

Dokumentacja Szyfr Polibiusza

wykonany w javascript.

Początek kodu, funkcja wrzucająca klucz

Będącym stringiem do tablicy, litera po literze

```
keyArr ▶ (5) ['k', 'l', 'u', 'c', 'z']
```

```
const keyToArr = function () {  
  let tmp;  
  tmp = klucz.split("");  
  
  for (let i = 0; i < klucz.length; i++) {  
    keyArr[i] = tmp[i];  
  }  
};
```

Ta funkcja tworzy tabelkę na stronie

Do której wpisuje się alfabet

```
const alfabetInputAppend = function () {  
  const tabela = document.querySelector(".alfa");  
  
  for (let i = 0; i < 5; i++) {  
    let tr = document.createElement("tr");  
  
    for (let j = 0; j < 7; j++) {  
      let input = document.createElement("input");  
      input.classList.add("form-control-sm", "input", "value" + i + j);  
      tr.appendChild(input);  
    }  
    tabela.appendChild(tr);  
  }  
};
```

Następnie po stworzeniu tabelki jest funkcja

Która wczytuje alfabet do tablicy dwu-

Wymiarowej, jest ona wywołana dopiero

Po wciśnięciu przycisku szyfruj bądź deszyfruj

```
const wczytajAlfabet = function () {  
  nowyAlfabet = [];  
  
  for (let i = 0; i < 5; i++) {  
    nowyAlfabet[i] = [];  
  
    for (let j = 0; j < 7; j++) {  
      let input = document.querySelector(".value" + i + j);  
  
      if (input) {  
        let inputValue = input.value;  
        nowyAlfabet[i][j] = inputValue;  
      }  
    }  
  }  
};
```

Funkcja czyPuste sprawdza alfabet wczytany

Przez poprzednią funkcję i określa czy zawiera

On puste stringi by potem stworzyć warunek

Funkcja wywołuje się po funkcji wczytajalfabet

```
const czyPuste = function () {  
  for (let i = 0; i < 5; i++) {  
    for (let j = 0; j < 7; j++) {  
      if (nowyAlfabet[i][j] === "") {  
        ilePustych++;  
      }  
    }  
  }  
};
```

Następnie jest obszerna funkcja Szyfrowanie, na początku kilka deklaracji zmiennych oraz warunków sprawdzających czy klucz oraz alfabet nie są puste, jeśli tak, wyświetlane na stronie są alerty oraz brane są domyślne wartości klucza oraz alfabetu

```
const szyfrowanie = function () {
  wczytajAlfabet();
  czyPuste();
  let kluczInput = document.querySelector(".klucz");
  klucz = kluczInput.value;
  iloscKolumn = klucz.length;
  if (ilePustych === 0) {
    alfabet = nowyAlfabet;
  } else {
    alert(
      "nie wprowadziłeś wszystkich liter alfabetu, zostanie wykorzystana domyślna kolejność alfabetu"
    );
  }
  if (klucz === "") {
    alert(
      "nie wprowadziłeś klucza, klucz przyjmie domyślną wartość: 'klucz'"
    );
    klucz = "klucz";
  }
  ilePustych = 0;
  alfabet = [
    //      0      1      2      3      4      5      6
    /*0*/ ["a", "ą", "b", "c", "ć", "d", "e"],
    /*1*/ ["ę", "f", "g", "h", "i", "j", "k"],
    /*2*/ ["l", "ł", "m", "n", "ń", "o", "ó"],
    /*3*/ ["p", "q", "r", "s", "ś", "t", "u"],
    /*4*/ ["v", "w", "x", "y", "z", "ż", "ź"],
  ];
};
```

Następnie funkcja wczytująca input Użytkownika przeznaczony do zaszyfrowania. Jej wynikiem jest tablica charArr gotowa do zaszyfrowania

```
let charArr = [];
let inputUser;
let lokacjaIndeks = [];
const wczytajinput = function () {
  inputUser = document.querySelector(".tekstjawnny").value;
  inputUser = inputUser.replace(/\s+/g, "");
  charArr = inputUser.split("");
};
```

