Principy vyšších programovacích jazyků

- **Každý programovací jazyk z principu slouží na:**
  - **Ukládání informací**
    - Data, bity
    - Kódování dat
  - **Zpracování informací**

- **Ascii** – tabulka hodnot s kterou jsme schopný zapsat písmena binárními hodnotami (jenom 128 znaků)
- V informatice se pro zobrazení více bytů používá 16ková soustava
  - 0–9, a–f
- **Unicode** – univerzální standard který zahrnuje i speciální znaky
  - **UTF** – 0-127 stejné jako ASCII, pak následují znaky novějšího standardu

- **\n** – nový řádek, \0 – ukončení řetězce (prázdný byte se používá v C), jiný způsob ukončení řetězce je že na začátku řetězce zapíšeme, jak je ten řetězec dlouhý

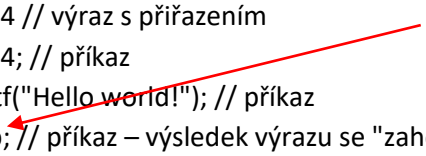
- Při posílaní a čtení dat je potřeba až 3 spojení (Adresa, Data, I/O (Jestli zapisuji nebo čtu))

- **Ukládaní informací**
  - **Datové typy**
    - Reprezentace údajů.
    - Kódování údajů.
    - Význam údajů.
  - **Datové struktury**
  - **Prvky programovacích jazyků**
  - **Proměnné, konstanty**
  - **Operátory a výrazy**
    - Aritmetické operátory (+, -, *, /, %)
    - Logické a relační operátory (NAND, XOR, OR, AND, NOT, …)
      - 0000 && 0111 = F && T = F
    - Bitové operátory (Bitwise XOR, Bitwise NOT, Bitwise AND, …)
      - 1010 & 1001 = 1000
    - Adresové operátory
    - Operátor přetypování
    - Operátor sizeof
    - Podmíněný operátor (ternární)
    - Operátor volání funkce
    - Přístupové operátory
    - Operátor čárka
    - Priorita operátorů

  - **Deklarace proměnné**

        - Je to konstrukce, která **přidělí proměnné jméno a typ**, ale **nevytvoří ji**. Najdeme je někdy v hlavičkových souborech, i když jejich použití tímto způsobem je neobvyklé.
        - **Definice proměnné** je v jazyce C současně deklarací, a kromě jména a datového typu přidělí proměnné **i paměťový prostor**
        - **Příklady:**
            - **int** pozice, delka;
            - **char** znak;
            - **float** podil,odmocnina;
        - **Inicializace proměnné**
            - Pokud již při definování proměnné víme, jakou bude mít počáteční hodnotu, můžeme ji inicializovat.
            - Příklady:
                - **char** znak = 'a';
                - **int** rozmer = 100;

- **Výraz**
    - Konstrukce jazyka, která má hodnotu (nějakého datového typu)
    - **Operandy**
        - **Operand –** je částí výrazu na kterou je aplikován jeden z **legálních** operátorů.
            - Má také hodnotu konkrétního datového typu.
            - Někdy se operandům říká podvýrazy, protože operandem může být i složitější výraz, např. $(a + b) * (c + d)$.
    - **Příkaz**
        - **Rozdíl**
            - **Výraz** nabývá dále použitelné hodnoty,
            - Primárním úkolem **příkazu** je vykonat nějaký kód.
        - Pokud je výsledkem kódu v příkazu nějaká hodnota a nejde o výraz s přiřazením, hodnota se zahodí (nepoužije, bude se ignorovat).
        - V jazyce C se **z výrazu stane příkaz tím**, **že jej ukončíme středníkem** (samozřejmě v místě, kde to dává smysl – uprostřed podmínky cyklu to samozřejmě nejde).
        - Samotný středník (;) představuje prázdný příkaz (null statement), který se někdy používá například v cyklech.
        - **Příklad**
            - x = 64 // výraz s přiřazením
            - x = 64; // příkaz
            - printf("Hello world!"); // příkaz
            - a + b; // příkaz – výsledek výrazu se "zahodí"/nepoužije



Mrtvý kód – compiler ho odstraní

- **Operátor přiřazení, L-hodnota a P-hodnota**
    - Operátor přiřazení (=) kopíruje hodnotu výrazu na své pravé straně do proměnné na levé straně.
    - V této souvislosti se mluví o L-hodnotě a P-hodnotě (anglicky L-value a R-value).
    - **L-hodnota** – je objekt v paměti, kterému lze přiřadit hodnotu.
        - proměnná, prvek pole či struktury nebo paměť odkazovaná přes ukazatel.
    - **P-hodnota** – výraz, který má vždy hodnotu a který vystupuje na pravé straně přiřazovacího operátoru.

