Methodology and Programming Techniques

Department of Telecommunications, **E&T**

Jarosław Bułat, PhD

kwant@agh.edu.pl

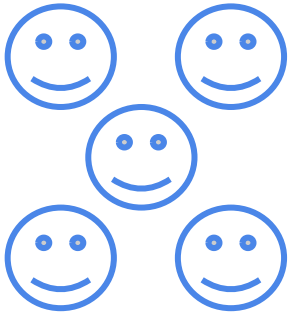
Outline

- Numeral system, positional notation,
- Logical operators, bitwise operators
- Type conversions
- Text in C++
- Floating-point type
- The algorithm, flow-chart, pseudocode
- GIT - branches

representation of numbers

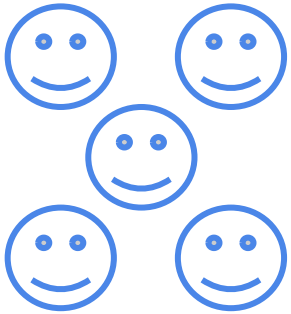
positional notation, conversion

Positional notation



» Write down in notation:

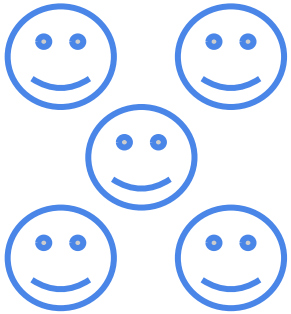
Positional notation



I I I I I

» Numeral system:
  – unary numeral system

Positional notation

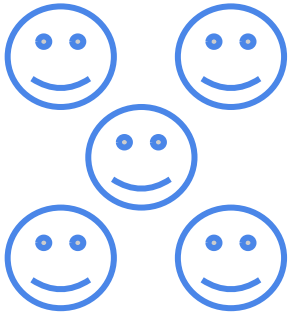


I I I I I

5

» Numeral system:
  – unary numeral system
  – decimal

Positional notation



I I I I I

5

12

101

» Numeral system:
  - unary numeral system
  - decimal
  - ternary
  - binary

Positional notation

123 = 1x100 + 2x10 + 3

» Rules of positional notation

Positional notation

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

» Rules of positional notation

Positional notation

$$123 = 1x100 + 2x10 + 3$$
$$= 1x100 + 2x10 + 3x1$$
$$= 1x10^2 + 2x10^1 + 3x10^0$$

- Rules of positional notation
- “Power” is the position number counting from the right hand side (LSB)

Positional notation

$$123 = 1 \times 100 + 2 \times 10 + 3$$
$$= 1 \times 100 + 2 \times 10 + 3 \times 1$$
$$= 1 \times 10^2 + 2 \times 10^1 + 3 \times 10^0$$

- Rules of positional notation
- “Power” is the position number counting from the right hand side (LSB)

Positional notation

$$123 = 1x100 + 2x10 + 3$$
$$= 1x100 + 2x10 + 3x1$$
$$= 1x10^2 + 2x10^1 + 3x10^0$$

$$101 = 1x4 + 0x2 + 1x1$$
$$= 1x2^2 + 0x2^1 + 1x2^0$$

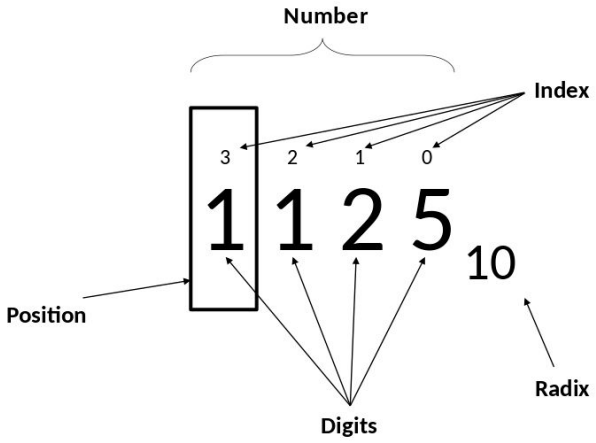
- Rules of positional notation
- “Power” is the position number counting from the right hand side (LSB)
- Radix base does not have to be 10
- It may be different eg. 2

Positional notation

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

» General relation for positional notation

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



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

Positional notation

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

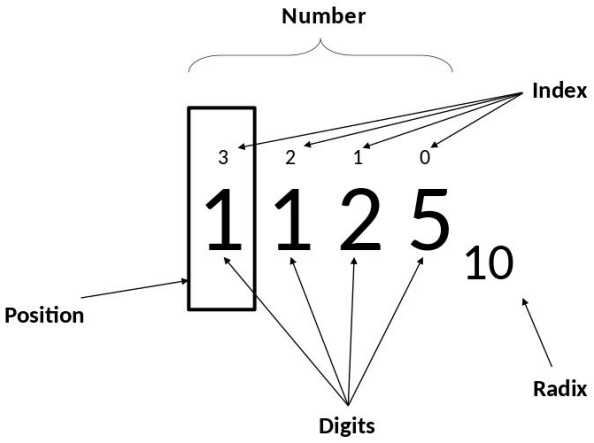
» General relation for positional notation

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

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

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

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



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

Conversion between numerical systems

bin-dec-hex conversion

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}

bin-dec-hex conversion

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: “used by computers”
- » dec/oct/hex: communication with humans
  - – printing
  - – on screen printing
  - – input from keyboards
  - – *.txt files
- » dec: **m2m** communication
  - – html
  - – json
  - – txt

bin-dec-hex conversion

0	1	0	0	0	1	1	0	$= 01000110_2$	<- 8 bits
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0		
128	64	32	16	8	4	2	1		<- weight

bin-dec-hex conversion

0	1	0	0	0	1	1	0	$= 01000110_2$	<- 8 bits
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0		
128	64	32	16	8	4	2	1		<- weight
0	64	0	0	0	4	2	0	$= 70_{10}$	<- results

bin-dec-hex conversion

0	1	0	0	0	1	1	0	$= 01000110_2$	<- 8 bits
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0		
128	64	32	16	8	4	2	1		<- weight
0	64	0	0	0	4	2	0	$= 70_{10}$	<- results

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

where:
N - result, R - radix base, D - bit value, i - bit number

bin-dec-hex conversion

0	1	0	0	**1**	1	1	0	$= 01000110_2$	<- 8 bits
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0		
128	64	32	16	8	4	2	1		<- dec weight
200	100	40	20	10	4	2	1		<- oct weight
80	40	80	10	8	4	2	1		<- hex weight
0	64	0	0	8	4	2	0	$= 78_{10}$	<- dec result
0	100	0	0	10	4	2	0	$= 116_8$	<- oct wynik
0	40	0	0	8	4	2	0	$= 4E_{16}$	<- dec wynik

bin-dec-hex conversion

0	1	0	0	1	1	1	0	$= 01000110_2$	<- 8 bits
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0		
128	64	32	16	8	4	2	1		<- dec weight
200	100	40	20	10	4	2	1		<- oct weight
80	40	80	10	8	4	2	1		<- hex weight
0	64	0	0	8	4	2	0	$= 78_{10}$	<- dec result
0	1	0	0	1	4	2	0	$= \{1\ 1\ 6\}_8$	<- dec result
0	4	0	0	8	4	2	0	$= \{4\ E\}_{16}$	<- dec result

words

- » {0,1} - bit (pl. **bit**)
- » **8 bits - byte** (pl. **bajt**)
- » 16 bits (2 bytes) - word (pl. **słowo**)
- » 32 bits (4 bytes) - word
- » 64 bits (8 bytes) - word

words

» {0,1}	- bit	(pl. **bit**)
» **8 bits**	**- byte**	(pl. **bajt**)
» 16 bits (2 bytes)	- word	(pl. **słowo**)
» 32 bits (4 bytes)	- word	
» 64 bits (8 bytes)	- word	

» 1 byte == 8 bits == 1 character (char, digit, special char, etc…)

words

» {0,1}	- bit	(pl. **bit**)
» **8 bits**	**- byte**	(pl. **bajt**)
» 16 bits (2 bytes)	- word	(pl. **słowo**)
» 32 bits (4 bytes)	- word	
» 64 bits (8 bytes)	- word	

» 1 byte == 8 bits == 1 character (char, digit, special char, etc…)

» **ISO/IEC 2382-1:1993** in 2^x convention: values 0-255 (256 states)

prefixes

Multiples of bits V · T · E

Decimal				Binary					
Value		SI		Value		IEC		JEDEC	
1000	10^3	kbit	kilobit	1024	2^{10}	Kibit	kibibit	Kbit	kilobit
1000^2	10^6	Mbit	megabit	1024^2	2^{20}	Mibit	mebibit	Mbit	megabit
1000^3	10^9	Gbit	gigabit	1024^3	2^{30}	Gibit	gibibit	Gbit	gigabit
1000^4	10^{12}	Tbit	terabit	1024^4	2^{40}	Tibit	tebibit		-
1000^5	10^{15}	Pbit	petabit	1024^5	2^{50}	Pibit	pebibit		-
1000^6	10^{18}	Ebit	exabit	1024^6	2^{60}	Eibit	exbibit		-
1000^7	10^{21}	Zbit	zettabit	1024^7	2^{70}	Zibit	zebibit		-
1000^8	10^{24}	Ybit	yottabit	1024^8	2^{80}	Yibit	yobibit		-

See also: Nibble · Byte · Orders of magnitude of data

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

How to compare in C++?

by means of operator

Comparison operators

```
#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;
}
```

- `==` equal to
- `!=` not equal to
- `>` greater than
- `<` **less than**
- `>=` Greater than or equal to
- `<=` Less than or equal to

» return a Boolean value
» because of compatibility with C:
  – false: 0
  – true: everything except 0

Comparison operators

```
#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;
}
```

must be in parentheses, otherwise the operator's low priority error

==	equal to
!=	not equal to
>	greater than
<	**less than**
>=	Greater than or equal to
<=	Less than or equal to

- » return a Boolean value
- » because of compatibility with C:
  - – false: 0
  - – true: everything except 0

Logical operators

```
// 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 !!!
```

||	"or" logical alternative
&&	"and" logical conjunction
!	"not" logical negation

» **x || y**
logical sum of x and y, the result is:
zero where x and y are zero
nonzero otherwise

» non-zero is other than 0,
can be 1, -1, 2342342, etc ...

Bitwise operators

```
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;
```

|	bitwise OR
&	bitwise AND
^	bitwise exlusive OR
<<	shift bits left
>>	shift bits right
~	bits inversion

- works with integers
- works on all bits at a time (bitwise)
- do not confuse **& with &&**...

Casting operator

```
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++ is a statically typed but it has a limited automatic conversion
- **Preferred explicit conversion**
- Possible misinterpretation:
  - compiler do not know data
  - runtime does not check it
- Recommended casting in C++:

dynamic_cast <new_type> (expression)

reinterpret_cast <new_type> (expression)

static_cast <new_type> (expression)

const_cast <new_type> (expression)

- Better control
- More options

sizeof() operator

```
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 (variable_name)
- sizeof (type)
- it is the operator, not a function!
- gives the size of the variable type
- size_t is* unsigned int
  - suggestion that the variable express size in bytes
  - it is non-negative
  - is a natural number**
  - often use for sequence, position, size of the array, ...
- operator typeid (...)
  - type of variable***
  - specific use

Operators priority (order)

- The order of calculation of the operands is not guaranteed due to the ability to optimize code, eg. out-of-order execution
- the logic operators &&, || may omit part of the calculation (if it does not affect the result)
  - **a + 4 || ++ b**
- use operands that affect the value of the other is not standardized, results are not defined
  - **int i = 1;**
  - **array[i] = ++i;** (not sure whether the array[1] and array[2])
- recommendation: **do not use complex calculations in one expression**, separate it into a few lines

Text in C/C++

+strange characters... ąęśćłóżź

Text in C++

```
char c = 'a';                    // single quote
char d = 65;                     // single character (sign, symbol, etc…)
```

» char is a short integer
  – possible values: -128 ... 0 ... 127
  – character in ASCII
  – 'a' is the ASCII character code

Text in C++

```
char c = 'a';                    // single quote
char d = 65;                     // single character (sign, symbol, etc…)


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


// source: http://www.stroustrup.com/bs_faq2.html#char
```

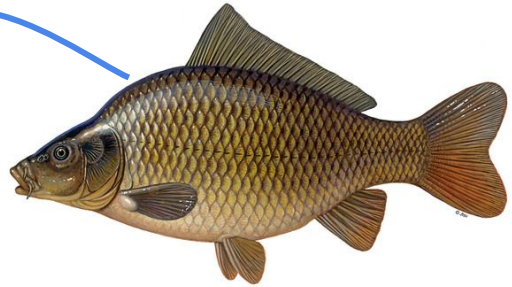
- char is a short integer
  - possible values: -128 ... 0 ... 127
  - character in ASCII
  - 'a' is the ASCII character code
- can be incremented
- pay attention to the automatic type conversion!!!
- special characters

Text in C++

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


// 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 is a short integer
  - – possible values: -128 ... 0 ... 127
  - – character in ASCII
  - – 'a' is the ASCII character code
- » can be incremented
- » pay attention to the automatic type conversion!!!
- » special characters

Text in 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 is a short integer
  - possible values: -128 ... 0 ... 127
  - character in ASCII
  - 'a' is the ASCII character code
- can be incremented
- pay attention to the automatic type conversion!!!
- special characters

ASCII

- » **A**merican **S**tandard for **C**ode **I**nformation Interchange
- » 128 codes:
  - – characters (large/ small)
  - – Digits
  - – additional characters
  - – special characters
- » '\n'== 0xA (10) unix
- » '\n '== 0xC 0xA Windows
- » the remaining 128 combinations:
  - – Symbols
  - – diacritic characters
- » EN: 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 TRANSM.	5 ^E ENQ ENQUIRY	6 ^F ACK ACKNOWL-EDGE	7 ^G BEL BELL	8 ^H BS BACKSP.	9 ^I HT CHARACT. TAB'TION	10 ^J LF LINE FEED	11 ^K VT LINE TAB'TION	12 ^L FF FORM FEED	13 ^M CR CARRIAGE RETURN	14 ^N SO SHIFT OUT	15 ^O SI SHIFT IN
1	16 ^P DLE DATALINK ESCAPE	17 ^Q DC1 DEVICE CONTROL1	18 ^R DC2 DEVICE CONTROL2	19 ^S DC3 DEVICE CONTROL3	20 ^T DC4 DEVICE CONTROL4	21 ^U NAK NEG.ACK-NOWLEDGE	22 ^V SYN SYNCHR. IDLE	23 ^W ETB END OF TRANS.	24 ^X CAN CANCEL	25 ^Y EM END OF MEDIUM	26 ^Z SUB SUBS-TITUTE	27 ^[ESC ESCAPE	28 ^\ FS INFO. SEP. 4	29 ^] GS INFO. SEP. 3	30 ^^ RS INFO. SEP. 2	31 ^_ US INFO. SEP. 1
2	SPACE	! excl ! EXCLAM. MARK	" quot " QUOT. MARK	# num # NUMBER SIGN	$ dollar $ DOLLAR SIGN	% percnt % PERCENT SIGN	& amp & AMPER-SAND	' apos ' APOS-TROPHE	(lpar (LEFT PAREN.	) rpar) RIGHT PAREN.	* ast * ASTERISK	+ plus + PLUS SIGN	, comma , COMMA	- - HYPHEN-MINUS	. period . FULL STOP	/ sol / SOLIDUS
3	0 0 DIGIT ZERO	1 1 DIGIT ONE	2 2 DIGIT TWO	3 3 DIGIT THREE	4 4 DIGIT FOUR	5 5 DIGIT FIVE	6 6 DIGIT SIX	7 7 DIGIT SEVEN	8 8 DIGIT EIGHT	9 9 DIGIT NINE	: colon : COLON	; semi ; SEMI-COLON	< lt < LS.-THAN SIGN	= equals = EQUALS SIGN	> gt > GR.-THAN SIGN	? quest ? QUEST-ION MARK
4	@ commat @ COMM'IAL AT	A A	B B	C C	D D	E E	F F	G G	H H	I I	J J	K K	L L	M M	N N	O O
5	P P	Q Q	R R	S S	T T	U U	V V	W W	X X	Y Y	Z Z	[lsqb [LEFT SQ. BRACKET	\ bsol \ REVERSE SOLIDUS	] rsqb] RT. SQR. BRACKET	^ hat ^ CIRCUM'X ACCENT	_ lowbar _ LOW LINE
6	` grave ` GRAVE ACCENT	a a	b b	c c	d d	e e	f f	g g	h h	i i	j j	k k	l l	m m	n n	o o
7	p p	q q	r r	s s	t t	u u	v v	w w	x x	y y	z z	{ { L. CURLY BRACKET		| VERTICAL LINE	} } R. CURLY BRACKET	~ ~ TILDE	127 ^? DEL DELETE

ASCII code table including entity references, control codes and Unicode names (1.1) © Tom Gibara July 2014

Unicode

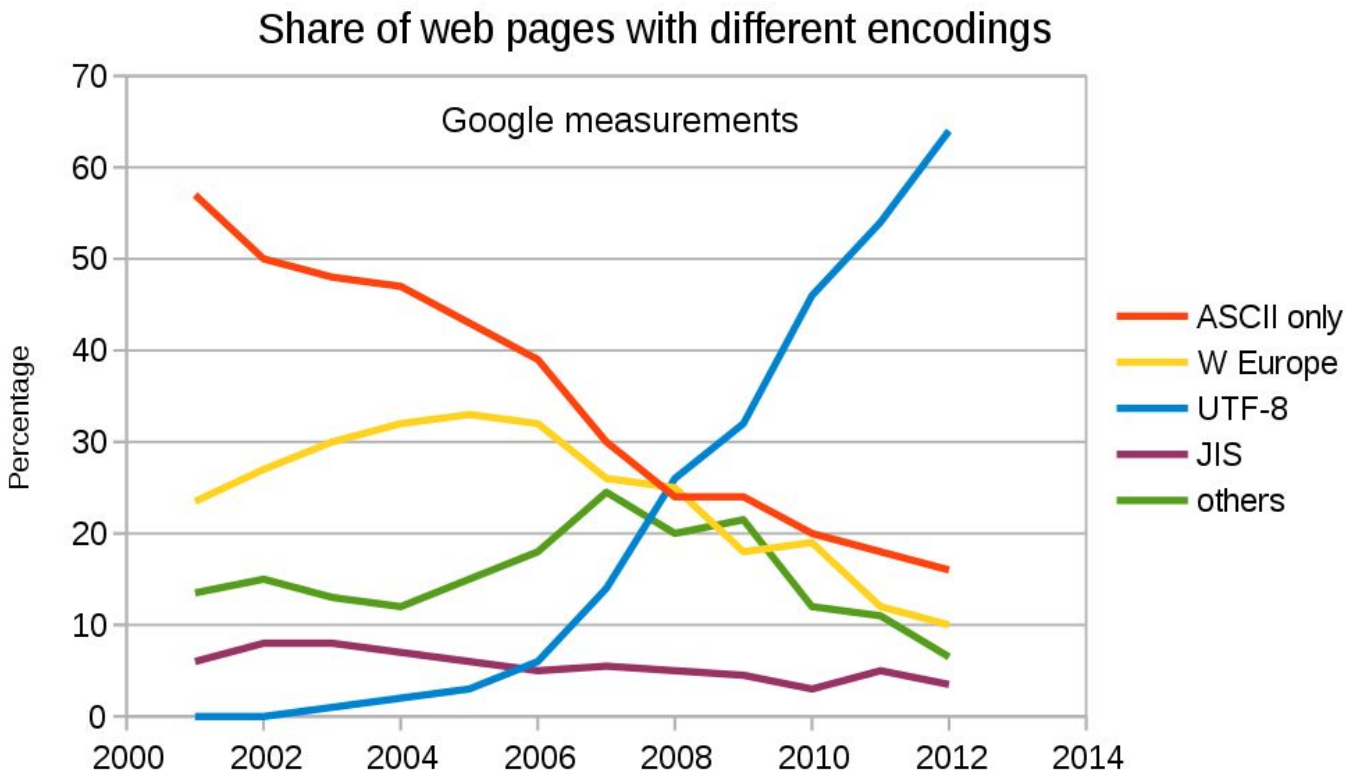
- Industry standard for encoding, representations and handling of text
- **UTF-8** - character encoding
- **Single character consists of 1-4 bytes**
- **Compatible with ASCII**
- Polish character on 2 bytes
- 1 112 064 unique characters

- 1 byte: **0xxxxxxx**, where „x” is a bit of information
- 2 bytes: **110xxxxx 10xxxxxx**
- 3 bytes: **1110xxxx 10xxxxxx 10xxxxxx**
- 4 bytes: **11110xxx 10xxxxxx 10xxxxxx 10xxxxxx**

- text length in characters! = text length in bytes
- incompatible with C
- difficulties in sorting taking account of national characters

UTF-32/UCS-4
UTF-16
UTF-8
UTF-7
UCS-2
UTF-9 UTF-18
UTF-EBCDIC
UTF-6
UTF-5

UTF-8

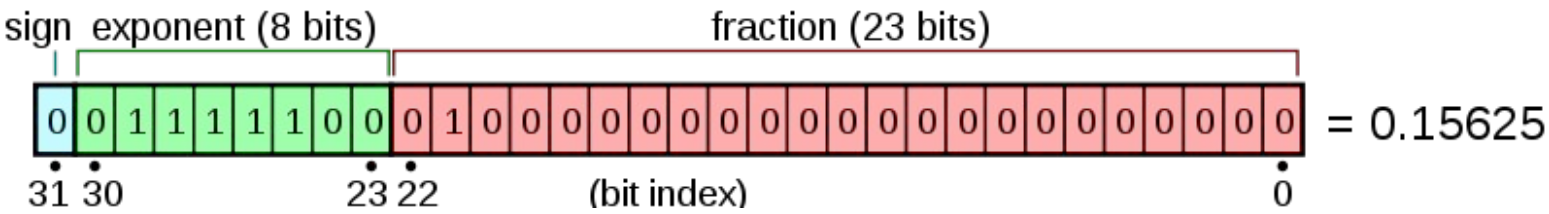


https://en.wikipedia.org/wiki/UTF-8

Let’s do the math

0.7 == 0.699999988079

IEEE-754 (float)



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

- 1 bit of sign, 8 bits of exponent, 23 bits of fractional = **32 bity**
- Accuracy: 7-8 digits
- Range: 10^{-45}...10^{38}
- Special bit values for NaN, zero, inf, -inf, etc...
- **Multiplication: z = x * y: 9 bits of exponent, 46 bits of fractional**
- **Writing result back to float type, need simplification (round off)**

https://pl.wikipedia.org/wiki/Liczba_zmiennoprzecinkowa

IEEE-754 (float)

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

int main(){

        float a = 9.2;
        cout << std::setprecision(10);
        cout << a << "\n";

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

- **Numerical calculations**
- Errors accumulate with complex calculations
- Order of operations matters
- Floating point representation is quantized
- **After each mathematical operation result is rounded!**

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
}
```

- **Numerical calculations**
- Errors accumulate with complex calculations
- Order of operations matters
- Floating point representation is quantized
- **After each mathematical operation result is rounded!**

ToDo
(for the best of the best ;)

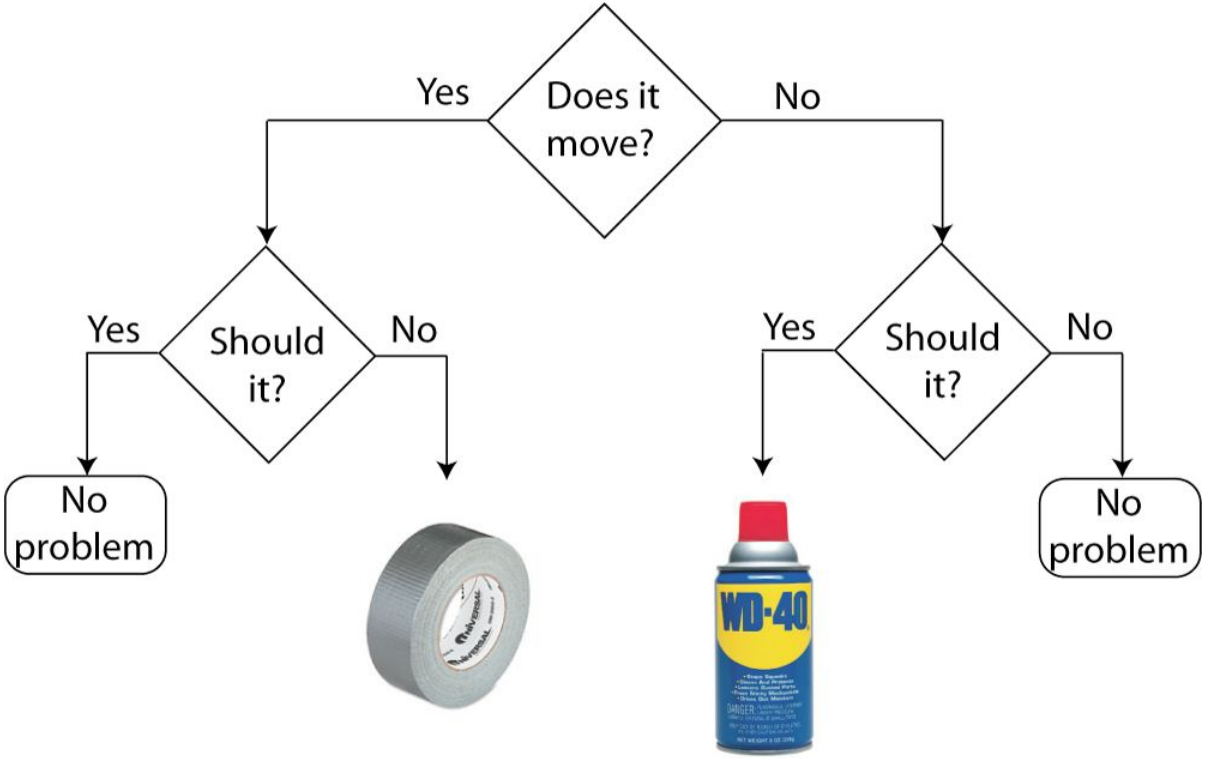
- Today, almost all CPU making use of **Two's complement** (U2) for integer representation
- As part of your (home) work:
  - read about U2
  - convert 'int' to bits, check whether int uses U2 code
  - write in bit level some floating-point number and check if it corresponds to C++ implementation of float type

What is an **algorithm**?

Algorithm

- **Specification of how to solve a class of problems** (wiki)
- Can perform calculation and data processing
- Expressed as a:
  - **flow chart**
  - **pseudocode**
  - **programming language**
  - Turing machine
  - natural language, ….
- Despite of different way of describing the algorithm, the result should be the same

Flow chart

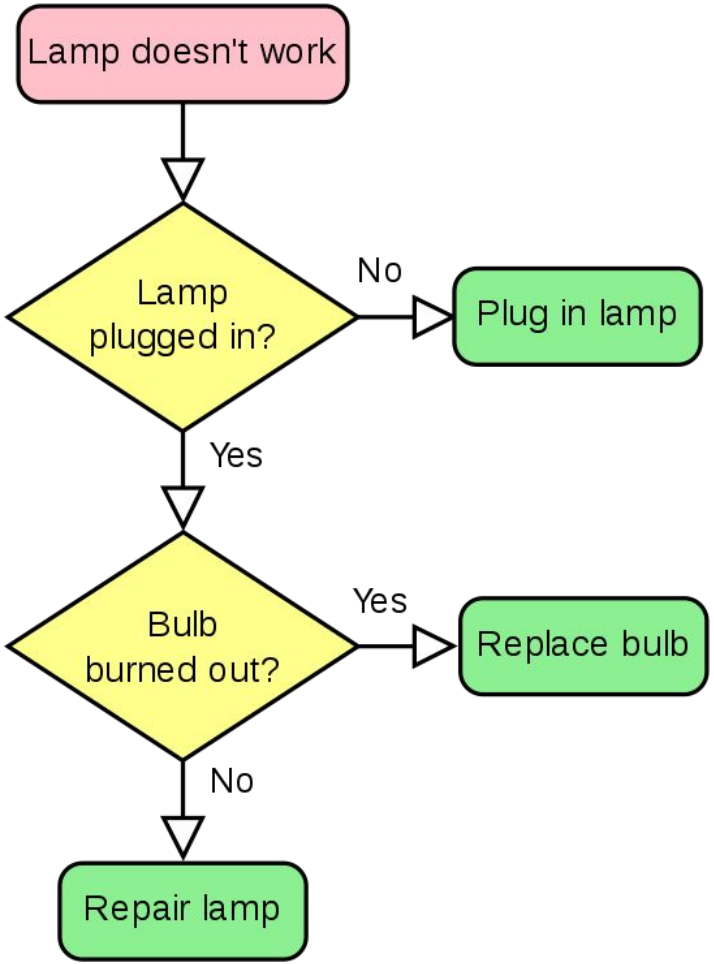


https://www.laserfiche.com/ecmblog/10-funny-flowcharts-to-beat-march-madness-fury/

Flow chart

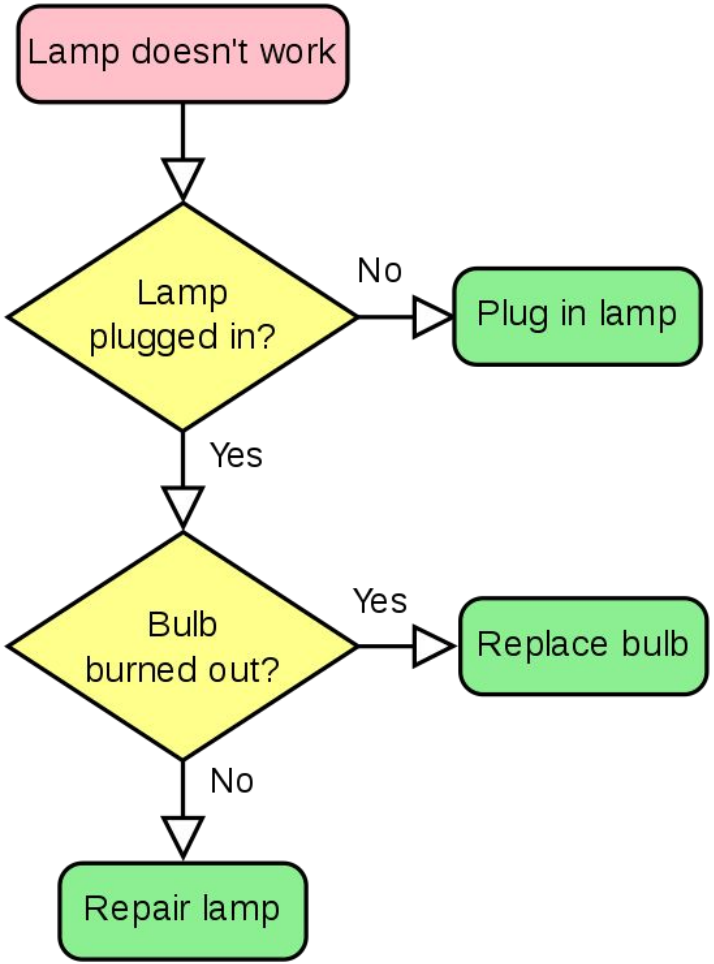
ANSI/ISO Shape	Name
	Flowline (Arrowhead)[15]
	Terminal[14]
	Process[15]
	Decision[15]

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



Flow chart

- Type of diagram that **represents an algorithm**, workflow or process
- Diagram that **showing the steps** of boxes of various kinds
- Illustrate **solution to a given problem**
- Used for analyzing, documenting and **designing computer programs**

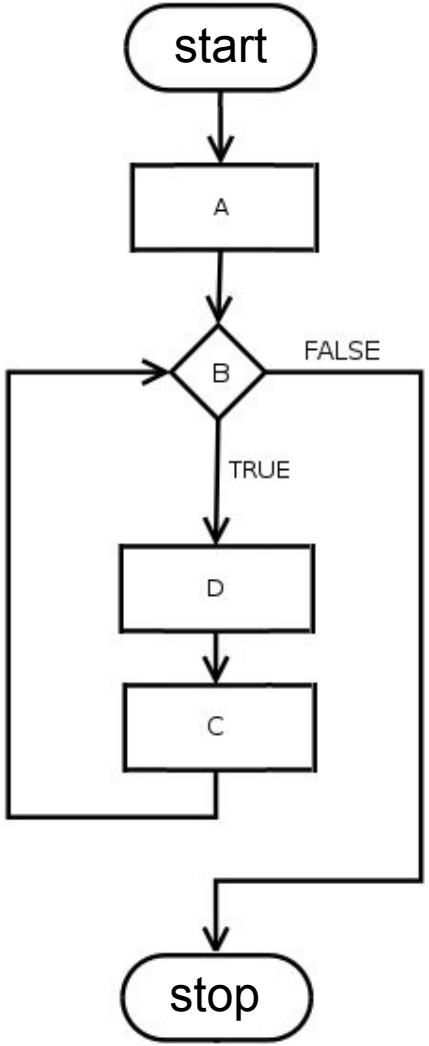


https://en.wikipedia.org/wiki/Flowchart

Flow chart

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

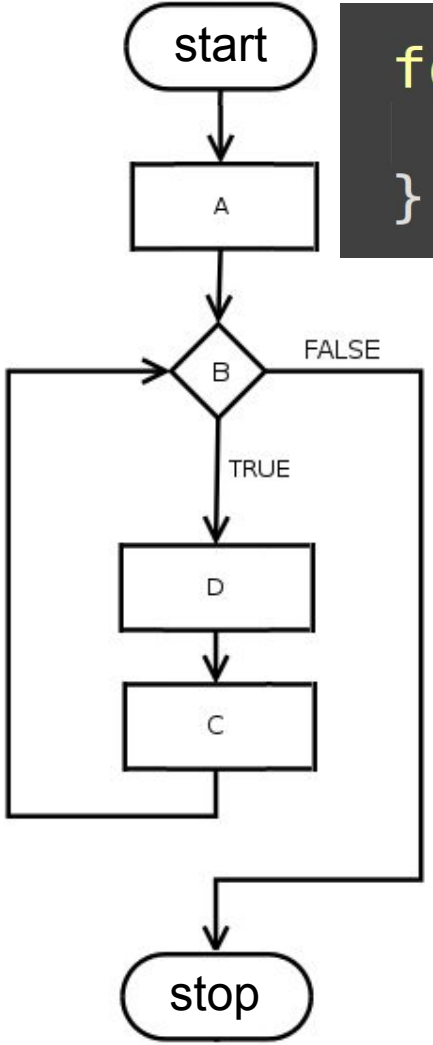
https://en.wikipedia.org/wiki/Flowchart



Flow chart

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

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




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

Pseudocode

- Type of high-level description that **represents an algorithm**
- It is **skeleton of algorithm**
- Do not contain details
- Only essential thing for understanding of the algorithm
- **Not standardized**, no common syntax
- Does not compile

```
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
```

Pseudocode

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 ⇦ Pseudocode

???

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
```

GIT

branches

GIT - branches

» **Master** == main branch, basic branch, stable code, often limited write permission

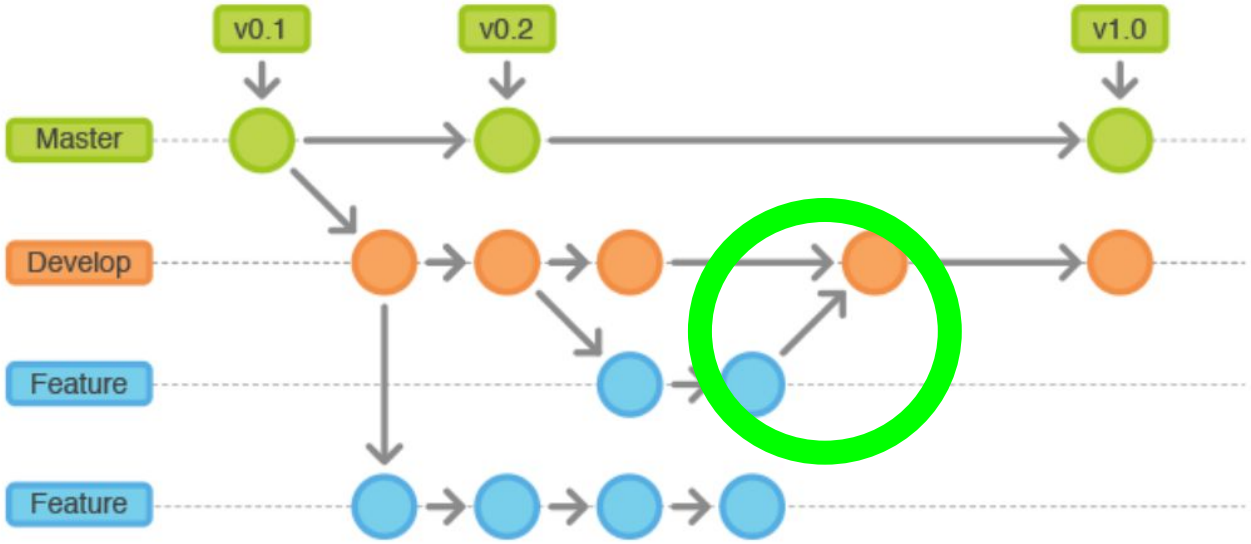
GIT - branches

» **Master** == main branch, basic branch, stable code, often limited write permission
» From every commit one can create (start) another, independent, different code version

GIT - branches

- **Master** == main branch, basic branch, stable code, often limited write permission
- From every commit one can create (start) another, independent, different code version
- Program development usually takes place in other branches (eg. Develop, Feature)

GIT - branches



- **Master** == main branch, basic branch, stable code, often limited write permission
- From every commit one can create (start) another, independent, different code version
- Program development usually takes place in other branches (eg. Develop, Feature)
- At some point merge can take place, means integration of two branches

GIT - branches

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

» Clone makes copy of Master only

GIT - branches

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

- Clone makes copy of Master only
- Show all branches: local and remote

GIT - branches

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

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

- Clone makes copy of Master only
- Show all branches: local and remote

GIT - branches

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

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

- Clone makes copy of Master only
- Show all branches: local and remote
- Switch branch

GIT - branches

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

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

- Clone makes copy of Master only
- Show all branches: local and remote
- Switch branch
- Switch branch (locally)

GIT - branches

```
> 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 makes copy of Master only
- Show all branches: local and remote
- Switch branch
- Switch branch (locally)
- Delete local branch

GIT - branches

```
> 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 makes copy of Master only
- Show all branches: local and remote
- Switch branch
- Switch branch (locally)
- Delete local branch
- Delete branch from server

GIT - branches

```
> 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 makes copy of Master only
- Show all branches: local and remote
- Switch branch
- Switch branch (locally)
- Delete local branch
- Delete branch from server
- Create a new branch (locally) of the name feature1

GIT - branches

```
> 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 makes copy of Master only
- Show all branches: local and remote
- Switch branch
- Switch branch (locally)
- Delete local branch
- Delete branch from server
- Create a new branch (locally) of the name feature1
- Push branch to the server
  - the main server is “origin”
  - only first time

GIT - branches

```
> 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 makes copy of Master only
- Show all branches: local and remote
- Switch branch
- Switch branch (locally)
- Delete local branch
- Delete branch from server
- Create a new branch (locally) of the name feature1
- Push branch to the server
  - the main server is “origin”
  - only first time
  - then only commit and push (not need to point -u origin)

Thank you!