Δομές Δεδομένων

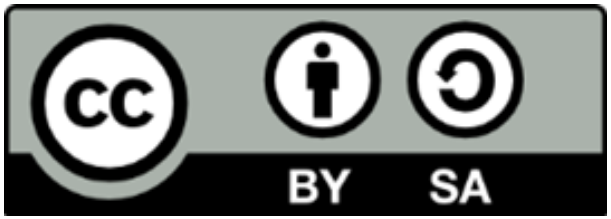
Ενότητα 2: Στοίβες – Εισαγωγή-Υλοποίηση ΑΤΔ Στοίβα με Πίνακα-Εφαρμογή Στοίβας: Αντίστροφη Πολωνική Γραφή

Καθηγήτρια Μαρία Σατρατζέμη

Τμήμα Εφαρμοσμένης Πληροφορικής

Άδειες Χρήσης

- Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons.
- Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τύπου άδειας χρήσης, η άδεια χρήσης αναφέρεται ρητώς.



Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Μακεδονίας» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Σκοποί ενότητας

- Εισαγωγή στην έννοια της στοίβας
- Επίλυση προβλημάτων με οργάνωση των δεδομένων με στοίβα
- Κατανόηση της υλοποίησης της στοίβας με πίνακα
- Μειονεκτήματα της υλοποίησης της στοίβας με πίνακα
- Να επιδείξει μια Εφαρμογή Στοίβας για την επίλυση του προβλήματος της Αντίστροφης Πολωνικής Γραφής και την υλοποίηση του αλγόριθμου με χρήση στοίβας.

Περιεχόμενα ενότητας[1]

- Η έννοια της στοίβας
- Αλγόριθμος μετατροπής δεκαδικού σε δυαδικό
- Ορισμός της στοίβας
- Βασικές λειτουργίες
- Ορισμός του ΑΤΔ Στοίβα
- Υλοποίηση αλγορίθμου μετατροπής
- Υλοποίηση στοίβας με πίνακα & δομής
- Υλοποίηση των λειτουργιών της στοίβας
- Δημιουργία κενής στοίβας
- Ώθηση στοιχείου

Περιεχόμενα ενότητας[2]

- Απώθηση στοιχείου
- Έλεγχος κενής / γεμάτης στοίβας
- Ενδοθεματική & μεταθεματική έκφραση
- Αντίστροφη Πολωνική Γραφή (RPN)
- Χρήση στοίβας για τον υπολογισμό της RPN
- Αλγόριθμος υπολογισμού RPN
- Χρήση στοίβας για τη μετατροπή Ενδοθεματικής σε RPN
- Αλγόριθμος μετατροπής Ενδοθεματική σε RPN

Η έννοια της στοίβας [1]

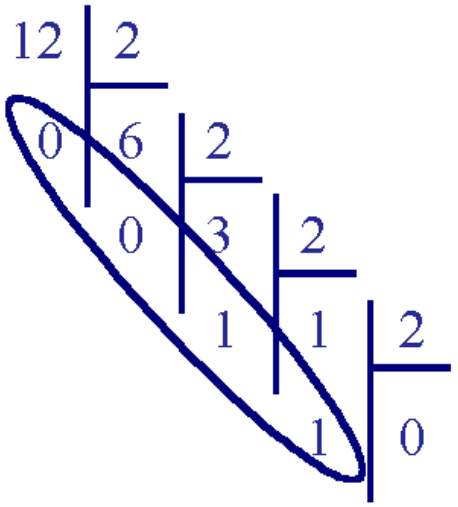
- Για να γίνει κατανοητή η έννοια της στοίβας ας θεωρήσουμε το πρόβλημα αποθήκευσης στη μνήμη του υπολογιστή αριθμών που παριστάνονται στο δεκαδικό σύστημα:
- Οι αριθμοί που παριστάνονται στο δεκαδικό σύστημα, προκειμένου να αποθηκευτούν στη μνήμη του υπολογιστή πρέπει να μετατραπούν στο δυαδικό.

Η έννοια της στοίβας [2]

- Μια τέτοια μετατροπή συνεπάγεται διαδοχικές διαιρέσεις με τον αριθμό 2 και ένα τρόπο αποθήκευσης των ακέραιων υπολοίπων των διαιρέσεων αυτών, ώστε να προκύψει στο τέλος ο δυαδικός αριθμός.

Παράδειγμα: μετατροπή του δεκαδικού 12 σε δυαδικό [1]

- διαιρούμε πρώτα το 12 με το 2 και έχουμε πηλίκο 6 και υπόλοιπο **0**
- στη συνέχεια, διαιρούμε το πηλίκο 6 με το 2 και έχουμε νέο πηλίκο 3 και νέο υπόλοιπο **0**
- το νέο πηλίκο 3 το διαιρούμε πάλι με 2 και παίρνουμε πηλίκο 1 και ακέραιο υπόλοιπο **1**
- τέλος, διαιρούμε το 1 με 2 και έχουμε πηλίκο 0 και ακέραιο υπόλοιπο **1**.



Παράδειγμα: μετατροπή του δεκαδικού 12 σε δυαδικό [2]

- Ύστερα από αυτές τις διαιρέσεις μπορούμε να σχηματίσουμε το δυαδικό αριθμό που αντιστοιχεί στο δεκαδικό 12, γράφοντας τα ακέραια υπόλοιπα των διαιρέσεων το ένα δίπλα στο άλλο, ξεκινώντας από το τελευταίο και προχωρώντας προς το πρώτο.
- Επομένως, ο δυαδικός αριθμός που αντιστοιχεί στο δεκαδικό **12** είναι ο **1100**.

