Wstęp do programowania w języku C

Lista zadań nr 4

Na zajęcia 13 listopada 2017

UWAGA! Rozwiązania zadań z tej listy powinny wykorzystywać, o ile to możliwe, funkcje z biblioteki standardowej. Kod powinien być czytelny, tj.:

- sformatowany[1] zgodnie z wybraną konwencją,
- zmienne i procedury powinny być nazwane zgodnie z ich przeznaczeniem,
- należy unikać powtarzającego się kodu poprzez zamykanie go w procedury,
- złożone zadania trzeba podzielić na podprocedury.

Dodatkowo swoje programy należy kompilować z flagami «`-g -Og -Wall -Werror`», aby ostrzeżenia kompilatora były traktowane jako potencjalne błędy!

Zadanie 1 (10*). Kody *Rice*'a, będące podzbiorem kodów *Golomb*'a[2] są wykorzystywane przez niektóre algorytmy kompresji. Kodowanie to sprawdza się, jeśli w danych wejściowych jest bardzo dużo małych liczb, a prawdopodobieństwo wystąpienia większych liczb maleje wykładniczo.

Kodujemy wyłącznie liczby bez znaku. Liczbę n można zapisać jako $p \cdot 2^k + q$, gdzie $q = n \bmod 2^k$. Przedział p będziemy kodować unarnie, tj. ciągiem jedynek zakończonych zerem. Zatem $0_{10} \to 0_2$, $1_{10} \to 10_2$, $2_{10} \to 110_2$, ... Resztę z dzielenia q będziemy kodować na stałej liczbie k bitów.

Przykład: Liczba 13_{10} zostanie zakodowana jako: $1111110\,1_2$ ($k = 1$), $1110\,01_2$ ($k = 2$) i $10\,101$ ($k = 3$).

Napisz procedurę «`void print_bin(unsigned)`», która wydrukuje na standardowe wyjście liczbę w zapisie binarnym począwszy od najbardziej znaczącej jedynki – tzn. dla liczb dodatnich wydruk ma zaczynać się jedynką. Po czym zaimplementuj procedury:

```
unsigned encode(unsigned number, unsigned k);
unsigned decode(unsigned code, unsigned k);
```

... które będą odpowiednio kodować liczby do kodu *Rice*'a i na odwrót.

Napisz program, który będzie wczytywał z linii poleceń programu (`argv`) trzy rzeczy: literę `'e'` (operacja kodowania) lub `'d'` (operacja dekodowania), liczbę lub kod, oraz liczbę k; a następnie wykonywał odpowiednią operację. Należy binarnie wydrukować liczbę przed i po wykonaniu operacji.

Zadanie 2 (10). Liczby stałopozycyjne[3] są często używane w programach działających na procesorach, które nie posiadają jednostki do obliczeń zmiennopozycyjnych (np. mikrokontrolery). Liczby te reprezentuje się za pomocą typów całowitoliczbowych z ustalonym wykładnikiem. Dla liczby n notacja `Qx.y` oznacza, że na część całkowitą i znak przydzielono `x` starszych bitów, a na ułamkową `y` młodszych bitów liczby b będącej reprezentacja binarną liczby n. Zatem $n = b \cdot 2^{-y}$, gdzie $b \in \mathbb{Z}$.

[1] `https://clang.llvm.org/docs/ClangFormatStyleOptions.html\#configurable-format-style-options`
[2] `https://en.wikipedia.org/wiki/Golomb_coding`
[3] `https://en.wikipedia.org/wiki/Fixed-point_arithmetic`

Zaprogramuj bibliotekę funkcji dla liczb `Q16.16` o typie `fp16`[4] realizujących podstawowe operacje:

- arytmetyczne: dodawanie, odejmowanie, mnożenie, dzielenie, reszta z dzielenia,
- konwersji danych: `float` $\leftrightarrow$ `fp16`, `int` $\leftrightarrow$ `fp16` (zaokrąglając do najbliższej).

Zauważ, że zwykły operator porównania dla liczb całkowitych będzie dział również dla liczb `Q16.16`. Każdą z funkcji należy zapisać w odrębnym pliku źródłowym. Dodatkowo trzeba stworzyć plik nagłówkowy zawierający deklaracje typu `fp16` i funkcji na nim operujących.

Napisz program, który oblicza pierwiastek kwadratowy metodą Newton'a z liczby w formacie `Q16.16` używając wyłącznie funkcji ze swojej biblioteki. Liczbę typu `float` należy pobrać z argumentów programu i skonwertować do liczby `Q16.16`. Na standardowym wyjściu ma pojawić się wynik skonwertowany z typu `Q16.16` do typu `float` i `int`.

Zadanie 3 (10). Napisać program, który rozwiązuje zadanie oznaczone jako *Lista 3 zadanie 3* w systemie Moodle. Rozwiązanie tego zadania będzie sprawdzane automatycznie z użyciem sprawdzarki.

[4] typ deklarujemy używając słowa kluczowego `typedef`