Maciej Byczko Bartosz Matysiak	Prowadzący: dr inż. Jacek Mazurkiewicz	Numer ćwiczenia 2
PN 10:50 TP	Temat ćwiczenia: Układy Kombinacyjne	Ocena:
Grupa: B	Data wykonania: 10 Października 2021	

Spis treści

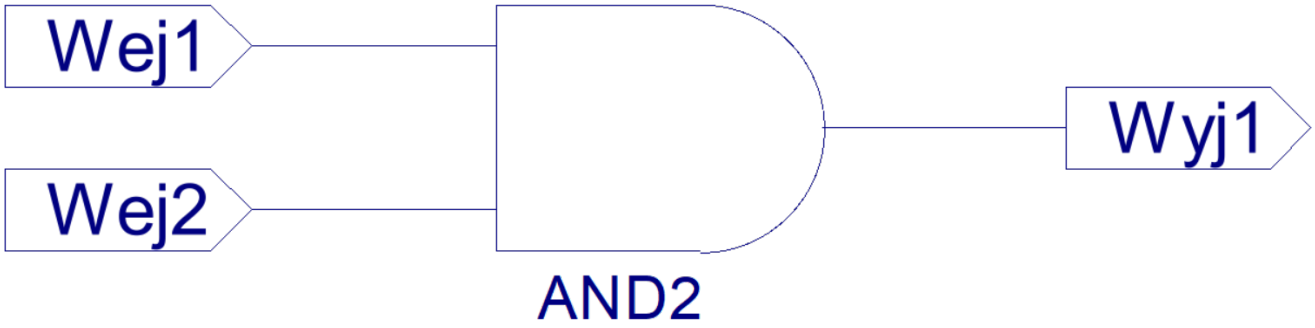
1 Zadanie 1

1.1 Polecenie

Wykonać dowolną bramkę - funktor: 2 wejścia, 1 wyjście

1.2 Rozwiązanie

1.2.1 Schemat układu

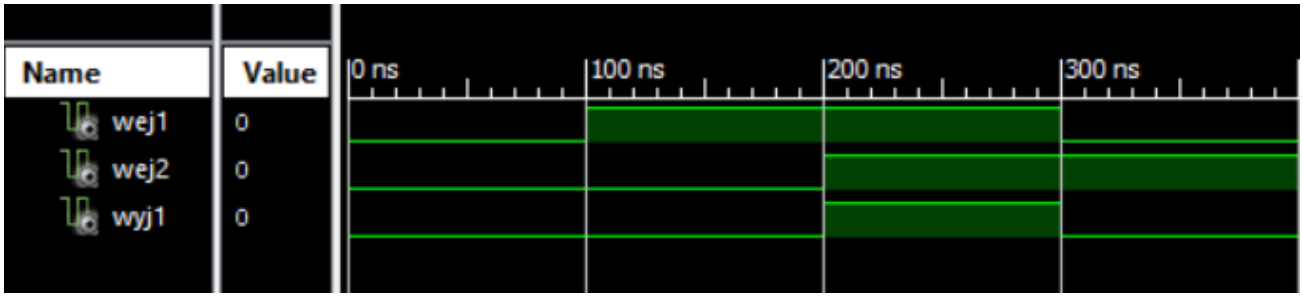


1.2.2 Kod VHDL

```
1  LIBRARY ieee;
2  USE ieee.std_logic_1164.ALL;
3  USE ieee.numeric_std.ALL;
4  LIBRARY UNISIM;
5  USE UNISIM.Vcomponents.ALL;
6  ENTITY schematic_zad1_schematic_zad1_sch_tb IS
7  END schematic_zad1_schematic_zad1_sch_tb;
8  ARCHITECTURE behavioral OF schematic_zad1_schematic_zad1_sch_tb IS
9
10    COMPONENT schematic_zad1
11        PORT (
12            Wej1 : IN STD_LOGIC;
13            Wej2 : IN STD_LOGIC;
14            Wyj1 : OUT STD_LOGIC);
15    END COMPONENT;
16
17    SIGNAL Wej1 : STD_LOGIC;
18    SIGNAL Wej2 : STD_LOGIC;
19    SIGNAL Wyj1 : STD_LOGIC;
20
21 BEGIN
22
23    UUT : schematic_zad1 PORT MAP(
24        Wej1 => Wej1,
25        Wej2 => Wej2,
26        Wyj1 => Wyj1
27    );
28
29    Wej1 <= '0', '1' AFTER 100 ns, '0' AFTER 300 ns;
30    Wej2 <= '0', '1' AFTER 200 ns, '0' AFTER 400 ns;
```

```
31
32 END;
```

