Использование стека для вычисления выражений

Стеки и постфиксная нотация

Стек





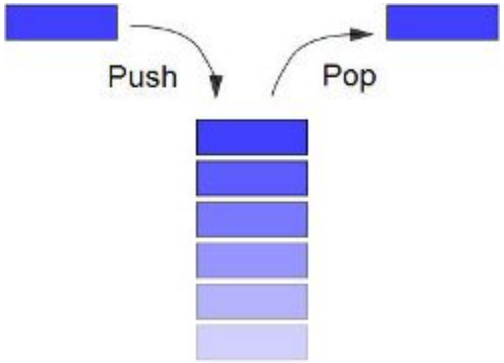
Стек – это линейная структура данных, в которой добавление и удаление элементов возможно только с одного конца (**вершины стека**). *Stack* = кипа, куча, стопка (англ.)

LIFO = *Last In – First Out*

«Кто последним вошел, тот первым вышел».

Операции со стеком:

1) добавить элемент на вершину (*Push* = втолкнуть);
2) снять элемент с вершины (*Pop* = вылететь со звуком).



Пример задачи

Задача: вводится символьная строка, в которой записано выражение со скобками трех типов: `[]`, `{}` и `()`. Определить, верно ли расставлены скобки (не обращая внимания на остальные символы). Примеры:

`[()]{}` `][` `[({)]}`

Упрощенная задача: то же самое, но с одним видом скобок.

Решение: счетчик вложенности скобок. Последовательность правильная, если в конце счетчик равен нулю и при проходе не разу не становился отрицательным.

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

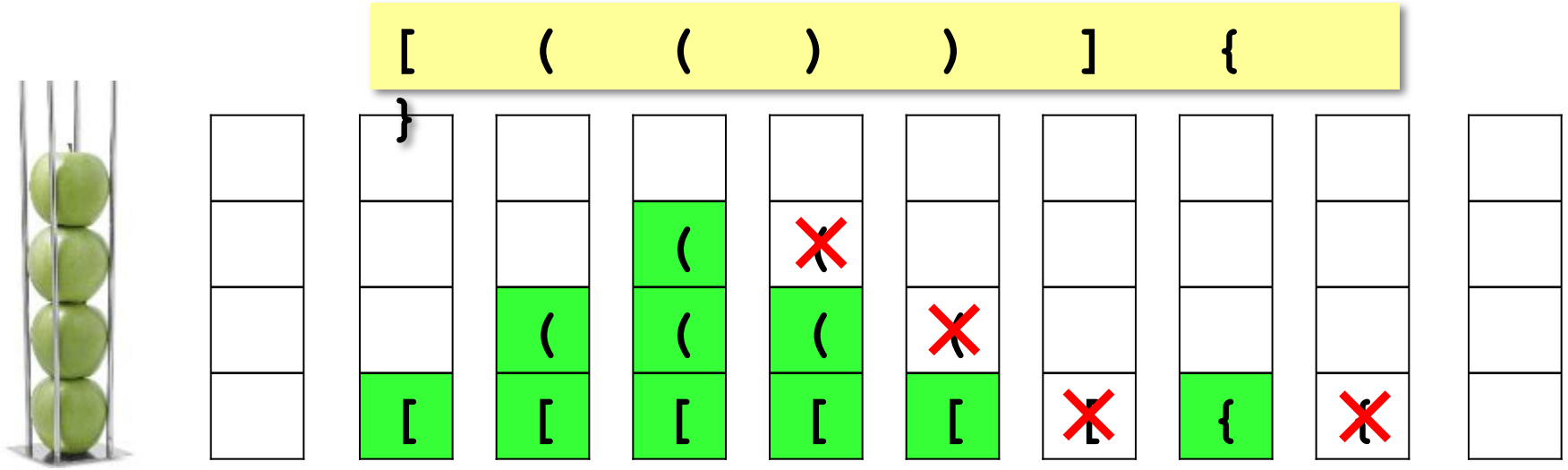
(	(	)	)	)	(
1	2	1	0	-1	0

(	(	)	)	(
1	2	1	0	1

Можно ли решить исходную задачу так же, но с тремя счетчиками?

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

Решение задачи со скобками



Алгоритм:

1) в начале стек пуст;
2) в цикле просматриваем все символы строки по порядку;
3) если очередной символ – открывающая скобка, заносим ее на вершину стека;
4) если символ – закрывающая скобка, проверяем вершину стека: там должна быть **соответствующая** открывающая скобка (если это не так, то ошибка);
5) если в конце стек не пуст, выражение неправильное.

