

Отчет по лабораторной работе

По дисциплине «Структуры и алгоритмы обработки данных»

На тему:

«Реализация стека/дека»

Выполнил:

Студент группы

БСТ1902

Игнатов В.С.

Вариант №5

Москва 2021

## Оглавление

Задание на лабораторную работу .....	3
Ход работы.....	5
Реализация структур данных .....	5
Задание №1 .....	7
Задание №2 .....	8
Задание №3 .....	10
Задание №4 .....	11
Задание №5 .....	12
Задание №6 .....	13
Задание №7 .....	14
Задание №8 .....	15
Задание №9 .....	16
Задание №10 .....	18
Задание №11 .....	19

## **Задание на лабораторную работу**

Реализовать следующие структуры данных:

Стек(stack). Операции для стека: инициализация, проверка на пустоту, добавление нового элемента в начало, извлечение элемента из начала

Дек (двусторонняя очередь, deque). Операции для дека: инициализация, проверка на пустоту, добавление нового элемента в начало, добавление нового элемента в конец, извлечение элемента из начала, извлечение элемента из конца.

Разработать программу обработки данных, содержащихся в заранее подготовленном txt-файле, в соответствии с заданиями, применив указанную в задании структуру данных. Результат работы программы вывести на экран и сохранить в отдельном txt-файле.

## **Задачи**

1. Отсортировать строки файла, содержащие названия книг, в алфавитном порядке с использованием двух деков.
2. Дек содержит последовательность символов для шифровки сообщений. Дан текстовый файл, содержащий зашифрованное сообщение. Пользуясь деком, расшифровать текст. Известно, что при шифровке каждый символ сообщения заменялся следующим за ним в деке по часовой стрелке через один.
3. Даны три стержня и  $n$  дисков различного размера. Диски можно надевать на стержни, образуя из них башни. Перенести  $n$  дисков со стержня А на стержень С, сохранив их первоначальный порядок. При переносе дисков необходимо соблюдать следующие правила:
  - на каждом шаге со стержня на стержень переносить только один диск;
  - диск нельзя помещать на диск меньшего размера;
  - для промежуточного хранения можно использовать стержень В.

Реализовать алгоритм, используя три стека вместо стержней А, В, С.  
Информация о дисках хранится в исходном файле.

4. Дан текстовый файл с программой на алгоритмическом языке. За один просмотр файла проверить баланс круглых скобок в тексте, используя стек.
5. Дан текстовый файл с программой на алгоритмическом языке. За один просмотр файла проверить баланс квадратных скобок в тексте, используя *дек*.
6. Дан файл из символов. Используя *стек*, за один просмотр файла напечатать сначала все цифры, затем все буквы, и, наконец, все остальные символы, сохраняя исходный порядок в каждой группе символов.
7. Дан файл из целых чисел. Используя *дек*, за один просмотр файла напечатать сначала все отрицательные числа, затем все положительные числа, сохраняя исходный порядок в каждой группе.
8. Дан текстовый файл. Используя *стек*, сформировать новый текстовый файл, содержащий строки исходного файла, записанные в обратном порядке: первая строка становится последней, вторая – предпоследней и т.д.

9. Дан текстовый файл. Используя *стек*, вычислить значение логического выражения, записанного в текстовом файле в следующей форме:

$$\langle \text{ЛВ} \rangle ::= \mathbf{T} \mid \mathbf{F} \mid (\mathbf{N}\langle \text{ЛВ} \rangle) \mid (\langle \text{ЛВ} \rangle \mathbf{A} \langle \text{ЛВ} \rangle) \mid (\langle \text{ЛВ} \rangle \mathbf{X} \langle \text{ЛВ} \rangle) \mid (\langle \text{ЛВ} \rangle \mathbf{O} \langle \text{ЛВ} \rangle),$$

где буквами обозначены логические константы и операции:

**T** – True, **F** – False, **N** – Not, **A** – And, **X** – Xor, **O** – Or.

10. Дан текстовый файл. В текстовом файле записана формула следующего вида:

$$\langle \text{Формула} \rangle ::= \langle \text{Цифра} \rangle \mid \mathbf{M}(\langle \text{Формула} \rangle, \langle \text{Формула} \rangle) \mid \mathbf{N}(\langle \text{Формула} \rangle, \langle \text{Формула} \rangle)$$

$$\langle \text{Цифра} \rangle ::= 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid$$

$$7 \mid 8 \mid 9 \text{ где буквами обозначены}$$

функции:

**М** – определение максимума, **Н** – определение минимума. Используя *стек*, вычислить значение заданного выражения.

11. Дан текстовый файл. Используя *стек*, проверить, является ли содержимое текстового файла правильной записью формулы вида:  
Формула > ::= < Терм > | < Терм > + < Формула > | < Терм > - < Формула >  
< Терм > ::= < Имя > | (< Формула >) < Имя > ::= x | y | z

## Ход работы

Реализация структур данных

Стек:

```
class Stack
{
    protected $stack;

    public function __construct() {
        // инициализация стека
        $this->stack = array();
    }

    public function len() {
        return count($this->stack);
    }

    public function peek() {
        return $this->stack[0];
    }

    public function push($item) {
        // проверяем, не полон ли наш стек
        if (count($this->stack) < $this->limit) {
            // добавляем новый элемент в начало массива
            array_unshift($this->stack, $item);
        } else {
            throw new RuntimeException('Стек переполнен!');
        }
    }

    public function pop() {
        if ($this->isEmpty()) {
            // проверка на пустоту стека
            throw new RuntimeException('Стек пуст!');
        } else {
            // Извлекаем первый элемент массива
            return array_shift($this->stack);
        }
    }
}
```

```

public function isEmpty() {
    return empty($this->stack);
}

public function typingInTXT($path, $newLine = true) {
    $formattedText = '';
    $length = count($this->stack);

    for ($i = 0; $i < $length; $i++) {
        if ($newLine)
            $formattedText .= $this->pop() . "\n";

        else
            $formattedText .= $this->pop();
    }

    $fp = fopen($path, "w");
    fwrite($fp, $formattedText);
    fclose($fp);
}
}

```

Дек:

```

class Deque
{
    protected $deque;

    public function __construct()
    {
        // инициализация дека
        $this->deque = array();
    }

    public function len() {
        return count($this->deque);
    }

    public function pushStart($item)
    {
        // Добавление в начало дека
        array_unshift($this->deque, $item);
    }

    public function pushEnd($item)
    {
        // Добавление в конец дека
        array_push($this->deque, $item);
    }

    public function popStart()
    {
        {
            if ($this->isEmpty()) {
                // проверка на пустоту дека
                throw new RuntimeException('Стек пуст!');
            } else {
                // Извлекаем начальный элемент дека
                return array_shift($this->deque);
            }
        }
    }
}

```

```

public function popEnd()
{
    if ($this->isEmpty()) {
        // проверка на пустоту дека
        throw new RuntimeException('Стек пуст!');
    } else {
        // Извлекаем конечный элемент дека
        return array_pop($this->deque);
    }
}

public function isEmpty()
{
    return empty($this->deque);
}

public function typingInTXT($path) {
    $formattedText = '';
    $length = count($this->deque);

    for ($i = 0; $i < $length; $i++) {
        $formattedText .= $this->popStart() . "\n";
    }

    $fp = fopen($path, "w");
    fwrite($fp, $formattedText);
    fclose($fp);
}
}

```

## Задание №1

```

require_once 'App/Deque.php';

$text = explode("\n", file_get_contents('../public/task№1.txt'));
$length = count($text);

$dequeFirst = new Deque();
$dequeSecond = new Deque();

foreach ($text as $book) {
    $dequeFirst->pushEnd($book);
}

recurse($length);

$dequeSecond->typingInTXT("public/task№1-answer.txt");

function recurse($iterator) {
    global $dequeSecond;
    global $dequeFirst;

    $dequeSecond->pushEnd($dequeFirst->popStart());

    if ($iterator <= 1)
        return;

    for ($i = 0; $i <= $iterator - 1; $i++) {
        $a = $dequeSecond->popEnd();
    }
}

```

```

        $b = $dequeFirst->popStart();

        if ($a < $b) {
            $dequeFirst->pushEnd($b);
            $dequeSecond->pushEnd($a);
        } else {
            $dequeFirst->pushEnd($a);
            $dequeSecond->pushEnd($b);
        }
    }

    recurse($iterator - 1);
}

```

Исходные данные:

1	Сто лет одиночества
2	Моби Дик, или Белый кит
3	Великий Гэтсби
4	Гроздь гнева
5	Улисс
6	Лолита
7	Шум и ярость
8	На маяк
9	Анна Каренина
10	Война и мир

Результат:

1	Анна Каренина
2	Великий Гэтсби
3	Война и мир
4	Гроздь гнева
5	Лолита
6	Моби Дик, или Белый кит
7	На маяк
8	Сто лет одиночества
9	Улисс
10	Шум и ярость

Задание №2

```
<?php
```



```

require_once 'App/Deque.php';

$text = preg_split('//u',
mb_strtolower(file_get_contents('./public/task№2.txt')));

$alph = 'абвгдеёжзийклмнопрстуфхцчшщъыьэюя';
$alph = preg_split('//u', $alph, -1, PREG_SPLIT_NO_EMPTY);
shuffle($alph);

$key = new Deque();

foreach ($alph as $letter)
    $key->pushEnd($letter);

function encode($a)
{
    global $key;

    for ($i = 0; $i < $key->len(); $i++) {
        $x = $key->popStart();

        if ($x == $a) {
            $key->pushEnd($x);
            $value = $key->popStart();
            $key->pushEnd($value);
            return $value;
        }

        $key->pushEnd($x);
    }
}

function decode($a)
{
    global $key;

    for ($i = 0; $i < $key->len(); $i++) {
        $x = $key->popEnd();

        if ($x == $a) {
            $key->pushStart($x);
            $value = $key->popEnd();
            $key->pushStart($value);
            return $value;
        }

        $key->pushStart($x);
    }
}

$encoded = '';
$decoded = '';

foreach ($text as $letter) {
    $encodedLetter = encode($letter);

    if ($encodedLetter)
        $encoded .= $encodedLetter;
    else
        $encoded .= $letter;
}

foreach (preg_split('//u', $encoded) as $letter) {

```

```


$decodedLetter = decode($letter);
if ($decodedLetter)
    $decoded .= $decodedLetter;
else
    $decoded .= $letter;
}

echo $encoded . "<br>";
echo $decoded;

```

Результат:

йуэмсв-пёяпчг ёчй.  
москва-третий рим.

 task№2.txt – Блокнот

Файл Правка Формат Вид Справка  
Москва-третий рим.

Задание №3

```

<?php

require_once 'App/Stack.php';

$kernelA = new Stack();
$kernelB = new Stack();
$kernelC = new Stack();
$counter = 1;

$d disks = [10, 7, 4, 3, 2, 1, 0];

foreach ($disks as $disk) {
    $kernelA->push($disk);
}

function move($a, $b)
{
    if ($a->len() == 0 && $b->len() > 0)
        $a->push($b->pop());
    elseif ($a->len() > 0 && $b->len() == 0)
        $b->push($a->pop());
    elseif ($a->peek() > $b->peek()) {
        $a->push($b->pop());
    } else
        $b->push($a->pop());

    // echo "<pre>";
    // echo print_r($a);
    // echo "</pre>";
    //
    // echo "<pre>";
    // echo print_r($b);
    // echo "</pre>";
}

```

```

if (count($disks) % 2 == 0) {
    while ($kernelC->len() != count($disks)) {

        move($kernelA, $kernelB);
        move($kernelA, $kernelC);
        move($kernelB, $kernelC);
    }
} else {
    while ($kernelC->len() != count($disks)) {
//      echo "Длина стержня A " . $kernelA->len() . "<br>";
//      echo "Длина стержня B " . $kernelB->len() . "<br>";
//      echo "Длина стержня C " . $kernelC->len() . "<br> <br>";

        move($kernelA, $kernelC);

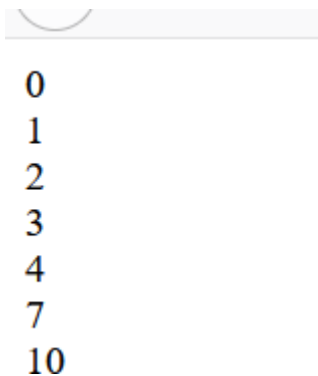
        if ($kernelC->len() == count($disks))
            break;

        move($kernelA, $kernelB);
        move($kernelB, $kernelC);
    }
}

while (!$kernelC->isEmpty()) {
    echo $kernelC->pop() . "<br>";
}

```

Результат:



```

0
1
2
3
4
7
10

```

Задание №4

```

<?php
require_once 'App/Stack.php';

$text = str_split(file_get_contents('./public/task№4.txt'));

$stack = new Stack();

balanceRoundedBrackets($text);

function balanceRoundedBrackets($text)
{
    global $stack;

