Programování a vývoj aplikací 2015-2016

1. Dvojková, osmičková a šestnáctková soustava, převody, použití
2. Algoritmy, algoritmizace, zápis, UML
3. Programovací jazyky – vývoj, rozdělení
4. Strukturované programování
5. Datové typy, pole a související
6. Operace, operátory, logické funkce, použití
7. Funkce, knihovny
8. Programovací jazyk C
9. Objektově orientované programování
10. Návrhové vzory
11. Programovací jazyk C++
12. Deklarativní programování a jazyk SQL
13. Skriptovací jazyky
14. Práce s textem, regulární výrazy
15. HTML
16. CSS
17. Javascript
18. Vývoj software, metody, workflow
19. Coding standards
20. Komentování zdrojového kódu
21. Ladění programu a programátorské chyby
22. Integrované vývojové prostředí, vlastnosti
23. Testování software
24. Licence a autorský zákon
25. Cloud computing, Iaas, Paas, SaaS

1. Dvojková, osmičková a šestnáctková soustava, převody, použití

SOUSTAVA	ZÁKLAD	ZNAKY	ZÁPIS	HISTORIE	VÝHODY	POUŽITÍ
DVOJKOVÁ - BINÁRNÍ	2	0,1	$101_{(2)}$, 101B	PŘEVOD LOGICKÝCH SLOVNÍCH VÝRAZŮ NA MATEMATICKÉ – 17. STOLETÍ (LEIBNIZ)	• K POPISU ČÍSLA STAČÍ JEN DVA STAVY – ZAPNUTO, VYPNUTO – 0,1 (SPÍNAČE) • JEDNODUCHÉ MATEMATICKÉ OPERACE (NAPŘ. 4 PRO SČÍTÁNÍ)	• DIGITÁLNÍ ELEKTRONICKÉ OBVODY, • LOGICKÉ ČLENY, • POČÍTAČE
OSMIČKOVÁ - OKTALOVÁ	8	0,1,2,3,4,5 ,6,7	$45712_{(8)}$ 45712_{O} 045712	KALIFORNSKÝ KMEN, MEXIKO, SWEDENBORG PRO KRÁLE KARLA XII – PŮVODNĚ ZÁKLAD 64, PAK ZÁKLAD 8	• SNADNÉ ROZDĚLENÍ CELKU NA POLOVINY, ČTVRTINY A OSMINY	• SYSTÉM UNIX, PŘÍSTUPOVÁ PRÁVA, • DATOVÁ SLOVA 12 BITŮ, 24 BITŮ, 36 BITŮ
ŠESTNÁCTKOVÁ - HEXADECIMÁLNÍ	16	0,1,2,3,4,5 ,6,7,8,9,A, B,C,D,E,F	$FF4B_{(16)}$ FF4B H 0XFF4B	TRADIČNÍ ČÍNSKÉ JEDNOTKY VÁHY	• PRO ZÁPIS VELKÉHO ČÍSLA STAČÍ MÁLO ZNAKŮ • SNAZŠÍ ZÁPIS DVOJKOVÉHO ČÍSLA	• ADRESY V OPERAČNÍ PAMĚTI POČÍTAČE • BARVY V HTML A CSS • ZOBRAZENÍ BINÁRNÍHO OBSAHU SOUBORŮ

POZIČNÍ SOUSTAVA A NEPOZIČNÍ SOUSTAVA – NEPOZIČNÍ NAPŘ. ŘÍMSKÉ ČÍSLICE, POZIČNÍ, TA, KTEROU POUŽÍVÁME

KAŽDÉ PŘIROZENÉ ČÍSLO *A* LZE ZAPSAT POMOCÍ POLYNOMU VE TVARU

$a = a_n z^n + a_{n-1} z^{n-1} + \dots + a_2 z^2 + a_1 z^1 + a_0 z^0$

z – **JE ZÁKLAD ČÍSELNÉ SOUSTAVY**

ai – **ČÍSELNÉ KOEFICIENTY JEDNOTLIVÝCH ŘÁDOVÝCH MÍST ČÍSLA (HODNOTA ČÍSLICE NA DANÉ POZICI)**

n, n-1, n-2, …, 2, 1, 0 - **JSOU JEDNOTLIVÉ MOCNINY ZÁKLADU Z – POZICE ČÍSLICE, ČÍSLOVÁNO ZPRAVA DOLEVA, OD NULY**

n - **MOCNINA NEJVÍCE VLEVO**

$z^n, z^{n-1}, z^2, z^1, z^0$ – **PŘEDSTAVUJÍ VÁHU ČÍSLICE NA DANÉ POZICI**

například: 25032 = 20000 + 5000 + 0 + 30 + 2 = 2.104 + 5.103 + 0.102 + 3.101 + 2.100

TERMÍNY: BINÁRNÍ, OKTALOVÁ, HEXADECIMÁLNÍ, POZIČNÍ, NEPOZIČNÍ, ZÁKLAD, VÁHA

BIT – JEDNOTKA INFORMACE, ZNAČKA b, **LSB** – LEAST SIGNIFICANT BIT – NEJMÉNĚ VÝZNAMNÝ BIT – NÁSOBÍ 2^0, **MSB** – MOST SIGNIFICANT BIT – NEJVÝZNAMNĚJŠÍ BIT – NÁSOBÍ 2^7

NIBBLE – 4 BITY (0110) – 1 POZICE V ŠESTNÁCTKOVÉ SOUSTAVĚ

BYTE – 8 BITŮ, ZNAČKA B

SLOVO – POČET BITŮ ZPRACOVANÝCH V POČÍTAČI NAJEDNOU (8, 16, 32, 64 BITŮ)

JEDNOTKY – 1kB = 1024B = 2^{10}B, MB = 1024kB = 1024 x 1024B, GB = 1024MB, TB = 1024GB

Převody:

Z NEDESÍTKOVÉ (16,8,2) DO DESÍTKOVÉ SOUSTAVY: DOPLNÍME DO POLYNOMU A VYPOČTEME:

DVOJKOVÉ ČÍSLO $10011_{(2)}$ – určíme pozice: $1^4 0^3 0^2 1^1 1^0$ a vypočteme $1x2^4 + 0x2^3 + 0x2^2 + 1x2^1 + 1x2^0$ (JINÉ SOUSTAVY OBDOBNĚ, **2^0=1!**)

Z DESÍTKOVÉ DO NEDESÍTKOVÉ(16,8,2) SOUSTAVY: DĚLÍME ZÁKLADEM, DO KTERÉ PŘEVÁDÍME:

123 : 2 = 61	zbytek 1
61 : 2 = 30	zbytek 1
30 : 2 = 15	zbytek 0
15 :2 = 7	zbytek 1
7 : 2 = 3	zbytek 1
3 : 2 = 1	zbytek 1
1 :2 = 0 (konec)	zbytek 1

Tedy $123_{(10)} = 1111011_{(2)}$

Z NEDESÍTKOVÉ DO NEDESÍTKOVÉ(16,8,2) SOUSTAVY:

NAPŘ:Z 2 -> 16

Dvojková = 1110 0101

Hexadecimální = E 5 = 0xE5

NAPŘ:Z 2 -> 8

Dvojková = 011 100 101

Oktalová = 3 4 5 = 0345

PRAKTICKÁ DOVEDNOST:

- VYPSAT TABULKU HODNOT V 16KOVÉ SOUSTAVĚ, 8KOVÉ, 2KOVÉ – PRO DESÍTKOVOU SOUSTAVU OD 0 DO15
- PŘEVÉST ČÍSLO V NEDESÍTKOVÉ SOUSTAVĚ (16,8,2) DO DESÍTKOVÉ
- PŘEVÉST ČÍSLO Z DESÍTKOVÉ SOUSTAVY DO NEDESÍTKOVÉ SOUSTAVY (16,8,2)
- PŘEVÉST ČÍSLO MEZI NEDESÍTKOVÝMI SOUSTAVAMI (16,8,2)

DOPORUČENÉ ZDROJE:

http://programovaci-ucebnice.g6.cz/ucebnice/UcebniceJazykaJava/2_Ciselne_soustavy.xhtml
http://it-slovnik.cz/navody/prevody-ciselnych-soustav
http://www.sallyx.org/sally/c/c02.php
https://www.spskladno.cz/stahuj.php?id=3007

ZÁPORNÁ ČÍSLA:

PŘÍMÝ KÓD: PRVNÍ BIT 1=ZÁPORNÉ ČÍSLO

INVERZNÍ KÓD: ZÁPORNÉ ČÍSLO SE VYTVOŘÍ BITOVOU NEGACÍ

DESETINNÁ ČÍSLA V BINÁRNÍ SOUSTAVĚ:

ZA DESETINOU ČÁRKOU JSOU $A_{-1}x2^{-1} + A_{-2}x2^{-2} + A_{-3}x2^{-3}$…

2. Algoritmy, algoritmizace, zápis, UML

ALGORITMUS JE KONEČNÁ POSLOUPNOST JEDNOZNAČNĚ URČENÝCH KROKŮ POPISUJÍCÍCH ŘEŠENÍ NĚJAKÉHO PROBLÉMU.
POUŽITÍ: PROGRAMY, OVLÁDÁNÍ ROBOTŮ, PRAČKA, MYČKA, RECEPTY VAŘENÍ, LEGOVÁNÍ OCELE, …
VLASTNOSTI ALGORITMU:

- **DETERMINOVANOST** (JEDNOZNAČNOST - POSTUP JEDNOZNAČNĚ URČENÝ A PROVEDITELNÝ – OPAKOVATELNOST)
- **REZULTATIVNOST** (KONEČNOST - PO KONEČNÉM MNOŽSTVÍ KROKŮ VEDE ALGORITMUS K DOSAŽENÍ ŘEŠENÍ, NECYKLÍ)
- **REAPLIKOVATELNOST** – HROMADNOST – UNIVERZÁLNOST (DOKÁŽE VYŘEŠIT NIKOLI JEN JEDNU JEDINOU ÚLOHU, ALE ALESPOŇ ÚLOHY URČITÉHO TYPU)
- **EFEKTIVNOST** (PROVEDITELNÝ POMOCÍ DOSTUPNÝCH PROSTŘEDKŮ, V KONEČNÉM ČASE, S CO NEJMENŠÍM ÚSILÍM – ČASOVÝM, PAMĚŤOVÝM)
- **SPRÁVNOST** (REZULTATIVNOST ŘEŠÍ SKUTEČNĚ ZADANÝ PROBLÉM A SPRÁVNÝM POSTUPEM)
- **ELEMENTÁRNOST** (SKLÁDÁ SE Z JEDNODUCHÝCH – ELEMENTÁRNÍCH KROKŮ, KTERÉ DOKÁŽEME – UMÍME VYKONAT)

ZÁPISY ALGORITMU:

- **TEXTOVĚ** – SLOVNĚ – BĚŽNÝ JAZYK (RECEPT, CESTA KAM)
- **GRAFICKY** – VÝVOJOVÉ DIAGRAMY, STRUKTUROGRAMY
- **MATEMATICKY** – ROVNICE, VZTAH MEZI VELIČINAMI
- **PROGRAMEM** – V PROGRAMOVACÍM JAZYCE

ALGORITMIZACE JE PROCES VYTVÁŘENÍ A SESTAVENÍ ALGORITMŮ
ZÁKLADNÍ PRINCIPY ALGORITMIZACE:

- **DEKOMPOZICE** – ROZDĚLOVÁNÍ NA PODPROBLÉMY
- **ABSTRAKCE** – ZJEDNODUŠENÝ POHLED NA SLOŽITÉ VĚCI (ODSTRANĚNÍ NEPODSTATNÝCH DETAILŮ – NALEZENÍ SPOLEČNÉHO, NADŘAZUJÍCÍHO ZASTŘEŠUJÍCÍHO VÝZNAMU)

METODOLOGIE ALGORITMIZACE:

- **SHORA DOLŮ** – UMOŽŇUJE ŘEŠIT PROBLÉMY S NADHLEDEM, ROZDĚLIT NA KLÍČOVÉ PODPROBLÉMY, A POSTUPNĚ TAK VIDĚT JEJICH SOUVISLOSTI A ZÁROVEŇ CO JE POTŘEBA PRO JEJICH JEDNOTLIVÉ ŘEŠENÍ – PŘI PRŮCHODU DOLŮ DO HIERARCHICKÉ STRUKTURY ŘEŠENÍ PROBLÉMU
- **ZDOLA NAHORU** – VYCHÁZÍME ZE ZÁKLADNÍCH PŘÍKAZŮ A JIŽ HOTOVÝCH ČÁSTÍ ŘEŠENÍ PROBLÉMU A POSTUPNĚ JE SPOJUJEME DOHROMADY, PRO ÚČEL CELKOVÉHO ŘEŠENÍ

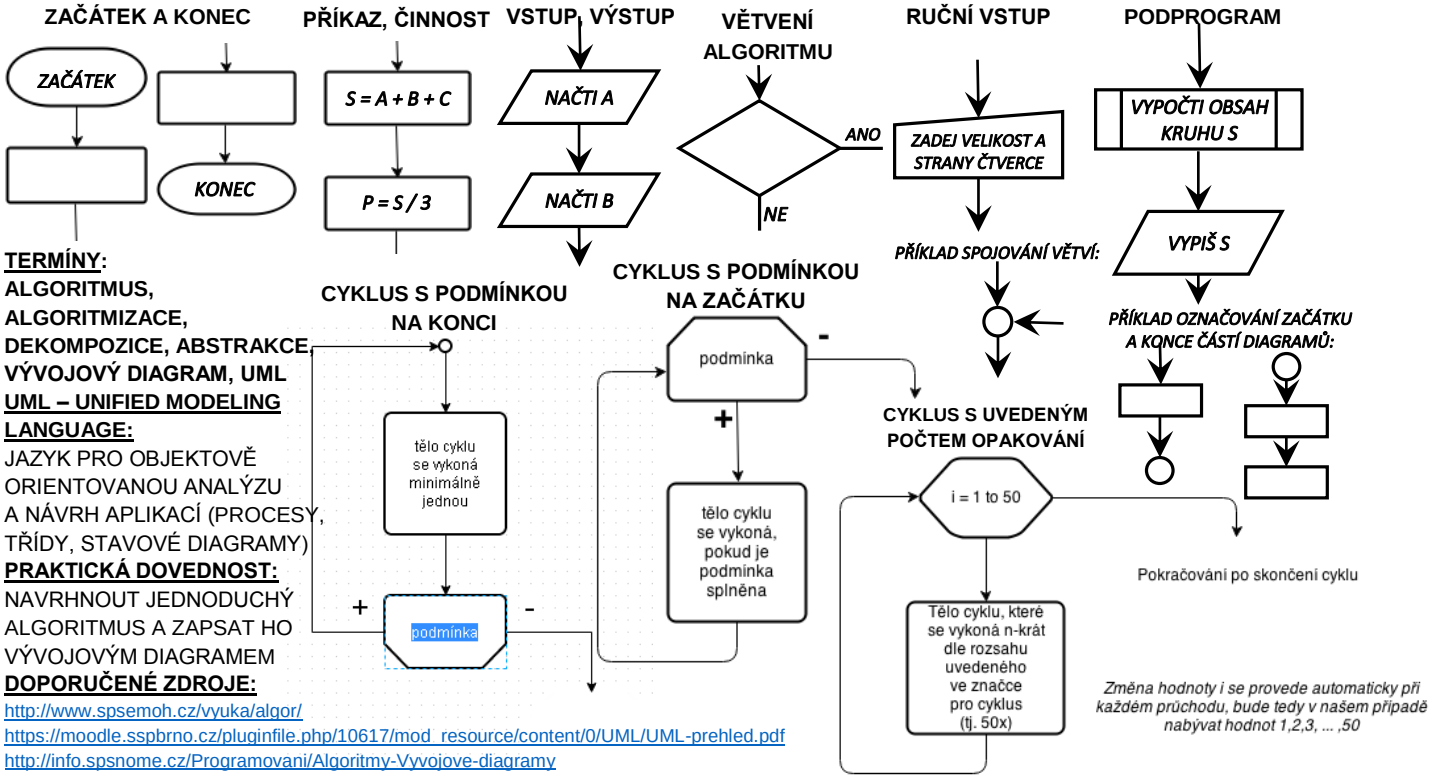
KVALITA ALGORITMŮ:

- OPERAČNÍ SLOŽITOST (RYCHLOST ZPRACOVÁNÍ, MNOŽSTVÍ INSTRUKCÍ)
- PAMĚŤOVÁ SLOŽITOST (POTŘEBA – MNOŽSTVÍ VYUŽITÉ PAMĚTI PRO ŘEŠENÍ ALGORITMU)
- PŘEHLEDNOST A SROZUMITELNOST (URČUJE I MOŽNOU ZNOVUPOUŽITELNOST NĚKÝM JINÝM)

VÝVOJOVÉ DIAGRAMY ZNÁZORŇUJÍ PRŮBĚH, ČI STRUKTURU ALGORITMU A TEDY I PROGRAMU, KTERÝ HO REALIZUJE.

- PŘEHLEDNÉ GRAFICKÉ ZOBRAZENÍ PRŮBĚHU PROGRAMU, PRO PROMYŠLENÍ JAK BUDE DANÝ PROGRAM FUNGOVAT.
- VÝVOJOVÉ DIAGRAMY SE SKLÁDAJÍ Z GRAFICKÝCH ZNAČEK, JSOU VZÁJEMNĚ PROPOJENY ORIENTOVANÝMI ŠIPKAMI.

GRAFICKÉ ZNAČKY VÝVOJOVÝCH DIAGRAMŮ:



TERMÍNY:
ALGORITMUS, ALGORITMIZACE, DEKOMPOZICE, ABSTRAKCE, VÝVOJOVÝ DIAGRAM, UML
UML – UNIFIED MODELING LANGUAGE:
JAZYK PRO OBJEKTOVĚ ORIENTOVANOU ANALÝZU A NÁVRH APLIKACÍ (PROCESY, TŘÍDY, STAVOVÉ DIAGRAMY)
PRAKTICKÁ DOVEDNOST:
NAVRHNOUT JEDNODUCHÝ ALGORITMUS A ZAPSAT HO VÝVOJOVÝM DIAGRAMEM
DOPORUČENÉ ZDROJE:
http://www.spsemoh.cz/vyuka/algor/
https://moodle.sspbrno.cz/pluginfile.php/10617/mod_resource/content/0/UML/UML-prehled.pdf
http://info.spsnome.cz/Programovani/Algoritmy-Vyvojove-diagramy

3. Programovací jazyky – vývoj, rozdělení

PROGRAMOVACÍ JAZYK: JE SOUBOR VÝRAZŮ A PRAVIDEL PRO ZÁPIS ALGORITMŮ, KTERÉ VYKONÁ POČÍTAČ

GENERACE	PROGRAMOVACÍ JAZYKY	VÝHODY	NEVÝHODY	POPIS
1. GENERACE 1GL	STROJOVÝ KÓD STROJOVÝ JAZYK	• RYCHLÉ VYKONÁVÁNÍ • PLNÁ KONTROLA HARDWARU	• ZÁVISÍ NA PROCESORU • HODNĚ KÓDOVÁNÍ, PŘI ZMĚNĚ HW, MNOHO KÓDU • VYŽADUJE PŘESNÉ PROGRAMOVÁNÍ • OBTÍŽNÝ VÝVOJ, LADĚNÍ	POŽÍVÁ **INSTRUKCE PROCESORU V BINÁRNÍ FORMĚ**, **DOBA TRVÁNÍ INSTRUKCÍ SOUVISÍ S CYKLEM PROCESORU, PŘÍMÝ MANAGEMENT PAMĚTI, PŘÍSTUP K REGISTRŮM**, **POUŽITÍ: ČÁSTI OS, POČÍTAČOVÉ VIRY, ČÁSTI PROGRAMŮ VYŽADUJÍCÍ VELKOU RYCHLOST**
2. GENERACE 2GL	JAZYK SYMBOLICKÝCH ADRES, JSA ASSEMBLER	• STEJNÉ JAKO 1. GENERACE • MNEMOTECHNICKÉ KODY SE ČLOVĚKU LÉPE PAMATUJÍ	• STEJNÉ JAKO 1. GENERACE	1GL + **TÉMĚŘ ŽÁDNÁ ABSTRAKCE SMĚREM OD HARDWARU**, **ASSEMBLER JE PROGRAM, KTERÝ PŘEKLÁDÁ JAZYK SYMBOLICKÝCH ADRES DO STROJOVÉHO JAZYKA**, LADĚNÍ V DEBUGGERU, PROGRAMY AŽ STOVKY kB
3. GENERACE 3GL TAHLE A VYŠŠÍ GENERACE VYŠŠÍ PROGRAMOVACÍ JAZYKY	FORTRAN, ALGOL, COBOL, BASIC, C, PASCAL	• POUŽÍVAJÍ SE ANGLICKÁ SLOVA • JEDNODUŠŠÍ • ÚČINNĚJŠÍ • POTŘEBUJEME MINIMÁLNÍ HW ZNALOSTI	• ZDROJOVÝ KÓD SE PŘEVÁDÍ DO STROJOVÉHO A PROTO JE VYKONÁVÁNÍ PROGRAMU POMALEJŠÍ • JAZYKY VYVINUTÉ PRO SPECIFICKÉ ÚČELY = VÍCE STANDARDŮ	**REAKCE NA NEVÝHODY DRUHÉ GENERACE** **HW A OS NEZÁVISLÉ,** HLAVNÍ JE APLIKACE A NE HW, POTŘEBA: **KOMPILÁTOR NEBO INTERPRET** IMPERATIVNÍ A FUNKCIONÁLNÍ JAZYKY STRUKTUROVANÉ PROGRAMOVÁNÍ: ŘÍDÍCÍ STRUKTURY, DATOVÉ STRUKTURY, PROCEDURY, FUNKCE
3,5. GENERACE	OOP JAZYKY: SIMULA67, C++, JAVA, OBJECT PASCAL, CLOS, SMALLTALK, EIFFEL, SATHER	• ZAPOUZDŘENOST – ZNÁME JEN ROZHRANÍ TŘÍD • MODULÁRNOST, SNADNÉ PŘIDÁVÁNÍ DALŠÍCH FUNKCÍ • ZNOVUPOUŽITELNÝ KÓD • KONTROLA PŘÍSTUPU K DATŮM	• VĚTŠINOU O NĚCO POMALEJŠÍ • SLOŽITĚJŠÍ KÓD – PROVÁZANĚJŠÍ, VÍCE SOUBORŮ (S VHODNÝM IDE OK)	**TŘÍDA, OBJEKT, POLYMORFISMUS, DĚDIČNOST, ZAPOUZDŘENÍ**
4. GENERACE 4GL	PARADOX, VISUAL FOXPRO, SQL, LOGO, VISUAL BASIC, ADA, NOMAD, FOCUS, MATLAB	• VÍCE SE BLÍŽÍ JAZYKU A MYŠLENÍ ČLOVĚKA • SNIŽUJÍ ČAS A ÚSILÍ VE VÝVOJI SW = SNIŽUJE CENU	• SPECIFICKY ZAMĚŘENÉ NA URČITÉ OBLASTI ŘEŠÍ SPECIFICKÉ OBOROVÉ PROBLÉMY	TEXTOVÉ NEBO **VIZUÁLNÍ VÝVOJOVÉ PROSTŘEDÍ, GRAFICKÉ NÁSTROJE** NĚKDY SE OZNAČUJÍ JAKO PROGRAMY GENERUJÍCÍ APLIKACE NĚKDY POPISOVANÉ – ŘÍKÁME, CO CHCEME, A NE JAK TO CHCEME - **DEKLARATIVNÍ** NAPŘ: **DATABÁZOVÉ DOTAZY, GENERÁTORY REPORTŮ, TVORBA GRAFICKÉHO ROZHRANÍ**
5. GENERACE 5GL – NENÍ JASNĚ URČENO JAKÉ JAZYKY	PROLOG, LISP, OPS5, MERCURY	• POUŽÍVÁ PŘIROZENÝ JAZYK • POČÍTAČ ŘEŠÍ PROBLÉM MÍSTO VÁS	• ZNAČNÁ VÝPOČETNÍ NÁROČNOST	ČASTO JEN **LOGICKÉ PROGRAMOVÁNÍ** PROGRAMÁTOR POMOCÍ NADEFINUJE OBJEKTY, PRAVIDLA A OMEZENÍ ŘEŠENÍ. **UMĚLÁ INTELIGENCE** A JEJÍ VÝVOJ POUŽÍVÁ EXPERTNÍ SYSTÉMY A ZNALOSTNÍ DATABÁZE, VYHLEDÁVAČE

ROZDĚLENÍ PODLE ZPŮSOBU PŘEKLADU DO STROJOVÉHO KÓDU:

KOMPILOVANÉ – **VÝHODY** (SYNTAKTICKÁ KONTROLA CELÉHO KÓDU JEŠTĚ PŘED SPUŠTĚNÍM V PRŮBĚHU KOMPILACE – PŘEKLADU, RYCHLÝ BĚH PROGRAMU, PROGRAM SAMOSTATNĚ SPUSTITELNÝ BEZ INTERPRETU), **NEVÝHODY** (PŘI ZMĚNĚ PROGRAMU ZNOVU PŘEKLAD CELÉHO PROGRAMU, ZVLÁŠ KOMPILACE PRO RŮZNÉ OS), JAZYKY (C, PASCAL), KOMPILÁTOR - PŘEKLADAČ PŘED PRVNÍM SPUŠTĚNÍM PROGRAMU SE CELÝ YDROJOVÝ KÓD PŘEVEDE DO STROJOVÉHO KÓDU A TEN SE ULOŽÍ NA DISK, PŘEKLADAČ **PROVÁDÍ SYNTAKTICKOU KONTROLU** CELÉHO ZDROJOVÉHO KÓDU

POSTUP KOMPILACE: ZPRACOVÁNÍ DIREKTIV PREPROCESORU (PREPROCESSOR NAHRADÍ INCLUDE SOUBORY, DEFINE MAKRY A KONSTANTAMI), KOMPILACE – PŘEKLAD (KOMPILÁTOR – VYTVOŘÍ OBJEKT SOUBOR .o, OZNÁMÍ CHYBY,), A SLINKOVÁNÍ (LINKER SPOJÍ VŠECHNY OBJEKT SOUBORY, SDÍLENÉ ČI DYNAMICKÉ KNIHOVNY DO JEDNOHO)

INTERPRETOVANÉ – ČASTO SKRIPTOVACÍ, **VÝHODY** (NENÍ TŘEBA PŘEKLAD, PŘI ZMĚNĚ ZDROJOVÉHO KÓDU, SNADNÁ PŘENOSITELNOST VERZÍ MEZI OS), **NEVÝHODY** (POMALEJŠÍ BĚH PROGRAMŮ, SYNTAKTICKÉ CHYBY ZJIŠTĚNY AŽ BĚHEM BĚHU PROGRAMU), **JAZYKY** (BASIC, PHP, JAVASCRIPT), INTERPRET SPOUŠÍ PROGRAMY, PŘEVÁDÍ INSTRUKCE DO STROJOVÉHO KÓDU A VYKONÁVÁ, NEVZNIKÁ SAMOSTATNĚ SPUSTITELNÝ PROGRAM,

HYBRIDNÍ – JAZYKY (JAVA, PYTHON), NEJPRVE SE KOMPILUJÍ DO OBECNÉHO STROJOVÉHO KÓDU, KTERÝ SE PAK INTERPRETUJE NA RŮZNÝCH POČÍTAČÍCH RŮZNÝMI ZPŮSOBY

ROZDĚLENÍ PODLE ZPŮSOBU ZÁPISU PROGRAMU:

IMPERATIVNÍ: POPISUJE ALGORITMUS JAKO SEKVENCE PŘÍKAZŮ VYKONÁVANÝCH JEDEN PO DRUHÉM. ŘEŠÍ JAK. PASCAL, C

FUNKCIONÁLNÍ: ZALOŽENO NA ZÁPISU PROGRAMU VE TVARU MATEMATICKÉHO VÝRAZU. LISP, SCHEME

DEKLARATIVNÍ: ŘÍKÁME JAZYKU TO, CO MÁ VYKONAT, ALE NIKOLIV JAK. SQL

LOGICKÉ: POUŽÍVAJÍ PRO POPIS ALGORITMŮ FORMÁLNÍ MATEMATICKOU LOGIKU A LOGICKÉ ODVOZOVÁNÍ. PROLOG

TERMÍNY: PROGRAMOVACÍ JAZYK, STROJOVÝ KÓD, ASSEMBLER, INSTRUKCE, REGISTRY, KOMPILÁTOR – PŘEKLADAČ, INTERPRET, LINKER, IMPERATIVNÍ, PROCEDURÁLNÍ – STRUKTUROVANÉ PROGRAMOVÁNÍ, OBJEKTOVĚ ORIENTOVANÉ, FUNKCIONÁLNÍ, DEKLARATIVNÍ, LOGICKÉ, STYTICKY TYPOVANÝ: TYP PROMĚNNÝCH ZNÁM V DOBĚ PŘEKLADU

POUŽITÍ JAZYK C: MIKROPROCESSORY, HW, VÝPOČTY, SYSTÉMOVÉ APLIKACE, RYCHLOST, **C++** I TŘEBA HRY, VELKÉ PROJEKTY

DOPORUČENÉ ZDROJE: http://www.fi.muni.cz/usr/jkucera/pv109/2002/xkriz1.htm, http://k-prog.wz.cz/progjaz/
http://www.ms.mff.cuni.cz/~husakr/ivt_programovani/prehled_programovacich_jazyku.pdf

4. Strukturované programování

POCHOPENÍ: PRO ŘEŠENÍ KOMPLIKOVANĚJŠÍCH PROBLÉMŮ, JE POTŘEBA PROBLÉMY ROZDĚLIT NA PODPROBLÉMY - DEKOMPOZICE. TY SE PAK ŘEŠÍ V JEDNOTLIVÝCH PODPROGRAMECH, KTERÉ VOLÁ HLAVNÍ PROGRAM. PODPROGRAMY JSOU REALIZOVÁNY FUNKCEMI NEBO PROCEDURAMI A UVNITŘ NICH ŘÍDÍCÍMI KONSTRUKCEMI (FOR, WHILE, IF, …). PODPROGRAMY, NEBO BLOKY KÓDU, MOHOU OBSAHOVAT DALŠÍ A DALŠÍ ZANOŘENÉ BLOKY KÓDU. VSTUP HODNOT PROMĚNNÝCH DO FUNKCÍ SE PROVÁDÍ POMOCÍ PARAMETRŮ NEBO GLOBÁLNÍCH PROMĚNNÝCH.
PROGRAM JE STRUKTUROVÁN NA ČÁSTI A TYTO JEDNOTLIVÉ ČÁSTI LZE VOLAT VÍCEKRÁT A ZNOVUPOUŽÍT. DÍKY TOMU SE KÓD TOLIK NEOPAKUJE.
MNOHDY JSOU FUNKCE ZABÝVAJÍCÍ SE ŘEŠENÍM PODOBNÝCH VĚCÍ SDRUŽOVÁNY DO MODULŮ (MODULÁRNÍ PROGRAMOVÁNÍ – FUNKČNĚ PODOBNÉ ČÁSTI PROGRAMU ODDĚLENÉ OD JINÝCH, VYHÝBAT SE GLOBÁLNÍM PROMĚNNÝM).