Στοίβα: μετατροπή δεκαδικού σε δυαδικό

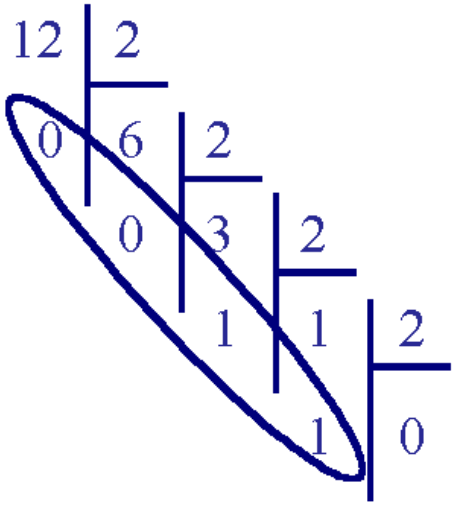
- Είναι φανερό ότι για την επίλυση του παραπάνω προβλήματος χρειάζεται να

  "στοιβάξουμε" τα ακέραια υπόλοιπα των διαιρέσεων

- και στο τέλος

  να τα ανακτήσουμε από τη στοίβα,

- παίρνοντας πρώτα το τελευταία αποθηκευμένο στοιχείο και προχωρώντας προς το πρώτο



Αλγόριθμος μετατροπής δεκαδικού δε δυαδικό [1]

/*Δέχεται: Θετικό ακέραιο αριθμό Number.

Λειτουργία: Μετατρέπει τον αριθμό Number από δεκαδικό σε δυαδικό.

Έξοδος: Η δυαδική αναπαράσταση του Number.*/

Αλγόριθμος μετατροπής δεκαδικού δε δυαδικό [2]

Δημιούργησε μια κενή στοίβα για την αποθήκευση των υπολοίπων

Όσο Number ≠ 0 επανάλαβε

Υπολόγισε το Ακέραιο Υπόλοιπο της διαίρεσης του Number με 2

Εισήγαγε το Ακέραιο Υπόλοιπο στην κορυφή της στοίβας των ακέραιων υπολοίπων

Αντικατέστησε τον αριθμό Number με το ακέραιο πηλίκο της διαίρεσης του Number με 2

Τέλος_επανάληψης

Αλγόριθμος μετατροπής δεκαδικού δε δυαδικό [3]

Όσο η στοίβα των ακέραιων υπολοίπων δεν είναι κενή επανάλαβε

Διάγραψε το Ακέραιο Υπόλοιπο από την κορυφή της στοίβας των υπολοίπων

Εμφάνισε το Ακέραιο Υπόλοιπο

Τέλος_επανάληψης

Ορισμός της στοίβας [1]

- Από τον παραπάνω αλγόριθμο φαίνεται ότι χρειαζόμαστε μια δομή δεδομένων με την οποία
- να αποθηκεύουμε στοιχεία με τη σειρά και
- να τα ανακτούμε με αντίστροφη σειρά,
- δηλαδή χρειαζόμαστε μια "τελευταίος μέσα - πρώτος έξω" (Last In - First Out, LIFO) δομή.
- Μια τέτοια δομή ονομάζεται **στοίβα (stack).**

Ορισμός της στοίβας [2]

- Μια στοίβα είναι μια **"τελευταίος μέσα - πρώτος έξω"** δομή και αποτελείται από μια συλλογή δεδομένων στην οποία όλες οι εισαγωγές και διαγραφές δεδομένων γίνονται στην μία άκρη της, που ονομάζεται **κορυφή (top)**.

Βασικές λειτουργίες στοίβας

- Δημιουργία μιας κενής στοίβας (CreateStack)
- Έλεγχος αν μια στοίβα είναι κενή (EmptyStack)
- Ανάκτηση και διαγραφή του στοιχείου που βρίσκεται στην κορυφή της στοίβας (Pop)
- Εισαγωγή νέου στοιχείου στην κορυφή της στοίβας (Push).
- Η λειτουργία ανάκτησης και διαγραφής του κορυφαίου στοιχείου της στοίβας ονομάζεται συνήθως **απώθηση (pop)** και
- η λειτουργία εισαγωγής ενός νέου στοιχείου στην κορυφή της στοίβας ονομάζεται **ώθηση (push)**.

Ορισμός του ΑΤΔ Στοίβα [1]

Συλλογή στοιχείων δεδομένων:

Μια συλλογή στοιχείων δεδομένων με γραμμική διάταξη προσπελάσιμη μόνο από μια θέση, που ονομάζεται **κορυφή της στοίβας**.

Βασικές λειτουργίες:

Δημιουργία κενής στοίβας (CreateStack)

Λειτουργία: Δημιουργεί μια κενή στοίβα.
Επιστρέφει: Μια κενή στοίβα.

Έλεγχος αν μια στοίβα είναι κενή (EmptyStack)

Δέχεται: Μια στοίβα.
Λειτουργία: Ελέγχει αν η στοίβα είναι κενή.
Επιστρέφει: TRUE, αν η στοίβα είναι κενή
FALSE διαφορετικά.

Ορισμός του ΑΤΔ Στοίβα [2]

Ανάκτηση και διαγραφή του στοιχείου που βρίσκεται στην κορυφή της στοίβας (Pop)
Δέχεται: Μια στοίβα.
Λειτουργία: Ανακτά και διαγράφει το στοιχείο στην κορυφή της στοίβας.
Επιστρέφει: Κορυφαίο στοιχείο της στοίβας και την τροποποιημένη στοίβα.

Εισαγωγή νέου στοιχείου στην κορυφή της στοίβας (Push)
Δέχεται: Μια στοίβα και ένα στοιχείο δεδομένων.
Λειτουργία: Εισάγει το στοιχείο στην κορυφή της στοίβας.
Επιστρέφει: Την τροποποιημένη στοίβα.