```

```

for ($i = 0; $i < count($text); $i++) {

    if ($text[$i] === '(')
        $stack->push($text[$i]);

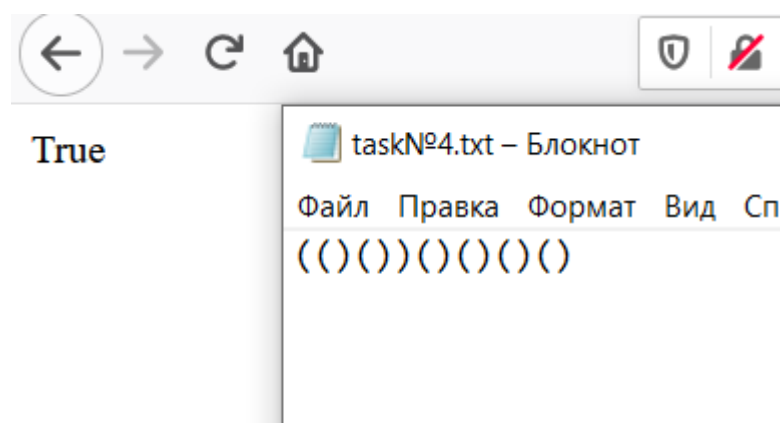
    elseif ($text[$i] === ')') {
        if ($stack->isEmpty()) {
            echo "False";
            return;
        }

        $stack->pop();
    }
}

if ($stack->isEmpty())
    echo "True";
else
    echo "False";
}

```

Результат:



Задание №5

```

<?php

require_once 'App/Deque.php';

$text = str_split(file_get_contents('./public/task№5.txt'));

$deque = new Deque();

balanceRectangleBrackets($text);

function balanceRectangleBrackets($text)
{
    global $deque;

    for ($i = 0; $i < count($text); $i++) {
        if ($text[$i] === '[')
            $deque->pushStart($text[$i]);
    }
}

```

```

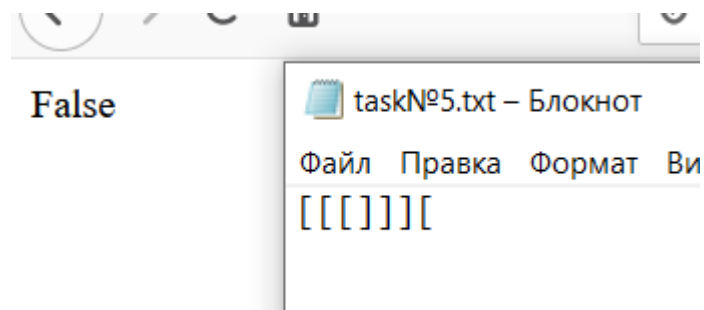
elseif ($text[$i] === ']') {
    if ($deque->isEmpty()) {
        echo "False";
        return;
    }

    $deque->popStart();
}

if ($deque->isEmpty())
    echo "True";
else
    echo "False";
}

```

Результат:



Задание №6

```

<?php
require_once 'App/Stack.php';

$text = preg_split('//u', file_get_contents('./public/task№6.txt'));

$stack = new Stack();

for ($i = 1; $i < 4; $i++) {
    foreach (array_reverse($text) as $symbol) {
        if ($i == 1 && !IntlChar::isalpha($symbol) &&
!IntlChar::isdigit($symbol)) {
            $stack->push($symbol);
        } else if ($i == 2 && IntlChar::isalpha($symbol)) {
            $stack->push($symbol);
        } else if ($i == 3 && IntlChar::isdigit($symbol) ) {
            $stack->push($symbol);
        }
    }
}

$stack->typingInTXT("public/task№6-answer.txt", false);

```

Результат:

task№6.txt – Блокнот

Файл Правка Формат Вид Справка

Мы рису123ем 93..\ \ \ когда алый ь }{закат @восхо53дит

task№6-answer.txt – Блокнот

Файл Правка Формат Вид Справка

1239353Мырисуемкогдаалыйъзакатвосходит ..\ \ \ }{ @

### Задание №7

```
<?php
require_once 'App/Deque.php';

$text = explode("\n", file_get_contents('../public/task№7.txt'));
$length = count($text);

$deque = new Deque();

for ($i = 0; $i < $length; $i++) {
    if ($i > 0) {
        $a = $deque->popEnd();

        if ($text[$i] <= 0 && $a >= 0) {
            while ($a >= 0) {
                $deque->pushStart($a);
                $a = $deque->popEnd();
            }

            $deque->pushEnd($a);

            $deque->pushEnd($text[$i]);

            $b = $deque->popStart();

            while ($b >= 0) {
                $deque->pushEnd($b);
                $b = $deque->popStart();
            }

            $deque->pushStart($b);
        } else {
            $deque->pushEnd($a);
            $deque->pushEnd($text[$i]);
        }
    } else {
        $deque->pushEnd($text[$i]);
    }
}

//echo "<pre>";
//echo print_r($deque);
//echo "</pre>";
```

```
$deque->typingInTXT("public/task№7-answer.txt");
```

