

ΠΑΡΟΥΣΙΑΣΗ ΑΣΚΗΣΗΣ 1

Το αρχείο `1.cpp` αφορά την προσέγγιση της προγραμματιστικής λύσης πάνω στο Θέμα 1, και αρχείο `1.exe` είναι η εκτελέσιμη λύση, στην οποία βρίσκεται η υλοποίηση της προσέγγισης. Τόσο το προγραμματιστικό, όσο και το εκτελέσιμο κομμάτι της λύσης, υλοποιήθηκε με τα εργαλεία της γλώσσας C++. Η εκτέλεση του προγράμματος πραγματοποιήθηκε με το Command Prompt & το Windows PowerShell.

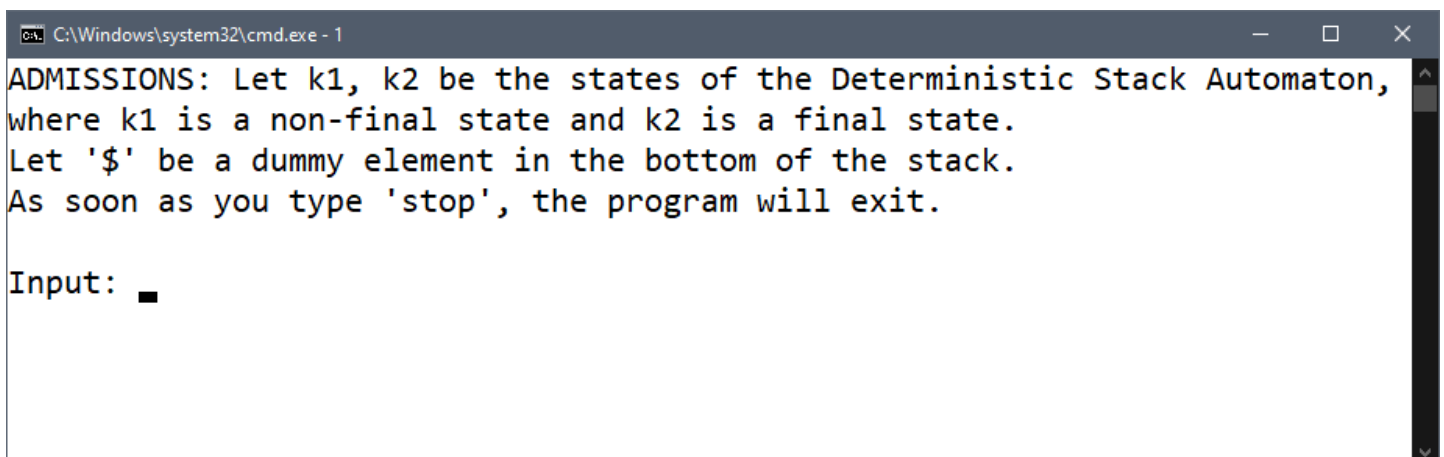
1. ΕΚΤΕΛΕΣΙΜΟ ΚΟΜΜΑΤΙ:

ΕΙΣΑΓΩΓΗ

Με τη χρήση του μεταγλωττιστή GNU και την εντολή `g++ 1.cpp -o 1.exe`, στο λειτουργικό σύστημα των Windows, παρήχθη η εκτελέσιμη λύση.

Το εκτελέσιμο αρχείο μπορεί να ανοιχθεί, είτε από ένα τερματικό, είτε πατώντας διπλό κλικ πάνω στο αρχείο, καθώς έχει προγραμματιστεί έτσι, ώστε να μην κλείνει το αρχείο χωρίς τη βούληση του χρήστη. Η εντολή για την μεταγλώττιση και την παραγωγή του αρχείου `1.exe`, δεν θα έπρεπε να επιφέρει παρατηρήσεις ή σφάλματα.

