**II Liceum Ogólnokształcące im. Mikołaja Kopernika w Lesznie z Oddziałami Dwujęzycznymi i Międzynarodowymi**

**ul. Prusa 33**

**64-100 Leszno**

# Algorytm szyfrowania – szyfr Cezara

**Praca przygotowana przez:**

**Mateusza Kmaka i Jakuba Szymkowiaka**

**klasa 3E**

**pod kierunkiem mgr. Dominika Siecińskiego**

**Leszno, 13.11.2018r.**

# Algorytm szyfrowania – szyfr Cezara

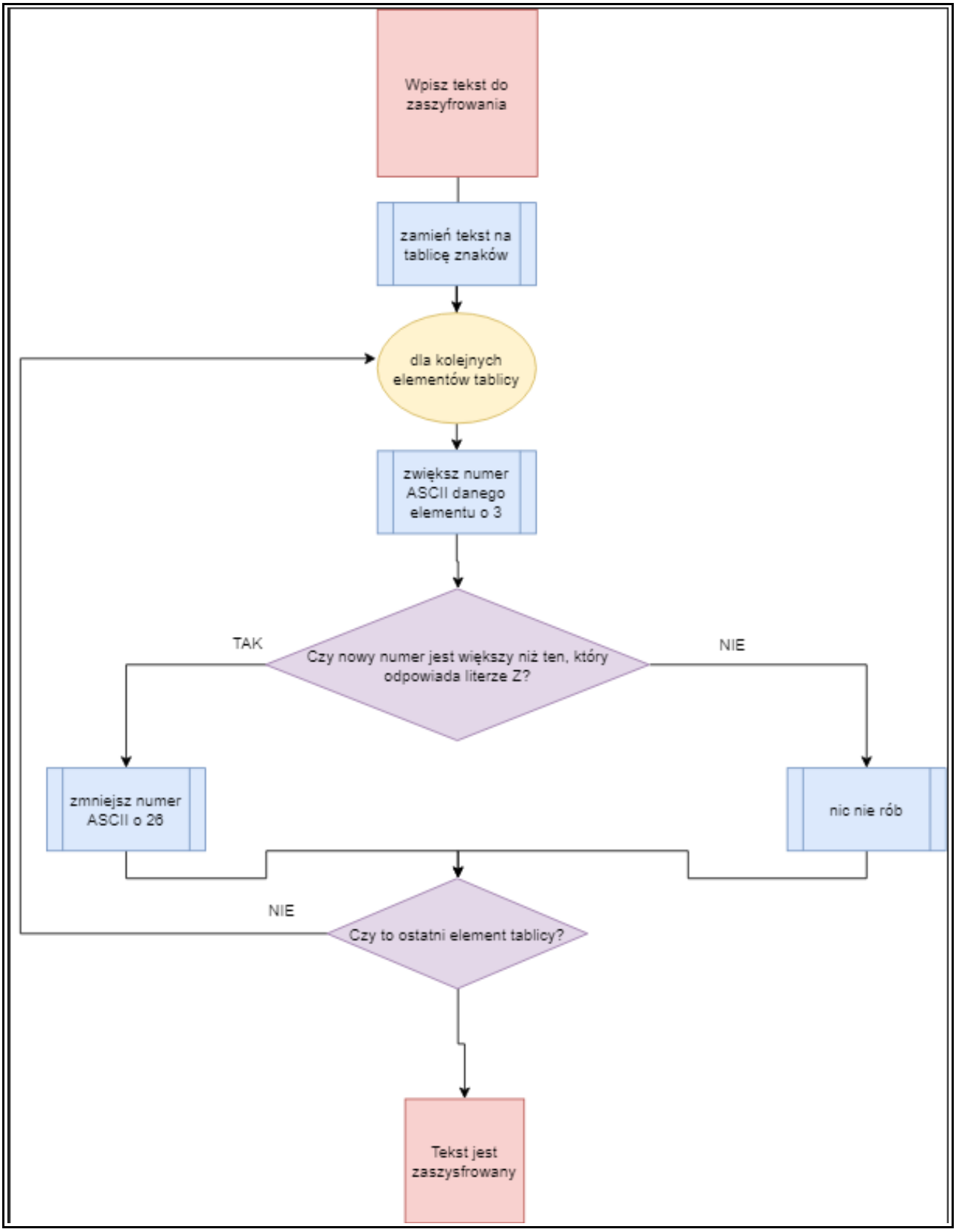
W tej pracy postanowiliśmy rozważyć twierdzenie Alberta Einsteina: *Jeśli nie potrafisz wytłumaczyć czegoś w prosty sposób, to znaczy, że tak naprawdę tego nie rozumiesz.* Aby potwierdzić tą hipotezę, postanowiliśmy przeprowadzić wykład w naszej klasie na temat algorytmu szyfrowania, a konkretnie szyfru Cezara. Algorytm ten działa w ten sposób, że każdą literę w szyfrowanym słowie lub zdaniu zamienia na literę o 3 miejsca dalej w alfabecie. W ten sposób słowo ABC staje się słowem DEF. Aby odszyfrować słowo należy działać odwrotnie, czyli zamieć litery na te o 3 miejsca wstecz. Algorytm ten przedstawiliśmy klasie na schemacie blokowym, w pseudokodzie, Swifcie, C++, Javie i Pythonie. Te przykłady przedstawimy teraz poniżej.

```
1
2  import string
3
4  word = input()
5
6  def moveLetter(L, n, alph):
7          if L in alph:
8                  oldIndex = alph.index(L)
9                  newIndex = (oldIndex + n) % len(alph)
10                 return alph[newIndex]
11 def moveWord(word, n, alph = string.ascii_lowercase):
12         alphUpper = alph.upper()
13         result = ""
14         for L in word:
15                 if L in alph:
16                         result += moveLetter(L, n, alph)
17                 elif L in alphUpper:
18                         result += moveLetter(L, n, alphUpper)
19                 else:
20                         result += L
21         return result
22
23 print(moveWord(word, 3))
24
```

