ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΜΑΚΕΔΟΝΙΑΣ

ΣΤΟΙΒΕΣ
2.2 Υλοποίηση ΑΤΔ Στοίβα με Πίνακα

Υλοποίηση του ΑΤΔ στοίβα με πίνακα

Για να υλοποιήσουμε μια στοίβα, είναι απαραίτητο να επιλέξουμε έναν τύπο δεδομένων 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 0 να λειτουργεί ως η κορυφή της στοίβας.

Υλοποίηση του ΑΤΔ στοίβα με πίνακα: παράδειγμα

Παράδειγμα: μετατροπή του αριθμού 12 από το δεκαδικό σύστημα στο δυαδικό.

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

Αφού διαιρέσουμε το 12 με το 2, αποθηκεύουμε τον αριθμό 0, δηλαδή το υπόλοιπο της διαίρεσης, στην θέση 0 ενός πίνακα Element.

Έτσι, λοιπόν, στην θέση Element[0] βρίσκεται ο αριθμός 0, όπως φαίνεται στο διπλανό σχήμα.

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

Στην συνέχεια, πρέπει να αποθηκεύσουμε το υπόλοιπο της δεύτερης διαίρεσης στη θέση 0 του πίνακα Element και κατά συνέπεια να μεταφέρουμε το στοιχείο Element[0] στη θέση 1 του πίνακα.

Με άλλα λόγια πρέπει να κάνουμε:

Element[1] ← **Element[0]**

Element[0] ← **0**

και ο πίνακας Element έχει την διπλανή μορφή.

Υλοποίηση του ΑΤΔ στοίβα με πίνακα: παράδειγμα

	Element
θέση	αριθμός
0	1
1	0
2	0
3	
. . .	
k	

Στο τέλος της τρίτης διαίρεσης χρειάζεται να αποθηκεύσουμε τον αριθμό 1, δηλαδή το υπόλοιπο, στην θέση 0 του πίνακα Element και να μετακινήσουμε τα στοιχεία, που υπάρχουν ήδη στον πίνακα, κατά μία θέση.

Αυτή η πράξη ισοδυναμεί με τα εξής:

Element[2] ← **Element[1]**
Element[1] ← **Element[0]**
Element[0] ← **1**

και ο πίνακας Element είναι τώρα ο διπλανός

Υλοποίηση του ΑΤΔ στοίβα με πίνακα: παράδειγμα

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

Τέλος, η διαίρεση του αριθμού 1 με το 2 μας αφήνει υπόλοιπο 1 και ο αριθμός αυτός πρέπει πάλι να αποθηκευτεί στη θέση 0 του πίνακα Element, προκαλώντας μετακίνηση των υπόλοιπων στοιχείων κατά μία θέση, δηλαδή,

Element[3] ← **Element[2]**
Element[2] ← **Element[1]**
Element[1] ← **Element[0]**
Element[0] ← **1**

και ο πίνακας Element είναι τώρα ο διπλανός

Υλοποίηση του ΑΤΔ στοίβα με πίνακα: παράδειγμα

- Για να εισαχθεί ένα νέο στοιχείο στην πρώτη θέση (θέση 0) 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 ενός στοιχείου από τον πίνακα.

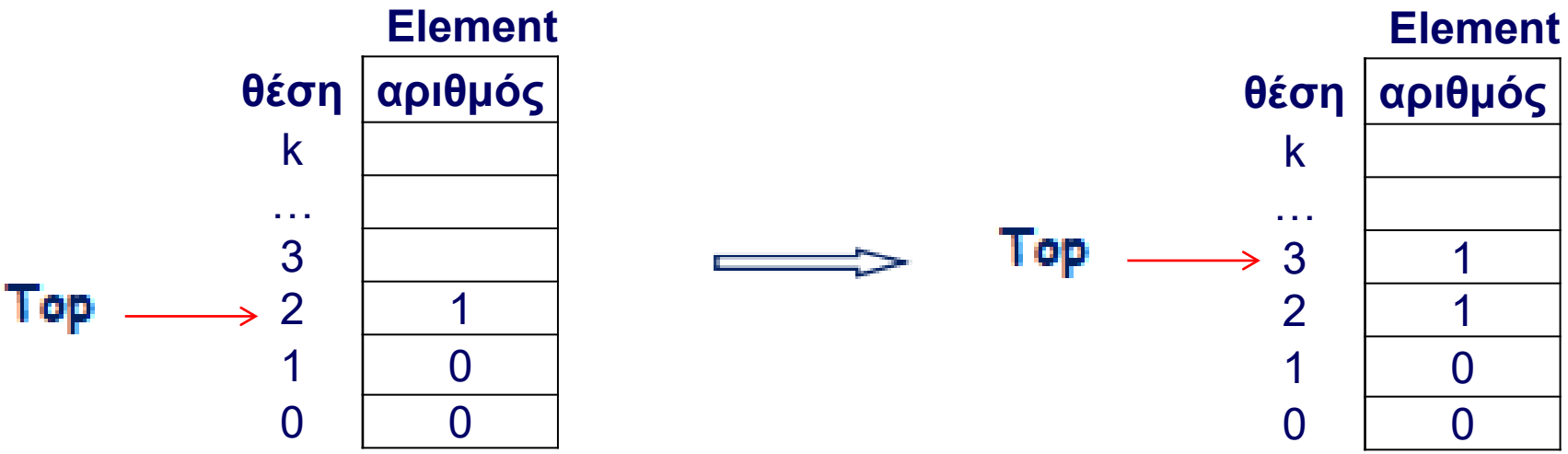
- Επειδή, όμως, αυτές οι μετακινήσεις είναι χρονοβόρες, μπορούμε να "γυρίσουμε ανάποδα" τον πίνακα, δηλαδή **να καθορίσουμε την πρώτη θέση του ως τον πυθμένα της στοίβας** και η στοίβα να αυξάνει συνεχώς προς την θέση κ.

