Διάλεξη 9: Στοίβες:Υλοποίηση & Εφαρμογές

Στην ενότητα αυτή θα μελετηθεί η χρήση στοιβών στις εξής εφαρμογές:

Υλοποίηση Στοιβών με Δυναμική Δέσμευση Μνήμης

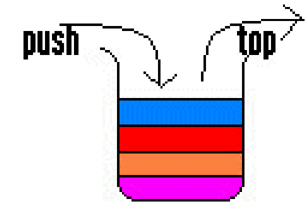
Εφαρμογή Στοιβών 1: Αναδρομικές συναρτήσεις

Εφαρμογή Στοιβών 2: Ισοζυγισμός Παρενθέσεων

Διδάσκων: Δημήτρης Ζεϊναλιπούρ

ΑΤΔ Στοίβες - Πράξεις

- Θυμηθείτε τον ΑΤΔ στοίβα με τις πράξεις του:



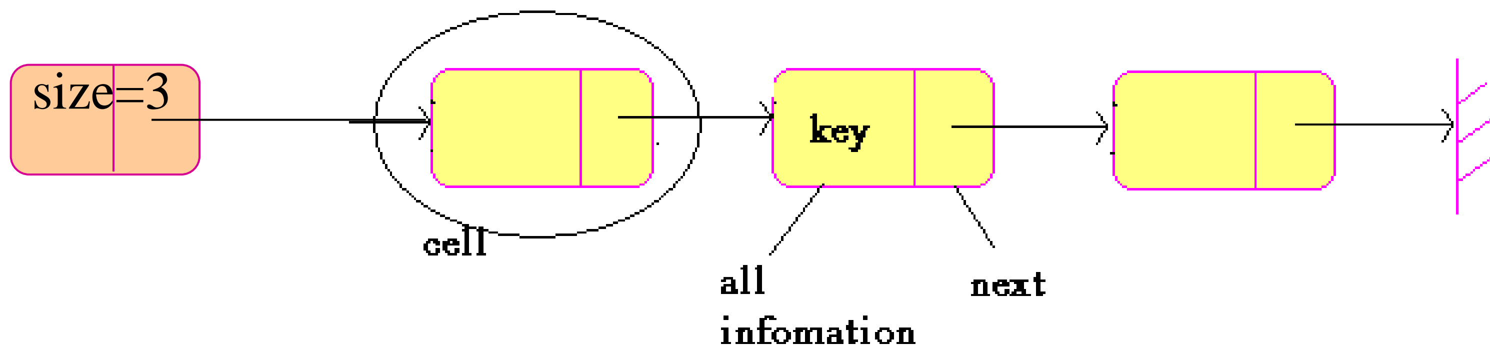
MakeEmptyStack()	δημιούργησε την κενή στοίβα ⟨⟩.
IsEmptyStack(S)	επέστρεψε τη λογική τιμή που εκφράζει το αν η S είναι κενή.
Push(x,S)	εισήγαγε τον κόμβο x στη στοίβα S.
Pop(S)	διέγραψε τον κόμβο κορυφής της S.
Top(S)	δώσε τον κόμβο κορυφής της S.

Είχαμε πει ότι αυτές οι πράξεις μπορούν
να υλοποιηθούν με την στατική δέσμευση μνήμης

```
typedef struct Stack{
        int list[size];
        int length;
} stack
```

Στοίβα με Δυναμική Δέσμευση Μνήμης

- Για την παράσταση μιας στοίβας με στοιχεία $\alpha_1, \alpha_2, ..., \alpha_n$ χρησιμοποιούμε μια συνδεδεμένη λίστα από κόμβους.
- Κάθε κόμβος αποτελείται από ένα στοιχείο (στοιχεία της στοίβας) και από ένα δείκτη (προς τον επόμενο κόμβο της στοίβας). Η κορυφή της στοίβας είναι ο πρώτος κόμβος της λίστας,
- Χρησιμοποιούμε μια μεταβλητή για να φυλάγουμε στοιχεία σχετικά με τη στοίβα π.χ. **μέγεθος** (size) και δείκτη προς την **κορυφή της στοίβας (head).**