OBVYKLÝ PRŮBĚH STRUKTUROVANÉHO PROGRAMU BÝVÁ: VSTUP DAT, ZPRACOVÁNÍ DAT, VÝSTUP ZPRACOVANÝCH DAT.

PROGRAM SE SKLÁDÁ Z:

- **PROMĚNNÝCH**
  - RŮZNÝCH **TYPŮ**: (CELÁ ČÍSLA, DESETINÁ ČÍSLA, ZNAKY, TEXTOVÉ ŘETĚZCE, POLE HODNOT, DATOVÉ STRUKTURY, NĚKDY LOGICKÉ PROMĚNNÉ)
  - S RŮZNOU **PLATNOSTÍ**: (**LOKÁLNÍ PROMĚNNÉ** – POUZE PRO DANÝ BLOK, FUNKCI, MODUL, **GLOBÁLNÍ PROMĚNNÉ** – PRO VŠECHNY PODŘÍZENÉ ČÁSTI STRUKTURY PROGRAMU)
- **KONSTANT**
- **PŘÍKAZŮ** (NAPŘÍKLAD VSTUP, VÝSTUP, PŘIŘAZENÍ HODNOTY, LOGICKÉ, MATEMATICKÉ, BITOVÉ OPERACE)
- **ŘÍDÍCÍCH STRUKTUR**, HLAVNÍ ŘÍDÍCÍ KONSTRUKCE, KTERÉ SDRUŽUJÍ PROGRAMY – STRUKTURY ŘÍDÍCÍ PROGRAM JSOU:
  - **SEKVENCE PŘÍKAZŮ** (JEDEN PŘÍKAZ, PODPROGRAM NÁSLEDUJE ZA DRUHÝM, V PŘESNĚ URČENÉ SEKVENCI)
  - **VĚTVENÍ PROGRAMU** (ROZHODOVÁNÍ, JAKÁ ČÁST PROGRAMU SE VYKONÁ – C JAZYK IF)
  - **CYKLY PROGRAMU** (URČITÁ ČÁST PROGRAMU SE BUDE VYKONÁVAT OPAKOVANĚ – C JAZYK WHILE, FOR)

SYNTAXE PROGRAMOVACÍHO JAZYKA JE SOUBOR PRAVIDEL ZÁPISU ZDROJOVÉHO KÓDU PROGRAMU A VŠECH JEHO ČÁSTÍ. CHYBA V SYNTAXI ZNAMENÁ, NAPŘÍKLAD OPOMENUTÍ STŘEDNÍKU NA KONCI PŘÍKAZU V JAZYCE C
SÉMATIKA PROGRAMOVACÍHO JAZYKA JE VÝZNAM VÝRAZŮ, PŘÍKAZŮ, V PODSTATĚ JDE O TO, ŽE POKUD JE CHYBA V TOMTO, JEDNÁ SE O JAKOUSI LOGICKOU CHYBU.
KLÍČOVÁ SLOVA: SLOVA VYHRAZENÁ PROGRAMOVACÍM JAZYKEM PRO POUŽITÍ K VYTVÁŘENÍ ZDROJOVÉHO KÓDU PROGRAMU

OBVYKLÉ ZÁSADY VYTVÁŘENÍ STRUKTUROVANÉHO PROGRAMU:

1. PŘI **NÁVRHU PROGRAMU SE POSTUPUJE „SHORA DOLŮ“** – VIZ ALGORITMIZACE. PAK NÁSLEDUJE DEKOMPOZICE NA ČÁSTI A PODČÁSTI.
2. **MODULY** – ČÁSTI, ZE KTERÝCH SE PROGRAM SKLÁDÁ, JSOU RELATIVNĚ SAMOSTATNÉ. NAVZÁJEM SE VOLAJÍ A JSOU **PROPOJENY TÍM, ŽE SI PŘEDÁVAJÍ DATA** A OČEKÁVAJÍ JEDEN OD DRUHÉHO KONKRÉTNÍ TYP TĚCHTO DAT. PODPROGRAMY – FUNKCE NEBO PROCEDURY MAJÍ JEDINÝ KONKRÉTNÍ VSTUP S JASNĚ DANÝMI VSTUPNÍMI PARAMETRY – ROZHRANÍM.
3. **MODULY MAJÍ HIERARCHICKOU STRUKTURU** A PLATÍ, ŽE MODUL NA VYŠŠÍ ÚROVNI VOLÁ MODULY NIŽŠÍCH ÚROVNÍ. **PO PROVEDENÍ FUNKCE VOLANÉHO MODULU NA NIŽŠÍ ÚROVNI, POKRAČUJE V ČINNOSTI MODUL, KTERÝ HO VOLAL** – VRACÍ SE DO NĚJ PRŮBĚH PROGRAMU.
4. **ZAKÁZÁNO JE POUŽÍVAT PŘÍKAZY SKOKU DO JINÝCH ČÁSTÍ PROGRAMU** (LZE SE JEN ODKAZOVAT NA FUNKCE, KTERÉ SE PO PROVEDENÍ VRÁTÍ DO PROGRAMU A POKRAČUJÍ DÁL) – VYŠŠÍ NEPŘEHLEDNOST.
5. **JEDNOTLIVÉ ŘÍDÍCÍ STRUKTURY SE DO SEBE MOHOU ZANOŘOVAT** A OBSAHOVAT JEDNA DRUHOU.

PROGRAMOVACÍ JAZYKY, VE KTERÝCH SE PROGRAMUJE STRUKTUROVANĚ: JAZYK C, PASCAL, BASIC, JAVASCRIPT, …

PŘÍKLADY NÁLEŽITOSTÍ STRUKTUROVANÉHO PROGRAMOVÁNÍ V JAZYCE C: VIZ OTÁZKA PROGRAMOVACÍ JAZYK C – DEFINICE A INICIALIZACE PROMĚNNÉ, CYKLY FOR, WHILE, VĚTVENÍ PROGRAMU

VÝHODY STRUKTUROVANÉHO PROGRAMOVÁNÍ: PROGRAM JE LÉPE ČITELNÝ, SROZUMITELNÝ, SNADNĚJI ROZŠIŘITELNÝ A UDRŽITELNÝ, DÍKY ROZDĚLENÍ DO STRUKTURY, MOHOU NĚKTERÉ ČÁSTI STRUKTURY PROGRAMU VYVÍJET JINÍ PROGRAMÁTOŘI,…

TERMÍNY: STRUKTUROVANÉ PROGRAMOVÁNÍ, PROCEDURÁLNÍ PROGRAMOVÁNÍ, FUNKCE A PROCEDURY VIZ OTÁZKA Č. 7, PROMĚNNÉ, LOKÁLNÍ PROMĚNNÉ, GLOBÁLNÍ PROMĚNNÉ, KONSTANTY, PŘÍKAZY, ŘÍDÍCÍ STRUKTURY, DATOVÉ TYPY, VĚTVENÍ PROGRAMU, CYKYLY, KLÍČOVÉ SLOVO

PRAKTICKÁ DOVEDNOST: NAPSAT JEDNODUCHÝ PROGRAM V JAZYCE C, POUŽITÍ ZÁKLADNÍCH ŘÍDÍCÍCH STRUKTUR V JAZYCE C.
DOPORUČENÉ ZDROJE:
http://lide.uhk.cz/pdf/ucitel/jehlivl1/vyuka/progrmovani2/1Strukturovan%C3%A9%20programov%C3%A1n%C3%AD.pdf
http://www.dusanpolansky.cz/clanky/banker.html

5. Datové typy, pole a související

PROGRAMY POUŽÍVAJÍ JAKO VSTUP I VÝSTUP, STEJNĚ TAK I V PRŮBĚHU ZPRACOVÁNÍ PROGRAMU **PROMĚNNÉ**, KTERÉ MAJÍ **JEDINEČNÝ NÁZEV**, JSOU **ULOŽENY V PAMĚTI** A **MAJÍ** SVOJI **HODNOTU**. TATO HODNOTA JE **V ROZSAHU DANÉM JEJICH TYPEM** NAPŘÍKLAD: int pocet; JE V JAZYCE C PROMĚNNÁ pocet TYPU int TEDY CELOČÍSELNÉHO TYPU.

IDENTIFIKÁTOR JE **JEDINEČNÉ OZNAČENÍ PROMĚNNÉ, KONSTANTY NEBO FUNKCE**, KTERÝM JE DANÁ ČÁST PROGRAMU VYVOLÁNA, NEBO POUŽITA. PRO IDENTIFIKÁTORY **PLATÍ URČITÁ SYNTAXE**, KTERÁ MŮŽE BÝT ROZDÍLNÁ PRO RŮZNÉ PROGRAMOVACÍ JAZYKY. DÁLE SE IDENTIFIKÁTORY MOHOU **VYTVÁŘET PODLE** URČITÝCH DOHODNUTÝCH PRAVIDEL, ZPŘEHLEDŇUJÍCÍCH KÓD, VIZ **CODING STANDARDS** NEBO ŠTÁBNÍ KULTURA.

VE VĚTŠINĚ JAZYKŮ JE POTŘEBA PROMĚNNÉ PŘED POUŽITÍM DEKLAROVAT A DEFINOVAT. POTÉ JE MŮŽEME INICIALIZOVAT.
DEKLARACE ZNAMENÁ, ŽE SE PROMĚNNÉ **PŘIŘADÍ IDENTIFIKÁTOR A TYP**
DEFINICE ZNAMENÁ, ŽE SE PROMĚNNÉ **PŘIŘADÍ IDENTIFIKÁTOR, TYP A MÍSTO V PAMĚTI**
INICIALIZACE PŘEDSTAVUJE **PŘIŘAZENÍ KONKRÉTNÍ HODNOTY DANÉHO TYPU DO** UVEDENÉ **PROMĚNNÉ**.

KONSTANTA JE V PODSTATĚ **PROMĚNNÁ**, KTERÁ **PO CELOU DOBU** PRŮBĚHU PROGRAMU **NABÝVÁ STEJNOU HODNOTU**.
LITERÁL: **ČÍSELNÝ** PŘEDSTAVUJE ČÍSLO ZAPSANÉ V PROGRAMU (123.01, 123E0) NEBO **ŘETĚZCOVÝ** PŘEDSTAVUJE ZNAKY ZAPSANÉ V PROGRAMU V UVOZOVKÁCH (‘Vypsaný text’) – **KONSTANTY NEMAJÍCÍ JMÉNO**

DATOVÉ TYPY:

- **ČÍSELNÉ** (**CELOČÍSELNÉ**, **DESETINNÉ** <POHYBLIVÝ ŘÁDOVÁ ČÁRKA> – ROZSAH URČEN POČTEM BYTŮ, KAM SE UKLÁDAJÍ)
- **TEXTOVÉ** (JEDEN A VÍCE ZNAKŮ)
- **LOGICKÉ, BOOL** (JEN DVĚ HODNOTY REPREZENTUJÍCÍ PRAVDU A NEPRAVDU)
- **UKAZATELE** (ODKAZUJÍ NA ADRESY V PAMĚTI)
- **POLE** (VĚTŠÍ MNOŽSTVÍ POLOŽEK SE STEJNÝM DATOVÝM TYPEM – POLOŽKY MOHOU BÝT BUĎ ČÍSLOVANÉ <V JAZYCE C: PRVNÍ POLOŽKA POLE a JE NAPŘÍKLAD a[0]> TAKZVANÝ INDEXEM JE V TOMTO OZNAČENÍ 0, a[5] – INDEX JE 5, NEBO JSOU OZNAČENY TEXTEM TZV. ASOCIATIVNÍ POLE NAPŘÍKLAD b[‘druhy‘])
- **STRUKTURY**, **TŘÍDY**

DATOVÉ TYPY V JAZYCE C A C++ - VIZ JAZYK C A C++
int pocet; definice celočíselné proměnné pocet | **char znak;** definice 8-bitové proměnné znak | **unsigned int velikost;** definice celočíselné proměnné velikost, která nabývá vždy kladných hodnot | **float vypocet;** definice proměnné vypocet s typem reálného čísla s plovoucí desetinou čárkou | **double vypocet;** definice reálné proměnné výpočet s dvojitou přesností, proti předchozí | **char b = ‘A’;** definice a inicializace proměnné b, která je osmibitová, může obsahovat znaky a je do ní vložen znak ‘A’. | **const double pi=3.14159265;** definice konstanty | **int pole[10];** deklarace pole celočíselných hodnot, s množstvím hodnot 10, První položka pole má index 0! nikoli 1! | **int pole1[5] = {1, 2, 3, 4, 5};** deklarace a inicializace celočíselného pole o pěti prvcích
SILNĚ TYPOVANÝ PROGRAMOVACÍ JAZYK PROVÁDÍ KONTROLU TYPŮ A ZJIŠTUJE JEJICH CHYBNÉ POUŽITÍ V PRŮBĚHU PROGRAMU (JAVA, C++, C)
SLABĚ TYPOVANÝ PROGRAMOVACÍ JAZYK NEPROVÁDÍ V PRŮBĚHU PROGRAMU KONTROLU TYPŮ – TEDY TO ZDA JE HODNOTA SPRÁVNÉHO TYPU PŘIŘAZENA DO SPRÁVNÉHO TYPU PROMĚNNÉ (JAVASCRIPT, PHP)
STATICKÉ TYPOVÁNÍ PROMĚNNÝCH – TYP PROMĚNNÉ JE URČEN V DOBĚ JEJÍ DEKLARACE. DATOVÉ TYPY PROMĚNNÝCH JSOU ZNÁMY JIŽ V PRŮBĚHU PŘEKLADU PROGRAMU
DYNAMICKY TYPOVANÉ PROMĚNNÉ – MAJÍ TYP ZÁVISLÝ NA TYPU HODNOTY, KTEROU MOMENTÁLNĚ PROMĚNNÁ OBSAHUJE. HODNOTY PROMĚNNÝCH VZNIKAJÍ AŽ V PRŮBĚHU PROGRAMU.
PŘETYPOVÁNÍ PŘEDSTAVUJE **ZMĚNU DATOVÉHO TYPU**.
IMPLICITNÍ PŘETYPOVÁNÍ JE PŘETYPOVÁNÍ, KTERÉ JE PROVÁDĚNO AUTOMATICKY PODLE KONTEXTU. NAPŘÍKLAD POKUD MÁ MÍT VÝSLEDEK HODNOTU VĚTŠÍ NEŽ JE MAXIMÁLNÍ ROZSAH PROMĚNNÉ, DO KTERÉ SE UKLÁDÁ, MŮŽE BÝT DANÁ PROMĚNNÁ IMPLICITNĚ PŘETYPOVÁNA NA TYP S VĚTŠÍM ROZSAHEM, DO KTERÉHO SE HODNOTA JIŽ VYJDE.
EXPLICITNÍ PŘETYPOVÁNÍ JE ZAPSÁNO PROGRAMÁTOREM VE ZDROJOVÉM KÓDU A URČUJE NA JAKÝ TYP SE PROMĚNNÁ PŘEVEDE NAPŘ. Promena1 = (double) Promena2
UKAZATEL JE PROMĚNNÁ, KTERÁ OBSAHUJE ADRESU PROMĚNNÉ URČITÉHO DATOVÉHO TYPU, UKAZUJE NA TUTO PROMĚNNOU (V JAZYCE C SE DEKLARUJE NAPŘ.: **int *promenna;** JEDNÁ SE O UKAZATEL NA PROMĚNNOU TYPU CELÉHO ČÍSLA.
promenna = &promenna2; PŘIŘADÍ ADRESU PROMĚNNÉ promenna2 DO UKAZATELE promenna.
promenna2=*promenna; PŘIŘADÍ HODNOTU NA ADRESE URČENOU UKAZATELEM promenna DO PROMĚNNÉ promenna2)
KDYŽ VYTVOŘÍME UKAZATEL NA NĚJAKOU PROMĚNNOU JE POTŘEBA PRO NÍ **REZERVOVAT PAMĚŤ** V JAZYCE C JE TO NAPŘÍKLAD: a=(int *)malloc(sizeof(int)) PŘEDSTAVUJE REZERVOVÁNÍ PAMĚTI PRO PROMĚNNOU a, KTERÁ JE UKAZATELEM NA HODNOTU OBSAHUJÍCÍ CELÉ ČÍSLO;
TERMÍNY: PROMĚNNÁ, KONSTANTA, DEKLARACE, DEFINICE, INICIALIZACE, IDENTIFIKÁTOR, LITERÁL, DATOVÝ TYP, SILNĚ A SLABĚ TYPOVANÝ, STATICKÉ A DYNAMICKÉ TYPOVÁNÍ PROMĚNNÝCH, PŘETYPOVÁNÍ, IMPLICITNÍ A EXPLICITNÍ PŘETYPOVÁNÍ, UKAZATEL, REZERVACE PAMĚTI.
PRAKTICKÁ DOVEDNOST: DEFINOVAT PROMĚNOU V JAZYCE C A POUŽÍT JI, DOKÁZAT TO VČETNĚ UKAZATELŮ, POLE A ŘETĚZCŮ

DOPORUČENÉ ZDROJE:
http://fsinet.fsid.cvut.cz/cz/U201/skrc.html
http://www.abclinuxu.cz/clanky/programovani-v-jazyce-d-3-typy-promenne-prace-s-cisly-literaly-a-funkce
https://support.dce.felk.cvut.cz/pos/cv1/jazykc.pdf

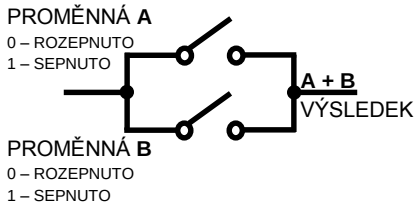
6. Operace, operátory, logické funkce, použití

MEZI PROMĚNNÝMI A PROMĚNNÝMI ČI KONSTANTAMI MŮŽEME PROVÁDĚT NEJRŮZNĚJŠÍ **OPERACE**. MOHOU BÝT MATEMATICKÉ, LOGICKÉ, BITOVÉ ČI JINÉ. **PRO ZÁPIS OPERACE** VE ZDROJOVÉM KÓDU PROGRAMOVACÍHO JAZYKU SLOUŽÍ **OPERÁTORY**. NAPŘÍKLAD: A+B UVÁDÍ OPERÁTOR +.
MEZI OPERÁTORY A ZNAKY POUŽÍVANÝMI V OPERACÍCH **EXISTUJE PRIORITA**. **NEJVYŠŠÍ PRIORITU** MAJÍ **VĚTŠINOU ZÁVORKY**, KTERÉ POMOHOU PŘEHLEDNĚ URČIT POŘADÍ JEDNOTLIVÝCH OPERACÍ.

MATEMATICKÉ OPERACE: SČÍTÁNÍ, ODČÍTÁNÍ, NÁSOBENÍ, DĚLENÍ, ZBYTEK PO CELOČÍSELNÉM DĚLENÍ, … (VĚTŠINOU PLATÍ PRO ČÍSELNÉ PROMĚNNÉ A KONSTANTY, ALE SČÍTAT SE NĚKDY MOHOU I ŘETĚZCOVÉ PROMĚNNÉ A KONSTANTY)
MATEMATICKÉ OPERÁTORY SE POUŽÍVAJÍ V MATEMATICKÝCH OPERACÍCH: (JAZYK C)+, -, /, *, %,…
LOGICKÉ OPERACE: LOGICKÝ SOUČET, LOGICKÝ SOUČIN, NEGACE **SE POUŽÍVAJÍ PRO VYTVÁŘENÍ PODMÍNEK, PŘI KTERÝCH SE VĚTVÍ PROGRAM**. POMOCÍ TĚCHTO LOGICKÝCH OPERACÍ MŮŽEME SESTAVIT JAKOUKOLIV LOGICKÝ VÝRAZ.
LOGICKÉ OPERÁTORY SE POUŽÍVAJÍ V LOGICKÝCH OPERACÍCH: (JAZYK C)&&, ||,
LOGICKÝ VÝRAZ JE VÝRAZ SESTAVENÝ Z **LOGICKÝCH PROMĚNNÝCH** A **LOGICKÝCH OPERÁTORŮ** A MŮŽE NABÝVAT DVOU LOGICKÝCH HODNOT: PRAVDA (TRUE, 1) ČI NEPRAVDA (FALSE, 0), JINAK **LOGICKOU JEDNIČKU ČI LOGICKOU NULU**.
PRO POPIS LOGICKÝCH FUNKCÍ A VÝRAZŮ SLOUŽÍ NAPŘÍKLAD **PRAVDIVOSTNÍ TABULKY**.
ZÁPIS LOGICKÝCH VÝRAZŮ: **LOGICKÝ SOUČET (+, OR, v), LOGICKÝ SOUČIN (., AND, ^, &), LOGICKÁ NEGACE (!, NOT, ¯)**

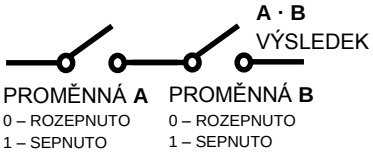
PRAVDIVOSTNÍ TABULKA PRO LOGICKÝ SOUČET:

A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1



PRAVDIVOSTNÍ TABULKA PRO LOGICKÝ SOUČIN:

A	B	A·B
0	0	0
0	1	0
1	0	0
1	1	1



JAKÉ DRUHY LOGICKÝCH OPERACÍ EXISTUJÍ: LOGICKÝ SOUČIN, NEGOVANÝ LOGICKÝ SOUČIN, LOGICKÝ SOUČET, NEGOVANÝ LOGICKÝ SOUČET, IMPLIKACE →, EKVIVALENCE ↔, EXKLUZIVNÍ LOGICKÝ SOUČET – XOR, …
JAK MŮŽEME LOGICKÉ FUNKCE ZAPISOVAT: SLOVNĚ, VZORCEM, PRAVDIVOSTNÍ TABULKOU, …
PRAVDIVOSTNÍ TABULKA: JSOU V NÍ UVEDENY VŠECHNY MOŽNÉ KOMBINACE VSTUPNÍCH HODNOT PROMĚNNÝCH A PŘÍSLUŠNÉ VÝSTUPNÍ HODNOTY. POČET KOMBINACÍ – ŘÁDKŮ TABULKY – JE ROVEN 2^N, KDE N JE POČET VSTUPNÍCH PROMĚNNÝCH.
BITOVÉ OPERACE: PŘEDSTAVUJÍ **OPERACE NA ÚROVNI BITŮ**. NAPŘÍKLAD: BITOVÝ SOUČET, BITOVÝ SOUČIN, BITOVÝ EXKLUZIVNÍ SOUČET, BUTOVÝ POSUN DOPRAVA A DOLEVA O N BITŮ, BITOVÉ ROTACE.

PŘÍKLAD BITOVÉHO SOUČTU 2 BINÁRNÍCH ČÍSEL:

A	0111
B	1001
VÝSLEDEK:	1111

PŘÍKLAD BITOVÉHO POSUVU O 2 BITY DOPRAVA

	00011001
VÝSLEDEK	00000110

RELAČNÍ OPERÁTORY: OPERÁTORY, KTERÉ POROVNÁVAJÍ DVĚ HODNOTY. JE ROVNO, NENÍ ROVNO, JE VĚTŠÍ, JE MENŠÍ, MENŠÍ NEBO ROVNO, VĚTŠÍ NEBO ROVNO. POMÁHÁ VYTVOŘIT LOGICKÉ FUNKCE: NAPŘÍKLAD (A JE ROVNO B MŮŽE MÍT LOGICKOU HODNOTU PRAVDA NEBO NEPRAVDA, TEDY 1 NEBO 0)
DRUHY OPERÁTORŮ: UNÁRNÍ – POUŽÍVÁ SE JEN NA JEDNU LOGICKOU PROMĚNNOU (+, -, !, ++, --), **BINÁRNÍ** – POUŽÍVÁ SE NA MINIMÁLNÍ DVĚ PROMĚNNÉ (+, -, *, /, =)
TERNÁRNÍ OPERÁTOR: JE PODMÍNĚNÝ VÝRAZ PRO ZKRÁCENÍ ZÁPISU VĚTVENÍ PROGRAMU – VYHODNOCENÍ URČITÉ PODMÍNKY, KTERÁ POKUD JE PRAVDIVÁ PROVEDE V PROGRAMU JEDNU SEKVENCI PŘÍKAZŮ A POKUD NENÍ PRAVDA, TAK PROVEDE JINOU SEKVENCI PŘÍKAZŮ. ZAVEDE PROGRAM JEDNÍM SMĚREM, JINAK DRUHÝM.
JAK TO FUNGUJE: PROMĚNNÁ = PODMÍNKA ? VÝRAZ1 : VÝRAZ2; VYHODNOTÍ SE PODMÍNKA, A POKUD JE PRAVDIVÁ, PROVEDE SE VÝRAZ1 JEHOŽ VÝSLEDEK BUDE VLOŽEN DO PROMĚNNÉ, JINAK SE PROVEDE VÝRAZ2
OPERACE A OPERÁTORY V JAZYCE C:
MATEMATICKÉ (+, -, = PŘIŘAZENÍ, /, *, ++, --, % - ZBYTEK PO CELOČÍSELNÉM DĚLENÍ, I++ (NEJPRVE JI POUŽIJE A PAK PŘIČTE), ++I (NEJPRVE PŘIČTE A PAK POUŽIJE VÝSLEDEK) – PŘIČTE K PROMĚNNÉ I JEDNIČKU), I--, --I ODEČTE OD PROMĚNNÉ JEDNIČKU, += PŘIČTE K PROMĚNNÉ NĚJAKOU HODNOTU A VRÁTÍ JI DO STEJNÉ PROMĚNNÉ, OBDOBNĚ: -=, *=, /=)
LOGICKÉ (&& LOGICKÝ SOUČIN, || LOGICKÝ SOUČET, ! LOGICKÁ NEGACE)
RELAČNÍ (== - JE ROVEN, != NENÍ ROVEN, < JE MENŠÍ, > JE VĚTŠÍ, <= JE MENŠÍ NEBO ROVNO, >= JE VĚTŠÍ NEBO ROVNO)
BITOVÉ (& BITOVÝ SOUČIN, | BITOVÝ SOUČET, >> BITOVÝ POSUN DOPRAVA, << BITOVÝ POSUN DOLEVA, ^ BITOVÝ XOR)
TERMÍNY: OPERACE, OPERÁTORY, PRIORITA, METEMATICKÉ, LOGICKÉ, BITOVÉ OPERACE, RELAČNÍ OPERÁTORY, VÝRAZ, TERNÁRNÍ OPERÁTOR, UNÁRNÍ A BINÁRNÍ OPERÁTORY, PRAVDIVOSTNÍ TABULKA, DRUHY LOGICKÝCH A OSTATNÍCH FUNKCÍ
PRAKTICKÁ DOVEDNOST: SESTAVIT PRAVDIVOSTNÍ TABULKU PRO ZADANOU LOGICKOU FUNKCI, SESTAVENÍ PODMÍNKY DO PODMÍNKY IF NEBO WHILE V JAZYCE C PODLE ZADÁNÍ: NAPŘÍKLAD PODMÍNKA, KTERÁ UKONČÍ CYKLUS V PŘÍPADĚ, ŽE A BUDE VĚTŠÍ NEŽ DESET.
DOPORUČENÉ ZDROJE: http://www.sallyx.org/sally/c/c10.php
http://gymnazium-milevsko.cz/gymnazium/stare/dokumenty/mat_ivt/logicke_operace.pdf
https://is.mendelu.cz/eknihovna/opory/zobraz_cast.pl?cast=7711