Όταν γίνει η εκτέλεση του προγράμματος, θα έπρεπε να εμφανίζεται το εξής μενού.



```
C:\Windows\system32\cmd.exe - 1
ADMISSIONS: Let k1, k2 be the states of the Deterministic Stack Automaton,
where k1 is a non-final state and k2 is a final state.
Let '$' be a dummy element in the bottom of the stack.
As soon as you type 'stop', the program will exit.

Input: █
```

ΠΑΡΑΔΟΧΕΣ

Όπως αναγράφεται και στην αρχή της εκτέλεσης του προγράμματος, υπάρχουν κάποιες παραδοχές, που πρέπει οπωσδήποτε να αναφερθούν, ώστε να γίνει κατανοητή η λύση. **Οι παραδοχές έχουν ως εξής:**

- Το πρόγραμμα θα δέχεται είσοδο μόνο από το τερματικό και όχι μέσω εξωτερικού αρχείου.
- Αποδεκτοί χαρακτήρες εισόδου είναι μόνο οι λατινικοί.
- Κάθε είσοδος μετατρέπεται, αυτόματα, από κεφαλαία γράμματα σε πεζά.
- Θεωρούμε k_1 μια μη τερματική κατάσταση του Αυτομάτου Στοίβας.
- Θεωρούμε k_2 μια τερματική κατάσταση του Αυτομάτου Στοίβας.
- Κάθε φορά, που αναγνωρίζεται μία είσοδος, βάζουμε στην Στοίβα ένα βοηθητικό αντικείμενο, που το ονομάζουμε '\$'.
- Το πρόγραμμα θα κλείνει, αν και μόνο αν ο χρήστης εισάγει την έκφραση 'stop'. (Βάσει της τρίτης παραδοχής, **εκφράσεις όπως π.χ. 'StOp' ή 'STOP' είναι αποδεκτές**)

ΑΝΑΛΥΣΗ ΕΙΣΟΔΟΥ

Μία έκφραση, αποτελούμενη μόνο από χαρακτήρες 'x' και 'y', είναι αποδεκτή και το πρόγραμμα θα προχωρήσει στη διαδικασία της ανάλυσης. Όπως αναφέρθηκε και πιο πριν, όποια κι αν είναι η είσοδος του χρήστη, αυτή θα μετατραπεί σε πεζά γράμματα. Οπότε, εκφράσεις όπως "XXXYYY" ή "xxxYYY", δεν θα αποτελέσουν πρόβλημα.

Το πρόγραμμα θα αναδείξει αναλυτικά τις ενέργειες στις οποίες προβαίνει, για την αποδοχή ή την απόρριψη της έκφρασης. **Κάθε φορά που αναλύεται μία έκφραση το πρόγραμμα προβάλλει:**

- 1) Τον αριθμό επανάληψης.
- 2) Την κατάσταση του Αυτομάτου Στοίβας.
- 3) Το περιεχόμενο της Στοίβας.
- 4) Το υπόλοιπο εισόδου.

Για λόγους υπόδειξης, **δίδονται τα παρακάτω τέσσερα παραδείγματα** (δύο ορθών και δύο μη ορθών) εισόδων.

ΟΡΘΕΣ ΕΙΣΟΔΟΙ:

Υπόδειξη 1: Πρώτο παράδειγμα ορθής εισόδου.

Input: xxxxyyyy

1. Current State: k1, Stack Items: x\$, Remaining Input: xxxyyyy
2. Current State: k1, Stack Items: xx\$, Remaining Input: xyyyy
3. Current State: k1, Stack Items: xxx\$, Remaining Input: yyyyy
4. Current State: k1, Stack Items: xxxx\$, Remaining Input: yyyy
5. Current State: k1, Stack Items: xxx\$, Remaining Input: yyy
6. Current State: k1, Stack Items: xx\$, Remaining Input: yy
7. Current State: k1, Stack Items: x\$, Remaining Input: y
8. Current State: k2, Stack Items: \$, Remaining Input: (empty)

String 'xxxxyyyy' is accepted.