Στοίβα με Δυναμική Δέσμευση Μνήμης

- Συνεπώς απαιτούνται οι παρακάτω δηλώσεις κόμβων:

```
typedef struct node {
      type          data;
      struct node *next;
 } NODE;
```

```
typedef struct stack {
      NODE   *head;
      int    size;
} STACK;
```

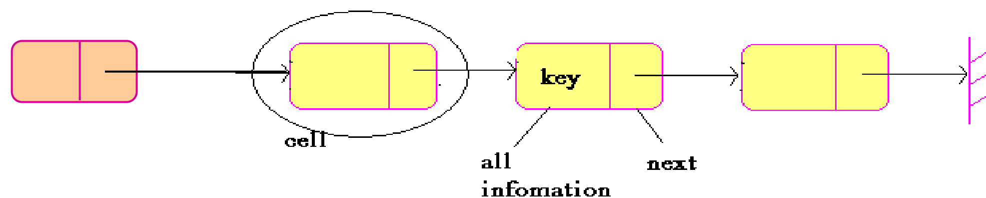
- Υπολείπεται η υλοποίηση των βασικών πράξεων στοίβας για αναπαράσταση με συνδεδεμένες λίστες.

- **Βασική προϋπόθεση:** όλες οι πράξεις να εκτελούνται χωρίς αναζήτηση οποιοδήποτε στοιχείου.

Στοίβα με Δυναμική Δέσμευση Μνήμης

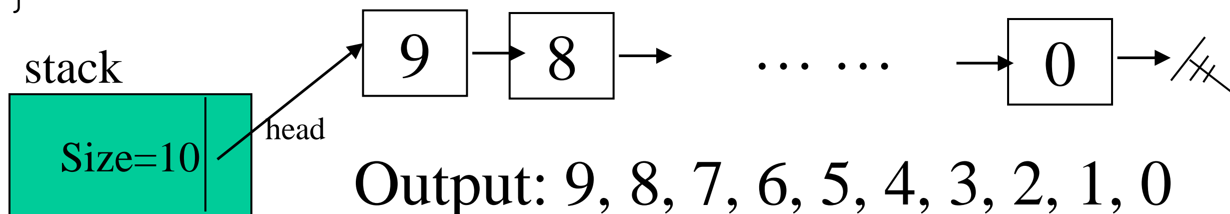
- Υλοποίηση πράξεων (ψευδοκώδικας):

```
int MakeEmptyStack(STACK *S)
    S->size = 0;
    S->head = NULL;

void Pop(STACK *S)
    if ((S->size) > 0)
            p = S->head;
            S->head = p->next;
            free(p);
            (S->size)--;

void Push(STACK *S, type x)
    p = (NODE *) malloc(sizeof(NODE));
     p->data = x;
     p->next = S->head;
     S->top = p;
     (S->size) ++;
```



Στοίβα με Δυναμική Δέσμευση Μνήμης

```
int main()  {
        STACK stack;
        MakeEmptyStack(&stack);
        // fill stack
        for(i=0; i<10; i++) {
                push(i, &stack);
        }
        // print stack
        for(i=0; i<10; i++) {
                top(&stack); pop(&stack);
        }
}
```



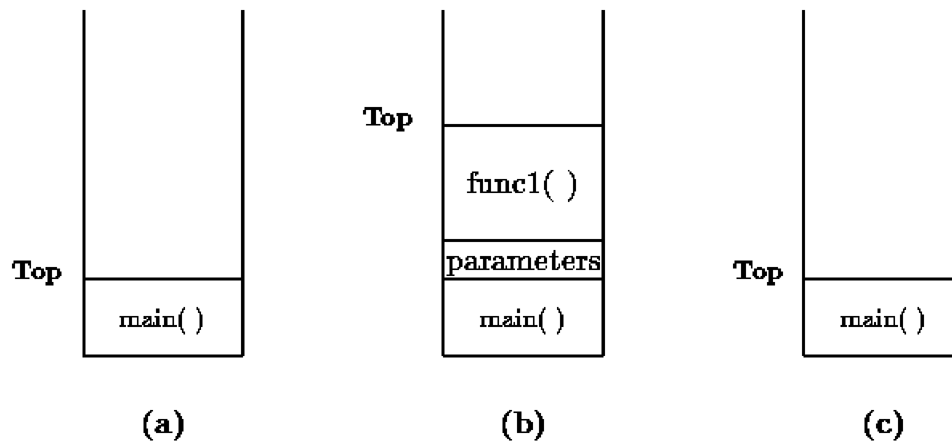
Εφαρμογή 1: Στοίβες και Αναδρομικές Διαδικασίες

