# Implementacja systemu T9

Matylda Taborek

Styczeń 2026

## Opis problemu i rozwiązania

### Problem

Celem projektu jest stworzenie systemu T9 (predictive text), stosowanego w telefonach komórkowych w celu ułatwienia pisania tekstów za pomocą klawiatury 3×4.

#### Opis działania słownika T9

Słownik T9 jest wykorzystywany podczas wpisywania tekstu w telefonie komórkowym. Metoda ta przyspiesza wprowadzanie tekstu na klawiaturze z ośmioma znaczącymi przyciskami. Działa ona w taki sposób, że aby wpisać słowo, należy jednokrotnie nacisnąć odpowiednie klawisze, na przykład:

Standardowo, aby wpisać słowo „cool", należy nacisnąć klawisze 12 razy (222 666 666 555), natomiast przy użyciu słownika T9 wystarczą tylko 4 naciśnięcia (2 6 6 5). Słownik automatycznie wyszukuje słowa odpowiadające danej kombinacji znaków i wyświetla je na liście, z której następnie można wybrać właściwe słowo, w tym przypadku „cool". Jeżeli jednak dane słowo nie istnieje w słowniku, można je wpisać standardowym sposobem.

### Rozwiązanie:

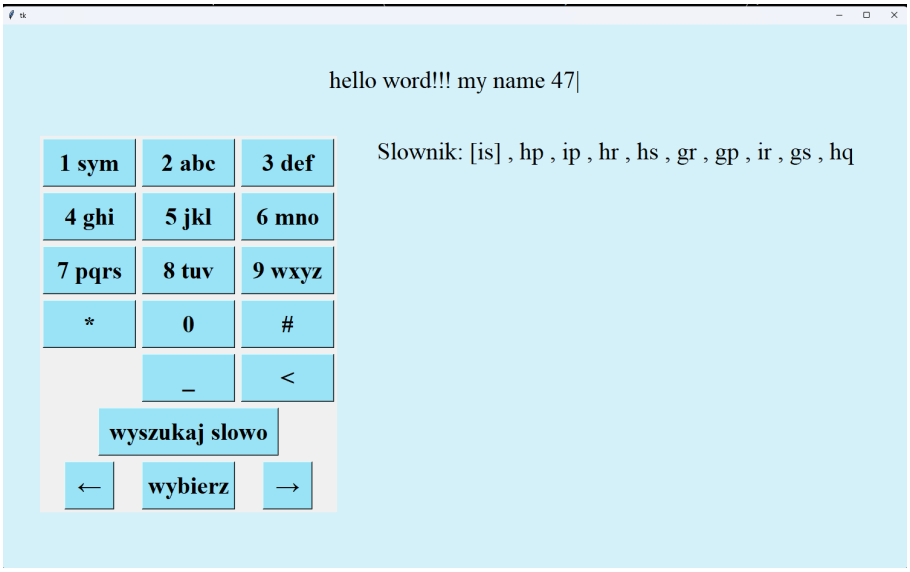
Program tworzy interaktywny interfejs w bibliotece Tkinter, który symuluje klawiaturę T9 i umożliwia wprowadzanie słów za pomocą przycisków numerycznych.

#### Opis skompilowanego programu

Po skompilowaniu programu na ekranie można zauważyć:

- Pole tekstowe znajdujące się w górnej części ekranu.
- Klawiaturę (po prawej stronie), składającą się z:
  - Przycisk (1 sym) – wpisuje cyfrę 1 oraz odpowiada za symbole: . , ? ! ' _ @ # $ % & - "
  - Przycisk (2 abc) – wpisuje cyfrę 2 i odpowiada za litery: a b c
  - Przycisk (3 def) – wpisuje cyfrę 3 i odpowiada za litery: d e f
  - Przycisk (4 ghi) – wpisuje cyfrę 4 i odpowiada za litery: g h i