- 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.

Υλοποίηση του ΑΤΔ στοίβα με πίνακα: παράδειγμα

- Στο παρακάτω σχήμα φαίνεται ο πίνακας Element μετά την εισαγωγή του τρίτου και τέταρτου αριθμού. Η Top "μετακινείται" (δείχνει) από την θέση 2 στην θέση 3 του πίνακα.



Αποθηκευτική δομή για μια στοίβα

Η αποθήκευση του ΑΤΔ στοίβα θα γίνει με:

- έναν **πίνακα Element**, στον οποίο **αποθηκεύονται τα στοιχεία της στοίβας**, και
- μια **μεταβλητή Top**, στην οποία αποθηκεύεται **η θέση του στοιχείου στην κορυφή της στοίβας**.

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

```
#define StackLimit 50                /*ενδεικτικό μέγιστο μέγεθος της στοίβας*/
typedef int StackElementType;        /*ενδεικτικός τύπος των στοιχείων της στοίβας*/
typedef struct  {
                    int Top;
                    StackElementType Element[StackLimit];
} StackType;
…..
StackType Stack;                     /*δήλωση μεταβλητής Stack για στοίβα*/
```

Stack

Top	πίνακας Element[ ] κάθε στοιχείο του τύπου StackElementType								
	0	1	2						StackLimit-1
2	12	8	23	?	?	?	?	…	?

Αποθηκευτική δομή για μια στοίβα

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.

Για τις λειτουργίες της διαγραφής και της εισαγωγής στοιχείου από ή στην στοίβα μπορούν να εφαρμοστούν οι αλγόριθμοι που παρουσιάζονται στη συνέχεια.

Αλγόριθμος διαγραφής στοιχείου από την κορυφή

POP

/*Δέχεται: Μια δομή για τη στοίβα Stack, η οποία περιλαμβάνει:

- μια μεταβλητή Top, η οποία δείχνει κάθε φορά την κορυφή της στοίβας
- έναν πίνακα Element, στον οποίο αποθηκεύονται τα στοιχεία της στοίβας

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.*/

1. Έλεγξε αν η στοίβα Stack είναι κενή
2. **Αν** η στοίβα δεν είναι κενή **τότε**

    α.Item ← Stack.Element[Stack.Top]
    β.Stack.Top ← Stack.Top - 1

   Αλλιώς

    Γράψε 'Η στοίβα είναι κενή'

   Τέλος_αν

Αλγόριθμος εισαγωγής στοιχείου στην κορυφή

PUSH

/*Δέχεται: Μια στοίβα Stack και ένα στοιχείο Item.
Λειτουργία: Εισάγει το στοιχείο Item στην στοίβα Stack εφόσον η Stack δεν είναι γεμάτη.
Επιστρέφει: Την τροποποιημένη στοίβα Stack.
Έξοδος: Μήνυμα γεμάτης στοίβας, αν η στοίβα Stack είναι γεμάτη.*/

1. Έλεγξε αν η στοίβα Stack είναι γεμάτη εξετάζοντας αν η μεταβλητή Stack.Top που δείχνει την κορυφή της στοίβας είναι ίση με το μέγεθος του πίνακα StackLimit

2. **Αν** η στοίβα δεν είναι γεμάτη **τότε**
   α. Stack.Top ← Stack.Top + 1
   β. Stack.Element[Stack.Top] ← Item
   Αλλιώς
   Γράψε 'Η στοίβα είναι γεμάτη'
   Τέλος_αν

Σημείωση

- **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.