Χρήση του ΑΤΔ Στοίβα

- Αν υποθέσουμε ότι ο παραπάνω ΑΤΔ έχει υλοποιηθεί έτσι ώστε οι διαδικασίες CreateStack, Pop και Push δηλωμένες ως εξής:
- void CreateStack(StackType *Stack);
- void Push(StackType *Stack, StackElementType Item);
- void Pop(StackType *Stack, StackElementType *Item);
- και η boolean συνάρτηση EmptyStack δηλωμένη ως:
- boolean EmptyStack(StackType Stack);

Υλοποίηση Αλγορίθμου μετατροπής

```
CreateStack(&StackOfRemainders);
while (Number!= 0)
  {
     Remainder=Number%2;
     Push(&StackOfRemainders,Remainder);
     Number=Number / 2;
  }
while (! EmptyStack(StackOfRemainders))
{
      Pop(&StackOfRemainders,&Remainder);
      printf("%d",Remainder);
  }
```

Υλοποίηση του ΑΤΔ στοίβα με πίνακα

Για να υλοποιήσουμε μια στοίβα, είναι απαραίτητο να επιλέξουμε μια δομή για την αποθήκευση των στοιχείων της στοίβας.

Εφόσον μια στοίβα είναι μια ακολουθία στοιχείων δεδομένων, μπορούμε να χρησιμοποιήσουμε έναν **πίνακα** για την αποθήκευση των στοιχείων αυτών,

- με κάθε στοιχείο της στοίβας να καταλαμβάνει μια θέση στον πίνακα
- και η θέση 1 να λειτουργεί ως η κορυφή της στοίβας.

Παράδειγμα υλοποίησης στοίβας με πίνακα -1-

Element

θέση	αριθμός
1	**0**
2	
3	
4	
. . .	
k	

- Αφού διαιρέσουμε το 12 με το 2, αποθηκεύουμε τον αριθμό 0, δηλαδή το υπόλοιπο της διαίρεσης, στην θέση 1 ενός πίνακα Element.
- Έτσι,λοιπόν, στην θέση Element[1] βρίσκεται ο αριθμός 0, όπως φαίνεται στο διπλανό σχήμα.

Παράδειγμα υλοποίησης στοίβας με πίνακα -2-

Element

θέση	αριθμός
1	**0**
2	**0**
3	
4	
. . .	
k	

- Αποθηκεύουμε το υπόλοιπο της δεύτερης διαίρεσης στην πρώτη θέση του πίνακα Element και μεταφέρουμε το στοιχείο Element[1] στη θέση 2 του πίνακα.

  Element[2] ← **Element[1]**

  Element[1] ← **0**

  και ο πίνακας Element έχει την διπλανή μορφή.

Παράδειγμα υλοποίησης στοίβας με πίνακα -3-

Element

θέση	**αριθμός**
1	**1**
2	**0**
3	**0**
4	
. . .	
k	

- Στο τέλος της τρίτης διαίρεσης χρειάζεται να αποθηκεύσουμε τον αριθμό 1, το υπόλοιπο, στην θέση 1 του πίνακα Element και να μετακινήσουμε τα στοιχεία, που υπάρχουν ήδη στον πίνακα, κατά μία θέση.

  Element[3] ← **Element[2]**

  Element[2] ← **Element[1]**

  Element[1] ← **1**

  και ο πίνακας Element είναι ο διπλανός

Παράδειγμα υλοποίησης στοίβας με πίνακα -4-

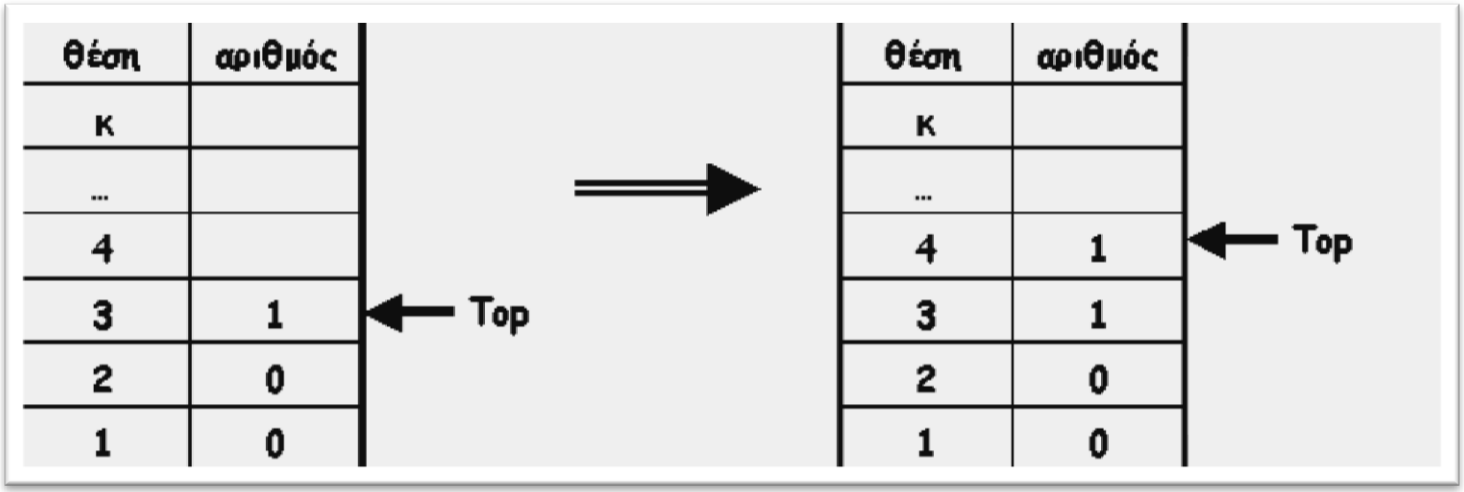
- Για να εισαχθεί ένα νέο στοιχείο στην πρώτη θέση του πίνακα Element, πρέπει να μετακινούμε κάθε φορά τα ήδη αποθηκευμένα στοιχεία κατά μία θέση προς τα κάτω. Το ίδιο ισχύει και για την διαγραφή.
- Οι μετακινήσεις είναι χρονοβόρες, γι' αυτό "γυρίζουμε ανάποδα" τον πίνακα, δηλαδή **καθορίζουμε την πρώτη θέση του ως τον πυθμένα της στοίβας** και η στοίβα να αυξάνει συνεχώς προς την θέση κ.