  - Przycisk (5 jkl) – wpisuje cyfrę 5 i odpowiada za litery: j k l
  - Przycisk (6 mno) – wpisuje cyfrę 6 i odpowiada za litery: m n o
  - Przycisk (7 pqrs) – wpisuje cyfrę 7 i odpowiada za litery: p q r s
  - Przycisk (8 tuv) – wpisuje cyfrę 8 i odpowiada za litery: t u v
  - Przycisk (9 wxyz) – wpisuje cyfrę 9 i odpowiada za litery: w x y z
  - Przycisk (*) – wpisuje *
  - Przycisk (#) – wpisuje #
  - Przycisk (_) – wprowadza spację
  - Przycisk (0) – wpisuje cyfrę 0
  - Przycisk (<) – usuwa znak znajdujący się przed kursorem (backspace)
  - Przycisk (wyszukaj słowo) – po jego naciśnięciu w słowniku po prawej stronie ekranu wyświetlane są wyrazy, które można utworzyć na podstawie wpisanych cyfr
  - Przycisk (←) – przesuwa zaznaczenie w lewo po liście wyrazów w słowniku
  - Przycisk (wybierz) – wybiera zaznaczony wyraz z listy słownika i zastępuje nim ciąg cyfr
  - Przycisk (→) – przesuwa zaznaczenie w prawo po liście wyrazów w słowniku
- Słownik po prawej stronie ekranu, który po naciśnięciu przycisku (wyszukaj słowo) wyświetla słowa odpowiadające wprowadzonym cyfrom.




Rysunek 1: Program po skompilowaniu

### Opis zawartości projektu

Projekt składa się z trzech plików:

- glowny.py – główny plik zawierający wszystkie funkcje oraz tworzący interfejs użytkownika.

- slownik_plik.txt – plik pobrany z linku: (https://apiacoa.org/publications/teaching/datasets/google-10000-english.txt?utm) zawierający słownik 10 013 wyrazów i symboli w języku angielskim.
- slownik.py – plik zawierający:
  - mapę T9, w której cyfry są kluczami, a wartościami to ciągi znaków możliwe do wprowadzenia poprzez naciśnięcie danego klawisza,
  - definicję ścieżke do pliku `slownik_plik.txt`, otwiera go oraz wczytuje słowa do listy `slownik_words`.

### Opis działania programu

Program działa w następujący sposób: najpierw użytkownik buduje słowo, wpisując cyfry na wirtualnej klawiaturze T9 wyświetlanej w oknie aplikacji. Każde naciśnięcie przycisku dodaje cyfrę do aktualnie budowanego słowa (zmienna `current_word`), które jest na bieżąco wyświetlane na ekranie. Po wprowadzeniu pełnej sekwencji cyfr użytkownik może użyć przycisku (wyszukaj słowo), co powoduje wywołanie funkcji `find`, przeszukującej wcześniej wczytany słownik i zwracającej wszystkie słowa pasujące do danej sekwencji.

Następnie program wyświetla listę dopasowanych słów, podświetlając aktualnie zaznaczoną pozycję. Użytkownik może poruszać się po liście za pomocą przycisków (←) i (→) oraz zatwierdzić wybrane słowo przyciskiem (wybierz), co skutkuje dodaniem go do tekstu (tablicy) oraz wyzerowaniem zmiennej `current_word`. Dodatkowo obsługiwane są funkcje usuwania ostatniego znaku (backspace) oraz wstawiania spacji, a cały interfejs graficzny jest dynamicznie aktualizowany w trakcie działania programu.

### Opisy głównych funkcji

- `find(sequence)`

```python
def find(sequence):
    result = []
    for word in sl.slownik_words:
        if len(word) != len(sequence):
            continue

        podobny = True
        for num, litera in zip(sequence, word):
            if num not in sl.t9 or litera not in sl.t9[num]:
                podobny = False
                break

        if podobny:
            result.append(word)

    return result
```

Funkcja przyjmuje argument `sequence`.
Na początku tworzy pustą listę `result` do przechowywania dopasowanych słów. Następnie w pętli przechodzi przez każde słowo w wczytanym słowniku i wykonuje:

- Sprawdza, czy długość słowa ze słownika jest taka sama jak długość `sequence`. Jeśli nie, przechodzi do następnego słowa.
- Tworzy wewnętrzną pętlę, która iteruje przez każdą cyfrę z `sequence` i odpowiadającą jej literę ze słowa. Sprawdza, czy cyfra pasuje do litery zgodnie z mapą T9. Jeśli warunek nie jest spełniony, zmienna `podobny` ustawiana jest na `False` i pętla zostaje przerwana.
- Jeśli po iteracji zmienna `podobny` jest `True`, słowo zostaje dodane do listy `result`.

