

Динамічні структури даних (мова Паскаль)

Стек

© К.Ю. Поляков, 2008-2010

Переклад: Р. М. Васильчик

Стек



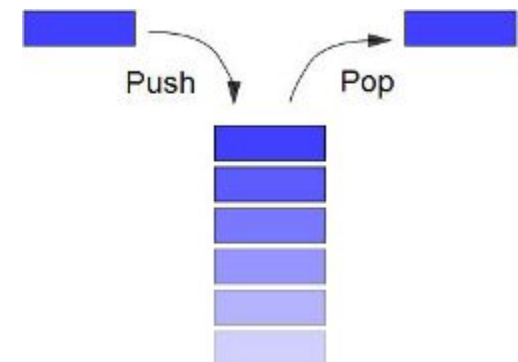
Стек – це лінійна структура даних, в якій додавання і видалення елементів можливо тільки з одного кінця (**вершини стека**). *Stack* = кипа, куча, стопка (англ.)

LIFO = Last In – First Out

«Хто останнім увійшов, той першим вийшов».

Операції зі стеком:

- 1) додати елемент на вершину (*Push* = заштовхнути);
- 2) зняти елемент з вершини (*Pop* = вилетіти зі звуком).



Приклад завдання

Завдання: вводиться символний рядок, в якому записано вираз з дужками трьох типів: $[]$, $\{ \}$ і $()$. Визначити, чи правильно розставлені дужки (не звертаючи уваги на інші символи). Приклади:

$[()] \{ \} \quad] [\quad [(\{)] \}$

Спрощене завдання: те ж саме, але з одним видом дужок.

Рішення: лічильник вкладеності дужок. Послідовність правильна, якщо в кінці лічильник дорівнює нулю і при проході ні разу не ставав негативним.

$(()) ()$
1 2 1 0 1 0

$(())) ($
1 2 1 0 -1 0

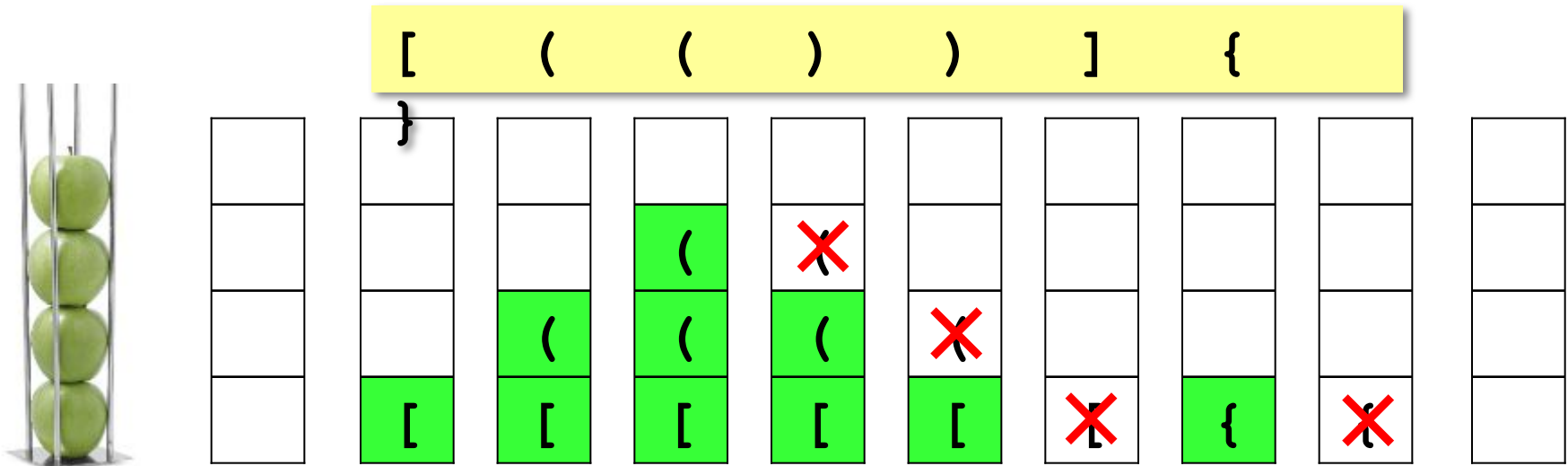
$(()) ($
1 2 1 0 1



Чи можна вирішити вихідне завдання так само, але з трьома лічильниками?

$[(\{)] \}$
(: 0 1 0
[: 0 1 0
{: 0 1 0

Рішення завдання з дужками



Алгоритм:

- 1) на початку стек порожній;
- 2) в циклі переглядаємо всі символи рядка по порядку;
- 3) якщо черговий символ – відкриваюча дужка, заносимо її на вершину стека;
- 4) якщо символ – закриваюча дужка, перевіряємо вершину стека: там повинна бути **відповідна** відкриваюча дужка (якщо це не так, то помилка);
- 5) якщо наприкінці стек не порожній, вираз неправильний.

Реалізація стека (масив)

Структура-стек:

```
const MAXSIZE = 100;  
type Stack = record { стек на 100 символів }  
  data: array[1..MAXSIZE] of char;  
  size: integer; { кількість елементів }  
end;
```

Додавання елемента:

```
procedure Push( var S: Stack; x: char );  
begin  
  if S.size = MAXSIZE then Exit;  
  S.size := S.size + 1;  
  S.data[S.size] := x;  
end;
```

ПОМИЛКА:
переповнення
стека

добавити елемент



Що погано?

