T: Odwrotna notacja polska.

1. Wyrażenia arytmetyczne można przedstawiać w różnej postaci:

a) **notacja infiksowa** — zapis konwencjonalny, w którym **operatory umieszczane są pomiędzy argumentami** (**operandami)**, dopuszcza używanie nawiasów, na przykład ***x + y***

b) **notacja prefiksowa — operatory umieszczane są przed argumentami**, na przykład **+ x y**

c) **notacja postfiksowa**, zwana **odwrotną notacją polską ONP** (ang. **Reverse Polish Notation, RPN**) — **operatory umieszczane są po argumentach**, na przykład **x y +**

*ONP przedstawiona została w 1920 roku przez polskiego matematyka Jana Łukasiewicza. Pozwala na **całkowitą rezygnację z użycia nawiasów w wyrażeniach**, jako że jednoznacznie określa kolejność wykonywanych działań.*

2. Zarówno algorytm konwersji notacji konwencjonalnej (infiksowej) na odwrotną notację polską (postfiksową), jak i algorytm obliczania wartości wyrażenia danego w ONP są bardzo proste i wykorzystują **stos**.

3. Przykłady:

Wyrażenia arytmetyczne Zapis wyrażenia wpostaci ONP:

4+8 4 8 +

(2+3)·5 2 3 + 5 ·

(7-3)/2 7 3 - 2/

2·(5-2) 2 5 2 - ·

(4-2)·(1+3) 4 2 - 1 3 + ·

((8-2)/3+(1+4)·2)/6 8 2 - 3 / 1 4 + 2 · + 6 /

((2+7)/3+(14−3)·4)/2 2 7 + 3 / 14 3 − 4 · + 2 /

4. Dokładnie rozpisany przykład:

((8-2)/3+(1+4)·2)/6 8 2 - 3 / 1 4 + 2 · + 6 /

Symbol pobrany z wyrażenia ONP	**Wykonane czynności**	**Aktualna zawartość stosu**

8	Odłożenie liczby 8 na stos	8
2	Odłożenie liczby 2 na stos	2 8

-	Pobranie dwóch wartości ze stosu:2 i 8, wykonanie działania 8 - 2 i odłożenie wyniku:6 na stoa	6
3	Odłożenie liczby 3 na stos	3 6
/	Pobranie dwóch wartości ze stosu: 3 i 6, wykonanie działania 6/3 i odłożenie wyniku: 2 na stos	2
1	Odłożenie liczby 1 na stos	1 2
4	Odłożenie liczby 4 na stos	4 1 2
+	Pobranie dwóch wartości ze stosu: 4 i 1, wykonanie działanie 1+4 i odłożenie wyniku: 5 na stos	5 2
2	Odłożenie liczby 2 na stos	2 5 2
·	Pobranie dwóch wartości ze stosu: 2 i 5, wykonanie działania 5 · 2 i odłożenie wyniku: 10 na stos	10 2
+	Pobranie dwóch wartości ze stosu: 10 i 2, wykonanie działania 2+ 10 i odłożenie wyniku: 12 na stos	12
6	Odłożenie liczby 6 na stos	6 12
/	Pobranie dwóch wartości ze stosu: 6 i 12 i odłożenie wyniu: 2 na stos	**2**

5. Ćwiczenia:

Zadanie 3.2. (*1 pkt*)

Poniżej zapisano wyrażenia w odwrotnej notacji polskiej (ONP). Wartościami tych wyrażeń są:

	Wyrażenie ONP	Wartość wyrażenia		
1.	7 3 – 2 /	2	**P**	**F**
2.	4 3 – 1 3 + *	8	**P**	**F**
3.	3 5 1 – *	12	**P**	**F**
4.	8 2 + 2 /	10	**P**	**F**

6. Algorytm w postaci listy kroków wyznaczający wartość wyrażenia opisanego w odwrotnej notacji polskiej:

Specyfikacja:

Dane: Łańcuch znaków: ***s*** (wyrażenie arytmetyczne zapisane w ONP).

