Zadanie 1

Napisz program, który wykorzystując stos przekształca wyrażenia algebraiczne na Odwrotną Notację Polską.

Teoria:

- Stos jest strukturą liniowo uporządkowanych danych, z których jedynie ostatni element, zwany wierzchołkiem, jest w danym momencie dostępny.
- W wierzchołku odbywa się dołączanie nowych elementów, również jedynie wierzchołek można usunąć.
- dane są zapisywane i pobierane metodą Last-In-First-Out (LIFO) (pierwszy wchodzi, ostatni wychodzi).
- Działanie stosu jest często porównywane do stosu talerzy:
    - nie można usunąć talerza znajdującego się na dnie stosu nie usuwając wcześniej wszystkich innych.
    - nie można także dodać nowego talerza gdzieś indziej, niż na samą górę.
- Odwrotna notacja polska: znak wykonywanej operacji umieszczony jest po operandach (zapis postfiksowy), a nie pomiędzy nimi jak w zapisie algebraicznym (zapis infiksowy). Na przykład zamiast 2 + 2 jest 2 2 +
- ONP nie wymaga używania w wyrażeniach nawiasów, co sprawia, że obliczenia w tej notacji są bardzo łatwe do przeprowadzania na komputerze.

Przykłady

Notacja normalna	ONP
3 + 2	3 2 +
3 + 2 * 5	3 2 5 * +
((2+3)*5-7)/6	2 3 + 5 * 7 - 6 /
2 * (5 + 2)	2 5 2 + *
(7 + 3) * (5 - 2) ^2	7 3 + 5 2 - 2 ^*
4 / (3 - 1) ^(2 * 3)	4 3 1 - 2 3 * ^/

Algorytm przekształcania wyrażeń na ONP

Analizuj wyrażenie po jednym elemencie (stałej, zmiennej lub ograniczniku).

2. Jeśli element jest stałą lub nazwą zmiennej, przekaż go na wyjście.

3. Jeśli element jest operatorem, to:

(a) jeśli priorytet badanego operatora jest wyższy od priorytetu operatora zajmującego szczyt stosu lub jeśli stos jest pusty - dopisz go na stos;

(b) jeśli na szczycie stosu znajduje się operator o wyższym lub równym priorytecie - odczytaj ze stosu i prześlij na wyjście wszystkie operatory o priorytecie wyższym bądź równym, aż do wystąpienia na szczycie stosu operatora o priorytecie niższym od priorytetu operatora nadchodzącego z wejścia; element badany dopisz na stos;

4. Jeśli element jest nawiasem, to:

(a) jeśli trafiłeś na nawias otwierający, dopisz go na stos;

(b) jeśli trafiłeś na nawias zamykający: zdejmij wszystkie operatory ze stosu i przekaż je na wyjście, aż do trafienia na nawias otwierający; nawiasów nie wypisuj na wyjście.

5. Jeśli badane wyrażenie nie zostało wyczerpane - wróć do punktu pierwszego;

6. Jeśli badane wyrażenie zostało wyczerpane, odczytaj wszystkie operatory ze stosu i przekaż je na wyjście automatu.

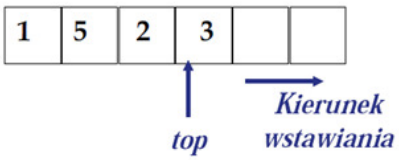
Implementacja stosu

- Wykorzystuje tablicę A złożoną z n elementów $A[i]$, gdzie n jest maksymalną liczbą spodziewanych elementów.
- Operacje polegają na manipulacji indeksami tablicy.
- Operacja wstawiania elementu do stosu - **PUSH**
- Operacja usuwania elementu ze stosu - **POP**

- $top[S]$ numer ostatniego elementu wstawionego do stosu.
- Stos składa się z elementów $S[1], \ldots, S[top[S]]$.
- $S[1]$ jest elementem na dnie stosu.
- $S[top[S]]$ jest elementem na szczycie stosu.
- Jeżeli $top[S] = 0$, to stos jest pusty.
- Do sprawdzenia, czy stos jest pusty używana jest operacja $Stack - Empty$.
- Próba zdjęcia elementu ze stosu sygnalizowana jest błędem niedomiaru.
- Jeżeli $top[S]$ jest większe niż ustalony z góry rozmiar tablicy, to stos jest przepełniony.

Push(S, x)

1: **if** $top[S] = length[S]$ **then**
2: error "przepełnienie"
3: **end if**
4: $top[S] = top[S] + 1$
5: $S[top[S]] = x$

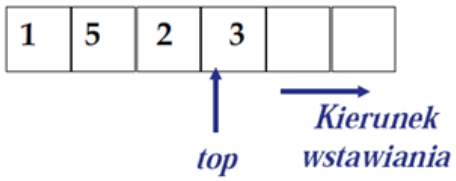


Stack-Empty(S)

1: **if** top[S]=0 **then**
2: **return** true
3: **else**
4: **return** false
5: **end if**

Pop(S)

1: **if** Stack-Empty(S) **then**
2: error "niedomiar"
3: **end if**
4: top[S]=top[S]-1
5: **return** S[top[S] + 1]



```
#inc lude <s t d i o . h>
#inc lude <s t d l i b . h>
#def ine DLUGOSC_MAX 20
i n t const STOS_PELNY=3;
i n t const STOS_PUSTY=2;
i n t const OK = 1;
i n t szczyt =0;
i n t Stos [DLUGOSC_MAX+1 ] ;
i n t push ( i n t x ) ;
i n t pop ( i n t w) ;
i n t StanStosu ( ) ;
void c l e a r ( ) { szczyt =0; }

int main(void){
 int a, i;
 for (i=0; i< DLUGOSC_MAX; i++ ) push(i);
 for (i=0; i< DLUGOSC_MAX; i++ ) {
     pop(&a);
     printf("%d, ",a);
 }
 return 0;
}
int StanStosu() {
  switch(szczyt) {
    case 0: return (STOS_PUSTY);
    case DLUGOSC_MAX+1: return (STOS_PELNY);
    default :return (OK);
   }
}

int push(int x){
  if (szczyt<=DLUGOSC_MAX) {
      Stos[szczyt++]=x;
      return (OK);
  }else
      return (STOS_PELNY);
}

int pop(int *w) {
  if (szczyt>0){
     *w=Stos[--szczyt];
     return (OK);
  }else
     return (STOS_PUSTY);
}
```