ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ
ΚΕΝΤΡΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΟΜΕΑΣ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΤΕΧΝΙΚΩΝ & ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Υλοποίηση αλγορίθμων κατακερματισμού και LZW
Εφαρμογή στη συμπίεση κειμένου

ΣΠΟΥΔΑΣΤΗΣ
Καρατσίν Ηλίας

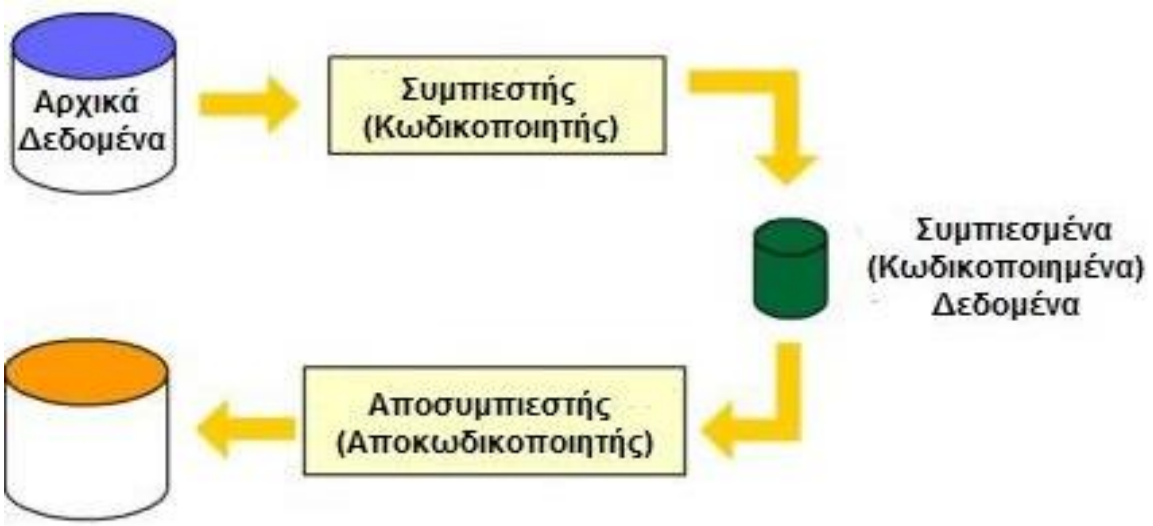
ΕΙΣΗΓΗΤΕΣ
Πάρις Αστ. Μαστοροκώστας - Επιβλέπων
Ευάγγελος Ούτσιος

Εισαγωγή στην Συμπίεση Δεδομένων

- Η συμπίεση δεδομένων είναι μια διαδικασία κατά την οποία το μέγεθος ενός ψηφιακού αρχείου μειώνετε με την διαδικασία της εκ νέου κωδικοποίησης του (re - encoding), κάνοντας το να απαιτεί λιγότερα bit.

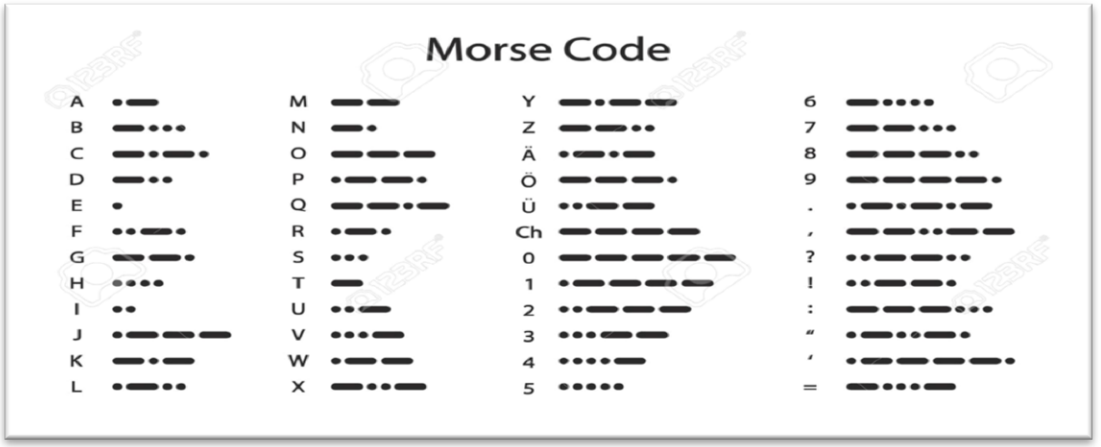
- Στην συνέχεια το αρχικό αρχείο μπορεί να ξαναδημιουργηθεί από το συμπιεσμένο αρχείο χρησιμοποιώντας την αντίστροφη διαδικασία από αυτήν της συμπίεσης, την αποσυμπίεση δεδομένων

- Συμβάλει στη μείωση του κόστους επικοινωνίας μέσω τις αύξησης της ταχύτητας μετάδοσης και στην αύξηση του αποθηκευτικού διαθέσιμου ελεύθερου χώρου.

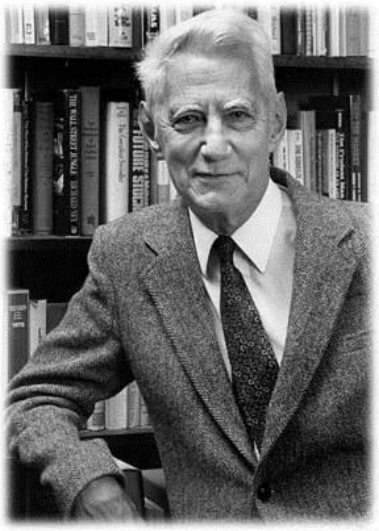


Ιστορική Αναδρομή

- 1838.
  Το ποιο παλιό παράδειγμα συμπίεσης δεδομένων, αυτο του κωδικα Morse (Morse Code), όπου οι ποιο συχνοί χαρακτήρες στο Αγγλικό αλφάβητο όπως το “t” και “e” έδιναν μικρότερους Morse κώδικες t =-, e =·.



- 1949. Πρώτοι μέγα υπολογιστές (mainframe computers). Οι Claude Shannon και Robert Fano εφεύραν την κωδικοποίηση Shannon-Fano. Ανάθεση κωδικών από bits σε σύμβολα ανάλογα με τις πιθανότητες εμφάνισής τους και διάταξη αυτών σε φθίνουσα διάταξη. Αυτοί οι κώδικες από bits είναι μικρότερη σε μέγεθος από τα bits όπου απαιτούν τα αρχικά σύμβολα.