**Zdjęcie nr 1 przedstawia algorytm w pseudokodzie**

```
word = "Jakieś zdanie"
n = 3  //Ilość przesunięć, w przypadku Cezara 3
alfabet = ["A", "B" ...] //wszystkie litery w tablicy
alfabetMaly = alfabet.lower() //zwraca alfabet z małymi literami
result = ""


for litera = word[0] to word[N-1] do    //N - liczba elementów tablicy
        if litera in alfabet then
                oldIndex = alfabet.index(litera)
                newIndex = (oldIndex + n) % length(alfabet)
                result += alfabet[newIndex]

        else if litera in alfabetMaly then
                oldIndex = alfabetMaly.index(litera)
                newIndex = (oldIndex + n) % length(alfabetMaly)
                result += alfabetMaly[newIndex]

        else
                result += litera
        endif
endfor

return result
```

**Zdjęcie nr 2 przedstawia algorytm w języku Python 3**




**Zdjęcie nr 3 przedstawia schemat blokowy algorytmu**

```swift

func cipher(messageToCipher: String, shift: Int) -> String {
    var ciphredMessage = ""

    for char in messageToCipher.unicodeScalars {
        var unicode = Int(char.value)

        if unicode > 64 && unicode < 123 {
            var modifiedShift = shift
            if unicode >= 65 && unicode <= 90 {
                while unicode + modifiedShift > 90 {
                    modifiedShift -= 26
                }
            } else if unicode >= 97 && unicode <= 122 {
                while unicode + modifiedShift > 122 {

                    modifiedShift -= 26
                }
            } else {
                modifiedShift = 0
            }

            unicode += modifiedShift
        }

        ciphredMessage += String(UnicodeScalar(unicode)!)
    }

    return(ciphredMessage)
}

var word = readLine()!
print(cipher(messageToCipher: word, shift: 3))

