TANKERINO.com

Lezione 73

https://www.tankerino.com/it/corsi/4/tepi-3/lezioni/73/rappresentazione-del-testo

Rappresentazione del Testo in Binario

La rappresentazione del testo in formati digitali, come quelli usati nei computer, è basata su codifiche che convertono le lettere, i numeri e altri caratteri in una sequenza di bit. Questa lezione copre i fondamenti di come il testo viene rappresentato in binario e il suo sviluppo storico.

Storia delle Codifiche del Testo

La storia delle codifiche testuali risale ai primi giorni della teletipografia e della comunicazione a distanza. Inizialmente, ogni produttore aveva la propria codifica, rendendo difficile lo scambio di informazioni tra sistemi diversi. La necessità di standardizzazione divenne evidente, portando alla creazione di ASCII negli anni '60 come primo tentativo di unificare la rappresentazione del testo. Una codifica è un sistema che mappa una serie di elementi (come lettere o numeri) a un altro insieme di valori (come numeri binari). È il fondamento per trasformare informazioni leggibili da umani in dati che un computer può elaborare.

La Codifica ASCII ed ASCII Estesa

La codifica ASCII (American Standard Code for Information Interchange) è un sistema di codifica dei caratteri molto diffuso. È stato originariamente creato per codificare i caratteri inglesi.

> Il carattere 'A' in ASCII è rappresentato dal numero 65.

> In ASCII, ogni carattere è rappresentato da un numero univoco compreso tra 0 e 127.

Questo sistema di codifica ha permesso la standardizzazione della trasmissione di testi in formati digitali, facilitando lo scambio di dati tra diverse piattaforme informatiche.

ASCII è limitato in quanto comprende solo caratteri inglesi, numeri, simboli di punteggiatura e comandi di controllo, ma non supporta accenti o caratteri speciali di altre lingue.

L'ASCII Estesa

L'ASCII Estesa è un'espansione del set ASCII originale. Esso include un totale di 256 caratteri, estendendo così la gamma per includere caratteri speciali, simboli grafici e caratteri di alcune lingue europee.

> Il carattere 'è' in ASCII Estesa potrebbe essere rappresentato dal numero 138.

> L'ASCII Estesa utilizza due byte per ogni carattere, permettendo la rappresentazione di 128 caratteri aggiuntivi rispetto all'ASCII standard.

Questa espansione è stata cruciale per includere la diversità linguistica e simbolica nelle comunicazioni digitali, sebbene non sia sufficiente per coprire tutte le lingue e i simboli esistenti.

Nonostante la sua utilità, l'ASCII Estesa ha alcune limitazioni, come la mancanza di supporto universale per tutte le lingue, portando all'adozione di altri sistemi di codifica come UTF-8.

Importanza dell'ASCII nella Programmazione

La conoscenza della codifica ASCII è fondamentale per i programmatori. Essa gioca un ruolo chiave nella rappresentazione dei dati e nella compatibilità tra diversi sistemi e software.

> Conoscere il codice ASCII dei caratteri può essere utile per la manipolazione di stringhe e dati in molti linguaggi di programmazione.

ASCII è anche fondamentale nella comprensione di concetti come il terminatore di stringa (null character) e il controllo dei flussi di dati.

Anche con l'avvento di nuove codifiche, la comprensione di ASCII rimane un elemento base nella formazione di ogni programmatore.

Dettagli su ASCII

ASCII utilizza 7 bit per rappresentare ogni carattere, permettendo di codificare 128 caratteri diversi. Questi includono lettere maiuscole e minuscole, numeri, segni di punteggiatura e caratteri di controllo come il ritorno a capo e la tabulazione.

> In ASCII, la lettera 'A' è rappresentata dal numero decimale 65 o dal numero binario '1000001'.

Decimale	Binario	Carattere
32	00100000	Spazio
65	01000001	A
66	01000010	B
97	01100001	a
98	01100010	b

Unicode: Un Linguaggio Universale per i Caratteri

> Unicode è un sistema di codifica dei caratteri che mira a rappresentare ogni carattere utilizzato nella scrittura di testo in tutte le lingue del mondo.

Si tratta di uno sforzo collaborativo per creare un "alfabeto universale", che offre una soluzione al problema dei numerosi sistemi di codifica incompatibili tra loro utilizzati in precedenza.

Storia e necessità

Prima dell'avvento di Unicode, c'erano centinaia di differenti sistemi di codifica per rappresentare i caratteri di diverse lingue. Questa frammentazione creava problemi di incompatibilità, in quanto un testo codificato con un sistema potrebbe non essere letto correttamente da un software che utilizza un altro sistema di codifica. Unicode è nato dalla necessità di avere un sistema di codifica universale (caratteri provenienti da tutte le lingue del mondo), che potesse essere utilizzato indipendentemente dalla lingua o dalla piattaforma.