- Ωστόσο, η συνθήκη γεμάτης στοίβας **δεν είναι "έμφυτη" σ' αυτήν την δομή δεδομένων**, γιατί, θεωρητικά, δεν υπάρχει όριο στο πλήθος των στοιχείων που μπορεί να έχει μια στοίβα.

- Οποιαδήποτε υλοποίηση στοίβας που χρησιμοποιεί πίνακα για την αποθήκευση των στοιχείων της δεν θα είναι πιστή αναπαράσταση στοίβας.

- Αυτός είναι ο λόγος που ο αλγόριθμος εισαγωγής στοιχείου (Push) περιλαμβάνει και τον έλεγχο γεμάτης στοίβας πριν γίνει η εισαγωγή του στοιχείου σ' αυτήν.

- Η υλοποίηση της στοίβας με **χρήση συνδεδεμένων λιστών**, η οποία θα παρουσιαστεί σε επόμενο κεφάλαιο δεν επιβάλλει ένα προκαθορισμένο όριο μεγέθους κι επομένως, **υλοποιεί την στοίβα πιο πιστά**.

Πακέτο για τον ΑΤΔ στοίβα

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.

Παρακάτω φαίνεται η υλοποίηση του τύπου δεδομένων Στοίβα, σε C, με το αρχείο κεφαλίδας StackADT.h και το αρχείο StackADT.c .

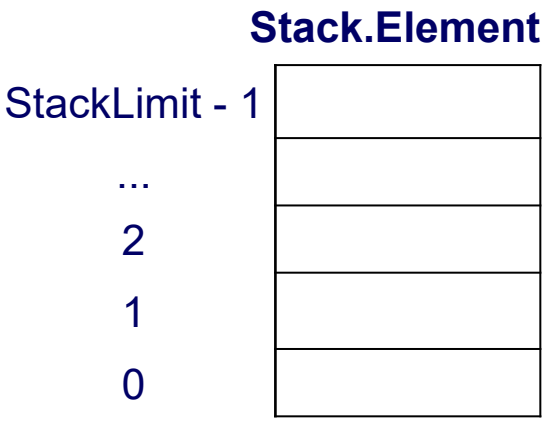
Πακέτο για τον ΑΤΔ στοίβα

```
// FILENAME StackADT.h

#define StackLimit 50                /*το όριο μεγέθους της στοίβας*/
typedef int StackElementType;        /*ενδεικτικός τύπος στοιχείων στοίβας*/
typedef struct {
                    int Top;
                    StackElementType Element[StackLimit];
} StackType;

typedef enum {
                    FALSE, TRUE
} boolean;

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

```
#define StackLimit 50                    /*το όριο μεγέθους της στοίβας*/
typedef int StackElementType;
typedef struct {
                        int Top;
                        StackElementType Element[StackLimit];
} StackType;

typedef enum {
                        FALSE, TRUE
} boolean;
```

void CreateStack(StackType *Stack);



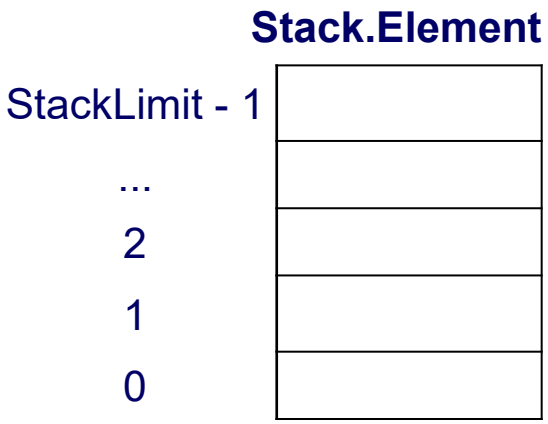
Stack.Top???

Δημιουργία στοίβας