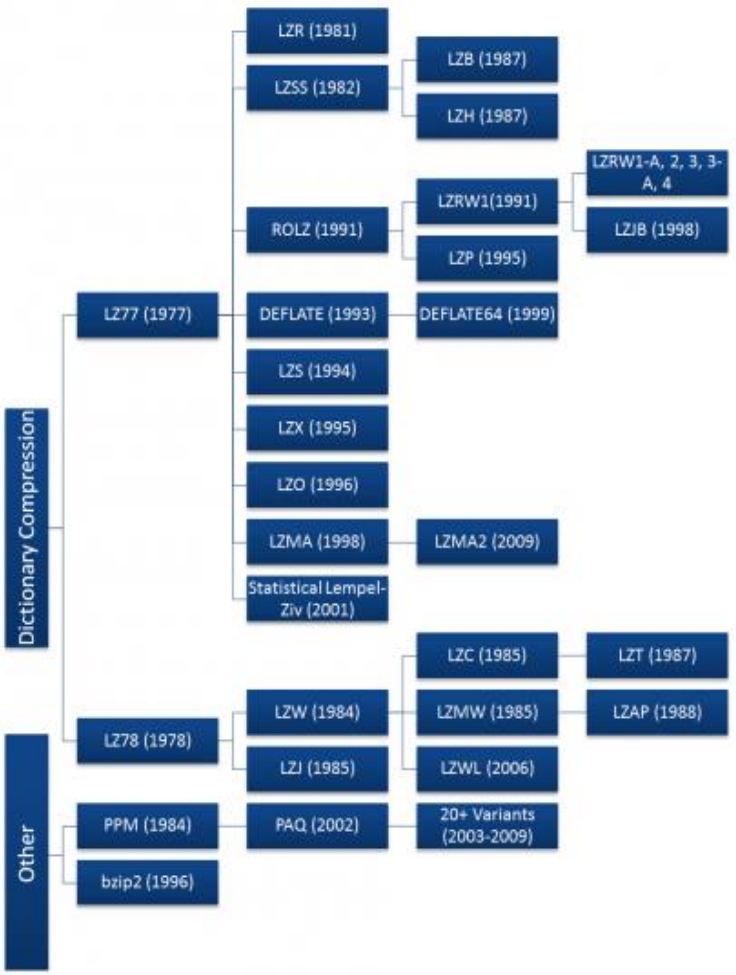
Claude Elwood Shannon (April 30, 1916 – February 24, 2001)
«Ο πατέρας της Θεωρίας της Πληροφορίας»

Ιστορική Αναδρομή

- 1952. Βελτίωση του αλγορίθμου Shannon, από τον Huffman. Οι κώδικες σχηματιζόντουσαν από τον αλγόριθμο με μια διαδικασία βημάτων, όπου στηριζόταν στην δημιουργία κόμβων με τα οποία αποτελούνται από ζευγάρια των 2 τελευταίων συμβόλων με την μικρότερη πιθανότητα εμφάνισης, μέχρι να έχουμε το τελευταίο σύμβολο με την μεγαλύτερα πιθανότητα εμφάνισης, αυτή η διαδικασία προσδιορίζετε ως από κάτω προς τα πάνω προσέγγιση (buttom - up), αντιστροφη απο αυτη του Shannon-Fano (top-buttom).

- 1977. Οι Abraham Lempel και Jacob Ziv δημοσίευσαν τον αλγόριθμο LZ77 όπου χρησιμοποιούσε ένα δυναμικό λεξικό όπου μερικές φορές το προσδιόριζαν ως συρόμενο παράθυρο.

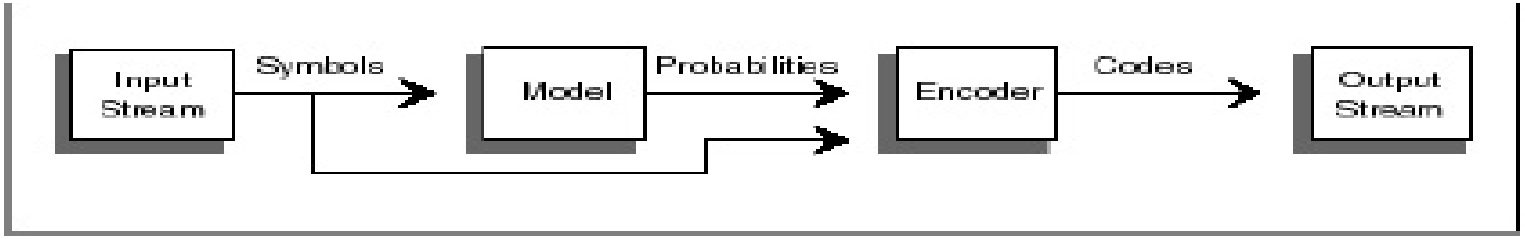
- Το 1978. Το ίδιο δίδυμο δημοσίευσε τον αλγόριθμο LZ78. Χρησιμοποιεί ένα στατικό λεξικό όπου το παράγει κατά την σάρωση των δεδομένων πριν ξεκινήσει η διαδικασία της συμπίεσης.



Ιεραρχία Αλγόριθμων συμπίεσης χωρίς απώλειες

Διαδικασία Συμπίεσης

- Χωρίζετε σε δυο κύρια κομμάτια. Το μοντέλο και τον κωδικοποιητή.
- Ένα πρόγραμμα Συμπίεσης χρησιμοποιεί το μοντέλο για να καθορίσει με ακρίβεια τις πιθανότητες ενός συμβόλου και τον κωδικοποιητή για να παράγει το κατάλληλο κωδικό με βάση αυτών των πιθανοτήτων.
- Διαχωρίζουμε την κωδικοποίηση με το μοντέλο διοτι υπάρχουν πάρα πολλοί τρόποι μοντελοποίησης των δεδομένων, χρησιμοποιώντας τον ίδιο κωδικοποιητή.
- Παράδειγμα : Κωδικοποίηση Huffman όπου το μοντέλο υπολογίζει την πιθανότητα εμφάνισης του συμβόλου με βάση όλα τα προηγούμενα δεδομένα. Σε ένα πιο εξειδικευμένο μοντέλο ο υπολογισμός της πιθανότητας γίνετε με βάση τα 10 τελευταία εισαγόμενα δεδομένα.



Στατιστικό Μοντέλο με κωδικοποίηση Huffman

Κατηγορίες Συμπίεσης Δεδομένων

Συμπίεση Χωρίς Απώλειες

- Τα δεδομένα μετά την αποσυμπίεσει των συμπιεσμένων δεδομένων, είναι ακριβώς τα ίδια με αυτά που είχαμε πριν την διαδικασία της συμπίεσης.
- Χρησιμοποιείτε σε περιπτώσεις όπου δεν θέλουμε να χαθεί ούτε ένα μπιτ από την αρχική μορφή των δεδομένων.
- Μονόδρομος στις περιπτώσεις συμπίεσης των εκτελέσιμων προγραμμάτων, αρχείων κειμένου και πηγαίου κώδικα.

Συμπίεση με Απώλειες

- Ένα ποσοστό χαμένων δεδομένων από το αρχικό σχήμα αυτών είναι αποδεκτό, όσο αυτά τα δεδομένα θεωρούντο ασήμαντες λεπτομέρειες.
- Επιτυγχάνετε μεγάλη μείωση σε μέγεθος των δεδομένων διατηρώντας πάντα την ουσία της αρχικής πληροφορίας.
- Επιτυγχάνετε λιγότερος χρόνος αποστολής των δεδομένων αυτών.
- Χρησιμοποιούνται για διαδικασίες αποστολής εικόνων, ήχων και video.

Βήματα αλγορίθμων Συμπίεσης χωρίς απώλειες

1. Δημιουργία του Μοντέλου
2. Δεδομένου αυτού του Μοντέλου κωδικοποιούμε τα δεδομένα.

