Znakowa reprezentacja liczb dwunastkowych

Kod dwunastkowy, jak można się spodziewać, używa 12 cyfr. Pierwszych dziesięć, to cyfry arabskie; cyfry jedenasta i dwunasta to odpowiednio ‘a’ (‘A’) i ‘b’ (‘B’). Wartości tych ‘literowych’ cyfr to odpowiednio 10 i 11. Domyślnie są używane małe litery, ale funkcja konwersji na wartość całkowitą musi akceptować zarówno małe, jak i wielkie litery.

Dla uproszczenia przyjmujemy, że reprezentacja znakowa ma zawsze 4 znaki, tzn. poprawne wejście lub wynik funkcji liczy dokładnie 4 znaki (wyrównanie długości jest uzyskane przez wiodące zera).

Funkcja `dd2int` (znaki → wartość `int`) dla niepoprawnego ciągu wyjściowego daje wartość -1.

Funkcja `int2dd` (wartość `int` → znaki) dla niepoprawnej wartości wejściowej daje jednoznakowy ciąg ”x”.

Przykłady

dd2int			int2dd	
aB	-1		24000	x
xxxx	-1		1	0001
0012	14		11	000b
12Ab	2147		2056	1234

Waszym zadaniem jest zaimplementowanie obu tych funkcji i przetestowanie ich, żeby mieć pewność, że działają poprawnie.

Pokazane powyżej testy część błędów są z pewnością w stanie wyłapać, ale nie wydają mi się wystarczające. Gdybym to ja implementował te funkcje, to mój zestaw testów byłby bogatszy. W istocie mam już taki zestaw testów i to za jego pomocą będę sprawdzać Wasze rozwiązania.

Szczegóły implementacji

W pliku nagłówkowym `duodecimal.h` prócz interesujących nas przede wszystkim funkcji konwersji liczb, widać deklaracje dwóch dodatkowych funkcji, które zamieniają tylko jeden znak lub cyfrę.

Choć na dobrą sprawę są to funkcje wewnętrzne modułu, to uwzględniłem je w specyfikacji, przede wszystkim dlatego, żeby ukierunkować dekompozycję problemu.

Ma to znaczenie nie tylko dla pisania kodu, ale także dla jego testowania: po sprawdzeniu funkcji `ddd2int` i `int2ddd` będzie można znacznie ograniczyć liczbę testów `dd2int` i `int2dd`.

Funkcja `ddd2int` (znak → wartość `int`) dla niepoprawnego znaku wyjściowego daje wartość -1.

Funkcja `int2ddd` (wartość `int` → znak) dla niepoprawnej wartości wejściowej („wartości cyfry”) daje znak `'x'`.

Implementacja

Jak można podejść do implementacji tego zadania dojdziemy robiąc w trakcie ćwiczeń konwertery `hex2int` oraz `int2hex` (cyfry szesnastkowe).
Zwrócimy uwagę na to, żeby na konsoli wyświetlać informacje, które przydadzą się przy uruchamianiu kodu. Np. wyświetlające, który dokładnie przypadek testowy dał błąd.
Popatrzymy też na możliwości debuggera, bo czasami samo oglądanie kodu nie wystarczy i trzeba sprawdzić wartości argumentów, zmiennych, zobaczyć w jakiej kolejności wykonują się instrukcje, itp., itd.

Implementacja

Projekt będzie składać się tym razem z trzech plików:

`duodecimal.h` – dostarczony przeze mnie plik nagłówkowy,

`duodecimal.cpp` – implementacja konwersji,

`zad_1.cpp` – plik zawierający testy obu funkcji (jak się okaże może trzeba będzie przetestować coś więcej…).

Warto rozwiązanie wysłać jak najwcześniej: jeśli nie przejdzie moich testów dostarczę autorowi raport z testów. Przy poprawnie działających funkcjach (albo we wtorek wieczorem) będę oceniać również sposób rozwiązania zadania. A jest to pierwsze zadanie duże – za 5 punktów.

Dostarczanie rozwiązania

Rozwiązaniem zadania jest plik **`duodecimal.cpp`**. Proszę wysyłać tylko ten jeden plik (oczywiście na początku pliku trzeba umieścić w komentarzu imię, nazwisko i nr albumu autora).

Rozwiązanie przekazać do:

3 listopada 2021 23:59.