```

**Zdjęcie nr 4 przedstawia algorytm w języku Swift**

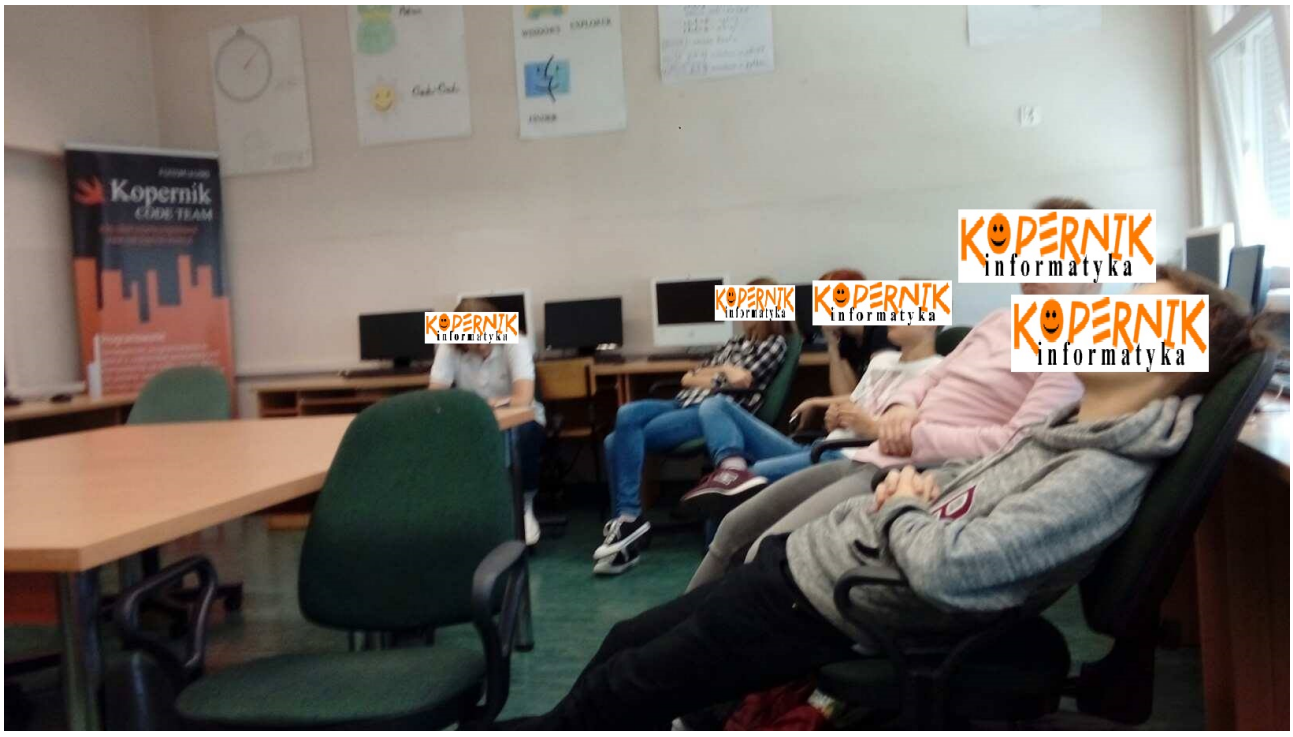
```java
import java.util.Scanner;

public class cezar{
        public static String ciphre(String text){
                char[] x = text.toCharArray();
                for(int i=0; i!=x.length; i++){
                        int n = x[i];
                        n+=3;
                        if((char)n>'Z'){n -= 26;}
                        x[i] = (char)n;
                }
                return new String(x);
        }
        public static void main(String[] args){
                System.out.println("Type the text to ciphre");
                String word;
                Scanner read = new Scanner(System.in);
                word = read.nextLine();
                String upperWord = word.toUpperCase();
                String result = ciphre(upperWord);
                System.out.println(result);
        }
}