Funzionamento

Al centro di Unicode c'è un vasto set di numeri, e ogni numero è associato a un unico carattere. Questi numeri sono chiamati "punti codice". Ad esempio, il punto codice per la lettera "A" è U+0041.

Tuttavia, è importante notare che Unicode in sé non definisce come questi punti codice dovrebbero essere memorizzati in termini di bit.

Esistono varie codifiche (come UTF-8, UTF-16, UTF-32) che definiscono come trasformare un punto codice Unicode in una sequenza di bit e viceversa.

UTF-8 e la sua relazione con Unicode

UTF-8 è attualmente la codifica di caratteri più popolare su Internet e ha un ruolo fondamentale nella diffusione di Unicode. Si tratta di una codifica a lunghezza variabile, che può rappresentare un punto codice Unicode utilizzando da 1 a 4 byte. La bellezza di UTF-8 è che è retrocompatibile con ASCII: i primi 128 caratteri di Unicode (che corrispondono ai caratteri ASCII) sono rappresentati in UTF-8 esattamente come in ASCII.

Unicode ha rivoluzionato il modo in cui pensiamo e lavoriamo con il testo in ambito informatico. Ha fornito un fondamento solido per la rappresentazione universale dei caratteri, garantendo che i testi di qualsiasi lingua possano essere elaborati, archiviati e trasmessi in modo coerente e affidabile su piattaforme e applicazioni diverse.

UTF-8, una delle sue codifiche, è retrocompatibile con ASCII e può rappresentare ogni carattere Unicode utilizzando sequenze di lunghezza variabile, da 1 a 4 byte.

> Nel sistema UTF-8, il carattere 'A' è ancora '1000001', ma caratteri come '€' hanno una rappresentazione binaria più complessa.

Oltre a UTF-8, Unicode può essere rappresentato anche con UTF-16 e UTF-32.

- Mentre UTF-8 utilizza sequenze di byte variabili
- UTF-16 utilizza 2 o 4 byte
- UTF-32 utilizza sempre 4 byte per carattere

Queste codifiche sono spesso utilizzate in contesti specifici a seconda delle esigenze di memoria e compatibilità.

In confronto, le codifiche a lunghezza fissa, come UTF-32, hanno una lunghezza definita per ogni carattere. In UTF-32, ogni carattere è rappresentato da esattamente 4 byte, indipendentemente dal carattere effettivo. Questa lunghezza fissa rende la codifica e la decodifica semplici e dirette, ma può anche portare a un utilizzo inefficace dello spazio per testi composti principalmente da caratteri che potrebbero essere rappresentati con codifiche più brevi, come ASCII o UTF-8.

UTF-8: lunghezza Variabile

UTF-8 è una codifica a lunghezza variabile che può rappresentare un carattere con una sequenza di 1 a 4 byte. Ciò che rende unico UTF-8 è il modo in cui utilizza i bit di alto ordine di ciascun byte per indicare se un byte fa parte di una sequenza multi-byte e, in caso affermativo, quale parte di quella sequenza rappresenta.

1. Byte singolo (ASCII): I caratteri che rientrano nel range ASCII (0-127) vengono codificati con un solo byte. Questo byte inizia sempre con lo `0` di bit di alto ordine.

   ```
   0xxxxxxx
   ```

2. 2 Byte: Se un carattere richiede due byte in UTF-8, il primo byte inizierà con `110`, e il secondo byte inizierà con `10`.

   ```
   110xxxxx 10xxxxxx
   ```

3. 3 Byte: Per i caratteri che richiedono tre byte, il primo byte inizia con `1110`, e i due byte successivi iniziano entrambi con `10`.

   ```
   1110xxxx 10xxxxxx 10xxxxxx
   ```

4. 4 Byte: Infine, i caratteri che richiedono quattro byte avranno il primo byte che inizia con `11110`, seguito da tre byte che iniziano tutti con `10`.

   ```
   11110xxx 10xxxxxx 10xxxxxx 10xxxxxx
   ```

In base al bit di alto ordine (o bit di avvio) di un byte, il decodificatore UTF-8 può determinare se il byte fa parte di una sequenza a lunghezza variabile e quale parte di quella sequenza rappresenta. Questo meccanismo consente a UTF-8 di essere retrocompatibile con ASCII (poiché i caratteri ASCII sono rappresentati da un singolo byte che inizia con `0`) e di gestire simultaneamente una vasta gamma di caratteri da molte lingue diverse.

Caratteri Speciali e di Controllo

Oltre ai caratteri visibili, esistono caratteri "di controllo" utilizzati per formattare e controllare la visualizzazione del testo. Per esempio, il "ritorno a capo" indica una nuova riga, mentre la "tabulazione" sposta il cursore a una posizione prestabilita.

Problemi Comuni con le Codifiche

La codifica dei caratteri è un elemento fondamentale nella rappresentazione e nel trasferimento dei dati testuali. Tuttavia, esistono diverse codifiche e non tutte sono compatibili tra loro. Ciò può portare

a vari problemi quando si lavora con il testo, specialmente in un contesto globale.

