© A. Itai, November 25, 1997

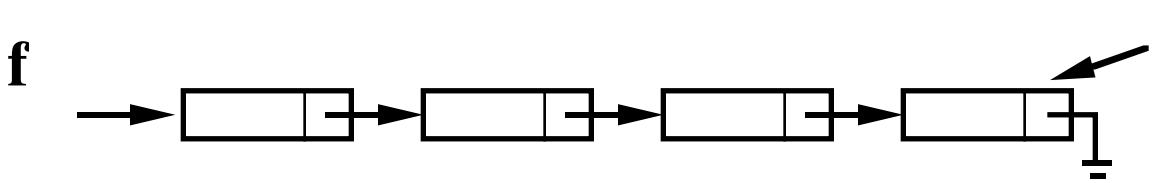
תור

create()	**פעולות: איתחול -יצירת תור ריק**
top(Q)	**ערך של ראש התור**
enqueue(x,Q)	**הכנסה (לסוף התור)**
dequeue(Q)	**הוצאה (לראש התור)**
is_empty(Q)	**בדיקת ריקנות**

FIFO = First In – First Out, ראשון נכנס–ראשון יוצא

מימושים של תור

1. רשימה מקושרת עם מצביע נוסף



איפה התחלת התור?

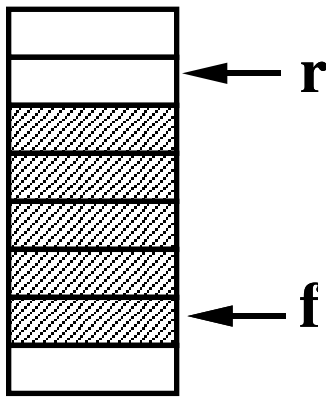
2. מערך a עם שני מחוונים

r = מקום אחרי האיבר בסוף התור

f = לאיבר בראש התור

r מקדים את f !

```
top(Q):          a[f];

enqueue(x,Q):    a[r]=x;
                 r++;

dequeue(Q):      f++;

is_empty(Q):     f == r;

create():        f = r = 0;
```



מאחר שהמערך מלא, ההתקדמות היא mod n

המימוש מלא אם f == (r+1) % n .

צריך לבדוק "לא מלא" לפני enqueue

ו"לא ריק" לפני dequeue.

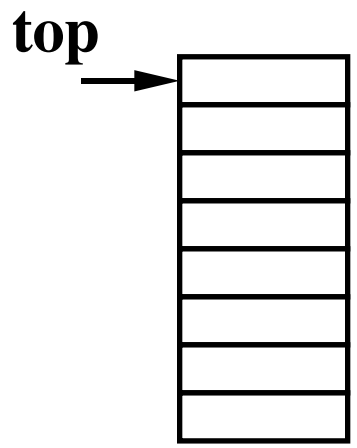
מחסנית כמבנה נתונים

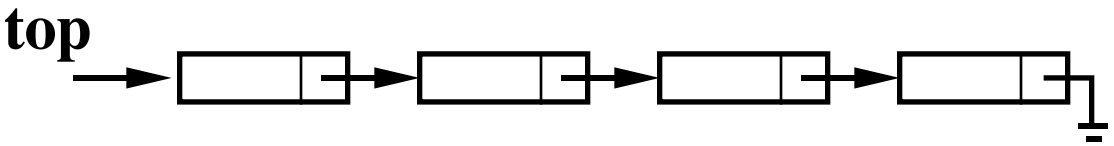
Last In – First Out :LIFO
אחרון נכנס - ראשון יוצא.

מחסנית מוגדרת ע"י הפעולות:

create( ) - מחזיר מחסנית ריקה חדשה.
push(s,x) - הכנסת איבר בעל ערך x למחסנית s.
top(s) - ערך האיבר בראש המחסנית s
(s לא משתנה).
pop(s) - הוצאת האיבר שבראש המחסנית s.
is_empty(s) - האם המחסנית s ריקה?