Następnie funkcja indeks przyjmująca w tym wypadku za parametr wyżej wspomnianą Tablicę charArr. Przechodzi ona podwójną Pętlę for przez tablice alfabet i szuka indeksu Liter zawartych w charArr jej przykładowy Wynik przy inpcie 'goryl' i domyślnym ustawieniu alfabetu:

```
const indeks = function (arr) {
  let i = 0;
  let j = 0;
  let x = 0;
  while (x < arr.length) {
    for (i = 0; i < alfabet.length; i++) {
      for (j = 0; j < 7; j++) {
        if (alfabet[i][j] === arr[x]) {
          lokacjaIndeks[x] = "" + i + j;
          x++;
          i = 0;
          j = 0;
        }
      }
    }
  }
};
keyToArr();
wczytajinput();
indeks(charArr);
```

```
lokacjaIndeks ▶ (5) ['12', '25', '32', '43', '20']
```

```
alfabet = [
  //      0      1      2      3      4      5      6
  /*0*/ ["a", "ą", "b", "c", "ć", "d", "e"],
  /*1*/ ["ę", "f", "g", "h", "i", "j", "k"],
  /*2*/ ["l", "ł", "m", "n", "ń", "o", "ó"],
  /*3*/ ["p", "q", "r", "s", "ś", "t", "u"],
  /*4*/ ["v", "w", "x", "y", "z", "ż", "ź"],
];
```

Następnie mamy pętlę która bierze lokacje indeks z poprzedniego bloku oraz rozбивa tablicę z dwucyfrowymi indeksami na pojedyncze cyfry:

```
for (let i = 0; i < lokacjaIndeks.length; i++) {  
  let digits = lokacjaIndeks[i].toString().split('').map(Number);  
  splitedLokacja = splitedLokacja.concat(digits);  
}
```

splitedLokacja

▶ (10) [1, 2, 2, 5, 3, 2, 4, 3, 2, 0]

Funkcja szyfrkluczem to właściwa funkcja

Szyfrująca:

Pierwsza pętla przygotowuje tablicę

Wypełniając ją pustymi rzędami

Druga pętla do pierwszego rzędu 'wpycha'

Klucz do pierwszego rzędu

By następnie przejść do głównej podwójnej

Pętli która wypełnia resztę rzędów cyframi

Z uprzedni przygotowanej

tablicy splitedlokacja

jednakże jeżeli cyfry w tablicy splitedlokacja

się kończą a rzędy w tablicy szyfrkluczArr nie są jeszcze pełne dorzucana jest liczba 9

```
const szyfrKluczem = function () {  
  iloscKolumn = klucz.length;  
  
  tmp2 = lokacjaIndeks.length * 2;  
  rowNumber = Math.ceil(tmp2 / iloscKolumn) + 1;  
  
  let x = 0;  
  for (let i = 0; i < rowNumber - 2; i++) {  
    szyfrkluczArr.push([]);  
  }  
  for (let i = 0; i < iloscKolumn; i++) {  
    szyfrkluczArr[0][i] = keyArr[i];  
  }  
  for (let j = 1; j < rowNumber; j++) {  
    for (let i = 0; i < iloscKolumn; i++) {  
      if (typeof splitedLokacja[x] == "undefined") {  
        splitedLokacja[x] = 9;  
        szyfrkluczArr[j][i] = splitedLokacja[x];  
        x++;  
      } else {  
        szyfrkluczArr[j][i] = splitedLokacja[x];  
        x++;  
      }  
    }  
  }  
};
```

input ▶ (6) ['g', 'o', 'r', 'y', 'l', 'a']

szyfrkluczArr ▼ (4) [Array(5), Array(5), Array(5), Array(5)]

- ▶ 0: (5) ['k', 'l', 'u', 'c', 'z']
- ▶ 1: (5) [1, 2, 2, 5, 3]
- ▶ 2: (5) [2, 4, 3, 2, 0]
- ▶ 3: (5) [0, 0, 9, 9, 9]

length: 4

▶ [[Prototype]]: Array(0)