Реализация стека (массив)

Структура-стек:

```
const MAXSIZE = 100;
struct Stack {
   char data[MAXSIZE]; // стек на 100 символов
   int  size;          // число элементов
   };
```

Добавление элемента:

```
int Push ( Stack &S, char x )
{
if ( S.size == MAXSIZE ) return 0;
S.data[S.size] = x;
S.size ++;
return 1;
}
```

ошибка: переполнение стека

добавить элемент

нет ошибки

Реализация стека (массив)

Снятие элемента с вершины:

```
char Pop ( Stack &S )
{
if ( S.size == 0 ) return char(255);
S.size --;
return S.data[S.size];
}
```

ошибка: стек пуст

Пустой или нет?

```
int isEmpty ( Stack &S )
{
if ( S.size == 0 )
     return 1;
else return 0;
}
```

```
int isEmpty ( Stack &S )
{
return (S.size == 0);
}
```

Программа

```
void main()
{
  char br1[3] = { '(', '[', '{' };
  char br2[3] = { ')', ']', '}' };
  char s[80], upper;
  int i, k, error = 0;
  Stack S;
  S.size = 0;
  printf("Введите выражение со скобками > ");
  gets ( s );
  ...  // здесь будет основной цикл обработки
  if ( ! error && (S.size == 0) )
       printf("\nВыражение правильное\n");
  else printf("\nВыражение неправильное\n");
}
```

открывающие скобки

закрывающие скобки

то, что сняли со стека

признак ошибки

Обработка строки (основной цикл)

```
for ( i = 0; i < strlen(s); i++ )
  {
  for ( k = 0; k < 3; k++ )
    {
    if ( s[i] == br1[k] )  // если открывающая скобка
      {
      Push ( S, s[i] ); // втолкнуть в стек
      break;
      }
    if ( s[i] == br2[k] ) // если закрывающая скобка
      {
      upper = Pop ( S );  // снять верхний элемент
      if ( upper != br1[k] ) error = 1;
      break;
      }
    }
  if ( error ) break;
  }
```

цикл по всем символам строки s

цикл по всем видам скобок

ошибка: стек пуст или не та скобка

была ошибка: дальше нет смысла проверять

Реализация стека (список)

Структура узла:

```
struct Node {
    char data;
       Node *next;
       };
typedef Node *PNode;
```

Добавление элемента:

```
void Push (PNode &Head, char x)
{
    PNode NewNode = new Node;
    NewNode->data = x;
    NewNode->next = Head;
    Head = NewNode;
}
```

Реализация стека (список)

Снятие элемента с вершины:

```
char Pop (PNode &Head) {
  char x;
  PNode q = Head;
  if ( Head == NULL ) return char(255);
  x = Head->data;
  Head = Head->next;
  delete q;
  return x;
}
```

стек пуст

Изменения в основной программе:

~~Stack~~ S;
~~S.size~~ = 0;

PNode S = NULL;

...

if (! error && (~~S.size~~ == 0))
    printf("\nВыражение правильное\n");
else printf("\nВыражение неправильное \n");

(S == NULL)

Вычисление арифметических выражений

Как вычислять автоматически:

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

Инфиксная запись
(знак операции **между** операндами)

необходимы скобки!

Префиксная запись (знак операции **до** операндов)

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

польская нотация,
Jan Łukasiewicz (1920)

скобки не нужны, можно однозначно вычислить!

Постфиксная запись (знак операции **после** операндов)