Wynik: Wartość wyrażenia zapisanego w ONP: ***wynik.***

Lista kroków:

Krok 0. Wczytaj ***s***.

Krok 1. Przypisz ***n =*** liczba symboli w ***s***.

Krok 2. Wyzeruj stos.
Krok 3. Dla kolejnych wartości ***i: 0, 1, ..., n-1,*** wykonuj krok 4., a następnie

przejdź 4 do kroku 9.

Krok 4. Jeśli ***s[i]*** jest liczbą, odłóż ***s[i]*** na stos i przejdź do kroku 3., w przeciwnym wypadku przejdź do kroku 5.

Krok 5. Jeśli ***s[i]*** jest operatorem, wykonaj kroki 6. - 8., a następnie przejdź do

kroku 3. Krok 6. Zdejmij ze stosu jeden element ***x***.

Krok 7. Zdejmij ze stosu kolejny element - ***y***.

Krok 8. Odłóż na stos wartość wyrażenia ***y s[i] x***, gdzie ***s[i]*** jest operatorem wykonywanego działania.

Krok 9. Zdejmij ze stosu i wypisz liczbę, która jest wartością obliczonego

wyrażenia. Zakończ algorytm.

W DOMU

Napisz program wyznaczający wartość wyrażenia zapisanego w ONP (możesz wykorzystać podaną na lekcji listę kroków). W wyrażenie wprowadzonym z klawiatury należy uwzględnić tylko nieujemne liczby jednocyfrowe. Możesz wykorzystać dynamiczne struktury do reprezentacji stosu lub adapter stosu z biblioteki standardowej.

Zamiana wyrażenia na ONP:

https://eduinf.waw.pl/inf/alg/001_search/0102.php

C++

```
// Przekształcanie wyrażenia na ONP
// Data: 19.08.2012
// (C)2020 mgr Jerzy Wałaszek
//-----------------------------

#include <iostream>

using namespace std;

const int S_MAX = 100; // rozmiar stosu operatorów

// Zwraca priorytet operatora
//---------------------------

int p (char c)
{
  switch (c)
  {
    case '+'     :
    case '-'     : return 1;
    case '*'     :
    case '/'     : return 2;
    case '^'     : return 3;
```

```
  }
  return 0;
}

//-------------------------------------
// Tutaj rozpoczyna się program główny
//-------------------------------------

int main()
{
  char S [S_MAX];                  // stos operatorów
  int sptr = 0;                      // wskaźnik stosu
   char c;                                  // kolejny znak
wyrażenia

  while(true)
  {
     cin >> c;                            // czytamy znak z
wejścia

    if(c == '=')                     // koniec wyrażenia?
    {
       while(sptr) cout << S [--sptr] << ' ';  // jeśli
tak, na wyjście przesyłamy
                                                // wszystkie
operatory ze stosu
      break;                          // i przerywamy pętlę
```

```
    }

      switch (c)                              // analizujemy
odczytany znak
    {
      case ' ' : break;                 // spację ignorujemy
       case '(' : S [sptr++] = '('; // nawias otwierający
zawsze na stos
                  break;
         case ')' : while(S [sptr-1] != '(') // nawias
zamykający
                    cout << S [--sptr] << ' '; // ze stosu
przesyłamy na wyjście
                                               // wszystkie
operatory aż do nawiasu otw.
                    sptr--;               // usuwamy ze stosu
nawias otwierający
                  break;
      case '+' :                    // operator
      case '-' :
      case '*' :
      case '/' :
      case '^' : while(sptr)
                  {
                        if((p (c) == 3) || (p (c) > p (S
[sptr - 1]))) break;
                        cout << S [--sptr] << ' '; // na
wyjście przesyłamy ze stosu wszystkie
                      }                       // operatory o
```

```
wyższych priorytetach
                    S [sptr++] = c; // operator umieszczamy
na stosie
                    break;
          default:     cout << c << ' '; // inaczej znak
przesyłamy na wyjście
                    break;
     }
   }


   cout << endl;


   return 0;
}
```