Παράδειγμα υλοποίησης στοίβας με πίνακα -5-

- Χρειαζόμαστε τότε μία **μεταβλητή Top, η οποία θα δείχνει κάθε φορά την κορυφή της στοίβας**.
- Επομένως, **οι εισαγωγές και διαγραφές στοιχείων θα γίνονται κάθε φορά στην θέση Stack[Top]** μέχρις ότου η μεταβλητή Top να γίνει ίση με κ.

Παράδειγμα υλοποίησης στοίβας με πίνακα -6-

- Στο παρακάτω σχήμα φαίνεται ο πίνακας Element μετά την εισαγωγή του τρίτου και τέταρτου αριθμού. Η Top "μετακινείται" (δείχνει) από την θέση 3 στην θέση 4 του πίνακα.



Αποθηκευτική δομή για μια στοίβα -1-

- Σε αυτήν την εφαρμογή, η αποθηκευτική δομή για μια στοίβα αποτελείται από:
  - έναν **πίνακα Element**, στον οποίο **αποθηκεύονται τα στοιχεία της στοίβας**, και
  - μια **μεταβλητή top**, στην οποία αποθηκεύεται **το στοιχείο της κορυφής της στοίβας**.

Αποθηκευτική δομή για μια στοίβα -2-

Μια τέτοια δομή μπορεί να υλοποιηθεί με μια εγγραφή (record), όπως φαίνεται παρακάτω:

```
#define StackLimit 50/*μέγιστο μέγεθος της στοίβας*/
typedef int StackElementType;  /*ο τύπος των
                                στοιχείων της στοίβας*/
typedef struct  {
        int Top;
       StackElementType Element[StackLimit];
} StackType;
```

Αποθηκευτική δομή για μια στοίβα -3-

- Για να ολοκληρωθεί η υλοποίηση της στοίβας, χρειάζονται διαδικασίες ή συναρτήσεις που να εκτελούν τις βασικές λειτουργίες της.
- Η δημιουργία μιας κενής στοίβας μπορεί να γίνει απλά με την εντολή

  Stack.Top = -1;
- *Χρησιμοποιούμε και τη μηδενική θέση του πίνακα, αν δεν τη χρησιμοποιούσαμε τότε θα θέταμε* **Stack.Top = 0;**
- Μια στοίβα θα είναι κενή όταν η boolean έκφραση **Stack.Top == -1** είναι αληθής.

Πακέτο για τον ΑΤΔ στοίβα -1-

- Επειδή οι στοίβες είναι χρήσιμες στην επίλυση ποικίλων προβλημάτων, θα ήταν ωφέλιμο να είχαμε έναν τύπο δεδομένων Στοίβα.
- Στην C μπορεί να κατασκευαστεί ένα **πακέτο** γι' αυτόν τον τύπο δεδομένων, που να περιλαμβάνει τις απαιτούμενες δηλώσεις καθώς και τις διαδικασίες και συναρτήσεις που υλοποιούν τις βασικές λειτουργίες και σχέσεις της στοίβας.

Πακέτο για τον ΑΤΔ στοίβα -2-

```
// FILENAME StackADT.h
#define StackLimit 50/*μέγιστο μέγεθος της στοίβας*/
typedef int StackElementType;  /*ο τύπος των
                                στοιχείων της στοίβας*/
typedef struct  {
        int Top;
            StackElementType
   Element[StackLimit];
} StackType;
```

Πακέτο για τον ΑΤΔ στοίβα -3-

```
// FILENAME StackADT.h
#define StackLimit 50   /*μέγιστο μέγεθος της στοίβας*/
typedef int StackElementType;      /*ο τύπος των
                                    στοιχείων της στοίβας*/
typedef struct  {
        int Top;
       StackElementType Element[StackLimit];
} StackType;
typedef enum {
                    FASLE, TRUE
   } boolean;
```

Πακέτο για τον ΑΤΔ στοίβα -4-

```
void CreateStack(StackType *Stack);
boolean EmptyStack(StackType Stack);
boolean FullStack(StackType Stack);
void Push(StackType *Stack,
StackElementType Item);
void Pop(StackType *Stack,
StackElementType *Item);
```

Δημιουργία στοίβας

```
void CreateStack(StackType *Stack)
/*Λειτουργία:     Δημιουργεί μια κενή στοίβα.
  Επιστρέφει:     Κενή Στοίβα.*/
{
    Stack -> Top = -1;
   // (*Stack).Top = -1;
}
```

Έλεγχος κενής στοίβας

```
boolean EmptyStack(StackType Stack)
/*Δέχεται:    Μια στοίβα Stack.
  Λειτουργία:Ελέγχει αν η στοίβα Stack είναι κενή.
  Επιστρέφει:TRUE αν η Stack είναι κενή, FALSE
    διαφορετικά.*/
{
    return (Stack.Top == -1);
}
```

Έλεγχος γεμάτης στοίβας

```
boolean FullStack(StackType Stack)
/*Δέχεται:  Μια στοίβα Stack.
  Λειτουργία:Ελέγχει αν η στοίβα Stack είναι
    γεμάτη.
  Επιστρέφει:TRUE αν η Stack είναι γεμάτη,
    FALSE διαφορετικά.*/
{
    return (Stack.Top == (StackLimit - 1));
}
```

Απώθηση στοιχειού -1-

void Pop(StackType *Stack,
StackElementType *Item)

/*Δέχεται: Μια στοίβα *Stack*.

Λειτουργία: Διαγράφει το στοιχείο *Item* από την κορυφή της Στοίβας αν η Στοίβα δεν είναι κενή.

Επιστρέφει: Το στοιχείο *Item* και την τροποποιημένη *Stack*.