```
#define StackLimit 50                    /*το όριο μεγέθους της στοίβας*/
typedef int StackElementType;
typedef struct {
                    int Top;
                    StackElementType Element[StackLimit];
} StackType;

typedef enum {
                    FALSE, TRUE
} boolean;
```

void CreateStack(StackType *Stack);

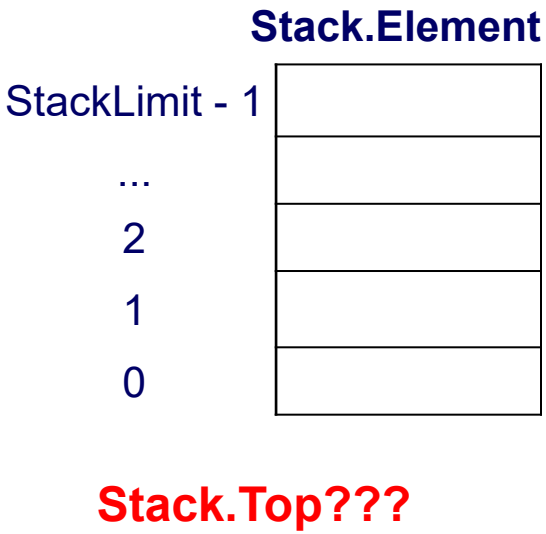
```
void CreateStack(StackType *Stack)
{
    Stack -> Top = -1;
}
```

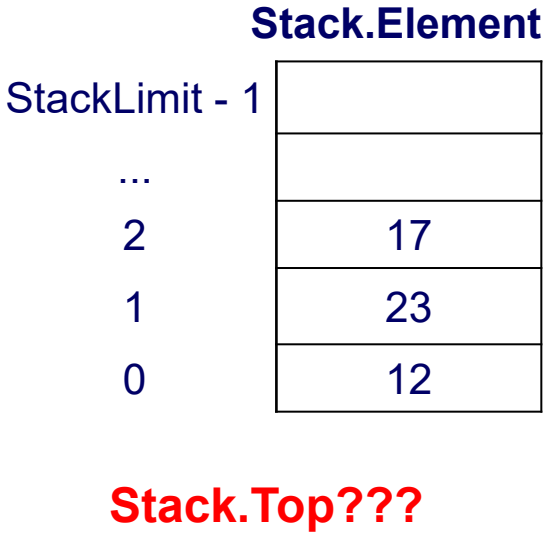


Κενή Στοίβα

```
#define StackLimit 50                    /*το όριο μεγέθους της στοίβας*/
typedef int StackElementType;
typedef struct {
                        int Top;
                        StackElementType Element[StackLimit];
} StackType;

typedef enum {
                        FALSE, TRUE
} boolean;
```

boolean EmptyStack(StackType Stack);

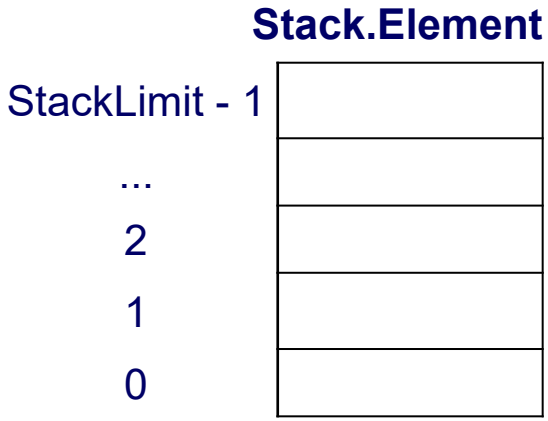


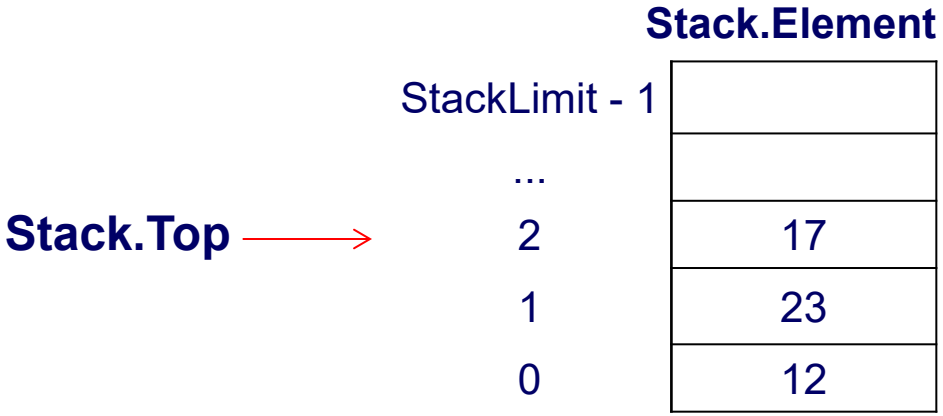


Κενή Στοίβα

```
#define StackLimit 50                    /*το όριο μεγέθους της στοίβας*/
typedef int StackElementType;
typedef struct {
                    int Top;
                    StackElementType Element[StackLimit];
} StackType;

typedef enum {
                    FALSE, TRUE
} boolean;

boolean  EmptyStack(StackType Stack);
```

```
boolean EmptyStack(StackType Stack)
{
   return (Stack.Top == -1);
}
```





Πακέτο για τον ΑΤΔ στοίβα

