Soluzione dell’esercizio “Sfida” sulla codifica Unicode (pag. 118)

Codifica UTF-8

Ricordiamo la tabella degli spazi occupati in memoria dai caratteri Unicode in funzione del loro codepoint:

Bit	Last codepoint	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
7	U+007F	0xxxxxxx					
11	U+07FF	110xxxxx	10xxxxxx				
16	U+FFFF	1110xxxx	10xxxxxx	10xxxxxx			
21	U+1FFFFF	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx		
26	U+3FFFFF	111110xx	10xxxxxx	10xxxxxx	10xxxxxx	10xxxxxx	
31	U+7FFFFFFF	1111110x	10xxxxxx	10xxxxxx	10xxxxxx	10xxxxxx	10xxxxxx

Carattere

Dalla tabella risulta che il codepoint U+AC80 è codificato su 16 bit, essendo maggiore di 07FF e minore di FFFF. Il carattere occuperà in memoria 3 byte, organizzati secondo la struttura di bit: 1110**xxxx** 10xxxxxx 10xxxxxx. Nei bit indicati con la X (16 in tutto, divisi in tre gruppi di bit:
4bit-**6bit**-**6bit**) si riporta la codifica binaria del codepoint.

Convertendo il valore AC80 su 16 bit otteniamo 1010110010000000 (non occorrono bit di padding).
Dividiamo la sequenza in tre gruppi secondo le dimensioni richieste: **1010** **110010** **000000**.
Inseriamo i gruppi nella struttura, al posto di simboli X, e ricaviamo i tre byte che costituiscono la codifica UTF-8 del carattere assegnato: 1110**1010** 10**110010** 10**000000**, equivalenti in esadecimale[1] a: **EA B2 80 (UTF-8)**.

È possibile verificare il risultato utilizzando uno dei convertitori online di caratteri Unicode, per esempio https://mothereff.in/utf-8:

UTF-8-decoded:

검

UTF-8-encoded: (permalink)

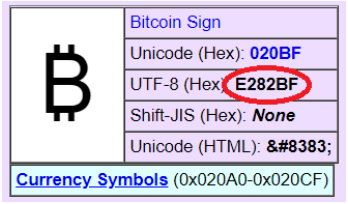
```
\xEA\xB2\x80
```

1

Carattere

Anche in questo caso il codepoint U+20BF è codificato su 16 bit: 0010000010111111 (si noti l’aggiunta di due bit di padding a sinistra). Dividendo la sequenza nei tre gruppi **0010** **000010** **111111** e riportando il tutto nella strutttura di tre byte, si ricava 1110**0010** 10**000010** 10**111111**, equivalente in esadecimale a **E2 82 BF (UTF-8)**.

Un ulteriore strumento di verifica del risultato ottenuto è il link: https://www.isthisthingon.org/unicode/index.phtml?glyph=20BF



Carattere

Il codepoint U+1F62D richiede 21 bit, essendo maggiore di FFFF ma minore di 1FFFFF. Dalla tabella risulta che lo spazio occupato in memoria dalla codifica UTF-8 di questo carattere è di quattro byte:
11110**xxx** 10**xxxxxx** 10**xxxxxx** 10**xxxxxx**: il codepoint binario è suddiviso quindi in gruppi di 3bit-6bit-6bit-6bit.
Convertiamo il codepoint esadecimale su 21 bit: 1F62D = **000**01111**01**1000**101101**.
Inserendo i quattro gruppi nella struttura di bit si ricava: 11110**000** 10**011111** 10**011000** 10**101101**, corrispondente in esadecimale a **F0 9F 98 AD (UTF-8)**.

Codifica UTF-16

Carattere

Il codepoint U+AC80, essendo minore di FFFF, appartiene al BMP (*Basic Multilingual Plane*) ed è codificabile in UTF-16 su 16 bit: AC80 = 1010110010000000, ovviamente corrispondente a AC 80:

UTF-16BE = **AC 80**
UTF-16LE = **80 AC**

Carattere

Poiché il codepoint è maggiore di FFFF, si procede alla determinazione della coppia surrogata nel seguente modo:

1. Si sottrae al codepoint esadecimale la costante $10000_{(16)}$:
   $1F62D_{(16)} - 10000_{(16)} = 0F62D_{(16)}$
2. Si converte la differenza in binario su 20 bit e si divide il risultato in due parti da 10 bit ciascuna (**parte alta** / **parte bassa**):
   0F62D = **0000111101 1000101101**
3. Si somma la parte alta della differenza **0000111101**, ottenuta al punto precedente, alla costante $D800_{(16)}$ (in binario: 1101100000000000) per ricavare il primo elemento della coppia surrogata:
   0000111101 + 1101100000000000 = 1101100000111101 = $D83D_{(16)}$
4. Si calcola il secondo elemento della coppia sommando la parte bassa della differenza **1000101101** alla costante $DC00_{(16)}$ (in binario: 1101110000000000):
   1000101101 + 1101110000000000 = 1101111000101101 = $DE2D_{(16)}$
5. La coppia surrogata è quindi D83D DE2D, corrispondente alle seguenti codifiche:
   UTF-16BE = **D8 3D DE 2D**
   UTF-16LE = **3D D8 2D DE**