Τύποι Μοντελοποίησης

Στατιστικό Μοντέλο

Υλοποιείτε με δυο διαφορετικούς τρόπους, τον στατικό (static) και τον προσαρμοστικό (adaptive).

- Στο **στατικό**, διαβάζετε και κωδικοποιείτε ένα σύμβολο την φορά χρησιμοποιώντας την πιθανότητα εμφάνισης του χαρακτήρα. Κάνοντας χρήση έτοιμους πίνακες αντιστοίχισης πιθανοτήτων με σύμβολα.
- Στα **προσαρμοστικά μοντέλα** τα στατιστικά παράγονται και τροποποιούνται διαρκώς κατά την διάρκεια όπου νέοι χαρακτήρες διαβάζονται και κωδικοποιούνται.

Μοντέλο όπου βασίζετε σε Λεξικό

- Στο μοντέλο όπου βασίζετε στο λεξικό, υλοποιείτε η μέθοδος όπου ένας κωδικός αντικαταστεί μια σειρά από σύμβολα.
- Καθώς διαβάζει τα εισαγόμενα δεδομένα, ψάχνει να βρει αντιστοιχία αυτου στο λεξικό. Αν βρεθεί αντιστοιχία, τότε εκτυπώνετε ο δείκτης του σημείου όπου βρέθηκε η αντιστοιχία στο λεξικό, αντί για το κωδικό του συμβόλου.
- Μεγαλύτερο το ταίριασμα, καλύτερη και η συμπίεση.

Αλγόριθμοι λεξικών όπου χρησιμοποιούν την μέθοδο του κυλιόμενου παραθύρου (Sliding Window)

LZ77

- Δημοσιεύτηκε το 1977, εισήγαγε την έννοια του << κυλιόμενου παραθύρου>>
- Χρησιμοποίει σαν λεξικό τα προηγουμένως διαβασμένα δεδομένα της εισόδου, συνήθως σε μέγεθος 2, 4 η 32KB, και με τους δείκτες του να αναζητούνται φράσης σε αυτό.
- Αυτό το μέγεθος δεν είναι μεταβλητό και καθορίζετε στην αρχή του αλγορίθμου.

Μορφή Κυλιόμενου Παράθυρου (Sliding Window)

LZ77 sliding window.

- History Buffer ή Search Buffer : το παρών λεξικό του αλγορίθμου και περιλαμβάνετε από σύμβολα όπου έχουν πρόσφατα εισήχθη και κωδικοποιηθεί.
- Look-ahead buffer :περιλαμβάνει το κείμενο όπου δεν έχει κωδικοποιηθεί ακόμα.
- Στον αλγόριθμο υπάρχει και ένας δείκτης όπου βρίσκετε στο κομμάτι του Lookahead Buffer

Αλγόριθμοι λεξικών όπου χρησιμοποιούν την μέθοδο του κυλιόμενου παραθύρου (Sliding Window) – Συνέχεια LZ77

Βήματα αλγορίθμου:

1. Βρίσκουμε το μεγαλύτερο τμήμα από σύμβολα όπου βρίσκονται σε σειρά στον Lookahead Buffer και ταυτίζονται με αντίστοιχη συμβολοσειρά στο dictionary ή History Buffer.

2. Καταγράφουμε την τριάδα (p,n,c) όπου:
   p : Το σημείο όπου εντός του dictionary αρχίζει η αντίστοιχη ίδια συμβολοσειρά.
   n : Το πλήθος των συμβόλων της συμβολοσειράς όπου βρέθηκε να υπάρχει στο dictionary.
   c : Το σύμβολο μετά το τέλος της συμβολοσειράς όπου ταυτίστηκε στο dictionary και βρίσκετε στον Lookahead Buffer.

3. Εφόσον δεν έχουμε συμπιέσει όλο το μήνυμα, ο δρομέας μετακινείτε κατά n+1 θέσεις δεξιά, δηλαδή αμέσως μετά το σύμβολο c και ξανά εκτελούμε τα βήματα από το την αρχή.

Example of a LZ77 compression sliding window

Line	12	11	10	9	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9		Output
1	(Empty)												a	a	c	a	a	c	a	b	c	a	⟹	(0,0,"**a**")
2	(Empty)											a	a	c	a	a	c	a	b	c	a	b	⟹	(1,1,"**c**")
3	(Empty)									a	a	c	a	a	c	a	b	c	a	b	a	a	⟹	(3,4,"**b**")
4	(Empty)				a	a	c	a	a	c	a	b	c	a	b	a	a	a	c		(Empty)		⟹	(3,3,"**a**")
5	a	a	c	a	a	c	a	b	c	a	b	a	a	a	c	(Empty)							⟹	(12,3,"**$**")
finished																								

LZSS

- 1982, James A. Storer και Thomas G. Szymanski. Βελτίωση του LZ77 αλγορίθμου
- Δυο διαφορές απο τον LZ77
  1. Αντί να στέλνετε κωδικοποιημένη τριάδα (p,n,c) στην έξοδο στέλνετε κωδικοποιημένη δυάδα.
     Η λογική της είναι ολόιδια με την προηγούμενη, μόνο ότι δεν στέλνετε η μεταβλητή c όπου αποθηκεύετε ο χαρακτήρας μετά το τέλος της συμβολοσειράς όπου ταυτίστηκε στο dictionary και βρίσκετε στον Look-ahead Buffer.
  2. Η κωδικοποίηση σε δυάδα (p,n) γίνετε μόνο αν η ακολουθία υπερβαίνει ένα ορισμένο ελάχιστο μήκος
- Τέλος ο LZSS χρησιμοποιεί ενός bit – flag για να προσδιορίσει αν τα επόμενα δεδομένα είναι κανονικά byte η αναφορά σε κωδικοποιημένο ζευγάρι.

DEFLATE

- Σχεδιαστικέ από τον Philip Katz ως μέρος του προγράμματος κωδικοποίησης αρχείων Zip και εφαρμόστηκε στο λογισμικό του με όνομα PKZIP
- Είναι ελεύθερης χρήσης (public domain), όπου επέτρεψε εφαρμογές όπως το Info-Zip’s Zip και Unzip να εμφανιστούν σε μια σειρά από πλατφόρμες.
- Πιο αξιοσημείωτη εφαρμογή του Deflate είναι η zlib , μια φορητή και δωρεάν συμπίεσης βιβλιοθήκη.
- Αυτή η βιβλιθήκη, υλοποιεί τις μορφές αρχείων ZLIB και GZIP, οι οποίες αποτελούν τον πυρήνα των περισσότερων DEFLATE εφαρμογών, συμπεριλαμβανομένου του δημοφιλούς λογισμικού Gzip.
- Ο DEFLATE βασίζετε σε μια παραλλαγή του LZSS συνδυασμένη με κώδικες Huffman.

Αλγόριθμοι λεξικών (Dictionary Algorithms)

LZ78 dictionary compression

- Δεν χρησιμοποιεί το sliding window.
- Χρησιμοποιεί ένα λεξικό όπου έχουμε αποθηκευμένα προηγούμενους χρησιμοποιημένους αλφαριθμητικούς χαρακτήρες.
- Ο αλγόριθμος εισάγει ενός ή μια σειρά από χαρακτήρες του μηνύματος στο λεξικό ελέγχοντας πάντα αν η συγκεκριμένη σειρά υπάρχει στο λεξικό, για να αποφευχθεί η επανάληψη εισαγωγής των ίδιων σειρών χαρακτήρων.

