Wstęp do programowania w C

Robert Ferens

Lista 4 - 10 listopada 2021 EDIT 11.11

Zadanie 1

Napisz funkcję o podanej formie:

```
int unicode_to_utf8(const unsigned int unicode, unsigned char utf8[5]);
```

W wyniku wywołania funkcja powinna zapełnić tablicę kodowaniem znaku o podanym unikodzie w UTF-8 (następny po znaku bajt to 0) i zwrócić ilość bajtów zapisu utf-8 znaku w wyniku, lub -1 w przypadku błędu.

Przykłady:

Jeśli zdefiniuję funkcję:

```
void test(unsigned int unicode)
{
    unsigned char utf8[5] = {0};
    printf("Zapis unicode: 0x%x\n", unicode);
    int nbytes = unicode_to_utf8(unicode, utf8);
    printf("%d-bajtowa reprezentacja UTF-8: %x %x %x %x %x\n",
        nbytes, utf8[0], utf8[1], utf8[2], utf8[3], utf8[4]);
    printf("s: %s \n", utf8);
}
```

i wykonam kod:

```
    test(L'Ą');
    test(0x104);  //ta sama liczba!
    test(260);    //ta sama liczba!
    test(0x03C0); //pi
    test(L'€');
    test(0x4eba);   // japoński/koreański/chiński znak na człowieka
    test(0x01D11E); //klucz wiolinowy
```

to powinienem otrzymać:

```
    Zapis unicode: 0x104
    2-bajtowa reprezentacja UTF-8: c4 84 0 0 0
    s: Ą
    Zapis unicode: 0x104
    2-bajtowa reprezentacja UTF-8: c4 84 0 0 0
```

```
s: Ą
Zapis unicode: 0x104
2-bajtowa reprezentacja UTF-8: c4 84 0 0 0
s: Ą
Zapis unicode: 0x3c0
2-bajtowa reprezentacja UTF-8: cf 80 0 0 0
s: [pi]
Zapis unicode: 0x20ac
3-bajtowa reprezentacja UTF-8: e2 82 ac 0 0
s: €
Zapis unicode: 0x4eba
3-bajtowa reprezentacja UTF-8: e4 ba ba 0 0
s: [chiński znak]
Zapis unicode: 0x1d11e
4-bajtowa reprezentacja UTF-8: f0 9d 84 9e 0
s: [klucz wiolinowy]
```

Uwaga! Osoby korzystające z windowsa mogą nie zobaczyć znaków wypisanych poprawnie, wystarczy że reprezentacja w tablicy jest dobra. Jeśli jednak chciałby to część znaków wypisze się poprawnie jeśli ręcznie w terminalu zmianią kodowanie na utf8 wpisując *chcp 65001*, a następnie uruchamiając swój program *.\prog.exe* w tym samym terminalu. Nie wszystkie jednak czcionki są zainstalowane i z niektórymi znakami nadal może być problem.

Materiały:

1. Poczytaj o formatach Unicode i UTF-8 naucz się zamieniać - najpierw na kartce! `https://pl.wikipedia.org/wiki/UTF-8`
2. Sprawdź różne kodowania znaków `https://www.compart.com/en/unicode/U+03C0` utf32 możesz traktować jako 16-kowy zapis liczby unicode.
3. Poszukaj jakichś źródeł i poczytaj o opoeratorach bitowych w c, np. `https://cybersecurity.umcs.lublin.pl/wp-content/uploads/kmazur/PP2017/bity_wsk_na_funkcje.pdf`

Zadanie 2

Wybierz jedno z poniższych:

1. Napisz prosty kalkulator bitowy, obsługujący operacje &, |, ~, <<, >>. Na początku kalkulator może zapytać o wybór operacji, a następnie poprosić o 2 liczby(lub jedną dla ~), wyliczyć wynik na tych dwóch liczbach, wypisać go i zakończyć działanie. Wybierz sposób podania i wypisu liczb jako binarny lub szesnastkowy (drugi parametr << i >> może być dziesiętny), obie implementacje będą spełniały wymogi zadania.
2. Napisz i przetestuj funkcję przeciwną do tej w zadaniu pierwszym, przyjmującą kodowanie utf8 i zwracającą kod unicode:

   ```
   int utf8_to_unicode(const unsigned char utf8[5]);
   ```

   Zwróć -1 gdy kod jest niepoprawny.

Zadanie 3

Dostępne ze sprawdzaczką na skosie.