7. Funkce, knihovny

PŘI REALIZACI STRUKTUROVANÉHO PROGRAMOVÁNÍ, ROZDĚLUJEME PROGRAM DO MENŠÍCH ČÁSTÍ, KTERÉ USPOŘÁDÁVÁME DO STRUKTURY. TYTO **MENŠÍ ČÁSTI, KTERÉ MAJÍ VĚTŠINU JEDEN ÚČEL, JEDINOU FUNKCI** JSOU V PODSTATĚ **TAKOVÉ PODPROGRAMY**. POKUD PODPROGRAM NEVRACÍ ŽÁDNOU HODNOTU, POUZE REALIZUJE URČITÉ NASTAVENÍ V OPERAČNÍM SYSTÉMU APOD. MŮŽEME HO OZNAČIT ZA **PROCEDURU** (V JAZYCE C JE JI NEJBLÍŽE FUNKCE BEZ NÁVRATOVÉ HODNOTY).
FUNKCE VRACÍ URČITOU HODNOTU, VÝSLEDEK SVÉ ČINNOSTI.
FUNKCE V JAZYCE C: FUNKCE JSOU V JAZYCE C DŮLEŽITÉ. **UMOŽŇUJÍ OPAKOVANĚ POUŽÍT ČÁST KÓDU** A VYUŽÍT TAK VÍCEKRÁT JEDNOU VYTVOŘENÝ ALGORITMUS.
DEKLARACE FUNKCE: DEKLARACE FUNKCE **URČUJE VŠE PRO TO, ABYCHOM MOHLI FUNKCI POUŽÍT MIMO** FUNKCI: **NÁZEV, VSTUPNÍ PARAMETRY, NÁVRATOVOU HODNOTU**. DEKLARACE FUNKCE PŘEDSTAVUJE **HLAVIČKU FUNKCE**.
PŘ: DEKLARACE FUNKCE SOUČTU – TEDY HLAVIČKA FUNKCE int soucet(int scitanec1, int scitanec2)
NA ZAČÁTKU JE int, KTERÝ PŘEDSTAVUJE NÁVRATOVOU HODNOTU FUNKCE A V TOMTO PŘÍPADĚ TO ZNAMENÁ, ŽE VRACÍ CELÁ ČÍSLA. V KULATÝCH ZÁVORKÁCH JSOU UMÍSTĚNY PARAMETRY FUNKCE, TY MAJÍ V PŘÍKLADU NÁZEV scitanec1 a scitanec2.
PARAMETRY JSOU V ZÁVORCE ODDĚLENY ČÁRKAMI. FUNKCE TAKÉ NEMUSÍ MÍT ŽÁDNÝ PARAMETR.
NÁZEV FUNKCE SMÍ OBSAHOVAT **POUZE ALFANUMERICKÉ ZNAKY A PODTRŽÍTKO, PRVNÍ ZNAK NESMÍ BÝT ČÍSLICE**.
DEFINICE FUNKCE JE UVNITŘ SLOŽENÝCH ZÁVOREK, PŘEDSTAVUJE PŘESNÝ POPIS TOHO, CO A JAK FUNKCE DĚLÁ.
JAK SE VYTVÁŘÍ FUNKCE V JAZYCE C:
KAŽDÝ PROGRAM V JAZYCE C MÁ HLAVNÍ FUNKCI, KTERÁ SE JMENUJE main. **VŠE CO CHCEME SPUSTIT, SE MUSÍ VLOŽIT DO TÉTO FUNKCE.** FUNKCE main, KTERÁ JE TAKOVÝM HLAVNÍM VSTUPEM PROGRAMU, VYPADÁ TAKTO:

```
int main() // int PŘEDSTAVUJE NÁVRATOVOU HODNOTU, main PŘEDSTAVUJE NÁZEV FUNKCE,
// V KULATÝCH ZÁVORKÁCH JE OBYČEJNĚ SEZNAM VSTUPNÍCH PARAMETRŮ, ODDĚLENÝCH ČÁRKOU
{       // ZAČÁTEK OBSAHU FUNKCE OZNAČEN ZAČÁTKEM SLOŽENÝCH ZÁVOREK
        // ZDE SE OBJEVÍ PROGRAM, KTERÝ CHCEME VYKONAT
        return 0; // PŘÍKAZ RETURN UKONČUJE FUNKCI A VRACÍ JEJÍ VÝSLEDEK, TO CO VRACÍ, MUSÍ MÍT STEJNÝ TYP,
        //JAKO JE URČENÁ NÁVRATOVÁ HODNOTA V HLAVIČCE FUNKCE
        // JAKÝKOLIV PŘÍKAZ, KTERÝ JE UVEDEN ZA PŘÍKAZEM returnSE UŽ NEPROVEDE
}       // KONEC OBSAHU FUNKCE OZNAČEN KONCEM SLOŽENÝCH ZÁVOREK
```

V KAŽDÉM PROGRAMU, JE VSTUPNÍ **FUNKCE main PRÁVĚ A POUZE JEDNA**.
V JAZYCE C, MUSÍ BÝT FUNKCE PŘED JEJÍM POUŽITÍM, ALESPOŇ DEKLAROVÁNA.
BUĎ SE **NA ZAČÁTEK KÓDU VLOŽÍ VŠECHNY DEKLARACE FUNKCÍ**, KTERÉ SE **PAK DEFINUJÍ AŽ ZA HLAVNÍ FUNKCÍ main**
V JEDNOTLIVÝCH DEFINICÍCH FUNKCE **NEBO** SE **NA ZAČÁTKU PROGRAMU ROVNOU DEFINUJÍ VŠECHNY FUNKCE**, TO ZNAMENÁ, ŽE MAJÍ NA ZAČÁTKU UVEDENU JAK HLAVIČKU, TAK I TĚLO FUNKCE, KTERÉ DEFINUJE TO, CO DĚLAJÍ. **PRO PŘEHLEDNOST, JE LEPŠÍ POUŽÍVAT PRVNÍ ZPŮSOB**.

PŘÍKLAD DEFINICE FUNKCE V JAZYCE C:

```
float deleni(int citatel, int jmenovatel)
{
        float vysledek;  // ZDE NÁSLEDUJE OBSAH FUNKCE,
        // KONKRÉTNĚ ŘÁDKEM DEFINUJÍCÍM PROMĚNNOU
        // vysledek

        vysledek = citatel / jmenovatel;
        return vysledek; // ZDE FUNKCE VRACÍ VÝSLEDEK
}
```

PŘÍKLAD DEKLARACE FUNKCE V JAZYCE C:
float deleni(int citatel, int jmenovatel); FUNKCE KTERÁ SE JMENUJE deleni, MÁ NÁVRATOVOU HODNOTU float – DESETINNÉ ČÍSLO A DVĚ VSTUPNÍ HODNOTY TYPU int S NÁZVEM citatel A jmenovatel.

PŘÍKLAD VOLÁNÍ FUNKCE V JAZYCE C:

```
float vypocet;
int delenec = 5;
int delitel = 40;

vypocet = deleni(delenec, delitel);
```

DEKLARACE FUNKCE, KTERÁ NEVRACÍ NIC:
void openPorts(int portNumber); **void PŘEDSTAVUJE PRÁZDNÝ NÁVRATOVÝ TYP.**
I KDYBY FUNKCE NEMĚLA ŽÁDNÉ PARAMETRY, JE POTŘEBA NAPSAT KULATÉ ZÁVORKY, TŘEBAŽE JSOU PRÁZDNÉ.
PROMĚNNÉ DEKLAROVANÉ UVNITŘ FUNKCE JSOU **LOKÁLNÍ PROMĚNNÉ** A **MIMO FUNKCI NEJSOU DEFINOVANÉ**, TEDY NEEXISTUJÍ. **REKURZE** V PROGRAMOVÁNÍ NASTÁVÁ TEHDY, KDYŽ FUNKCE VE SVÉ DEFINICI VOLÁ SAMA SEBE NEBO KDYŽ VOLÁ FUNKCE, KTERÁ PAK VOLÁ ZASE JI SAMOTNOU. POUŽÍVÁ SE NAPŘÍKLAD PRO VÝPOČET FAKTORIÁLU NEBO HODNOT NEJRŮZNĚJŠÍCH MATEMATICKÝCH ŘAD.
KNIHOVNAMI, HLAVIČKOVÝMI SOUBORY, ROZUMÍME SOUBORY **OBSAHUJÍCÍ ZDROJOVÝ KÓD** – VĚTŠINOU **VÍCE DEFINOVANÝCH FUNKCÍ A PROMĚNNÝCH**, KTERÝ **ŘEŠÍ URČITÉ KONKRÉTNÍ OKRUH ÚLOH**, NAPŘÍKLAD VSTUP A VÝSTUP PROGRAMU.
DO PROGRAMU SE VKLÁDAJÍ POMOCÍ DIREKTIVY PREPROCESORU include. TAK NAPŘÍKLAD ZAČLENÍME DO PROGRAMU ÚPLNĚ NA ZAČÁTKU KNIHOVNU stdio.h NÁSLEDUJÍCÍ DIREKTIVOU: **#include <stdio.h>**. VŠECHNY **STANDARDNÍ KNIHOVNY SE UZAVÍRAJÍ DO ŠPIČATÝCH ZÁVOREK** – **HLEDAJÍ SE VE STANDARDNÍM ADRESÁŘI**. **KNIHOVNY**, KTERÉ MAJÍ **JMÉNA UZAVŘENA DO DVOJITÝCH UVOZOVEK**, SE **HLEDAJÍ V AKTUÁLNÍM ADRESÁŘI**. JSOU TO KNIHOVNY, KTERÉ VYTVOŘÍME SAMI.
PŘÍKLADY KNIHOVEN: string.h – PRÁCE S ŘETĚZCI, time.h – PRÁCE S ČASEM, math.h – PRÁCE S MATEMATICKÝMI FUNKCEMI, stdlib.h – PŘEVOD ŘETĚZCŮ, NÁHODNÁ ČÍSLA, ALOKACE PAMĚTI…
TERMÍNY: FUNKCE, PROCEDURA, DEKLARACE A DEFINICE FUNKCE, HLAVIČKA FUNKCE, NÁVRATOVÁ HODNOTA, PARAMETRY FUNKCE, main FUNKCE, return, REKURZE, LOKÁLNÍ PROMĚNNÉ, void, KNIHOVNA, HLAVIČKOVÉ SOUBORY, DIREKTIVY PREPROCESORU, include, INKLUDOVAT
PRAKTICKÁ DOVEDNOST: VLOŽENÍ POUŽÍVANÉ KNIHOVNY DO PROGRAMU, DEKLARACE A DEFINICE FUNKCE, VOLÁNÍ FUNKCE, POUŽITÍ FUNKCÍ printf A scanf Z KNIHOVNY stdio.h
DOPORUČENÉ ZDROJE: http://www.sallyx.org/sally/c/c13.php, http://www.sallyx.org/sally/c/c11.php, http://kmlinux.fjfi.cvut.cz/~fabiadav/cecko/poznamky-k-jazyku-c/funkce, http://www.gjszlin.cz/ivt/esf/algoritmizace/procedury-a-funkce-c.php

8. Programovací jazyk C

JAZYK C JE **JEDNÍM Z NEJROZŠÍŘENĚJŠÍCH PROGRAMOVACÍCH JAZYKŮ NA SVĚTĚ**. PRVNÍ NEBO DRUHÝ, PODLE RŮZNÝCH ZDROJŮ: ÚNOR 2016. NA PRVNÍM MÍSTĚ SE STŘÍDÁ S JAVOU. **JAZYK C** BYL **VYVINUT KOLEM ROKU 1972 DENNISEM RITCHIE**, ABY SE S JEHO POMOCÍ PŘEPSALO **JÁDRO OS UNIX**, DALŠÍ STARŠÍ PROJEKTY, KTERÉ JSOU DODNES ŽIVÉ A MAJÍ ČÁSTI Z JAZYKA C: **UNIX, ORACLE, WINDOWS, LINUX, MS SQL, OVLADAČE ZAŘÍZENÍ, GRAFICKÉ PROGRAMY, TEXTOVÉ A TABULKOVÉ PROCESORY, KOMPILÁTORY, INTERPRETY, ELEKTRONICKÁ ZAŘÍZENÍ**. 1989 – **STANDARDIZOVÁNÍ JAZYKA** C **ANSI C** – C89/C90 A NOVÁ Z ROKU 1999 - C99..

PROGRAMOVACÍ JAZYK C JE VĚTŠINOU ZAŘAZEN DO **VYŠŠÍCH PROGRAMOVACÍCH JAZYKŮ**. DÁLE PATŘÍ MEZI **STRUKTUROVANÉ JAZYKY**. PROGRAMOVACÍ JAZYK C JE KOMPILOVANÝ JAZYK. JAZYK C BYL NAPSÁN V ASSEMBLERU

MOŽNÁ VÝVOJOVÁ PROSTŘEDÍ PRO PRÁCI S JAZYKEM C: Code::Blocks, Dev-C++, CodeLite, Visual Studio, Eclipse, NetBeans

POSTUP KOMPILACE – PŘEKLADU (BUILD):

1. ZPRACOVÁNÍ **PREPROCESOREM** – ZPRACUJE DIREKTIVY PREPROCESORU (VKLÁDÁ HLAVIČKOVÉ SOUBORY A MAKRA, …)
2. **PARSOVÁNÍ** KÓDU A **PŘEKLAD** - KOMPILACE – PŘEVEDE KÓD DO ASSEMBLERU → KÓD V ASSEMBLERU
3. **ASSEMBLER** – PŘEVEDE KÓD TÉMĚŘ DO STROJOVÉHO KÓDU → OBJEKTOVÝ KÓD .o
4. **LINKER** SLINKUJE STROJOVÝ KÓD S FUNKCEMI OBSAŽENÝMI V POUŽITÝCH KNIHOVNÁCH → STROJOVÉ INSTRUKCE, SPUSTITELNÝ KÓD .exe

ZÁKLADNÍ STRUKTURA PROGRAMU:

ZAČÁTEK PROGRAMU VĚTŠINOU OBSAHUJE DIREKTIVY PREPROCESORU, URČUJÍCÍ, JAKÉ KNIHOVNY KÓDU BUDE PROGRAM POUŽÍVAT. **PŘÍKLAD PROGRAMU**:

```
#include <stdio.h>              // VLOŽENÍ KNIHOVNY, HLAVIČKOVÉHO SOUBORU, ABY ŠEL POUŽÍT PŘÍKAZ printf();
// ZDE MOHOU NÁSLEDOVAT DEKLARACE POUŽITÝCH FUNKCÍ
int main()                      // ZAČÁTEK HLAVNÍ FUNKCE main
{
        printf("Ahoj svete");   //POUŽITÍ FUNKCE Z KNIHOVNY stdio.h
        return 0;               //KONEC PROGRAMU, VRACÍ NÁVRATOVOU HODNOTU PŘÍKAZEM return
}
// ZDE MOHOU NÁSLEDOVAT DEFINICE FUNKCÍ, DEKLAROVANÝCH NA ZAČÁTKU PROGRAMU
```

NA KONCI KAŽDÉ ŘÁDKY V PROGRAMU MUSÍ BÝT STŘEDNÍK – VŠIMNĚTE SI!!!

PROGRAM SE SKLÁDÁ Z DIREKTIV PREPROCESSORU, VLOŽENÝCH KNIHOVEN, FUNKCE MAIN, VLASTNÍCH FUNKCÍ V NICH POUŽÍVANÝCH PROMĚNNÝCH A ŘÍDÍCÍCH STRUKTUR A KOMENTÁŘŮ

KOMENTÁŘE:

JEDNOŘÁDKOVÉ: // TOTO JE JEDNOŘÁDKOVÝ KOMENTÁŘ

VÍCEŘÁDKOVÉ KOMENTÁŘE: /* TOTO JE VÍCEŘÁDKOVÝ KOMENTÁŘ KTERÝ ZABÍRÁ VÍCE NEŽ JEDNU ŘÁDKU */

POUŽÍVANÉ PROMĚNNÉ MAJÍ **DATOVÉ TYPY**:

PŘÍKLAD DEFINICE PROMĚNNÝCH:

```
int počet, i, j, k;
float vypocet;
char znak = 'Z';
char znaky[4];  //pole znaků řetězec
float *ukazatel; //ukazatel na reálné číslo
```

CELOČÍSELNÉ: int, long int (DNES PRO ZÁPIS ČÍSLA STEJNÝ POČET BYTŮ JAKO int), unsigned int (NEZÁPORNÝ CELOČÍSELNÝ)

REÁLNÉ: float, double (DVOJITÁ PŘESNOST) – DESETINÁ ČÍSLA S PLOVOUCÍ ŘÁDOVOU ČÁRKOU

ZNAKOVÉ: char OBSAHUJE ZNAK, KTERÝ MÁ HODNOTU 0 - 255

POLE: POLE PROMĚNNÝCH O URČITÉM TYPU NAPŘÍKLAD CELÝCH ČÍSEL, DESETINÝCH, POKUD SE JEDNÁ O POLE ZNAKŮ, JE TO **ŘETĚZEC**: POLE ZNAKŮ

UKAZATELE: UKAZUJE NA ADRESU V PAMĚTI, KDE JE UMÍSTĚNA URČITÁ PROMĚNNÁ DANÉHO TYPU – UKAZATEL NA ČÍSLO…

ŘÍDÍCÍ STRUKTURY: JSOU STRUKTURY, KTERÉ ŘÍDÍ BĚH PROGRAMU:

STRUKTURY PRO VĚTVENÍ PROGRAMU:

```
if (a > 5) { b = b+5; OSTATNÍ PŘÍKAZY; } else if (…) { PŘÍKAZY…} else { PŘÍKAZY KTERÉ SE PROVEDOU, KDYŽ NEJSOU SPLNĚNY ŽÁDNÉ PODMÍNKY }
switch (vyraz)
{  case JEDNA Z MOŽNÝCH HODNOT VÝRAZU (CELOČÍSELNÁ, …):
       PŘÍKAZY, KTERÉ SE PROVEDOU POKAŽDÉ, KDYŽ MÁ VÝRAZ HODNOTU Z NEJBLIŽŠÍHO case PŘED;
       break; // PŘESKOČÍ ZBYTEK ŘÍDÍCÍ KONSTRUKCE A UKONČÍ JI.
   case JEDNA Z MOŽNÝCH HODNOT VÝRAZU (CELOČÍSELNÁ, …):
       PŘÍKAZY, KTERÉ SE PROVEDOU POKAŽDÉ, KDYŽ MÁ VÝRAZ HODNOTU Z NEJBLIŽŠÍHO case PŘED;
       break;
   default:
       PŘÍKAZY, KTERÉ SE PROVEDOU POKAŽDÉ, KDYŽ NIC JINÉHO NEPLATÍ;
       break;
}
```

STRUKTURY PRO CYKLY:

```
for (inicializace; podmínka; výraz)
     TĚLO CYKLU (DANÝ POČET OPAK.)

while ( PODMÍNKA )
    telo cyklu
CYKLUS S PODMÍNKOU NA ZAČ.
do
    telo cyklu
while ( PODMÍNKA );
CYKLUS S PODMÍNKOU NA KONC.
```

continue; PRO VYNUCENÍ DALŠÍHO OPAKOVÁNÍ CYKLU A PŘESKOČENÍ ZBYTKU

break; PRO OPUŠTĚNÍ ŘÍDÍCÍ BLOKU ŘÍDÍCÍ STRUKTURY (IF, SWITCH, WHILE, FOR, …)

JAZYK C MÁ **MOŽNOST VYTVÁŘET FUNKCE** A TÍM STRUKTUROVAT PROGRAM

ZNÁMÉ FUNKCE: printf, scanf, fopen… **OPERÁTORY**: UNÁRNÍ, BINÁRNÍ, RELAČNÍ, +-*/%, &&, ||

TERMÍNY: KOMPILACE, PREPROCESSOR, ASSEMBLER, LINKER, PROMĚNNÉ, OPERÁTORY, ŘÍDÍCÍ STRUKTURY, DATOVÉ TYPY, KOMENTÁŘE, UKAZATELE, POLE, ŘETĚZEC, KNIHOVNA, HLAVIČKOVÝ SOUBOR, CYKLUS, VĚTVENÍ PROGRAMU, …

PRAKTICKÁ DOVEDNOST: VYTVOŘIT JEDNODUCHÝ PROGRAM V JAZYCE C, DEFINOVAT, INICIALIZOVAT, NAČÍST A VYPSAT PROMĚNNÉ, POUŽÍVÁNÍ IF A SWITCH, STEJNĚ JAKO CYKLŮ FOR, WHILE, DO WHILE, INKLUDOVAT HLAVIČKOVÝ SOUBOR.

DOPORUČENÉ ZDROJE: http://nora.fpf.slu.cz/~bakala/U%E8ebnice%20jazyka%20C%5B1%5D.pdf, http://www.sallyx.org/sally/c/c03.php, http://kmlinux.fjfi.cvut.cz/~fabiadav/cecko/, http://www.arara.cz/product/184936

9. Objektově orientované programování - OOP

OMEZENÍ STRUKTUROVANÉHO PROGRAMOVÁNÍ: ODDĚLENÍ DAT A OPERACÍ S NIMI, LZE ŠPATNĚ POUŽÍT VE VELKÝCH SYSTÉMECH (ŠPATNĚ SE UDRŽUJE, ROZŠIŘUJE, DEBUGUJE), MÁ LIMITOVANÉ ZNOVUPOUŽITÍ KÓDU
VÝHODY OOP: PROGRAMÁTOROVI STAČÍ ZNÁT JEN ROZHRANÍ TŘÍD, KNIHOVEN, VYTVÁŘENÝ PROGRAM JE VÍCE MODULÁRNĚJŠÍ A SNADNĚJI SE DÁ ROZŠIŘOVAT, OPRAVOVAT I UDRŽOVAT, VYVYNUTÝ SOFTWARE JE KVALITNĚJŠÍ A PRODUKTIVITA PROGRAMÁTORA JE VYŠŠÍ.
HISTORIE + SOUČASNOST: PRVNÍ OBJEKTOVĚ ORIENTOVANÝ JAZYK 1962 – SIMULA, DNES NEJČASTĚJI POUŽÍVANÉ OBJEKTOVĚ ORIENTOVANÉ JAZYKY: JAVA, C++, C#, PYTHON, OBJECT PASCAL DNES JE OBJEKTOVĚ ORIENTOVANÉ I PHP

ZÁKLADNÍ VLASTNOSTI OOP:

ŘEŠENÍ PROBLÉMŮ OBJEKTOVĚ ORIENTOVANÝM PŘÍSTUPEM SE VÍCE PODOBÁ PŘÍSTUPU ČLOVĚKA K ŘEŠENÍ. PRACUJE SE S OBJEKTY, KTERÉ SDRUŽUJÍ VLASTNOSTI – DATA I METODY, S KTERÝMI PRACUJÍ. PROGRAM SE SKLÁDÁ Z MNOŽINY VZÁJEMNĚ SPOLUPRACUJÍCÍCH OBJEKTŮ.
ZAPOUZDŘENÍ (ENCAPSULATION): ZAPOUZDŘENÍ UMOŽŇUJE PŘED OSTATNÍMI OBJEKTY ZATAJIT VNITŘNÍ STAV OBJEKTU. PŘI NÁHODNÝCH OPERACÍCH NEMOHOU JINÉ OBJEKTY MĚNIT STAV OBJEKTU PŘÍMO A ZANÉST DO NĚJ NECHTĚNÉ CHYBY.
DĚDIČNOST (INHERITANCE): TŘÍDY NA NIŽŠÍ ÚROVNI MOHOU DĚDIT VLASTNOSTI A CHOVÁNÍ TŘÍD NACHÁZEJÍCÍCH SE NA VYŠŠÍ ÚROVNI. NOVOU TŘÍDU MŮŽEME ODVODIT Z PODOBNÉ EXISTUJÍCÍ TŘÍDY PŘIDÁNÍM POŽADOVANÝCH FUNKCÍ A VLASTNOSTÍ. PŮVODNÍ TŘÍDA = RODIČOVSKÁ TŘÍDA, ODVOZENÁ TŘÍDA = TŘÍDA POTOMKA.
POLYMORFISMUS (POLYMORPHISM): JEDNOTNÉ ZACHÁZENÍ (VOLÁNÍ, ROZHRANÍ) RŮZNÝCH (POLYMORFNÍCH) OBJEKTŮ, KTERÉ MAJÍ NĚKTERÉ SPOLEČNÉ ZDĚDĚNÉ VLASTNOSTI. JE TAK MOŽNÉ MÍT RŮZNÉ TŘÍDY SE STEJNĚ NAZVANÝMI METODAMI, KTERÉ MAJÍ STEJNÉ PRARAMETRY (ROZHRANÍ) ALE MAJÍ ROZDÍLNOU FUNKČNOST. ROZHRANÍ SE MŮŽE V NĚKTERÝCH PŘÍPADECH LIŠIT.

ZÁKLADNÍ TERMÍNY A JEJICH POUŽITÍ:

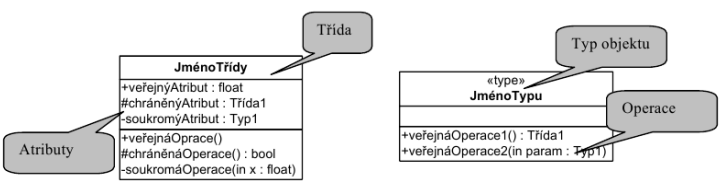
TŘÍDA (CLASS): JE ŠABLONA, PODLE KTERÉ SE VYTVÁŘEJÍ KONKRÉTNÍ OBJEKTY (JEJÍ INSTANCE), SE KTERÝMI SE POZDĚJI PRACUJE. KAŽDÝ OBJEKT MÁ STEJNOU STRUKTURU A CHOVÁNÍ JAKO TŘÍDA, JEJÍŽ JE INSTANCÍ. TŘÍDA JE TO CO NAVRHUJEME, PROGRAMUJEME. **PŘEDEK** TŘÍDY JE TŘÍDA, Z KTERÉ SE DĚDÍ ATRIBUTY A METODY. **POTOMEK** TŘÍDY JE TŘÍDA, KTERÁ DĚDÍ Z DANÉ TŘÍDY ATRIBUTY A METODY.
OBJEKT, INSTANCE TŘÍDY (INSTANCE): VYTVÁŘÍ SE ZE „ŠABLONY“ V PRŮBĚHU PROGRAMU“. S OBJEKTEM V PRŮBĚHU PROGRAMU PRACUJEME A VOLÁME JEHO METODY A MODIFIKUJEME JEHO ATRIBUTY. Z JEDNÉ TŘÍDY MŮŽEME VYTVOŘIT LIBOVOLNÝ POČET OBJEKTŮ. STAV PROMĚNNÝCH JEDNÉ INSTANCE JE NEZÁVISLY NA STAVU PROMĚNNÝCH DRUHÉ INSTANCE JEDNÉ A TÉ SAMÉ TŘÍDY.
ATRIBUTY (NEBO VLASTNOSTI, PROMĚNNÉ, DATOVÉ POLOŽKY) OBJEKTU: JSOU ČLENY TŘÍDY A PŘEDSTAVUJÍ JEJÍ VLASTNOSTI. JSOU VĚTŠINOU MODIFIKOVÁNY METODAMI TŘÍDY. PŘI VYTVOŘENÍ INSTANCE TŘÍDY SE INICIALIZUJÍ HODNOTY ATRIBUTŮ A MŮŽE SE PRO NĚ ALOKOVAT MÍSTO V PAMĚTI.
STATICKÉ PROMĚNNÉ (ATRIBUTY): JSOU ATRIBUTY SPOLEČNÉ PRO VŠECHNY INSTANCE Z JEDNÉ TŘÍDY. NEJSOU VÁZÁNY NA KONKRÉTNÍ INSTANCI. POUŽÍVAJÍ SE PRO SLEDOVÁNÍ STAVU INSTANCÍ, NAPŘÍKLAD JE POČÍTAJÍ, …
METODY: JSOU ČLENY TŘÍDY A PRACUJÍ S VNITŘNÍM STAVEM INSTANCE, UPRAVUJÍ JEJÍ ATRIBUTY. JEDNÁ SE V PODSTATĚ O FUNKCE, KTERÉ JSOU SPOLU S ATRIBUTY DEFINOVÁNY VE TŘÍDĚ. ATRIBUTY INSTANCE TŘÍDY LZE VOLAT AŽ PO VYTVOŘENÍ KONKRÉTNÍ INSTANCE TŘÍDY. EXISTUJÍ I **STATICKÉ METODY** URČENÉ PRO PRÁCI SE STATICKÝMI PROMĚNNÝMI.
KONSTRUKTOR: MÁ ZA ÚKOL INICIALIZOVAT STAV OBJEKTU – INSTANCE, JEJÍ PROMĚNNÉ A ALOKOVAT PRO NĚ MÍSTO V PAMĚTI. JE VOLÁN AUTOMATICKY PŘI VYTVÁŘENÍ INSTANCE TŘÍDY. MŮŽE BÝT BEZPARAMETRICKÝ NEBO PARAMETRICKÝ, PODLE TOHO ZDA MÁ ČI NEMÁ PARAMETRY.
DESTRUKTOR: VOLÁN PŘI RUŠENÍ INSTANCE OBJEKTU. VĚTŠINOU DEALOKUJE PAMĚŤ PŘIDĚLENOU OBJEKTU.
MODIFIKÁTORY PŘÍSTUPU KE ČLENŮM TŘÍDY: USKUTEČNĚNÍ PRINCIPU ZAPOUZDŘENÍ – ŘÍDÍ – POVOLUJE / ZAKAZUJE PŘÍSTUP KE ČLENŮM TŘÍDY: PROMĚNNÝM I METODÁM.
PUBLIC: PŘÍSTUP Z JAKÉKOLIV TŘÍDY
PRIVATE: PŘÍSTUP POUZE Z VLASTNÍ TŘÍDY, KDE BYLA DEKLAROVÁNA
PROTECTED: PŘÍSTUP Z VLASTNÍ TŘÍDY, A ZE TŘÍD POTOMKŮ NA ZAČÁTKU TŘÍDY SE VĚTŠINOU ŘADÍ PRVNÍ PROMĚNNÉ A METODY, KTERÉ JSOU PUBLIC.
PŘETĚŽOVÁNÍ METOD: V JEDNÉ TŘÍDĚ LZE DEFINOVAT VÍCE METOD STEJNÉHO NÁZVU, MUSÍ SE VŠAK LIŠIT TYPEM A POČTEM ARGUMENTŮ. TO PLATÍ I PRO KONSTRUKTORY.
VOLÁNÍ METOD A POUŽÍVÁNÍ ATRIBUTŮ: PŘED VOLÁNÍM METODY NEBO POUŽITÍM ATRIBUTU MUSÍME MÍT NEJPRVE VYTVOŘENU INSTANCI. A PAK JE MŮŽEME VOLAT.

```
penezenka mojePenezenka; // VYTVOŘENÍ INSTANCE TŘÍDY penezenka S NÁZVEM mojePenezenka
mojePenezenka.nabij(100);  // VOLÁNÍ METODY OBJEKTU mojePenezenka nabij S PARAMETRY
mojePenezenka.mod = 5;     // ZMĚNA HODNOTY ATRIBUTU mod OBJEKTU mojePenezenka
```

TERMÍNY: ZAPOUZDŘENÍ, DĚDIČNOST, POLYMORFISMUS, TŘÍDA, PŘEDEK, POTOMEK, INSTANCE TŘÍDY = OBJEKT, METODA, ATRIBUT, STATICKÉ PROMĚNNÉ, STATICKÉ METODY, KONSTRUKTOR, DESTRUKTOR, MODIFIKÁTORY PŘÍSTUPU, PUBLIC, PRIVATE, PROTECTED, PŘETĚŽOVÁNÍ
PRAKTICKÁ DOVEDNOST: VYTVOŘIT JEDNODUCHOU TŘÍDU V JAZYCE C++, VYTVOŘIT JEJÍ INSTANCI A POUŽÍT JEJÍ METODY A ATRIBUTY.

