





Scenariusz zajęć nr 11 Temat: Binarny podatek

Cele:

W trakcie zajęć uczeń zapozna się z następującymi pojęciami:

- Ciągi znaków (analiza zawartości),
- Zastosowanie reguł przekształcania ciągów,
- Obsługa przypadków brzegowych (ang. edge case).

Wstęp:

Uruchomienie środowiska programistycznego, zapoznanie się z edytorem kodu źródłowego i sposobem kompilacji programu (kontynuacja).

Dla nauczyciela:

W razie braku zainstalowanego środowiska można skorzystać z dowolnego środowiska dostępnego w sieci Internet, na przykład cpp.sh.

Przebieg zajęć:

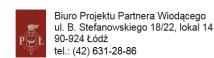
Zapoznanie się z treścią zadania:

Treść zadania jest dołączona do scenariusza zajęć w formie pliku cpp_11-podatek.pdf.

Dla nauczyciela:

Tematem zajęć jest zadanie *Binarny podatek* dostępne na serwisie szkopul.edu.pl, wzorowane na zadaniu *Minimum Binary Number* (976A) z serwisu codeforces.com. Zadanie można rozwiązać w dowolnym języku programowania, a jego stopień trudności należy ocenić jako właściwy dla uczestników podstawowego kursu programowania.











Wskazówki do rozwiązania zadania:

Zadanie to pozornie wygląda na dość trudne: trzeba wszakże zastosować abstrakcyjne reguły zamiany elementów ciągu znaków. Jednak jeśli przyjrzymy się uważnie drugiej regule, wtedy zobaczymy, że prowadzi ona do zredukowania wszystkich cyfr 1 do jednej (o ile w ogóle one występowały). Nie należy się martwić, że początkowo jedynki mogą nie stać koło siebie – reguła pierwsza pozwala dowolnie zmieniać kolejność cyfr. Natomiast nie jesteśmy zdolni w żaden sposób usunąć żadnego zera (o ile takowe występują). Wynikowa liczba nie może zawierać zer wiodących, zatem powinna mieć postać: 100...0, gdzie ilość zer musi być taka, jak w wejściowej liczbie.

Jest jeden wyjątek: kiedy wejściowa liczba nie zawiera żadnych jedynek, wtedy po prostu należy zwrócić jednocyfrową liczbę 0.

Kod programu w języku C++:

```
#include<iostream>
using namespace std;
void kwota()
{
  int z = 0, j = 0;
  string s; cin >> s;
  for(size_t i = 0; i < s.size(); i++)</pre>
    if(s[i] == '1') ++ j;
    else ++z;
  if(j) cout << 1;
  while(z--) cout << 0;
  cout << endl;</pre>
}
int main()
  int n; cin >> n;
  while(n--)
    kwota();
  return 0;
}
```

Zmienna z oznacza ilość zer, a zmienna j – ilość jedynek.











Podsumowanie i uwagi końcowe:

Ważne jest podkreślenie uczniom, że wcale nie musimy zastanawiać się, jak po kolei zachodziły zmiany – zresztą nie można tego określić w sposób jednoznaczny. Istotny jest tylko efekt końcowy – jak najmniejsza kwota podatku.

Często zapomina się, że przy tego rodzaju przetwarzaniu danych należy po każdym wyniku wypisać znak końca wiersza, inaczej wynik programu będzie błędny.