Έξοδος: Μήνυμα κενής στοίβας αν η *Stack* είναι κενή.*/

Απώθηση στοιχείου -2-

```
{
    if (! EmptyStack(*Stack)) {
        *Item = Stack -> Element[Stack -> Top];
        Stack -> Top--;
    }
   else
        printf("Empty Stack...");
}
```

Ωθηση στοιχείου -1-

void Push(StackType *Stack, StackElementType Item)

/*Δέχεται: Μια στοίβα *Stack* και ένα στοιχείο *Item*.

Λειτουργία: Εισάγει το στοιχείο *Item* στην στοίβα *Stack* αν η *Stack* δεν είναι γεμάτη.

Επιστρέφει: Την τροποποιημένη *Stack*.

Έξοδος: Μήνυμα γεμάτης στοίβας, αν η στοίβα *Stack* είναι γεμάτη.*/

Ωθηση στοιχείου -2-

```
{
    if (! FullStack(*Stack)) {
        Stack -> Top++;
        Stack -> Element[Stack -> Top] = Item;
    }
  else
        printf("Full Stack...");
}
```

Σημείωση -1-

- **Η πιθανότητα να επαληθεύεται η συνθήκη της κενής στοίβας είναι "έμφυτη" στον ορισμό της στοίβας** και δεν προκύπτει, εξ αιτίας του τρόπου με τον οποίο υλοποιείται η στοίβα.
- Η συνθήκη της γεμάτης στοίβας **δεν είναι "έμφυτη" σ' αυτήν την δομή δεδομένων**, γιατί, θεωρητικά, δεν υπάρχει όριο στον αριθμό των στοιχείων που μπορεί να έχει μια στοίβα.

Σημείωση -2-

- Οποιαδήποτε υλοποίηση στοίβας που χρησιμοποιεί πίνακα για την αποθήκευση των στοιχείων της δεν θα είναι πιστή αναπαράσταση στοίβας.
- Αυτός είναι ο λόγος που η εισαγωγή στοιχείου (Push) περιλαμβάνει και τον έλεγχο γεμάτης στοίβας πριν γίνει η εισαγωγή του στοιχείου σ' αυτήν.

Αριθμητικές παραστάσεις

- Οι αριθμητικές εκφράσεις αποτελούνται από **τελεστές** (**operators**) και μεταβλητές ή **τελεστέους** (**operands**).
- Οι τελεστές διακρίνονται σε **δυαδικούς** (**binary**) και **μοναδικούς** (**unary**).
- Δυαδικοί ονομάζονται οι τελεστές που αφορούν δύο μεταβλητές ή τελεστέους, ενώ μοναδικοί ονομάζονται οι τελεστές που αφορούν έναν τελεστέο..

Ενδοθεματική παράσταση

- Συνήθως, όταν παριστάνουμε αριθμητικές εκφράσεις, ο μοναδικός τελεστής τίθεται πριν από τον τελεστέο και ο δυαδικός τελεστής μεταξύ των δύο τελεστέων, δηλαδή χρησιμοποιείται ο **ένθετος** ή **ενδοθεματικός** τρόπος παράστασης (**infix** notation). Αυτή η μορφή χρησιμοποιείται και στις περισσότερες γλώσσες προγραμματισμού.

Μεταθεματική παράσταση

- Κατά την μεταγλώττιση (compilation), όμως, του πηγαίου κώδικα σε εντολές γλώσσας μηχανής, οι αριθμητικές εκφράσεις μετατρέπονται πρώτα από την ενδοθεματική στην **επιθεματική** ή **μεταθεματική** μορφή (**postfix** notation) και μετά υπολογίζονται. Στην μεταθεματική μορφή, οι τελεστές έπονται των τελεστέων.

Ενδοθεματική & Μεταθεματική παράσταση -1-

Για παράδειγμα, η παρακάτω αριθμητική έκφραση που είναι γραμμένη σε ενδοθεματική μορφή

Α * Β - Γ + Δ

θα γραφεί

Α Β * Γ - Δ +

σε μεταθεματική μορφή.

Ενδοθεματική & Μεταθεματική παράσταση -2-

Για την ενδοθεματική μορφή, χρειάζεται συχνά να χρησιμοποιηθούν παρενθέσεις για να καθορίσουν τη σειρά με την οποία θα εκτελεστούν οι πράξεις.

Για παράδειγμα, η έκφραση: **Α * (Β - Γ)**

δείχνει ότι πρώτα πρέπει να γίνει η αφαίρεση και μετά ο πολλαπλασιασμός.

Στην περίπτωση που οι παρενθέσεις απουσιάζουν, προκύπτει η έκφραση: **Α * Β - Γ**

στην οποία εκτελείται πρώτα ο πολλαπλασιασμός και έπειτα η αφαίρεση, σύμφωνα με τους κανόνες προτεραιότητας στις πράξεις.

Αντίστροφη Πολωνική γραφή

Στις αρχές της δεκαετίας του 1950, ο Πολωνός μαθηματικός Jan Lukasiewicz παρατήρησε ότι οι παρενθέσεις δεν είναι απαραίτητες στην μεταθεματική γραφή, η οποία ονομάζεται επίσης **Αντίστροφη Πολωνική Γραφή** (Reverse Polish Notation, **RPN**).

Για παράδειγμα, η ενδοθεματική έκφραση

A * (B - Γ)

μπορεί να γραφτεί σε Αντίστροφη Πολωνική Γραφή ως εξής:

A B Γ - *

Αντίστροφη Πολωνική γραφή: Παράδειγμα -1-

Ας πάρουμε για παράδειγμα την έκφραση

4 3 - 9 2 5 + - *

που αντιστοιχεί στην ενδοθεματική έκφραση

(4 - 3) * (9 - (2 + 5))

Διατρέχουμε την RPN έκφραση από τα αριστερά προς τα δεξιά μέχρι να συναντήσουμε έναν τελεστή. Οι δύο τελευταίοι τελεστέοι πριν τον τελεστή συνδυάζονται με τον τελεστή αυτό.