Następnie kolejny ważny blok kodu

Mianowicie transpozycja, najpierw

Jednak funkcja keyOrd która do

Sortuje klucz alfabetycznie oraz

Wrzuca to do tablicy keyOrder

```
const transposition = function () {  
  let x = 0;  
  let sortedKey = [...keyArr].sort();  
  // [...keyArr] tworzenie nowej tablicy zeby przy  
  // wywołaniu funkcji drugi raz nie tablica nie byla poprzestawiana  
  const keyOrd = function () {  
    while (x !== keyArr.length)  
      for (let i = 0; i < klucz.length; i++) {  
        if (szyfrkluczArr[0][i] === sortedKey[x]) {  
          keyOrder[x] = i;  
          x++;  
        }  
      }  
  }  
};  
keyOrd();
```

Następnie właściwa transpozycja
 Zmienna extractedColumn wyciąga
 Pierwszą kolumną względem już
 Wcześniej uporządkowanego klucza

```
let extractedColumn = szyfrkluczArr.map((row) => [row[keyOrder[0]]]);

console.log(extractedColumn);
for (let i = 1; i < klucz.length; i++) {
  let extractedColumn2 = szyfrkluczArr.map((row) => [row[keyOrder[i]]]);

  extractedColumn = extractedColumn.map((column, index) =>
    column.concat(extractedColumn2[index])
  );
}
```

Następnie za pomocą pętli wyciąga z oryginalnej
 Tablicy kolejne kolumny w kolejności alfabetycznej
 Oraz spaja je przy pomocy concat w jedną tablicę

```
▼ (4) [Array(1), Array(1), Array(1), Array(1)] t
  ▶ 0: ['c']
  ▶ 1: [5]
  ▶ 2: [2]
  ▶ 3: [9]
  length: 4
  ▶ [[Prototype]]: Array(0)
```

W tym przypadku jedynie kolumna c jest
 przestawiana

```
▼ (4) [Array(5), Array(5), Array(5), Array(5)] t
  ▶ 0: (5) ['c', 'k', 'l', 'u', 'z']
  ▶ 1: (5) [5, 1, 2, 2, 3]
  ▶ 2: (5) [2, 2, 4, 3, 0]
  ▶ 3: (5) [9, 0, 0, 9, 9]
  length: 4
  ▶ [[Prototype]]: Array(0)
```

Następnie ta tablica jest wrzucana do tablicy
 jednowymiarowej

```
let doStringa = [];
let z = 0;
for (let i = 1; i < extractedColumn.length; i++) {
  for (let j = 0; j < keyArr.length; j++) {
    doStringa[z] = extractedColumn[i][j];
    z++;
  }
}
```

```
input ▶ (6) ['g', 'o', 'r', 'y', 'l', 'a']
doStringa ▶ (15) [5, 1, 2, 2, 3, 2, 2, 4, 3, 0, 9, 0, 0, 9, 9]
```

oraz na sam koniec tablicę doStringa konwertujemy na string oraz usuwamy przecinki a także
 dodajemy spacje co drugi znak dla lepszej czytelności

```
let zaszyfrowanaWiadomosc = doStringa.join("").replace(/,/g, "");
let result = zaszyfrowanaWiadomosc.replace(/(.{2})/g, "$1 "); //d
let SzyfrResult = document.querySelector(".zaszyfrowanytekst");
SzyfrResult.value = result;
};
transposition();
```

końcowy wynik pojawia się na stronie w zablokowanym inputcie o klasie zaszyfrowany tekst

tekst jawny

goryla	szyfruj
--------	---------

tekst zaszyfrowany

51 22 32 24 30 90 09 9

Na sam koniec w kodzie dodajemy listener na przycisk szyfrowanie który wywołuje całą funkcję.

```
const submitBtn = document.querySelector(".submitBtn");
submitBtn.addEventListener("click", szyfrowanie);
```