Δυο σημαντικές λειτουργικές διαφορές από τον LZ77

1. Φράσεις κρατιούνται στο λεξικό του LZ78 για όσο χρειαστεί, ανεξάρτητα από το που βρέθηκαν στην ροή των δεδομένων.

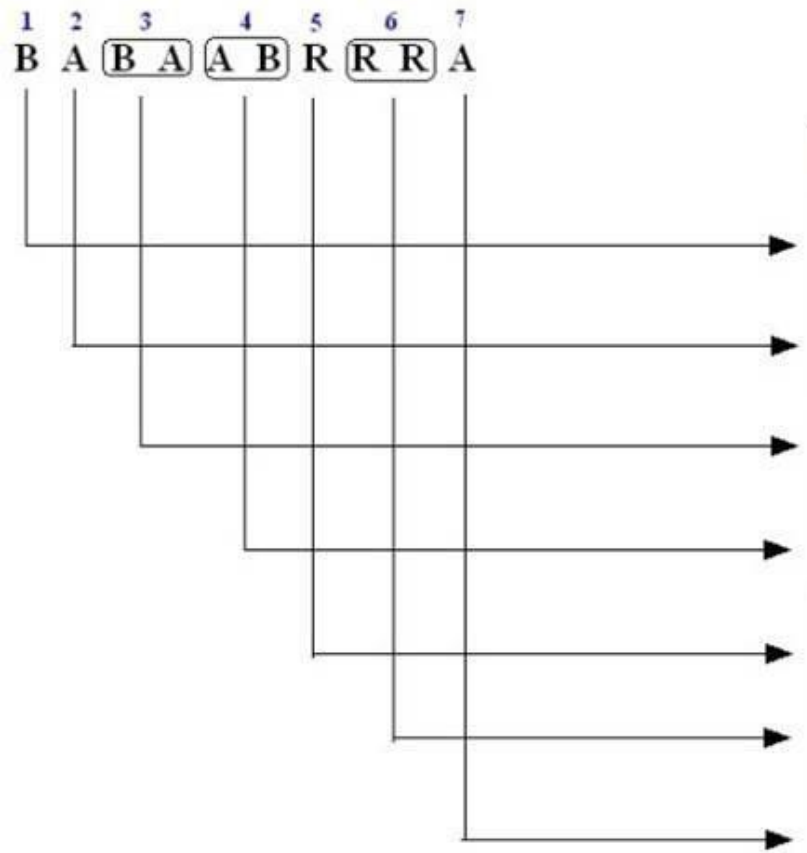
2. Στον LZ78 δεν υπάρχει Lookahead-buffer διότι δεν χρειάζεται να περιορίζει το μέγεθος των string, διότι το λεξικό μπορεί να περιέχει μεγάλα strings όπου μπορούν αποτελεσματικά να αναζητηθούν.

Περιπτώσεις συμπιεσμένων αποτελεσμάτων αλγορίθμου LZ78.

(0, τρέχων χαρακτήρας)	Αν ένας χαρακτήρας δεν βρίσκετε στο Λεξικό
(κωδικήΛέξηΠροθέματος,τρέχων χαρακτήρας)	Αν μια σειρά από χαρακτήρες δεν βρίσκονται αποθηκευμένα στο Λεξικό
(κωδικήΛέξηΠροθέματος,)	Αν μια σειρά από χαρακτήρες η ένας χαρακτήρας βρίσκονται αποθηκευμένα στο Λεξικό

Αλγόριθμοι λεξικών (Dictionary Algorithms) - Συνέχεια LZ78

- Παράδειγμα κωδικοποίησης του μηνύματος **BABAABRRRA.**



	Λεξικό	
Εκτύπωση	Δείκτης	Χαρακτήρες
(0, B)	1	B
(0, A)	2	A
(1, A)	3	BA
(2, B)	4	AB
(0, R)	5	R
(5, R)	6	RR
(2,)		

Αλγόριθμοι λεξικών (Dictionary Algorithms) – LZW

- Προσαρμοστικός αλγόριθμος συμπίεσης δεδομένων χωρίς απώλειες
- Δημιουργήθηκε το 1984 από τους ομώνυμους ερευνητές Abraham Lempel, Jacob Ziv και Terry Welch
- Βελτιομένη έκδοση του αλγορίθμου συμπίεσης LZ78
- Ο αλγόριθμος στηρίζετε στην δημιουργία και συντήρηση ενός λεξικού, στο οποίο αποθηκεύει ζευγάρια από σειρές συμβόλων με τον δείκτη τους
- Το Λεξικό αρχικοποιήτε με τις πρώτες 2^8 = 256 καταχωρίσεις γεμάτες, αυτές οι θέσεις αποτελούνται από όλα τα πιθανά ενός byte σύμβολα (ASCII table) με τους δείκτες τους (από 0 έως 255)
- Το λεξικό ενημερώνετε με μια νέα σειρά συμβόλων μόνο και μόνο αν δεν υπάρχει αντίστοιχη σειρά αποθηκευμένη.

Αλγόριθμοι λεξικών (Dictionary Algorithms) – Συνέχεια LZW

- Ο Αλγόριθμος εκμεταλλεύεται το φαινόμενο της επανάληψης ίδιων ακολουθιών δεδομένων μέσα από την ροή δεδομένων όπου κωδικοποιεί.

- Αποτελεσματικός σε μεγάλες ακολουθίες ίδιων δεδομένων

Συμαντικά Πλεονεκτήματα

1. Δεν αποθηκεύουμε το λεξικό στα συμπιεσμένα δεδομένα, ούτε κάθε άλλο είδος βοηθητικών μεταβλητών για την διαδικασία αποκωδικοποίησης.

2. Δυναμική δημιουργία του ίδιου Λεξικού, στις διαδικασίες κωδικοποίησης και αποκωδικοποίησης.

Βελτιώσεις αλγορίθμου LZW σε σύγκριση του LZ78

1. Ποτέ δεν κωδικοποιείται μονό σύμβολο, πάντα στον LZW κωδικοποιούνται σειρά από σύμβολα.