Αντίστροφη Πολωνική γραφή: Παράδειγμα -2-

Στο παράδειγμά μας, ο πρώτος τελεστής που συναντάται είναι ο "-" και οι τελεστέοι που αντιστοιχούν σ' αυτόν είναι οι 4 και 3, όπως φαίνεται με την επισήμανση παρακάτω:

4 3 - 9 2 5 + - *

Το αποτέλεσμα της πράξης αυτής είναι 1, οπότε προκύπτει η ακόλουθη RPN έκφραση

1 9 2 5 + - *

Αντίστροφη Πολωνική γραφή: Παράδειγμα -3-

Συνεχίζοντας την αναζήτηση από αριστερά προς δεξιά, συναντάμε τον τελεστή "+" στον οποίο αντιστοιχούν οι τελεστέοι 2 και 5, όπως φαίνεται και παρακάτω:

1 9 **2 5 +** - *

Εκτελώντας την πράξη, παίρνουμε αποτέλεσμα 7 και η αρχική μας έκφραση γράφεται τώρα:

1 9 7 - *

Αντίστροφη Πολωνική γραφή: Παράδειγμα -4-

Στην συνέχεια, συναντάμε τον τελεστή "-" με τελεστέους τους 9 και 7, δηλαδή

1 **9 7 -** *

και μετά από την πράξη της αφαίρεσης, έχουμε

1 2 *

Ο τελευταίος τελεστής είναι ο "*" και σ' αυτόν αντιστοιχούν οι τελεστέοι 1 και 2. Το αποτέλεσμα της πράξης είναι 2 και αυτή είναι και η τιμή της έκφρασης που εξετάζουμε.

Αντίστροφη Πολωνική γραφή: υλοποίηση -1-

- Αυτή η μέθοδος υπολογισμού μιας RPN έκφρασης απαιτεί την αποθήκευση των τελεστέων μέχρι να συναντήσουμε έναν τελεστή στην από αριστερά προς δεξιά αναζήτηση.
- Μόλις συναντήσουμε τελεστή, χρειάζεται να ανακτήσουμε τους δυο τελευταίους τελεστέους και να τους συνδυάσουμε με τον τελεστή αυτόν.
- Κάτι τέτοιο υποθέτει την χρήση μιας **τελευταίος μέσα - πρώτος έξω δομής**, μιας **στοίβας**, για την αποθήκευση των τελεστέων.

Αντίστροφη Πολωνική γραφή: υλοποίηση -2-

- Κάθε φορά που συναντάμε έναν τελεστέο, αυτός εισάγεται μέσα στην στοίβα και, μόλις συναντήσουμε τελεστή, οι δυο κορυφαίες τιμές διαγράφονται από τη στοίβα, εφαρμόζεται η πράξη και το αποτέλεσμα εισάγεται στη στοίβα.

- Η διαδικασία αυτή φαίνεται στον παρακάτω αλγόριθμο.

Αλγόριθμος υπολογισμού RPN εκφράσεων -1-

/*Δέχεται: Μία RPN έκφραση.

Λειτουργία: Υπολογίζει την έκφραση.

Έξοδος: Η τιμή της RPN έκφρασης.

Σημείωση: Χρήση στοίβας για την αποθήκευση των τελεστέων.*/

1. Αρχικοποίησε μια κενή στοίβα

Αλγόριθμος υπολογισμού RPN εκφράσεων -2-

2. **Επανάλαβε**

Πάρε τον επόμενο χαρακτήρα από την RPN έκφραση

Αν ο χαρακτήρας είναι τελεστέος **τότε**

εισήγαγε τον χαρακτήρα στη στοίβα

Αλλιώς_αν ο χαρακτήρας είναι τελεστής **τότε**

Διάγραψε τις 2 κορυφαίες τιμές της στοίβας

Εφάρμοσε την πράξη που δηλώνει ο τελεστής, δηλαδή ο τελευταίος χαρακτήρας της PRN έκφρασης, σ' αυτές τις 2 τιμές

Εισήγαγε το αποτέλεσμα της πράξης στη στοίβα

Τέλος_αν

Μέχρις_ότου να φτάσεις στο τέλος της έκφρασης

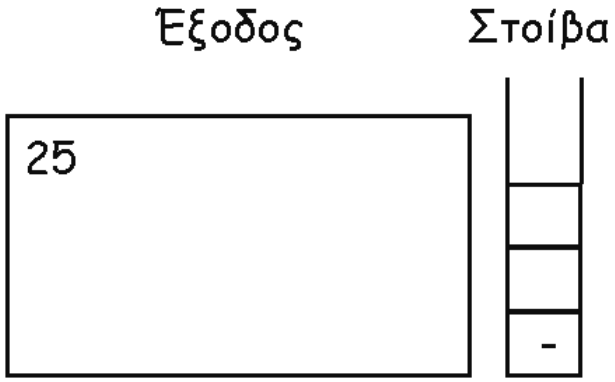
Αλγόριθμος υπολογισμού RPN εκφράσεων -3-

3. Όταν φτάσεις στο τέλος της PRN έκφρασης, η τιμή της βρίσκεται στην κορυφή της στοίβας

Μετατροπή έκφρασης από ενδοθεματική σε RPN -1-

Έστω, για παράδειγμα η έκφραση

25 - 10 * 2 + 4

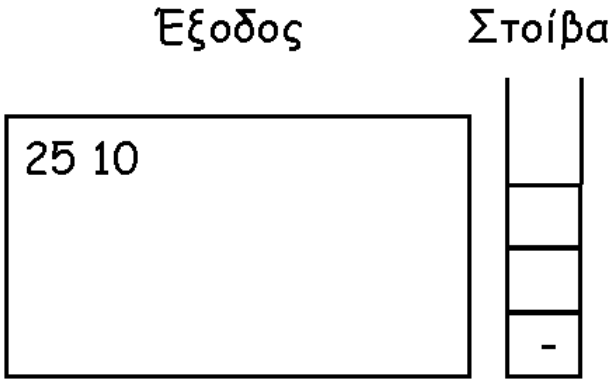


- Διατρέχοντας την έκφραση από αριστερά προς δεξιά, συναντάμε πρώτα τον αριθμό 25, ο οποίος μπορεί να εμφανιστεί απευθείας στην έξοδο.
- Εν συνεχεία συναντάμε τον τελεστή "-".
- Καθώς όμως δεν έχει εμφανιστεί ακόμα ο δεξιός τελεστέος του (ο αριστερός είναι προφανώς ο αριθμός 25), δεν μπορούμε ακόμα να τον εμφανίσουμε και γι' αυτό χρειάζεται να τον αποθηκεύσουμε, εισάγοντάς τον σε μια στοίβα τελεστών.