1.2.3 Symulacja



2 Zadanie 2

2.1 Polecenie

Implementacja funkcji logicznej $G(w,x,y,z) = \prod(0,2,3,4,6,7,9,11,12,13,15)$

2.2 Rozwiązanie

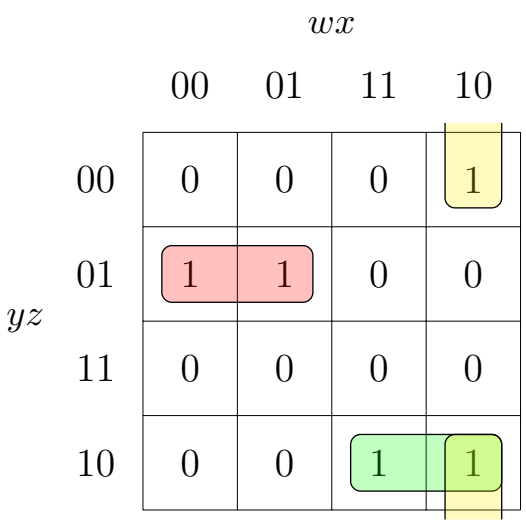
2.2.1 Wyprowadzenie

$$G(w,x,y,z) = \prod(0,2,3,4,6,7,9,11,12,13,15) \tag{1}$$
$$= \sum(1,5,8,10,14) = \sum(0001,0101,1000,1010,1110) \tag{2}$$
$$= \overline{w}\overline{x}yz + \overline{w}x\overline{y}z + w\overline{x}\overline{y}\overline{z} + w\overline{x}y\overline{z} + wxy\overline{z} \tag{3}$$
$$= \overline{w}\overline{y}z(\overline{x}+x) + w\overline{z}(\overline{x}\overline{y} + \overline{x}y + xy) \tag{4}$$
$$= \overline{w}\overline{y}z + w\overline{z}(\overline{x}(\overline{y}+y) + xy) \tag{5}$$
$$= \overline{w}\overline{y}z + w\overline{z}(\overline{x} + xy) \tag{6}$$
$$= \overline{w}\overline{y}z + w\overline{z}((\overline{x}+x)(\overline{x}+y)) \tag{7}$$
$$= \overline{w}\overline{y}z + w\overline{z}((\overline{x}+y)) \tag{8}$$
$$= \overline{w}\overline{y}z + w\overline{x}\overline{z} + w\overline{z}y \tag{9}$$