2. Τα κωδικοποιημένα δεδομένα στον LZW δεν χρειάζεται να αποτελούνται και από βοηθητικά bits για την διαδικασία της αποκωδικοποίησης

Συμπίεση LZW

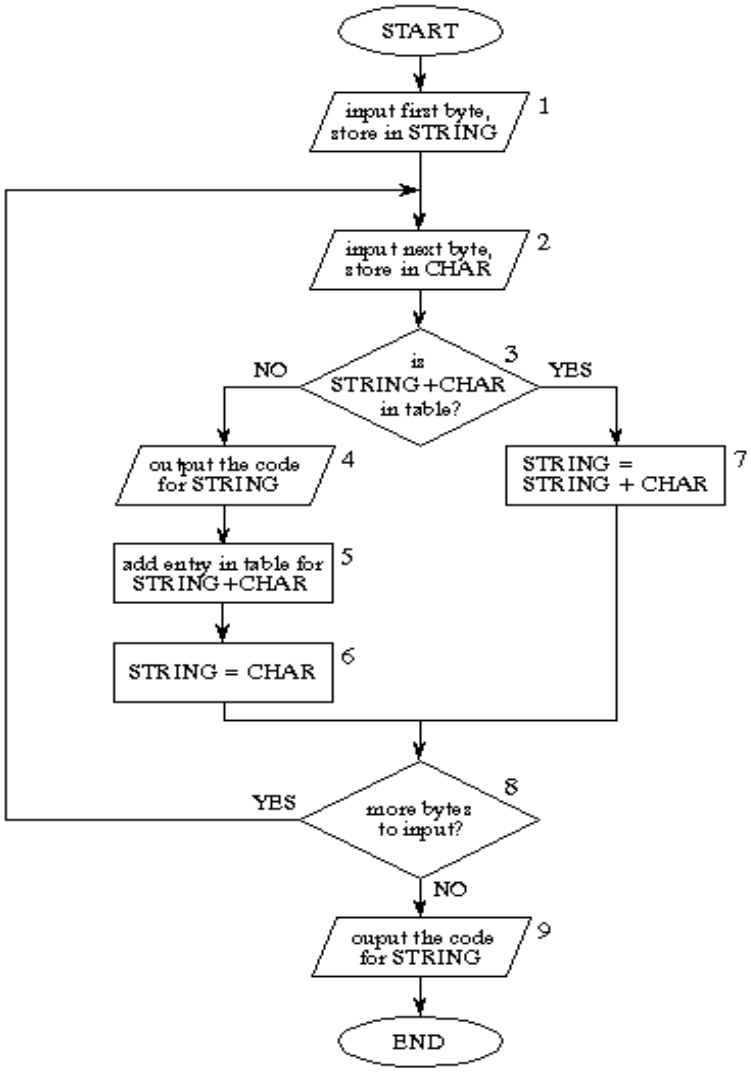
- **Ο Ψευδοκώδικας του αλγορίθμου συμπίεσης:**

```
string p;

char c;
...

p = άδειο string;
Εφόσον (υπάρχουν ακόμα δεδομένα να διαβαστούν)
{
   c = διάβασε τον τρέχον χαρακτήρα;
   Αν (Το Λεξικό εμπεριέχει p+c)
   {
      p = p+c;
   }
   Αλλιώς
   {
      κωδικοποίησε p->index στην κωδικοποιημένη έξοδο;
      εισήγαγε p+c + new index στο Λεξικό
      p = c;
   }
}
κωδικοποίησε p->index στην κωδικοποιημένη έξοδο;
```

- **Το διάγραμμα ροής του αλγορίθμου συμπίεσης LZW δίνετε παρακάτω.**



Συμπίεση LZW - Συνέχεια

- **Τα βήματα όπου ακολουθούνται είναι τα εξής:**

1) Αρχικοποίηση Λεξικού με τα πρώτα 256 ζευγάρια δείκτη-χαρακτήρα
   προηγούμενος Χαρακτήρας = null

2) τρέχων Χαρακτήρας = το επόμενο σύμβολο των εισαγόμενων δεδομένων

3) Έλεγχος αν το αποτέλεσμα της ένωσης των μεταβλητών
   προηγούμενος Χαρακτήρας + τρέχων Χαρακτήρας υπάρχει στο λεξικό.
   α) Αν υπάρχει
      προηγούμενος Χαρακτήρας =
         προηγούμενος Χαρακτήρας + τρέχων Χαρακτήρας
   β) Αν δεν υπάρχει
      i) εκτύπωσε τον δείκτη όπου δείχνει στο λεξικό για την σειρά
         δεδομένων : προηγούμενος Χαρακτήρας
      ii) πρόσθεσε στο λεξικό το νέο ζευγάρι δείκτη – σειρά Δεδομένων
         (προηγούμενος Χαρακτήρας + τρέχων Χαρακτήρας)
      iii) προηγούμενος Χαρακτήρας = τρέχων Χαρακτήρας

4) Όταν τελειώσουν τα σύμβολα εκτύπωσε τον δείκτη όπου δείχνει στο λεξικό
   για τον προηγούμενο Χαρακτήρα.

Παράδειγμα συμπίεσης αρχείου κειμένου LZW

Ένα απλό παράδειγμα συμπίεσης αρχείου κειμένου : wabbawabba

Σύμβολα	w	a	b	b	a	w	a	b	b	a
Σειρά Εμφάνισης	1	2	3	4	5	6	7	8	9	10

Αρχική μορφή Λεξικού.

Δείκτης	Χαρακτήρες
1	a
2	b
3	w

- Επαναλήψεις

1)

P = null, C = w, PC = P + C = w
Το PC υπάρχει στο Λεξικό άρα P = PC = w

Μορφή Λεξικού πρώτης επανάληψης

Δείκτης	Χαρακτήρες
1	a
2	b
3	w

Παράδειγμα συμπίεσης αρχείου κειμένου LZW – Συνέχεια 1

Επαναλήψεις

2)

P = w, C = a, PC = P + C = wa

Το PC δεν υπάρχει στο Λεξικό άρα

Εκτύπωσε τον δείκτη όπου δείχνει στην μεταβλητή P, output : 3

Πρόσθεσε το PC στο λεξικό, νέος δείκτης = 4, νέος χαρακτήρας **wa**

P = C => P = a

Λεξικό μετά την δεύτερη επανάληψη

Δείκτης	Χαρακτήρες
1	a
2	b
3	w
4	wa

3)

P = a, C = b, PC = P + C = ab

Το PC δεν υπάρχει στο Λεξικό άρα

Εκτύπωσε τον δείκτη όπου δείχνει στην μεταβλητή P, output : 1

Πρόσθεσε το PC στο λεξικό, νέος δείκτης = 5, νέος χαρακτήρας ab

P = C => P = b

Λεξικό μετά την τρίτη επανάληψη

Δείκτης	Χαρακτήρες
1	a
2	b
3	w
4	wa
5	ab

Παράδειγμα συμπίεσης αρχείου κειμένου LZW – Συνέχεια 2

Επαναλήψεις

4)

P = b, C = b, PC = P + C = bb
Το PC δεν υπάρχει στο Λεξικό άρα
Εκτύπωσε τον δείκτη όπου δείχνει στην μεταβλητή P, output : 2
Πρόσθεσε το PC στο λεξικό, νέος δείκτης = 6, νέος χαρακτήρας **bb**
P = C => P = b

5)

P = b, C = a, PC = P + C = ba
Το PC δεν υπάρχει στο Λεξικό άρα
Εκτύπωσε τον δείκτη όπου δείχνει στην μεταβλητή P, output : 2
Πρόσθεσε το PC στο λεξικό, νέος δείκτης = 7, νέος χαρακτήρας **ba**
P = C => P = a

6)

P = a, C = w, PC = P + C = aw
Το PC δεν υπάρχει στο Λεξικό άρα
Εκτύπωσε τον δείκτη όπου δείχνει στην μεταβλητή P, output : 1
Πρόσθεσε το PC στο λεξικό, νέος δείκτης = 8, νέος χαρακτήρας **aw**
P = C => P = w