```
// FILENAME StackADT.c
#include <stdio.h>
#include "StackADT.h"

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

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);
}
```

Γεμάτη στοίβα

```
#define StackLimit 50                    /*το όριο μεγέθους της στοίβας*/
typedef int StackElementType;
typedef struct {
                    int Top;
                    StackElementType Element[StackLimit];
} StackType;

typedef enum {
                    FALSE, TRUE
} boolean;
```

boolean FullStack(StackType Stack);

Stack.Element

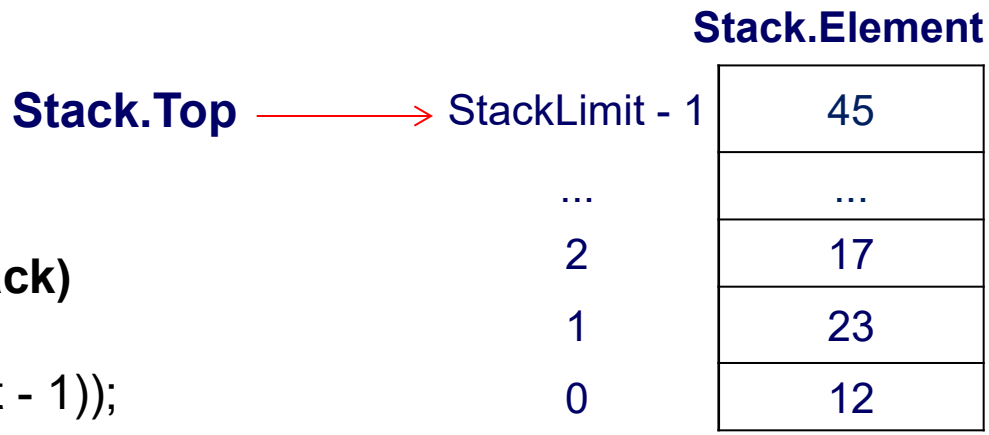
StackLimit - 1	45
...	...
2	17
1	23
0	12

Stack.Top???

Γεμάτη στοίβα

```
#define StackLimit 50                    /*το όριο μεγέθους της στοίβας*/
typedef int StackElementType;
typedef struct {
                    int Top;
                    StackElementType Element[StackLimit];
} StackType;

typedef enum {
                    FALSE, TRUE
} boolean;
```

boolean FullStack(StackType Stack);




```
boolean FullStack(StackType Stack)
{
    return (Stack.Top == (StackLimit - 1));
}
```

Πακέτο για τον ΑΤΔ στοίβα

```
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));
}
```

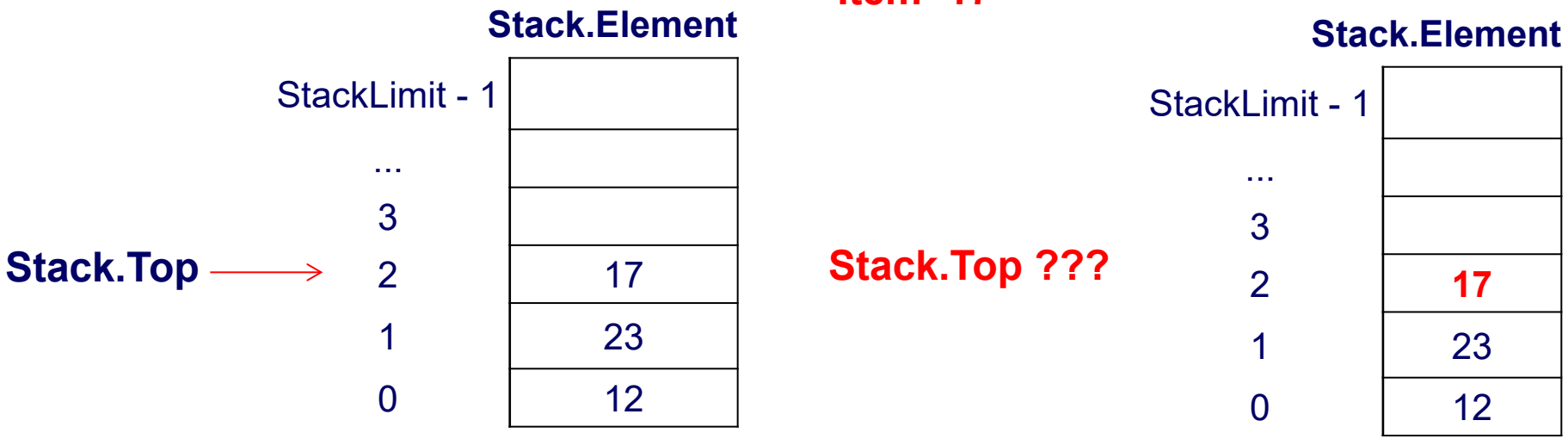
Pop