מימושים
מערך עם מצביע ל-top.
רשימה מקושרת חד-כיוונית עם מצביע להתחלה.
ישירות בחומרה.





חישוב ביטוי Postfix

infix הפעולה מופיעה בין האופרנדים

דוגמאות: 3 + 5 (3 + 5)*6 3 + 5*6

postfix הפעולה מופיעה אחרי האופרנדים

דוגמא: 3 5 + 3 5 + 6 * 3 5 6 * +

אלגוריתם לחישוב ביטוי postfix

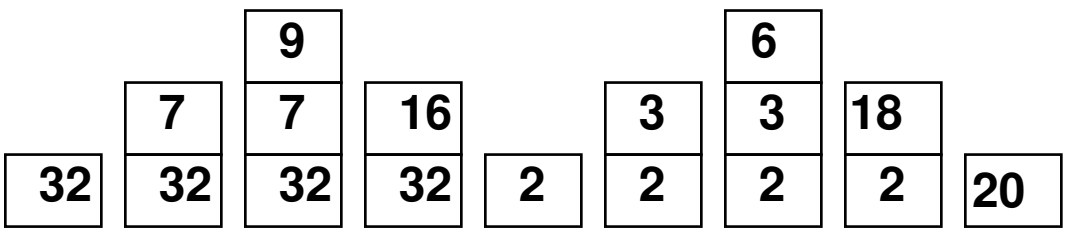
1. התחל עם מחסנית ריקה.
2. עבור על כל הביטוי משמאל לימין:
3. אם האיבר הבא הוא אופרנד -- הכנס אותו למחסנית.
4. אם הוא פעולה --הפעל את הפעולה על שני האיברים שבראש המחסנית והכנס את התוצאה למחסנית.

```
e1 = top();
pop();
e2 = top();
pop();
push(e2 op e1 );
```

INFIX: 32/(7+9) + 3*6

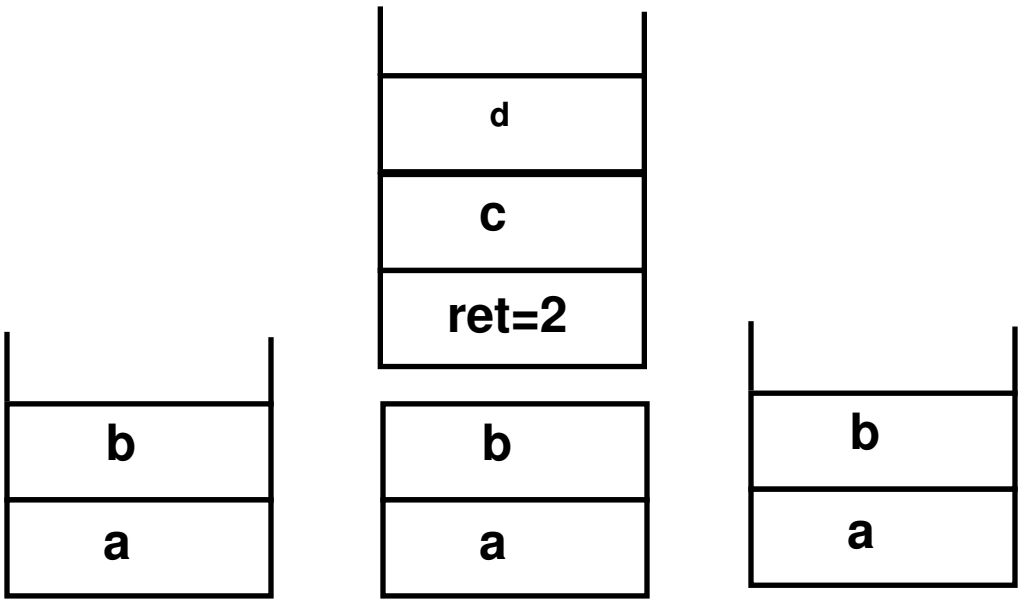
POSTFIX:

32 7 9 + / 3 6 * +

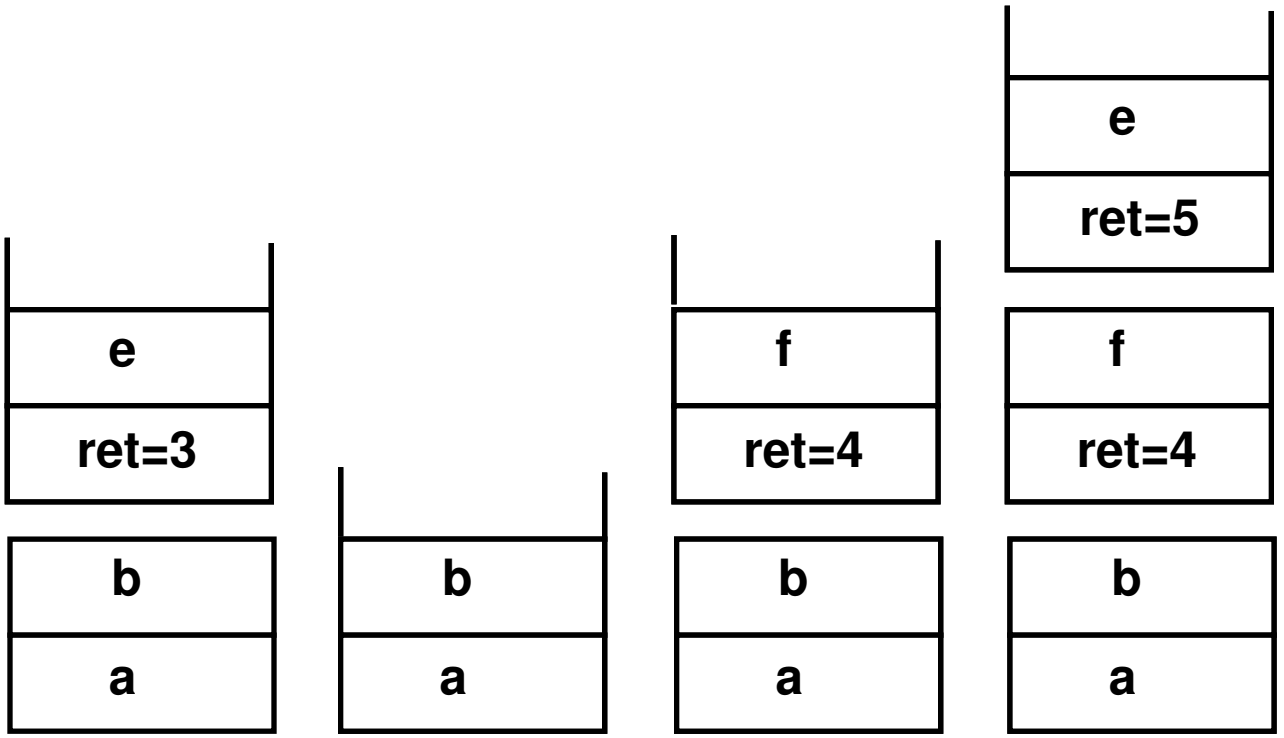


ניהול זכרון בעזרת מחסנית

```
void P()
{
    int c,d;
    .
    .
    .
}

void Q()
{
    int e;
    .
    .
    .
}

void R()
{
    int f;
    . . .
    Q();
5:  . . .
}

main()
{
    int a,b;
1:  P();
2:  Q();
3:  R();
4:  . . .
}
```



