Table of Contents

1. Character numbering 2
2. Directory structure 5
3. File and directory structure diagram 7
4. Form code 8
5. Character database 9
6. Graphical file containing a single char 9
7. ocr_files direction and data import description 10
8. Description decode_binarized.py file 11
9. The disk content 12

1. Character numbering

Each of the below characters appears min. three times in the PHSF:
a. Digits: 0-9
b. Lowercase letters of the Latin alphabet: a-z
c. Uppercase letters of the Latin alphabet:: A-Z
d. Lowercase letters of the Polish alphabet: ą, ć, ę, ł, ń, ó, ś, ź, ż
e. Uppercase letters of the Polish alphabet: Ą, Ć, Ę, Ł, Ń, Ó, Ź, Ż
f. Special characters: + - : ; $! ? @ .

The presented table below contains the code which has been assigned to each character:
a. Digits: 0-9

Number	Character
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9

b. Lowercase letters of the Latin alphabet: a-z

Number	Character
10	a
11	b
12	c
13	d
14	e
15	f
16	g
17	h
18	i
19	j
20	k
21	l
22	m
23	n
24	o
25	p
26	q
27	r
28	s
29	t
30	u
31	v

32	w
33	x
34	y
35	z

c. Uppercase letters of the Latin alphabet: A-Z

Number	Character
36	A
37	B
38	C
39	D
40	E
41	F
42	G
43	H
44	I
45	J
46	K
47	L
48	M
49	N
50	O
51	P
52	Q
53	R
54	S
55	T
56	U
57	V
58	W
59	X
60	Y
61	Z

d. Lowercase letters of the Polish alphabet: ą, ć, ę, ł, ń, ó, ś, ź, ż

Number	Character
62	ą
63	ć
64	ę
65	ł
66	ń
67	ó
68	ś
69	ź
70	ż

e. Uppercase letters of the Polish alphabet: Ą, Ć, Ę, Ł, Ń, Ó, Ź, Ż

Number	Character
71	Ą
72	Ć
73	Ę
74	Ł
75	Ń
76	Ó
77	Ś
78	Ź
79	Ż

f. Special characters: + - : ; $! ? @ .

Number	Character
80	+
81	-
82	:
83	;
84	$
85	!
86	?
87	@
88	.

2. Directory structure

phsf - the name of the main directory where two directories are located:
- directory *znaki* containing directories, in which the characters are located
- directory *inwokacja*, containing scanned fragments of text from the field in which the respondent rewrote the content of the poem (invocation).

1) Directory description: *phsf/znaki*

The *znaki* directory contains the *png* directory which contains directories entitled with numbers of individual characters where the files are stored in *png* format.

phsf/znaki/png/directory_number
Directory names are in the form of numbers that have been assigned to individual characters (according to the tables in item 1).

Example:
phsf/znaki/png/83 – directory containing the samples of '!' character in png format

Files placed in the *directory_number* directory have the following format:
CharNumber_FileNumber_BirthYear_Sex_FormCode.extension

CharNumber – the numer of the character corresponding to the specific character in the accordance with the directory in which it is located and with tables above (see Chapter 1).

FileNumber – the file order number in the given directory. File numbering starts from 0000 and the maximal number is 9999.

BirthYear – the last two digits of the year of birth (from 00 to 99) of the person completed the form.

Sex – the gender of the person completed the form in the format: K or M

FormCode – the information on belonging the person having filled the form to a defined group, it is organized in the following format: *NumberLetter*

extension – the file extension: '*.png'*.

Examples:
phsf/znaki/png/47/47_0000_94_K_1A.png – file number: 0000 with char code: 47 (character: M), filled by the person born in 1994, woman, belonging to 1A group. The file format is png.

phsf/znaki/png/47/47_0104_56_M_3A.png – file number: 0104, the character code: 47 (character: M), filled by the person born in 1956, man, belonging to 3A group. The file format is png.