DOPORUČENÉ ZDROJE:

https://is.muni.cz/el/1433/podzim2014/IB111/um/IB111_OOP.pdf
http://www1.osu.cz/~fojtik/cpp/cpp01.pdf
http://www.ncbr.muni.cz/~martinp/C3220/PCChem_EX2.pdf
https://web.natur.cuni.cz/~bayertom/Prog2/prog2_1.pdf



10. Návrhové vzory

V ANGLIČTINĚ DESIGN PATTERNS
NÁVRHOVÉ VZORY JSOU OBDOBOU MATEMATICKÝCH VZORECKŮ PRO PROGRAMÁTORY. NÁVRHOVÉ VZORYVYCHÁZEJÍ ZE ZKUŠENOSTÍ, KDY OPAKOVANĚ SPOLEHLIVĚ FUNGOVALY A TAK, PROTOŽE SE JEJICH ZPŮSOB ŘEŠENÍ OSVĚDČIL, EXISTUJE TU SNAHA JE POUŽÍVAT ZNOVU A VYLEPŠOVAT JE. VZOR JE TAKOVÁ ŠABLONA PRO ŘEŠENÍ, NIKOLI IMPLEMENTACE ŘEŠENÍ.
VÝHODY NÁVRHOVÝCH VZORŮ: ZNOVUPOUŽITELNOST

1. NÁVRH SE ZRYCHLÍ (ŘEŠENÍ SE JEN POUŽIJE BEZ PŘEMÝŠLENÍ)
2. ZKVALITNÍ NÁVRH
3. ZJEDNODUŠUJÍ A ZPŘESŇUJÍ KOMUNIKACI MEZI ČLENY TÝMU (PODMÍNKOU JE ZNALOST)
4. JSOU JIŽ OVĚŘENÝMI ŘEŠENÍMI A TAK SNIŽUJÍ MOŽNOST POTENCIÁLNÍCH CHYB.

PRVKY NÁVRHOVÉHO VZORU:

- NÁZEV – KRÁTKÝ POPIS NÁVRHOVÉHO PROBLÉMU
- PROBLÉM – PODMÍNKY POTŘEBNÉ PRO SMYSLUPLNÉ POUŽITÍ VZORU
- ŘEŠENÍ – ABSTRAKTNÍ POPIS PROBLÉMU A POPIS PRVKŮ NÁVRHU, VZTAHŮ …
- DŮSLEDKY – VÝSLEDKY A KOMPROMISY PŘI JEJICH POUŽITÍ (VLIV NA ROZŠIŘITELNOST A PŘENOSITELNOST)

VZORY SE MOHOU TÝKAT TŘÍD (VZTAHY MEZI TŘÍDAMI A PODTŘÍDAMI) A NEBO OBJEKTŮ (VZATHY MEZI OBJEKTY)
ZÁKLADNÍ ROZDĚLENÍ VZORŮ:

- **TVOŘIVÝ** (TÝKÁ SE PROCESU TVORBY OBJEKTŮ)
  - **TOVÁRNÍ METODA (FACTORY METHOD)**, ABSTRAKTNÍ TOVÁRNA (ABSTRACT FACTORY), **JEDINÁČEK (SINGLETON)**, PROTOTYP (PROTOTYPE), STAVITEL (BUILDER)
- **STRUKTURÁLNÍ** (TÝKÁ SE VNITŘNÍ STRUKTURY TŘÍD ČI OBJEKTŮ)
  - ADAPTÉR (ADAPTER), DEKORÁTOR (DECORATOR), FASÁDA (FACADE), MOST (BRIDGE), MUŠÍ VÁHA (FLYWEIGHT), SKLADBA (COMPOSITE), ZÁSTUPCE (PROXY)
- **CHOVÁNÍ** (TÝKÁ SE ZPŮSOBŮ VZÁJEMNÉ KOMUNIKACE MEZI OBJEKTY NEBO TŘÍDAMI)
  - INTERPRET (INTERPRETER), ŠABLONOVÁ METODA, ITERÁTOR (ITERATOR), NÁVŠTĚVNÍK (VISITOR), OBNOVITEL (MEMENTO), POZOROVATEL (OBSERVER), PROSTŘEDNÍK (MEDIATOR), **PŘÍKAZ (COMMAND)**, ŘETĚZ ODPOVĚDNOSTI (CHAIN OF RESPONSIBILITY), STAV (STATE), STRATEGIE (STRATEGY)

PŘÍKLADY NÁVRHOVÝCH VZORŮ:
NÁZEV: JEDINÁČEK - SINGLETON
PROBLÉM: PŘEDSTAVUJE TŘÍDU, KTERÁ MŮŽE MÍT NANEJVÝŠ JEDNU GLOBÁLNĚ SDÍLENOU INSTANCI. SAMA TŘÍDA CENTRALIZUJE K TÉTO SVÉ INSTANCI PŘÍSTUP. JEDNÁ SE NAPŘÍKLAD O DATABÁZOVÉ PŘIPOJENÍ, KDY CELÝ PROGRAM POTŘEBUJE JEDNO JEDINÉ PŘIPOJENÍ NEBO TISKOVÁ FRONTA.
ŘEŠENÍ:

- UŽIVATELI ZAKÁŽEME TVOŘIT INSTANCI A TO TAK, ŽE IMPLEMENTUJEME PRÁZDNÝ PRIVÁTNÍ KONSTRUKTOR. KE KONSTRUKTORU SE TEDY DOSTANEME POUZE POMOCÍ JINÉ METODY, KTERÁ BUDE VEŘEJNÁ.
- VYTVOŘÍME BĚŽNOU INSTANČNÍ PROMĚNNOU A DO NÍ VLOŽÍME INSTANCI, KTEROU CHCEME SDÍLET.
- TŘÍDA SI VYTVOŘÍ INSTANCI SAMA SEBE A ULOŽÍ JI DO STATICKÉ PROMĚNNÉ. INSTANCI TEDY SPRAVUJE TŘÍDA A UŽIVATEL SE K NÍ NEDOSTANE JINAK NEŽ PŘES TŘÍDU. INSTANCI NASTAVÍME JAKO PRIVÁTNÍ.
- VYTVOŘÍME VEŘEJNOU METODU, KTERÁ BUDE MÍT PŘÍSTUP K INSTANCI A BUDE JI MOCI VRACET.

DALŠÍ VÝZNAMNÝ NÁVRHOVÝ VZOR JE VZOR FACTORY, JINAK TOVÁRNÍ METODA. TENTO NÁVRHOVÝ VZOR UPRAVUJE ZPŮSOB VYTVÁŘENÍ OBJEKTU. UMOŽŇUJE, ABY SI ZPŮSOB VYTVOŘENÍ OBJEKTU ROZHODLI SAMI POTOMCI TŘÍDY PODLE TOHO, JAK BUDE POTŘEBOVAT. TENTO ZPŮSOB UMOŽNÍ ODDĚLIT KÓD, KTERÝ SE NEMĚNÍ TAK ČASTO, OD KÓDU, KTERÝ SE POMĚRNĚ ČASTO MĚNÍ.

PŘÍKLAD SINGLETON:

```
public final class Ucet {
        private int cislo;
        private int stav;
        private static Ucet ucetSingleton; // tridni, staticka promenna
        // vsechny konstruktory jsou private
        private Ucet(int cislo, int stav)
        { this.cislo = cislo; this.stav = stav; }

public static Ucet getInstance(){
        // metoda pro ziskani instance uctu
        if(ucetSingleton == null) ucetSingleton = new Ucet(1, 0);
        return ucetSingleton;
}
```

FRAMEWORKY A JEJICH VYUŽITÍ V JAZYCE C++:
FRAMEWORK JE V PODSTATĚ SOUBOR KNIHOVEN, KTERÉ JSOU TÉMATICKY ZAMĚŘENÉ NA URČITOU OBLAST, KTEROU ŘEŠÍ. FRAMEWORK DEFINUJE I ZPŮSOBY PRÁCE JAKÝM ZPŮSOBEM URČITÉ VĚCI ŘEŠIT. KDYŽ JE TO JASNĚ DANÉ MŮŽE SNADNĚJI PRACOVAT CELÝ TÝM PROGRAMÁTORŮ.

TERMÍNY: NÁVRHOVÉ VZORY, DESIGN PATTERNS, FACTORY, SINGLETON, COMMAND, ZNOVUPOUŽITELNOST, FRAMEWORK
PRAKTICKÁ DOVEDNOST: POUŽITÍ JEDNODUCHÉHO NÁVRHOVÉHO VZORU A VYSVĚTLENÍ JEHO FUNKČNOSTI - SINGLETON
DOPORUČENÉ ZDROJE: http://www.itnetwork.cz/navrhove-vzory/singleton-navrhovy-vzor, http://programujte.com/clanek/2012032900-serial-navrhovych-vzoru-1-dil/, http://www1.osu.cz/~hunka/vyuka/javaOOP/obop2/jvcv10/Vzory_2.pdf, http://voho.cz/wiki/factory-method/, http://voho.cz/wiki/singleton/, http://majda.cz/blog/265

11. Programovací jazyk C++

JAZYK C++ A JEHO HISTORIE: PROGRAMOVACÍ JAZYK C++ PATŘÍ MEZI NEJČASTĚJI POUŽÍVANÝ PROGRAMOVACÍ JAZYK NA SVĚTĚ. VYCHÁZÍ Z JAZYKA C. 1983 – NÁZEV C++. INSPIROVAL VZNIK JAZYKŮ JAVA A C#(SÍ ŠARP). PROŠEL TŘEMI FÁZEMI STANDARDIZACE: 1998, 2003 A 2011.
VÝHODY: ŠIROCE POUŽÍVANÝ = SPOUSTU JIŽ VYŘEŠENÝCH PROBLÉMŮ A KVALITNÍCH POSTUPŮ, OOP PŘÍSTUP, ZÁKLAD V JAZYCE C- ČASEM OVĚŘENÝ, RYCHLOST PROGRAMŮ – JE KOMPILOVANÝ, PŘEKLADAČE PRO VŠECHNY MOŽNÉ HW PROSTŘEDKY.
POUŽITÍ: JÁDRO MNOHA POČÍTAČOVÝCH HER, ADOBE PHOTOSHOP, ILLUSTRATOR, PREMIERE, FIREFOX, THUNDERBIRD, MYSQL, MAYA 3D, WINAMP MEDIA PLAYER,
ZDROJOVÉ SOUBORY V JAZYCE C++ MAJÍ PŘÍPONU: .cpp NEBO .c, .cc
ZPŮSOB VYTVÁŘENÍ TŘÍD, OBJEKTŮ, VOLÁNÍ METOD, POUŽÍVÁNÍ ATRIBUTŮ

```
class Obdelnik {                        // začátek deklarace třídy Obdelnik
        int sirka, vyska;               // atributy třídy Obdelnik celočíselného typu
public:                                 // modifikátor přístupu ke členům třídy public znamená, že jsou volně přístupné odkudkoli
        Obdelnik (int x,int y);         // deklarace konstruktoru třídy Obdelnik
        int plocha() { return sirka*vyska; }   // definice metody třídy Obdelnik s nazvem plocha
};                                      // konec deklarace třídy Obdelnik
Obdelnik::Obdelnik (int x, int y) {     // definice konstruktoru třídy Obdelnik
        sirka = x;                      // metoda pracuje s atributy třídy
        vyska = y;
}
int main () {                           // hlavní funkce main
        Obdelnik obd(3,4);              // vytvoření instance třídy Obdelnik s názvem obd, zavolá konstruktor a nastaví hodnoty
        cout << "plocha: " << obd.plocha();   // vypsání návratové hodnoty metody plocha objektu obd
        return 0;
}
```

VSTUPY A VÝSTUPY V JAZYCE C++
PRO POUŽITÍ VSTUPŮ A VÝSTUPŮ Z KLÁVESNICE MUSÍME INKLUDOVAT HLAVIČKOVÝ SOUBOR IOSTREAM.
VÝSTUP – TISK NA OBRAZOVKU: **cout << "Vytisteny text ";** pokud není deklarován prostor jmen **using namespace std;** Budeme muset použít ve formě **std::cout << "Vytisteny text";** VÝSTUP SE MŮŽE ŘETĚZIT: **cout << "Text1" << "Text2" << promena << endl;** PŘI VÝSTUPU SE KONVERTUJÍ VŠECHNY PROMĚNNÉ I CELOČÍSELNÉ A REÁLNÉ NA TEXTOVÉ ŘETĚZCE. **ENDL** PŘEDSTAVUJE KONEC ŘÁDKU.
VSTUP – VSTUP Z KLÁVESNICE: cin >> text; VLOŽÍ VSTUP Z KLÁVESNICE DO PROMENNÉ S NÁZVEM text. VSTUPY Z KLÁVESNICE SE MOHOU TAKÉ ŘETĚZIT: **cin >> text1 >> text2 >> c;** TAKTO PŘEČTE VSTUP Z KLÁVESNICE POSTUPNĚ DO PROMĚNNÝCH V POŘADÍ ZLEVA DOPRAVA. text1, text2, c. ZADÁNÍ JEDNOTLIVÝCH PROMĚNNÝCH LZE ODDĚLIT NETISKNUTELNÝMI ZNAKY JAKO NAPŘÍKLAD ODŘÁDKOVÁNÍ, TABULÁTOR, MEZERA. PRO POUŽITÍ FUNKCE BEZ std:: JE TAKÉ POTŘEBA DEKLAROVAT JMENNÝ PROSTOR POMOCI using namespace std;
DATOVÉ TYPY: PO PŘIPOJENÍ HLAVIČKOVÉHO SOUBORU string LZE POUŽÍVAT DATOVÝ TYP string JAKO JINÉ DATOVÉ TYPY, STRINGY – ŘETĚZCE SE MOHOU ROVNAT, PŘIŘAZOVAT, POROVNÁVAT, KTERÝ JE VĚTŠÍ, SČÍTAT. DÁLE EXISTUJE DATOVÝ TYP bool. JE TO LOGICKÝ TYP NABÝVAJÍCÍCH HODNOT true A NEBO false.
KONSTRUKTOR V JAZYCE C++: KONSTRUKTOR MÁ STEJNÉ JMÉNO JAKO TŘÍDA, KE KTERÉ PATŘÍ. MŮŽE BÝT BEZPARAMETRICKÝ NEBO PARAMETRICKÝ. JAKO METODA, KTERÁ NEMÁ A MÁ PARAMETRY. KONSTRUKTOR SE VOLÁ PŘI KAŽDÉM VYTVÁŘENÍ INSTANCE TŘÍDY. POUŽÍVÁ SE VĚTŠINOU K NASTAVENÍ VNITŘNÍCH PROMĚNNÝCH OBJEKTU A PODOBNĚ. DEKLARACE MŮŽE VYPADAT NAPŘÍKLAD: *Obdelnik (int,int);*
DESTRUKTOR V JAZYCE C++: DESTRUKTOR JE VOLÁN AUTOMATICKY PO ZRUŠENÍ INSTANCE TŘÍDY. JMÉNO DESTZRUKTORU JE TVOŘENO ZNAKEM ~ A JMÉNEM TŘÍDY. DESTRUKTORY SE POUŽÍVAYJÍ HLAVNĚ PRO UVOLNĚNÍ DYNAMICKY ALOKOVANÉ PAMĚTI. DESTRUKTORY NEMAJÍ NÁVRATOVOU HODNOTU, ANI NEMOHOU PŘIJÍMAT PARAMETRY. DEKLARACE DESTRUKTORU: ~Obdelnik();
ŠTÁBNÍ KULTURA – CODING STANDARD: POUŽÍVANÉ ZVYKLOSTI PŘI VYTVÁŘENÍ NÁZVŮ. JMÉNA TŘÍD SE ZAČÍNAJÍ VELKÝM PÍSMENEM, JMÉNA ATRIBUTŮ – DATOVÝCH PROMĚNNÝCH OBJEKTŮ A METOD ZAČÍNAJÍ MALÝM PÍSMENEM. POKUD SE NÁZEV PROMĚNNÉ NEBO METODY SKLÁDÁ Z VÍCE SLOV, TAK KAŽDÉ DALŠÍ SLOVO ZAČÍNÁ VELKÝM PÍSMENEM – CAMELCASE.
DEFINICE TŘÍDY POTOMKA: Př: **class Kruh : public Grafika** TŘÍDA Kruh JE POTOMKEM TŘÍDY Grafika A DĚDÍ OD NÍ.
METODY POTOMKŮ MAJÍ PŘÍSTUP JEN K TĚM ATRIBUTŮM A METODÁM PŘEDKŮ, KTERÉ JSOU OZNAČENY public NEBO protected. PŘÍSTUP K ATRIBUTŮM A METODÁM PŘEDKA JE STEJNÝ JAKO K VLASTNÍM, POUZE JE ZAVOLÁM, JAKOBYCH VOLAL VLASTNÍ METODU TŘÍDY POTOMKA, A POKUD JSOU V PŘEDKOVI DEFINOVÁNY, PROVEDOU SE.
POKUD MÁ POTOMEK STEJNÝ NÁZEV METODY JAKO PŘEDEK, TAK POTOMEK PŘEKRYJE METODU PŘEDKA SVOJI VLASTNÍ A V PŘÍPADĚ JEJÍHO VOLÁNÍ VOLÁ VLASTNÍ METODU A NIKOLIV METODU DEFINOVANOU V PŘEDKOVI.
TERMÍNY: ZDROJOVÉ SOUBORY, ZDROJOVÝ KÓD, TŘÍDA, OBJEKT, INSTANCE, METODA, ATRIBUT, MODIFIKÁTOR PŘÍSTUPU, KONSTRUKTOR, DESTRUKTOR, ŠTÁBNÍ KULTURA, CODING STANDARD, CAMELCASE, DEKLARACE, DEFINICE, POTOMEK, PŘEDEK, VSTUP, VÝSTUP, NÁVRATOVÁ HODNOTA, HLAVIČKOVÝ SOUBOR, PŘEKRYTÍ
PRAKTICKÁ DOVEDNOST: – NAPSAT DEFINICI TŘÍDY A JEJÍCH METOD, ZAPSAT KÓD PRO VYPSÁNÍ V C++ NA OBRAZOVKU NĚJAKÉHO TEXTU NEBO PROMĚNNÉ, ZAPSAT ZDROJOVÝ KÓD, KTERÝ NAČTE DO PROMĚNNÝCH VSTUP Z KLÁVESNICE.
DOPORUČENÉ ZDROJE: http://www.ncbr.muni.cz/~martinp/C3220/, http://www.tutorialspoint.com/cplusplus/, http://www.cplusplus.com/doc/tutorial/

12. Deklarativní programování a jazyk SQL

DEKLARATIVNÍ PROGRAMOVÁNÍ SE ZABÝVÁ TÍM, CO SE MÁ UDĚLAT (VYŘEŠIT), ALE NIKOLIV JAK TO VYŘEŠIT. CELÝ PROGRAM JE BRÁN JAKO FUNKCE, KTERÝ NA ZÁKLADĚ VSTUPŮ VRÁTÍ VÝSTUP. DEKLARATIVNÍ JAZYKY SPECIFIKUJÍ CÍL A ALGORITMIZACI PONECHÁVAJÍ PROGRAMU (INTERPRETU) DANÉHO JAZYKA.
DEKLARATIVNÍ JAZYKY DĚLÍME NA: FUNKCIONÁLNÍ (LISP) – RELACE JSOU VYJÁDŘENY FUNKCEMI, **LOGICKÉ** (PROLOG) – RELACE VYJÁDŘENY LOGICKÝMI PRAVIDLY, **DOTAZOVACÍ** (SQL) – RELACE URČENÉ KE ZPRACOVÁNÍ DAT
SQL = STRUCTURED QUERY LANGUAGE PROGRAMOVACÍ JAZYK PRO PRÁCI S RELAČNÍMI DATABÁZEMI. PRVNÍ KOMERČNÍ VERZE 1979 – ORACLE. STANDARDIZOVÁN ANSI 1986, ISO 1987. SQL VYTVÁŘÍ JAK STRUKTURU DATABÁZE A TABULEK TAK DOKÁŽE I MANIPULOVAT S DATY UVNITŘ NICH.
NEJZNÁMĚJŠÍ IMPLEMENTACE SQL: MICROSOFT SQL SERVER, MySQL, Oracle SQL, Postgre SQL…
ZÁKLADNÍ POJMY DATABÁZÍ: DATABÁZE JE SOUBOR DAT SLOUŽÍCÍ K POPISU REÁLNÉHO SVĚTA – OBSAHUJE DATA POPISUJÍCÍ TENTO SVĚT. **ENTITA** JE PRVKEM REÁLNÉHO SVĚTA, V DATABÁZI URČEN JEDNÍM ZÁZNAMEM. KAŽDÁ ENTITA MÁ VLASTNOSTI. (ČLOVĚK). **ATRIBUTY** PŘEDSTAVUJÍ VLASTNOSTI ENTITY, V PODSTATĚ SLOUPCE TABULKY. (JMÉNO, PŘÍJMENÍ, STAV, PLAT)
MEZI ENTITAMI EXISTUJÍ VAZBY URČITÉ VZTAHY. NAPŘÍKLAD **VAZBA 1:1** – KDY KAŽDÝ ČLOVĚK MÁ PRÁVĚ JEDNO RODNÉ ČÍSLO. NEBO EXISTUJE **VAZBA 1:N**, KDY KAŽDÝ ČLOVĚK MŮŽE MÍT N KREDITNÍCH KARET. ALE JEDNA KREDITNÍ KARTA MŮŽE BÝT VLASTNĚNA POUZE JEDNÍM ČLOVĚKEM. DALŠÍM MOŽNÝM TYPEM **VAZBY JE M:N**, KDE NENÍ OMEZENÍ. NAPŘÍKLAD STUDENT NA VYSOKÉ ŠKOLE SI MŮŽE ZAPSAT N PŘEDMĚTŮ A KAŽDÝ PŘEDMĚT MŮŽE MÍT ZAPSÁNO M STUDENTŮ.
DATABÁZOVÝ MODEL PŘEDSTAVUJE ZPŮSOB, JAKÝM JSOU POPSÁNY DATA A VAZBY MEZI ENTITAMI. EXISTUJE **HIERARCHICKÝ**, **SÍŤOVÝ** A DODNES SE POUŽÍVÁ **RELAČNÍ**.
RELAČNÍ DATABÁZE: ZÁKLADNÍ POJEM RELACE COŽ JE V PODSTATĚ TABULKA SKLÁDAJÍCÍ SE ZE **SLOUPCŮ** A ŘÁDKŮ. **SLOUPCE** PŘEDSTAVUJÍ VLASTNOSTI ENTITY. JEDEN **ŘÁDEK** PŘEDSTAVUJE JEDEN DATABÁZOVÝ **ZÁZNAM**.
TABULKA JE ZÁKLADNÍ ČÁST, ZE KTERÉ SE SKLÁDÁ DATABÁZE. **SOUBOR TABULEK TVOŘÍ CELOU DATABÁZI**.
UŽIVATELSKÁ DATA MAJÍ URČITOU HODNOTU. **HODNOTY** JSOU URČITÉHO TYPU. **DATOVÉHO TYPU**.
DRUHY DATOVÝCH TYPŮ (CELÉ ČÍSLO, ŘETĚZEC, DATUM, LOGICKÁ HODNOTA, DESETINÉ ČÍSLO)
CELOČÍSELNÉ TYPY: INTEGER, SMALLINT, BIGINT, **DESETINNÁ ČÍSLA S PLOVOUCÍ ŘÁDOVOU ČÁRKOU: FLOAT**, **PŘESNÁ DESETINNÁ ČÍSLA: DECIMAL(5,2)** 3 ČÍSLICE PŘED A 2 ČÍSLICE PO DESETINÉ ČÁRCE, **ČASOVÉ DATOVÉ TYPY: DATE, TIME**, TIMESTAMP, **DELŠÍ TEXT: TEXT**, **BINÁRNÍ DATA: BINARY, ŘETĚZCE: CHAR, VARCHAR** (VARCHAR – POKUD DATA NEMAJÍ DEKLAROVANOU DÉLKU, TAK JI NEZAPLNÍ A ZŮSTANE VOLNÁ PRO JINÁ DATA)
PRIMÁRNÍ KLÍČ, KAŽDÁ TABULKA BY MĚLA MÍT SLOUPEC – ATRIBUT, KTERÝ JEDNOZNAČNĚ IDENTIFIKUJE KAŽDÝ ZÁZNAM. HODNOTY V TOMTO SLOUPCI SE NESMÍ OPAKOVAT. SLOUPEC SE NAZÝVÁ PRIMÁRNÍ KLÍČ, ČASTO SE NAZÝVÁ ID. JE MOŽNÉ ZA PRIMÁRNÍ KLÍČ DEFINOVAT I KOMBINACI VÍCE SLOUPCŮ – PAK NESMÍ BÝT KOMBINACE JEJICH HODNOT NIKDY STEJNÁ – MUSÍ BÝT JEDINEČNÁ. **CIZÍ KLÍČ**: UKAZUJE SPOJENÍ ENTIT V RÁMCI DATABÁZE. PŘEDSTAVUJE VĚTŠINOU PRIMÁRNÍ KLÍČE Z JINÝCH TABULEK V DATABÁZI A UKAZUJE NA CO JE ZÁZNAM NAPOJEN V JINÉ TABULCE.
ROZDĚLENÍ PŘÍKAZŮ JAZYKA SQL: DDL (DATA DEFINITION LANGUAGE) – **VYTVÁŘEJÍ NEBO UPRAVUJÍ STRUKTURU DATABÁZE, TABULEK**. Př.: **CREATE, ALTER, DROP**…, **DML** (DATA MANIPULATION LANGUAGE) – SLOUŽÍ K **ZÍSKÁVÁNÍ, UKLÁDÁNÍ A MAZÁNÍ DAT V DATABÁZI**. Př.: **SELECT, INSERT, UPDATE, DELETE…, DCL** (DATA CONTROL LANGUAGE) – **SPRAVUJÍ UŽIVATELSKÉ ROLE A PRÁVA**., **TCL** (TRANSACTIONAL CONTROL LANGUAGE) – **SPRAVUJÍ DATABÁZOVÉ TRANSAKCE**. Př.: BEGIN, COMMIT…

TVORBA TABULKY:
```
CREATE TABLE Persons (
PersonID INT NOT NULL UNIQUE PRIMARY KEY AUTO_INCREMENT,
LastName VARCHAR(255),
FirstName VARCHAR(255),
Address VARCHAR(255),
City VARCHAR(255) DEFAULT 'Pilsen'
) ENGINE = INNODB CHARACTER SET=utf8 COLLATE utf8_czech_ci;
```

SMAZÁNÍ TABULKY:
```
DROP TABLE zamestnanci;
```
PŘIDÁNÍ SLOUPCE:
```
ALTER TABLE zamestnanci
ADD narozen DATETIME NOT NULL;
```
ODEBRÁNÍ SLOUPCE:
```
ALTER TABLE zamestnanci DROP narozen;
```

VLOŽENÍ DAT DO TABULKY: `INSERT INTO rezervace (film,misto,jmeno) VALUES ('Terminátor','A44','Sarah Connor'), ('Matrix','B31','Trinity');`
VLOŽÍ DATA DO TABULKY rezervace. VLOŽÍ TAM DVA ZÁZNAMY, DO SLOUPCŮ film, místo, jmeno

ZMĚNA HODNOT V TABULCE: `UPDATE stranky SET pocitadlo=pocitadlo+1,posledniNavsteva=NOW() WHERE url='produkty'`
ZMĚNÍ HODNOTY V TABULCE stranky A NASTAVÍ SLOUPCE pocitadlo A posledniNavsteva. NASTAVÍ JEN PRO ZÁZNAMY, KDE JE url='produkty'

VÝBĚR DAT Z TABULKY: `SELECT jmeno,prijmeni FROM zamestnanci WHERE (mesto='Praha') ORDER BY prijmeni,jmeno ASC LIMIT 10;`
VYBERE Z TABULKY zamestnanci DATA ZE SLOUPCŮ jmeno A prijmeni, JEN PRO MĚSTO PRAHA, SEŘADÍ VZESTUPNĚ PODLE ABECEDY PODLE PŘÍJMENÍ A JMÉNA. VYPÍŠE JICH JEN DESET. ŘAZENÍ JE MOŽNÉ POUŽÍT ASC VZRŮSTAJÍCÍ NEBO DESC KLESAJÍCÍ

SMAZANI DAT Z TABULKY: `DELETE FROM zamestnanci;` **(VYMAŽE VŠECHNY ZÁZNAMY, ALE AUTOINCREMENT NERESETUJE) | TRUNCATE** `zamestnanci;` **TAKÉ RESETUJE AUTOINCREMENT |** `DELETE FROM zamestnanci WHERE (mesto = 'Plzeň')` **VYMAŽE JEN URČITÉ ZÁZNAMY**

MODERNÍ DATABÁZE SPLŇUJÍ **PODMÍNKY ACID**: **ATOMICITY** – V RÁMCI TRANSAKCE PROBĚHNOU BUĎ VŠECHNY NEBO ŽÁDNÉ, **CONSISTENCY** – PO OPERACI SE NACHÁZÍ DATA V KONZISTENTNÍM STAVU, **INDEPENDENCE** – DVĚ TRANSAKCE SE NAVZÁJEM NEOVLIVNÍ, JAKO BY SE SPUSTILY PO SOBĚ., **DURABILITY** – PO TRANSAKCI ZMĚNY TRVALE ULOŽENY.
ENGINE: ZPŮSOB ULOŽENÍ DAT (PRO MYSQL NAPŘ. **MyISAM** – RYCHLÉ ČTENÍ, FULLTEXTOVÉ VYHLEDÁVÁNÍ, **InnoDB** – PODPORUJE TRANSAKCE), **KÓDOVÁNÍ**: NAPŘ. UTF8, WIN1250, LATIN2, …, **POROVNÁVÁNÍ**: URČUJE POŘADÍ ZNAKŮ V KONKRÉTNÍM JAZYCE A TAK UMOŽNÍ V DANÉM JAZYCE ŘADIT ABECEDNĚ. KAŽDÁ TABULKA V DATABÁZI MŮŽE MÍT VLASTNÍ KÓDOVÁNÍ I POROVNÁVÁNÍ. **TRANSAKCE** JE OPERACE S DATABÁZÍ, KTERÁ SPLŇUJE ACID. **NORMALIZACE TABULEK** REDUKUJE TABULKY, TAK ABY SE NEOPAKOVALA DATA – EXISTUJE NĚKOLIK NORMÁLNÍCH FOREM, NEJČASTĚJI SE POUŽÍVAJÍ PRVNÍ TŘI…
TERMÍNY: DEKLARATIVNÍ, SQL, ACID, TABULKA, DATABÁZE, ENTITA, ZÁZNAM, ATRIBUT, PRIMÁRNÍ KLÍČ, RELACE, TRANSAKCE, NORMALIZACE
PRAKTICKÁ DOVEDNOST: V SQL VYTVOŘIT TABULKU, VYBRAT Z TABULKY, VLOŽIT DO TABULKY, ZMĚNIT HODNOTY, SMAZAT, PRVNÍ TŘI NORMÁLNÍ FORMY A JEJICH POUŽITÍ, **DOPORUČENÉ ZDROJE:** http://www.w3schools.com/sql/default.asp, http://progopedia.com/language/sql/, https://www.interval.cz/clanky/databaze-a-jazyk-sql/, http://voho.cz/wiki/sql/, http://dev.mysql.com/doc/refman/5.7/en/

13. Skriptovací jazyky

SKRIPTOVACÍ JAZYKY JSOU **PRIMÁRNĚ URČENY PRO RYCHLÝ A SNADNÝ VÝVOJ PROGRAMŮ**. JSOU TO **INTERPRETOVANÉ JAZYKY**, KTERÉ NEJSOU EXTRÉMĚ RYCHLÉ A PROTO SE PRIMÁRNĚ NEPOUŽÍVAJÍ NA KOMPLIKOVANÉ VÝPOČTY A PRÁCI SE SLOŽITÝMI DATOVÝMI STRUKTURAMI. OBVYKLE **SE JEDNÁ O SLABĚ TYPOVANÉ NEBO NETYPOVANÉ JAZYKY (DYNAMICKY TYPOVANÉ)** = AUTOMATICKÉ KONVERZE TYPŮ A PROMĚNNÉ MOHOU OBSAHOVAT COKOLIV – PROMĚNNÉ MAJÍ TYP, ALE ČASTO TO PROGRAMÁTOROVI NEDÁVAJÍ NAJEVO A PODLE JEJICH POUŽITÍ HO MĚNÍ. JEDEN VÝRAZ VE SKRIPTOVACÍM JAZYCE MŮŽE OBSAHOVAT 100 AŽ 1000 STROJOVÝCH INSTRUKCÍ. SKRIPTOVACÍ JAZYKY POVĚTŠINOU PRACUJÍ S JIŽ VESTAVĚNÝMI KOMPLIKOVANĚJŠÍMI KOMPONENTAMI.