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

обратная польская нотация,
F. L. BauerF. L. Bauer and E. W. Dijkstra

Запишите в постфиксной форме

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

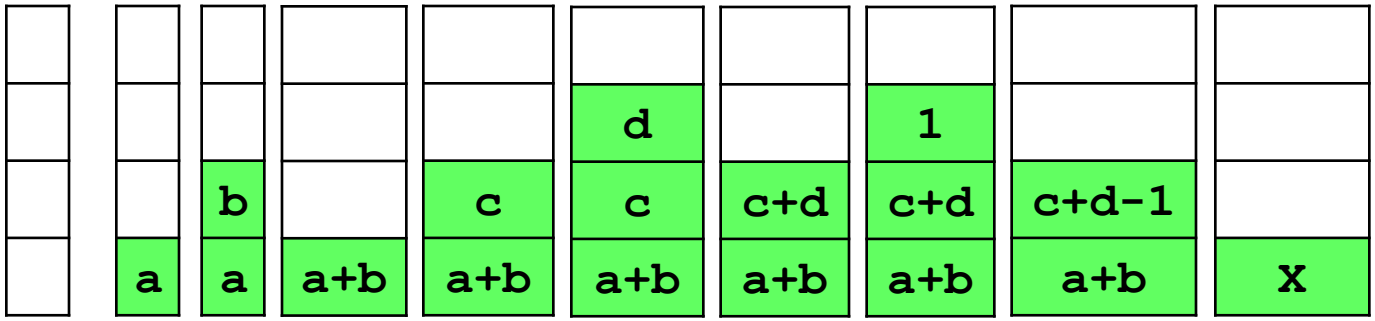
(2*4+3*5)*(2*3+18/3*2)*(12-3)

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

Вычисление выражений

Постфиксная форма:

X =	a b + c d + 1 - /



Алгоритм:

1) взять очередной элемент;
2) если это не знак операции, добавить его в стек;
3) если это знак операции, то
   - взять из стека два операнда;
   - выполнить операцию и записать результат в стек;
4) перейти к шагу 1.

Получение постфиксной формы из скобочной с помощью стека

Во-вторых, получение обратной польской записи из исходного выражения может осуществляться весьма просто на основе предложенного Дейкстрой алгоритма. Для этого вводится понятие стекового приоритета операций:

Операция	Приоритет
(	0
)	1
+ | -	2
* | /	3
Возведение в степень	4

Получение постфиксной формы из скобочной с помощью стека

Просматривается исходная строка символов слева направо, операнды сразу переписываются в выходную строку, а знаки операций заносятся в стек так:

а) если стек пуст, то операция из входной строки переписывается в стек;

б) если очередной символ из исходной строки есть открывающая скобка, то он проталкивается в стек;

в) закрывающая круглая скобка выталкивает все операции из стека до ближайшей открывающей скобки, сами скобки в выходную строку не переписываются, а уничтожают друг друга.

г) операция выталкивает из стека все операции с большим или равным приоритетом в выходную строку, после чего сама операция заталкивается в стек;

Процесс получения обратной польской записи выражения (A+B)*(C+D)-E:

Просматриваемый символ	1	2	3	4	5	6	7	8	9	10	11	12	13	
Входная строка		(	A	+	B	)	*	(	C	+	D	)	-	E
Состояние стека		(	(	+(	+(	*	(*	(*	+(*	+(*	*	-	-	
Выходная строка		A		B		+		C		D	+	*	E	-

На C++ можно и нужно использовать библиотеки STL:

```
#include <stack>
#include <string>
#include <cmath>
using namespace std;
int calc(string r){…}
string revpol(string s){…}
main(){    string s,r;
     getline(cin,s);
     r=revppol(s);
     cout<<r<<endl;
     cout<<calc(r);
}
```

```
string revpol(string s){ string r="",p="()+-*/^"; stack<char> v;
    int j=0,k,q[7]={0,1,2,2,3,3,4};
    for(auto e:s){
      if(e>='0'&&e<='9') r+=e;
      else {
          if(v.empty()||e=='(') v.push(e);
          else if(e==')'){
            while(v.top()!='(') {r+=v.top();v.pop();}
            v.pop(); }
          else{ k=q[p.find(e)];
             while(!v.empty()&&v.top()!='('&&k<=q[p.find(v.top())])
            { r+=v.top();v.pop(); }
             v.push(e); }
      }
    }
    while(!v.empty()){r+=v.top();v.pop();}
    return r;
}
```

```
int calc(string r){   // для однозначных операндов >=0
    stack<int> v;int a,b;
    for(auto e:r){
     if(e>='0‘ and e<='9')
              v.push(e-’0’);
     else {b=v.top(); v.pop(); a=v.top(); v.pop();
         if(e=='+')v.push(a+b);
         if(e=='-')v.push(a-b);
         if(e=='*')v.push(a*b);
         if(e=='/')if(b==0){cout<<"Error /0!";return 0;}
          else v.push(a/b);
         if(e=='^')v.push(pow(a,b));} }
   if(!v.empty()){a=v.top(); v.pop();return a;} else return 0;
}
```

Конец фильма