2.2.2 Tabela prawdy

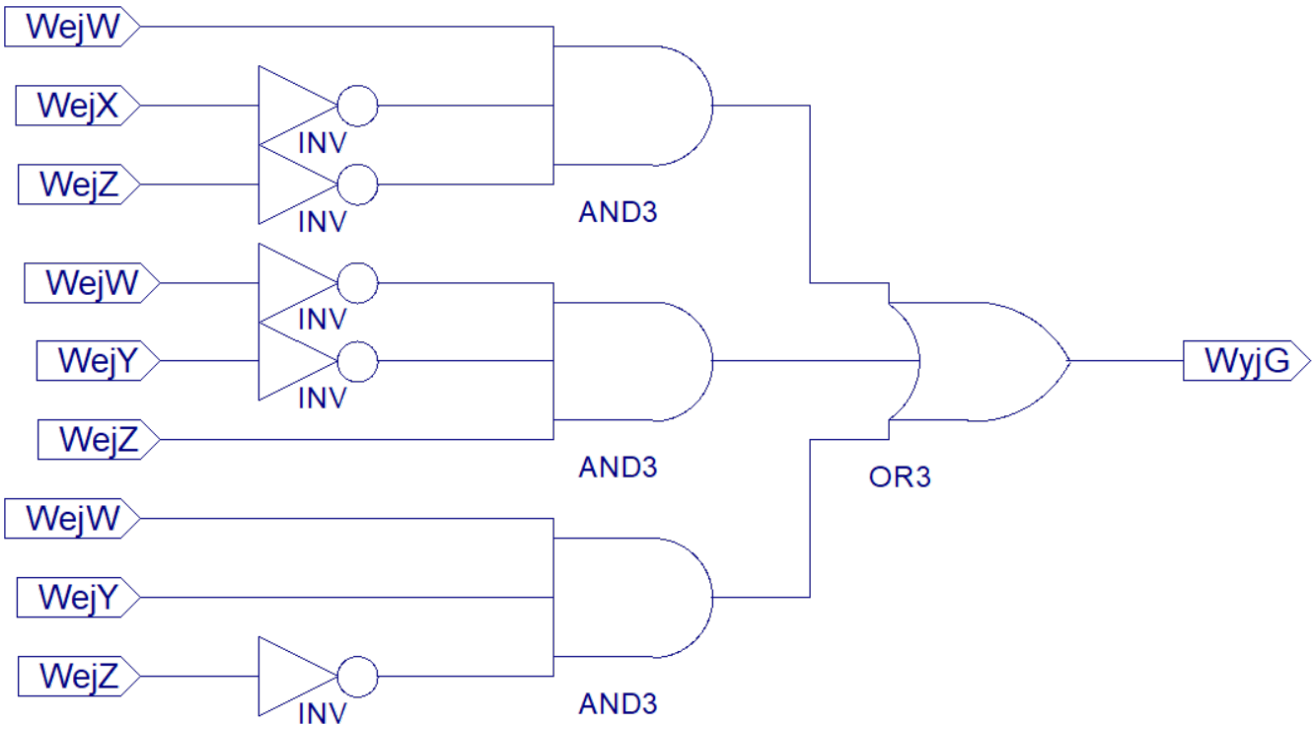
Kod dziesiętny	w	x	y	z	G
0	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	0
3	0	0	1	1	0
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	0
7	0	1	1	1	0
8	1	0	0	0	1
9	1	0	0	1	0
10	1	0	1	0	1
11	1	0	1	1	0
12	1	1	0	0	0
13	1	1	0	1	0
14	1	1	1	0	1
15	1	1	1	1	0

2.2.3 Siatka Karnaugh



Rysunek 1: $Wyj_G = w\overline{x}\overline{z} + \overline{w}\overline{y}z + wy\overline{z}$

2.2.4 Schemat układu

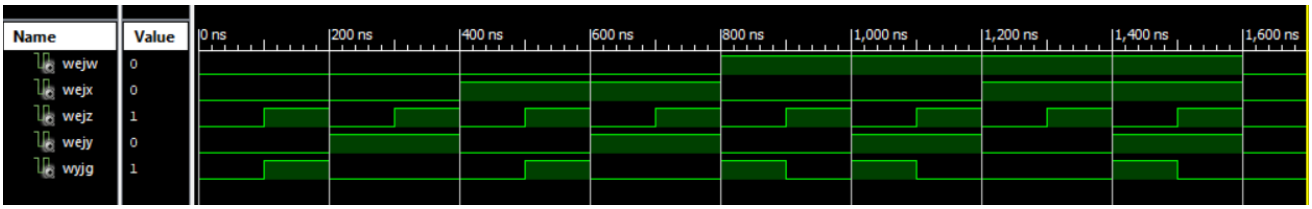


2.2.5 Kod VHDL

```
1  LIBRARY ieee;
2  USE ieee.std_logic_1164.ALL;
3  USE ieee.numeric_std.ALL;
4  LIBRARY UNISIM;
5  USE UNISIM.Vcomponents.ALL;
6  ENTITY schematic_zad2_schematic_zad2_sch_tb IS
7  END schematic_zad2_schematic_zad2_sch_tb;
8  ARCHITECTURE behavioral OF schematic_zad2_schematic_zad2_sch_tb IS
9
10     COMPONENT schematic_zad2
11         PORT (
12             WejW : IN STD_LOGIC;
13             WejX : IN STD_LOGIC;
14             WejZ : IN STD_LOGIC;
15             WejY : IN STD_LOGIC;
16             WyjG : OUT STD_LOGIC);
17     END COMPONENT;
18
19     SIGNAL WejW : STD_LOGIC := '0';
20     SIGNAL WejX : STD_LOGIC := '0';
21     SIGNAL WejZ : STD_LOGIC := '0';
22     SIGNAL WejY : STD_LOGIC := '0';
23     SIGNAL WyjG : STD_LOGIC := '0';
24
25 BEGIN
26
27     UUT : schematic_zad2 PORT MAP(
```

```
28          WejW => WejW,
29          WejX => WejX,
30          WejZ => WejZ,
31          WejY => WejY,
32          WyjG => WyjG
33      );
34
35      WejW <= NOT WejW AFTER 800ns;
36      WejX <= NOT WejX AFTER 400ns;
37      WejY <= NOT WejY AFTER 200ns;
38      WejZ <= NOT WejZ AFTER 100ns;
39  END;
```