```
#define StackLimit 50                    /*το όριο μεγέθους της στοίβας*/
typedef int StackElementType;
typedef struct {
                    int Top;
                    StackElementType Element[StackLimit];
} StackType;

typedef enum {
                    FALSE, TRUE
} boolean;
```

void Pop(StackType *Stack, StackElementType *Item);

Pop(&Stack, &Item);

***Item=17**



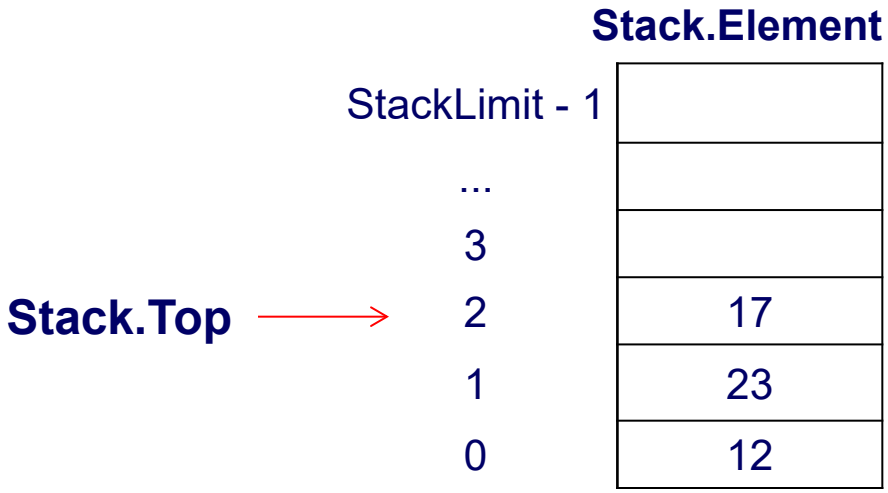
Pop

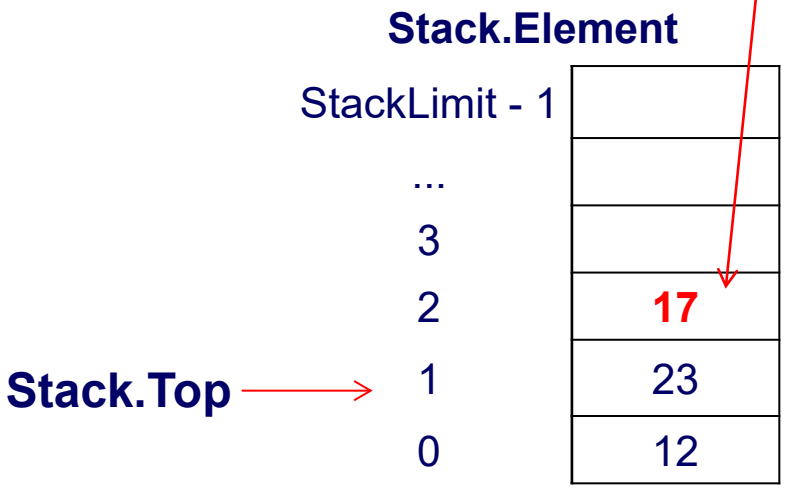
```
void Pop(StackType *Stack, StackElementType *Item)
{
    if (! EmptyStack(*Stack)) {
                        *Item = Stack -> Element[Stack -> Top];
                         Stack -> Top--;
    }
   else
        printf("Empty Stack...");
}
```

το στοιχείο στη θέση Top δε διαγράφεται «φυσικά» (ως στοιχείο του πίνακα) αλλά «λογικά» δηλ δεν μπορεί πλέον να προσπελαστεί αφού δεν είναι στοιχείο της στοίβας

Pop(&Stack, &Item);

***Item=17**





Πακέτο για τον ΑΤΔ στοίβα

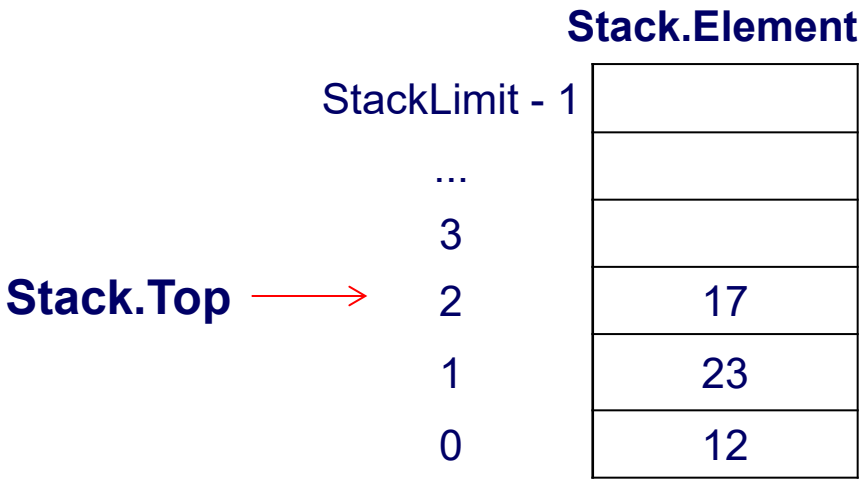
```
void Pop(StackType *Stack, StackElementType *Item)
/*Δέχεται:        Μια στοίβα Stack.
  Λειτουργία:     Διαγράφει το στοιχείο Item από την κορυφή της Στοίβας αν η
                  Στοίβα δεν είναι κενή.
  Επιστρέφει:     Το στοιχείο Item και την τροποποιημένη Stack.
  Έξοδος:         Μήνυμα κενής στοίβας αν η Stack είναι κενή.*/
{
    if (! EmptyStack(*Stack)) {
                    *Item = Stack -> Element[Stack -> Top];
                     Stack -> Top--;
    }
   else
        printf("Empty Stack...");
}
```

Push

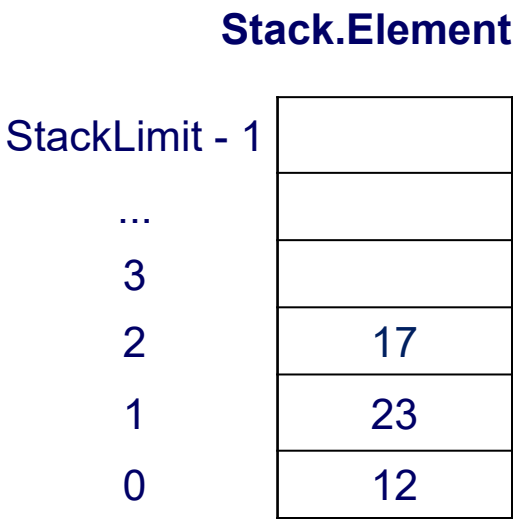
```
#define StackLimit 50                    /*το όριο μεγέθους της στοίβας*/
typedef int StackElementType;
typedef struct {
                    int Top;
                    StackElementType Element[StackLimit];
} StackType;