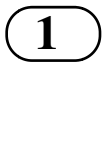
Enter main call P return from P

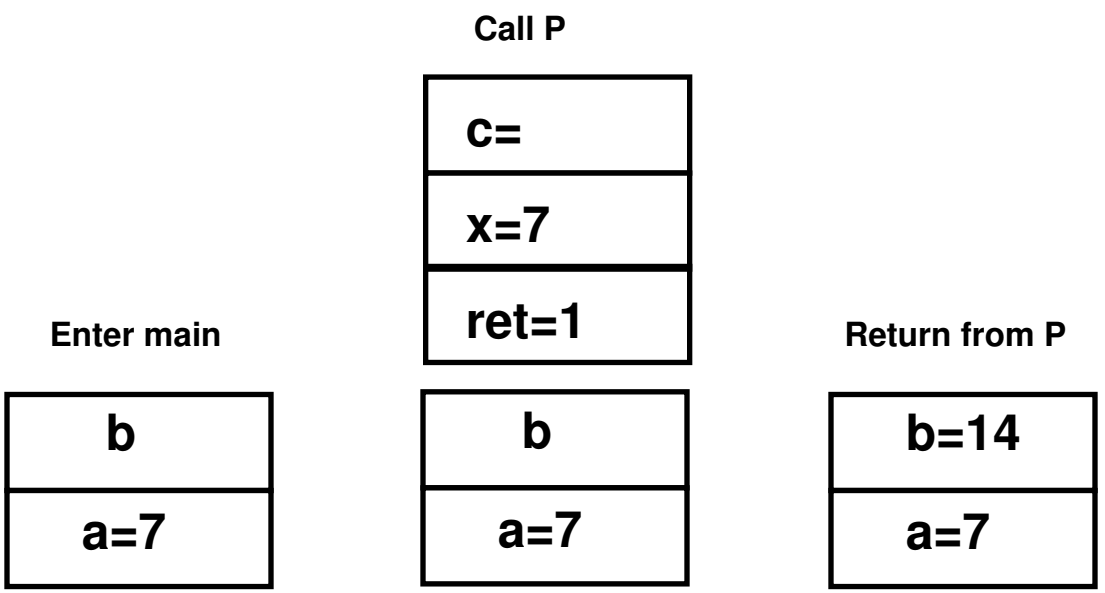


call Q return from Q call R call Q from R

פרוצדורות עם פרמטרים

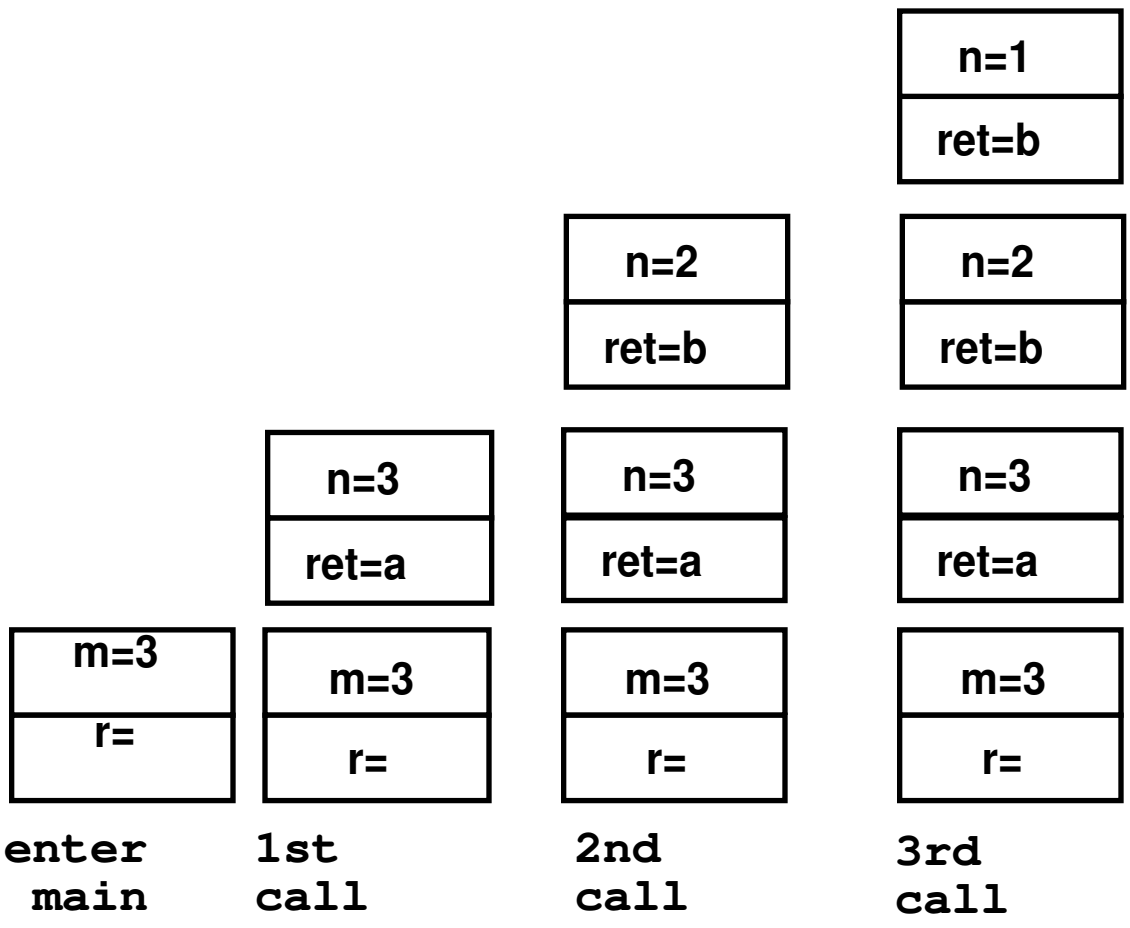
```
int P(int x)
{
    int c;
    c=x;
    return c+x;
}

main()
{
    int a,b;

    a=7;

    b = P(a);
    printf("My favorite number is %d\n",
                                        b);
}
```





רקורסיה

```
int factorial(int n)
{
    if(n <= 1)
        return 1;
    return n * factorial(n-1);
}

main()
{
    int m, r;
    scanf("%d", &m);
    r = factorial(m);

    printf("%d! = %d\n",m,r);
}
```

מימוש רקורסיה בשפה ללא רקורסיה

נשנה את התכנית הרקורסיבית:

נחליף את הקריאה הרקורסיבית:
נשמור את הפרמטרים והמשתנים המקומיים הישנים על המחסנית (push),
נשמור את כתובת הפקודה הבאה על המחסנית (push),
נחשב את הפרמטרים החדשים,
נלך לתחילת הפונקציה (goto).

נחליף את החזרה:
אם המחסנית ריקה ≡ return