DESZYFRACJA

Na samym starcie analogicznie

Jak przy szyfracji wywołują się

Funkcje odpowiedzialne za

Wczytanie alfabetu oraz

Warunki sprawdzające czy

Alfabet oraz klucz są puste

A także od razu następuje

Przekształcenie klucza w tablice

```
const funkcjaDeszyfracji = function () {
  wczytajAlfabet();
  czyPuste();
  let kluczInput = document.querySelector(".klucz");
  klucz = kluczInput.value;
  if (ilePustych === 0) {
    alfabet = nowyAlfabet;
  } else {
    alert(
      "nie wprowadziłeś wszystkich liter alfabetu, zostanie wykorzystana domyślna kolejność alfabetu"
    );
  }
  if (klucz === "") {
    alert(
      "nie wprowadziłeś klucza, klucz przyjmie domyślną wartość: 'klucz'"
    );
    klucz = "klucz";
  }
  keyToArr();
}
```

Zaraz po tym kilka deklaracji zmiennych

Oraz wczytanie inputu od użytkownika

```
let charArrayDecipher = [];
let inputUser2;
let odszyfrowanyTekst = [];
let deszyfrkluczArr = [
  //      0      1      2      3      4      5      6
  /*0*/ ["", ""],
  /*1*/ ["", ""],
];

const wczytajinput2 = function () {
  inputUser2 = document.querySelector(".tekstniejawny").value;
  inputUser2 = inputUser2.replace(/\s+/g, "");
  charArrayDecipher = inputUser2.split("");
  // console.log("xd");
};
wczytajinput2();
```

Kolejnie, przygotowujemy tablicę do właściwej

Deszyfracji, czyli analogicznie do szyfrowania:

Pierwsza pętla przygotowuje tablicę

Wypełniając ją pustymi rzędami

Druga pętla do pierwszego rzędu 'wpycha'

Klucz do pierwszego rzędu

By następnie przejść do głównej podwójnej

Pętli która wypełnia resztę rzędów cyframi

Z inputu omijając pierwszy

Rząd z kluczem (iteracja j od 1)

Z uprzednio przygotowanej tablicy

deszyfrkluczArr

wynik po wykonaniu pętli

```
const stringTo2dAr = function () {
  let tmp2 = charArrayDecipher.length * 2;
  let rowNumber = Math.ceil(tmp2 / iloscKolumn) + 1;

  for (let i = 0; i < rowNumber - 2; i++) {
    deszyfrkluczArr.push([]);
  }
  let sortedKeyArr = [...keyArr].sort();
  for (let i = 0; i < iloscKolumn; i++) {
    deszyfrkluczArr[0][i] = sortedKeyArr[i];
  }
  let x = 0;
  for (let j = 1; j < rowNumber; j++) {
    for (let i = 0; i < iloscKolumn; i++) {
      if (x < charArrayDecipher.length) {
        deszyfrkluczArr[j][i] = charArrayDecipher[x];
        x++;
      }
    }
  }
};
stringTo2dAr();
```

deszyfrkluczArr

```
▼ (7) [Array(5), Array(5), Array(5), Array(5),
  ▶ 0: (5) ['c', 'k', 'l', 'u', 'z']
  ▶ 1: (5) ['5', '1', '2', '2', '3']
  ▶ 2: (5) ['2', '2', '4', '3', '0']
  ▶ 3: (5) ['9', '0', '0', '9', '9']
```

następnie tak samo jak w przypadku szyfrowania

tworzona jest nowa tablica z kluczem rozdzielonym

litera po literze oraz później ten klucz w pętli while

jest sortowany alfabetycznie

```
const keyToArr2 = function () {
  let tmp;
  tmp = klucz.split("");

  for (let i = 0; i < klucz.length; i++) {
    keyArr2[i] = tmp[i];
  }
};
keyToArr2();

let keyOrder2 = [];
let y = 0;

while (y !== keyArr2.length)
  for (let i = 0; i < klucz.length; i++) {
    if (deszyfrkluczArr[0][i] === keyArr2[y]) {
      keyOrder2[y] = i;
      y++;
    }
  }
}
```

Następnie transpozycja

Analogicznie do szyfracji

```
deszyfrkluczArr = deszyfrkluczArr.filter((row) => row.length > 0);
let extractedColumn3;
const transpozycja2 = function () {
    extractedColumn3 = deszyfrkluczArr.map((row) => [row[keyOrder2[0]]]);

    for (let i = 1; i < klucz.length; i++) {
        let extractedColumn5 = deszyfrkluczArr.map((row) => row[keyOrder2[i]]);

        extractedColumn3 = extractedColumn3.map((column, index) =>
            column.concat(extractedColumn5[index])
        );
    }
};
transpozycja2();
```

