

Podstawy Informatyki

Katedra Telekomunikacji, EiT

dr inż. Jarosław Bułat

kwant@agh.edu.pl

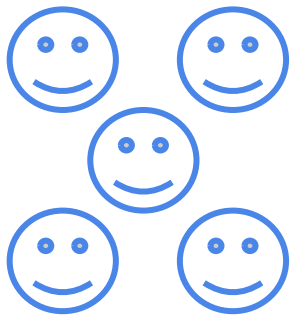
Plan prezentacji

- » Systemy liczbowe, systemy pozycyjne
- » Operatory porównania, operacje bitowe
- » Konwersje typów
- » Tekst w C++
- » Typ zmienne-przecinkowy
- » Algorytm: schemat blokowy, pseudocode, implementacja
- » GIT - gałęzie

Jak zapisać liczbę?

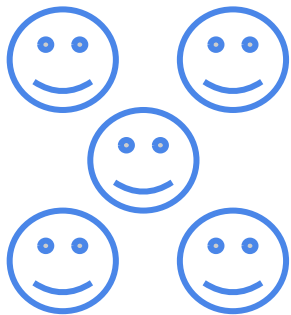
w systemie pozycyjnym, binarnym

Systemy pozycyjne



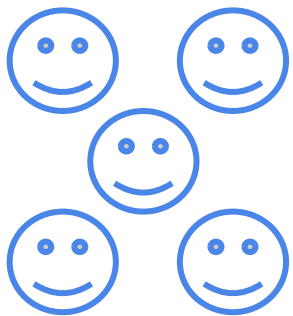
» Zapis w systemie

Systemy pozycyjne



- » Zapis w systemie
– jedynekowym

Systemy pozycyjne

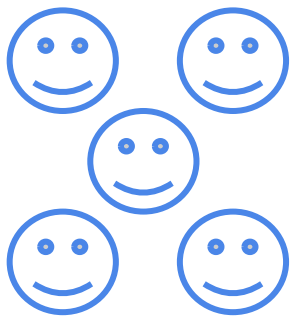


|||||

5

- » Zapis w systemie
- jedynekowym
 - dziesiętnym

Systemy pozycyjne



|||||

5

12

101

- » Zapis w systemie
- jedynekowym
 - dziesiętnym
 - trójkowym
 - binarnym

Systemy pozycyjne

$$123 = 1 \times 100 + 2 \times 10 + 3$$

» Reguły zapisu pozycyjnego

Systemy pozycyjne

$$\begin{aligned} 123 &= 1 \times 100 + 2 \times 10 + 3 \\ &= 1 \times 100 + 2 \times 10 + 3 \times 1 \end{aligned}$$

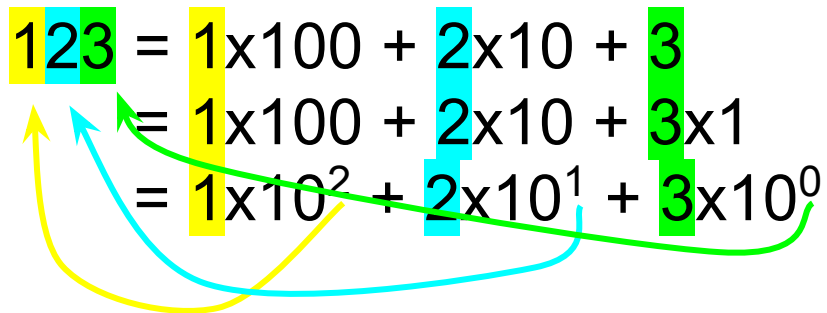
» Reguły zapisu pozycyjnego

Systemy pozycyjne

$$\begin{aligned} 123 &= 1 \times 100 + 2 \times 10 + 3 \\ &= 1 \times 100 + 2 \times 10 + 3 \times 1 \\ &= 1 \times 10^2 + 2 \times 10^1 + 3 \times 10^0 \end{aligned}$$

- » Reguły zapisu pozycyjnego
- » Potęgi to numer pozycji cyfry licząc od prawej strony (LSB)

Systemy pozycyjne


$$\begin{aligned}123 &= 1 \times 100 + 2 \times 10 + 3 \\&= 1 \times 100 + 2 \times 10 + 3 \times 1 \\&= 1 \times 10^2 + 2 \times 10^1 + 3 \times 10^0\end{aligned}$$

- » Reguły zapisu pozycyjnego
- » Potęgi to numer pozycji cyfry licząc od prawej strony (LSB)

Systemy pozycyjne

$$\begin{aligned} 123 &= 1 \times 100 + 2 \times 10 + 3 \\ &= 1 \times 100 + 2 \times 10 + 3 \times 1 \\ &= 1 \times 10^2 + 2 \times 10^1 + 3 \times 10^0 \end{aligned}$$

$$\begin{aligned} 101 &= 1 \times 4 + 0 \times 2 + 1 \times 1 \\ &= 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 \end{aligned}$$

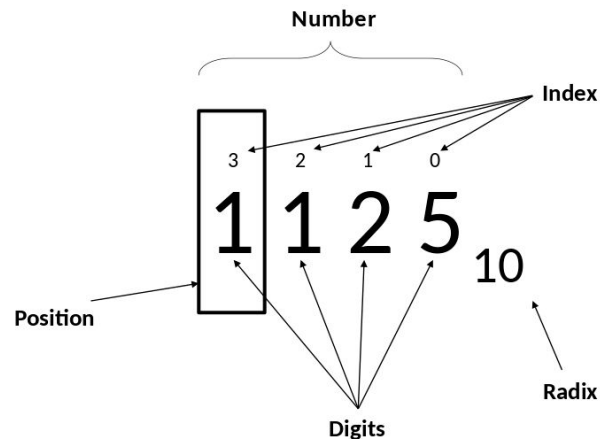
- » Reguły zapisu pozycyjnego
- » Potęgi to numer pozycji cyfry licząc od prawej strony (LSB)
- » Podstawa systemu pozycyjnego nie musi być 10
- » Może być inny, np. 2

