Znakowa reprezentacja liczb piętnastkowych

Kod piętnastkowy, jak można się spodziewać, używa 15 cyfr. Pierwszych dziesięć, to cyfry arabskie; cyfry od jedenastej do piętnastej to litery od ‘a’ (‘A’) do ‘e’ (‘E’). Wartości tych ‘literowych’ cyfr to oczywiście 10, 11, 12, 13 i 14. Domyślnie są używane małe litery, ale funkcja konwersji na wartość całkowitą musi akceptować zarówno małe, jak i wielkie litery.

Grzechem byłoby nie skorzystać z przygotowanych w zadaniu małym czwartym funkcji konwersji pojedynczych cyfr:

```
int pentadecimal2int(char digit);
char int2pentadecimal(int value);
```

Liczbami (czyli sekwencjami cyfr) będą zajmować się funkcje:

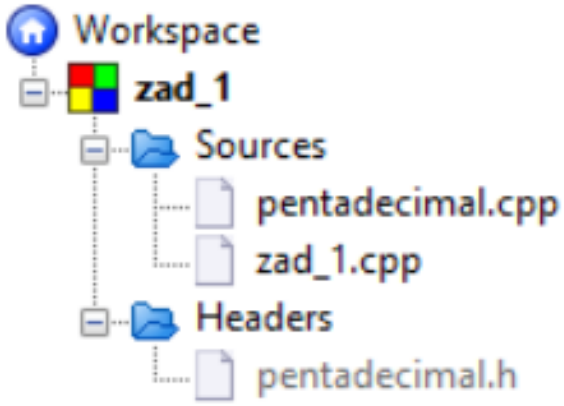
```
int penta2int(const string& number);
string int2penta(int value);
```

Trochę objaśnień

W tym ćwiczeniu już nie tylko symulujemy współpracę z innymi programistami, ale tę współpracę realizujemy. Resztę świata, która korzysta z napisanych przez Was funkcji, reprezentuję ja.

Podstawą współpracy są pliki nagłówkowe: to w nich znajdę deklaracje funkcji, które od Was dostaję. Umówmy się, że deklaracje powinny wyglądać tak, jak na poprzedniej stronie (ale będę konieczne pewne modyfikacje).

Struktura projektu będzie następująca:



W pliku zad_1.cpp umieścicie Państwo własne testy funkcji konwersji.

Moje testy, które posłużą do oceny rozwiązania, będą dość gruntowne i nie będziecie mieć do nich dostępu (natomiast do raportów z moich testów – tak).

Przykłady z komentarzem

`penta2int`			`int2penta`	
	`-1`		`-1`	`x`
`F`	`-1`		`0`	`0`
`12`	`19`		`11`	`b`
`abcde9`	`8194269`		`78524`	`183ee`

To oczywiście tylko próbka testów i warto nad nimi popracować. Więcej, warto się nimi podzielić, żeby trochę mniej się napracować i mieć większe nadzieje, że funkcje działają we wszystkich przypadkach.

Funkcja `int2penta` jest stosukowo prosta: błędne są wszystkie wartości ujemne i w tym przypadku funkcja zwraca łańcuch `"x"`.

Z `penta2int` będzie więcej uciechy. Na pierwszy rzut oka widać, że pojawienie się w łańcuchu nie-cyfry daje błąd.

Przekroczenie zakresu w penta2int

To, co może Państwu dać w kość, jest przekroczenie zakresu wartości, które można reprezentować w typie `int`.
Typowo, wartość liczby liczymy krok po kroku przeliczając aktualną wartość i nową (najmniej znaczącą w danej chwili) cyfrę:

```
value = value * code_base + digit_value
```

`code_base` jest w naszym przypadku oczywiste: 15. Do wydobycia `digit_value` mamy już napisaną funkcję `pentadecimal2int`.
Problem w tym, że w obliczeniach po prawej stronie = możemy przekroczyć zakres wartości `int`. Żeby dowiedzieć się, jaka jest ta maksymalna wartość skorzystamy z definicji z `numeric_limits`:

```
numeric_limits<int>::max()
```

Główna podpowiedź z mojej strony jest taka: trzeba to zrobić *przed* wyliczeniem wartości wyrażenia.
Przygotowanie przypadków testowych może nas nakierować na rozwiązanie problemu.

Implementacja `penta2int`

Zasadnicza część działań w `penta2int` wiąże się z przejściem po cyfrach argumentu `number`. We współczesnym C++ takie przejście organizuje się tak:

```
for (char digit : number)
{
    // digit jest pojedynczym znakiem z number
}
```

Co zrobić w bloku instrukcji `for`, jest stosunkowo jasne:

1. Sprawdzić, czy digit jest faktycznie cyfrą (jak nie – błąd).
2. Sprawdzić, czy nie wystąpi nadmiar (jak tak – błąd).
3. Obliczyć nową wartość liczby uwzględniającą już `digit`.

Implementacja `int2penta`

Konwersja wartości do reprezentacji znakowej jest na ogół wykonywana od najmłodszej cyfry. Tę cyfrę dostajemy jako resztę z dzielenia wartości przez podstawę kodu.

Wartość, która zostaje nam do konwersji, po ustaleniu cyfry, to wynik dzielenia całkowitego przez podstawę kodu.

Kroki powtarzamy, póki wartość pozostała do konwersji jest niezerowa.

Wartość, która początkowo jest 0 trzeba poddać konwersji – nie chcemy dostać liczby pustej, a jednocyfrowe „0”. Tu właściwa będzie pętla do – while:

```
do {
    // kod dla jednej cyfry wyniku
} while (value != 0);
```

Implementacja – podsumowanie

Projekt składa się z trzech plików:

`pentadecimal.h` – plik nagłówkowy z prototypami funkcji,

`pentadecimal.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 w środę wieczorem) będę oceniać również sposób rozwiązania zadania. A jest to pierwsze zadanie duże – za 5 punktów.

Dostarczanie rozwiązania

Rozwiązaniem zadania są pliki **`pentadecimal.h`** i **`pentadecimal.cpp`**. Proszę ładować te dwa pliki w Moodle (oczywiście na początku pliku trzeba umieścić w komentarzu imię, nazwisko i nr albumu autora).

Pod nagłówkiem `Tydzień 5 - 2-6.11` znajdziecie Państwo zadanie zatytułowane `Liczby piętnastkowe`. W tym zadaniu każdy z Was powinien załadować pliki do:

16 listopada 2022 23:59.