7)

P = w, C = a, PC = P + C = wa
Το PC υπάρχει στο Λεξικό άρα
P = PC => P = wa

8)

P = wa, C = b, PC = P + C = wab
Το PC δεν υπάρχει στο Λεξικό άρα
Εκτύπωσε τον δείκτη όπου δείχνει στην μεταβλητή P, output : 4
Πρόσθεσε το PC στο λεξικό, νέος δείκτης = 9, νέος χαρακτήρας **wab**
P = C => P = b

Παράδειγμα συμπίεσης αρχείου κειμένου LZW – Συνέχεια 3

Επαναλήψεις

9)

P = b, C = b, PC = P + C = bb
Το PC υπάρχει στο Λεξικό άρα
P = PC => P = bb

10)

P = bb, C = a, PC = P + C = bba
Το PC δεν υπάρχει στο Λεξικό άρα
Εκτύπωσε τον δείκτη όπου δείχνει στην μεταβλητή P, output : 6
Πρόσθεσε το PC στο λεξικό, νέος δείκτης = 10, νέος χαρακτήρας **bba**
P = C => P = a

11)

Δεν υπάρχει Άλος χαρακτήρας
Εκτυπώνουμε τον δείκτη του P = a, output : 1

Το τελικό συμπιεσμένο, του κειμένου **wabbawabba** είναι : **3 1 2 2 1 4 6 1**

Τελική μορφή λεξικού

Δείκτης	Χαρακτήρες
1	a
2	b
3	w
4	wa
5	ab
6	bb
7	ba
8	aw
9	wab
10	bba

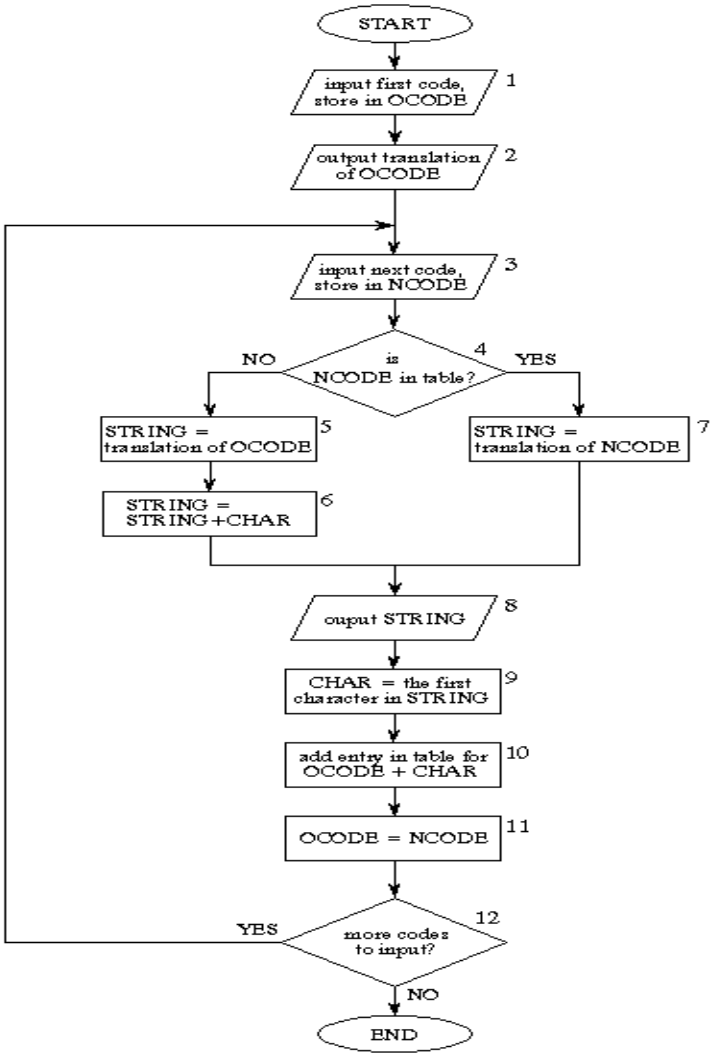
Αποσυμπίεση LZW

- **Ο Ψευδοκώδικας του αλγορίθμου αποσυμπίεσης:**

```
string entry;
char c;
int prevcode, currcode;
...

currentCode = διάβασε τον επόμενο κωδικό;
αποκωδικοποίησε/εκτύπωσε currentCode;
previousCode = currentCode;
Εφόσον (υπάρχουν και άλλα δεδομένα να διαβάσεις)
{
    currentCode = διάβασε τον επόμενο κωδικό;
    Αν (Το Λεξικό εμπεριέχει το
        currentCode->Συμβολοσειρα)
    {
      Εκτύπωσε currentCode->Συμβολοσειρά
      previousCodeValue = previousCode
                              ->Συμβολοσειρά
      currentCodeValue = currentCode->Συμβολοσειρα[0]
                          (πρώτος χαρακτήρας)
      πρόσθεσε στο Λεξικό (previousCodeValue
                            + currentCodeValue)
    }
    Αλλιώς
    {
      currentCodeValue = previousCode
                          ->Συμβολοσειρά]
      πρόσθεσε στο Λεξικό και Εκτύπωσε
        (previousCodeValue + currentCodeValue)
    }
    previousCode = currentCode
}
```

- **Το διάγραμμα ροής του αλγορίθμου αποσυμπίεσης LZW δίνετε παρακάτω.**



Αποσυμπίεση LZW - Συνέχεια

- **Τα βήματα όπου ακολουθούνται είναι τα εξής:**

1) Αρχικοποίηση Λεξικού με τα πρώτα 256 ζευγάρια δείκτη-χαρακτήρα
   προηγούμενος Χαρακτήρας = null
   τρέχων Κωδικός = διάβασε πρώτο κωδικό
   αποκωδικοποίησε και εκτύπωσε αυτόν τον κωδικό

2) προηγούμενος Κωδικός = τρέχων Κωδικός
   τρέχων Κωδικός = διάβασε επόμενο κωδικό

3) Έλεγχος τρέχων Κωδικός.String υπάρχει στο λεξικό
   α) Αν υπάρχει
      i) εκτύπωσε τρέχων Κωδικός.String
      ii) προηγούμενα Σύμβολα = προηγούμενος Κωδικός.String
      iii) τρέχων Σύμβολο = ο πρώτος χαρακτήρας του
                τρέχων Κωδικού.String
      iv) πρόσθεσε τα σύμβολα : προηγούμενα Σύμβολα+ τρέχων Σύμβολο
                στο Λεξικό
   β) Αν δεν υπάρχει
      i) προηγούμενα Σύμβολα = προηγούμενος Κωδικός.String
      ii) τρέχων Σύμβολο = ο πρώτος χαρακτήρας του
                προηγούμενος Κωδικός.String
      iii) πρόσθεσε τα σύμβολα : προηγούμενα Σύμβολα+ τρέχων Σύμβολο
                στο Λεξικό και εκτύπωσε τα

4) Αν υπάρχουν και άλλοι κώδικες πάνε στο βήμα 2
   Αλλιώς Τέλος.