2) Directory description: *phsf/inwokacja*

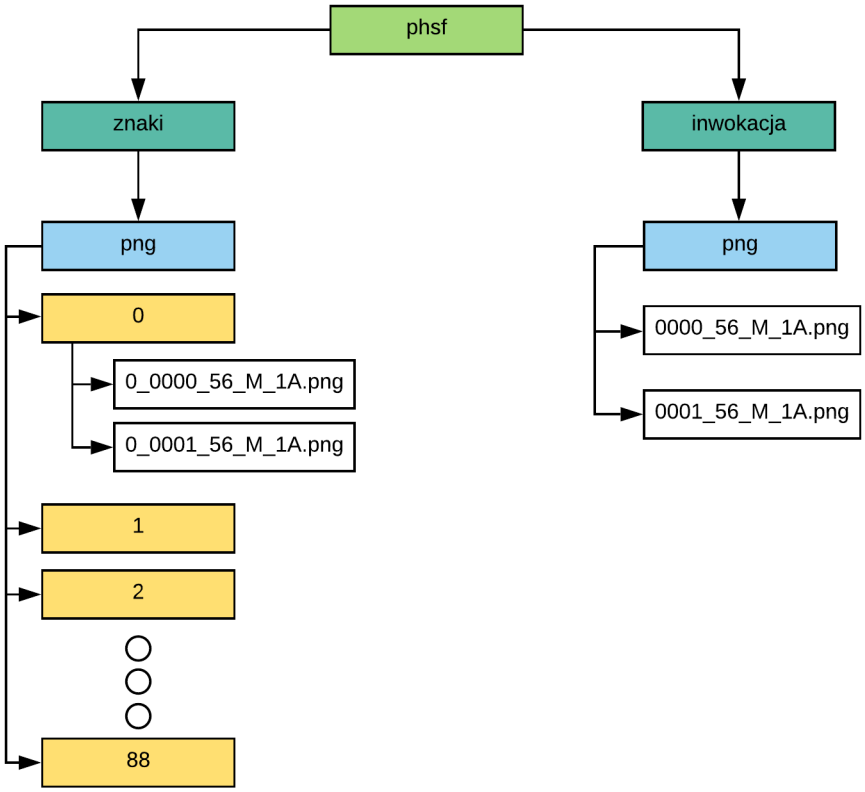
It contains the *png* directory which contains the .png files with the scanned fragments of text from the field in which the respondents rewrote the content of the poem(invocation).

Example:
phsf/inwokacja/png – directory containing files with invocations int the *png* format.

The names of the files located in the *png* directory have the following format:
FileNumber_BirthYear_Sex_FormCode.extension

3. File and directory structure diagram

Directories are marked with color. The diagram presents the structure described in the Chapter 2.



4. Form code

A form code is corresponding to the professional/social group of the person filling in the form. It has the following format: *Number-Letter*.

Letter – a capital letter of latin alphabet, beginning from ‘A’, according to ASCII.
Number – integer numer starting from 1.

Example:
1-A

The below table contains the groups and subgroups:

Numer grupy	Grupa	Numer podgrupy	Podgrupa
1	University employees	A	Others
		B	Institute of Computer Science
		C	Electrical Engineering, other than Computer Science
		D	Management Faculty
2	Students	A	others
		B	Computer Science
		C	Electrical Engineering
		D	Management
		E	Economics
		F	Mechanical Engineering Faculty
		G	Architecture and Civil Engineering Faculty
3	Family/relatives	A	Others

Examples:
1-B – the form was filled by the University employee working at the Computer Science Institute
2-B– the form was filled by a student of Computer Science
3-A- the form was filled by family members or relatives

5. Character database

The created database contains at leas 6000 entries of every character listed in the Chapter 1. A code has been assigned to every char, as it was described in the Chapter 1.

Every char is stored in *.png* format. The Chapter 2 contains the file and directory structure description.

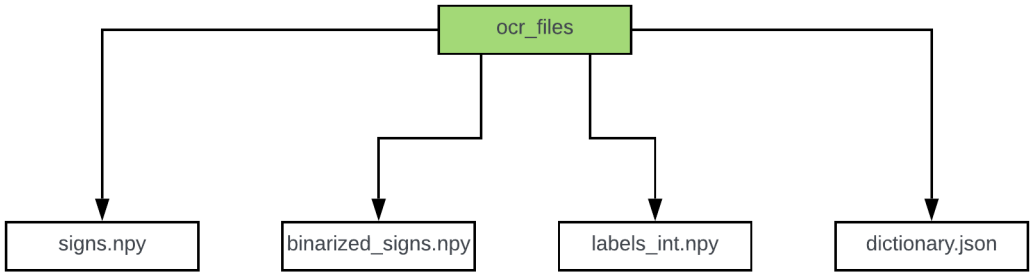
Additionally, the *ocr_files* directory was created. It contains four files containing all the gathered data. Detailed description of the files is presented in the Chapter 7.