typedef enum {
                    FALSE, TRUE
} boolean;
```

void Push(StackType *Stack, StackElementType Item);

Push(&Stack, 32);



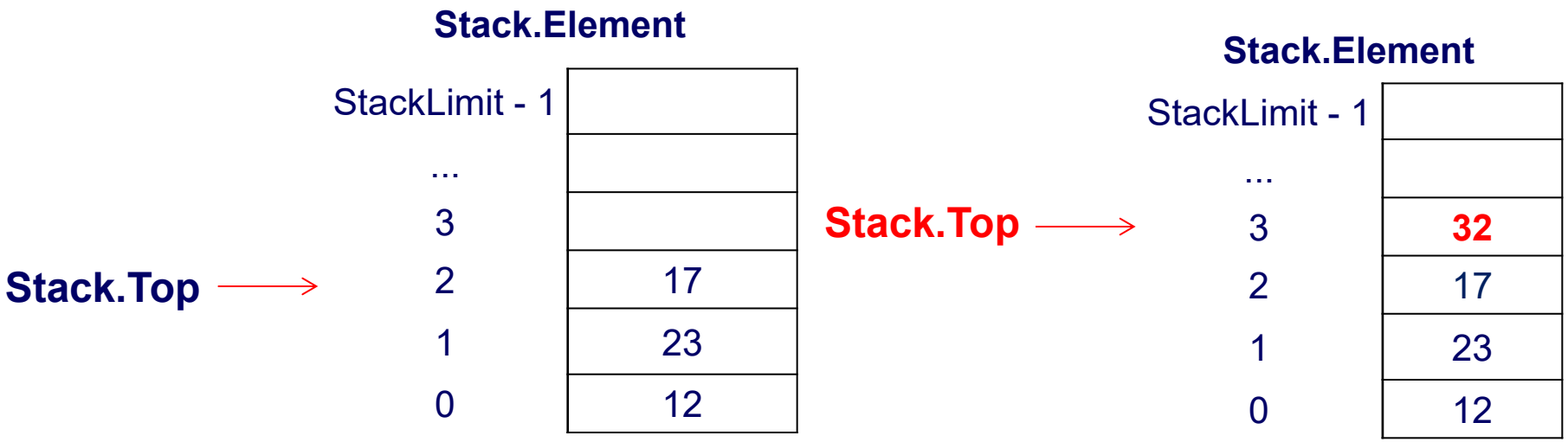
Stack.Top ???



Push

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

Push(&Stack, 32);



Πακέτο για τον ΑΤΔ στοίβα

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

Έλεγχος του StackADT

Παρακάτω φαίνεται ένα πρόγραμμα-πελάτης, TestStack.c, που χρησιμοποιεί τη διασύνδεση StackADT.h για τη στοίβα και εξετάζει αν η διασύνδεση StackADT.h και η υλοποιησή της StackADT.c λειτουργούν σωστά.

Το TestStack.c επιτρέπει στο χρήστη να εκτελέσει όλες τις βασικές λειτουργίες που σχετίζονται με τις στοίβες :

- να δημιουργήσει μια κενή στοίβα,
- να εισάγει στοιχεία σ' αυτήν,
- να διαγράφει στοιχεία από αυτήν και
- να ελέγχει αν η στοίβα είναι κενή ή γεμάτη.

Έλεγχος του StackADT