Παράδειγμα συμπίεσης αρχείου κειμένου LZW

Ένα απλό παράδειγμα αποσυμπίεσης αρχείου κειμένου : 3 1 2 2 1 4 6 1, το συμπιεσμένο κείμενο του παραδείγματος συμπίεσης.

Είσοδος

Κωδικοί	3	1	2	2	1	4	6	1
Σειρά Εμφάνισης	1	2	3	4	5	6	7	8

Αρχική μορφή Λεξικού.

Δείκτης	**Χαρακτήρες**
1	**a**
2	**b**
3	**w**

Επαναλήψεις

1)

pw = , cw = 3

Πρώτη φορά εκτέλεσης άρα εκτύπωσε την συμβολοσειρά όπου δείχνει ο δείκτης του cw.

Εκτύπωσε cs.string = w

pw = cw

Παράδειγμα συμπίεσης αρχείου κειμένου LZW – Συνέχεια 1

Επαναλήψεις

2)

pw = 3, cw = 1

Υπάρχει στο Λεξικό το cw.string άρα:
Εκτύπωσε cw.string = a
pw.string = pw.string = w
cw.string = cw.string[0] => cw.string = a
PC = pw.string + cw.string
Πρόσθεσε στο Λεξικό [PC,Δείκτης] = [wa,4]

Λεξικό μετά την δεύτερη επανάληψη

Δείκτης	Χαρακτήρες
1	a
2	b
3	w
4	wa

3)

pw = 1, cw = 2

Υπάρχει στο Λεξικό το cw.string αρα:
Εκτύπωσε cw.string = b
pw.string = pw.string = a
cw.string = cw.string[0] => cw.string = b
PC = pw.string + cw.string
Πρόσθεσε στο Λεξικό [PC,Δείκτης] = [ab,5]

4)

pw = 2, cw = 2
Υπάρχει στο Λεξικό το cw.string άρα:
Εκτύπωσε cw.string = b
pw.string = pw.string = b
cw.string = cw.string[0] => cw.string = b
PC = pw.string + cw.string
Πρόσθεσε στο Λεξικό [PC,Δείκτης] = [bb,6]

Παράδειγμα συμπίεσης αρχείου κειμένου LZW – Συνέχεια 2

Επαναλήψεις

5)

pw = 2, cw = 1

Υπάρχει στο Λεξικό το cw.string άρα:
Εκτύπωσε cw.string = a
pw.string = pw.string = b
cw.string = cw.string[0] => cw.string = a
PC = pw.string + cw.string
Πρόσθεσε στο Λεξικό [PC,Δείκτης] = [ba,7]

6)

pw = 1, cw = 4

Υπάρχει στο Λεξικό το cw.string άρα:
Εκτύπωσε cw.string = wa
pw.string = pw.string = a
cw.string = cw.string[0] => cw.string = w
PC = pw.string + cw.string
Πρόσθεσε στο Λεξικό [PC,Δείκτης+1] = [aw,8]

7)

pw = 4, cw = 6

Υπάρχει στο Λεξικό το cw.string άρα:
Εκτύπωσε cw.string = bb
pw.string = pw.string = wa
cw.string = cw.string[0] => cw.string = b
PC = pw.string + cw.string
Πρόσθεσε στο Λεξικό [PC, Δείκτης +1] = [wab,9]

8)

pw = 6, cw = 1

Υπάρχει στο Λεξικό το cw.string άρα:
Εκτύπωσε cw.string = a
pw.string = pw.string = bb
cw.string = cw.string[0] => cw.string = a
PC = pw.string + cw.string
Πρόσθεσε στο Λεξικό [PC,Δείκτης] = [bba,10]

Παράδειγμα συμπίεσης αρχείου κειμένου LZW – Συνέχεια 3

Το τελικό αποσυμπιεσμένο κείμενο, της κωδικοποιημένης μορφής: 3 1 2 2 1 4 6 1 είναι : wabbawabba

Η τελική μορφή του Λεξιλογίου της διαδικασίας αποσυμπίεσες είναι:

Δείκτης	Χαρακτήρες
1	a
2	b
3	w
4	wa
5	ab
6	bb
7	ba
8	aw
9	wab
10	bba

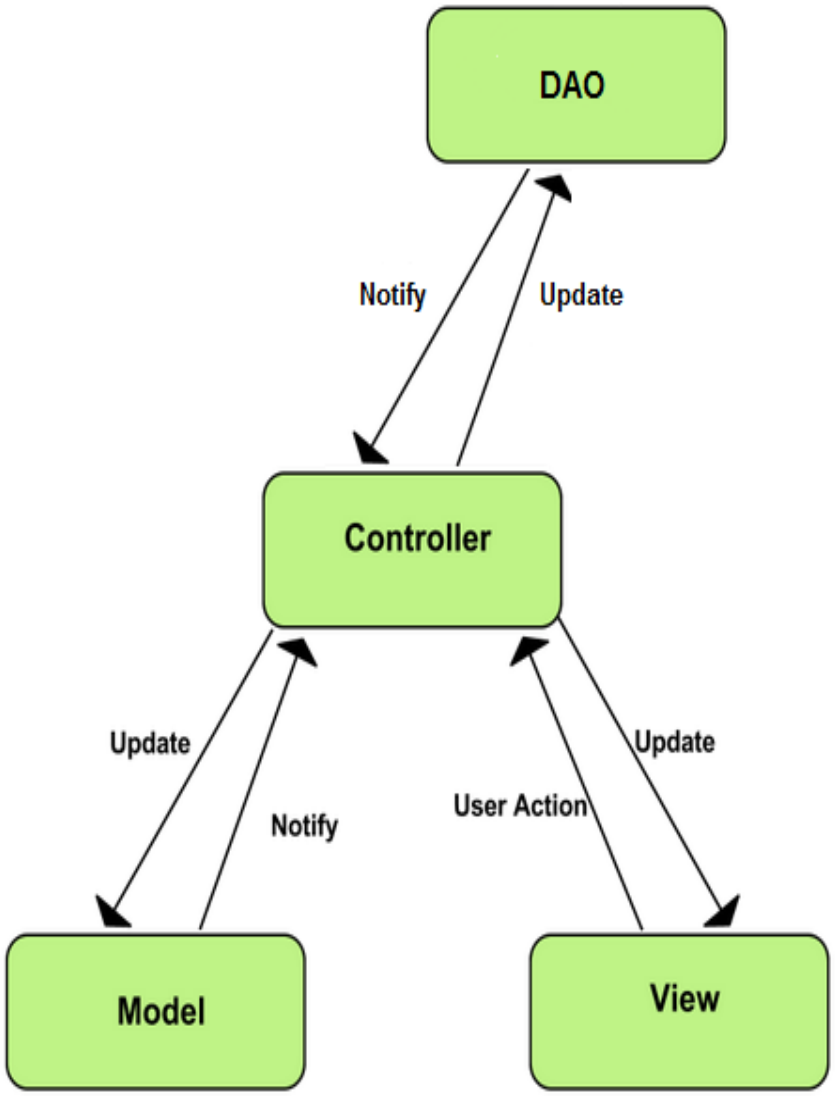
Αρχιτεκτονική Προγράμματος

- **Μοτίβο Σχεδίασης MVC (Model-View-Controller)**

1. Χωρίζει το πρόγραμμα σε 3 κυρια κομμάτια. Το Model, View και Controller.
2. Αποσυνδέει το Model από το View.
3. Καθιστούν τον κώδικα πιο εύκολο στην ανάγνωση και κατανόησή του, αφού μια μεγάλη σειρά ενεργειών χωρίζεται σε πολλές μικρές ενέργειες.
4. Μελλοντικές αλλαγές και επεκτάσεις πάνω στον κώδικα θα είναι πολύ πιο εύκολα υλοποιήσιμες.