Następnie extractedcolumn3 czyli posortowana

Względem klucza tablica po transpozycji jest

Przekształcana w tablice jednowymiarową oraz

Usuwany jest pierwszy rząd z kluczem

```
const usunPierwszyRow = function () {
    let doStringa = [];
    let z = 0;

    for (let i = 1; i < extractedColumn3.length; i++) {
        for (let j = 0; j < keyArr2.length; j++) {
            z++;
            if (extractedColumn3[i][j] != 9)
                tablicaDoOdszyfr[z - 1] = extractedColumn3[i][j];
        }
    }
};
```

```
extractedColumn3 ▾ (4) [Array(5), Array(5), Array(5), Array(5)]
  ▶ 0: (5) ['k', '1', 'u', 'c', 'z']
  ▶ 1: (5) ['1', '7', '7', '8', '3']
  ▶ 2: (5) ['2', '4', '3', '2', '0']
  ▶ 3: (5) ['0', '0', '9', '9', '9']
    length: 4
  ▶ [[Prototype]]: Array(0)
```

```
tablicaDoOdszyfr ▶ (12) ['1', '7', '7', '8', '3', '2', '4', '3', '2', '0', '0', '0']
```

Na koniec funkcja deszyfracja

Czyli sprawdzenie indeksów z

Wcześniej przygotowanej tablicy

Z indeksami alfabetu

Oraz dodanie listenera by

Końcowy wynik wyświetlić

W inputcie

```
const deszyfracja = function (arr) {
    let z = 0;
    for (let i = 0; i < arr.length; i += 2) {
        odszyfrowanyTekst[z] = alfabet[arr[i]][arr[i + 1]];
        z++;
    }
};
deszyfracja(tablicaDoOdszyfr);
let koncowyWynik = odszyfrowanyTekst.join("");
let SzyfrResult2 = document.querySelector(".odszyfrowanytekst");
SzyfrResult2.value = koncowyWynik;
};
submitBtn2.addEventListener("click", funkcjadeszyfracji);
```

tekst niejawny

tekst odszyfrowany

Input podczas screenów w tej prezentacji najpierw jest 'goryl', następnie 'goryla' żeby pokazać że gdy w rzędzie brakuje cyfer, jest on dopełniany przy pomocy 9

Ponadto podczas tworzenia prezentacji zorientowałem się, że nie utworzyłem **dodatkowej funkcji szyfrującej**, dlatego na screenshotach podczas omawiania deszyfracji mogą się pojawiać nie wyjaśnione wcześniej cyfry 7 oraz 8

Dodatkowa funkcja szyfrująca:

Szyfracja

Funkcja, tuż przed przemianieniem splittedLokacja w podwójny array

splitedLokacja

▶ (12) [1, 2, 2, 5, 3, 2, 4, 3, 2, 0, 0, 0]

```
szyfrkluczArr ▼ (4) [Array(5), Array(5), Array(5), Array(5)]  
  ▶ 0: (5) ['k', 'l', 'u', 'c', 'z']  
  ▶ 1: (5) [1, 2, 2, 5, 3]  
  ▶ 2: (5) [2, 4, 3, 2, 0]  
  ▶ 3: (5) [0, 0, 9, 9, 9]  
  length: 4  
  ▶ [[Prototype]]: Array(0)
```