Реалізація стека (масив)

Зняття елемента з вершини:

```
function Pop ( var S:Stack ) : char;  
begin  
  if S.size = 0 then begin  
    Result := char(255);  
    Exit;  
  end;  
  Result := S.data[S.size];  
  S.size := S.size - 1;  
end;
```

ПОМИЛКА:
стек порожній

Порожній чи ні?

```
function isEmpty ( S: Stack ) : Boolean;  
begin  
  Result := (S.size = 0);  
end;
```

Програма

```
var br1, br2, expr: string;
    i, k: integer;
    upper: char;      { то, що зняли зі стека }
    error: Boolean;   { ознака помилки }
    S: Stack;
begin
    br1 := '([{' ; br2 := ')]}' ;
    S.size := 0;
    error := False;
    writeln('Введіть вираз з дужками');
    readln(expr);
    ... { тут буде основний цикл обробки }
    if not error and isEmpty(S) then
        writeln('Вираз правильний.')
    else writeln('Вираз неправильний.')
end.
```

Відкриваючі дужки

закриваючі дужки

Обробка рядка (основний цикл)

```
for i:=1 to length(expr)
```

```
do begin
```

```
  for k:=1 to 3 do begin
```

```
    if expr[i] = br1[k] then begin { відкр. дужка }
```

```
      Push(S, expr[i]); { заштовхнути в стек }
```

```
      break;
```

```
    end;
```

```
    if expr[i] = br2[k] then begin { закр. дужка }
```

```
      upper := Pop(S); { зняти символ із стека }
```

```
      error := upper <> br1[k];
```

```
      break;
```

```
    end;
```

```
  end;
```

```
  if error then break;
```

```
end;
```

ЦИКЛ ПО ВСІХ СИМВОЛАХ
рядка expr

ЦИКЛ ЗА ВСІМА ВИДАМИ ДУЖОК

ПОМИЛКА: СТЕК
порожній або не
та дужка

була помилка: далі немає
сенсу перевіряти

Реалізація стека (список)

Структура вузла:

```
type PNode = ^Node; { вказівник на вузол }
   Node = record
       data: char; { дані }
       next: PNode; { вказівник на наст. елемент }
   end;
```

Додавання елемента:

```
procedure Push( var Head: PNode; x: char);
var NewNode: PNode;
begin
    New(NewNode); { виділити пам'ять }
    NewNode^.data := x; { записати символ }
    NewNode^.next := Head; { зробити першим вузлом }
    Head := NewNode;
end;
```

Реалізація стека (список)

Зняття елемента з вершини:

```

function Pop ( var Head: PNode ): char;
var q: PNode;
begin
  if Head = nil then begin { стек порожній }
    Result := char(255); {невикористовуваний символ}
    Exit;
  end;
  Result := Head^.data;    { взяти верхній символ }
  q := Head;              { запам'ятати вершину }
  Head := Head^.next;    { видалити вершину з стека }
  Dispose (q);           { видалити з пам'яті }
end;

```



Чи можна переставляти оператори?

Реалізація стека (список)

Порожній чи ні?

```
function isEmpty ( S: Stack ): Boolean;  
begin  
    Result := (S = nil);  
end;
```

Зміни в основній програмі:

```
var S: Stack;      var S: PNode;  
...  
S.size = 0;      S := nil;
```



Більше нічого не змінюється!

Обчислення арифметичних виразів

Як обчислювати автоматично:

$(a + b) / (c + d - 1)$

Інфіксийний запис

(знак операції між операндами)



необхідні дужки!

Префіксийний запис (знак операції до операндів)

$/$ $a + b$ $c + d - 1$

польська нотація,
[Jan Łukasiewicz](#) (1920)



дужки не потрібні, можна однозначно обчислити!

Постфіксийний запис (знак операції після операндів)

$a + b$ $c + d - 1$ $/$

зворотна польська нотація,
[F. L. Bauer](#) F. L. Bauer and [E. W. Dijkstra](#)

Запишіть в постфіксній формі

$$(32 * 6 - 5) * (2 * 3 + 4) / (3 + 7 * 2)$$

$$(2 * 4 + 3 * 5) * (2 * 3 + 18 / 3 * 2) * (12 - 3)$$

$$(4 - 2 * 3) * (3 - 12 / 3 / 4) * (24 - 3 * 12)$$

Обчислення виразів

Постфіксна форма:

$x = a b + c d + 1 - /$

					d		1		
		b		c	c	c+d	c+d	c+d-1	
	a	a	a+b	a+b	a+b	a+b	a+b	a+b	x

Алгоритм:

- 1) взяти черговий елемент;
- 2) якщо це не знак операції, добавить його в стек;
- 3) якщо це знак операції, то
 - взяти з стека два операнди;
 - виконати операцію і записати результат в стек;
- 4) перейти до кроку 1.

Системний стек (*Windows – 1 Мб*)

Використовується для

- 1) розміщення **локальних змінних**;
- 2) зберігання **адрес повернення** (за якими переходить програма після виконання функції або процедури);
- 3) передачі **параметрів** в функції та процедури;
- 4) тимчасового зберігання даних (в програмах на мові *Асемблер*).

Переповнення стека (*stack overflow*):

- 1) занадто багато локальних змінних
(**вихід** – використовувати динамічні масиви);
- 2) дуже багато рекурсивних викликів функцій і процедур
(**вихід** – переробити алгоритм так, щоб зменшити глибину рекурсії або відмовитися від неї взагалі).