Υπόδειξη 2: Δεύτερο παράδειγμα ορθής εισόδου.

Input: xxxxyyxyyyy

1. Current State: k1, Stack Items: x\$, Remaining Input: xxxyyxyyyy
2. Current State: k1, Stack Items: xx\$, Remaining Input: xxyyxyyyy
3. Current State: k1, Stack Items: xxx\$, Remaining Input: xyxyxyyyy
4. Current State: k1, Stack Items: xxxx\$, Remaining Input: yyxyxyyyy
5. Current State: k1, Stack Items: xxx\$, Remaining Input: yxyyyy
6. Current State: k1, Stack Items: xx\$, Remaining Input: xxyyyy
7. Current State: k1, Stack Items: xxx\$, Remaining Input: yyyyy
8. Current State: k1, Stack Items: xxxx\$, Remaining Input: yyyy
9. Current State: k1, Stack Items: xxx\$, Remaining Input: yyy
10. Current State: k1, Stack Items: xx\$, Remaining Input: yy
11. Current State: k1, Stack Items: x\$, Remaining Input: y
12. Current State: k2, Stack Items: \$, Remaining Input: (empty)

String 'xxxxyyxyyyy' is accepted.

Όλες οι παραπάνω εισοδοί, βρέθηκαν στην (τερματική) κατάσταση k₂, πάνω στην λήξη της ανάλυσης της έκφρασης. Παρατηρείται επίσης, ότι σε κάθε ορθή έκφραση, η στοίβα έχει ως μοναδικό στοιχείο το '\$' στο τελευταίο βήμα (αυτό θα εξηγηθεί παρακάτω).

Ωστόσο, αξίζει να δούμε τι γίνεται, όταν βάλουμε, για είσοδο, μία μη ορθή έκφραση.

ΜΗ ΟΡΘΕΣ ΕΙΣΟΔΟΙ

Υπόδειξη 3: Πρώτο παράδειγμα μη ορθής εισόδου.

Input: xxxxyyy

1. Current State: k_1 , Stack Items: $x\$$, Remaining Input: xxxxyyy
2. Current State: k_1 , Stack Items: $xx\$$, Remaining Input: xxxyyy
3. Current State: k_1 , Stack Items: $xxx\$$, Remaining Input: xxyyy
4. Current State: k_1 , Stack Items: $xxxx\$$, Remaining Input: xyyy
5. Current State: k_1 , Stack Items: $xxxxx\$$, Remaining Input: yyy
6. Current State: k_1 , Stack Items: $xxxx\$$, Remaining Input: yy
7. Current State: k_1 , Stack Items: $xxx\$$, Remaining Input: y
8. Current State: k_1 , Stack Items: $xx\$$, Remaining Input: (empty)

String 'xxxxyyy' is NOT accepted.

Error message: Stack still has items to iterate.

Υπόδειξη 4: Δεύτερο παράδειγμα μη ορθής εισόδου.

Input: xyxyyyxx

1. Current State: k_1 , Stack Items: $x\$$, Remaining Input: xyxyyyxx
 2. Current State: k_1 , Stack Items: $xx\$$, Remaining Input: yxyyyxx
 3. Current State: k_1 , Stack Items: $x\$$, Remaining Input: yyyxx
 4. Current State: k_1 , Stack Items: $\$$, Remaining Input: yyxx
- Tried to pop the element '\$'. Cannot continue.

String 'xyxyyyxx' is NOT accepted.

Error message: Tried to pop the element '\$'.