Zlicza ile dwójek oraz piątek

Występuje w tablicy splittedLokacja

Następnie losowany jest zakres

Od zera do tylu ile dwójek/piątek

Występuje w tablicy

By później zamienić daną ilość

Razy dwójki na 7

Piątki na 8

Może się zdarzyć przypadek w

Którym funkcja w ogóle nie zmieni

Tablicy, ale na tym polega losowość

```
const dodatkowaSzyfracja = function (arr) {  
  let ileDwojek = 0;  
  let ilePiatek = 0;  
  for (let i = 0; i < arr.length; i++) {  
    if (arr[i] === 2) ileDwojek++;  
    if (arr[i] === 5) ilePiatek++;  
  }  
  function losowaLiczbaZakres(min, max) {  
    return Math.floor(Math.random() * (max - min + 1)) + min;  
  }  
  const ileLosDwojek = losowaLiczbaZakres(0, ileDwojek);  
  const ileLosPiatek = losowaLiczbaZakres(0, ilePiatek);  
  let tmp3 = 0;  
  for (let i = 0; i < arr.length; i++) {  
    if (arr[i] === 2 && tmp3 < ileLosDwojek) {  
      arr[i] = 7;  
      tmp3++;  
    }  
  }  
  let tmp4 = 0;  
  for (let i = 0; i < arr.length; i++) {  
    if (arr[i] === 5 && tmp4 < ileLosPiatek) {  
      arr[i] = 8;  
      tmp4++;  
    }  
  }  
};  
dodatkowaSzyfracja(splitedLokacja);
```


Poniżej przedstawiam przykładowe działanie dodatkowej funkcji szyfrującej

```
splitedLokacja przed ▶ (12) [1, 2, 2, 5, 3, 2, 4, 3, 2, 0, 0, 0]
ileDwojek 4
ileLosDwojek 1
ilePiatek 1
ileLosPiatek 1
splitedLokacja po ▶ (12) [1, 7, 2, 8, 3, 2, 4, 3, 2, 0, 0, 0]
input ▶ (6) ['g', 'o', 'r', 'y', 'l', 'a']
```

Zostal wylosowany zakres 1 dla dwojek 1 dla piątek, co oznacza ze w splitedlokacja jedna z dwójka staje się siódmką a jedna piątka staje się ósmką

Deszyfracja

W przypadku deszyfracji funkcja
Iteruje przez całą tablice i po prostu
Zamienia spowrotem wszystkie
Siódemki na dwójki oraz
Ósemki na piątki

```
const dodatkowaDeszyfracja = function (arr) {
  for (let i = 0; i < arr.length; i++) {
    if (arr[i] == 7) arr[i] = 2;
    if (arr[i] == 8) arr[i] = 5;
  }
};
dodatkowaDeszyfracja(tablicaDoOdszyfr);
```

```
Przed deszyfracją: ▶ (12) ['1', '7', '2', '8', '3', '2', '4', '3', '2', '0', '0', '0']
Po deszyfracji: ▶ (12) ['1', 2, '2', 5, '3', '2', '4', '3', '2', '0', '0', '0']
```

Podsumowanie

Warto zaznaczyć że w przypadku zostawienia pustego pola klucz oraz tabeli na alfabet przyjmą one wartości domyśle tj alfabet:

Klucz: 'klucz'

```
alfabet = [
  //      0      1      2      3      4      5      6
  /*0*/ ["a", "ą", "b", "c", "ć", "d", "e"],
  /*1*/ ["ę", "f", "g", "h", "i", "j", "k"],
  /*2*/ ["l", "ł", "m", "n", "ń", "o", "ó"],
  /*3*/ ["p", "q", "r", "s", "ś", "t", "u"],
  /*4*/ ["v", "w", "x", "y", "z", "ż", "ź"],
];
```

Przykładowe działanie aplikacji:

tekst jawny

tajna wiadomosc

szyfruj

tekst zaszyfrowany

07 00 55 57 72 39 30 39

tekst niejawny

03 50 10 57 30 44 11 07 0

deszyfruj

tekst odszyfrowany

tajnwiadomosc

klucz

tekst jawny

tajne haslo

szyfruj

tekst zaszyfrowany

1 12 82 38 10 00 11 21 46

tekst niejawny

03 03 12 82 38 10 00 11 2

deszyfruj

tekst odszyfrowany

tajnehaslo

a	h	s	r	p	u	ć
a	b	g	f	o	m	l
e	t	c	e	i	n	k
t	u	d	z	ż	j	ń
v	w	y	z	ó	q	ś

haslo

Wykonał:

Filip Stochel

Nr albumu: 14645

Grupa: lab3/2/PROG S

Git: <https://github.com/tsquix/szyfr-polibiusza>