Systemy pozycyjne

$$123 = 1 \times 10^2 + 2 \times 10^1 + 3 \times 10^0$$

» Ogólna zależność systemu pozycyjnego

$$\text{Number} = \sum_i D_i * R^i$$



https://en.wikipedia.org/wiki/Positional_notation

Systemy pozycyjne

$$123 = 1 \times 10^2 + 2 \times 10^1 + 3 \times 10^0$$

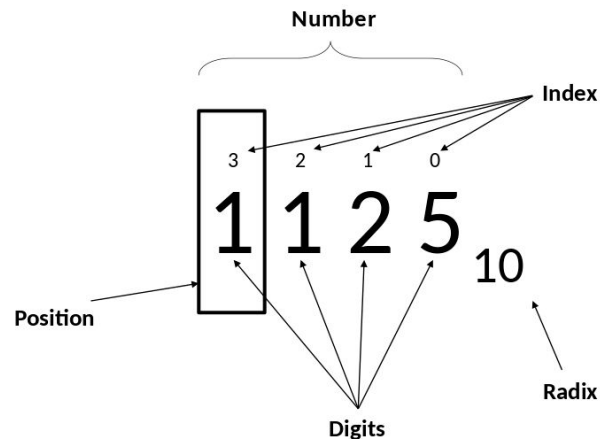
» Ogólna zależność systemu pozycyjnego

$$\text{Number} = \sum_i D_i * R^i$$

$$1125 = D_3 * R^3 + D_2 * R^2 + D_1 * R^1 + D_0 * R^0$$

$$1125 = 1 * 10^3 + 1 * 10^2 + 2 * 10^1 + 5 * 10^0$$

$$1125 = 1 * 1000 + 1 * 100 + 2 * 10 + 5 * 1$$



https://en.wikipedia.org/wiki/Positional_notation

Konwersja pomiędzy systemami liczbowymi

konwersja bin-dec-oct-hex

b	bb	bbb	bbbb	d	o	h
--	---	----	-----	-	-	-
0	00	000	0000	0	0	0
1	01	001	0001	1	1	1
	10	010	0010	2	2	2
	11	011	0011	3	3	3
		100	0100	4	4	4
		101	0101	5	5	5
		110	0110	6	6	6
		111	0111	7	7	7
			1000	8	10	8
			1001	9	11	9
			1010	10	12	A
			1011	11	13	B
			1100	12	14	C
			1101	13	15	D
			1110	14	16	E
			1111	15	17	F

b = {0, 1}

o = {0, 1, 2, 3, 4, 5, 6, 7}

d = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

h = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F}

konwersja bin-dec-oct-hex

b	bb	bbb	bbbb	d	o	h
--	---	----	-----	-	-	-
0	00	000	0000	0	0	0
1	01	001	0001	1	1	1
	10	010	0010	2	2	2
	11	011	0011	3	3	3
		100	0100	4	4	4
		101	0101	5	5	5
		110	0110	6	6	6
		111	0111	7	7	7
			1000	8	10	8
			1001	9	11	9
			1010	10	12	A
			1011	11	13	B
			1100	12	14	C
			1101	13	15	D
			1110	14	16	E
			1111	15	17	F

- » **bin**: "używają komputery"
- » **dec/oct/hex**: komunikacja z człowiekiem
 - drukowanie
 - wypisywanie na ekranie
 - wprowadzanie (klawiatura)
 - *.txt
- » **dec**: komunikacja **m2m**
 - html
 - json
 - txt

konwersja bin-dec-oct-hex

0 1 0 0 0 1 1 0 = 01000110₂ <- 8 bitów

2^7 2^6 2^5 2^4 2^3 2^2 2^1 2^0

konwersja bin-dec-oct-hex

0 1 0 0 0 1 1 0 = 01000110_2 <- 8 bitów

 2^7 2^6 2^5 2^4 2^3 2^2 2^1 2^0
128 64 32 16 8 4 2 1 <- wagi

konwersja bin-dec-oct-hex

0 1 0 0 0 1 1 0 = 01000110_2 <- 8 bitów

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1

<- wagi

0	64	0	0	0	4	2	0

= 70_{10}

<- wynik

konwersja bin-dec-oct-hex

0 1 0 0 0 1 1 0 = 01000110_2 <- 8 bitów

 2^7 2^6 2^5 2^4 2^3 2^2 2^1 2^0

128 64 32 16 8 4 2 1 <- wagi

 0 64 0 0 0 4 2 0 = 70_{10} <- wynik

$$N = \text{sum}_i(d_i r^i) = d_n r^n + \dots + d_3 r^3 + d_2 r^2 + d_1 r^1 + d_0 \quad \text{Number} = \sum_i D_i * R^i$$

gdzie:

N - wynik, r - podstawa systemu, d - wartość znaku, i - numer znaku

konwersja bin-dec-oct-hex

0 1 0 0 **1** 1 1 0 = 01000110₂

<- 8 bitów

 2^7 2^6 2^5 2^4 2^3 2^2 2^1 2^0

128	64	32	16	8	4	2	1
200	100	40	20	10	4	2	1
80	40	80	10	8	4	2	1

<- wagi dec

<- wagi oct

<- wagi hex

0	64	0	0	8	4	2	0	= 78 ₁₀
0	100	0	0	10	4	2	0	= 116 ₈
0	40	0	0	8	4	2	0	= 4E ₁₆

<- wynik dec

<- wynik oct

<- wynik hex

konwersja bin-dec-oct-hex

0 1 0 0 1 1 1 0 = 01000110_2 <- 8 bitów

 2^7 2^6 2^5 2^4 2^3 2^2 2^1 2^0

128 64 32 16 8 4 2 1

200 100 40 20 10 4 2 1

80 40 20 10 8 4 2 1

 0 64 0 0 8 4 2 0 = 78_{10}

0 1 0 0 1 4 2 0 = $\{1\ 1\ 6\}_8$

0 4 0 0 8 4 2 0 = $\{4\ E\}_{16}$

<- wagi dec

<- wagi oct

<- wagi hex

<- wynik dec

<- wynik dec

<- wynik dec



quiz

PI03_bit

socrative.com

- login
- student login

Room name:

KWANTAGH

słowa

- » $\{0,1\}$ - bit (ang. **bit**)
- » 8 bitów - bajt (ang. **byte**)
- » 16 bitów (2 bajty) - słowo (ang. **word**)
- » 32 bitów (4 bajty) - słowo
- » 64 bity (8 bajtów) - słowo

słowa

- » {0,1} - bit (ang. **bit**)
 - » **8 bitów** - **bajt** (ang. **byte**)
 - » 16 bitów (2 bajty) - słowo (ang. **word**)
 - » 32 bitów (4 bajty) - słowo
 - » 64 bity (8 bajtów) - słowo
-
- » 1 bajt == 8 bitów == 1 znak (liczba, cyfra, znaki specjalne, etc...)

słowa

- » {0,1} - bit (ang. **bit**)
 - » **8 bitów** - **bajt** (ang. **byte**)
 - » 16 bitów (2 bajty) - słowo (ang. **word**)
 - » 32 bitów (4 bajty) - słowo
 - » 64 bity (8 bajtów) - słowo
-
- » 1 bajt == 8 bitów == 1 znak (liczba, cyfra, znaki specjalne, etc...)
 - » **ISO/IEC 2382-1:1993** w konwencji 2^x czyli wartości 0-255 (256 stanów)

przedrostki

Wielokrotności bitów					
Przedrostki dziesiętne (SI)			Przedrostki binarne (IEC 60027-2)		
Nazwa	Symbol	Mnożnik	Nazwa	Symbol	Mnożnik
kilobit	kb	$10^3=1000^1$	kibibit	Kib	$2^{10}=1024^1$
<u>megabit</u>	Mb	$10^6=1000^2$	mebibit	Mib	$2^{20}=1024^2$
gigabit	Gb	$10^9=1000^3$	gibibit	Gib	$2^{30}=1024^3$
terabit	Tb	$10^{12}=1000^4$	tebibit	Tib	$2^{40}=1024^4$
petabit	Pb	$10^{15}=1000^5$	pebibit	Pib	$2^{50}=1024^5$
eksabit	Eb	$10^{18}=1000^6$	eksbibit	Eib	$2^{60}=1024^6$
zettabit	Zb	$10^{21}=1000^7$	zebibit	Zib	$2^{70}=1024^7$
jottabit	Yb	$10^{24}=1000^8$	jobibit	Yib	$2^{80}=1024^8$

Jak porównać coś w C++?

operatorem

Operatory porównania

```
#include <iostream>
```

```
int main(){
```

```
    int a = 3;
```

```
    int b = 0;
```

```
    int c = (a < b);
```

```
    cout << c << endl;
```

```
    cout << (a > b) << endl;
```

```
    cout << (4 + 2 <= 2 * a) << endl;
```

```
    bool b1 = true;           // or false
```

```
    bool b2 = (4 + 2) <= (2 * a);
```

```
    bool b3 = 4 + 2 < 2 * a;
```

```
}
```

== równe

!= różne

> większe

< **mniejsze**

>= większe bądź równe

<= mniejsze bądź równe

- » zwracają wartość logiczną
- » z powodu kompatybilności z C:
 - false: 0
 - true: wszystko oprócz 0

Operatory porównania

```
#include <iostream>
```

```
int main(){
```

```
    int a = 3;
```

```
    int b = 0;
```

```
    int c = (a < b);
```

```
    cout << c << endl;
```

```
    cout << (a > b) << endl;
```

```
    cout << (4 + 2 <= 2 * a) << endl;
```

```
    bool b1 = true;           // or false
```

```
    bool b2 = (4 + 2) <= (2 * a);
```

```
    bool b3 = 4 + 2 < 2 * a;
```

```
}
```

musi być w nawiasie, inaczej błąd niski priorytet operatora <=

== równe

!= różne

> większe

< **mniejsze**

>= większe bądź równe

<= mniejsze bądź równe

- » zwracają wartość logiczną
- » z powodu kompatybilności z C:
 - false: 0
 - true: wszystko oprócz 0

Operatory logiczne

```
// interval: 0 <= x < 1
int x = 0;
bool res1 = (x >= 0 && x < 1);
bool res2 = (x >= 0 || x < 1);           // always true

bool res3 = (x >= 0 || ++x < 1);        // NEVER EVER !!!

cout << x;
```

|| suma **logiczna**
&& iloczyn **logiczny**
! negacja **logiczna**

» **x || y**
suma logiczna x i y, wynik to:
zero jeśli x i y mają wartość zero
niezerowy w innym przypadku

» niezerowy oznacza inny niż 0,
może być 1, -1, 2342342, etc...

Operatory bitowe

```
int a1 = 12;           // 00001100
int a2 = 24;           // 00011000

int b1 = a1 | a2;       // 00011100 == 28
int b2 = a1 & a2;       // 00001000 == 8
int b3 = b2 >> 1;       // 00000100 == 4
int b4 = b1 << 2;       // 01110000 == 112
int b5 = b4 >> 0;       // 01110000 == 112
int b6 = b4 >> 9;       // 00000000 == 0
```

```
// did b5 has "1" on 4th position?
```

```
bool res = b5 & (1 << 4);
cout << res << endl;
```

| suma **bitowa**
& iloczyn **bitowy**
^ modulo2 **bitowe**
<< przesunięcie **bitowe** L
>> przesunięcie **bitowe** P
~ negacja **bitowa**

- » działa na liczbach całkowitych
- » działa na **wszystkich** bitach na raz
- » **nie mylić & z &&...**



quiz

PI03_cmp

socrative.com

- login
- student login

Room name:

KWANTAGH

Operator rzutowania

```
int a = 1 << 20;           // 32 bits
short b = a;               // 16 bits !!!

b = 1234;
a = b;                     // ok

float f = 1.9;
int x = f;                 // x == 1

int c = -1;
unsigned int d = c;        // 4294954873

b = (short)a;              // C
b = short(a);              // C
b = static_cast<short>(a); // C++
```

- » C++ jest statycznie typowany ale posiada ograniczoną automatyczną konwersję
- » **preferowana jawna konwersja**
- » możliwe przekłamania
 - kompilator nie zna danych
 - “runtime” nie sprawdza
- » W C++ rekomendowane:

dynamic_cast <new_type> (expression)
reinterpret_cast <new_type> (expression)
static_cast <new_type> (expression)
const_cast <new_type> (expression)

- » lepsza kontrola
- » większe możliwości

Operator sizeof()

```
int a;  
unsigned int b;  
long int c;  
float d;
```

```
size_t sd = sizeof(d);
```

```
cout << sizeof(a) << "\n";  
cout << sizeof(b) << "\n";  
cout << sizeof(c) << "\n";  
cout << sd << "\n";
```

- » sizeof(etykieta_zmiennej)
- » sizeof(typ)
- » to jest operator, nie funkcja!
- » podaje w bajtach rozmiar typu
- » **size_t** to jest* **unsigned int**
 - sugestia że zmienna wyraża **rozmiar** w bajtach
 - jest nieujemny
 - jest liczbą naturalną**
 - często wyraża kolejność, pozycję, rozmiar tablicy, ...
- » operator **typeid(...)**
 - specyficzne użycie
 - na razie nie potrzebujemy, wiemy jaki jest typ zmiennej

Operator - kolejność

- » kolejność obliczania operandów nie jest gwarantowana ze względu na możliwość optymalizacji kodu lub out-of-order
- » przy operacjach logicznych `&&`, `||` część obliczeń może zostać pominięta (jeżeli nie wpływa na wynik)
 - `a>0 || ++b<4`
- » użycie operandów które wpływają na wartość pozostałych nie jest ustandaryzowane co do kolejności, różnice pomiędzy operatorami
 - `int i =1;`
 - `tablica[i]=++i;` (nie jest pewne czy `tablica[1]` czy `tablica[2]`)
- » wniosek: **zrezygnuj ze złożonych obliczeń w jednym wyrażeniu**, rozbij na kilka wyrażeń (na kilka linii)

Tekst w C/C++

+dziwne literki... ąęśćłóźż

tekst w C++

```
char c = 'a';           // single quote
char d = 65;
```

- » **char** to krótki integer
 - liczba: -128...0...127
 - znak w kodzie ASCII
 - **'a'** to kod ASCII znaku

tekst w C++

```
char c = 'a';           // single quote
char d = 65;
```

```
// char == character
// "character" is pronounced "ka-rak-ter"
// "char" is usually pronounced "tchar", not "kar"
```

```
// source: http://www.stroustrup.com/bs\_faq2.html#char
```

- » **char** to krótki integer
 - liczba: -128...0...127
 - znak w kodzie ASCII
 - **'a'** to kod ASCII znaku

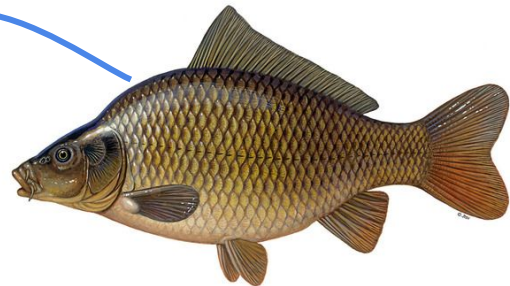
tekst w C++

```
char c = 'a';  
char d = 65;
```

// single quote

```
// char == character  
// "character" is pronounced "ka-rak-ter"  
// "char" is usually pronounced "tchar", not "kar"
```

```
// source: http://www.stroustrup.com/bs\_faq2.html#char
```



- » **char** to krótki inteager
 - liczba: -128...0...127
 - znak w kodzie ASCII
 - **'a'** to kod ASCII znaku

tekst w C++

```
char c = 'a';           // single quote
char d = 65;
char e = '1' + 1;
cout << c << "\n";      // character
cout << ++c << "\n";     // "next" char
cout << c+1 << "\n";     // conv. to int !!!
cout << char(c + 1) << "\n"; // print as char
cout << d << "\n";       // A == 65 in ASCII
cout << e << "\n";       // '2'
cout << (int)e << "\n";  // '2' means ASCII(50)
```

- » **char** to krótki integer
 - liczba: -128...0...127
 - znak w kodzie ASCII
 - **'a'** to kod ASCII znaku
- » można inkrementować
- » zwrócić uwagę na automatyczną konwersję typu !!
- » znaki specjalne



ASCII

- » American Standard for Code Information Interchange
- » **128 kodów:**
 - litery (duże/male)
 - cyfry
 - dodatkowe znaki
 - znaki specjalne
- » '\n' == 0xA (10) Unix
- » '\n' == 0xC 0xA Windows
- » pozostałe 128 kombinacji:
 - symbole
 - znaki narodowościowe
- » **PL: ISO8859-2 vs CP1250 latin-2**

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0 ^@ NUL NULL	1 ^A SOH START OF HEADING	2 ^B STX START OF TEXT	3 ^C ETX END OF TEXT	4 ^D EOT END OF TRANS.	5 ^E ENQ ENQUIRY	6 ^F ACK ACKNOWLEDGE	7 ^G BEL BELL	8 ^H BS BACKSP.	9 ^I HT CHARACT. TAB.	10 ^J LF LINE FEED	11 ^K VT LINE TAB.	12 ^L FF FORM FEED	13 ^M CR CARRIAGE RETURN	14 ^N SO SHIFT OUT	15 ^O SI SHIFT IN
1	16 ^P DLE	17 ^Q DC1	18 ^R DC2	19 ^S DC3	20 ^T DC4	21 ^U NAK	22 ^V SYN	23 ^W ETB	24 ^X CAN	25 ^Y EM	26 ^Z SUB	27 ^[ESC	28 ^\ FS	29 ^] GS	30 ^^ RS	31 ^_ US
2	32 SPACE	33 ^_ EXCLAM. MARK	34 ^" QUOT. MARK	35 ^# NUMBER SIGN	36 ^\$ DOLLAR SIGN	37 %^ PERCENT SIGN	38 ^& AMPER-SAND	39 ^' APOS-TROPHE	40 ^(LEFT PAREN.	41 ^) RIGHT PAREN.	42 ^* ASTERISK	43 ^+ PLUS SIGN	44 ^, COMMA	45 ^- HYPHEN-MINUS	46 ^. FULL STOP	47 ^/ SOLIDUS
3	48 DIGIT ZERO	49 DIGIT ONE	50 DIGIT TWO	51 DIGIT THREE	52 DIGIT FOUR	53 DIGIT FIVE	54 DIGIT SIX	55 DIGIT SEVEN	56 DIGIT EIGHT	57 DIGIT NINE	58 : colon	59 ; semi	60 : lt	61 = equals	62 > gt	63 ? quest
4	64 ^@ COMM'IAL AT	65 A	66 B	67 C	68 D	69 E	70 F	71 G	72 H	73 I	74 J	75 K	76 L	77 M	78 N	79 O
5	80 P	81 Q	82 R	83 S	84 T	85 U	86 V	87 W	88 X	89 Y	90 Z	91 [lsqb	92 \ bsol	93] rsqb	94 ^ hat	95 _ lowbar
6	96 ^` GRAVE ACCENT	97 a	98 b	99 c	100 d	101 e	102 f	103 g	104 h	105 i	106 j	107 k	108 l	109 m	110 n	111 o
7	112 p	113 q	114 r	115 s	116 t	117 u	118 v	119 w	120 x	121 y	122 z	123 { L. CURLY BRACKET	124 VERTICAL LINE	125 } R. CURLY BRACKET	126 ~ TILDE	127 ^? DEL DELETE

Unicode

» **UTF-8**

- » zestaw znaków mający obejmować wszystkie pisma świata
- » **jeden znak to 1-4 bajtów**
- » **kompatybilny** (w dół) **z ASCII**
- » pliterki 2 bajty/sztuka
- » 1 112 064 unikalne znaki

- » 1 bajt: **0xxxxxxx**, gdzie kolejne „x” to bity
- » 2 bajty: **110xxxxx 10xxxxxx**
- » 3 bajty: **1110xxxx 10xxxxxx 10xxxxxx**
- » 4 bajty: **11110xxx 10xxxxxx 10xxxxxx 10xxxxxx**

- » długość tekstu w znakach != długość tekstu w bajtach
- » niekompatybilne z C
- » trudności w sortowaniu z uwzględnieniem znaków narodowych

UTF-32/UCS-4

UTF-16

UTF-8

UTF-7

UCS-2

UTF-9 UTF-18

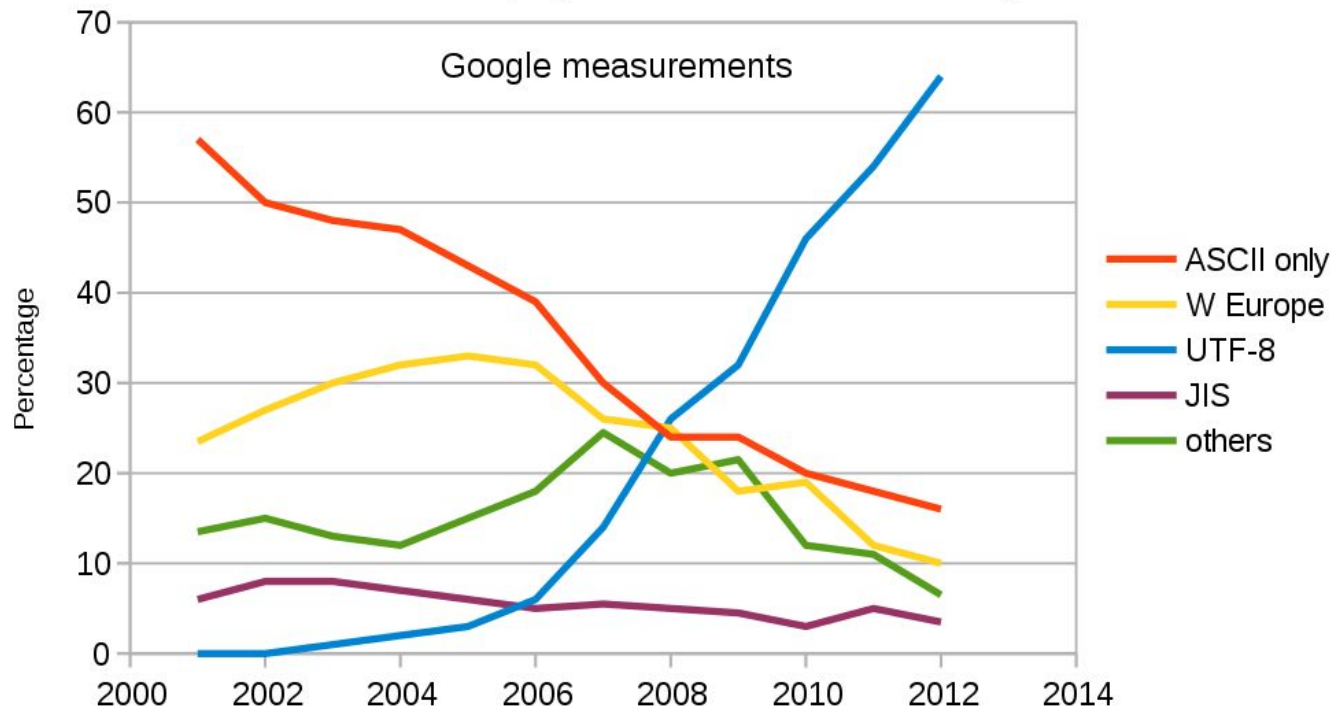
UTF-EBCDIC

UTF-6

UTF-5

UTF-8

Share of web pages with different encodings





quiz

PI03_char

socrative.com

- login
- student login

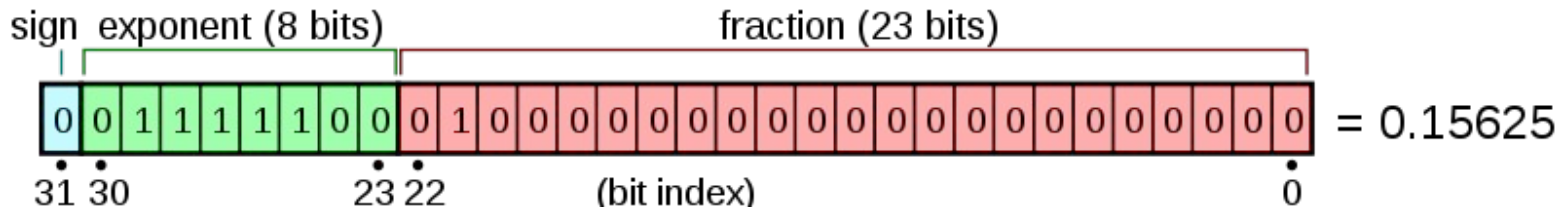
Room name:

KWANTAGH

Trochę arytmetyki...

$$0.7 == 0.6999999988079$$

IEEE-754 (float)



$$x = (-1)^S M \cdot 2^{E-\text{bias}}$$

- » 1 bit znaku, 8 bitów wykładnika (cechy), 23 bity mantysy, = **32 bity**
- » dokładność 7-8 cyfr
- » zakres $10^{-45} \dots 10^{38}$
- » specjalne wartości bitowa dla NaN, zero, inf
- » **mnożenie: $z=x*y \rightarrow$ liczba bitów wyniku: wykładnik 9, mantysa 46**
- » **wpisując z do zmiennej float wykonuje się uproszczenie**

IEEE-754 (float)

```
#include <iostream>
#include <iomanip>
using namespace std;

int main(){

    float a = 9.2;
    cout << std::setprecision(10); // "nothing"
    cout << a << "\n";           // 9.199999809

    float big = 100000000;         //1e8
    float small = 1;
    float res1 = (big + small) - big;
    float res2 = (big - big) + small;
    cout << res1 << "\n";
    cout << res2 << "\n";
}
```

- » **obliczenia numeryczne**
- » błędy kumulują się przy złożonych obliczeniach
- » kolejność wykonywania operacji arytmetycznych ma znaczenie
- » reprezentacja zmiennoprzecinkowa jest skwantowana
- » **po każdej operacji matematycznej wynik jest zaokrąglany!**

IEEE-754 (float)

```
#include <iostream>
#include <iomanip>
using namespace std;

int main(){

    float a = 9.2;
    cout << std::setprecision(10); // "nothing"
    cout << a << "\n";           // 9.199999809

    float big = 100000000;         //1e8
    float small = 1;
    float res1 = (big + small) - big;
    float res2 = (big - big) + small;
    cout << res1 << "\n";          // 0
    cout << res2 << "\n";          // 1
}
```

- » **obliczenia numeryczne**
- » błędy kumulują się przy złożonych obliczeniach
- » kolejność wykonywania operacji arytmetycznych ma znaczenie
- » reprezentacja zmiennoprzecinkowa jest skwantowana
- » **po każdej operacji matematycznej wynik jest zaokrąglany!**

ToDo - zadanie domowe

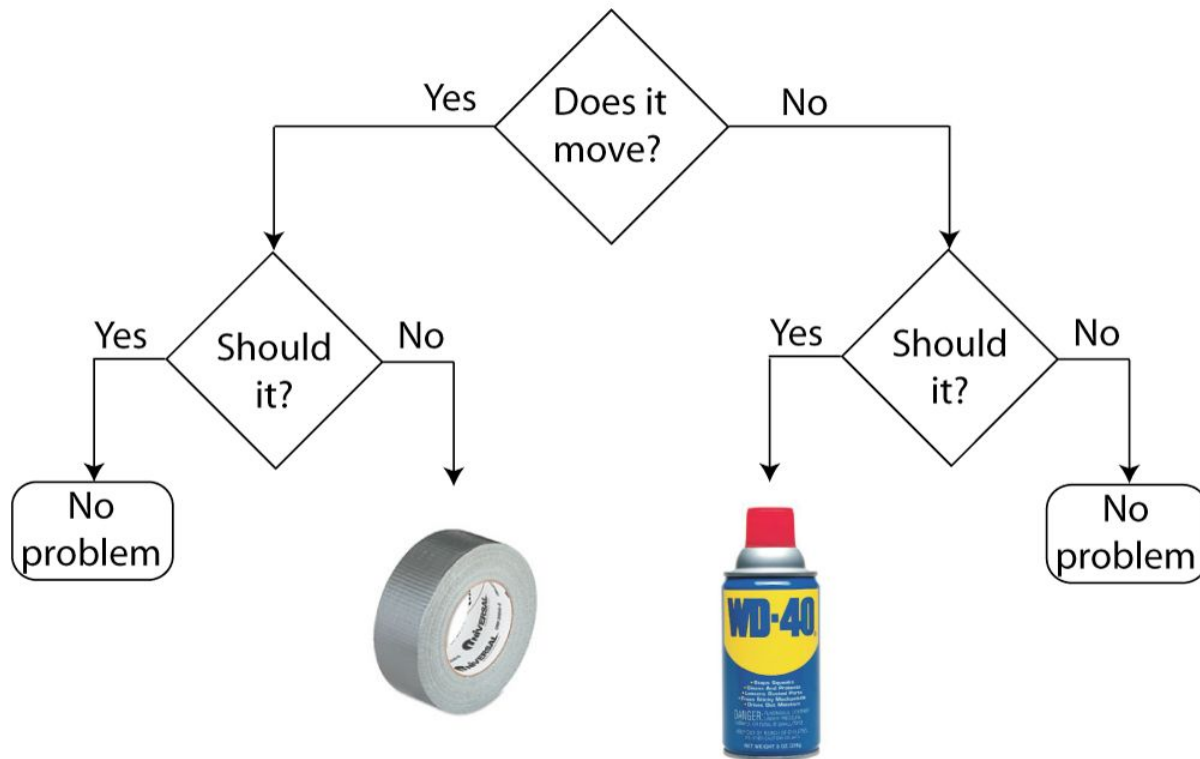
- » Współcześnie, wartości całkowito-liczbowe są reprezentowane w tzw. **“kodzie uzupełnień do dwóch”** (w skrócie **U2**)
- » W ramach pracy własnej:
 - poczytaj o **U2**
 - przekonwertuj “int” na bity, sprawdź czy jest w **U2**
 - rozpisz bitowo dowolną liczbę float a następnie sprawdź czy reprezentacja w C++ wygląda tak samo

Co to jest algorytm?

Algorytm




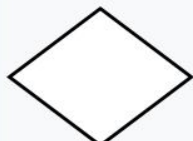
- » **Jasno zdefiniowane czynności konieczne do wykonania pewnego rodzaju zadania (wiki)**
- » Umożliwia wykonanie obliczeń, przetwarzania danych,
...
- » Może być wyrażony jako:
 - **schemat blokowy**
 - **pseudo-kod**
 - **kod źródłowy w języku programowania**
 - maszyna Turinga
 - język naturalny,
- » Pomimo różnego sposobu zapisu, wynik powinien być taki sam

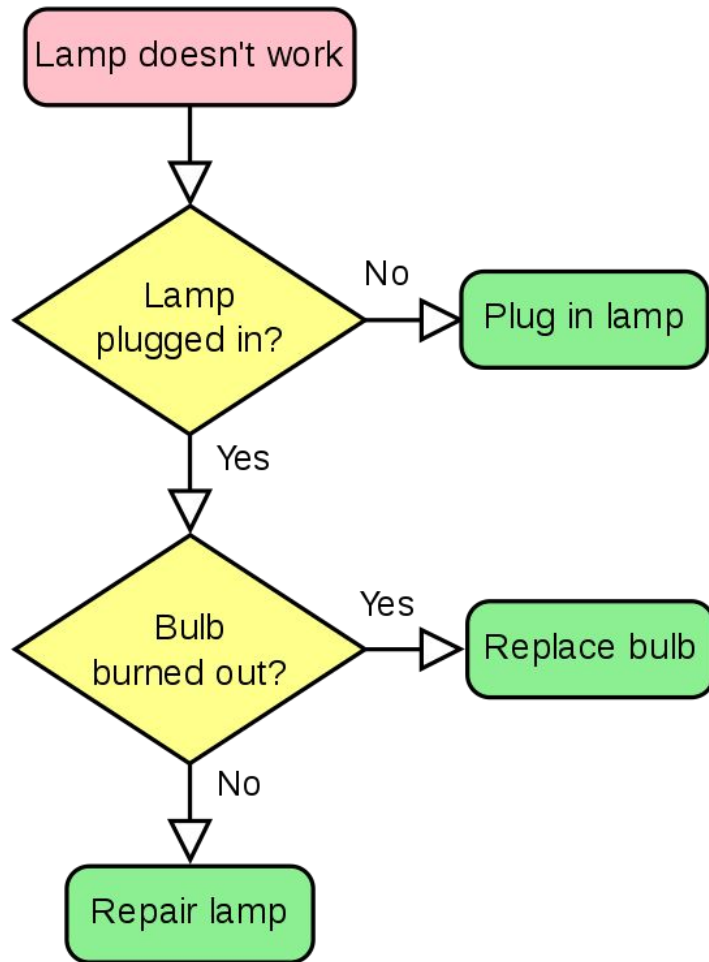
Schemat blokowy



Schemat blokowy

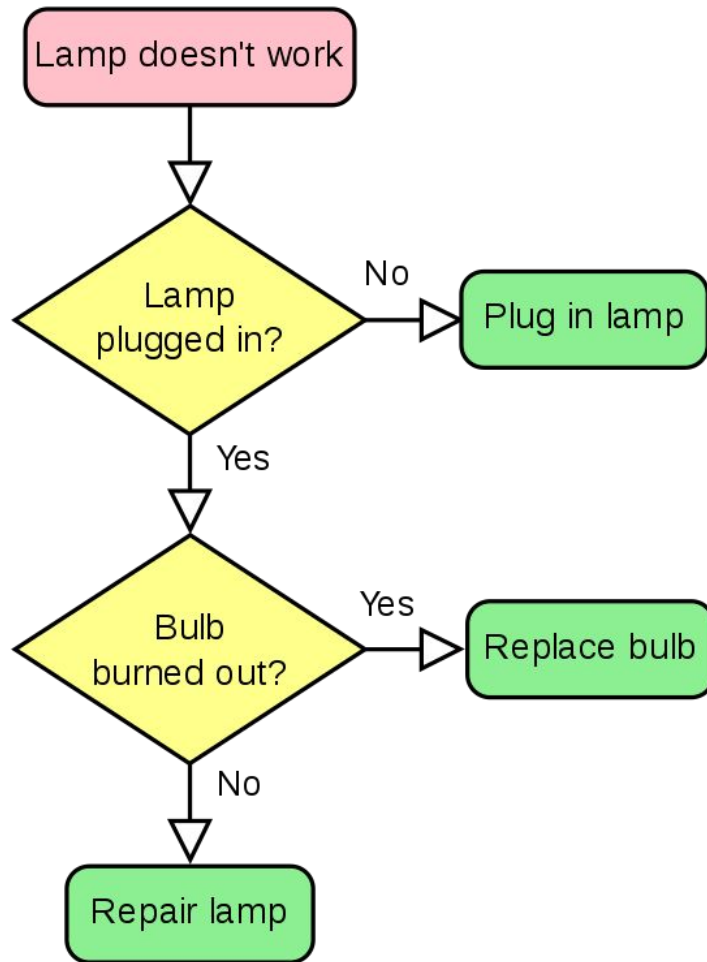
<https://en.wikipedia.org/wiki/Flowchart>

ANSI/ISO Shape	Name
	Flowline (Arrowhead) ^[15]
	Terminal ^[14]
	Process ^[15]
	Decision ^[15]





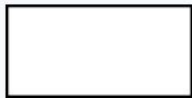
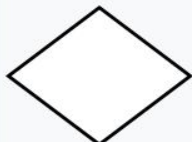
Schemat blokowy

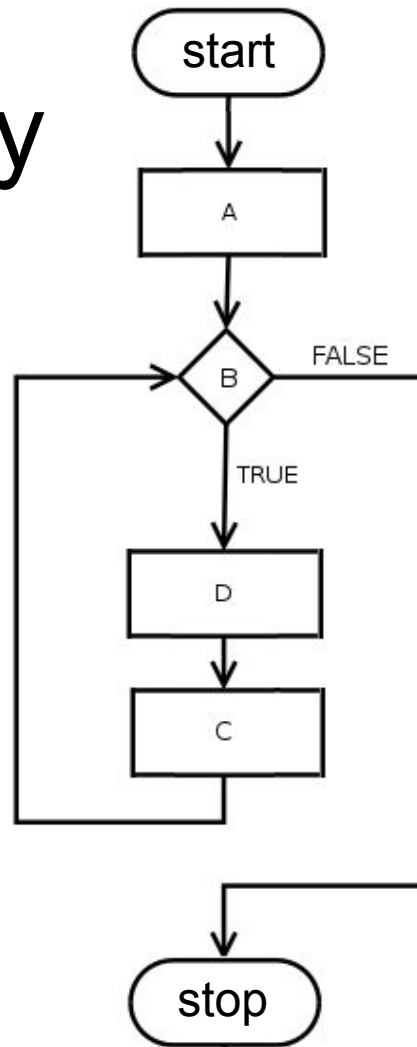
- » Typ diagramu **przedstawiający kolejne czynności** w algorytmie
- » Pokazuje kolejne kroki w postaci figur geometrycznych połączonych strzałkami
- » Ilustruje **sposób rozwiązania danego problemu**
- » Wykorzystywany do analizy, dokumentacji i **projektowania programów komputerowych**



Schemat blokowy



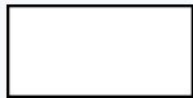
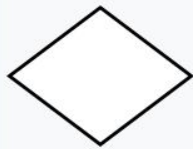
<https://en.wikipedia.org/wiki/Flowchart>

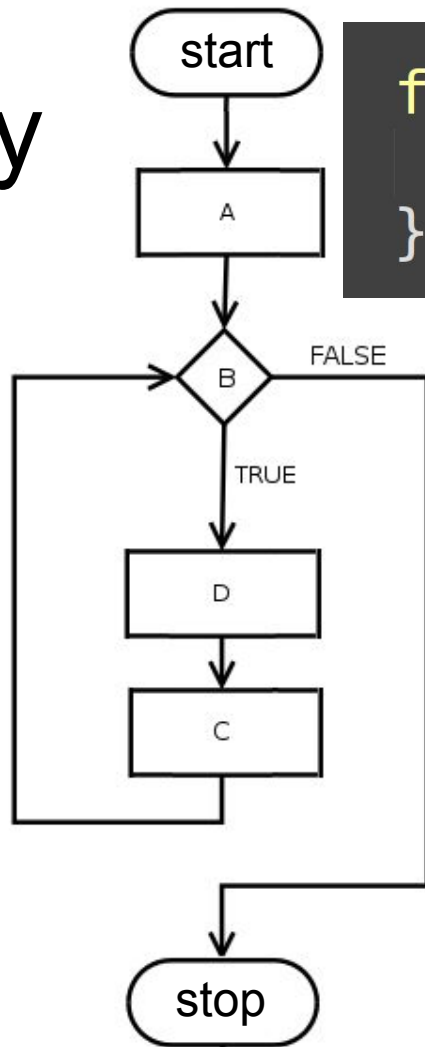
ANSI/ISO Shape	Name
	Flowline (Arrowhead) ^[15]
	Terminal ^[14]
	Process ^[15]
	Decision ^[15]



Schemat blokowy

<https://en.wikipedia.org/wiki/Flowchart>

ANSI/ISO Shape	Name
	Flowline (Arrowhead) ^[15]
	Terminal ^[14]
	Process ^[15]
	Decision ^[15]



```
for(A;B;C){  
    D;  
}
```

Pseudokod

- » Typ wysokopoziomowego opisu, który **reprezentuje algorytm**
- » **Szkielet algorytmu**
- » Nie zawiera detali
- » Tylko najważniejsze koncepcje pozwalające zrozumieć algorytm
- » **Zapis nie jest ustandaryzowany**
- » Nie można skompilować :(

```
Do i = 1 to 100
  set p to true
  If i is divisible by 3
    print "Fizz"
    set p to false
  If i is divisible by 5
    print "Buzz"
    set p to false
  If p
    print i
  print a newline
```

Pseudokod

Pascal style pseudo code

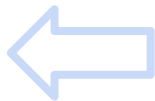
```
procedure fizzbuzz
  For i := 1 to 100 do
    set print_number to true;
    If i is divisible by 3 then
      print "Fizz";
    set print_number to false;
    If i is divisible by 5 then
      print "Buzz";
    set print_number to false;
    If print_number, print i;
    print a newline;
  end
```


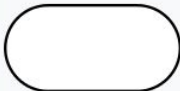

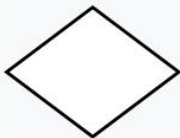
C style pseudo code:

```
void function fizzbuzz {
  for (i = 1; i <= 100; i++) {
    set print_number to true;
    If i is divisible by 3 {
      print "Fizz";
      set print_number to false; }
    If i is divisible by 5 {
      print "Buzz";
      set print_number to false; }
    If print_number, print i;
    print a newline;
  }
}
```

Flow chart Pseudokod

???



ANSI/ISO Shape	Name
	Flowline (Arrowhead) ^[15]
	Terminal ^[14]
	Process ^[15]
	Decision ^[15]

```

Do i = 1 to 100
  set p to true
  If i is divisible by 3
    print "Fizz"
  set p to false
  If i is divisible by 5
    print "Buzz"
  set p to false
  If p
    print i
  print a newline
  
```



quiz

PI03_alg

socrative.com

- login
- student login

Room name:

KWANTAGH



GIT

gałęzie

GIT - gałęzie



- » **Master** == główna gałąź, podstawowa, stabilny kod, często ograniczone prawa zapisu

GIT - gałęzie



- » **Master** == główna gałąź, podstawowa, stabilny kod, często ograniczone prawa zapisu
- » Z każdego miejsca (commit) mogę utworzyć inne, niezależne wersje kodu

GIT - gałęzie



- » **Master** == główna gałąź, podstawowa, stabilny kod, często ograniczone prawa zapisu
- » Z każdego miejsca (commit) mogę utworzyć inne, niezależne wersje kodu
- » Rozwój programu odbywa się zazwyczaj w innych gałęziach (np. **Develop**)

GIT - gałęzie



- » **Master** == główna gałąź, podstawowa, stabilny kod, często ograniczone prawa zapisu
- » Z każdego miejsca (commit) mogę utworzyć inne, niezależne wersje kodu
- » Rozwój programu odbywa się zazwyczaj w innych gałęziach (np. **Develop**)
- » W pewnym momencie może nastąpić **scalenie** (**merge**) gałęzi,

GIT - gałęzie

```
> git clone https://gitlab.com/gr/pro.git  
> cd pro/
```

» Clone kopiuje tylko gałąź Master

GIT - gałęzie

```
> git clone https://gitlab.com/gr/pro.git  
> cd pro/  
> git branch -a
```

- » Clone kopiuje tylko gałąź Master
- » Pokaż **wszystkie** gałęzie: local i remote

GIT - gałęzie

```
> git clone https://gitlab.com/gr/pro.git  
> cd pro/  
> git branch -a
```

- » Clone kopiuje tylko gałąź Master
- » Pokaż wszystkie gałęzie: local i remote

```
git branch -a  
* master  
remotes/origin/AbandonedGUI  
remotes/origin/HEAD -> origin/master  
remotes/origin/master  
remotes/origin/v3beta
```

GIT - gałęzie

```
> git clone https://gitlab.com/gr/pro.git  
> cd pro/  
> git branch -a  
> git checkout v3beta
```

- » Clone kopiuje tylko gałąź Master
- » Pokaż wszystkie gałęzie: local i remote
- » **Przełączanie** gałęzi (ściąga na dysk!)

```
git branch -a  
master  
* v3beta  
remotes/origin/AbandonedGUI  
remotes/origin/HEAD -> origin/master  
remotes/origin/master  
remotes/origin/v3beta
```

GIT - gałęzie

```
> git clone https://gitlab.com/gr/pro.git  
> cd pro/  
> git branch -a  
> git checkout v3beta  
> git checkout master
```

- » Clone kopiuje tylko gałąź Master
- » Pokaż wszystkie gałęzie: local i remote
- » Przełączanie gałęzi
- » Przełączenie gałęzi (lokalnie)

```
git branch -a  
* master  
v3beta  
remotes/origin/AbandonedGUI  
remotes/origin/HEAD -> origin/master  
remotes/origin/master  
remotes/origin/v3beta
```


GIT - gałęzie

```
> git clone https://gitlab.com/gr/pro.git
> cd pro/
> git branch -a
> git checkout v3beta
> git checkout master
> git branch -d v3beta
```

```
git branch -a
* master
remotes/origin/AbandonedGUI
remotes/origin/HEAD -> origin/master
remotes/origin/master
remotes/origin/v3beta
```

- » Clone kopiuje tylko gałąź Master
- » Pokaż wszystkie gałęzie: local i remote
- » Przełączanie gałęzi
- » Przełączenie gałęzi (lokalnie)
- » **Skasowanie** gałęzi lokalnie

GIT - gałęzie

```
> git clone https://gitlab.com/gr/pro.git  
> cd pro/  
> git branch -a  
> git checkout v3beta  
> git checkout master  
> git branch -d v3beta  
> git push origin --delete v3beta
```

- » Clone kopiuje tylko gałąź Master
- » Pokaż wszystkie gałęzie: local i remote
- » Przełączanie gałęzi
- » Przełączenie gałęzi (lokalnie)
- » Skasowanie gałęzi lokalnie
- » Skasowanie gałęzi **na serwerze**

GIT - gałęzie

```
> git clone https://gitlab.com/gr/pro.git
> cd pro/
> git branch -a
> git checkout v3beta
> git checkout master
> git branch -d v3beta
> git push origin --delete v3beta
> git checkout -b feature1
```

- » Clone kopiuje tylko gałąź Master
- » Pokaż wszystkie gałęzie: local i remote
- » Przełączanie gałęzi
- » Przełączenie gałęzi (lokalnie)
- » Skasowanie gałęzi lokalnie
- » Skasowanie gałęzi na serwerze
- » Utworzenie **nowej gałęzi** (lokalnie) o nazwie **feature1**

GIT - gałęzie

```
> git clone https://gitlab.com/gr/pro.git
> cd pro/
> git branch -a
> git checkout v3beta
> git checkout master
> git branch -d v3beta
> git push origin --delete v3beta
> git checkout -b feature1
> git push -u origin feature1
```

- » Clone kopiuje tylko gałąź Master
- » Pokaż wszystkie gałęzie: local i remote
- » Przełączanie gałęzi
- » Przełączenie gałęzi (lokalnie)
- » Skasowanie gałęzi lokalnie
- » Skasowanie gałęzi na serwerze
- » Utworzenie nowej gałęzi (lokalnie) o nazwie feature1
- » **Wysłanie** gałęzi na **serwer**
 - główny serwer to “origin”
 - gałąź umieszcza się tylko raz na serwerze

GIT - gałęzie

```
> git clone https://gitlab.com/gr/pro.git
> cd pro/
> git branch -a
> git checkout v3beta
> git checkout master
> git branch -d v3beta
> git push origin --delete v3beta
> git checkout -b feature1
> git push -u origin feature1
> # change something
> git commit -am "hot fix"
> git push
```

- » Clone kopiuje tylko gałąź Master
- » Pokaż wszystkie gałęzie: local i remote
- » Przełączanie gałęzi
- » Przełączenie gałęzi (lokalnie)
- » Skasowanie gałęzi lokalnie
- » Skasowanie gałęzi na serwerze
- » Utworzenie nowej gałęzi (lokalnie) o nazwie feature1
- » Wysłanie gałęzi na serwer
 - główny serwer to "origin"
 - gałąź umieszcza się tylko raz na serwerze
 - później już tylko commit i push (nie trzeba wskazywać -u origin)

Dziękuję