- **Příklady:**
  - float x = -c / b;
  - pole[i] = 25;
  - *cislo = 4;
  - souradnice->x = 17;

  - (a + b) = 68; // nelze!
  - faktorial(5) = 120; // nelze!
- **Přiřazení**
  - V **jazyce C** zavedeno jako **operátor (=).**
  - V praxi to znamená, že operace přiřazení má vlastní hodnotu, kterou lze použít dále ve složitějším výrazu.
  - Může se tedy vyskytovat i v P-výrazech (x=a=b), ale nepoužívá se to, snažíme se tomu vyhnout
- Jazyk C ještě poskytuje celou kolekci rozšířených přiřazovacích operátorů, které zjednodušují zápis často používaných výrazů typu:
  - L-hodnota = L-hodnota operátor výraz;
  - V jazyce C jsou definovány tyto rozšířené přiřazovací operátory:
    - +=, -=, *=, /=, %=, >>= (1100 >>= 0011, posunutí o 2 bity), <<=, &=, ^=, |=
  - **Příklad:**
    - x *= a + b; je totéž jako x = x * (a + b);
    - x * = c + b;
      - // syntaktická chyba – mezi operátorem a = nesmí být mezera

- **Aritmetické operátory**
  - V jazyce C definovány nad číselnými datovými typy.
  - **Tři skupiny:**
    - **unární,**
    - **binární aritmetické operátory,**
    - **speciální unární operátory**

  - **Unární**
    - Ke změně znaménka výrazu.
    - Lze použít jak s celočíselnými, tak i s racionálními typy.
    - Unární plus (+), Unární mínus (-)

  - **Speciální unární operátory**
    - Unární přičtení jedničky (++), Unární odečtení jedničky (--),
    - L-hodnota++
      - Inkrementace po použití hodnoty
      - Nejprve vrátí původní hodnotu (pokud je použit ve výrazu) a poté přičte k L-hodnotě jedničku
      - **Příklad:**
        - i++; // inkrementace proměnné i
    - Chyba zápisu:
      - $x = + + i;$ Do x se zapíše inkrementovaná hodnota

- $x = i++;$ Do x se zapíše buď inkrementovaná nebo neinkrementována hodnota (není to zřejmé)

- **Operátory (+ - * /)** fungují kontextově podle datového typu operandů.
  - int op int … celočíselné dělení – výsledkem je int
  - int op float … dělení v pohyblivé řádové čárce – výsledkem je float
  - float op int … dělení v pohyblivé řádové čárce – výsledkem je float
  - float op float … dělení v pohyblivé řádové čárce – výsledkem je float

  - Multiplikativní operátory (*, /, %) mají vyšší prioritu než aditivní operátory (+, -), takže matematické výrazy můžeme zapisovat přirozeným způsobem:
    - x = 10 + 2 * 3; // výsledkem je 16

- **Logické a relační operátory**
  - Pracují s logickými hodnotami.
  - V jazyce C produkují výsledky typu **int**
    - C interpretuje hodnotu 0 jako nepravdu,
    - nenulovou hodnotu jako pravdu.
  - ISO C99 – datový typ **bool (false, true)**
    - kompatibilní s typem **int**
    - nutno dovézt rozhraní <stdbool.h>
    - identifikátor **true** je definován jako konstanta s hodnotou 1
      - Nikdy nedělejte toto: if (vyraz == true)

Logický součin	Logický součet	Logická negace
&&	||	!