```

**Zdjęcie nr 5 przedstawia algorytm w języku Java**

```cpp
#include<iostream>
#include<algorithm>
using namespace std;

string ciphre(string text, int key){
        if(key>26){key = key%26;}

        for(int i=0; i<text.length(); i++){
                text[i] += key;
                if(text[i]>'Z'){text[i] -= 26;}
        }
        return text;
}

int main(){
        string text;
        cout<<"Type the text to ciphre"<<endl;
        cin>>text;
        transform(text.begin(), text.end(), text.begin(), ::toupper);
        const string newText = ciphre(text, 3);
        cout<<newText<<endl;
}
```

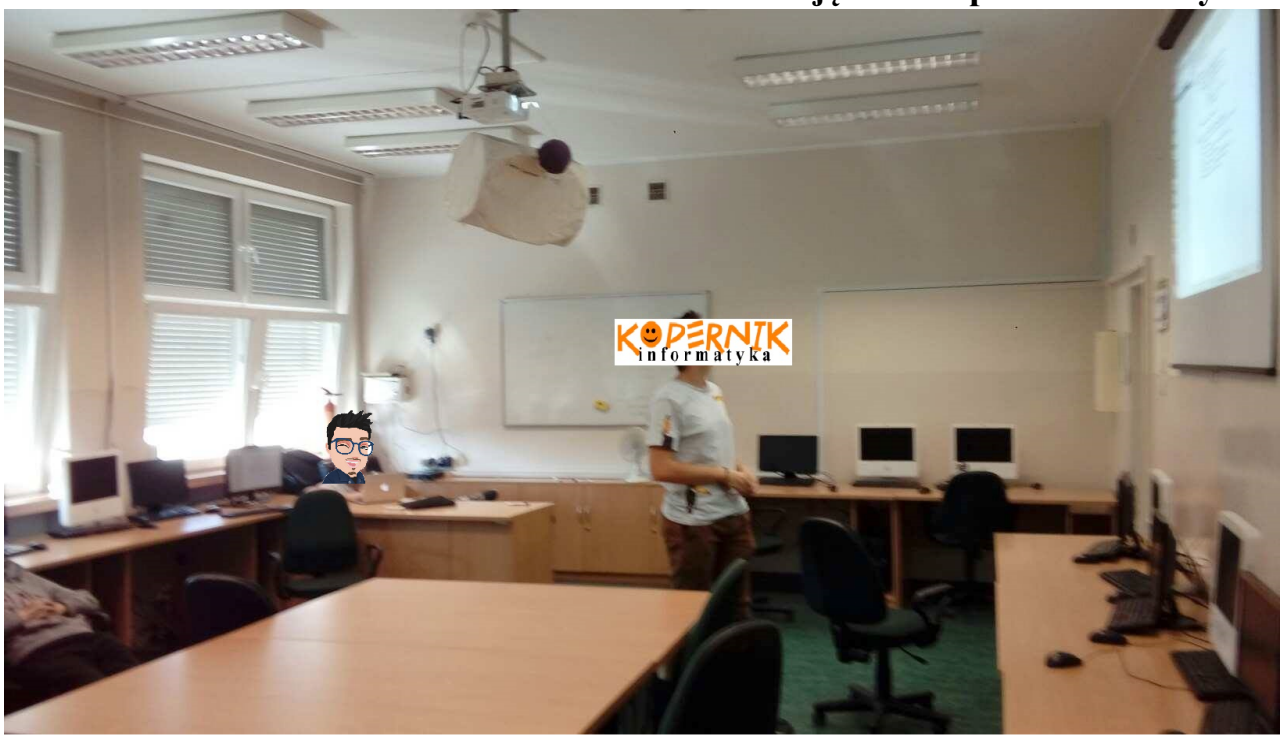
**Zdjęcie nr 6 przedstawia algorytm w języku C++**

Po przeprowadzonym wykładzie podjęliśmy decyzję, by przetestować nową wiedzę kolegów i koleżanek, więc poddaliśmy im krótkiemu testowi, który miał właśnie takie zadanie. Poniżej przedstawimy zdjęcia z wykładu oraz wyniki testu.