VÝHODY SKRIPTOVACÍCH JAZYKŮ: RYCHLÝ VÝVOJ APLIKACÍ, **SNADNÁ INSTALACE APLIKACÍ**: JEN ZKOPÍROVANÝ ZDROJOVÝ KÓD, **SNADNO A RYCHLE SE NAUČÍ**, **DYNAMICKÉ VLASTNOSTI: TYPOVÁNÍ, ROZSAHY POLÍ**,…

NEVÝHODY SKRIPTOVACÍCH JAZYKŮ: ČASTO **NENÍ ÚPLNĚ SPRÁVNÝ NÁVRH PROGRAMU PODLE PRAVIDEL STRUKTUROVANÉHO NEBO OBJEKTOVĚ ORIENTOVANÉHO PROGRAMOVÁNÍ**, **MENŠÍ RYCHLOST V POROVNÁNÍ S OSTATNÍMI JAZYKY**, POMOCÍ SKRIPTOVACÍCH JAZYKŮ ČASTO **NELZE NAPROGRAMOVAT VŠE**, JSOU URČENY PRO SPOLUPRÁCI S OSTATNÍMI JAZYKY, KTERÉ TO DOKÁŽÍ.

POUŽITÍ SKRIPTOVACÍCH JAZYKŮ: SPRÁVA SYSTÉMU (SKRIPTY, SHELL, DÁVKOVÉ OPERACE, PO STARTU OS, ARCHIVACE), **AUTOMATIZACE PROGRAMŮ** (PŘEKLAD, INSTALACE), **PŘÍZPŮSOBENÍ FUNKČNOSTI APLIKACÍ** (MAKRA V TEXTOVÝCH EDITORECH – MS OFFICE, WINDOWS SCRIPTING)

OBLASTI VYUŽITÍ: GRAFICKÁ UŽIVATELSKÁ PROSTŘEDÍ (VISUAL BASIC), **INTERNET** (PERL, JAVASCRIPT, PHP)

MEZI SKRIPTOVACÍ JAZYKY PATŘÍ: JAVASCRIPT, PYTHON, PHP, PERL

JAVASCRIPT: SKRIPTOVACÍ PROGRAMOVACÍ JAZYK. POČÁTKY JAVASCRIPTU FIRMA NETSCAPE ROKU 1995. OD ROKU 2012 JE JAVASCRIPT STANDARDIZOVÁN PODLE ECMASCRIPTU. POUŽÍVÁ SE **PŘI TVORBĚ WEBOVÝCH STRÁNEK**. **NEEXISTUJE ŽÁDNÝ VZTAH MEZI JAZYKY JAVA A JAVASCRIPT!!!**

POUŽITÍ JAVASCRIPTU:

- **OVĚŘOVÁNÍ VALIDITY DAT** VLOŽENÝCH DO WEBOVÝCH FORMULÁŘŮ – KONTAKTNÍ A PŘIHLAŠOVACÍ FORMULÁŘE.
- **UPDATOVÁNÍ INFORMACÍ NA WEBSTRÁNCE** BEZ NUTNOSTI JI OBNOVOVAT
- **MANIPULACE S DOM** (DOCUMENT OBJECT MODEL) ÚPRAVA VZHLEDU A FUNKČNOSTI HTML DOKUMENTU A JEHO VLASTNOSTÍ.
- **VYTVÁŘENÍ UPOZORNĚNÍ NA URČITÉ UDÁLOSTI** V PROHLÍŽEČI NEBO NOVÁ OKNA
- **VYTVÁŘENÍ ANIMACÍ**

JAVASCRIPT **SLOUŽÍ K VYTVÁŘENÍ DYNAMICKÝCH PRVKŮ WEB STRÁNEK**. **BĚŽÍ NA STRANĚ KLIENTA, ZPRACOVÁVÁ HO INTERNETOVÝ BROWSER, KTERÝ SE STÁVÁ JEHO INTERPRETEM**. JAVASCRIPT **JE CASE SENSITIVE** = **ZÁLEŽÍ U NĚJ NA VELIKOSTI POUŽITÝCH PÍSMEN**! AHOJ JE JINÁ PROMĚNNÁ NEŽ Ahoj. JE TO **OBJEKTOVĚ ORIENTOVANÝ JAZYK**.

PŘÍKLAD:

```
<head>
<script type="text/javascript">
      var text = "IT4";
      function MojeTrida()
      {
         alert ("Toto je má třída" + text);
      }
</script>
</head>
<body><p><button onclick="MojeTrida()">Vytvořit dialogové okno!</button></p></body>
```

PHP: JE POPULÁRNÍ **OPEN SOURCE SKRIPTOVACÍ JAZYK**, KTERÝ SE POUŽÍVÁ **PROGRAMOVÁNÍ APLIKACÍ KLIENT – SERVER**, **VYTVÁŘENÍ WEBSTRÁNEK**. K DISPOZICI PRO RŮZNÉ OS ZDARMA. **SKRIPTY JSOU SPOUŠTĚNÉ BUĎ NA STRANĚ SERVERU A TO TÉMĚŘ VÝHRADNĚ**, NEBO Z PŘÍKAZOVÉHO ŘÁDKU. **SYNTAXE JE PODOBNÁ S C/C++.** VE VERZI **PHP 5 JIŽ PLNĚ PODPOROVANÉ OBJEKTOVĚ ORIENTOVANÉ PROGRAMOVÁNÍ**. JEDEN Z NEJOBLÍBENĚJŠÍCH SKRIPTOVACÍCH JAZYKŮ. JE DOSTUPNÝ NA VĚTŠINĚ HOSTINGOVÝCH SLUŽEB. VZNIKL V ROCE 1994, PRO OBOHACENÍ WEBSTRÁNEK O DYNAMICKÉ PRVKY. PHP JE **INTERPRETOVANÝ JAZYK SE SLABÝM TYPOVÁNÍM**. PHP **SPOLUPRACUJE S MNOHA DATABÁZEMI: MySQL, PostgreSQL, ODBC, MS SQL**, Oracle,… PHP MÁ PŘIPRAVEN I PŘÍSTUP K DALŠÍM SLUŽBÁM: IMAP, POP3, HTTP,… EXISTUJE ZDE **VELMI DOBRÁ PODPORA ZPRACOVÁNÍ TEXTU, PRÁCE S REGULÁRNÍMI VÝRAZY. PHP SKRIPTY SE MOHOU PŘÍMO VKLÁDAT DO HTML STRÁNEK:**

```
<body><?php
echo "My first PHP script!"; // VYPÍŠE TEXT
?></body>
```

```
z $x = 10;
if($x > 0) echo "$x je kladné";
```

UŽIVATELSKÉ PROMĚNNÉ SE NEDEKLARUJÍ, JMÉNO PROMĚNNÉ ZAČÍNÁ ZNAKEM $.

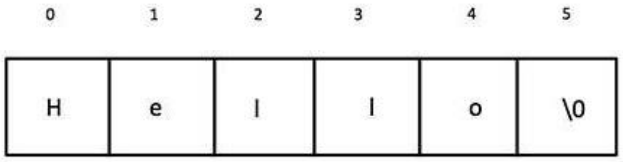
TERMÍNY: SKRIPTOVACÍ, INTERPRETOVANÉ, SILNĚ, SLABĚ TYPOVANÉ JAZYKY, DYNAMICKY TYPOVANÉ JAZYKY, JAVASCRIPT, PHP, PYTHON, PERL, REGULÁRNÍ VÝRAZY, CASE SENSITIVE,…

PRAKTICKÁ DOVEDNOST: NAPSAT JEDNODUCHÝ PROGRAM V JAVASCRIPTU. POPSAT JAK SE VKLÁDÁ PHP DO WEBSTRÁNEK.

DOPORUČENÉ ZDROJE: http://www.cs.vsb.cz/benes/vyuka/pte/prednasky/04-skriptj.pdf, http://referaty.atlas.sk/ostatne/informatika/37917/?print=1, http://blog.voracek.net/wp-content/uploads/2013/09/JanVoracek_SkriptovaciJazykyProTvorbuWebovychAplikaci.pdf, http://reboot.cz/howto/programovani/skriptovaci-programovaci-jazyky/articles.html?id=153, https://developer.mozilla.org/cs/docs/Web/JavaScript, http://polopate.jakpsatweb.cz/index.php?page=trpaslik-uvod, http://www.jakpsatweb.cz/php/, http://www.w3schools.com/php/, http://www.itnetwork.cz/php/zaklady/php-tutorial-uvod-do-webovych-aplikaci,

14. Práce s textem, regulární výrazy

TEXT V PROGRAMOVÁNÍ SE ČASTO OZNAČUJE JAKO **ŘETĚZEC**. V KAŽDÉM JAZYCE SE S ŘETĚZCI PRACUJE TROCHU JINAK. **V JAZYCE C** SE S ŘETĚZCI PRACUJE **JAKO S POLI ZNAKŮ**. **PŘÍKLAD DEKLARACE PROMĚNNÉ** UCHOVÁVAJÍCÍ ŘETĚZEC V JAZYCE C: **char citat[215];** DÉLKA ŘETĚZCE URČÍ, JAK VELKÉ MÍSTO SE VYHRADÍ PRO ULOŽENÍ ŘETĚZCE, POKUD SE NEUVEDE, PAK PŘEKLADAČ BĚHEM PŘEKLADU SPOČÍTÁ ZNAKY V UVOZOVKÁCH A VYHRADÍ MÍSTO PRO ŘETĚZEC O JEDNO MÍSTO VĚTŠÍ. **INICIALIZACE PROMĚNNÉ: char citat[215] = "Byl tak";** (TENTO ZPŮSOB SE POUŽÍVÁ MNOHEM ČASTĚJI, JE POHODLNĚJŠÍ, NÁSLEDUJÍCÍ, JE KOMPLIKOVANÝ) **char citat[215] = {'B', 'y', 'l', ' ', 't', 'a', 'k', '\0'};** V JAZYCE C SE **VŠECHNY ŘETĚZCE UKONČUJÍ HODNOTOU NULA**. BINÁRNÍ NULA PŘEDSTAVUJE **ZNAK '\0'**. **PROTO SE REZERVUJE AUTOMATICKY PROSTOR O JEDNO VĚTŠÍ**, ABY ZBYL NA NULOVÝ ZNAK. I MY, KDYŽ DEKLARUJEME DÉLKU ŘETĚZCE, MUSÍME DEKLAROVAT ŘETĚZEC, KTERÝ BUDE O JEDEN VĚTŠÍ. PŘI **URČENÍ DÉLKY ŘETĚZCE** ZJISTÍME **POČET ZNAKŮ AŽ DO NULOVÉHO ZNAKU** A MÁME VYPOČÍTANOU JEHO DÉLKU. V JAZYCE C SE **NEJČASTĚJI POUŽÍVANÉ FUNKCE PRO PRÁCI S ŘETĚZCI** NACHÁZEJÍ **V HLAVIČKOVÉM SOUBORU string.h.**



size_t strlen(const char *ret)	VRACÍ DÉLKU ŘETĚZCE VE ZNACÍCH, NIKOLI CELÉ VYHRAZENÉ MÍSTO	string.h
char* strcat(char *cil, const char *zdroj)	SPOJÍ DVA ŘETĚZCE A VÝSLEDEK VLOŽÍ DO cil. VRACÍ UKAZATEL NA cil.	string.h
int strcmp(const char *ret1, const char *ret2)	LEXIKOGRAFICKY POROVNÁ DVA ŘETĚZCE. VRÁTÍ ZÁPORNOU HODNOTU, POKUD JE PRVNÍ ŘETĚZEC LEXIKOGRAFICKY MENŠÍ, NULU, POKUD JSOU STEJNÍ A KLADNOU, POKUD JE PRVNÍ VĚTŠÍ.	string.h
char *strcpy(char *dest, const char *src)	ZKOPÍRUJE ŘETĚZEC src DO ŘETĚZCE dest. VRACÍ UKAZATEL NA dest. V JAZYCE C NEMŮŽEME JEDNODUŠE KOPÍROVAT ŘETĚZCE, PROTOŽE PŘEDSTAVUJÍ POLE. POUŽÍVÁME PROTO TUTO FUNKCI.	string.h
char *strchr(const char *s, int c)	VRACÍ UKAZATEL NA PRVNÍ POZICI, KDE SE VYSKYTUJE ZNAK c. NEBO VRACÍ NULL, KDYŽ HO NENAJDE.	string.h
char *strstr(const char *haystack, const char *needle);	VRACÍ UKAZATEL NA PRVNÍ VÝSKYT ŘETĚZCE needle V ŘETĚZCI haystack. POKUD HO NENAJDE, VRACÍ NULL.	string.h
void putchar(int znak)	ZAPÍŠE NA VÝSTUP JEDEN ZNAK	stdio.h
int getchar()	NAČTE MAX JEDEN ZNAK ZE VSTUPU, ULOŽÍ HO PŘETYPOVANÝ NA INT	stdio.h

ŘETĚZCOVÉ PROMĚNNÉ V JAZYCE C NEMŮŽEME **POROVNÁVAT** JAKO ČÍSLA POROVNÁVACÍM OPERÁTORM ==, PROTOŽE TO JSOU V PODSTATĚ POLE. JEDINÝ ZPŮSOB JAK JE POROVNÁVAT JE **POMOCÍ FUNKCE strcmp. PŘÍKLAD POUŽITÍ:**
if (strcmp(textovy, "jablko") == 0), HLEDÁNÍ ŘETĚZCE V ŘETĚZCI: char s1 [] = "My House is small"; strstr (s1, "House");
HLEDÁNÍ PRVNÍHO VÝSKYTU ZNAKU V ŘETĚZCI: char buf [] = "This is a test"; s = strchr (buf, 't');
VYPSÁNÍ ŘETĚZCE NA OBRAZOVKU A ZADÁVÁNÍ ŘETĚZCE Z KLÁVESNICE:
printf("Vypsany retezec: %s", nekolikSlov); // V PROMĚNNÉ nekolikSlov MŮŽE BÝT ULOŽEN ŘETĚZEC, KTERÝ SE VYPÍŠE MÍSTO %s **scanf("%80s", poleZnaku);.** // DO PROMĚNNÉ poleZnaku BUDE ZADÁNA HODNOTA ŘETĚZCE ZADANÉHO Z KLÁVESNICE O DÉLCE NEJMÉNĚ 80 ZNAKŮ DOPLNĚNÝCH MEZERAMI. KDYŽ DÁME %.8s BUDE TO ZNAMENAT, ŽE MAXIMÁLNĚ VYTISKNEME OSM ZNAKŮ.
FORMÁTOVANÝ VSTUP A VÝSTUP: % [flags] **[width] [.prec] type_char**

položka	*význam*
flags	zarovnání výstupu, zobrazeni znaménka a desetinných míst u čísel, úvodní nuly, prefix pro osmičkový a šestnáctkový výstup
width	minimální počet znaků na výstupu, mohou být uvedeny mezerami nebo nulami
.prec	maximální počet znaků na výstupu, pro celá čísla minimum zobrazených znaků, pro racionální počet míst za desetinnou tečkou
type_char	povinný znak, určuje typ konverze

symbol	*Typ konverze*
d, i	Celé číslo se znaménkem
u	Celé číslo bez znaménka
f	Racionální číslo bez exponentu, implicitně 6 desetinných míst
c	znak
s	řetězec

REGULÁRNÍ VÝRAZY: JSOU SPECIÁLNÍ TEXTOVÉ ŘETĚZCE, KTERÉ **POPISUJÍ URČITOU MASKU (VZOR),** KTERÉ MÁ ODPOVÍDAT URČITÝ TEXTOVÝ ŘETĚZEC. REGULÁRNÍ VÝRAZY SE POUŽÍVAJÍ **V MNOHA PROGRAMOVACÍCH JAZYCÍCH: PERL, JAVA, C#, JAVASCRIPT, PHP**, …. **VYUŽÍVAJÍ SE PŘI HLEDÁNÍ URČITÉHO VZORU TEXTU A NEBO OVĚŘENÍ SPRÁVNÉHO ZADÁNÍ TEXTU DO FORMULÁŘŮ**.
METAZNAKY, KTERÉ V REGULÁRNÍCH VÝRAZECH PLNÍ ZVLÁŠTNÍ ÚKOL: **\, ^, $, ., [,], |, (,), ?, *, +, {, },** **** VRACÍ VŠEM METAZNAKŮM V REGULÁRNÍCH VÝRAZECH JEJICH PŮVODNÍ VÝZNAM, **.** ZASTUPUJE V REGULÁRNÍCH VÝRAZECH JEDEN LIBOVOLNÝ ZNAK (KROMĚ ZNAKU NOVÉHO ŘÁDKU), **SKUPINY ZNAKŮ** JSOU TAKÉ DEFINOVÁNY: **\d** – ČÍSLICE 0-9, **\D** – COKOLIV JEN NE 0-9, **\s** – BÍLÉ ZNAKY = MEZERA, TABULÁTOR, …, **\S** – JAKÝKOLIV ZNAK KROMĚ BÍLÝCH ZNAKŮ, **[abcd]** – SKUPINA ZNAKŮ DANÁ VÝČTEM, **[a-z]** – SKUPINA ZNAKŮ DANÁ INTERVALEM, A MNOHO DALŠÍCH.
PŘÍKLADY POUŽITÍ REGULÁRNÍCH VÝRAZŮ: \d{4} (SEKVENCE ČTYŘ ČÍSLIC ZA SEBOU), ^.luh$ VYHOVUJÍ ŘETĚZCE "pluh", "dluh", NEVYHOVUJE ŘETĚZEC "dluhy"

KVANTIFIKÁTORY	
?	Znak se opakuje minimálně 0 krát, maximálně 1 krát
*****	Minimálně 0 krát, maximálně neomezeno
+	Minimálně 1 krát, maximálně neomezeno
{n}	Právě n krát
{m,n}	Minimálně mkrát, maximálně nkrát
{m,}	Minimálně mkrát, maximálně neomezeno

TERMÍNY:ŘETĚZEC, POLE, DÉLKA ŘETĚZCE, SPOJENÍ ŘETĚZCŮ, POROVNÁNÍ ŘETĚZCŮ, VSTUP A VÝSTUP ŘETĚZCE, FORMÁTOVANÝ VÝSTUP, REGULÁRNÍ VÝRAZY, METAZNAKY, BÍLÉ ZNAKY, KVANTIFIKÁTORY
PRAKTICKÁ DOVEDNOST: – PRÁCE S ŘETĚZCI – DEKLARACE, VÝSTUP, VSTUP, ULOŽENÍ ŘETĚZCE DO PAMĚTI, VYHLEDÁVÁNÍ SLOV V ŘETĚZCI, JEDNODUCHÝ REGULÁRNÍ VÝRAZ – VYSVĚTLIT FUNKČNOST A NEBO SESTAVIT
DOPORUČENÉ ZDROJE: http://www.itnetwork.cz/cplusplus/cecko-zaklady/tutorial-jazyk-c-textove-retezce/, http://www.sallyx.org/sally/c/c07.php, https://www.smnd.sk/anino/programming/c/saloun/kap08.htm, http://www.tutorialspoint.com/ansi_c/c_strchr.htm, http://www.tutorialspoint.com/ansi_c/c_strstr.htm, https://www.interval.cz/clanky/regularni-vyrazy-a-javascript-uvod/, http://www.regularnivyrazy.info/serial-javascript-regexp.html#.VtYX9vnhDak, http://www.itnetwork.cz/javascript/javascript-tutorial-regularni-vyrazy-regex/,https://regex101.com/, http://regexr.com/, **https://www.interval.cz/clanky/regularni-vyrazy-v-prikladech/**

15. HTML

HTML JE ZNAČKOVACÍ JAZYK ZKRATKA **HYPERTEXT MARKUP LANGUAGE**. **SLOUŽÍ PRO TVORBU WEB STRÁNEK**. JEHO HLAVNÍM ÚČELEM JE **VYTVOŘIT MULTIMEDIÁLNÍ DOKUMENT**, KTERÝ LZE PREZENTOVAT NA INTERNETU. **UMOŽŇUJE VLOŽIT TEXT, OBRÁZKY A OSTATNÍ MULTIMEDIÁLNÍ ZDROJE**. DOKUMENT JE PAK **HIERARCHICKY STRUKTUROVÁN** A **JEDNOTLIVÉ ČÁSTI MAJÍ URČEN JAKÝ VÝZNAM MÁ JEJICH OBSAH**. HTML **NESLOUŽÍ PRO ÚPRAVU VZHLEDU STRÁNKY**. HTML VZNIKL V ROCE 1990.
VERZE HTML: V SOUČASNOSTI (**2015**) **NEJPOUŽÍVANĚJŠÍ: HTML 5 – 70%, XHTML 1.0** 15% A **HTML 4.01** 3%, HTML VZNIKLO SE ŽÁDNÝM ZNAČENÍM VERZE, PAK PŘES HTML 2 A DÁL AŽ K **HTML 4.01** JEŠTĚ SE POUŽÍVÁ (**JASNĚ DEKLAROVÁN ODDĚLENÍ SMYSLU (OBSAHU, STRUKTURY) A VZHLEDU (STYLU, FORMY, PREZENTACE), BRALO V ÚVAHU, ŽE EXISTUJE CSS**), **XHTML 1.0** (**PŘENESENÍ HTML 4.01 DO STANDARDU XML** – UZAVÍRÁNÍ TAGŮ, PSANÍ TAGŮ A ATRIBUTŮ MALÝMI PÍSMENY ATD.– VYŽADUJE PEČLIVĚJŠÍ KÓDOVÁNÍ), **HTML5** – **NAVAZUJE NA VERZI 4.01** A NA EXISTUJÍCÍ OSVĚDČENÉ POSTUPY. POPRVÉ JE VE SPICIFIKACI I PŘESNĚ DEFINOVÁNO JAK SE TO MÁ VYKRESLIT, KDYŽ JE TO CHYBNĚ KÓDOVÁNO. UMOŽŇUJE PSÁT JAK HTML, TAK I XHTML STYLEM. **OBSAHUJE I JAVASCRIPTOVÉ API, OFFLINE APLIKACE, KRESLENÍ V APLIKACI, VIDEO, ZVUK, GEOLOKACI, …**
XHTML: JE HTML ZALOŽENÉ NA XML. ZKRATKA: E**X**TENSIBLE **H**YPERT**E**XT **M**ARKUP **L**ANGUAGE. **PŘÍSNĚJŠÍ KÓDOVÁNÍ**. (XHTML DOCTYPE, XMLNS ATRIBUT V HTML TAGU, KAŽDÝ ATRIBUT MUSÍ MÍT HODNOTU V UVOZOVKÁCH. PŘI NEPÁROVÉM TAGU MUSÍ BÝT UKONČEN />. U VERZE STRICT NEJSOU FORMÁTOVACÍ TAGY: B, I, FONT, SKRIPTY VKLÁDAT DO SEKCE CDATA, ZNAK & PŘEVÁDĚT NA HTML ENTITU)
KÓDOVÁNÍ WEBSTRÁNEK: JAKÁ SADA ZNAKŮ BUDE POUŽITÁ - windows-1250, utf-8, iso-8859-2. **UPŘEDNOSTŇOVAT UTF-8!!!**

STRUKTURA HTML SOUBORU PODLE STANDARDU HTML5

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>titulek dokumentu, na titulní liště</title>
    </head>
    <body>
        <p>Obsah HTML dokumentu...</p>
    </body>
</html>
```

STRUKTURA HTML SOUBORU PODLE STANDARDU XHTML5 XML

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta charset="UTF-8" />
        <title>titulek dokumentu, na titulní liště prohlížeče</title>
    </head>
    <body>
        <p>Obsah HTML dokumentu...</p>
    </body>
</html>
```

DOCTYPE: JE **ÚPLNĚ NA ZAČÁTKU DOKUMENTU**. JE **PRVNÍ ELEMENT WEB STRÁNKY**, NEPÁROVÝ A NENÍ UZAVŘENÝ. **DÁVÁ** PROHLÍŽEČI **NAJEVO, O JAKÝ DOKUMENT JDE**. DTD – DOCUMENT TYPE DEFINITION TYP DOKUMENTU.
XHTML 1.0 I HTML 4.01 MAJÍ TŘI MOŽNÉ TYPY DOKUMENTŮ (**STRICT** – BEZ PREZENTAČNÍCH ELEMENTŮ A ELEMENTŮ KTERÉ NEJSOU DOPORUČENY, NELZE POUŽÍVAT FRAMESETS, **TRANSITIONAL** – OBSAHUJE VŠECHNY HTML TAGY, NEJSOU DOVOLENY FRAMESETS, **FRAMESET** – JE STEJNÝ JAKO TRANSITIONAL, ALE JSOU DOVOLENY FRAMESETS)
METATAGY: V HLAVIČCE WEBU, OBSAHUJÍ: KÓDOVÁNÍ, AUTORA, COPYRIGHT, KLÍČOVÁ SLOVA, POPIS OBSAHU WEBSTRÁNKY …
TAG JE ZÁKLADNÍ PRVEK HTML JAZYKA UZAVŘENÝ V OSTRÝCH ZÁVORKÁCH. , <a>
ZNAČKA INTERPRETUJE TAG V HTML KÓDU, U **PÁROVÝCH TAGŮ** ROZEZNÁVÁME **ÚVODNÍ A KONCOVOU ZNAČKU**. MÁME **PÁROVÉ** , <a> A **NEPÁROVÉ TAGY**
, <hr>, <img>, <input>. **ELEMENT** JE TO CO TAG VYKRESLUJE A VŠE CO REPREZENTUJE. POČÁTEČNÍ I KONEČNÁ ZNAČKA I JEHO OBSAH. EXISTUJÍ DVA ZÁKLADNÍ DRUHY ELEMENTŮ PODLE TOHO, JAK MAJÍ DEFAULTNĚ NASTAVEN CSS STYL DISPLAY: TYTO DVA **DRUHY ELEMENTŮ** JSOU **BLOKOVÉ A ŘÁDKOVÉ ELEMENTY**, DISPLAY MÁ HODNOTU **BLOCK NEBO INLINE**. **BLOKOVÉ (BLOCK) ELEMENTY** ZA SEBOU VYTVOŘÍ KONEC ŘÁDKY A LZE JIM NASTAVIT VÝŠKA A ŠÍŘKA. ZABÍRAJÍ NORMÁLNĚ CELOU ŠÍŘKU ŘÁDKY. VŽDY ZAČÍNAJÍ NA NOVÉ ŘÁDCE. (div, h1-h6, p, form) **ŘÁDKOVÉ (INLINE) ELEMENTY** ZA SEBOU KONEC ŘÁDKY NEDĚLAJÍ A ANI ŠÍŘKA ANI DÉLKA BY JIM NEMĚLA JÍT NASTAVIT. NEZAČÍNAJÍ TAKÉ NA NOVÉ ŘÁDCE A ZABÍRAJÍ NA NÍ JEN NEZBYTNĚ NUTNÉ MÍSTO, ABY SE ELEMENT VYKRESLIL. (span, a , img). ELEMENTY BLOCK BY SE NEMĚLI VKLÁDAT DO ELEMENTŮ INLINE.