  -

Rovnost	Nerovnost	Menší než	Menší nebo rovno	Větší než	Větší nebo rovno
==	!=	<	<=	>	>=

  - **Zkrácené vyhodnocování výrazů**
    - Jazyk C používá zkrácené vyhodnocování (short circuit) logických výrazů.
    - Logické výrazy se vyhodnocují zleva doprava a jakmile je jistý výsledek, vyhodnocování se ukončí.
    - **Např.: nalevo v logickém součinu vyjde False, tak se pravá strana ani nepočítá**
    - **Příklad:**
      - if (x != 0 && y / x < z)
      - Zde **nedojde k dělení nulou**, protože pokud x je rovno nule, po vyhodnocení podvýrazu x != 0, je jasný výsledek celého výrazu a druhá část už se nebude vyhodnocovat.
      - V jazyce C je **priorita** aritmetických a relačních operátorů vyšší než priorita logických operátorů => logické výrazy není třeba přehnaně závorkovat.

- **Bitové operátory**
  - **Výsledkem je číselná hodnota!**
  - **Výsledkem není logická hodnota!**
  - **Příklady:**
    - 0xF0 && 0x88 // logický součin – výsledkem je true (1)
    - 0xF0 & 0x88 // bitový součin – výsledkem je 0x80

- **Operátor přetypování**
  - **(typ)výraz**
  - Pro explicitní změnu datového typu výrazu.
  - Programátor přebírá zodpovědnost za chování typového systému jazyka.
  - Nevhodné použití explicitních konverzí způsobuje problémy!
  - Programátor by si měl být vědom, jaké dopady bude mít tato násilná změna datového typu

  - Často používán s ukazateli, zejména s obecným ukazatelem (void *).
  - **Příklady:**
    - (char)ordinální_hodnota … získání znaku z ordinální hodnoty
    - (int)výraz_float … oříznutí desetinné části (Pozor! Problémy s velkými čísly. Raději trunc(), round(), …)
    - (float)výraz_int … převod na racionální typ

- **Operátor sizeof**
  - **sizeof(typ)**
  - **sizeof(výraz)**
  - Vrací počet bajtů, které zabírá konkrétní datový typ nebo výraz.
  - Norma jazyka nezaručuje přesnou velikost číselných datových typů!

- **Ternární operátor**
  - Pro vytváření výrazů, jejichž výsledek závisí na vstupní podmínce.
  - Syntaxe: $\boldsymbol{podmínka}\ ?\ \boldsymbol{varianta\_true}\ :\ \boldsymbol{varianta\_false}$
  - Pokud je podmínka pravdivá, má výraz hodnotu $\boldsymbol{varianta\_true}$, jinak má hodnotu $\boldsymbol{varianta\_false}$.
  - **Příklad:**
    - $int\ max\ =\ (a\ >\ b)?\ a\ :\ b;\ //\ maximum\ z\ a, b$

- **Operátor čárka**
  - Vyhodnocuje se zleva doprava a celkový výraz nabývá hodnoty nejpravějšího podvýrazu.
  - Používá se například v příkazu for.
  - **Příklad:**
    - $for\ (int\ i\ =\ 0, j\ =\ 10;\ i\ !=\ j;\ i++, j--)\ //\ i++, j--)\ //\ použití\ operátoru\ čárka$
    - $int\ vysledek\ =\ (a+b, c+d);\ //\ výsledkem\ je\ c\ +\ d, a+b\ se\ zahodí$
    - $vysledek\ =\ a+b\ , c+d;\ //\ výsledkem\ je\ a+b, c+d\ je\ mrtvý\ kód$
  - **Zásada:**
    - raději nepoužívat, smysluplně se dá použít jen v minimálním počtu případů.

- **Priorita operátorů**

Priorita	**Operátory**	**Asociativita**	**skupina op.**
1.	() [] -> .	zleva doprava	primární
2.	! ~ ++ -- + - (typ) * & **sizeof**	**zprava doleva**	unární
3.	* / %	zleva doprava	multiplikativní
4.	+ -	zleva doprava	aditivní
5.	<< >>	zleva doprava	posuny
6.	< <= > >=	zleva doprava	relační
7.	== !=	zleva doprava	relační
8.	&	zleva doprava	bitový součin
9.	^	zleva doprava	exkl. bit. souči.
10.	|	zleva doprava	bitový součet
11.	&&	zleva doprava	logický součin
12.	||	zleva doprava	logický součet
13.	?:	**zprava doleva**	ternární podm.
14.	= += -= *= /= %= >>= <<= &= |= ^=	**zprava doleva**	přiřazení
15.	,	zleva doprava	op. čárka

  -