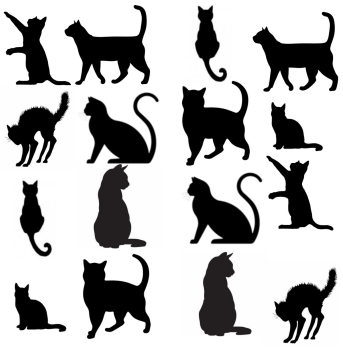
Kot szesnastkowy



#układ_szesnastkowy
#const #isdigit()

Szesnastkowy system liczbowy jest popularny w wielu zastosowaniach informatycznych, bo wiele rzeczy (na przykład adresy w pamięci komputera) wyraża się właśnie w ten sposób. Nikt nikogo nie zmusza, aby się tego uczyć, ale naprawdę warto znać podstawy, gdy ktoś podczas impry rzuci jakimś *hex*-em.* Po prostu nie wypada tego nie znać.

Popularny *hex* to liczbowy system pozycyjny o podstawie równej $2^4 = 16$. Jeśli podstawa układu przewyższa 10, to znaczy, że braknie nam zwykłych arabskich cyfr na przedstawienie postaci liczby. Zwyczajowo używa się tu początkowych liter alfabetu łacińskiego od **a** do **f**, przy czym **a** oznacza „cyfrę" 10, zaś **f** to 15.† Da się wytrzymać.

Ten program będzie właściwie kalką z programu o układzie binarnym, z jednym wyjątkiem. Zestaw cyfr generowanych przez funkcję `hexadecimal()` (wymawiaj: *heksadesimal*, z akcentem na drugie *e*) jest dość szeroki, więc najlepiej zadeklarować stałą typu `string`, która zawiera wszystkie możliwe cyfry:

```
const string H{ "0123456789abcdef" };
```

Teraz definicja odpowiedniej funkcji pójdzie już gładko:

```
string hexadecimal(int n)
{
  const string H{ "0123456789abcdef" };
  string result;
  do
  {
    result = H[n % 16] + result;
    n /= 16;
  }
  while(n > 0);
  return result;
}
```

Takie to proste, prawda?

*Zwłaszcza podczas melanżu uczestników OI-a (wymawiaj: *oja*).
†Czasem stosuje się wielkie litery, ale nie róbmy z tego problemu.

A w drugę stronę?

Napiszemy teraz funkcję zamieniającą liczbę w postaci szesnastkowej na postać dziesiętną. Potrzebna jest tu odrobina ostrożności, gdyż zbiór cyfr szesnastkowych składa się z dwóch rozłącznych podzbiorów znaków niesąsiadujących ze sobą w alfabecie ASCII. Ale damy radę: skorzystamy znowu z metody Hornera. Funkcję nazwiemy `decimal_from_hex()` – jej argument będzie typu `string`, zaś rezultat – typu `int`:

```
int decimal_from_hex(string s)
{
  int w{ };
  for(char c : s)
    . . .
}
```

W treści pętli powinna znaleźć się instrukcja warunkowa sprawdzająca, czy mamy do czynienia z cyfrą czy literą (**a** – **f**). Możemy posłużyć się biblioteczną funkcją `isdigit()`, która odpowie nam na to pytanie:

```
int value;
if( isdigit(c) )
  value = c - '0';
else
  value = c - 'a' + 10;
w = w * 16 + value;
```

Pomocnicza zmienna *value* oznacza numeryczną wartość danej cyfry. Proszę zwrócić uwagę, że w przypadku litery musimy do wartości cyfry dodać jeszcze 10, gdyż na przykład znakowi **a** odpowiada wartość 10.

Kompletna funkcja prezentuje się następująco:

```
int decimal_from_hex(string s)
{
  int w{ };
  for(char c : s)
  {
    int value;
    if( isdigit(c) )
      value = c - '0';
    else
      value = c - 'a' + 10;
    w = w * 16 + value;
  }
  return w;
}
```

Niech Czytelnik zastanowi się, jak zmienić powyższą funkcję, aby poprawnie przeliczała liczby szesnastkowe na dziesiętne bez względu na wielkość wprowadzonych liter.