Układy logiczne

Technika cyfrowa i mikroprocesorowa
dr inż. Dominik Łuczak

1

Algebra Boole'a

2

Poznan University of Technology
Institute of Control, Robotics and Information Engineering

Operatory

$\wedge$	0	1
0	0	0
1	0	1

$\vee$	0	1
0	0	1
1	1	1

$\rightarrow$	0	1
0	1	1
1	0	1

$\oplus$	0	1
0	0	1
1	1	0

Figure 1. Truth tables

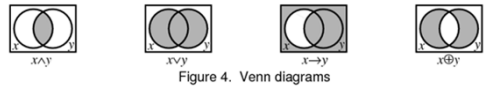


Figure 4. Venn diagrams

3

Operatory

Tablica prawdy dla negacji[3]:

p	$\neg p$
0	1
1	0

	Schröder Peirce	Peano Russell	Hilbert	Łukasiewicz
Negacja	p'	$\sim p$	$\bar{p}$	Np

4

Operatory

Tablica prawdy dla alternatywy

p	q	$p \vee q$
0	0	0
0	1	1
1	0	1
1	1	1

	Schröder Peirce	Peano, Russell Hilbert	Łukasiewicz
Alternatywa	$p + q$	$p \vee q$	Apq

5

Operatory - Binegacja

A	B	A NOR B
0	0	1
0	1	0
1	0	0
1	1	0

$\overline{A \vee B}$

6

Operatory

Tablica prawdy dla alternatywy rozłącznej

p	q	$p \veebar q$
0	0	0
0	1	1
1	0	1
1	1	0

EOR, **EXOR**, $\veebar$, $\dot{\vee}$, $\dot{\mathrm{v}}$, $\oplus$

7

negacji alternatywy wykluczającej

Input		Output
A	B	A XNOR B
0	0	1
0	1	0
1	0	0
1	1	1

XNOR operation is S = A ⊙ B.

8

Operatory

Tablica prawdy dla koniunkcji[2]:

p	q	$p \wedge q$
0	0	0
0	1	0
1	0	0
1	1	1

	Schröder Peirce	Peano Russell	Hilbert	Łukasiewicz
Koniunkcja	$p \cdot q$	$p.q$	$p \& q$	Kpq

9

Operatory

Tablica prawdy dla dysjunkcji

p	q	p/q
0	0	1
0	1	1
1	0	1
1	1	0

$\overline{p \wedge q}$

10

Operatory – wybrane zestawienie

NEGACJA $\neg$
KONIUNKCJA $\wedge$
ALTERNATYWA $\vee$
ALTERNATYWA WYKLUCZAJĄCA $\oplus$

11

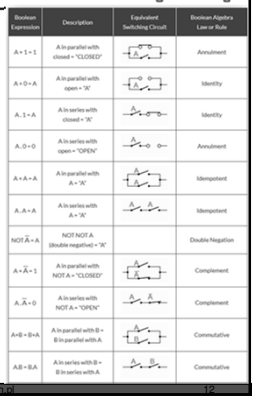
Tożsamości

$A \wedge 0 = 0$ $\quad A \vee 0 = A$
$A \wedge 1 = A$ $\quad A \vee 1 = 1$
$A \wedge A = A$ $\quad A \vee A = A$
$A \wedge \bar{A} = 0$ $\quad A \vee \bar{A} = 1$

Przemienności

$$A \wedge B = B \wedge A$$

$$A \vee B = B \vee A$$



12

łączności

$$A \wedge (B \wedge C) = (A \wedge B) \wedge C$$

$$A \vee (B \vee C) = (A \vee B) \vee C$$

13

rozdzielczości

$$A \wedge (B \vee C) = A \wedge B \vee A \wedge C$$
$$A \vee B \wedge C = (A \vee B) \wedge (A \vee C)$$

14

$$A \wedge \bar{A} = 0$$
$$A \vee \bar{A} = 1$$

15

$$\bar{\bar{A}} = A$$

$$\overline{\overline{A \wedge B}} = A \wedge B$$
$$\overline{\overline{A \vee B}} = A \vee B$$

16

De Morgana

$$\overline{A \wedge B} = \bar{A} \vee \bar{B}$$
$$\overline{A \vee B} = \bar{A} \wedge \bar{B}$$

$\overline{\boldsymbol{A \wedge B}} \neq \bar{\boldsymbol{A}} \wedge \bar{\boldsymbol{B}}$ **różne**
$\overline{\boldsymbol{A \vee B}} \neq \bar{\boldsymbol{A}} \vee \bar{\boldsymbol{B}}$ **różne**

17

De Morgana

$$\overline{A \wedge B \wedge C \wedge \ldots \wedge D} = \bar{A} \vee \bar{B} \vee \bar{C} \vee \ldots \vee \bar{D}$$

$$\overline{A \vee B \vee C \vee \ldots \vee D} = \bar{A} \wedge \bar{B} \wedge \bar{C} \wedge \ldots \wedge \bar{D}$$

18

Redukcja przez absorbcję

$$A \vee (A \wedge B) =$$
$$A \wedge (1 \vee (1 \wedge B)) =$$
$$= A$$

$$A \wedge (A \vee B) = A \wedge A \vee A \wedge B = A \vee A \wedge B = A$$

19

Redukcja

$$(A \wedge B) \vee (A \wedge \overline{B}) = A \, (B \wedge \overline{B}) = A$$

$$(A \vee B) \wedge (A \vee \overline{B}) =$$
$$(A \wedge A \vee A \wedge \overline{B}) \vee (B \wedge A \vee B \wedge \overline{B})$$
$$= A$$

20

Redukcja

$$A \vee (\overline{A} \wedge B) = A \vee B$$

$$A \wedge (\overline{A} \vee B) = A \wedge B$$

21

Częściowe zestawienie

AND operations		OR operations		INV operations	
Truth table	Laws	Truth table	Laws	Truth table	Laws
0 · 0 = 0	A · 0 = 0	0 + 0 = 0	A + 0 = A	0' = 1	A'' = A
1 · 0 = 0	A · 1 = A	1 + 0 = 1	A + 1 = 1	1' = 0	
0 · 1 = 0	A · A = A	0 + 1 = 1	A + A = A		
1 · 1 = 1	A · A' = 0	1 + 1 = 1	A + A' = 1		

Associative Laws	Commutative Laws	Distributive Laws
(A·B)·C = A·(B·C) = A·B·C	A·B·C = B·A·C = ...	A·(B+C) = (A·B) + (A·C)
(A+B)+C = A+(B+C) = A+B+C	A+B+C = B+C+A = ...	A+(B·C) = (A+B) · (A+C)

Truth tables to verify distributive laws

A	B	C	A+B	B+C	A+C	A·B	B·C	A·C	A·(B+C)	(A·B)+(A·C)	A+(B·C)	(A+B)·(A+C)
0	0	0	0	0	0	0	0	0	**0**	**0**	**0**	**0**
0	0	1	0	1	1	0	0	0	**0**	**0**	**0**	**0**
0	1	0	1	1	0	0	0	0	**0**	**0**	**0**	**0**
0	1	1	1	1	1	0	1	0	**0**	**0**	**1**	**1**
1	0	0	1	0	1	0	0	0	**0**	**0**	**1**	**1**
1	0	1	1	1	1	0	0	1	**1**	**1**	**1**	**1**
1	1	0	1	1	1	1	0	0	**1**	**1**	**1**	**1**
1	1	1	1	1	1	1	1	1	**1**	**1**	**1**	**1**

22

Zadanie

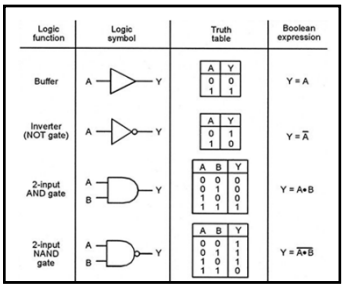
$d=(\overline{a \wedge \overline{b \vee c}})$

23

Implementacja sprzętowa

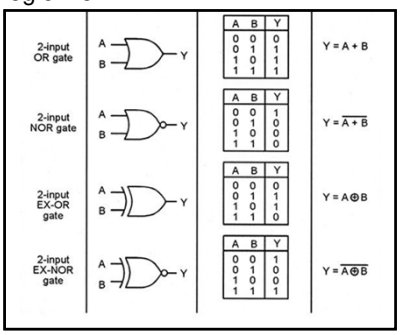
24

Bramki logiczne



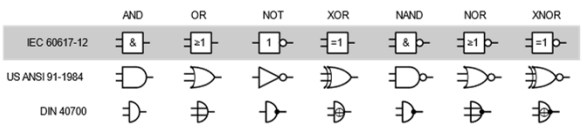
25

Bramki logiczne

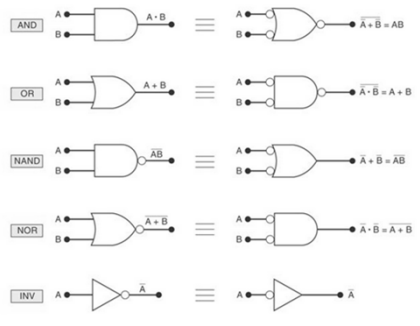


26

Symbole



27



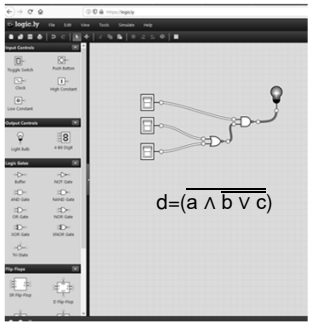
28

Implementacja

$d=\overline{(a \wedge \overline{b} \vee c)}$

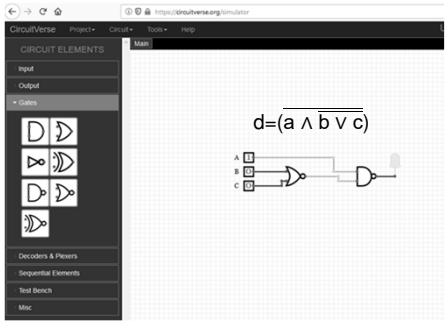
29

https://logic.ly/



30

https://circuitverse.org/simulator

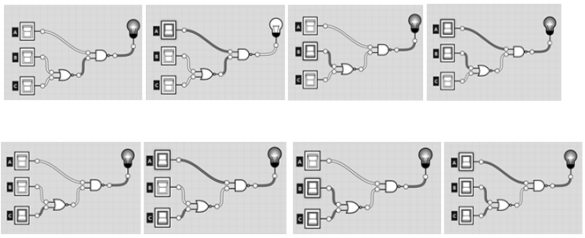


$d=\overline{(a \wedge \overline{b \vee c})}$

31

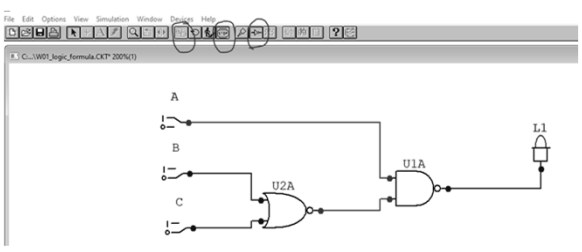
https://logic.ly/

$d=\overline{(a \wedge \overline{b \vee c})}$



32

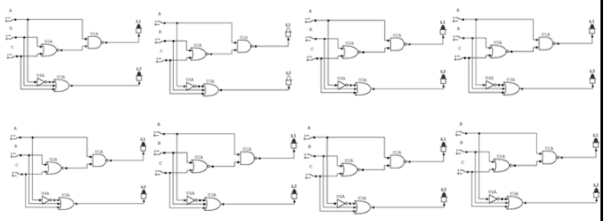
$d=\overline{(a \wedge \overline{b \vee c})}$



33

Poznan University of Technology
Institute of Control, Robotics and Information Engineering

$d=\overline{a \wedge \overline{b \vee c}}=(\bar{a} \vee b \vee c)$



34

Poznan University of Technology
Institute of Control, Robotics and Information Engineering

Zadanie domowe

$d=\overline{(a \wedge b \vee \overline{a \wedge \overline{b \vee c}})}$

35

Implementacja programowa

36

C/C++

Operator	Meaning	Example	Result
&&	Logical and	(5<2)&&(5>3)	False
||	Logical or	(5<2)||(5>3)	True
!	Logical not	!(5<2)	True

37

C/C++

Operator	precedence
!	High
&&	Medium
||	Low

A || B && C	means	A || (B && C)
A && B || C && D	means	(A && B) || (C && D)
A && B && C || D	means	((A && B) && C) || D
!A && B || C	means	((!A) && B) || C

38

Język C – operatory logiczne

Symbol	Nazwa	Przykład	Zapis matematyczny
!	Logiczna negacja	!a	¬a lub $\overline{a}$
||	Logiczna alternatywa (OR)	a || b	a | b lub a ∨ b
&&	Logiczna koniunkcja (AND)	a && b	a & b lub a ∧ b

39

Język C – operatory logiczne

operator	rodzaj porównania
==	czy równe
>	większy
>=	większy bądź równy
<	mniejszy
<=	mniejszy bądź równy
!=	czy różny (nierówny)

40

Język C – operatory logiczne

Operator	Przykład	Wynik	Przykład II	Wynik II
==	'a'==1	fałsz	'a'==97	prawda
>	2>3	fałsz	2>1	prawda
>=	2>=3	fałsz	2>=2	prawda
<	2<1	fałsz	1<'a'	prawda
<=	3.5<=2.5	fałsz	2.5<=3.5	prawda
!=	'a'!='a'	fałsz	'a'!='b'	prawda

41

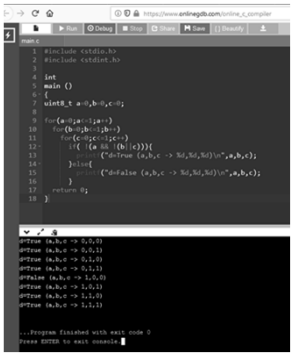
Operatory logiczne w językach komputerowych

język komputerowy	jednoargumentowe	dwuargumentowe					
	negacja	koniunkcja	alternatywa	alternatywa rozłączna	implikacja	równoważność	inne
ABAP[47]	NOT	AND	OR	×	×	×	×
Ada[6]	not	and and then	or or else	xor	×	×	×
C[48][9], C++[9]	!	&&	||	!=	×	==	×
Clipper[9]	.not.	.and.	.or.	×	×	×	×
COBOL[10]	NOT	AND	OR	×	×	×	×
Forth[11]	×	AND	OR	XOR	×	×	×
Fortran 1900[12]	.NOT.	.AND.	.OR.	×	×	×	×
Fortran 77[13]	.NOT.	.AND.	.OR.	.NEQV.	×	.EQV.	×
JEAN[14][15]	NOT	AND	OR	IS	×	×	×
Java[16]	!	&&	||	!=	×	==	×
Modula-2[17]	NOT	AND &	OR	×	×	×	×
Pascal[6]	not	and	or	xor	×	×	×
Perl[18]	! not	&& and	|| // or	xor	×	×	×
PHP[19]	!	&& and	|| or	xor	×	×	×
PL/I[20]	¬	&	|	×	×	×	×
PL/M[21][22]	NOT	AND	OR	XOR	×	×	×
Python[23]	not	and	or	×	×	×	×
REXX[24][25]	\ ¬	&	|	&&	×	×	×
VHDL[26]	not	and	or	xor	×	xnor	nand nor
Visual Basic[28], VBA[27]	Not	And AndAlso	Or OrElse	Xor	Imp	Eqv	×

42

C/C++ **d=(a ∧ $\overline{b}$ ∨ c)**

```
//Wariant 1
if( !(a && !(b||c))){
}
```



43

C/C++ **d=(a ∧ $\overline{b}$ ∨ c)=**
=($\overline{a}$ ∨ b ∨ c)

```
//Wariant 1
if( !(a && !(b||c))){
}
//Wariant 2 – po uproszczeniu
if( !a || b || c){
}
```

44

C/C++ Zadanie domowe

d=$\overline{(a \wedge \overline{b} \vee \overline{a \wedge \overline{b}} \vee c)}$

45

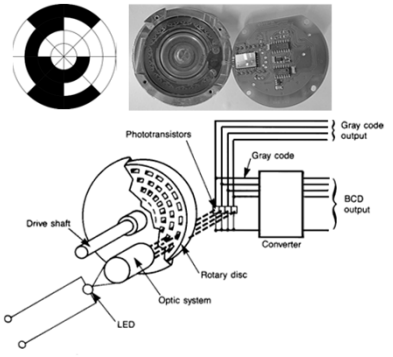
Gray code -> Binary

46

Poznan University of Technology
Institute of Control, Robotics and Information Engineering

Gray

Decimal	Binary	Gray
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

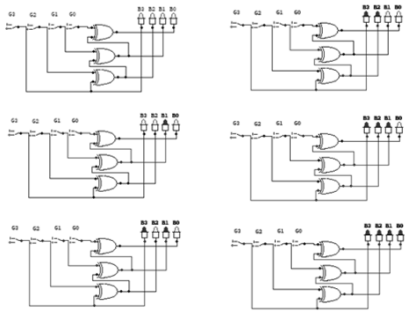


47

Poznan University of Technology
Institute of Control, Robotics and Information Engineering

Gray

Decimal	Binary	Gray
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000



48

Gray code -> Binary
C/C++

49

Poznan University of Technology
Institute of Control, Robotics and Information Engineering

Język C – operatory bitowe

Symbol	Nazwa
~	Bitowa negacja
|	Bitowa alternatywa (OR)
&	Bitowa koniunkcja (AND)
^	Bitowa różnica symetryczna (XOR)
>>	Bitowe przesunięcie w prawo
<<	Bitowe przesunięcie w lewo

50

Poznan University of Technology
Institute of Control, Robotics and Information Engineering

Język C – operatory bitowe

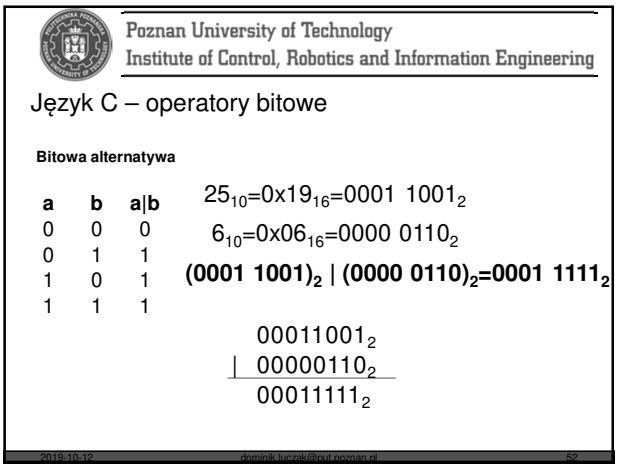
Bitowa negacja

a	~a
0	1
1	0

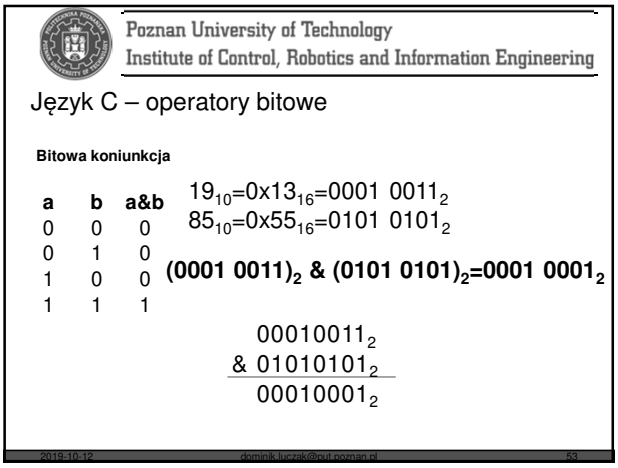
$25_{10}=0x19_{16}=0001\ 1001_2$

$\sim(25)_{10}=\sim(0001\ 1001)_2=1110\ 0110_2$

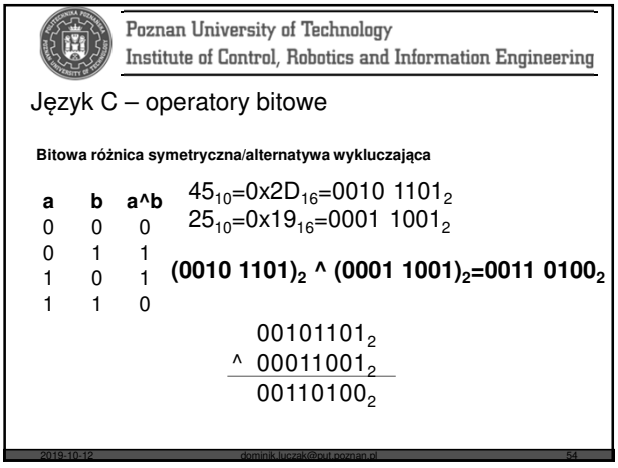
51


Poznan University of Technology
Institute of Control, Robotics and Information Engineering
Język C – operatory bitowe
Bitowa alternatywa
a b a|b
0 0 0
0 1 1
1 0 1
1 1 1
25_{10}=0x19_{16}=0001 1001_2
6_{10}=0x06_{16}=0000 0110_2
(0001 1001)$_2$ | (0000 0110)$_2$=0001 1111_2
00011001_2
| 00000110_2
00011111_2
2019-10-12 dominik.luczak@put.poznan.pl 52

52


Poznan University of Technology
Institute of Control, Robotics and Information Engineering
Język C – operatory bitowe
Bitowa koniunkcja
a b a&b
0 0 0
0 1 0
1 0 0
1 1 1
19_{10}=0x13_{16}=0001 0011_2
85_{10}=0x55_{16}=0101 0101_2
(0001 0011)$_2$ & (0101 0101)$_2$=0001 0001_2
00010011_2
& 01010101_2
00010001_2
2019-10-12 dominik.luczak@put.poznan.pl 53

53


Poznan University of Technology
Institute of Control, Robotics and Information Engineering
Język C – operatory bitowe
Bitowa różnica symetryczna/alternatywa wykluczająca
a b a^b
0 0 0
0 1 1
1 0 1
1 1 0
45_{10}=0x$2D_{16}$=0010 1101_2
25_{10}=0x19_{16}=0001 1001_2
(0010 1101)$_2$ ^ (0001 1001)$_2$=0011 0100_2
00101101_2
^ 00011001_2
00110100_2
2019-10-12 dominik.luczak@put.poznan.pl 54

54

Język C – operatory bitowe

Bitowe przesunięcie w prawo (>>)

a >> b

$a=48_{10}=0x30_{16}=0011\ 0000_2$

(a>>2) = (0011 0000$_2$ >> 2) = 0000 1100$_2$ = 12$_{10}$

$$\frac{48}{2^2} = \frac{48}{4} = 12$$

55

Język C – operatory bitowe

Bitowe przesunięcie w lewo (<<)

a << b

$a=48_{10}=0x30_{16}=0011\ 0000_2$

(a<<2) = (0011 0000$_2$ << 2) = 1100 0000$_2$ = 192$_{10}$

$$48 \cdot 2^2 = 48 \cdot 4 = 192$$

56

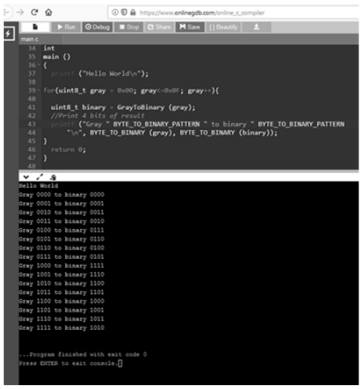
https://www.onlinegdb.com/online_c_compiler

Decimal	Binary	Gray
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

57

https://www.onlinegdb.com/online_c_compiler

Decimal	Binary	Gray
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

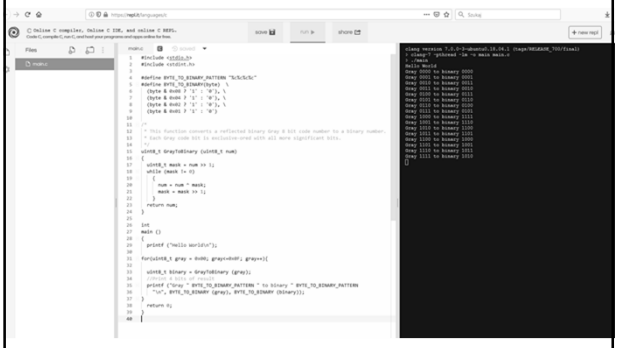


```
int
main ()
{
  printf ("Hello World\n");

for(uint8_t gray = 0x00; gray<=0x0F; gray++){

  uint8_t binary = GrayToBinary (gray);
  //Print 4 bits of result
  printf ("Gray " BYTE_TO_BINARY_PATTERN " to binary " BYTE_TO_BINARY_PATTERN
    "\n", BYTE_TO_BINARY (gray), BYTE_TO_BINARY (binary));
}
 return 0;
}
```

```
Hello World
Gray 0000 to binary 0000
Gray 0001 to binary 0001
Gray 0010 to binary 0011
Gray 0011 to binary 0010
Gray 0100 to binary 0111
Gray 0101 to binary 0110
Gray 0110 to binary 0100
Gray 0111 to binary 0101
Gray 1000 to binary 1111
Gray 1001 to binary 1110
Gray 1010 to binary 1100
Gray 1011 to binary 1101
Gray 1100 to binary 1000
Gray 1101 to binary 1001
Gray 1110 to binary 1011
Gray 1111 to binary 1010

...Program finished with exit code 0
Press ENTER to exit console.
```


58

https://repl.it/languages/c



59