<tag atribut="hodnota atributu" atribut2=hodnota>Obsah tagu</tag>

ZÁKLADNÍ HTML TAGY URČUJÍ STRUKTURU DOKUMENTU:
<html>… </html> - ZAČÁTEK A KONEC DOKUMENTU – V PODSTATĚ KONTEJNER, KTERÝ OBSAHUJE RŮZNÉ ELEMENTY
<head>… </head> - **HLAVIČKA DOKUMENTU** - NEZOBRAZUJE SE, OBSAHUJE INFORMACE O STRÁNCE, PROPOJUJE STRÁNKU S JEJÍM STYLEM V CSS, JEJÍM DYNAMICKÝM CHOVÁNÍM V JAVASCRIPTU, METATAGY, KLÍČOVÁ SLOVA, POPIS A TITULEK WEB STRÁNKY A JEJÍ JAZYK – PRO DOBRÉ ZOBRAZENÍ RŮZNÝCH JAZYKŮ, JAK SE MÁ TISKNOUT NEBO ZOBRAZOVAT NA RŮZNÝCH ZAŘÍZENÍCH. **<body>… </body>** - **TĚLO DOKUMENTU** – OBSAHUJE VEŠKERÝ ZOBRAZOVANÝ OBSAH STRÁNKY

TABULKA PŘÍKLAD:

NADPIS1	NADPIS2
5	12
27	10

```
<table>
  <tr><th>NADPIS1</th><th>NADPIS2</th></tr>
  <tr><td>5</td><td>12</td></tr>
  <tr><td>27</td><td>10</td></tr>
</table>
```

TAGY UVNITŘ TĚLA DOKUMENTU: ** OBSAHUJE ODSTAVEC **, **<h1>**OBSAHUJE **NADPIS NEJVYŠŠÍ ÚROVNĚ**, JEN JEDEN NA STRÁNCE NEBO V CEKCI**</h1>**, **<h2>podnadpis</h2>**, **<h3>podnadpis nižší úrovně</h3>** …, ****ZOBRAZENÝ ODKAZ NA WEBSTRÁNKU****, **<img src="soubor_obrazku.jpg" alt="textový popis obrázku">**, **
** - ZALOMENÍ ŘÁDKU, ** **ELEMENT PRO SNAZŠÍ STYLOVÁNÍ STRÁNKY** **, **** ELEMENT PRO SNAZŠÍ STYLOVÁNÍ STRÁNKY ****, **<table>**OBSAH TABULKY**</table>**, **<tr>**OBSAH ŘÁDKYTABULKY**</tr>**, **<td>**OBSAH BUŇKY TABULKY**</td>**, **<th>**OBSAH BUŇKY ZÁHLAVÍ TABULKY - ZVÝRAZNĚNO**</th>**, **
OBSAH ČÍSLOVANÉHO SEZNAMU
**, **
OBSAH NEČÍSLOVANÉHO SEZNAMU
**, **
POLOŽKA ČÍSLOVANÉHO I NEČÍSLOVANÉHO SEZNAMU
**

ČÍSLOVANÝ SEZNAM

1. MÁSLO
2. CHLEBA
3. 10 VAJEC

```
<ol>
  <li>MÁSLO</li>
  <li>CHLEBA</li>
  <li>10 VAJEC</li>
</ol>
```

VALIDACE WEBOVÝCH STRÁNEK OVĚŘUJE, ZDALI JSOU KÓDOVÁNY PODLE STANDARDŮ JAZYKA HTML, SLOUŽÍ K TOMU ONLINE VALIDÁTORY NAPŘ: https://validator.w3.org/
TERMÍNY: TAG, ZNAČKA, ELEMENT, ATRIBUT, HODNOTA ATRIBUTU, OBSAH TAGU, HTML, XHTML, METATAGY, HLAVIČKA, TĚLO, PÁROVÉ TAGY, NEPÁROVÉ TAGY, ŘÁDKOVÉ ELEMENTY, BLOKOVÉ ELEMENTY, DOCTYPE
PRAKTICKÁ DOVEDNOST: SESTAVIT Z TAGŮ STRÁNKY JEJÍ ZÁKLADNÍ STRUKTURU, VLOŽIT TAM ZÁKLADNÍ TAGY, ODKAZ, ODSTAVEC, SEZNMY, TABULKU
DOPORUČENÉ ZDROJE: http://programujte.com/clanek/2010082200-html5-nove-vlastnosti/,http://www.jakpsatweb.cz, http://www.w3schools.com/html, http://www.webtvorba.cz/xhtml/, https://html.spec.whatwg.org/multipage/, http://owebu.org/cze/html/index.php, http://jecas.cz/html-kostra,

16. CSS

CSS JE ZKRATKA PRO CASCADING STYLE SHEETS – ČESKY KASKÁDOVÉ STYLY. SLOUŽÍ **K FORMÁTOVÁNÍ VZHLEDU HTML STRÁNEK**. NA ROZDÍL OD HTML **NEDÁVÁ OBSAHU VÝZNAM, ALE VZHLED**. URČUJE FONTY, BARVY, STYLY, POZICE ATD. TEXTU, ODSTAVCŮ, ODKAZŮ, OBRÁZKŮ APODOBNĚ. **JSOU VÝSLEDKEM SNAHY ODDĚLIT VÝZNAM OD VZHLEDU.**
CSS VZNIKL V ROCE 1996, ZE ZAČÁTKU HO MNOHO PROHLÍŽEČŮ NEPODPOROVALO. VERZE CSS 1: 1996, CSS 2: 1998, CSS 2.1: REVIDOVÁNO CSS 2 V ROCE 2011, **CSS 3 V SOUČASNOSTI IMPLEMENTOVANÉ DO URČITÉ MÍRY VE VŠECH HLAVNÍCH PROHLÍŽEČÍCH**. NAKOLIK JE IMPLEMENTOVANÉ: http://css3test.com/
CSS 3 ZAHRNUJE: SELEKTORY, POZADÍ A RÁMEČKY, BOX MODEL, OBRÁZKY A OBSAH, KTERÝ JE NAHRAZUJE, TEXTOVÉ EFEKTY, 2D A 3D TRANSFORMACE, ANIMACE, VÍCESLOUPCOVÝ LAYOUT, UŽIVATELSKÉ ROZHRANÍ.
CSS SE DO WEB STRÁNKY VKLÁDÁ TŘEMI MOŽNÝMI ZPŮSOBY:

- **PŘÍMO = IN-LINE** VKLÁDÁNÍ **PŘÍMO DO TAGU HTML, KTERÝ CHCEME STYLOVAT**: ** Tento odstavec bude modrý. ** VLOŽÍ SE **POMOCÍ ATRIBUTU STYLE**. TENTO ZPŮSOB SE NORMÁLNĚ NEPOUŽÍVÁ, PROTOŽE ZNAČNĚ ZNEPŘEHLEDŇUJE HTML KÓD A SNIŽUJE JEHO UDRŽITELNOST. PŘESTO JE POUŽÍVÁN ZEJMÉNA PŘI STYLOVÁNÍ EMAILŮ, KDY JE TO ČASTO JEDINÝ ZPŮSOB, KDY STYLY ZAFUNGUJÍ.
- **DO HLAVIČKY HTML DOKUMENTU. V ČÁSTI <head></head> DO TAGU STYLE**. PŘÍKLAD NÍŽE:
```
<style type="text/css">
   a {text-decoration: none}          /* Odkazy nebudou podtrhávané */
   u {font-weight: bold; color: blue} /* Co je podtržené, bude navíc tučné a modré */
</style>
```
- **DO EXTERNÍHO SOOUBORU**, KTERÝ JE **V HLAVIČCE DOKUMENTU SLINKOVÁN - PŘIPOJEN** K SOUBORU HTML. TENTO **SOUBOR MÁ PŘÍPONU CSS.** PŘÍKLAD NÍŽE:
```
<link rel="stylesheet" type="text/css" href="styly.css">
```

SYNTAX CSS: SE SKLÁDÁ ZE **SELEKTORŮ** A **DEKLARACE**. SELEKTORY URČUJÍ, PRO KTERÉ ČÁSTI WEB STRÁNKY DEKLARACE PLATÍ. EXITUJE NĚKOLIK DRUHŮ SELEKTORŮ. **BLOK DEKLARACÍ JE OHRANIČEN SLOŽENÝMI ZÁVORKAMI** A **OBSAHUJE JEDNU NEBO VÍCE DEKLARACÍ**, **ODDĚLENÝCH STŘEDNÍKEM.** KAŽDÁ **DEKLARACE SE SKLÁDÁ Z VLASTNOSTI A Z JEJÍ HODNOTY, KTERÉ JSOU ODDĚLENY DVOJTEČKOU.**

DRUHY SELEKTORŮ:

- **HVĚZDIČKOVÝ SELEKTOR** – URČUJE HLAVNÍ **STYL PRO CELÝ DOKUMENT** NEBO VŠECHNY TAGY URČITÉHO TYPU. PŘ: *** {color: red}**
- **TYPOVÝ – ELEMENTOVÝ SELEKTOR** – URČUJE **STYL PRO URČITÉ TYPY ELEMENTŮ**. PŘ: **div {color: red}**
- **ATRIBUTOVÝ SELEKTOR** – URČUJE **STYL PRO TAG OBSAHUJÍCÍ URČITÝ ATRIBUT**. PŘ: **h1[id] {color: red}**
- **SELEKTOR IDENTIFIKÁTORU – ID** – URČUJE **STYL TAGU, KTERÝ MÁ ATRIBUT ID O KONKRÉTNÍ HODNOTĚ.** VĚTŠINOU SE POUŽÍVÁ **PRO PRVEK, KTERÝ JE V RÁMCI STRÁNKY JEDINEČNÝ. NA WEB STRÁNCE BY MĚL MÍT POUZE JEDEN ELEMENT KONKRÉTNÍ ID.** PŘ: **#idecko {color: blue}** UPRAVUJE STYL ELEMENTU, KTERÝ JE V HTML NAPŘ: ** Modrý text **
- **SELEKTOR TŘÍDY – CLASS** – URČUJE **STYL TAGU, KTERÝ MÁ ATRIBUT CLASS O KONKRÉTNÍ HODNOTĚ**. VĚTŠINOU **NA ELEMENTY, KTERÉ SE NA STRÁNCE NĚKOLIKRÁT OPAKUJÍ**. **NA STRÁNCE MŮŽE EXISTOVAT VÍCE NEŽ JEDEN TAG S KONKRÉTNÍ TŘÍDOU.** MOHOU TO BÝT POLOŽKY MENU, ZOBRAZENÍ PRODUKTŮ V JEJICH SEZNAMU ATD. PŘ: **.trida {color: red} UPRAVUJE STYL ELEMENTU, KTERÝ JE V HTML ZAPSÁN NAPŘ: Červený text **
- **PSEUDOTŘÍDY UŽIVATELSKÝCH AKCÍ A ODKAZŮ**– URČUJE **STYL TAGU NAPŘÍKLAD U ODKAZŮ, JESTLI JIŽ BYL ODKAZ NAVŠTÍVEN, ZDA TO JE ODKAZ, POKUD SE NA NĚJ NAJELO MYŠÍ, ATD. PŘ: a:visited {color: blue}.** PŘÍKLADY POUŽÍVANÝCH **PSEUDOTŘÍD: link, visited, hover, active, focus**

NEJZNÁMNĚJŠÍ CSS VLASTNOSTI:

POZADÍ	**background-color**, background-image
FONTY, TEXT	**font-family: font, font-style: styl, font-weight: tučnost, font-size: velikost, color, text-align: zarovnání textu**, text-decoration: podtržení, přeškrtnutí, **text-indent: odsazení 1. řádku**
VIDITELNOST, VRSTVY	visibility (hidden, visible), **z-index** (hodnota představuje vrstvy, vyšší číslo, výše), display
POZICOVÁNÍ	**position (relative – vůči nejbližšímu rodiči pozicovanému absolutně, absolute – vůči levému hornímu rohu obrazovky, fixed – vůči obrazovce, nehýbá se, i když se skroluje), left, right, top, bottom**
OBRÁZKY	**border, height, width**
ODKAZY	:link, :visited, :hover, :active
RÁMEČKY, OKRAJE, VNITŘNÍ OKRAJE	border-color, border-style, border-width, margin, padding
ROZMĚRY	**height, width**

VLOŽENÍ STYLU PRO TISK: <link rel="stylesheet" type="text/css" media="print" href="mystyle.css">
VLOŽENÍ STYLU PRO JINÁ MÉDIA – TABLETY, … <link rel="stylesheet" type="text/css" media="handheld" href="foo.css">
DÉLKOVÉ JEDNOTKY V CSS: % - VŮČI ELEMENTU, VE KTERÉM SE NACHÁZÍ, **px** – PIXELY NA OBRAZOVCE, **mm** – milimetry, …
BARVY V CSS: V RGB KÓDU: PŘ: p{color:#FF0000;}, NEBO KLÍČOVÝM SLOVEM: PŘ: p{color:red;}
DĚDĚNÉ VLASTNOSTI CSS STYLŮ OD RODIČOVSKÝCH ELEMENTŮ: font- většina vlastností písma, color, text-align, …
I CSS LZE VALIDOVAT: https://jigsaw.w3.org/css-validator/, **FRAMEWORKY PRO SNAZŠÍ STYLOVÁNÍ**: http://getbootstrap.com/, http://purecss.io/
TERMÍNY: KASKÁDOVÉ STYLY, CSS3, INLINE, LINK, SLINKOVÁNÍ, SELEKTOR, DEKLARACE, VLSTNOST, HODNOTA, ID, TŘÍDA, PSEUDOTŘÍDA, VLASTNOSTI CSS A JEJICH PŘÍKLADY, DÉLKOVÉ JEDNOTKY, BARVY
PRAKTICKÁ DOVEDNOST: – NASTYLOVAT JEDNODUCHÝ ODSTAVEC TEXTU: BARVA, FONT, ZAROVNAT, ODSADIT
DOPORUČENÉ ZDROJE: http://owebu.org/cze/css/index.php, http://www.w3schools.com/css/, http://programujte.com/clanek/2005052003-css-1-lekce/, http://www.tutorialspoint.com/css/what_is_css.htm, http://css3test.com/, https://www.w3.org/Style/CSS/specs, https://developer.mozilla.org/cs/docs/Web/CSS, http://jecas.cz/css-selektory, http://www.tvorba-webu.cz/css/,

17. Javascript

JAVASCRIPT JE SKRIPTOVACÍ, INTERPRETOVANÝ, SLABĚ TYPOVANÝ, OBJEKTOVĚ ORIENTOVANÝ JAZYK **POUŽÍVANÝ PRO VZNIK DYNAMICKY REAGUJÍCÍCH WEBOVÝCH STRÁNEK OBSAHUJÍCÍ INTERAKTIVNÍ PRVKY. JAVASCRIPT BĚŽÍ NA STRANĚ KLIENTA V JEHO INTERNETOVÉM PROHLÍŽEČI**. **VYTVOŘEN** FIRMOU NETSCAPE V ROCE 1995 BĚHEM TŘÍ TÝDNŮ. JAVASCRIPT NENÍ JAVA, JE TO ÚPLNĚ JINÝ JAZYK! V ROCE 1996 MICROSOFT IMPLEMENTUJE JSCRIPT. V ROCE 1998 STANDARDIZOVÁN JAKO ECMASCRIPT. JAVASCRIPT JE OBCHODNÍ NÁZEV ECMASCRIPTU. DLOUHOU DOBU BYLA IMPLEMENTACE V RŮZNÁ U RŮZNÝCH PROHLÍŽEČŮ.

POUŽITÍ JAVASCRIPTU: ROZBALOVACÍ MENU, ROLETKY, MĚNÍCÍ SE BANNERY, GALERIE OBRÁZKŮ, RŮZNÉ EFEKTY NA STRÁNKÁCH, VALIDACE INTERNETOVÝCH FORMULÁŘŮ, UPDATE ČÁSTÍ STRÁNEK BEZ RELOADU, ONLINE WYSIWYG EDITORY. NENÍ DOBRÉ NIC, NA ČEM ZÁVISÍ PŘÍLIŠ BEZPEČNOST. ANGRYBIRDS PRO PROHLÍŽEČ CHROME V JAVASCRIPTU. JAVASCRIPTOVÁ APLIKACE JE MULTIPLATFORMNÍ: KDE BĚŽÍ PROHLÍŽEČ, VĚTŠINOU BĚŽÍ I JAVASCRIPT: POČÍTAČ, TABLET, MOBIL… JAVASCRIPT SE ZAČÍNÁ POUŽÍVAT I NA SERVERU: NODE.JS

OBECNĚ JAVASCRIPT MŮŽE UPRAVOVAT HTML TAGY, JEJICH ATRIBUTY, CSS VLASTNOSTI, VALIDOVAT VSTUPNÍ HODNOTY VE FORMULÁŘÍCH.

JAVASCRIPT LZE VKLÁDAT JAK DO HEAD TAK I DO BODY ČÁSTI HTML DKOUMENTU. DOPORUČUJE SE ALE VKLÁDAT POUZE ODKAZ NA JAVASCRIPTOVÝ SOUBOR A NEDÁVAT JAVASCRIPTOVÉ PŘÍKAZY A DLOUHÉ PROGRAMY PŘÍMO DO DOKUMENTU. SKRIPT V JAVASCRIPTU MŮŽEME VKLÁDAT NEOMEZENĚKRÁT JAK DO HLAVIČKY TAK TĚLA DOKUMENTU.

LINK NA JAVASCRIPTOVÝ SOUBOR V HLAVIČCE DOKUMENTU (SAMOSTATNÉ SOUBORY S JAVASCRIPTEM MAJÍ PŘÍPONU .JS):

```
<script type="text/javascript" src="soubor_s_javascriptem.js"></script>
```

VÝHODA EXTERNÍCH SOUBORŮ: ODDĚLENÍ JEDNOTLIVÝCH KÓDŮ (HTML A JAVASCRIPTU) – PŘEHLEDNĚJŠÍ. NACACHOVANÉ SOUBORY JAVASCRIPTU URYCHLUJÍ NAHRÁNÍ WEB STRÁNKY.

JAVASCRIPTOVÝ KÓD VLOŽENÝ DO WEB STRÁNKY (MŮŽE SE OBJEVIT V HLAVIČCE I TĚLE STRÁNKY):

```
<script type="text/javascript">
.. javascript tělo skriptu ..
</script>
```

```
// JEDNOŘÁDKOVÝ KOMENTÁŘ
/* TOTO JE DELŠÍ VÍCEŘÁDKOVÝ
KOMENTÁŘ */
```

SYNTAX JAZYKA:

PŘÍKAZY SE ODDĚLUJÍ STŘEDNÍKEM. HODNOTY JSOU V JAVASCRIPTU BUĎ V **PROMĚNNÝCH**, NEBO V **LITERÁLECH**. **TEXT** SE MŮŽE ZAPISOVAT **JAK DO DVOJITÝCH TAK I DO JEDNODUCHÝCH UVOZOVEK**. **PROMĚNNÉ SE DEKLARUJÍ** POMOCÍ KLÍČOVÉHO SLOVA **VAR**: var x; **INICIALIZACE PROMĚNNÉ**: x=6; **OPERÁTORY**: +-*/=%++--, OPERÁTORY JSOU ARITMETICKÉ, LOGICKÉ RELAČNÍ. ŘETĚZCE MŮŽEME SPOJOVAT OPERÁTOREM +. **KOMENTÁŘE**: **JEDNOŘÁDKOVÉ** //, **VÍCEŘÁDKOVÉ** KOMENTÁŘE ZAČÍNAJÍ /* A KONČÍ */. PŘÍKAZY A VŠE UVNITŘ KOMENTÁŘŮ SE BĚHEM PRŮBĚHU PROGRAMU NEVYKONÁVAJÍ. **IDENTIFIKÁTORY** JSOU JMÉNA SLOUŽÍCÍ K OZNAČENÍ PROMĚNNÝCH. PRVNÍ ZNAK IDENTIFIKÁTORU MUSÍ BÝT PÍSMENO, PODTRŽÍTKO NEBO ZNAK DOLARU. DALŠÍ ZNAKY MOHOU BÝT I ČÍSLA.

U IDENTIFIKÁTORŮ ZÁLEŽÍ NA VELIKOSTI PÍSMEN. prvnijmeno A prvniJmeno JSOU DVA ODLIŠNÉ IDENTIFIKÁTORY DVOU ODLIŠNÝCH PROMĚNNÝCH. TO SAMÉ PLATÍ PRO JAKÁKOLIV KLÍČOVÁ SLOVA JAZYKA JAVASCRIPT.

PŘÍKLAD DYNAMICKÉHO TYPOVÁNÍ:

```
var x;          // x je nyní nedefinováno
var x = 5;      // x je nyní číslo
var x = "John"; // x je nyní řetězec
```

DEKLARACE PROMĚNNÉ: var jmenoAuta;

DEKLARACE A INICIALIZACE VÍCE PROMĚNNÝCH NAJEDNOU:

```
var person = "John Doe", carName = "Volvo", price = 200;
```

PROMĚNNÁ, KTERÁ **NENÍ INICIALIZOVANÁ** MÁ HODNOTU **undefined**.

POKUD PROMĚNNOU **DEKLARUJEME ZNOVU**, TAK NEZTRÁCÍ SVOJI HODNOTU.

PODMÍNKA – VĚTVENÍ PROGRAMU:

```
if (hodina < 18) {
    pozdrav = "Dobrý den";
} else {
    pozdrav = "Dobrý večer";
}
```

DATOVÉ TYPY SE POZNAJÍ PODLE HODNOT, KTERÝMI PROMĚNNOU INICIALIZUJEME:

```
var auto = "Opel Astra";                  // řetězec
var počet_valcu = 5;                      // číslo
var automaticka_prevodovka = false;       // boolean – logická hodnota
var auta = ["Saab", "Volvo", "BMW"];      // pole
```

CYKLY:

S PODMÍNKOU NA ZAČÁTKU

```
while (i < 10) {
    text += "Číslo je " + i;
    i++;
}
```

S PODMÍNKOU NA KONCI

```
do {
    text += "Číslo je " + i;
    i++;
} while (i < 10);
```

S DANÝM OPAKOVÁNÍM

```
for (i = 0; i < auta.length; i++) {
    text += auta[i] + "<br>";
}
```

KLÍČOVÁ SLOVA: var, if … else, switch, return, for, do … while, function, break, continue…

ŘÍDÍCÍ KONSTRUKCE ZAHRNUJÍ VĚTVENÍ (IF, SWITCH), CYKLY (FOR, WHILE, DO).

VÝSTUPY – VÝPISY: UPOZORŇOVACÍ DIALOG: window.alert("POZOR NA JAVASCRIPT!");

TEXT NA MÍSTO V HTML KÓDU, KDE SE UMÍSTÍ SKRIPT.document.write(5 + 6);

VLOŽENÍ DO URČITÉHO TAGU: document.getElementById("demo").innerHTML = 5 + 6;

ÚPRAVA KONKRÉTNÍHO TAGU POMOCÍ id: document.getElementById("toto_id").

ZJIŠTĚNÍ HODNOTY V EDITAČNÍM PRVKU: x = document.getElementById("numb").value;

SPUSTÍ PO KLIKNUTÍ NA TLAČÍTKO: <button type="button" onclick="máFunkce()">Click Me!</button>

DOM – DOCUMENT OBJECT MODEL – V HTML JE VŠE JAKO UZEL. UZLY JSOU VNOŘENÉ.

AJAX: UMOŽNÍ NA POZADÍ BEZ RELOADU KOMUNIKOVAT SE SERVEREM, UPDATOVAT ČÁSTI WEBU, **JAVASCRIPTOVÉ FRAMEWORKY** ŘEŠÍ KOMPLEXNÍ ÚLOHY JEDNODUŠE. NAPŘ. **JQUERY**, **MOOTOOLS**, **LADĚNÍ JAVASCRIPTU:** VĚTŠINA MODERNÍCH BROWSERŮ MÁ DEBUGGER.

TERMÍNY: PROMĚNNÉ, TERMÍNY JAKO U C, SLABĚ TYPOVANÝ, DYNAMICKÉ TYPOVÁNÍ, DOM, AJAX, …

PRAKTICKÁ DOVEDNOST: VLOŽIT SKRIPT DO HTML, VLOŽIT ODKAZ NA JS SOUBOR DO HTML, VYVOLAT UPOZORŇOVACÍ OKNO V JAVASCRIPTU, ZMĚNIT NĚKTERÉ CSS VLASTNOSTI PRVKU O URČITÉM ID, ZJISTIT HODNOTU PRVKU FORMULÁŘE

FUNKCE:

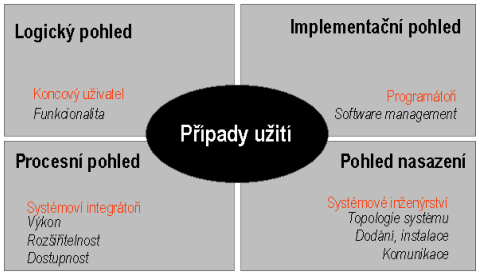
```
function nasob(a, b) { // DEFINICE
    return a * b;
}
var x = nasob(4, 3); // VOLÁNÍ FUNKCE
```

DOPORUČENÉ ZDROJE: http://www.itnetwork.cz/javascript/zaklady, https://jquery.com/, http://www.javascript.cz/?chap=4&cat=1, https://developer.mozilla.org/cs/docs/Web/JavaScript, http://www.tvorba-webu.cz/javascript/, http://www.w3schools.com/js/, http://www.w3schools.com/jsref/,

PŘÍKLADY: http://www.w3schools.com/js/js_examples.asp,

18. Vývoj software, metody, workflow

4+1 POHLEDY NA ARCHITEKTURU A VÝVOJ:



- LOGICKÝ – CO BY MĚL SYSTÉM VYKONÁVAT (DATA A FUNKCE)
- IMPLEMENTAČNÍ – JAK TO ZREALIZOVAT POMOCÍ KOMPONENT A ČÁSTÍ, ROZČLENĚNÍ, ZDROJOVÝ KÓD - PROGRAMÁTOŘI
- PROCESNÍ – CHOVÁNÍ, ODEZVA SYSTÉMU, ROZŠIŘITELNOST, VÝKONNOST, PROPUSTNOST SYSTÉMU
- NASAZENÍ – NAVÁZÁNÍ SYSTÉMU NA JIŽ EXISTUJÍCÍ SW A HARDWARE, NASAZENÍ, INSTALACE, LADĚNÍ VÝKONU
- PŘÍPADY UŽITÍ – SPOJUJÍ PŘEDEŠLÉ POHLEDY V KONKRÉTNÍCH PŘÍPADECH POUŽITÍ, CO SE MŮŽE STÁT

KOMPONENTA: SAMOSTATNĚ NAVRŽENÁ, VYVINUTÁ A OTESTOVANÁ ČÁST SYSTÉMU. NAVRHUJÍ SE S POMOCÍ DEKOMPOZICE.
DEKOMPOZICE: ROZDĚLENÍ SLOŽITĚJŠÍHO PROBLÉMU NA PODPROBLÉMY. DEKOMPOZICE HORIZONTÁLNÍ A VERTIKÁLNÍ.