Καμμία από τις παραπάνω εισόδους δεν βρέθηκαν στο τέλος της ανάλυσης σε κατάσταση k_2 , πράγμα που σημαίνει ότι αυτές οι εισοδοί δεν μπορούν να είναι αποδεκτές. Στην k_2 , θα βρισκόμαστε αν και μόνο αν στο τέλος έχουμε στην στοίβα μόνο το στοιχείο '\$', αλλά παράλληλα δεν εκκρεμεί και υπόλοιπο εισόδου. Π.χ., στο δεύτερο παράδειγμα, στο βήμα 4, μολονότι έχουμε μοναδικό στοιχείο το '\$', εκκρεμεί το υπόλοιπο εισόδου, άρα η κατάσταση παραμένει k_1 .

ΕΣΦΑΛΜΕΝΕΣ ΕΙΣΟΔΟΙ

Κάθε φορά που ο χρήστης εισάγει μία είσοδο, που δεν αποτελείται μονάχα από χαρακτήρες 'x' και 'y', το πρόγραμμα δεν βαίνει στην διαδικασία ανάλυσης. Ωστόσο, δεν τερματίζεται! Ο χρήστης μπορεί να δώσει όσες εισόδους θέλει, καθώς το πρόγραμμα δεν τερματίζεται ποτέ, παρά μόνον κατόπιν εντολής του χρήστη.

Υπόδειξη 5: Παραδείγματα μη αναγνωρίσιμων εκφράσεων.

```
Input: TESTING
Incorrect Input. Only x or y characters are allowed. If you want to exit, type 'stop'.

Input: testing
Incorrect Input. Only x or y characters are allowed. If you want to exit, type 'stop'.

Input: test
Incorrect Input. Only x or y characters are allowed. If you want to exit, type 'stop'.

Input:
```

ΟΡΘΟΣ ΤΕΡΜΑΤΙΣΜΟΣ

Αν ο χρήστης θέλει να τερματίσει την λειτουργία του προγράμματος, με τον βέλτιστο δυνατό τρόπο, τότε θα πρέπει να γράψει την λέξη "stop". Όταν γίνεται αυτό, το πρόγραμμα τερματίζεται με κωδικό εξόδου 0, πράγμα που εξασφαλίζει τη ασφαλέστερη δυνατή λήξη του προγράμματος. Ο χρήστης είναι πιθανό να δει και ένα ανάλογο μήνυμα, κατόπιν της λήξης.

Υπόδειξη 6: Ορθός τερματισμός του προγράμματος.

```
As soon as you type 'stop', the program will exit.
```

```
Input: stop
Program stopped regularly.
```

2. ΠΡΟΓΡΑΜΜΑΤΙΣΤΙΚΗ ΥΛΟΠΟΙΗΣΗ:

ΛΟΓΙΚΗ

Για να διευκολυνθεί ο χειρισμός του προγράμματος, **θα πρέπει το πρόγραμμα να μην τερματίζεται, όταν τελειώνει η ανάλυση της έκφρασης. Αυτό σημαίνει ότι ο χρήστης μπορεί να επαναλάβει όσες φορές επιθυμεί τη διαδικασία ανάλυσης εισόδου, χωρίς να χρειάζεται να ξανά-τρέξει το πρόγραμμα από την αρχή.** Με ένα σχέδιο, το πρόγραμμα θα μπορεί να ταυτοποιηθεί με το εξής διάγραμμα:



Έναν ακόμη παράγοντα διευκόλυνσης χειρισμού θα αποτελέσει **η μετατροπή της εισόδου του χρήστη σε πεζά γράμματα.** Αυτό δεν θα διευκολύνει **μόνχα** **χρήστη, αλλά και την κωδικοποίηση,** καθώς, θα χρειάζονται σαφέστερα λιγότεροι έλεγχοι, κανονικές εκφράσεις, μηνύματα σφάλματος και λοιποί άλλοι παράγοντες.