Результат:

task№7.txt – Блокнот					task№7-answer.txt – Блокнот				
Файл	Правка	Формат	Вид	Справка	Файл	Правка	Формат	Вид	Справка
-1					-1				
2					-4				
-4					-13				
-13					-2				
-2					-5				
-5					-5				
3					2				
51					3				
32					51				
13					32				
-5					13				
8					8				
19					19				

Задание №8

```
<?php
require_once 'App/Stack.php';

$text = explode("\n", file_get_contents('./public/task№8.txt'));

$stack = new Stack();

foreach ($text as $str) {
    $stack->push($str);
}

$stack->typingInTXT("public/task№8-answer.txt");
```

Результат:

task№8.txt – Блокнот

Файл Правка Формат Вид Справка

Сто лет одиночества  
Моби Дик, или Белый кит  
Великий Гэтсби  
Гроздь гнева  
Улисс  
Лолита  
Шум и ярость  
На маяк  
Анна Каренина  
Война и мир

task№8-answer.txt – Блокнот

Файл Правка Формат Вид Справка

Война и мир  
Анна Каренина  
На маяк  
Шум и ярость  
Лолита  
Улисс  
Гроздь гнева  
Великий Гэтсби  
Моби Дик, или Белый кит  
Сто лет одиночества

Задание №9

```
<?php
require_once 'App/Stack.php';

$text = str_split(file_get_contents('..public/task№9.txt'));

$opstack = new Stack();
$vstack = new Stack();
$cur = 0;

while(true) {
    $read = false;

    if (!$opstack->isEmpty()) {
        $elem = $opstack->pop();
        if ($elem == "N") {
            $opstack->push($elem);
            if ($vstack->isEmpty()) {
                $read = true;
            } else {
                if ($vstack->pop() == "T") {
                    $vstack->push("F");
                } else {
                    $vstack->push("T");
                }
            }
            $opstack->pop();
        }
    } else if ($elem == "A") {
        $opstack->push($elem);
        if ($vstack->len() < 2) {
            $read = true;
        } else {
            $a = $vstack->pop();
```



```

        $b = $vstack->pop();
        if ($a == $b and $b == "T") {
            $vstack->push("T");
        } else {
            $vstack->push("F");
        }
        $opstack->pop();
    }
} else if ($elem == "O") {
    $opstack->push($elem);
    if ($vstack->len() < 2) {
        $read = true;
    } else {
        $a = $vstack->pop();
        $b = $vstack->pop();
        if ($a == "T" || $b == "T") {
            $vstack->push("T");
        } else {
            $vstack->push("F");
        }
        $opstack->pop();
    }
}
} else if ($elem == "X") {
    $opstack->push($elem);
    if ($vstack->len() < 2) {
        $read = true;
    } else {
        $a = $vstack->pop();
        $b = $vstack->pop();
        if ($a != $b) {
            $vstack->push("T");
        } else {
            $vstack->push("F");
        }
        $opstack->pop();
    }
}
} else if ($elem == "(") {
    $opstack->push($elem);
    $read = true;
} else if ($elem == ")") {
    $opstack->push($elem);
    $opstack->pop();
    $opstack->pop();
}
} else {
    $read = true;
}

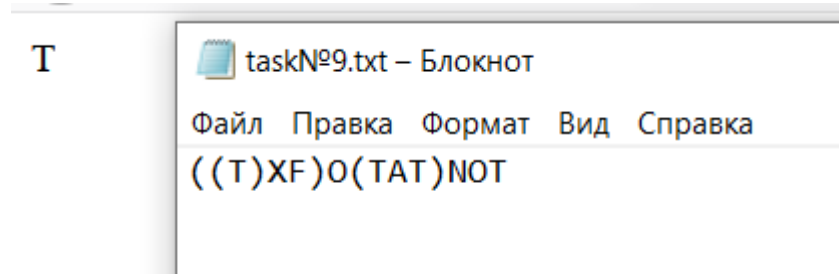
if ($read) {
    $i = $text[$cur];
    if (strpos('FT', $i) !== false) {
        $vstack->push($i);
    }
    if (strpos('AXON()', $i) !== false) {
        $opstack->push($i);
    }
    $cur++;
}

if ($cur == count($text) && $opstack->len() == 0) {
    break;
}
}