- **HORIZONTÁLNÍ DEKOMPOZICE:** KOMPONENTY STEJNÉ VRSTVY, TYPU ŘEŠÍ RŮZNÉ ÚLOHY.
- **VERTIKÁLNÍ DEKOMPOZICE:** VZNIKAJÍ VRSTVY KOMPONENT. **MOŽNÉ VRSTVY JSOU:**
  - PRÁCE S DATY APLIKACE
  - LOGIKA APLIKACE – ALGORITMICKÉ ZPRACOVÁNÍ DAT, STRUKTURA TŘÍD
  - UŽIVATELSKÉ ROZHRANÍ

WORKFLOW:
POSTUP PRÁCE, KROKY, KTERÉ SE MUSÍ POSTUPNĚ PROJÍT

ABSTRAKCE: ZJEDNODUŠENÝ POHLED NA SLOŽITÉ VĚCI, ZANEDBÁNÍ NEPODSTATNÝCH VĚCÍ A FUNKCÍ.
ŽIVOTNÍ CYKLUS SOFTWARU: ANALÝZA PROBLÉMU (SPECIFIKACE) | NÁVRH PROJEKTU (STRUKTURA PROGRAMU) | IMPLEMENTACE PROJEKTU | TEST (TEST ČÁSTÍ | TEST CELKU) | DOKUMENTACE|INSTALACE|UDRŽOVÁNÍ
KRIZE VE VÝVOJI SOFTWARU, PŘELOM 60. – 70. LET 20. STOLETÍ:
RYCHLE NAROSTL VÝKON POČÍTAČŮ A KOMPLEXNOST ŘEŠENÝCH PROBLÉMŮ, SOFTWAROVÁ TECHNOLOGIE NEBYLA SCHOPNÁ ÚSPĚŠNĚ ŘEŠIT TAK KOMPLEXNÍ PROBLÉMY, **VÝSLEDEK:** NEUDRŽITELNÝ KÓD, PŘETAŽENÉ ROZPOČTY, PŘETAŽENÉ ČASOVÉ PLÁNY, ŠPATNÁ KVALITA SOFTWARU, SOFTWARE ČASTO NESPLNIL OČEKÁVÁNÍ

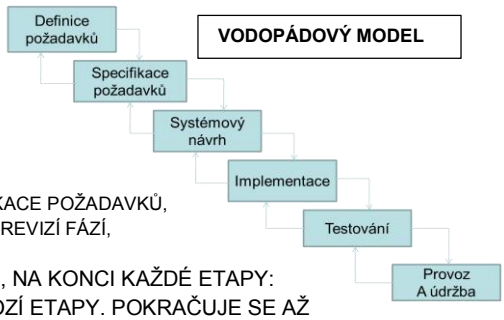
PROBLÉMY PŘI VÝVOJI SOFTWARU:
ZPOŽDĚNÍ, CHYBOVOST, NESPRÁVNÁ FUNKČNOST, MALÝ VÝKON, PŘÍLIŠ KOMPLIKOVANÉ UŽIVATELSKÉ PROSTŘEDÍ, SLOŽITÁ UDRŽOVATELNOST PROGRAMU

ŘEŠENÍ PROBLÉMŮ VE VÝVOJI SOFTWARU:

- ŠPATNÁ KOMUNIKACE: **ODDĚLENÍ ANALYTIKA OD VÝVOJÁŘE**
- NESPRÁVNÝ PŘÍSTUP K VÝVOJI: **DŮRAZ NA TÝMOVOU SPOLUPRÁCI, HLAVNÍ JE SPOKOJENOST ZÁKAZNÍKA**
- ŠPATNÉ PLÁNOVÁNÍ: **VZNIK METODIK VÝVOJE SOFTWARU**
- NÍZKÁ PRODUKTIVITA: **TÝM S PŘIDĚLENÝMI ROLEMI, KOORDINACE VÝVOJE, VÝVOJ PODLE SPECIFIKACÍ**
- PODCENĚNÍ HROZEB A RIZIK: **METODIKY PRO ANALÝZU RIZIK**

PŘÍČINY PROBLÉMŮ PŘI VÝVOJI SOFTWARU:
PODCENĚNÍ PROJEKTU A ŠPATNÝ ODHAD (ČAS, NÁKLADY), ŠPANÁ KOMUNIKACE (ZÁKAZNÍK - PROGRAMÁTOR), ŠPATNÝ PŘÍSTUP K VÝVOJI (PROGRAMÁTOR SE PŘEDVÁDÍ), ŠPATNÉ ZADÁNÍ, NEDOSTATEČNÁ ANALÝZA, PŘÍLIŠ SLOŽITÝ PROJEKT, NÍZKÁ PRODUKTIVITA (KONCENTRACE NA ŠPATNÉ ÚKOLY), PŘEHNANÝ DŮRAZ NA TECHNOLOGII (NOVINKY BEZ ZKUŠENOSTI), ŠPATNÁ KVALITA PROGRAMOVÉHO KÓDU, NEVHODNÉ METODIKY, POSTUPY, TECHNOLOGIE, NEDOSTATEČNÉ TESTOVÁNÍ, ŠPATNÉ PROJEKTOVÉ ŘÍZENÍ, PLÁNOVÁNÍ

VODOPÁDOVÝ MODEL



METODIKY VÝVOJE SOFTWARU A ŽIVOTNÍ CYKLUS:

- **TRADIČNÍ METODIKY (ANALÝZA, PROJEKTOVÝ MANAGEMENT)**
  - MODEL „NAPIŠ – OPRAV“ (BUILD AND FIX) = IMPLEMENTACE, DODÁNÍ, OPRAVA CHYB
  - STRIKTNÍ POSLOUPNOST FÁZÍ (STAGEWISE) FÁZE (DEFINICE PROBLÉMU, ANALÝZA, SPECIFIKACE POŽADAVKŮ, NÁVRH, ARCHITEKTURA, IMPLEMENTACE + TESTOVÁNÍ, PROVOZ) – BEZ ZPĚTNÉ VAZBY, BEZ REVIZÍ FÁZÍ, POŽADAVKŮ, HODNOCENÍ RIZIK
  - **VODOPÁDOVÝ MODEL (WATERFALLI)** – KAŽDÁ ETAPA MÁ JASNĚ STANOVENÝ CÍL, NA KONCI KAŽDÉ ETAPY: JEJÍ ZHODNOCENÍ, PŘÍPADNĚ OPRAVENÍ, MŮŽEME SE VRÁTIT ZPĚT DO PŘEDCHOZÍ ETAPY. POKRAČUJE SE AŽ TECHDY, JE-LI ETAPA ZCELA DOKONČENA A SCHVÁLENA.
    - **VÝHODY:** JEDNODUCHÝ, IDEÁLNÍ PRO ŘÍZENÍ, DISCIPLÍNA VE VÝVOJI
    - **NEVÝHODY:** NEPRUŽNOST, MEZI ANALÝZOU A NASAZENÍM SE MOHOU ZMĚNIT POŽADAVKY
  - **SPIRÁLOVÝ MODEL** – ITERATIVNÍ PŘÍSTUP, OPAKOVANÁ DŮSLEDNÁ ANALÝZA RIZIK. TENTO MODEL VYCHÁZÍ Z TOHO, ŽE NA POČÁTKU JE OBTÍŽNÉ, NE-LI NEMOŽNÉ PŘESNĚ URČIT VŠECHNY FUNKCE.
    - **VÝHODY:** PROSTŘEDÍ PRO VÝVOJ ZNOVU POUŽITELNÝCH KOMPONENT, VHODNÝ I PRO SLOŽITÉ PROJEKTY, VČAS VYLOUČÍ NEVHODNÁ ŘEŠENÍ
    - **NEVÝHODY:** KOMPLIKOVANOST, SOFTWARE NENÍ UVOLNĚN PŘED UKONČENÍM POSLEDNÍHO CYKLU, NEPRUŽNÝ PRO INTERNETOVÉ APLIKACE, VHODNÁ ZMĚNA POŽADAVKŮ POUZE PO DOKONČENÍ CYKLU
  - **DALŠÍ METODIKY**: PŘÍRŮSTKOVÝ – PROTOTYPOVÝ, **RUP – RATIONAL UNIFIED PROCESS** VELKÉ PROJEKTY

- **AGILNÍ METODIKY (RYCHLEJŠÍ VÝVOJ, SPOJENÍ ZÁKAZNÍK-VÝVOJÁŘI, REAKCE NA ZMĚNY)**
  - EXTRÉMNÍ PROGRAMOVÁNÍ (KOMUNIKACE, JEDNODUCHOST, ZPĚTNÁ VAZBA, ODVAHA)
  - TEST DRIVEN DEVELOPMENT (TESTOVACÍ KÓD HOTOV PŘED ZAČÁTKEM PSANÍ KÓDU) – KVALITNÍ SOFTWARE
  - ADAPTIVNÍ VÝVOJ SW, LEAN DEVELOPMENT, SCRUM, CRYSTAL

TERMÍNY:KOMPONENTA, DEKOMPOZICE, ABSTRAKCE, METODIKY, VODOPÁDOVÝ, SPIRÁLOVÝ MODEL, AGILNÍ METODIKA, EXTRÉMNÍ PROGRAMOVÁNÍ, TEST DRIVEN DEVELOPMENT, WORKFLOW, ŽIVOTNÍ CYKLUS SOFTWARU, KRIZE VE VÝVOJI SW.:
DOPORUČENÉ ZDROJE: http://blog.nny.cz/740763-klasicke-a-agilni-metodiky-vyvoje-software.php, http://www.is.mendelu.cz/eknihovna/opory/index.pl?cast=18357;lang=cz, http://homel.vsb.cz/~dan11/is2011/4%20Informacni%20systemy%20-%20SW%20inzenyrstvi.ppt, https://edux.fit.cvut.cz/oppa/BI-SI1/prednasky/BI-SI1-P10m.pdf, http://slideplayer.cz/slide/1965124/

19. Coding standards

ŠTÁBNÍ KULTURA – CODING STANDARDS PŘEDSTAVUJE ZPŮSOB PSANÍ ZDROJOVÉHO KÓDU PROGRAMOVACÍHO JAZYKA, VYTVÁŘENÍ INDENTIFIKÁTORŮ ATD.
CODING STANDARDS SE ZAVÁDĚJÍ **Z DŮVODU JEDNOTNÉ STRUKTURY ZDROJOVÉHO KÓDU PŘI PRÁCI V TÝMU**, A OBECNĚ Z DŮVODU **ZVÝŠENÍ PŘEHLEDNOSTI KÓDU** A TAK JEHO **LEPŠÍ UDRŽITELNOSTI**.
CO OVLIVŇUJÍ CODING STANDARDS:

- **JMÉNA**
  - **VÝZNAM**
    - VHODNĚ ZVOLENÉ, POPISNÉ, ALE NEPŘÍLIŠ DLOUHÉ (RYCHLÉ PSANÍ A SNADNÉ PAMATOVÁNÍ),
    - POZOR NEZAPLNIT JMENNÝ PROSTOR JMÉNY, KTERÉ SE HODÍ V BUDOUCNU,
  - **ZPŮSOB VYTVÁŘENÍ NÁZVŮ:**
    - POUŽITÉ ZNAKY – VĚTŠINOU BEZ MEZER
    - OMEZENÍ MAXIMÁLNÍ DÉLKY JMÉNA PRO IDENTIFIKACI
    - ODDĚLENÍ VÍCE SLOV
      - **POMOCÍ ROZDĚLOVNÍKU**: parkovaci-misto (UNIX, CSS, COBOL, LISP)
      - **POMOCÍ PODTRŽÍTKA**: parkovaci_misto (C, Python, FORTRAN, ALGOL) **snake_case**, VELKÁ PÍSMENA: **MACRO_CASE**
      - **POMOCÍ VELKÝCH POČÁTEČNÍCH PÍSMEN:** ParkovaciMisto (Pascal, Java, C#, Visula Basic) **CamelCase. UpperCamelCase** – ZAČÍNÁ S VELKÝM PÍSMENEM, **lowerCamelCase** – ZAČÍNÁ S MALÝM PÍSMENEM
    - PRVNÍ PÍSMENO MŮŽE BÝT VELKÉ ČI MALÉ (VELKÉ NAPŘÍKLAD U NÁZVŮ TŘÍD, METODY A PROMĚNNÉ)
    - VŠECHNA PÍSMENA VELKÁ NEBO MALÁ
    - PÍSMENA NA ZAČÁTKU, BUĎ ODDĚLENÁ PODTRŽÍTKEM, NEBO JEN MALÁ NA ROZDÍL OD OSTATNÍCH. NAPŘÍKLAD g_ BUDE NA ZAČÁTKU GLOBÁLNÍ PROMĚNNÉ. sObsah – NAPŘÍKLAD ŘETĚZEC – STRING. PÍSMENA NA ZAČÁTKU, NEBO PODTRŽÍTKA MOHOU OZNAČOVAT TAKÉ, ZDA JE PROMĚNNÁ NEBO METODA PRIVÁTNÍ - PRIVATE, VEŘEJNÁ - PUBLIC NEBO PROTECTED
  - **JMÉNA FUNKCÍ**
    - JMÉNO MUSÍ, JASNĚ POPSAT CO DĚLÁ, POUŽÍT SLOVESO,
    - PŘÍPONY NAPŘ: MAX, MIN, KEY
    - PŘEDPONY: IS (OTÁZKA NA NĚCO), SET (NASTAVOVÁNÍ STAVU), GET (ZÍSKÁNÍ STAVU)
- **FORMÁTOVÁNÍ**
  - ŘÍDÍCÍ KONSTRUKCE

NAPŘ: C, C++, PHP
PODMÍNKA – VĚTVENÍ PROGRAMU

```
if (PODMÍNKA) {
        KÓD V PŘÍPADĚ ŽE JE PODMÍNKA SPLNĚNA
} else {
        KÓD V PŘÍPADĚ ŽE PODMÍNKA SPLNĚNA NANÍ
}
```

PODMÍNKA V JEDNÉ ŘÁDCE

```
if (PODMÍNKA) nejakaHodnota = 2;
```

U PODMÍNEK POUŽÍVAT PROMĚNNOU VLEVO A LITERÁL V PRAVO.

CYKLUS DO WHILE

```
do {
        jmeno_funkce(parametr);
} while (PODMÍNKA);
```

CYKLUS WHILE

```
while (PODMÍNKA) {
        OBSAH CYKLU
}
```

PŘEPÍNAČ SWITCH

```
switch (PROMĚNNÁ)
{
        case 'B':
          color = BLACK;
          break;
        case 'Y':
          color = YELLOW;
          break;
        default:
          color = GREEN;
          break;
}
```

- OSTATNÍ
  - **KÓDOVÁNÍ** - VĚTŠINOU UTF8, **CELÝ DOKUMENT JEN V JEDNOM JAZYCE, NEKOMBINOVAT ČEŠTINU A ANGLIČTINU!**
  - **DÉLKA ŘÁDKY** (V JAZYCE C NAPŘ. NE VÍCE NEŽ 78 ZNAKŮ, NEBO JSOU LIMITY 80, 120 – ŠIROKÉ MONITORY)
  - **POČET PŘÍKAZŮ NA ŘÁDKU** – ČASTO BÝVÁ 1 PŘÍKAZ NA ŘÁDKU A PŘI DEKLARACÍCH JEDNA PROMĚNNÁ. NA KONCI KAŽDÉHO ŘÁDKU STŘEDNÍK – NAPŘ. U JAZYKA C
  - **ODSAZENÍ** – PRŮZKUM UKÁZAL, ŽE IDEÁLNÍ ODSAZENÍ JE 4 MEZERY (REALIZOVÁNO MEZERAMI NEBO TAB.)
  - **POUŽITÍ PRÁZDNÝCH ŘÁDEK** – PRO ZLEPŠENÍ POCHOPITELNOSTI KÓDU, ODDĚLOVAT ČÁSTI, KTERÉ K SOBĚ PATŘÍ
  - **NÁZVY SOUBORŮ SE ZDROJOVÝM KÓDEM** – NAPŘÍKLAD STEJNÝ JAKO NÁZEV TŘÍD, KTERÉ OBSAHUJÍ
- **PSANÍ KOMENTÁŘŮ:**
  - PROMĚNNÉ: ZA JEJICH DEKLARACÍ, VYSVĚTLUJÍ VÝZNAM // NEBO /* */ U JAZYKA C
  - METODY A FUNKCE: PŘED NIMI, POPISUJÍ FUNKČNOST, NÁVRATOCVÉ HODNOTY, PARAMETRY
  - CELÉ HLAVIČKOVÉ SOUBORY A PROGRAMY: AUTOR, DATUM VZNIKU, TO DO, CO JEŠTĚ UDĚLAT

TERMÍNY: CODING STANDARDS, UDRŽITELNOST, ČITELNOST, IDENTIFIKÁTORY, CAMELCASE, FORMÁTOVÁNÍ, KOMENTÁŘE, ODSAZENÍ
PRAKTICKÉ DOVEDNOSTI: napsat kód v C nebo v C++ podle coding standardu
DOPORUČENÉ ZDROJE: http://umich.edu/~eecs381/handouts/C_Coding_Standards.pdf, **CODING STANDARD C:** https://users.ece.cmu.edu/~eno/coding/CCodingStandard.html, http://homepages.inf.ed.ac.uk/dts/pm/Papers/nasa-c-style.pdf, https://www.gnu.org/prep/standards/html_node/Writing-C.html, http://ieng9.ucsd.edu/~cs30x/indhill-cstyle.html, **CODING STANDARD C++:** https://users.ece.cmu.edu/~eno/coding/CppCodingStandard.html, **CODING STANDARD PHP - PEAR:** https://pear.php.net/manual/en/standards.php, **CODING STANDARD NETTE:** https://nette.org/en/coding-standard,

20. Komentování zdrojového kódu

TYPY KOMENTÁŘŮ: V JAZYCE C, C++, PHP, JAVASCRIPT, JAVA, C# POUŽÍVÁME
JEDNOŘÁDKOVÉ KOMENTÁŘE: // ZA DVOJITÝM LOMÍTKEM NAJDEME TEXT KOMENTÁŘŮ
VÍCEŘÁDKOVÉ KOMENTÁŘE: /* LOMÍTKO A HVĚZDIČKA ZAČÍNÁ BLOK DELŠÍHO TEXTU NA VÍCE ŘÁDKŮ, KTERÝ MŮŽE POPISOVAT DELŠÍ OBLAST KÓDU. NA NĚKTERÉM Z DALŠÍCH ŘÁDKŮ SE UKONČÍ POMOCÍ HVĚZDIČKY A LOMÍTKA */

```
/**
* VĚTŠÍ ČÁST KÓDU, NAPŘÍKLAD KNIHOVNA, TŘÍDA, FUNKCE SE KOMENTUJE KOMENTÁŘEM, KTERÝ MÁ NÁSLEDUJÍCÍ FORMÁT.
* OBSAHUJE VĚTŠINOU KRÁTKÝ POPIS ČÁSTI, KTERÁ NÁSLEDUJE, DATUM VZNIKU, AUTORA A TO CO KÓD PRO SPRÁVNOU
* FUNKCI VYŽADUJE
*/
```

ÚČEL A SMYSL KOMENTÁŘŮ VE ZDROJOVÉM KÓDU

- RYCHLEJŠÍ POCHOPENÍ KÓDU PRO SEBE PO ČASE, PRO DRUHÉ, CO BUDOU MÍT TU ČEST S VAŠÍM ZDROJOVÝM KÓDEM
- POZNÁMKY KE KÓDU: CO DODĚLAT, JAKÉ OBTÍŽE PŘINÁŠÍ TENTO KÓD, JINÉ MOŽNOSTI ŘEŠENÍ
- POPIS ZÁMĚRU, SMYSLU, ÚČELU KÓDU
- SOUHRNNÝ POHLED NA KÓD – SHRNUJÍCÍ VĚTŠÍ ČÁST KÓDU
- GENEROVÁNÍ DOKUMENTACE – NĚKTERÉ PROGRAMY JSOU Z KOMENTÁŘŮ KE KÓDU SCHOPNY VYTVOŘIT DOKUMENTACI K SOFTWARU. NAPŘ. V PHP TO JE phpDocumentor.

DVA OBECNÉ EXTRÉMNÍ PŘÍSTUPY KE KOMENTOVÁNÍ ZDROJOVÉHO KÓDU:

- ZDROJOVÝ KÓD MÁ BÝT SEBEVYSVĚTLUJÍCÍ=SELF-DOCUMENTING A KOMENTÁŘE NEJSOU POTŘEBA: CODING STANDARD
- VŠE SE MUSÍ KOMENTOVAT A BEZ KOMENTÁŘŮ NELZE POVAŽOVAT ZDROJOVÝ KÓD ZA HOTOVÝ

ŠPATNĚ PSANÝ KOMENTÁŘ:

- KOMENTÁŘ JEN OPISUJE TO, CO SE DÁ VYČÍST Z KÓDU
- PŘEČTENÍ A POCHOPENÍ KOMENTÁŘE JE OBTÍŽNĚJŠÍ NEŽ PŘEČTENÍ A POCHOPENÍ KÓDU

DRUHY DOBŘE PSANÝCH KOMENTÁŘŮ = KOMENTÁŘŮ MAJÍCÍCH SMYSL

- **ZÁSTUPNÉ KOMENTÁŘE:** POZNÁMKY PRO NÁS, CO DODĚLAT, DOČASNÁ ŘEŠENÍ, POCHYBNOSTI, MOŽNOSTI – TODO, …
- **SHRNUJÍCÍ KOMENTÁŘE:** KOMENTÁŘE, KTERÉ JEDNODUŠE A PŘEHLEDNĚ SHRNOU VĚTŠÍ ČÁST KÓDU. DĚLAJÍ TO RYCHLEJI A LÉPE NEŽ KÓD
- **KOMENTÁŘE VYSVĚTLUJÍCÍ ZÁMĚR:** VYSVĚTLUJÍ NIKOLI, JAK KÓD FUNGUJE, ALE PROČ TAM KÓD JE
- **KOMENTÁŘE PRO AUTOMATICKÉ GENEROVÁNÍ** DOKUMENTACE – SPOJENÍ VYTVÁŘENÍ DOKUMENTACE Z JIŽ NAPSANÝCH KOMENTÁŘŮ

PROBLÉM SPOJENÝ S KOMENTÁŘI: PO ÚPRAVĚ ZDROJOVÉHO KÓDU SE MŮŽE STÁT, ŽE KOMENTÁŘ ZASTARÁ A NEPOPISUJE TO CO BY MĚL (U KOMENTÁŘŮ POPISUJÍCÍCH ZÁMĚR SE TO ČASTO NESTÁVÁ)

PŘÍKLADY KOMENTOVÁNÍ KÓDU:

KOMENTÁŘ SOUBORU / PROGRAMU Z is.mendelu.cz

```
/**
* @brief Toto je strucny popis programu
*
* Toto se v dokumentaci ukaze jako podrobny popis .
* Zejmena se soustredte na rozdeleni do souboru ...
*
* @author David Prochazka
* @author Mr. Hyde
* @version 1.1
* @date 2011 -08 -11
* @pre Nejsou zadne zvlastni pozadavky na incicalizaci .
* @bug Je nutne vyresit krizovy odkaz mezi technikem a fakturou .
* @warning Berte pouze jako cvicebni priklad
* @copyright GNU GLP v. 3
*/
```

KOMENTÁŘ TŘÍDY A ATRIBUTU Z is.mendelu.cz

```
/**
* @brief Kratky popis tridy
*
* Podrobny popis tridy
* ozdobeny hvezdickami
*/
class Zakaznik {
private :
/// Komentar pred deklaraci jen na radku
/// nebo dve
string m_jmeno ;
int m_vek ; /// < komentar az za deklaraci
public :
```

KOMENTÁŘ METODY (POPISUJE I PARAMETRY A NÁVRATOVÉ HODNOTY METODY) Z is.mendelu.cz

```
/**
* @brief Metoda resici zpracovani zakazky .
*
* @param zakazka je ukazatel na existujici ...
* @param cenaOpravy je cena za opravu ...
* @return vraci novou instanci tridy # Faktura ...
*
* Metoda by v praxi mela provadet vypocet ...
*
* @bug Slabina je v moznem nenaplneni parametru \a zakazka .
*/
Faktura * zpracujZakazku ( Zakazka * zakazka , float cenaOpravy );
```

AUTOMATICKÉ SESTAVENÍ DOKUMENTACE Z KOMENTÁŘŮ VE ZDROJOVÉM KÓDU: DOXYGEN (C, PHP, Java, C#) GENEROVÁNÍ ONLINE – HTML NEBO OFFLINE DOKUMENTACE – LATEX, PDF, RTF (OBDOBA DOXYGEN: Javadoc, PHPDocumentor)
TERMÍNY: KOMENTÁŘE, JEDNOŘÁDKOVÉ, VÍCEŘÁDKOVÉ, ZDROJOVÝ KÓD, SPECIÁLNÍ ZNAČKY: @param, @return, @author
PRAKTICKÁ DOVEDNOST: NAPSAT KOMENTÁŘ K METODĚ, KE TŘÍDĚ, NAPSAT RŮZNÉ DRUHY KOMENTÁŘŮ V C A C++
DOPORUČENÉ ZDROJE: MIKE GUNDERLOY: Z KODÉRA VÝVOJÁŘEM, http://d3s.mff.cuni.cz/teaching/programming_practices/lectures/index-l10.html, http://homel.vsb.cz/~s1a10/educ/ZP/prednasky/11-zdroj-kod-dokumentace.pdf, https://users.ece.cmu.edu/~eno/coding/CCodingStandard.html#documentation, http://www.stack.nl/~dimitri/doxygen/, http://while1.wz.cz/t_komentar.php, https://is.mendelu.cz/eknihovna/opory/318/Knihovna%20k%20projektu/Z%E1klady%20objektov%EC%20orientovan%E9ho%20n%E1vrhu/P%F8edn%E1%B9ky%20v%20PDF/ZOO_06_velke_projekty.pdf,

21. Ladění programu a programátorské chyby

LADĚNÍ JE PROCES HLEDÁNÍ, POCHOPENÍ TOHO JAK VZNIKAJÍ A OPRAVA CHYB.
CHYBY VZNIKAJÍ PŘIROZENĚ V KAŽDÉM PROGRAMU. CHYBA OZNAČOVANÁ **ANGLICKY BUG**.
DRUHY CHYB:

- **VNĚJŠÍ CHYBY** – CHYBY VZNIKAJÍCÍ MIMO PROGRAM – HW A SW CHYBY, NA KTERÉ PROGRAMÁTOR NEMÁ VLIV (MÁLO PAMĚTI, CHYBA VE SPOLUPRACUJÍCÍM PROGRAMU, V OPERAČNÍM SYSTÉMU, …) NĚKDY LZE OŠETŘIT VÝJIMKAMI
- **SYNTAKTICKÉ CHYBY** – CHYBY ZPŮSOBENÉ TÍM, ŽE **ZDROJVOVÝ KÓD NEBYL ZAPSÁN PODLE PRAVIDEL (GRAMATIKY)** PROGRAMOVACÍHO JAZYKA. VĚTŠINOU JE **ODHALÍ PŘEKLADAČ, INTERPRET NEBO EDITOR**. TYPICKOU SYNTAKTICKOU CHYBOU JE OPOMENUTÍ STŘEDNÍKU NA KONCI PŘÍKAZU V JAZYCE C.
- **SÉMANTICKÉ CHYBY** – JSOU CHYBY, KTERÉ **ZPŮSOBUJÍ CHYBNOU FUNKCI PROGRAMU**. SÉMANTIKA ZNAMENÁ VÝZNAM. NAPŘÍKLAD SE PROGRAM ZACYKLÍ, NEBO ZAMĚNÍME DVĚ PROMĚNNÉ. **V PRŮBĚHU LADĚNÍ VYHLEDÁVÁME PRÁVĚ TYTO DRUHY CHYB**.

JE DOBRÉ LADIT A HLEDAT CHYBY VŽDY PO NAPSÁNÍ KRATŠÍHO KÓDU, ABY NEBYLO NAJEDNOU AŽ PŘÍLIŠ MNOHO CHYB.
LADĚNÍ PROGRAMU PROBÍHÁ **V PROGRAMU, KTERÉMU SE ŘÍKÁ DEBUGGER** – DALO BY SE VOLNĚ PŘELOŽIT JAKO ODVŠIVOVAČ.
DEBUGGER JE VĚTŠINOU SOUČÁSTÍ VÝVOJOVÉHO PROSTŘEDÍ VYŠŠÍCH PROGRAMOVACÍCH JAZYKŮ.
ZÁKLADNÍ FUNKCE DEBUGGERU JSOU:

- **KROKOVAT PROGRAM** (PO VYKONÁNÍ JEDNOHO ŘÁDKU PROGRAMU, SE PROGRAM ZASTAVÍ A ČEKÁ NA DALŠÍ KROK, KTERÝ BUDE PŘEDSTAVOVAT VYKONÁNÍ DALŠÍ, NÁSLEDUJÍCÍ ŘÁDKY PROGRAMU) S KROKOVÁNÍM SE LZE I VNOŘOVAT DO EXITUJÍCÍCH FUNKCÍ A PROCHÁZET ČINNOST PROGRAMU UVNITŘ NICH.
- **ZASTAVIT PROGRAM V PŘEDEM DEFINOVANÉM MÍSTĚ PROGRAMU =** NA KONKRÉTNÍ ŘÁDCE PROGRAMU – UMÍSTÍME ZDE **BREAKPOINT**, LZE PAK VŽDY DOBĚHNOUT DO DALŠÍHO BREAKPOINTU, POKUD UŽ NECHCEME PROGRAM KROKOVAT
- **ZOBRAZIT AKTUÁLNÍ STAV JEDNOTLIVÝCH PROMĚNNÝCH PROGRAMU** V OKAMŽIKU JEHO ZASTAVENÍ – **WATCHLIST PŘEDSTAVUJE SEZNAM ZOBRAZOVANÝCH PROMĚNNÝCH**
- **PROHLÍŽENÍ STAVU PAMĚTI, REGISTRŮ, ZÁSOBNÍKU** V OKAMŽIKU ZASTAVENÍ PROGRAMU

VŠECHNY TYTO FUNKČNOSTI, **UMOŽNÍ PROGRAMÁTOROVI ODHALIT**, **CO SE V PRŮBĚHU PROGRAMU SKUTEČNĚ DĚJE** A TAK POCHOPÍ, **KDE VZNIKÁ PŘÍPADNÁ CHYBA VE VÝSLEDKU**. POCHOPÍ, **PROČ PROGRAM NEVRACÍ TO, CO BYCHOM OČEKÁVALI**.
DEBUGGER SPOUŠTÍ PŘELOŽENÝ BINÁRNÍ KÓD A ZOBRAZUJE K NĚMU ODPOVÍDAJÍCÍ ZDROJOVÉ ŘÁDKY ZE ZDROJOVÉHO SOUBORU. PRO SPRÁVNOU FUNKČNOST DEBUGGERU, JE POTŘEBA PŘELOŽIT ZDROJOVÝ KÓD I **S LADICÍMI INFORMACEMI**, KTERÉ UMOŽNÍ PROPOJIT BINÁRNÍ KÓD S KONKRÉTNÍ ŘÁDKOU ZDROJOVÉHO KÓDU.
ČASTO POUŽÍVANÝ DEBUGGER: **GDB – GNU DEBUGGER**. JE TO PROGRAM OVLÁDANÝ Z PŘÍKAZOVÉHO ŘÁDKU. FUNGUJE JAK NA UNIXU, TAK NA WINDOWS. PROGRAM, KTERÝ SE V TOMTO DEBUGGERU DEBUGGUJE MŮŽE BÝT NAPSÁN NAPŘÍKLAD V C, C++, PASCALU, ADĚ, OBJECTIVE-C, FOTRAN, MODULA-2.
DEBUGGER MŮŽE BÝT SOUČÁSTÍ VÝVOJOVÉHO PROSTŘEDÍ NEBO SAMOSTATNĚ PRACUJÍCÍ PROGRAM.
POSTUP LADĚNÍ PROGRAMU POMOCÍ DEBUGGERU:

1. NEJPRVE JE POTŘEBA **PROGRAM PŘELOŽIT S LADICÍMI INFORMACEMI** – VE VÝVOJOVÉM PROSTŘEDÍ SE JEDNÁ O NASTAVENÍ PŘEKLADAČE

2. DO ZDROJOVÉHO KÓDU **UMÍSTĚTE ZARÁŽKY – BREAKPOINTY**. MÍSTA, KDE SE VYKONÁVÁNÍ PROGRAMU ZASTAVÍ. OBVYKLE SE TO USKUTEČNÍ, KLIKNUTÍM MYŠI PŘED ŘÁDEK ZDROJOVÉHO TEXTU, NĚKDE U ČÍSLA ČÁDKY. POVĚTŠINĚ SE OBJEVÍ ČERVENÝ KROUŽEK NEBO OSMIÚHELNÍK PŘEDSTAVUJÍCÍ STOPKU. ZARÁŽKU MŮŽETE VLOŽIT I KONTEXTOVÝM MENU. NA TÉTO ZARÁŽCE DEBUGGER POZDĚJI ZASTAVÍ VYKONÁVÁNÍ PROGRAMU. PAK PŮJDE SLEDOVAT STAV PROGRAMU. TYPY ZARÁŽEK:

a. **PEVNÉ**: PROGRAM SE ZDE ZASTAVÍ VŽDY (NE V DLOUHÝCH CYKLECH, LADICÍ UDÁLOST MŮŽE NASTAT AŽ PO NĚKOLIKA TISÍCÍCH CYKLECH A KDO BY TO CHTĚL ODKLIKÁVAT)
b. **PODMÍNĚNÉ**: PROGRAM SE ZDE ZASTAVÍ JEN PŘI SPLNĚNÍ JISTÝCH PODMÍNEK, JINAK DEBUGGER ZARÁŽKU PŘESKOČÍ.