2.2.6 Symulacja



3 Zadanie 3

3.1 Polecenie

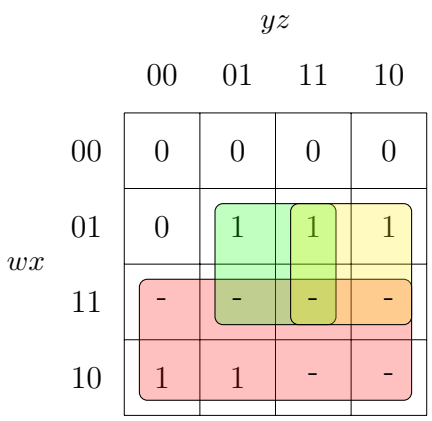
Implementacja układu translatora kodu **4-bit kod NKB na 4-bit kod Aikena**

3.2 Rozwiązanie

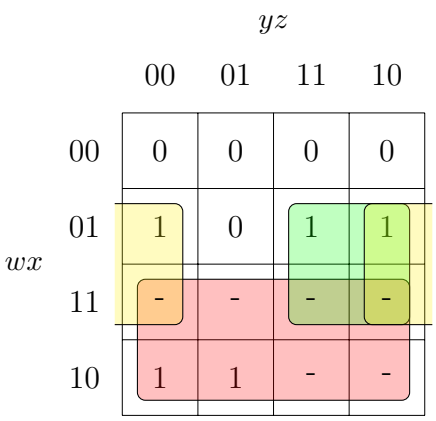
3.2.1 Tabela Prawdy

Kod dziesiętny	NKB				Kod Aikena			
	w	x	y	z	w	x	y	z
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	0
3	0	0	1	1	0	0	1	1
4	0	1	0	0	0	1	0	0
5	0	1	0	1	1	0	1	1
6	0	1	1	0	1	1	0	0
7	0	1	1	1	1	1	0	1
8	1	0	0	0	1	1	1	0
9	1	0	0	1	1	1	1	1
10	1	0	1	0	-	-	-	-
11	1	0	1	1	-	-	-	-
12	1	1	0	0	-	-	-	-
13	1	1	0	1	-	-	-	-
14	1	1	1	0	-	-	-	-
15	1	1	1	1	-	-	-	-

3.2.2 Siatki Karnaugh



$$w_A = xz + xy + w$$



$$x_A = x\overline{z} + xy + w$$

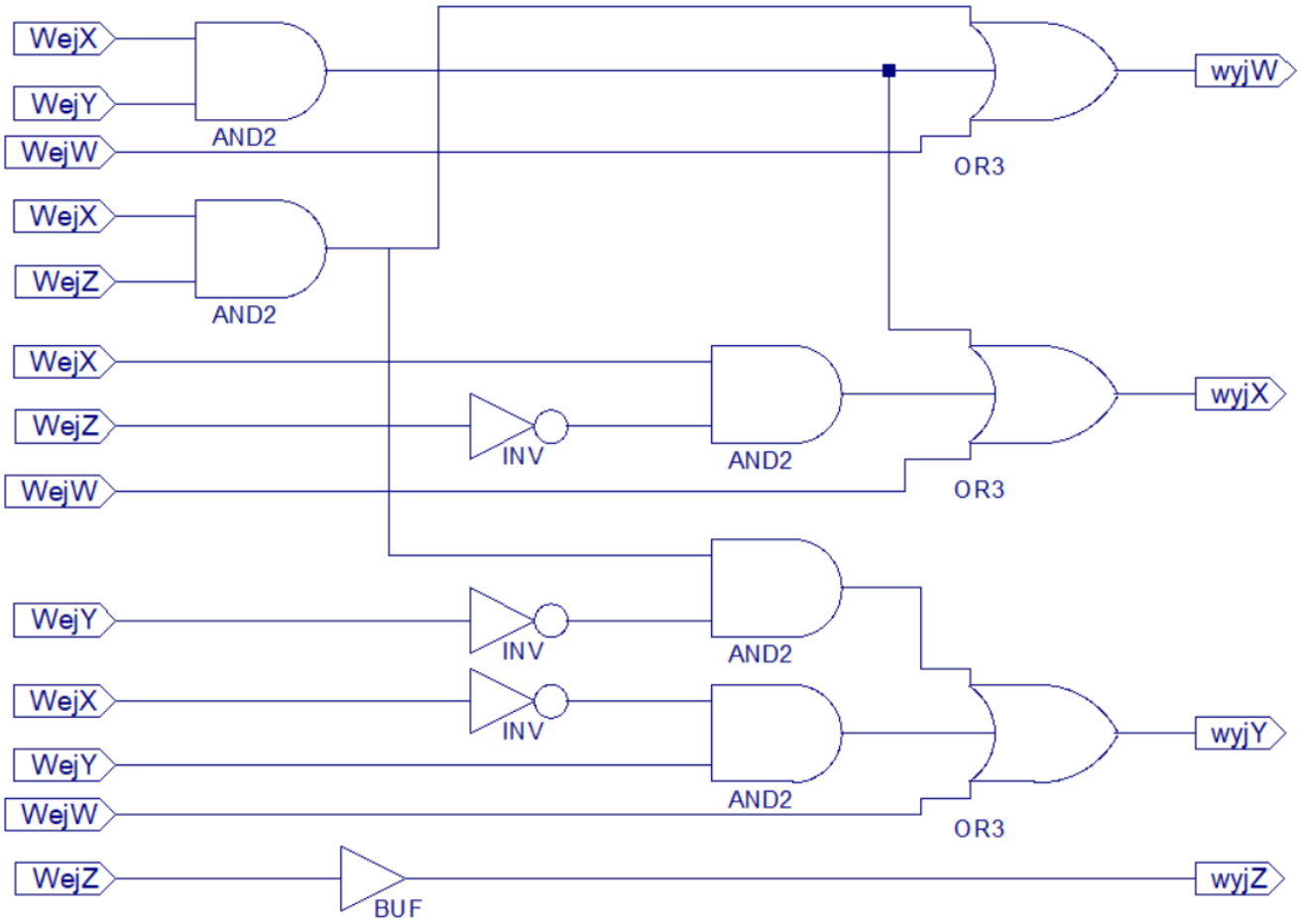
wx \ yz	00	01	11	10
00	0	0	1	1
01	0	1	0	0
11	-	-	-	-
10	1	1	-	-

$$y_A = \overline{x}y + x\overline{y}z + w$$

wx \ yz	00	01	11	10
00	0	1	1	0
01	0	1	1	0
11	-	-	-	-
10	0	1	-	-

$$z_A = z$$

3.2.3 Schemat układu

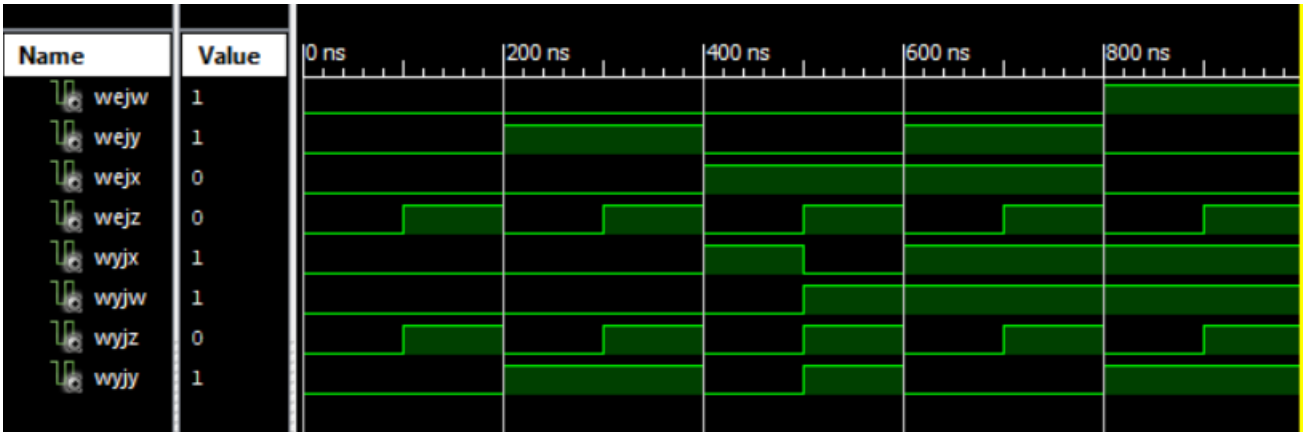


3.2.4 Kod VHDL

```
1 LIBRARY ieee;
2 USE ieee.std_logic_1164.ALL;
3 USE ieee.numeric_std.ALL;
4 LIBRARY UNISIM;
5 USE UNISIM.Vcomponents.ALL;
6 ENTITY schematic_zad3_schematic_zad3_sch_tb IS
7 END schematic_zad3_schematic_zad3_sch_tb;
```

```
8  ARCHITECTURE behavioral OF schematic_zad3_schematic_zad3_sch_tb IS
9
10     COMPONENT schematic_zad3
11         PORT (
12             WejW : IN STD_LOGIC;
13             WejY : IN STD_LOGIC;
14             WejX : IN STD_LOGIC;
15             WejZ : IN STD_LOGIC;
16             wyjX : OUT STD_LOGIC;
17             wyjW : OUT STD_LOGIC;
18             wyjZ : OUT STD_LOGIC;
19             wyjY : OUT STD_LOGIC);
20     END COMPONENT;
21
22     SIGNAL WejW : STD_LOGIC := '0';
23     SIGNAL WejY : STD_LOGIC := '0';
24     SIGNAL WejX : STD_LOGIC := '0';
25     SIGNAL WejZ : STD_LOGIC := '0';
26     SIGNAL wyjX : STD_LOGIC;
27     SIGNAL wyjW : STD_LOGIC;
28     SIGNAL wyjZ : STD_LOGIC;
29     SIGNAL wyjY : STD_LOGIC;
30
31 BEGIN
32
33     UUT : schematic_zad3 PORT MAP(
34         WejW => WejW,
35         WejY => WejY,
36         WejX => WejX,
37         WejZ => WejZ,
38         wyjX => wyjX,
39         wyjW => wyjW,
40         wyjZ => wyjZ,
41         wyjY => wyjY
42     );
43
44     WejW <= NOT WejW AFTER 800ns;
45     WejX <= NOT WejX AFTER 400ns;
46     WejY <= NOT WejY AFTER 200ns;
47     WejZ <= NOT WejZ AFTER 100ns;
48 END;
```

3.2.5 Symulacja



4 Wnioski

Do wykonania zadań wymagana była podstawowa wiedza z układów cyfrowych oraz bramek logicznych.
Pierwsze zadanie polegało na przetestowaniu jednej z dostępnych bramek aby pokazać działanie symulatora oraz nauczyć nas wgrywania programu na dostępny układ.
Drugie zadanie wymagało od nas zastosowania funkcji boolowskich oraz algebry Boole'a.
Trzecie zadanie wymagało od nas napisania translatora z kodu naturalnego binarnego do kodu Aikena. Kod ten nie jest zupełny dlatego zastosowaliśmy technikę niekreślonych wyjściowych wektorów bitowych, dla wejść znajdujących się poza dziedziną funkcji translatora. W alternatywnym podejściu możliwe byłoby ustalenie wartości spoza zbioru wartości funkcji translatora oraz zastosowanie jej jako kodu błędu.
Podczas zajęć nauczyliśmy się także, że symulator chociaż proponuje własne nazwy wejść/wyjść to w późniejszych etapach nie przyjmuje ich podczas budowania wersji programowalnej, głównie chodzi tutaj o symbole specjalne typu "_".