Compressione e decompressione con algoritmo LZSS

Introduzione:

Questo progetto è stato sviluppato interamente da me, partendo da pochissima esperienza in programmazione.

Le idee e gli algoritmi di risoluzione per eseguire una compressione LZSS e successivamente decomprimere sono stati frutto di molte ore di studio e sviluppo.

Questo progetto permette di comprimere qualsiasi tipo di file utilizzando l'algoritmo di compressione LZSS e successivamente di decomprimere il file precedentemente generato.

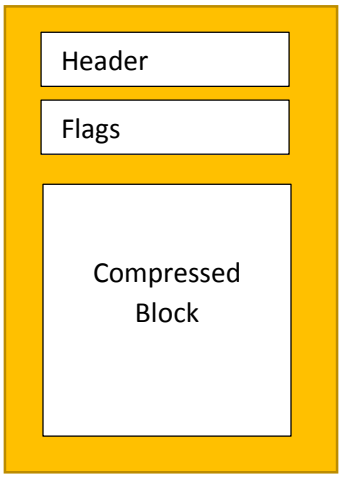
Descrizione:

La qualità di compressione è generalmente ottimale, dipende tuttavia dal tipo di file che si sta comprimendo: I files già compressi difficilmente saranno ancora comprimibili.

L'utilizzo del mio compressore e decompressore è possibile attraverso le piattaforme windows e linux, ma pure su altre piattaforme che compilano codice C è possibile utilizzarlo.

I problemi che ho riscontrato maggiormente sono stati durante la decompressione, poichè lavorando bit a bit con il solo ausiglio degli operatori shift e delle maschere è stato parecchio complicato.

Per implementare LZSS ho deciso che per praticità nel codice, la struttura di un file compresso è la seguente:



Header:

Composto da 2 Byte e serve al decompressore per sapere quante flags sono state utilizzate

Flags:

Bit che descrivono se un determinato blocco di codice è da ritenersi compresso oppure no

Compressed Block:

Serie di byte che se separati correttamente in combinazione con le flags, formano il messaggio originale.

Utilizzo:

L'utilizzo del mio eseguibile è molto semplice e diretto.

Unix:

./LZSS <c/d> <file name>

Windows:

LZSS.exe <c/d> <file name>

Dove:

c è per comprimere

d è per decomprimere

file name è il percorso e il nome del file che si vuole comprimere/decomprimere.

Nel caso dell'utilizzo dell'algoritmo di compressione, dato un file di input chiamato "test", si avrà lo stesso file, compresso, in output, chiamato "test.niko"

Nel caso dell'utilizzo dell'algoritmo di decompressione, dato un file di input chiamato "test.niko" (nota l'estensione è obbligatoria), si avrà un file di output chiamato "test".

Il progetto è compilabile in versione debug!

Grazie alla versione di debugging è possibile analizzarne il funzionamento e capirne i processi.

È consigliato comunque l'utilizzo di piccoli file per il debugging poichè la stampa a schermo rallenta notevolmente la compressione/decompressione.

Per praticità ho allestito un video per dimostrazione e presentazione.

http://youtu.be/D23nKx0V1TA

Grazie mille e buona giornata,

Niko Storni