- Οι στοίβες βρίσκουν μεγάλη χρήση στην πληροφορική για δημιουργία άλλων δομών και σε βασικό λογισμικό.
- Κλασικό παράδειγμα αφορά την **κλήση υποπρογραμμάτων** (function calls) και **αναδρομικών διαδικασιών**.
  - Σε κάθε κλήση οποιασδήποτε συνάρτησης ένα σύνολο από λέξεις (stack frame) φυλάσσεται σε μια στοίβα, από όπου μπορεί να ανασυρθεί.
  - Όταν μια συνάρτηση καλεί μια άλλη συνάρτηση οι **παράμετροι της συνάρτησης**, η **διεύθυνση επιστροφής** και οι **τοπικές μεταβλητές** της καλούσας συνάρτησης φυλάσσονται μέσα στη στοίβα του προγράμματος.
  - Έτσι, όταν η κληθείσα **συνάρτηση τερματίσει**, το περιβάλλον την καλούσας συνάρτησης **ανασύρεται** από τη στοίβα για να συνεχιστεί κανονικά η εκτέλεσή της.

Εφαρμογή 1: Στοίβες και Αναδρομικές Διαδικασίες

- Όταν ένα πρόγραμμα φορτώνετε στην μνήμη του υπολογιστή, τότε το η δεσμευμένη μνήμη οργανώνετε σε τρεις περιοχές (segments):

  a) text (code) segment, b) *stack segment*, and c) *heap segment*.
- Το *text segment* περιέχει τον κώδικα υπό εκτέλεση, ενώ το *heap segment* επιτρέπει στον πρόγραμμα να δεσμεύσει μνήμη (με την malloc!). Αυτή η μνήμη παραμένει μέχρι να αποδεσμευτεί ρητά (με την free!) ή να τερματιστεί το πρόγραμμα.
- Το *stack segment* από την άλλη χρησιμεύει για να φυλάγονται όλες οι πληροφορίες σχετικές με την εκτέλεση συναρτήσεων.



Η εντολή <u>ulimit -a</u> στο unix δείχνει τους διαθέσιμους πόρους που είναι διαθέσιμες σε ένα πρόγραμμα: virtual memory, stack, cpu time, processes, κτλ.

Εφαρμογή : Ισοζυγισμός Παρενθέσεων

- Ο έλεγχος σύνταξης (π.χ. ενός προγράμματος) απαιτεί να ταιριάξουμε σύμβολα/λέξεις όπως:

  `begin` με `end`

  `else` με `if`

  παρενθέσεις `{` με `}`

- Ας υποθέσουμε την ύπαρξη του συνόλου χαρακτήρων: { , } , [,] , (,).

- Πρόβλημα: να διαπιστώσετε αν μια συμβολοσειρά που περιέχει τους πιο πάνω χαρακτήρες είναι ισοζυγισμένη, δηλαδή όλες οι παρενθέσεις ταιριάζουν.

- π.χ. { [ ] }

  ([{ } { } [ ]))

  ([ ] { ( ) })

Παράδειγμα Εκτέλεσης

- Ανά πάσα στιγμή, η στοίβα περιέχει όλες τις `αριστερές΄ παρενθέσεις που δεν έχουν ακόμη `ταιριαστεί΄.

Ισοζυγισμός Παρενθέσεων

Λύση βασισμένη σε στοίβες

```
MakeEmpty (S);
while (c = nextcharacter() and (c is not EOF))
    if c != (,[,{, } , ], ) continue;

    if c = (,[,{
          Push(c,S);
    else // pop item from stack
       if IsEmpty(S) report error;
       else
          d = Top(S); Pop(S);
          if c does not match d
          report error
 }
 if IsEmpty(S) report success
 else report error
```