```
//filename TestStack.c

#include <stdio.h>
#include <stdlib.h>
#include "StackADT.h"

void TraverseStack(StackType Stack);

void menu(int *choice);

main()
{
    int i, choice;
    StackElementType AnItem;
    StackType AStack;

do
  {
      menu(&choice);
      switch(choice)
      {
```

Έλεγχος του StackADT

```
{
        case 1: CreateStack(&AStack);
                 break;
        case 2: if (EmptyStack(AStack))
                    printf("Empty Stack\n");
                 else
                    while (!EmptyStack(AStack))
                    {
                          Pop(&AStack, &AnItem);
                          printf("KORYFAIO STOIXEIO %d\n",AnItem);
                    }
                  break;
         case 3: if (EmptyStack(AStack))
                      printf("Empty Stack\n");
                  else printf("Not an Empty Stack\n");
                  break;
        case 4: if (!EmptyStack(AStack))
                    Pop(&AStack, &AnItem);
                    printf("\nKORYFAIO STOIXEIO %d\n",AnItem);
                 else printf("Empty Stack\n");
                 break;
```

Έλεγχος του StackADT

```
            case 5: printf("DWSE STOIXEIO GIA W8HSH: ");
                    scanf("%d",&AnItem);
                    Push(&AStack,AnItem);
                    break;
            case 6: if (EmptyStack(AStack))
                            printf("Empty Stack\n");
                    else TraverseStack(AStack);
                    break;
        }
    } while (choice!=7);
    return 0;
}
```

Έλεγχος του StackADT

```
void menu(int *choice)
{
        printf("\nXRHSIMOPOIHSE TIS PARAKATW ENTOLES GIA NA
        ELEGJEIS TO StackADT\n");
        printf("1 ---- DHMIOYRGIA STOIBAS\n");
        printf("2 ---- ADEIASMA THS STOIBAS\n");
        printf("3 ---- ELEGXOS KENHS STOIBAS\n");
        printf("4 ---  POP STOIXEIOY APO TH KORYFH THS STOIBA\n");
        printf("5 ---  PUSH STH KORYFH THS STOIBAS\n");
        printf("6 ---- EMFANISH STOIXEIWN STOIBAS (BOH8HTHKH)\n");
        printf("7 ---- EXIT\n");
        do
        {
                scanf("%d", choice);
        } while (*choice<1 && *choice>7);
}
```

Έλεγχος του StackADT

```
// διάσχιση stack βοηθητική συνάρτηση για έλεγχο των καταχωρήσεων
// 1ος τρόπος
void TraverseStack(StackType Stack)
{
    int i;
    printf("\nplithos sto stack %d\n",Stack.Top+1);
    for (i=0;i<=Stack.Top;i++)
        printf("%d, ",Stack.Element[i]);

    printf("\n");
}
// 2ος τρόπος διάσχιση από το κορυφαίο στοιχείο και προς τα «κάτω»
void TraverseStack(StackType Stack)
{
    int i;
    printf("\nplithos sto stack %d\n",Stack.Top+1);
    for (i=Stack.Top;i>=0;i--) {
        printf("%d, ",Stack.Element[i]);
    }
    printf("\n");
}
```