Η ανάλυση της έκφρασης θα γίνεται γράμμα-προς-γράμμα, ώστε να έχουμε πλήρη εικόνα της έκφρασης. Κατά τη διαδικασία της ανάλυσης, θα μπορούμε να βάζουμε χαρακτήρες ίδιους με την έκφραση. **Από τη στιγμή που έχουμε μόνο δύο πιθανούς χαρακτήρες ('x' και 'y'), μία «εύκολη» λύση είναι, όποτε πέφτουμε σε**

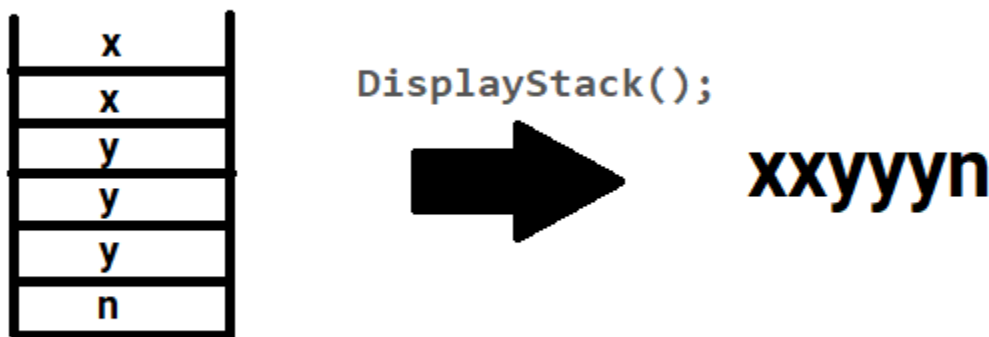
ένα γράμμα 'x', να βάζουμε και στη στοίβα τον χαρακτήρα 'x'. Αντιθέτως, όταν θα πέφτουμε σε ένα γράμμα 'y', θα βγάζουμε ένα στοιχείο από τη στοίβα. Για να επιτυγχάνεται αυτό, θα πρέπει να προσθέσουμε, στο βάθος της στοίβας, κι ένα βοηθητικό στοιχείο, το οποίο θα μας βοηθάει να εγκρίνουμε ή να απορρίπτουμε την έκφραση, ανάλογα με την παρουσία του στη κορυφή της στοίβας(ή την απουσία του) στο τέλος της διαδικασίας.

ΕΦΑΡΜΟΓΗ

Όλα τα παραπάνω επιτυγχάνονται με βοηθητικές βιβλιοθήκες της γλώσσας C++, ώστε να βελτιστοποιηθεί η κωδικοποίηση. **Συγκεκριμένα, οι κατ' ελάχιστον τέσσερις βιβλιοθήκες που χρειαζόμαστε, είναι:**

1. Για την επικοινωνία υπολογιστή/χρήστη (in out stream ή iostream).
2. Για να την υλοποίηση της στοίβας.
3. Η <string>, που θα βοηθήσει στην επεξεργασία της έκφρασης.
4. Για ελέγχους με κανονικές εκφράσεις.

Ένα πρόβλημα, που δημιουργείται με την βιβλιοθήκη της στοίβας, είναι ότι δεν παρέχεται συνάρτηση προβολής των στοιχείων μιας στοίβας. Οπότε, δημιουργήθηκε μία από εμάς. **Η συνάρτηση παίρνει ένα αντίγραφο της στοίβας (ώστε να μην τροποποιηθεί η πρωτότυπη) και προβάλλει τα στοιχεία ένα-προς-ένα, ξεκινώντας από την κορυφή.**



Ύστερα, αφού το πρόγραμμα έχει δείξει τις παραδοχές και έχει πάρει την είσοδο του χρήστη, κάνει έναν έλεγχο με τα τρία πιθανά σενάρια, που έχουν επεξηγηθεί παραπάνω. **Συγκεκριμένα, ελέγχουμε τα εξής:**