```

```
while (!$vstack->isEmpty()) {
    echo $vstack->pop();
}
```

Результат:



Задание №10

```
<?php

require_once 'App/Stack.php';

$text = str_split(file_get_contents('..public/task№10.txt'));

$op = new Stack();
$num = new Stack();
$num = '';
$cur = 0;

while ($cur < count($text)) {
    $i = $text[$cur];

    if (IntlChar::isDigit($i)) {
        $num .= $i;
    } else if ($num !== '') {
        $num->push((int) $num);
        $num = '';
    }

    if (strpos('MN', $i) !== false) {
        $op->push($i);
    }

    $cur++;
}

while (!$op->isEmpty()) {
    $a = $num->pop();
    $b = $num->pop();

    if ($a < $b) {
        list($a, $b) = [$b, $a];
    }

    if ($op->pop() == 'M') {
        $num->push($a);
    } else {
        $num->push($b);
    }
}
```

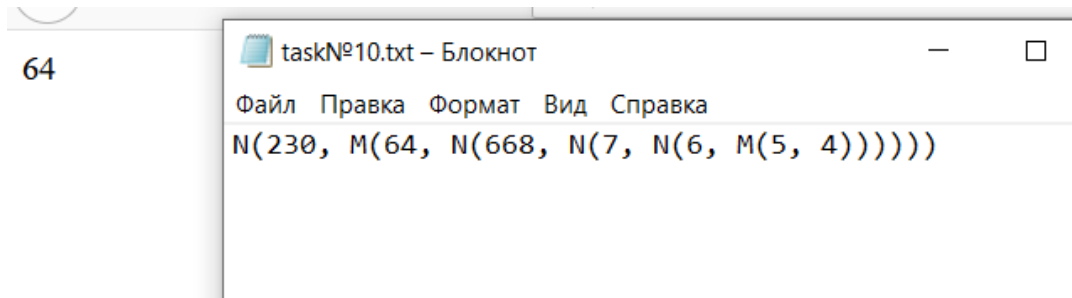
```

    }
}

while (!$nums->isEmpty()) {
    echo $nums->pop();
}

```

Результат:



Задание №11

```

<?php
require_once 'App/Stack.php';

$text = str_split(file_get_contents('./public/task№11.txt'));

echo var_dump(formula($text));

function formula($text) {
    $stack = new Stack();
    $cur = 0;

    while(true) {
        $read = false;

        if (!$stack->isEmpty()) {
            $elem = $stack->pop();
            if ($elem == '(') {
                $stack->push($elem);
                $read = true;
            } else if ($elem == ')') {
                $stack->push($elem);
                if ($stack->len() < 2 || $stack->pop() !== 'formula' ||
                    $stack->pop() !== '(')
                    return false;
                $stack->push('formula');
            } else if ($elem == 'formula') {
                $stack->push($elem);
                $newElem = $stack->pop();
                if ($stack->len() > 1 && strpos('+-', $newElem) !== false) {
                    $stack->push($newElem);
                    if (strpos('+-', $stack->pop()) !== false && $stack->
                        pop() == "formula")
                        $stack->push('formula');
                    else
                        return false;
                }
            }
        }
    }
}

```

```

        } else {
            $stack->push($newElem);
            $stack->push('formula');
            $read = true;
        }
    } else {
        $read = true;
    }
} else {
    $read = true;
}

if ($read) {
    $i = $text[$cur];

    if (strpos('xyz', $i) !== false) {
        $stack->push('formula');
    } elseif (strpos('()+-)', $i) !== false) {
        $stack->push($i);
    }

    $cur++;
}

if ($cur == count($text) && $stack->len() == 1)
    break;
}

return true;
}

```

Результат:

bool(false)

task№11.txt – Блокнот

Файл Правка Формат Вид Справка

|((x + y) + (x - y) + z))

## Вывод

Я изучил структуры данных – стек и дек, работу с ними, и реализовал их на практике.