Μετατροπή έκφρασης από ενδοθεματική σε RPN -2-

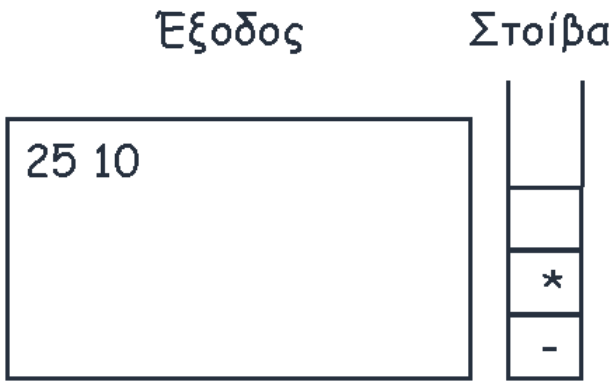
25 - 10 * 2 + 4

- Μετά από το "-" βρίσκεται ο αριθμός 10, ο οποίος μπορεί να εμφανιστεί αμέσως στην έξοδο.



Μετατροπή έκφρασης από ενδοθεματική σε RPN -3-

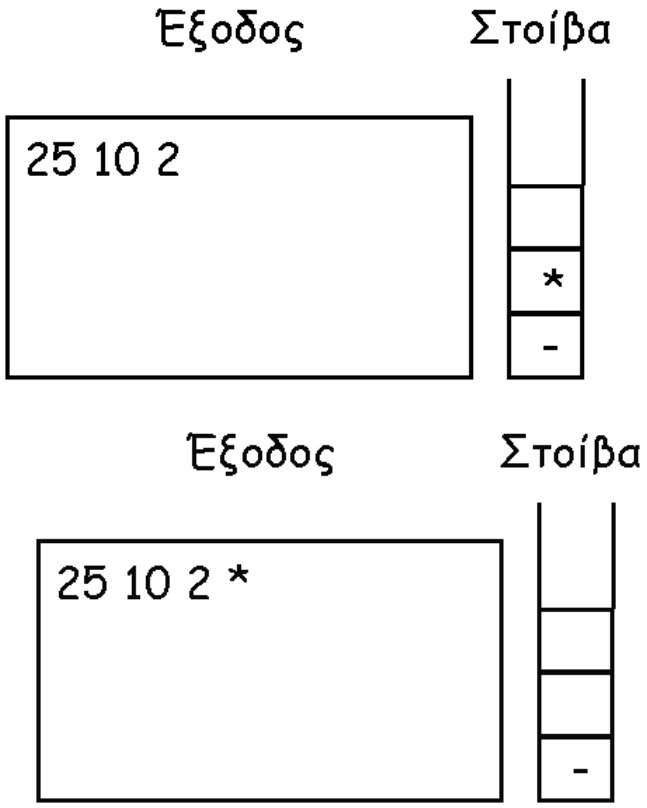
25 - 10 * 2 + 4



- Στο σημείο αυτό, πρέπει να καθοριστεί αν το 10 είναι ο δεξιός τελεστέος του "-", που αποθηκεύσαμε πριν λίγο, ή αν είναι ο αριστερός τελεστέος του επόμενου τελεστή.
- Συγκρίνοντας τον τελεστή "-", που βρίσκεται στην κορυφή της στοίβας, με τον τελεστή "*", που ακολουθεί, βλέπουμε ότι ο δεύτερος έχει μεγαλύτερη προτεραιότητα από τον πρώτο κι επομένως ο αριθμός 10 είναι ο αριστερός τελεστέος του "*".
- Άρα ο τελεστής "*" αποθηκεύεται στη στοίβα και συνεχίζουμε για να βρούμε τον δεξιό τελεστέο του.
-