1. Mismatch di Codifica

Molto spesso, un documento o una risorsa web viene salvato con una determinata codifica, ma viene interpretato con una diversa codifica.

> Immagina di avere un documento scritto in Unicode ma lo apri con un editor di testo che lo interpreta come ASCII. Probabilmente vedrai un sacco di caratteri strani o simboli di sostituzione al posto del testo atteso.

2. Caratteri Non Rappresentabili

Alcune codifiche possono non supportare determinati caratteri. Se provi a rappresentare un carattere che non esiste in una specifica codifica, potresti finire con un simbolo di sostituzione o con un errore.

3. Problemi di Retrocompatibilità

Non tutte le codifiche sono retrocompatibili con versioni precedenti o con altri standard. Ad esempio, mentre UTF-8 è retrocompatibile con ASCII, non lo è con molte altre codifiche.

4. Efficienza della Memoria

Codifiche diverse utilizzano quantità diverse di memoria per rappresentare lo stesso testo. Mentre alcune codifiche sono ottimizzate per la compattezza, altre (come UTF-32) utilizzano una quantità fissa di memoria per ogni carattere, che potrebbe non essere efficiente per lunghi testi.

5. Problemi di Trasmissione e Archiviazione

Quando i dati vengono trasferiti tra sistemi o archiviati, potrebbero esserci problemi se il sistema di destinazione non supporta la codifica utilizzata. Questo è particolarmente vero per i database o i servizi web che interagiscono con client globali.

6. Problemi nell'Interfaccia Utente

Se un'applicazione non gestisce correttamente le codifiche, l'utente potrebbe vedere caratteri errati o messaggi di errore incomprensibili. Questo può portare a frustrazione e confusione.

> È fondamentale testare le proprie applicazioni in scenari reali per garantire che gestiscano

correttamente le codifiche e forniscano un'esperienza utente ottimale.

Estensioni e Future Codifiche

Mentre il mondo digitale continua a evolversi, le codifiche dei caratteri si stanno adattando per far fronte alle esigenze emergenti. La storia delle codifiche è stata segnata da molteplici sviluppi, con l'intento di creare un sistema universale che possa rappresentare ogni lingua e ogni carattere.

Storia delle Codifiche: Date Chiave

- 1963: Emergenza della codifica ASCII (American Standard Code for Information Interchange). ASCII è diventata la codifica standard per i computer e ha dato il via all'evoluzione dei sistemi di codifica.
- 1981: Introduzione dell'ISO 8859-1, conosciuto anche come Latin-1. Questo standard era più ampio dell'ASCII e poteva rappresentare molte altre lingue occidentali.
- 1991: Unicode viene presentato al mondo. Progettato per includere ogni carattere di ogni lingua, Unicode ha cambiato radicalmente il panorama delle codifiche.
- 1996: Viene introdotto UTF-8, che diventerà rapidamente la codifica preferita su Internet grazie alla sua retrocompatibilità con ASCII e alla sua efficienza nella rappresentazione dei caratteri.

Estensioni in Corso

Con la crescente digitalizzazione delle lingue minoritarie e dei sistemi di scrittura antichi, Unicode sta costantemente aggiungendo nuovi caratteri. Le recenti versioni di Unicode hanno incluso simboli per lingue in estinzione, geroglifici e persino emoji.

Guardando al Futuro

Mentre Unicode ha rivoluzionato la rappresentazione dei caratteri, il mondo digitale potrebbe avere bisogno di ulteriori innovazioni per far fronte alle sfide future. Alcuni ambiti di interesse potrebbero includere:

- Codifiche Ottimizzate per la Machine Learning e l'IA: Con l'ascesa dell'intelligenza artificiale, potrebbero emergere nuove codifiche ottimizzate per l'elaborazione del linguaggio naturale e altre attività di IA.
- Codifiche per la Realtà Virtuale e Aumentata: Con l'immersione in mondi digitali, potrebbero esserci esigenze di rappresentare il testo in modi nuovi e innovativi.
- Maggiore Integrazione con la Grafica: Man mano che i confini tra testo e grafica diventano sempre più sfumati (ad es. con le emoji), potremmo vedere codifiche che integrano strettamente la

rappresentazione di testo e immagini.

Mentre guardiamo al futuro delle codifiche, è essenziale mantenere l'apertura verso le innovazioni e essere pronti ad adottare nuovi standard che emergono in risposta alle esigenze mutevoli del mondo digitale.

Il campo delle codifiche dei caratteri ha una storia affascinante e un futuro altrettanto entusiasmante. Man mano che il mondo digitale continua a evolversi, è probabile che vedremo ulteriori innovazioni e cambiamenti in questo dominio.

Questa lezione e' stata realizzata grazie al contributo di:

Baobab.school
Risorse per la scuola
https://www.baobab.school

FrancescoBelloni.it
software development
Siti web a Varese
https://www.francescobelloni.it