkie możliwe kolejności cyfr
  # i wypisujemy odpowiedni wynik dla tej kolejności.
  # W języku Python musimy zwrócić uwagę na to, że print
  # domyślnie dodaje spacje pomiędzy kolejnymi wypisywanymi zmiennymi,
  # dlatego używamy argumentu sep, aby pozbyć się tego separatora.
  if x <= y <= z: print(x, y, z, sep='')
  elif x <= z <= y: print(x, z, y, sep='')
  elif y <= x <= z: print(y, x, z, sep='')
  elif y <= z <= x: print(y, z, x, sep='')
  elif z <= x <= y: print(z, x, y, sep='')
  else: print(z, y, x, sep='')

main()
```

Wygodniej posortować podane trzy cyfry korzystając z funkcji `sort` na tablicy, liście bądź wektorze. Przykładowe programy dostępne są poniżej.

Olimpiada finansowana jest ze środków Ministerstwa Edukacji Narodowej w ramach zadania publicznego „Organizacja i przeprowadzenie olimpiad i turniejów w latach szkolnych 2019/2020, 2020/2021, 2021/2022".

lic_bez_zer_sort_tablica.cpp

```cpp
#include <iostream>
#include <algorithm>

using namespace std;

int main() {
  // Deklarujemy tablicę 3-elementową na cyfry.
  int tab[3];
  // Wczytujemy cyfry z wejścia do tablicy.
  cin >> tab[0] >> tab[1] >> tab[2];
  // Sortujemy tablicę.
  sort(tab, tab+3);
  // Wypisujemy kolejno posortowane cyfry.
  cout << tab[0] << tab[1] << tab[2] << "\n";
}
```

lic_bez_zer_sort_wektor.cpp

```cpp
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;

int main() {
  // Deklarujemy vector 3-elementowy na cyfry.
  vector<int> tab(3);
  // Wczytujemy cyfry z wejścia do wektora.
  cin >> tab[0] >> tab[1] >> tab[2];
  // Sortujemy wektor.
  sort(tab.begin(), tab.end());
  // Wypisujemy kolejno posortowane cyfry.
  cout << tab[0] << tab[1] << tab[2] << "\n";
}
```

lic_bez_zer_sort.py

```python
def main():
  # Wczytujemy cyfry z wejścia do listy.
  tab = list(map(int, input().split()))
  # Sortujemy listę.
  tab.sort()
  # Wypisujemy kolejno posortowane cyfry.
  print(tab[0], tab[1], tab[2], sep='')

main()
```

Wróćmy teraz do oryginalnej wersji zadania, w której mamy do czynienia z zerami. W powyższym rozwiązaniu, po posortowaniu cyfr zawsze otrzymamy zera z lewej strony, które nie są dozwolone (na przykład dla cyfr 7 0 0, otrzymamy liczbę 007), musimy więc przypadki z zerem obsłużyć inaczej. W zadaniu mamy jednak zagwarantowane, że co najmniej jedna z cyfr będzie dodatnia, zatem mamy tylko dwa przypadki:

- dokładnie jedna cyfra jest równa 0 – zakładając, że dwie pozostałe cyfry A i B spełniają $A \leq B$, łatwo sprawdzić, że $\overline{A0B}$ będzie najmniejszą dozwoloną liczbą ($AB0$, $B0A$ i $BA0$ są co najmniej tak samo duże).
- dwie cyfry są zerami – jedyną dozwoloną liczbą jest $\overline{C00}$, gdzie C jest niezerową cyfrą.

Aby uzyskać krótki kod programu, możemy lekko zmodyfikować powyższe rozwiązania z sortowaniem, by działały także w przypadkach z zerami:

Olimpiada finansowana jest ze środków Ministerstwa Edukacji Narodowej w ramach zadania publicznego „Organizacja i przeprowadzenie olimpiad i turniejów w latach szkolnych 2019/2020, 2020/2021, 2021/2022".

- Jeżeli jedna cyfra będzie równa zero (po posortowaniu będzie to ta cyfra z lewej strony), to wystarczy że ją wymienimy z drugą cyfrą w kolejności. Na przykład z 023 stworzymy 203.
- Jeżeli dwie cyfry będą równe zero, to wymienimy ostatnią cyfrę (niezerową) z pierwszą. Dla przykładu 007 zamienimy na 700.

Dodajemy zatem powyższe dwie instrukcje do kodu z sortowaniem i otrzymujemy 100 punktów za to zadanie:

lic.cpp

```cpp
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;

int main() {
  // Deklarujemy vector 3-elementowy na cyfry.
  vector<int> tab(3);
  // Wczytujemy cyfry z wejścia do wektora.
  cin >> tab[0] >> tab[1] >> tab[2];
  // Sortujemy wektor.
  sort(tab.begin(), tab.end());
  // Jeżeli dwie pierwsze cyfry to 0, to zamień miejscami
  // pierwszą i ostatnią cyfrę.
  if (tab[0] == 0 and tab[1] == 0) swap(tab[0], tab[2]);
  // W przeciwnym wypadku, jeżeli tylko jedna cyfra jest równa 0,
  // to zamień miejscami pierwszą i drugą cyfrę.
  else if (tab[0] == 0) swap(tab[0], tab[1]);
  // Wypisujemy kolejno posortowane cyfry.
  cout << tab[0] << tab[1] << tab[2] << "\n";
}
```

lic.py

```python
def main():
  # Wczytujemy cyfry z wejścia do listy.
  tab = list(map(int, input().split()))
  # Sortujemy listę.
  tab.sort()
  # Jeżeli dwie pierwsze cyfry to 0, to zamień miejscami
  # pierwszą i ostatnią cyfrę.
  if tab[0] == 0 and tab[1] == 0:
    (tab[0], tab[2]) = (tab[2], tab[0])
  # W przeciwnym wypadku, jeżeli tylko jedna cyfra jest równa 0,
  # to zamień miejscami pierwszą i drugą cyfrę.
  elif tab[0] == 0:
    (tab[0], tab[1]) = (tab[1], tab[0])
  # Wypisujemy kolejno posortowane cyfry.
  print(tab[0], tab[1], tab[2], sep='')

main()
```


Olimpiada finansowana jest ze środków Ministerstwa Edukacji Narodowej w ramach zadania publicznego „Organizacja i przeprowadzenie olimpiad i turniejów w latach szkolnych 2019/2020, 2020/2021, 2021/2022".