6. Graphical file containing a single char

The size of the stored char is 20px (width) by 32 px (height). The complete image size is 32 by 32 px. The stored character is centered.

7. ocr_files direction and data import description

Below the ocr_files directory structure is presented.



The data are stored with the use of *array* data structure of Python language *numpy* library. Four files were generated. The files contain the following data:

1. **"signs.npy":**
   - scaled to 32x32 px images having not undergone binarization process
   - the images are stored as 2D arrays of uint8 type filled with 0 or 255.
   - the arrays are collected into one 3D array having the following dimensions: (image_number x 32 x 32)

2. **"binarized_signs.npy":**
   - scaled to 32x32 px images having undergone binarization process.
   - binarization was performer in the following way: a 32x32 px image was transformed into 128 byte array (in total 32x32 = 128x8 bit). Every byte of the array contained the values of 8 pixels: if a pixel was black: the bit was low and, correspondingly: when the pixel was white, the bit was high.
   - the binarized images were collected into 2D array of uint8 data type and of image_number x 128 size.
   - Every row of the array contains an image after binarization process.

3. **"labels_int.npy":**
   - class labels stored in the uint8 data type arrays. The labels are char codes described in the chapters 1 and 2 (0-88).
   - the array size is (image_number x 1).
   - *i*-th element of the class label array corresponds to the *i*-th elements of *signs.npy* and *binarized_signs.npy* files.

4. **"dictionary.json":**
   - the dictionary of the character codes assigned to the characters according to the Chapter 1.

Data import from the files is carried out with the use of *load* function of *numpy* library. *load_files* function is used for importing the complete set of the files.

load_dictionary function is used in order to read the data stored in the *dictionary.json* file.
The *load_files* function returns the values read from the files.
The *load_dictionary* function requires the file path to be passed as an argument. The path to the directory where the files are stored has to be passed to *load_files* function.
Below the listing containing implementations of *load_dictionary* and *load_files* functions.
The mentioned listing is saved in the *load_files.py* file which is attached to the files written on the disk.

```
# -*- coding: utf-8 -*-

import numpy as np
import json

def load_dictionary(path):
    with open(path, 'r') as JSON:
        sign_dict = json.load(JSON)
    sign_dict = {int(number):sign for number, sign in sign_dict.items()}
    return sign_dict

def load_files(load_path):
    signs = np.load(load_path + "\\" + "signs.npy")
    binarized_signs = np.load(load_path + "\\" + "binarized_signs.npy")
    labels = np.load(load_path + "\\" + "labels_int.npy")
    signs_dictionary = load_dictionary(load_path + "\\" + "dictionary.json")
    return signs, binarized_signs, labels, signs_dictionary
```

8. Description decode_binarized.py file

Below the listing of *decode_image_from_binarized* function, responsible for decoding a single character stored in a separate row of the array from the *binarized_signs.npy* file to a 2D array similar to an array stored in the *signs.npy* file.
The *decode_image_from_binarized* function requires a 1D array of 128x1 size.

```
# -*- coding: utf-8 -*-
import numpy as np

def decode_image_from_binarized(binarized, idle_pixel = 0):
    if (idle_pixel == 0):
        occupied_pixel = 255
    else:
        occupied_pixel = 0
    IMG_SIDE_LENGTH= 32
    bits_in_byte = 8
    img_arr = np.ones((IMG_SIDE_LENGTH*IMG_SIDE_LENGTH, 1))*idle_pixel
    for i, byte in enumerate(binarized):
        for j in range(bits_in_byte):
            if (byte >= 2**(bits_in_byte - j - 1)):
                img_arr[i*bits_in_byte + j] = occupied_pixel
                byte -= 2**(bits_in_byte - j - 1)
            else:
                img_arr[i*bits_in_byte + j] = idle_pixel
    img_arr = img_arr.reshape((IMG_SIDE_LENGTH, IMG_SIDE_LENGTH))
    return img_arr
```

9. The disk content

The disk contains:
- phs folder f
- ocr_files folder
- documentation.pdf file
- load_files.py Python source file
- decode_binarized.py Python source file