Funkcja zwraca listę `result`.

- `refresh()`

```
def refresh():
    if cursor_visible:
        text.config(text="".join(words + [current_word]) + "|")
    else:
        text.config(text="".join(words + [current_word]) + " ")
```

Funkcja nie przyjmuje argumentów.
Funkcja sprawdza stan zmiennej `cursor_visible`:

- Jeśli `cursor_visible == True`: aktualizuje tekst, wypisując listę `words` wraz z `current_word`, a na końcu dodaje znak | imitujący kursor.
- Jeśli `cursor_visible == False`: aktualizuje tekst w ten sam sposób, ale na końcu dodaje spację zamiast |, aby tekst się nie przesuwał.

Funkcja nie zwraca wartości.

- `update_wybrane_slowo()`

```
def update_wybrane_slowo():
    global wypisz_slownik
    if not wyszukane_slowa:
        wypisz_slownik = "Slownik: Brak slow"
    else:
        tab = []
        for i, word in enumerate(wyszukane_slowa):
            if i == pointer:
                tab.append(f"[{word}]")
            else:
                tab.append(word)
        wypisz_slownik = "Slownik: " + " , ".join(tab)

    wypisny_slownik.config(text=wypisz_slownik)
```

Funkcja nie przyjmuje argumentów.
Najpierw sprawdza, czy lista `wyszukane_slowa` jest pusta. Jeśli tak, zmienna `wypisz_slownik`

przyjmuje wartość "Słownik: Brak słów". Jeśli lista nie jest pusta, funkcja iteruje po słowach, dodając je do listy `tab`. Słowo wskazywane przez `pointer` zostaje umieszczone w nawiasach kwadratowych. Następnie zmienna `wypisz_slownik` jest aktualizowana i wyświetlana w interfejsie.
Funkcja nie zwraca wartości.

- `wpisz(x)`

```
def wpisz(x):
    global current_word
    current_word += x
    refresh()
```

Funkcja przyjmuje argument `x`.
Dodaje `x` na końcu zmiennej `current_word` i wywołuje funkcję `refresh()`.
Funkcja nie zwraca wartości.

- `spacja()`

```
def spacja():
    global current_word
    words.append(current_word + " ")
    current_word = ""
    refresh()
```

Funkcja nie przyjmuje argumentów.
Dodaje `current_word` wraz ze spacją do listy `words`, zeruje `current_word` i wywołuje `refresh()`.
Funkcja nie zwraca wartości.

- `backspace()`

```
def backspace():
    global current_word

    if current_word:
        current_word = current_word[:-1]
    elif words:
        current_word = words.pop()
        current_word = current_word[:-1]
    refresh()
```

Funkcja nie przyjmuje argumentów.
Jeśli `current_word` nie jest puste, usuwa ostatni znak. Jeśli jest puste, sprawdza listę `words`: jeśli nie jest pusta, ostatni element jest przenoszony do `current_word` i z niego usuwany jest ostatni znak. Na końcu wywołuje `refresh()`.
Funkcja nie zwraca wartości.

- `left()`

```
def left():
    global pointer
    if not wyszukane_slowa:
        return

    if pointer == 0:
        pointer = len(wyszukane_slowa) - 1
    else:
        pointer = pointer-1
    update_wybrane_slowo()
```

Funkcja nie przyjmuje argumentów.
Jeśli lista `wyszukane_slowa` jest pusta, funkcja kończy działanie. W przeciwnym wypadku zmniejsza `pointer` o 1 lub, jeśli `pointer == 0`, ustawia go na długość listy minus 1. Wywołuje funkcję `update_wybrane_slowo()`.
Funkcja nie zwraca wartości.

- `wybierz()`

```
def wybierz():
    global current_word, wyszukane_slowa
    if not wyszukane_slowa:
        return

    current_word = wyszukane_slowa[pointer]
    words.append(current_word)
    current_word = ""
    wyszukane_slowa = []
    refresh()
    update_wybrane_slowo()
```

Funkcja nie przyjmuje argumentów.
Jeśli lista `wyszukane_slowa` jest pusta, funkcja kończy działanie. W przeciwnym wypadku ustawia `current_word` na wyraz wskazywany przez `pointer`, dodaje go do listy `words`, zeruje `current_word` i listę `wyszukane_slowa`, a następnie wywołuje funkcje `update_wybrane_slowo()` i `refresh()`.
Funkcja nie zwraca wartości.

- `right()`

```python
def right():
    global pointer
    if not wyszukane_slowa:
        return

    if pointer == len(wyszukane_slowa)-1:
        pointer = 0
    else:
        pointer = pointer+1
    update_wybrane_slowo()
```

Funkcja nie przyjmuje argumentów.
Jeśli lista `wyszukane_slowa` jest pusta, funkcja kończy działanie. W przeciwnym wypadku zwiększa `pointer` o 1 lub, jeśli osiągnie ostatni indeks listy, ustawia go na 0. Wywołuje funkcję `update_wybrane_slowo()`.
Funkcja nie zwraca wartości.

- `predict()`

```python
def predict():
    global wypisz_slownik, wyszukane_slowa, pointer
    wyszukane_slowa = find(current_word)
    pointer = 0
    update_wybrane_slowo()
```

Funkcja nie przyjmuje argumentów.
Wywołuje funkcję `find` z argumentem `current_word` i zapisuje wynik do listy `wyszukane_slowa`. Ustawia `pointer` na 0 i wywołuje funkcję `update_wybrane_slowo()`.
Funkcja nie zwraca wartości.

- `blink_cursor()`

```python
def blink_cursor():
    global cursor_visible
    cursor_visible = not cursor_visible
    refresh()
    root.after(500, blink_cursor)
```

Funkcja nie przyjmuje argumentów.
Zmienia wartość zmiennej `cursor_visible` na przeciwną i wywołuje funkcję `refresh()`.
Następnie planuje kolejne wywołanie samej siebie po 500 ms.
Funkcja nie zwraca wartości.

### Dodatkowe informacje

W programie tworzone są dwa kontenery: `root` i `frame`.

- `root` – główne okno programu, reprezentuje całą aplikację: jej rozmiar, tło, układ oraz cykl życia.
- `frame` – kontener wewnątrz `root`, w którym znajdują się przyciski klawiatury rozmieszczone na siatce.

# Dokumentacja użytkowa

## Jak uruchomić program:

1. Pobierz wszystkie pliki i umieść je w jednym folderze.
2. Uruchom program, wpisując w terminalu lub wierszu poleceń:

```
py glowny.py
```

## Jak wprowadzać dane:

Dane wprowadza się za pomocą klawiatury pokazanej na ekranie.

## Dostępne opcje programu:

1. Wpisywanie cyfr za pomocą przycisków numerowanych od 1 do 9 oraz 0.
2. Wstawianie spacji między słowami.
3. Usuwanie ostatniego znaku za pomocą przycisku `<`.
4. Wyświetlanie listy dostępnych słów.
5. Wybieranie słów z listy po prawej stronie za pomocą strzałek i przycisku `wybierz`.
6. Zmiana ciągu cyfr na słowo: należy nacisnąć przycisk `wyszukaj słowo` i wybrać odpowiednie słowo z listy po prawej za pomocą strzałek i przycisku `wybierz`.
7. Wstawianie znaku specjalnego: należy wpisać `1`, nacisnąć przycisk `wyszukaj słowo` i wybrać odpowiedni znak specjalny z listy po prawej za pomocą strzałek i przycisku `wybierz`.
8. Zakończenie programu.

# Źródła


- https://pl.wikipedia.org/wiki/S%C5%82ownik_T9
- https://en.wikipedia.org/wiki/T9_(predictive_text)
- https://apiacoa.org/publications/teaching/datasets/google-10000-english.txt?utm