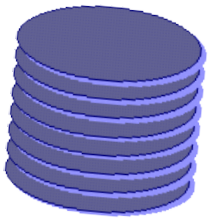


2.2 Υλοποίηση ΑΤΔ Στοίβα με πίνακα

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 σαν την κορυφή της στοίβας.

Element

θέση	αριθμός
0	0
1	
2	
3	
...	
κ	

Ας πάρουμε το παράδειγμα της μετατροπής του αριθμού 12 από το δεκαδικό σύστημα στο δυαδικό. Αφού διαιρέσουμε το 12 με το 2, αποθηκεύουμε τον αριθμό 0, δηλαδή το υπόλοιπο της διαίρεσης, στην θέση 0 ενός πίνακα, τον οποίο μπορούμε να ονομάσουμε Element. Έτσι, λοιπόν, στην θέση Element[0] βρίσκεται ο αριθμός

Element

θέση	αριθμός
0	0
1	0
2	
3	
...	
κ	

Στην συνέχεια, πρέπει να αποθηκεύσουμε το υπόλοιπο της δεύτερης διαίρεσης στην πρώτη θέση 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

Element

θέση	αριθμός
0	1
1	0
2	0
3	
...	
κ	

Στο τέλος της τρίτης διαίρεσης χρειάζεται να αποθηκεύσουμε τον αριθμό 1, δηλαδή το υπόλοιπο, στην θέση 1 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.

Element

θέση	αριθμός
0	1
1	1
2	0
3	0
...	
κ	

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

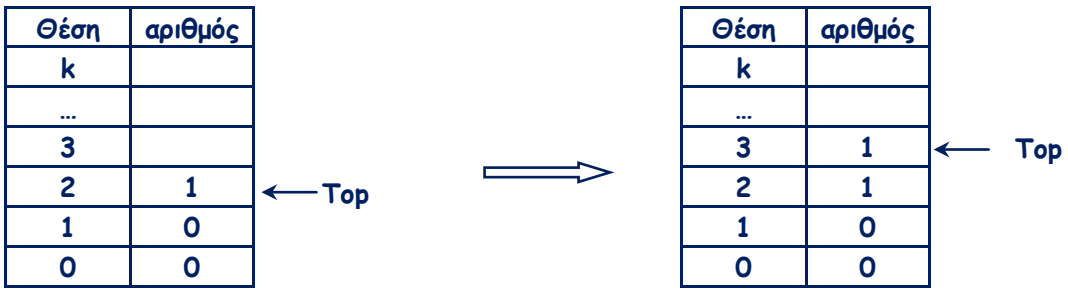
Element[3] ← Element[2]

Element[2] ← Element[1]

Element[1] ← Element[0]

Element[0] ← 1.

Έτσι, λοιπόν, 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 ενός στοιχείου από τον πίνακα: κάθε φορά που θέλουμε να διαγράψουμε το στοιχείο της κορυφής, προκαλούμε μετακίνηση των υπόλοιπων κατά μία θέση προς τα πάνω. 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. 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.



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

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

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=0;

και μια στοίβα θα είναι κενή όταν η boolean έκφραση Stack.Top=0 είναι αληθής.

Για τις λειτουργίες της διαγραφής και της εισαγωγής στοιχείου από ή στην στοίβα μπορούν να εφαρμοστούν οι ακόλουθοι αλγόριθμοι αντίστοιχα:

POP

/*Αλγόριθμος διαγραφής στοιχείου από την κορυφή της στοίβας*/

/*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]*

/*Θέσε στη μεταβλητή *Item* το στοιχείο που βρίσκεται στην κορυφή της στοίβας, δηλαδή το στοιχείο που βρίσκεται στη θέση *Stack.Top* του πίνακα *Stack.Element*/

β. *Stack.Top* ← *Stack.Top* - 1

/*Μείωσε την τιμή της *Stack.Top*, η οποία δείχνει την κορυφή της στοίβας, κατά 1*/

Αλλιώς

Γράψε 'Η στοίβα είναι κενή'

Τέλος_ αν

PUSH

/*Αλγόριθμος εισαγωγής του στοιχείου *Item* στην κορυφή της στοίβας*/

/*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

   /*Αύξησε τη μεταβλητή *Stack.Top* που δείχνει στην κορυφή της στοίβας κατά 1*/

   β.*Stack.Element[Stack.Top]* ← *Item*

   /*Θέσε στην κορυφή της στοίβας, δηλαδή στη θέση *Stack.Top* του πίνακα *Stack.Element*, την τιμή της *Item*/

   Αλλιώς

   Γράψε 'Η στοίβα είναι γεμάτη'

   Τέλος_ αν

ΣΗΜΕΙΩΣΗ: 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 στον αριθμό των στοιχείων που μπορεί να έχει μια στοίβα. Έτσι, λοιπόν, οποιαδήποτε υλοποίηση στοίβας που χρησιμοποιεί πίνακα για την αποθήκευση των στοιχείων της δεν θα είναι πιστή αναπαράσταση στοίβας, γιατί ένας πίνακας έχει σταθερό μέγεθος, πράγμα το οποίο συνεπάγεται την ύπαρξη ενός ανώτατου ορίου στο μέγεθος της στοίβας. Αυτός είναι, επομένως, ο λόγος που ο αλγόριθμος εισαγωγής στοιχείου (Push) περιλαμβάνει και τον έλεγχο *γεμάτης στοίβας* πριν γίνει η εισαγωγή του στοιχείου σ' αυτήν. Στο Κεφάλαιο 4 θεωρούμε μια εναλλακτική υλοποίηση της στοίβας με χρήση συνδεδεμένων λιστών, η οποία δεν επιβάλλει ένα προκαθορισμένο όριο μεγέθους κι επομένως, υλοποιεί την στοίβα πιο πιστά.