Μετατροπή έκφρασης από ενδοθεματική σε RPN -4-

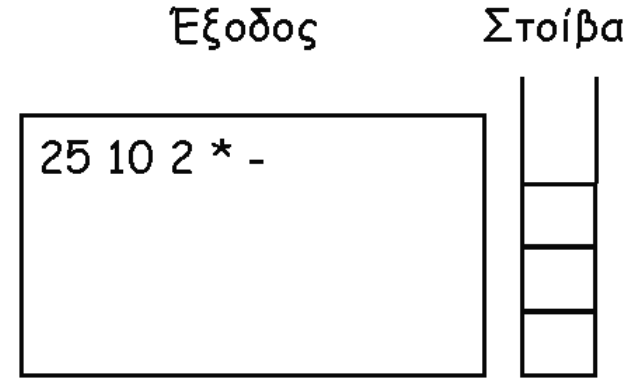
25 - 10 * 2 + 4



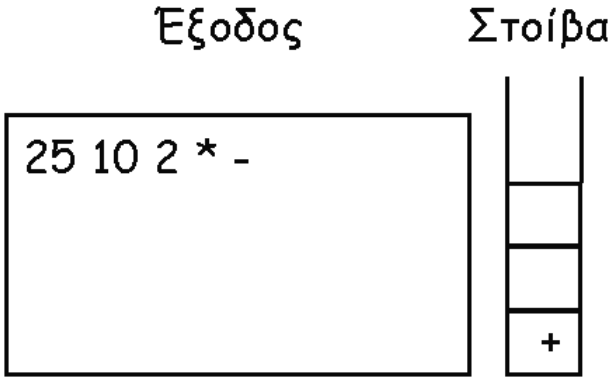
- Ακολουθεί ο αριθμός 2, τον οποίο εμφανίζουμε στην έξοδο.
- Τώρα πάλι πρέπει να ελέγξουμε αν ο αριθμός 2 είναι ο δεξιός τελεστέος του τελεστή "*" ή ο αριστερός τελεστέος του επόμενου τελεστή.
- Ο τελεστής που ακολουθεί είναι ο "+" και έχει μικρότερη προτεραιότητα από τον "*", επομένως συμπεραίνουμε ότι ο 2 είναι ο δεξιός τελεστέος του "*".
- Εχουν εμφανιστεί και ο αριστερός και ο δεξιός τελεστέος του "*", μπορεί να βγει από τη στοίβα και να εμφανιστεί στην έξοδο και ο ίδιος ο τελεστής.

Μετατροπή έκφρασης από ενδοθεματική σε RPN -5-

25 - 10 * 2 + 4



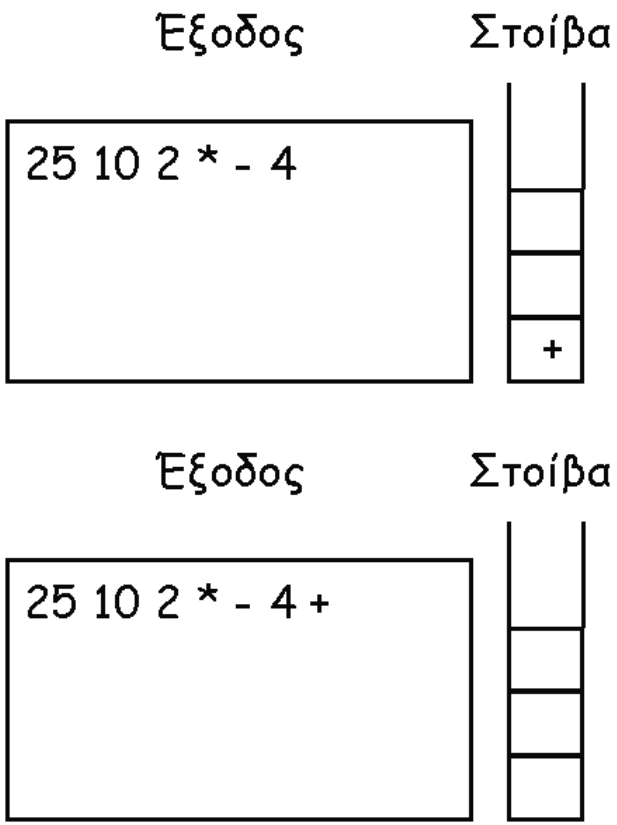
- Η παράσταση "10 2 *" αποτελεί τον δεξιό τελεστέο του "-" που αποθηκεύτηκε αρχικά, επομένως βγαίνει και αυτός από τη στοίβα και εμφανίζεται στην έξοδο



- Ο τελεστής "+" πρέπει να αποθηκευτεί στη στοίβα μέχρι να εμφανιστεί ο δεξιός τελεστέος του, όπως φαίνεται δίπλα:

Μετατροπή έκφρασης από ενδοθεματική σε RPN -6-

25 - 10 * 2 + 4



- Μετά από τον "+" ακολουθεί ο αριθμός 4, ο οποίος εμφανίζεται απευθείας στην έξοδο.
- Φτάσαμε πλέον στο τέλος της αριθμητικής έκφρασης, πράγμα που σημαίνει ότι ο αριθμός 4 είναι ο δεξιός τελεστέος του "+" και ότι ο τελεστής "+" βγαίνει από τη στοίβα και εμφανίζεται στην έξοδο.
- Έτσι, η στοίβα είναι τώρα κενή και στην έξοδο εμφανίζεται η αριθμητική έκφραση σε μορφή RPN

Αλγόριθμος μετατροπής ενδοθεματικής εκφ. σε RPN -1-

1. Αρχικοποίησε μια κενή στοίβα τελεστών
2. **Όσο** δεν έχει εμφανιστεί λάθος **και** δεν έχεις φτάσει στο τέλος της έκφρασης **επανάλαβε**

   α. Πάρε τον επόμενο χαρακτήρα στην ενδοθεματική έκφραση

   β. **Επίλεξε** χαρακτήρας

   Περίπτωση αριστερή παρένθεση

   Εισήγαγέ την στη στοίβα

Αλγόριθμος μετατροπής ενδοθεματικής εκφ. σε RPN -2-

Περίπτωση δεξιά παρένθεση

Διάγραψε και εμφάνισε τα στοιχεία της στοίβας μέχρι να διαγραφεί μια αριστερή παρένθεση, χωρίς όμως να την εμφανίσεις

Περίπτωση τελεστής

Αν η στοίβα είναι κενή **ή** ο χαρακτήρας έχει μεγαλύτερη προτεραιότητα από το κορυφαίο στοιχείο της στοίβας **τότε**

Εισήγαγε τον χαρακτήρα στη στοίβα

Αλγόριθμος μετατροπής ενδοθεματικής εκφ. σε RPN -3-

Αλλιώς

Διάγραψε και εμφάνισε το κορυφαίο στοιχείο της στοίβας

Τέλος_αν

Επανάλαβε την σύγκριση του χαρακτήρα με το νέο κορυφαίο στοιχείο

Περίπτωση τελεστέος

Εμφάνισέ τον

Τέλος_Επιλογών

Τέλος_επανάληψης

Αλγόριθμος μετατροπής ενδοθεματικής εκφ. σε RPN -4-

3. Όταν φτάσεις στο τέλος της αριθμητικής έκφρασης, διάγραψε και εμφάνισε τα στοιχεία της στοίβας μέχρι να αδειάσει η στοίβα

Τέλος Ενότητας

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης