

# Medvědův průvodce po Céčku

Martin Mareš  
mj@ucw.cz

1972 Kernighan & Ritchie  
K&R

1980x ANSI, ISO C89

Katedra Aplikované Matematiky  
MFF UK Praha

dialekty GCC

C++

2019

1990x

C99

↓

C11

↓

C17

panětoný  
model

# Typový systém: Základní typy

## Základní typy: (norma, GCC/amd64)

### • Celočíselné typy:

- `_Bool` (bool) *1 bit*
- `char`, (un)signed char *8 bitů, ∇ znak*
- (unsigned) short int *≥ 16 bitů*
- (unsigned) int *≥ 16 bitů*
- (unsigned) long int *≥ 32 bitů*
- (unsigned) long long int *≥ 64 bitů*
- bit-fields

### • Floating-point typy:

- float *≥ 6 číslic, ≥ 10<sup>37</sup> 32 b*
- double *≥ 10 číslic, ≥ 10<sup>37</sup> 64 b*
- long double *≥ 10 číslic, ≥ 10<sup>37</sup> 80 b*
- jejich komplexní varianty (`_Complex`)

### • Výčty (enum), chovají se celočíselně

### • Typ void

*norma*  
↓

*GCC  
AMD64*  
↓

*--xyz ...  
\_A...*

*true false  
#include <stdbool.h>*

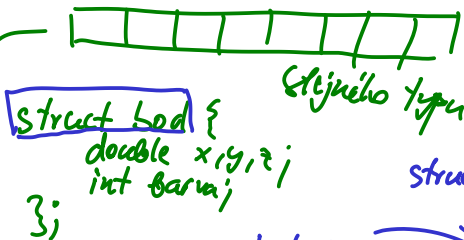
*enum Lycei {  
ABC = 3,  
DEF = 5,  
}*

*neú mocnik 2 !*

# Typový systém: Odvozené typy

## Odvozené typy:

- Pole
- Struktury
- Uniony *union*
- Funkce
- Ukazatele (speciální: `void *`)



`struct bod A;`

`int` `*a;`



`malloc (velikost) -> void *`

## Částečné typy:

- `struct who_knows_what *`
- `int f()` vs. `int f(void)`
- `int x[]`

# Typový systém: Repräsentace hodnot

## Repräsentace:

- Hodnoty jsou složeny z **bytů** (charů), výjimka: bitová pole
- K repräsentaci lze přistupovat pomocí unsigned charů
- Může obsahovat **padding** a vyžadovat **zarovnání**
- Některé repräsentace mohou být nekorektní (**trap**)

↳ - Alignof (typ)

## Celá čísla:

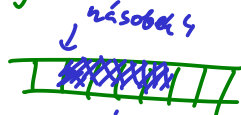
- unsigned čistě binární, může mít padding (třeba paritu)
- unsigned char nemá padding
- signed má znaménkový bit, bity hodnoty a padding
- Hodnota z signed  $\cap$  unsigned vypadá v obou stejně

## Složené typy:

- Struktury: padding mezi prvky, prefixová vlastnost
- Pole: bez paddingu



```
struct {  
    int x: 11;  
    unsigned y: 3; } int
```



```
int i;  
(unsigned char *) &i
```

```
struct {  
    char c;  
    int x;  
};
```

3B }  
padding



```
{ char x; 1  
  int i; 4  
  char y; 1
```

- Integery: 12345, 0xdeadbeef, 0177 *oktalové*
- Typované integery: 12345U, 12345L, 12345ULL
- Floaty: 1.5, 1., .1, 1e20, 1e-20
- Typované floaty: 1.5D, 1.5L *long double* *double* 0.1
- Hexadecimální floaty: 0x1.ffffe10
- Znaky (int): 'x', '\n', '\033', '\x1b'
- Řetězce (const char []): "brekekx\n", "str"\_"ing"
- "Široké" znaky (wchar\_t): L'ž', L"žlučoučký kůň"
- Unicode: L' \u2302', L' \U00002302' (též v identif.)
- Složené literály: (char []){ 1, 2, 3 }  
(pozor na to, kde jsou uloženy)
- Pojmenovaně: (struct point){ .x=1, .y=2 }

*struct point p = {  
}*

# Operátory a jejich priority

- 1 (podvýraz) ident literál `_Generic`
- 2 `[index]` (volání) `.prvek`  $\rightarrow$  `prvek` `++` `--` (postfixové)
- 3 `++` `--` `sizeof` `_Alignof` `&` `*` `+` `-` `~` `!` (typ) (prefixové)

4 `*` `/` `%` modulo

5 `+` `-`  $\leftarrow$  teď pointer + číslo  $\rightarrow$  pointer (unsigned)  
 6 `<<` `>>`  $\leftarrow$  bit posuny pointer - pointer  $\rightarrow$  číslo  
 7 `<` `>` `<=` `>=`  $\leftarrow$  nejvýše 0 kbitu typu - 1

8 `==` `!=`  $\leftarrow$  i na pointerech  
 9 `&`  $\leftarrow$  pozor na floaty!

xND  
xor  
OR

10 `^`  
11 `|`  
12 `&&`  
13 `||`

14 `?:` `p ? x : y`  
 15 `=` `+=` `*=` ... (asoc. zprava)

$x=2, y=3$

$x=y$   
 $x+=y$  ( $x=x+y$ )  
 $x=y=z=0$

Průvodce: `int i;`  
`!i` `!value`

bool. negace  
 bitové negace

`int *p;`  
`*p = *p + 1`  
`x[2]`  
`f(5, 6)`  
`s.x`  
`struct s *q;`  
`(*q).x`  
`q -> x`

v bajtech } `sizeof typ`  
`sizeof (výraz)`



$a = x++$   ~~$x = x++$~~   
 $a = ++x$   
 Hádanka:  $a++++a$

$!42 == 0$   
 $!!x$   
 $D: a[b] \equiv *(a+b) \equiv *(b+a) \equiv b[a] \quad 2[x]$