**Zdjęcie nr 7 przedstawia wykład**

**Zdjęcie nr 8 przedstawia wykład**



**Zdjęcie nr 9 przedstawia wykład**

| | Pytanie1 | Pytanie2 | Pytanie3 | Pytanie4 | Pytanie5 | Pytanie6 | Suma punktów | Maksymalne punkty | Wynik procentowy |
|---|---|---|---|---|---|---|---|---|---|
| U1 | 0 | 1 | 1 | 1 | 1 | 1 | 5 | 6 | 83,33% |
| U2 | 0 | 1 | 1 | 1 | 1 | 1 | 5 | 6 | 83,33% |
| U3 | 1 | 0 | 1 | 1 | 1 | 1 | 5 | 6 | 83,33% |
| U4 | 0 | 1 | 1 | 1 | 1 | 1 | 5 | 6 | 83,33% |
| U5 | 1 | 1 | 1 | 1 | 1 | 1 | 6 | 6 | 100,00% |
| U6 | 1 | 1 | 1 | 1 | 1 | 0 | 5 | 6 | 83,33% |
| Wynik procentowy klasy | 86,11% | | | | | | | | |

**Tabela nr 10 przedstawia wyniki testu**

Podsumowując to doświadczenie, ciekawym przeżyciem było wcielenie się w rolę nauczyciela. Sposób myślenia potrzebny do wytłumaczenia czegoś komuś, kto przedtem nie zagłębiał się ani trochę w omawiany temat, jest zupełnie inny od nauki samodzielnej i dużo bardziej wymagający. Z drugiej strony pozwala na lepsze zrozumienie zagadnienia oraz budowę drugiej perspektywy, która na pewno potrafi pomóc w utrwaleniu materiału. Nie można zapomnieć również o tezie Alberta Einsteina, której prawdę staramy się dowieść. Trzeba przyznać, że aby coś prosto wytłumaczyć, a konkretnie właśnie szyfr Cezara w postaci algorytmu, najpierw musieliśmy się mocniej zastanowić nad jego działaniem i implementacją. W tym sensie teza Alberta Einsteina jest jak najbardziej prawdziwa. Z drugiej strony należy przemyśleć to, czy klasa zrozumiała nasz wykład w pełni. Przeprowadzony przez nas test sugeruje, że cel ten został osiągnięty, gdyż wynik z niego nie spada poniżej 83%, co jest dość dobrym wynikiem. Oczywiście nie mierzy on pełnego zrozumienia tematu, a jedynie najważniejsze zagadnienia jak szyfrowanie danego słowa lub wiadomości ogólne o algorytmie i numerach ASCII, którymi kodowane są litery alfabetu w większości języków programowania. Pewną trudnością okazało się również to, że implementacja algorytmu potrafi się dramatycznie różnić w zależności od użytego języka. Różne języki programowania inaczej traktują litery alfabetu oraz mają różne funkcje, którymi można na nie działać. W ten sposób na przykład w Pythonie możemy mieć dwa razy krótszy program niż w Swifcie(w przybliżeniu). Różnią się też wydajności tych rozwiązań, ale te nie zostały przez nas przetestowane. Taka różnica mogła utrudnić przyswojenie tematu przez słuchaczy, którzy przedtem nie mieli z nim styczności. Niezaprzeczalnym jest jednak, że Albert Einstein miał rację, gdyż jeśli przyjmiemy, że materiał został zrozumiany, to oznacza, że wytłumaczyliśmy go w prosty sposób, a aby tego dokonać musieliśmy najpierw dogłębnie przyswoić sobie temat, gdyż bez tego nie dalibyśmy rady. A więc : *Jeśli nie potrafisz wytłumaczyć czegoś w prosty sposób, to znaczy, że tak naprawdę tego nie rozumiesz.*

## Źródła:

1. www.code.kopernik-leszno.pl

2. www.freecodecamp.org

## Spis zdjęć, rysunków, tabel i wykresów:



Zgadzamy się na udostępnienie naszego projektu na stronie www.code.kopernik-leszno.pl, aby kolejni uczniowie mogli skorzystać z tego materiały przy nauce programowania.