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                /*μέγιστο μέγεθος της στοίβας ενδεικτικά
                                       ίσο με 50 */
typedef intStackElementType;         /*ο τύπος των στοιχείων της στοίβας*/
typedef struct {
          int Top;
          StackElementType Element[StackLimit];
} StackType;

typedef enum {
           FASLE, 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);

// 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, FASLE διαφορετικά.*/
{
    return (Stack.Top == -1);
}
```

```
boolean FullStack(StackType Stack)
/*Δέχεται:              Μια στοίβα Stack.
  Λειτουργία:           Ελέγχει αν η στοίβα Stack είναι γεμάτη.
  Επιστρέφει:           TRUE αν η Stack είναι γεμάτη, FASLE διαφορετικά.*/
{
    return (Stack.Top == (StackLimit - 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.*/
{
   if (!EmptyStack(*Stack)) {
      *Item = Stack -> Element[Stack -> Top];
      Stack -> Top--;
   }
   else
      printf("Empty Stack...");
}

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...");
}
```

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 υλοποίηση της StackADT.c λειτουργούν σωστά. Το TestStack.c επιτρέπει στο χρήστη να εκτελέσει όλες τις βασικές λειτουργίες που σχετίζονται με τις στοίβες, δηλαδή 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.

```
// Filename: TestStack.c
#include <stdio.h>
#include <stdlib.h>
#include "StackADT.h"
void TraverseStack(StackType Stack);

void menu(int *choice);

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

    do
    {
         menu(&choice);
        switch(choice)
        {
             case 1: CreateStack(&AStack);
                     break;
            case 2: if (EmptyStack(AStack))
                        printf("Empty Stack\n");
                    else
                        while (!EmptyStack(AStack))
                        {
                            Pop(&AStack, &AnItem);
                             printf("ΣΤΟΙΧΕΙΟ ΚΟΡΥΦΗΣ ΛΙΣΤΑΣ %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("\nΣΤΟΙΧΕΙΟ ΚΟΡΥΦΗΣ ΛΙΣΤΑΣ %d\n",AnItem);
                      }
                      else printf("Empty Stack\n");
                     break;
             case 5: printf("ΔΩΣΕ ΤΟ ΣΤΟΙΧΕΙΟ ΓΙΑ ΩΘΗΣΗ ΣΤΗ ΣΤΟΙΒΑ: ");
                     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;
}
void menu(int *choice)
{
  printf("\nΧΡΗΣΙΜΟΠΟΙΗΣΕ ΤΙΣ ΠΑΡΑΚΑΤΩ ΕΝΤΟΛΕΣ ΓΙΑ ΝΑ ΕΛΕΓΞΕΙΣ ΤΟ
StackADT\n");
  printf("1 ---- ΔΗΜΙΟΥΡΓΙΑ ΣΤΟΙΒΑΣ\n");
  printf("2 ---- ΑΔΕΙΑΣΜΑ ΤΗΣ ΣΤΟΙΒΑΣ\n");
  printf("3 ---- ΕΛΕΓΧΟΣ ΚΕΝΗΣ ΣΤΟΙΒΑΣ\n");
  printf("4 ---  POP ΚΟΡΥΦΗΣ ΣΤΟΙΒΑΣ\n");
  printf("5 ---  PUSH ΣΤΗ ΚΟΡΥΦΗ ΣΤΟΙΒΑΣ\n");
  printf("6 ---- ΕΜΦΑΝΙΣΗ ΣΤΟΙΧΕΙΩΝ ΣΤΟΙΒΑΣ (ΒΟΗΘΗΤΙΚΗ)\n");
  printf("7 ---- ΕΞΟΔΟΣ\n");

do
    {
        scanf("%d", choice);
    } while (*choice<1 && *choice>7);
}
void TraverseStack(StackType Stack)
{
    int i;
    printf("\nΠΛΗΘΟΣ ΣΤΟΙΧΕΙΩΝ ΣΤΗ ΣΤΟΙΒΑ %d\n",Stack.Top+1);
    for (i=0;i<=Stack.Top;i++)
        printf("%d, ",Stack.Element[i]);

    printf("\n");
}
```

Ο αλγόριθμος μετατροπής ενός ακέραιου αριθμού από το δεκαδικό σύστημα στο δυαδικό υλοποιείται με το πρόγραμμα DecToBin.c το οποίο χρησιμοποιεί το StackADT.c &StackADT.h.