אחרת: איחזור הפרמטרים והמשתנים המקומיים הישנים (pop),
נוציא את כתובת הפקודה הבאה מהמחסנית,
נשים את הערך המוחזר על המחסנית
נלך לכתובת החזרה (goto).

אחרי הקריאה: ניקח את ערך הפונקציה מהמחסנית.

דוגמא:

```
int rec_factorial(int n){
    int z;
1:  if(n≤1)
        return 1;
    z = factorial(n-1);
2:  z = n * z;
3:  return z;
}
```

ע"י הפעלת תהליך חילוץ הרקורסיה נקבל את התכנית:

```
int non_rec_factorial(int n){
    int z, ret_address;
1:  if(n ≤ 1){
        z = 1;
        goto 3;
    }
    push(n);
    push(2);
    n--;
    goto 1;
2:  /* after the call */
    z = pop(); /* z =value of recursive call */
    z = n*z;
    goto 3;
3:  /* function return */
    if(empty())
        return z;
    ret_address = pop();
    n = pop();
    push(z);

    /*goto ret_address;*/
    switch (ret_address){
    case 2: goto 2;
    }
}
```

הערה: התכנית הזאת התקבלה באופן מכני מהפעלת תהליך החילוץ ואינה דוגמא לסגנון תכנות טוב.