- Η πρώτη περίπτωση είναι μια έκφραση αποτελούμενη από χαρακτήρες ‘x’ και ‘y’. Εδώ έρχεται στο προσκήνιο η βιβλιοθήκη των κανονικών εκφράσεων. **Συγκεκριμένα, με την κανονική έκφραση “[xy]+”, βλέπουμε αν υπάρχουν αποκλειστικά χαρακτήρες είτε ‘x’, είτε ‘y’, τουλάχιστον μία φορά.**
- Ύστερα, ελέγχουμε αν ο χρήστης έχει γράψει ακριβώς την έκφραση “stop”. Σε αυτήν την περίπτωση, **το πρόγραμμα τερματίζεται με ασφαλή τρόπο, με κωδικό εξόδου 0.**
- Αν δεν υποθέσουμε σε κάποιες από τις δύο παραπάνω περιπτώσεις, αναζητάμε πάλι είσοδο από το χρήστη και κάνουμε ξανά τους ελέγχους.

ΑΝΑΛΥΣΗ ΕΚΦΡΑΣΗΣ

Αν έχουμε φτάσει στο σημείο της ανάλυσης της έκφρασης, σημαίνει ότι θα χρησιμοποιήσουμε τη στοίβα. **Άρα, πριν ξεκινήσει το οτιδήποτε, πρέπει να αρχικοποιήσουμε τη στοίβα, η οποία θα κρατάει χαρακτήρες μέσα της. Αμέσως μετά, βάζουμε τον βοηθητικό χαρακτήρα ‘\$’ εντός της στοίβας.** Παρακάτω, θα εξηγηθεί περεταίρω η χρήση του, αλλά και η βοήθεια που θα μας προσφέρει.

Για να αναλύσουμε μια έκφραση αποτελούμενη από ‘x’ και ‘y’, πρέπει πρώτα να την δούμε κατά γράμμα. **Άρα, θα φτιάξουμε μια επανάληψη, που εκτελείται ίσες φορές με τον αριθμό των γραμμάτων της έκφρασης.** Αν, για παράδειγμα έχουμε την έκφραση “xxxxxyyyyy”, θα φτιάξουμε μια δομή που θα επαναλαμβάνεται δέκα φορές.

Εντός της προαναφερθείσας επανάληψης, υπάρχουν δύο πιθανότητες: να πέσουμε σε χαρακτήρα ‘x’, ή ‘y’. **Όταν πέφτουμε σε χαρακτήρα ‘x’, τον βάζουμε στη στοίβα. Αντιθέτως, όταν πέφτουμε σε χαρακτήρα ‘y’, βγάζουμε το κορυφαίο στοιχείο της στοίβας.** Ύστερα των ελέγχων, προβάλλουμε στον χρήστη την παρούσα κατάσταση, το περιεχόμενο της στοίβας και το υπόλοιπο της εισόδου. Το υπόλοιπο της εισόδου, το αποθηκεύουμε σε άλλη μεταβλητή και το

προβάλλουμε με την μέθοδο του `substring`, ώστε να μην τροποποιηθεί το ίδιο το `string` της εισόδου.

Όταν τελειώσει η επανάληψη, θα μας βοηθήσει αρκετά ο χαρακτήρας '\$' με την παρουσία του. Συγκεκριμένα, θα ελέγξουμε το κορυφαίο στοιχείο της στοίβας. Θα εγκρίνουμε την έκφραση, αν και μόνο αν το κορυφαίο στοιχείο της στοίβας είναι το '\$' και η είσοδος έχει διαβαστεί. Σε οποιαδήποτε άλλη περίπτωση, η έκφραση απορρίπτεται, διότι δεν θα υπάρχει ίσος αριθμός 'x' και 'y' σε αυτή, ή θα υπάρχει, αλλά τα 'y' θα είναι περισσότερα από τα 'x' από τα αριστερά στα δεξιά.

ΤΕΛΟΣ ΠΑΡΟΥΣΙΑΣΗΣ ΑΣΚΗΣΗΣ 1.

Σύνταξη παρουσίασης και δημιουργία άσκησης:
Γεώργιος Σεϊμένης Π19204