- **Μοτίβο σχεδίασης DAO (Data Access Object)**

1. Χρησιμοποιείτε για να αποσυνδεθούν τα πρώτα επίπεδα τις αρχιτεκτονικής του προγράμματος μας με το να επικοινωνούν με εξωτερικού τύπου δεδομένα.



.NET Framework

- Πλατφόρμα όπου χρησιμοποιείτε από το λειτουργικό σύστημα των Windows
- Παρέχει στους προγραμματιστές έναν πιο εύκολο τρόπο να φτιάχνουν εφαρμογές, οι οποίες θα είναι διαλειτουργικές, ασφαλής, με καλύτερη διαχείριση της μνήμης και χειρισμό εξαιρέσεων
- Παρέχει μια μεγάλη βιβλιοθήκη κλάσεων, η οποία χωρίζετε σε δυο μέρη: την βασική βιβλιοθήκη κλάσεων BCL (Base Class Library) και την βιβλιοθήκη κλάσεων της πλατφόρμας FCL (Framework Class Library)

1. BCL : Χαρακτηρίζετε ο πυρήνας του FCL, εμπεριέχονται κλάσεις όπου αποτελούνται από έναν μεγάλο αριθμό συναρτήσεων, με τις οποίες παρέχονται οι θεμελιώδεις λειτουργίες όπως την πρόσβαση αρχείων, χειρισμό αλφαριθμητικών, την μορφοποίηση δεδομένων, συλλογές και πολλά άλλα

2. Η FCL παρέχει παρόμοιες υπηρεσίες με αυτές του API (Windows application programming interface) των Windows. . Αυτή η βιβλιοθήκη κλάσεων ενσωματώνεται με το CLR (Common Language Runtime), όπου είναι υπεύθυνο για την διαχείριση της μνήμης, τον χειρισμό εξαιρέσεων και την ασφάλεια.

❖ Εκμεταλευτήκαμε πλήρος την πλατφόρμα .NET Framework. Η βελτίωση του χρόνου εκτέλεσης και η καλύτερη διαχείριση της μνήμης, επιτεύχθηκε χρησιμοποιώντας κυρίως τις συλλογές δεδομένων Dictionary <TKey,TValue>, List<T>. Επίσης τεράστιο όφελος είχαμε και στην εκμετάλλευση της κλάσης StringBuilder.

Πρόγραμμα LZW

Το Πρόγραμμα αποτελείτε από δυο εκδόσεις του αλγόριθμου LZW

1. Η πρώτη έκδοση του αλγορίθμου συμπίεσης LZW, όπου κωδικοποιεί και ακολουθεί ακριβώς τα βήματα που δείξαμε παραπάνω. Το αποτέλεσμα είναι η αποθήκευση των παραγόμενων κωδικοποιημένων κωδικών στο αρχείο με ένα διαχωριστικό κενό μεταξύ τους.

2. Η δεύτερη έκδοση όπου βελτιώνει τον χρόνο υλοποίησης των μεθόδων, μειώνει την χρήση της μνήμης και βελτιώνει το παραγόμενο συμπιεσμένο.

Βελτιώσεις δεύτερης έκδοσης αλγορίθμου LZW.

1. Το Λεξικό ξανά αρχικοποιείτε με τα 256 πρώτα διαθέσιμα ASCII σύμβολα όταν το Λεξικό περνά το μέγιστο αριθμό από ζευγάρια, όπου μπορεί να καθοριστεί από τον χρήστη κατά την εκτέλεση του προγράμματος.

2. Πλέον δεν αποθηκεύουμε δείκτες λεξικών σαν ακέραιους αριθμούς με κενό, αλά τα μετατρέπουμε στην δυαδική τους μορφή, μήκους ανάλογα με την κατάσταση του Λεξικού και το πλήθος των αποθηκευμένων δεικτών, στην συνέχεια μετατρέπετε όλη η συνολική δυαδική σειρά συμβόλων σε bytes των 8 bits και τέλος αποθηκεύονται τα bytes αυτά στο συμπιεσμένο αρχείο.

Παραδείγματα συμπίεσης κειμένω με την βελτιομένη έκδοση LZW

Με μέγιστο μέγεθος Λεξικού 4095, με Bit = 12.

Περιπλοκότητα Κειμένου	Μέγεθος	Χρόνος Συμπίεσης	Χρόνος Αποσυμπιέσεις	Ποσοστό Συμπιεσμένων Δεδομένων	Αναλογία Συμπίεσης
Καθόλου περίπλοκο	100 KB	0.023 sec	0.004 sec	92,91 %	7,09 %
Πολύ περίπλοκο	100 KB	0.041 sec	0.033 sec	51,20 %	48,8 %
Καθόλου περίπλοκο	7 MB	1.665 sec	0.317 sec	93,55 %	6,45 %
Πολύ περίπλοκο	7 MB	3.399 sec	2.397 sec	51,82 %	48,18 %

Με μέγιστο μέγεθος Λεξικού 65535, με Bit = 16.

Περιπλοκότητα Κειμένου	Μέγεθος	Χρόνος Συμπίεσης	Χρόνος Αποσυμπιέσεις	Ποσοστό Συμπιεσμένων Δεδομένων	Αναλογία Συμπίεσης
Καθόλου περίπλοκο	100 KB	0.022 sec	0.003 sec	93,9 %	6,10 %
Πολύ περίπλοκο	100 KB	0.039 sec	0.025 sec	56,83 %	43,17 %
Καθόλου περίπλοκο	7 MB	3.812 sec	0.133 sec	98,2 %	1,80 %
Πολύ περίπλοκο	7 MB	3.281 sec	1.877 sec	58,74 %	41,26 %

Συμπεράσματα Αποτελεσμάτων

- Όσο ανεβάζουμε τα Bit του Λεξικού, μπορεί να επιτυγχάνουμε καλύτερη συμπίεση στις περιπτώσεις των μεγάλων και περίπλοκων κειμένων με ελάχιστη καλυτέρευση στα μικρά κείμενα, αλλά από την άλλη ο χρόνος συμπίεσης ανεβαίνει.

- Στην συμπίεση κειμένου ο αλγόριθμος ανταποκρίνεται καλύτερα με Λεξικό μεγέθους 12 bit.

- Στις περιπτώσεις συμπίεσης εικόνων και video ο LZW αλγόριθμος επίσης θα είναι βέλτιστος όταν θα συμπιέζει με 12 Bit Λεξικό, διότι αυτά τα αρχεία μπορεί να είναι στην πλειοψηφία μεγάλα αλλά έχουν πολλά επαναλαμβανόμενα δεδομένα.

Ευχαριστώ!

Καρατσίν Ηλίας