3. **SPUSŤTE DEBUGGOVÁNÍ PROGRAMU**, NĚKDY JE POTŘEBA SPUSTIT BINÁRNÍ SOUBOR S POMOCÍ DEBUGGERU. V INTEGROVANÝCH VÝVOJOVÝCH PROSTŘEDÍCH TO ZNAMENÁ V MENU LADĚNÍ ČI SPUSTIT KLIKNOUT NA PŘÍKAZ DEBUGG.

4. **PROGRAM SE ZASTAVÍ NA PRVNÍ ZARÁŽCE – BREAKPOINTU**. OD TOHOTO ŘÁDKU MŮŽEME ZAČÍT **PROGRAM KROKOVAT**. TOTO JSOU RŮZNÉ MÓDY KROKOVÁNÍ PROGRAMU.

a. **DALŠÍ KROK (STEP)** – TO JE ZÁKLADNÍ OPERACE, KTEROU UMÍ VŠECHNY DEBUGGERY. VYKONÁ JEDEN ŘÁDEK PROGRAMU A ZASTAVÍ SE. POKUD JE NA TOMTO ŘÁDKU VOLÁNÍ FUNKCE, TAK JI VYKONÁ JAKO JEDEN PŘÍKAZ.
b. **KROK DOVNITŘ (STEP INSIDE)** – OPĚT ZÁKLADNÍ OPERACE VŠECH DEBUGGERŮ. VYKONÁ JEDEN ŘÁDEK PROGRAMU A ZASTAVÍ SE. POKUD JE NA DANÉM ŘÁDKU VOLÁNÍ FUNKCE, PŘEJDE SE NA PRVNÍ ŘÁDEK TĚLA TÉTO FUNKCE.
c. **POKRAČUJ (CONTINUE)** - POKRAČUJE VE VYKONÁVÁNÍ PROGRAMU, DOKUD NENARAZÍ NA ZARÁŽKU U KONCE PROGRAMU.
d. **POKRAČUJ DO VYBRANÉHO ŘÁDKU** - PROGRAM POKRAČUJE, AŽ DO ŘÁDKU NA KTERÉM JE KURZOR.
e. **POKRAČUJ DO KONCE AKTUÁLNÍ FUNKCE** - POKRAČUJE VE VYKONÁVÁNÍ AŽ DO KONCE FUNKCE KDE SE ZARAZÍ.

5.PŘI ZASTAVENÍ PROGRAMU **LZE SLEDOVAT**: V OKNU WATCH – WATCHES: **PROMĚNNÉ, PARAMETRY FUNKCÍ, VÝRAZY; PAMĚŤ – MEMORY, MEMORY DUMP** (JAK SE DATA UKLÁDAJÍ DO PAMĚTI – SLEDOVÁNÍ PŘEKRÝVÁNÍ POLÍ); **REGISTRY** V OKNĚ CPU REGISTERS, REGISTRY; **STAV ZÁSOBNÍKU – STACK, CALL STACK** – ZOBRAZÍ SE FUNKCE V POŘADÍ, JAK BYLY VOLÁNY
6.MŮŽETE MĚNIT HODNOTY PROMĚNNÝCH V OKNĚ, KDE SE SLEDUJÍ A DÁL PROGRAM PRACUJE S NIMI. NĚKDY **LZE EDITOVAT A OPRAVOVAT I PROGRAM SÁM**. JE PAK ALE POTŘEBA PROGRAM ZNOVU PŘELOŽIT.
PRO LADĚNÍ PROGRAMU JE KLÍČOVÉ ZNÁT ČÍSLA ŘÁDKŮ. BEZ JEJICH ČÍSEL TO VYPADÁ, JAKO BYCHOM SE POHYBOVALI PO NEZNÁMÉM TERÉNU BEZ MAPY.
TERMÍNY: BUG, DEBUGGER, KROKOVÁNÍ, DEBUGGOVÁNÍ, LADĚNÍ, BREAKPOINT, VNĚJŠÍ, SYNTAKTICKÉ, SÉMANTICKÉ CHYBY
PRAKTICKÁ DOVEDNOST: UVÉST PŘÍKLADY KONKRÉTNÍCH CHYB, URČENÍ O JAKÝ TYP CHYBY SE JEDNÁ A JEJICH ODHALENÍ.
DOPORUČENÉ ZDROJE: http://www.fit.vutbr.cz/~martinek/clang/debug.html, http://jaknaprojekty.davidm.cz/ladeni.html#programatorske-chyby, http://jaknaprojekty.davidm.cz/ladeni.html, http://www.gnu.org/software/gdb/, http://vyuka.pecinovsky.cz/animace/ladeni/index.html,

22. Integrované vývojové prostředí, vlastnosti, překladače, debuggery

INTEGROVANÉ VÝVOJOVÉ PROSTŘEDÍ – IDE JE PROGRAM, KTERÝ OBSAHUJE KOMPLEXNI VLASTNOSTI PRO VÝVOJ PROGRAMŮ.

CO IDE MŮŽE OBSAHOVAT:

- **EDITOR ZDROJOVÉHO KÓDU**
- **KOMPILER NEBO INTERPRET**
- **DEBUGGER**
- **MANAŽER SOUBORŮ, PROJEKTŮ**
- **PROHLÍŽEČ OBJEKTŮ - TŘÍD**
- **EDITOR GRAFICKÉHO UŽIVATELSKÉHO PROSTŘEDÍ - GUI, EDITOR FORMULÁŘŮ**

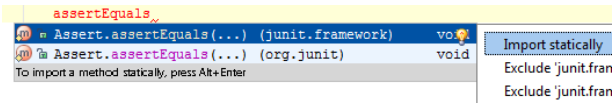
S ČÍM MŮŽE VÝVOJOVÉ PROSTŘEDÍ JEŠTĚ SPOLUPRACOVAT

- **INTEGROVANÉ SYSTÉMY PRO SPRÁVU VERZÍ** JAKO JE SUBVERSION, GIT, BAZZAR, MERCURIAL
- **NÁSTROJE PRO TESTOVÁNÍ KÓDU**
- **SOFTWARE PRO SPRÁVU DATABÁZÍ**
- **NÁSTROJE PRO DEPLOYING NA FTP SERVERY**
- **SYSTÉM PRO TÝMOVOU SPOLUPRÁCI, ROZDĚLOVÁNÍ ÚKOLŮ, PROJEKTOVÝ MANAGEMENT**
- **NÁSTROJE PRO MODELOVÁNÍ TŘÍD V UML**
- **NÁSTROJE PRO POROVNÁVÁNÍ ZMĚN V KÓDU A SLUČOVÁNÍ DVOU SOUBORŮ** – MERGING, MERGE

MNOHO VÝVOJOVÝCH PROSTŘEDÍ JE MOŽNO VYUŽÍT PRO **PROGRAMOVÁNÍ V NĚKOLIKERÝCH PROGRAMOVACÍCH JAZYCÍCH**. PŘÍKLADEM TAKOVÝCH VÝVOJOVÝCH PROSTŘEDÍ JE **NAPŘÍKLAD ECLIPSE, NETBEANS**, KOMODO, APTANA, GEANY.
PRO SNADNÝ VÝVOJ KDEKOLIV NA SVĚTĚ BEZ INSTALOVANÝCH SPECIFICKÝCH NÁSTROJŮ SE ZAČÍNAJÍ VÍCE A VÍCE VYUŽIVAT **CLOUDOVÁ NEBO WEBOVÁ VÝVOJOVÁ PROSTŘEDÍ: CODEANYWHERE**, CODERUN, SHIFTEDIT.
IDE PRO JAZYK C A C++: CODE::BLOCKS, CODELITE, **Microsoft Visual Studio Express**, **Bloodshed Dev-C++**
IDE PRO JAZYK JAVA: JIKES, INTELLIJ IDEA, **NETBEANS, ECLIPSE**

NĚKTERÉ UŽITEČNÉ VLASTNOSTI INTEGROVANÉHO VÝVOJOVÉHO PROSTŘEDÍ

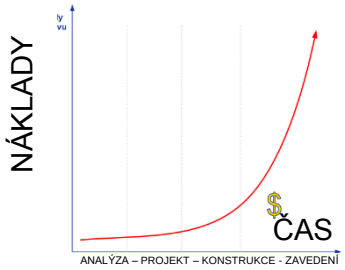
- **NASTAVITELNÉ PROSTŘEDÍ** PRO RŮZNÉ ČINNOSTI – RŮZNÉ TOOLBOXY, OKNA
- **UZAMYKÁNÍ SOUBORŮ PŘED ZMĚNAMI – READ ONLY**
- **ŠABLONY A SNIPPETY** (MOŽNOST VKLÁDAT ČASTO POUŽÍVANÉ KRÁTKÉ ČÁSTI KÓDU, NEBO VZORY)
- **ČÍSLA ŘÁDEK** ZDROJOVÉHO KÓDU
- **CODE-FOLDING** – LOGICKÉ BLOKY ZDROJOVÉHO KÓDU MOŽNÉ SKRÝT POD ROZKLIKÁVACÍ KŘÍŽEK. ČÁST KÓDU SE SBALÍ
- **RYCHLÁ LOKACE DEFINIC A REFERENCÍ**, VOLÁNÍ METOD A PROMĚNNÝCH, **JUMP-TO-DEFINITION**.
- **ZVÝRAZNĚNÍ SYNTAXE – SYNTAX HIGHLIHTING**: ROZPOZNÁ A ZVÝRAZNÍ KLÍČOVÁ SLOVA, ŘETĚZCE, PROMĚNNÉ, …
- **ZVÝRAZNĚNÍ ZAČÁTKU A KONCE BLOKŮ PROGRAMU – HIGHLIGHTING BRACES**
- **ZVÝRAZNĚNÍ POZMĚNĚNÝCH ŘÁDEK**
- **INTELIGENTNÍ NAŠEPTÁVAČ – INTELIGENT CODE COMPLETION, INTELLISENSE, AUTOCOMPLETE**: ZATÍMCO UŽIVATEL PÍŠE TEXT, NABÍDNE SE MU SLOVO NEBO SLOVA, KTERÝ TEXT DOKONČÍ. NAPŘÍKLAD U IDENTIFIKÁTORU PROMĚNNÝCH, METOD NEBO KLÍČOVÝCH SLOV. STAČÍ PAK VYBRAT NAVRŽENÝ TERMÍN A ODKLIKNOUT – NAVRŽENÁ SLOVA SE ZAPÍŠÍ DO KÓDU.



- **KONTROLA PRAVOPISU – SPELLCHECKING**
- **AUTOMATICKÁ KONTROLA SYNTAXE**, OZNAČENÍ CHYB A NĚKDY I JEJICH OPRAVA
- **AUTOMATICKÉ UKONČOVÁNÍ BLOKŮ KÓDU** – ZAČÁTKY A KONCE FUNKCÍ, ŘÍDÍCÍCH KONSTRUKCÍ – ZÁVORKAMI NEBO JINAK
- **AUTOMATICKÉ FORMÁTOVÁNÍ KÓDU, CODE-RESTYLING**, AUTOMATICKÁ DETEKCE ODSAZENÍ A MÓDU KONCE ŘÁDKY, AUTOMATICKÉ ODSAZOVÁNÍ (CHANGE INDENTATION) – PODLE NASTAVENÍ **CODE STANDARD**.
- **REFAKTOROVÁNÍ – REFACTORING:** PROVÁDĚNÍ ZMĚN V PROGRAMU, KTERÉ NEMĚNÍ FUNKCIONALITU, CHOVÁNÍ, ALE MĚNÍ STRUKTURU. MĚL BY VZNIKNOUT ČISTŠÍ A PŘEHLEDNĚJŠÍ KÓD, KTERÝ SE LÉPE ROZŠIŘUJE A UDRŽUJE. NAPŘÍKLAD PŘESOUVÁNÍ PRVKŮ MEZI OBJEKTY. MNOHO DRUHŮ REFAKTORINGU.
- **MULTIPLE CURSORS** – NASTAVIT VÍCE KURZORŮ A PAK PSÁT STEJNÉ NA VÍCE MÍSTECH.

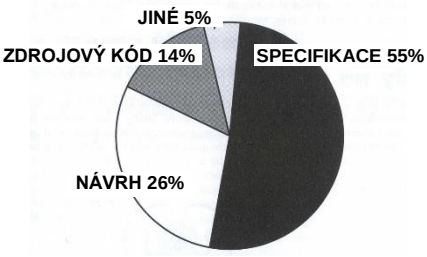
JEDNO Z NEJPROPRACOVANĚJŠÍCH IDE SOUČASNOSTI: https://www.jetbrains.com/ **IntelliJ IDEA**
PODLE ČEHO SI VYBRAT VHODNÝ IDE: 1. PLATFORMA, 2. MOJE ZNALOST PROGRAMOVACÍHO JAZYKA, 3. VYUŽITELNÉ KNIHOVNY, PRO REALIZACE PROJEKTU, 4. SNADNOST POUŽITÍ (DEBUGGER - ZDA EXISTUJE A CO UMÍ, VLASTNOSTI IDE), 5. PODLE VELIKOSTI PROJEKTU - PODPORA PRÁCE V TÝMU, 6. PŘIZPŮSOBENÍ SE REÁLNÝM MOŽNOSTEM (CENA, NÁROČNOST NA HARDWARE)
TERMÍNY: IDE, KOMPILER, DEBUGGER, GUI, SYSTÉMY PRO SPRÁVU VERZÍ, SOUČÁSTI IDE, ZVÝRAZNĚNÍ SYNTAXE, INTELLISENSE, CODE-RESTYLING – CODE STANDARD
PRAKTICKÁ DOVEDNOST: KTERÉ VLASTNOSTI JSOU U IDE DŮLEŽITÉ, JAK VYBRAT VHODNÝ IDE
DOPORUČENÉ ZDROJE: http://documentation.basis.com/BASISHelp/WebHelp/ide2/ide_features_basis.htm, http://programmers.stackexchange.com/questions/102018/what-features-of-an-ide-would-make-it-more-useful-than-a-general-purpose-editor, https://blog.profitbricks.com/top-integrated-developer-environments-ides/, https://www.jetbrains.com/help/idea/15.0/intellij-idea-editor.html?origin=old_help, http://refactoring.com/

23. Testování software



PROČ SE TESTUJE? ABY SE ZAJISTILA KVALITA. KVALITA JE SCHOPNOST OBJEKTU, PROGRAMU BÝT POUŽIT TAK JAK JE ZAMÝŠLENO A S CO NEJLEPŠÍMI VÝSLEDKY.
TESTOVÁNÍ SOFTWARU ODSTRAŇUJE CHYBY, KTERÉ SNIŽUJÍ KVALITU SOFTWARU. CHYBA MŮŽE BÝT NĚCO CO: 1. SOFTWARE NEDĚLÁ A PODLE SPECIFIKACE BY MĚL, **2.** SOFTWARE DĚLÁ NĚCO, CO BY PODLE SPECIFIKACE DĚLAT NEMĚL, **3.** SOFTWARE DĚLÁ NĚCO, CO SPECIFIKACE NEZMIŇUJE, **4.** SOFTWARE DĚLÁ NĚCO, CO SPECIFIKACE NEZMIŇUJE, ALE ZMIŇOVAT MĚLA, 5. SOFTWARE JE OBTÍŽNĚ SROZUMITELNÝ, POMALÝ, TĚŽKO SE S NÍM PRACUJE
ČÍM DŘÍVE JE CHYBA ODHALENA, TÍM NIŽŠÍ JSOU NÁKLADFY NA JEJÍ ODSTRANĚNÍ.
JE PROTO POTŘEBA **TESTOVAT POKUD MOŽNO VE VŠECH STUPNÍCH VÝVOJE**. ABY SE CHYBY ODHALILY CO MOŽNÁ NEJDŘÍVE. CÍLEM JE TAKÉ **CO NEJRYCHLEJI SNÍŽIT MNOŽSTVÍ CHYB**, KTERÉ EXISTUJÍ A ZNAČNĚ ZNESNADŇUJÍ I SAMOTNÉ TESTOVÁNÍ.
TEST JE SEZNAM KROKŮ. MUSÍ BÝT OPAKOVATELNÝ. OBSAHUJE OVĚŘOVACÍ KROKY, CO SE DAJÍ JEDNOZNAČNĚ VYHODNOTIT.

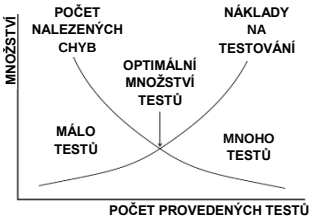
KDO MŮŽE ZA CHYBY?



TECHNIKY TESTOVÁNÍ PODLE ZPŮSOBU PROVEDENÍ TESTŮ

- **ČERNÁ A BÍLÁ SKŘÍŇKA:**
  - **TESTOVÁNÍ ČERNÉ SKŘÍŇKY** – BLACK-BOX TEST – PROVOZNÍ TESTOVÁNÍ: TESTER NEZNÁ VNITŘNÍ LOGIKU TESTOVANÉHO SOFTWARU, NENÍ K DISPOZICI ZDROJOVÝ KÓD, NEVÍME, JAK SYSTÉM PRACUJE S DATY. SLEDUJEME VSTUPNÍ A VÝSTUPNÍ DATA A VYHODNOSUJEME ZDA PŘI VHODNÝCH VSTUPNÍCH DATECH ZÍSKÁME VHODNÁ VÝSTUPNÍ.
  - **TESTOVÁNÍ BÍLÉ SKŘÍŇKY** – WHITE-BOX TEST – STRUKTURÁLNÍ TESTOVÁNÍ: VYCHÁZÍ ZE ZNALOSTI VNITŘNÍ LOGIKY SYSTÉMU A ZDROJOVÝCH KÓDŮ. NETESTUJÍ SE JEN VÝSTUPY PŘI DANÝCH VSTUPECH. TESTUJE SE I TO JAK VÝSTUPY ZÍSKÁ. NAPŘÍKLAD TO, JESTLI SE NĚKTERÉ OPERACE ZBYTEČNĚ NEOPAKUJÍ A ZDA KÓD NEOBSAHUJE SKRYTÉ CHYBY. **VÝHODY:** LÉPE SE DAJÍ VYTIPOVAT A NAPLÁNOVAT TESTOVÉ PŘÍPADY, **NEVÝHODY:** ZNALOST LOGIKY SYSTÉMU NĚKDY VEDE KE ZPŮSOBU TESTOVÁNÍ, NESOUVISÍCÍ S CHOVÁNÍM UŽIVATELE.
  - **TESTOVÁNÍ ŠEDÉ SKŘÍŇKY** – KOMBINACE OBOU ZMÍNĚNÝCH. VYUŽITÍ VÝHOD OBOU.
- **MANUÁLNÍ** (TESTUJE ČLOVĚK – VHODNÉ PRO SLOŽITOU LOGIKU SYSTÉMU) A **AUTOMATICKÉ TESTOVÁNÍ** (TESTUJE ZA NÁS SOFTWARE – PRO JEDNODUCHÉ, DOBŘE ALGORITMIZOVANÉ OPERACE, TESTOVÁNÍ MNOHA JEDNODUCHÝCH POŽADAVKŮ)
- **STATICKÉ** (NEVYŽADUJÍ BĚH SOFTWARU – ČASTO V POČÁTEČNÍCH FÁZÍCH VÝVOJE. I PŘED ZAČÁTKEM PSANÍ KÓDU – KONTROLA SPECIFIKACE, ANALÝZA KÓDU, SYNTAXE, ALGORITMŮ. I KOMPILACE PROGRAMU KONTROLUJE STATICKY.)
  A DYNAMICKÉ TESTOVÁNÍ (TESTOVÁNÍ PROGRAMU ZA BĚHU – JAKO BLACK NEBO WHITE BOX, MANUÁLNÍ NEBO AUTOMATICKÉ. JE POTŘEBA MÍT SPUSTITELNÝ SOFTWARE. VYUŽITÍ V POZDĚJŠÍCH FÁZÍCH VÝVOJE. ZAMĚŘENY NA PROVOZ.)
- **TESTY SPLNĚNÍM A SELHÁNÍM** – (OBA DVA DRUHY SE PŘI TESTOVÁNÍ MOHOU PROLÍNAT): **TESTY SPLNĚNÍM – TEST-TO-PASS** – POZITIVNÍ TESTY: TESTOVÁNÍ ZÁKLADNÍ FUNKČNOSTI, NIKOLI CO PROGRAM VYDRŽÍ, ALE JESTLI ZVLÁDNE JEN TO NEJZÁKLADNĚJŠÍ VŮBEC VYKONAT. TESTY FUNKCÍ, KTERÉ JSOU V APLIKACI IMPLEMENTOVÁNY A PŘEDSTAVUJÍ ZÁKLADNÍ FUNKČNOST SYSTÉMU. TEST ZDA FUNGUJÍ SPRÁVNĚ; **TESTY SELHÁNÍM – TEST-TO-FAIL** – NEGATIVNÍ TESTY: PROVÁDÍ SE AŽ TEHDY, KDYŽ JE ZABEZPEČENA ZÁKLADNÍ FUNKČNOST SYSTÉMU. PROVÁDÍ SE V ZÁVĚREČNÉ FÁZI TESTOVÁNÍ. VYVÝJÍ SE NA SYSTÉM EXTRÉMNÍ ZÁTĚŽ. TESTUJE SE STABILITA SYSTÉMU, MEZNÍ HODNOTY VSTUPNÍCH DAT ATD. JAKOBY JSME SE SNAŽILI SYSTÉM SCHODIT. HLEDÁME MAXIMÁLNÍ MNOŽSTVÍ CHYB, KTERÉ V SYSTÉMU JEŠTĚ JE.
- **FUNKČNÍ A NEFUNKČNÍ TESTY: FUNKČNÍ** TESTUJÍ FUNKCE, PRO KTERÉ JE APLIKACE PŘÍMO URČENA – VŠECHNY FUNKCE IMPLEMENTOVANÉ V APLIKACI – ZDA JSOU OBSAŽENY A ZDA FUNGUJÍ. **NEFUNKČNÍ** TESTY TESTUJÍ VŠECHNY VLASTNOSTI APLIKACE, KTERÉ PŘÍMO NESOUVISÍ S JEJÍMI FUNKCEMI, ALE JSOU TAKÉ DŮLEŽITÉ NAPŘ: VÝKONOVÉ TESTOVÁNÍ – PERFORMANCE TESTING, TESTOVÁNÍ JAK ZATĚŽUJE HARDWARE.

TECHNIKY TESTOVÁNÍ PODLE ÚROVNĚ VÝVOJE:



- **TESTOVÁNÍ V PRŮBĚHU VÝVOJE – DEVELOPER TESTING** – TEST PŘI PSANÍ KÓDU, NE TESTER
- **JEDNOTKOVÉ TESTY** – **UNIT TESTING** – TESTOVÁNÍ NEJMENŠÍCH TESTOVATELNÝCH ČÁSTÍ SYSTÉMU. PROVÁDÍ PROGRAMÁTOR PO IMPLEMENTACI MODULU.
- **INTEGRAČNÍ TESTY** – **INTEGRATION TESTING** – OVĚŘUJÍ SPOLEČNOU FUNKČNOST ČÁSTÍ SYSTÉMU. ZDA SPOLUPRACUJÍ A DOBŘE KOMUNIKUJÍ. TESTUJE SE HLAVNĚ ROZHRANÍ KOMPONENT A NEDOKONČENÉ ČÁSTI.
- **SMOKE TESTY** – ZDA JSOU VŠECHNY ČÁSTI APLIKACE IMPLEMENTOVÁNY, NAINSTALOVÁNY A SPUŠTĚNY – ZÁKLADNÍ FUNKCE.
- **SYSTÉMOVÉ TESTY** – **SYSTEM TESTING**: SYSTÉM SE TESTUJE JAKO CELEK, PO ZAVEDENÍ HLAVNÍCH ČÁSTÍ SYSTÉMU.
- **AKCEPTAČNÍ TESTY – ACCEPTANCE TESTING**: FINÁLNÍ TESTY PŘED DISTRIBUCÍ ZÁKAZNÍKŮM. MŮŽE SE UŽ POUŽÍVAT? – **FORMÁLNÍ:** PŘESNĚ NAPLÁNOVÁN, ZDOKUMENTOVÁN, INTERNÍ TESTEŘI, **NEFORMÁLNÍ:** NENÍ PŘESNĚ NAPLÁNOVÁNA – TESTER SI ZVOLÍ CO CHCE TESTOVAT, PROVÁDÍ INTERNÍ TESTEŘI, **BETA:** TESTUJÍ NEZÁVISLÉ OSOBY Z ŘAD UŽIVATELŮ.

DRUHY TESTŮ KVALITY PODLE RUP (RATIONAL UNIFIED PROCESS) – DIMENZE KVALITY: FUNKČNOST (FUNKCE, **TEST BEZPEČNOSTI**, KAPACITA), **POUŽITELNOST** (**TEST POUŽITELNOSTI**, UŽIVATELSKÉ ROZHRANÍ, UŽIVATELSKÁ DOKUMENTACE), **SPOLEHLIVOST** (NÁCHYLNOST K CHYBÁM, PŘESNOST, ZOTAVITELNOST SYSTÉMU), **VÝKON** (**ZÁTĚŽOVÝ TEST**, RYCHLOST, DOSTUPNOST, ODEZVA, DOBA ZDROJŮ, POUŽÍVÁNÍ ZDROJŮ), **PODPOROVATELNOST** (**TEST KOMPATIBILITY**, TESTOVATELNOST, ROZŠIŘITELNOSTI, PŘIZPŮSOBIVOST, SPRAVOVATELNOST, KONFIGUROVATELNOST, LOKALIZOVATELNOST),
VERZE: PRE-ALFA – PŘED TESTY, **ALFA** – TESTY TESTERŮ, **BETA** – TESTY UŽIVATELŮ, **RELEASE CANDIDATE** – BETA MUZE BYT FINALNI PRODUKT
TERMÍNY: TEST, TESTOVÁNÍ, CHYBA, KVALITA, ČERNÁ A BÍLÁ SKŘÍŇKA, AUTOMATICKÉ A MANULÁNÍ TESTOVÁNÍ, STATICKÉ A DYNAMICKÉ, SPLNĚNÍM A NESPLNĚNÍM, FUNKČNÍ A NEFUNKČNÍ, JEDNOTKOVÉ, INTEGRAČNÍ, SMOKE, SYSTÉMOVÉ TESTY…
DOPORUČENÉ ZDROJE: http://testovanisoftwaru.cz/category/metodika-testovani/, https://www.vse.cz/vskp/id/46, Testování Softwaru – Ron Patton, http://www1.osu.cz/~zacek/sweng/2013/07.pdf, http://nb.vse.cz/~buchalc/clanky/testovani.pdf, http://slideplayer.cz/slide/2347905/

24. Licence a autorský zákon

DUŠEVNÍ VLASTNICTVÍ: JE PRÁVO K NAKLÁDÁNÍ S NEHMOTNÝMI VÝSLEDKY PROCESU LIDSKÉ TVOŘIVOSTI, ZKOUMÁNÍ A MYŠLENÍ (DÍLA, VYNÁLEZY, OCHRANNÉ ZNÁMKY, VÝKONY UMĚLCŮ, DOBRÁ POVĚST, OBCHODNÍ TAJEMSTVÍ, KNOW-HOW, NÁVODY, ŘEŠENÍ, …).
ZA DUŠEVNÍ VLASTNICTVÍ LZE POVAŽOVAT TO, CO JE DOSTATEČNĚ JEDINEČNÉ, ORIGINÁLNÍ. HODNOTA DUŠEVNÍHO VLASTNICTVÍ ZÁVISÍ NA TOM, CO PŘINÁŠÍ PRO JEDINCE I SPOLEČNOST.

PRÁVA K DUŠEVNÍMU VLASTNICTVÍ SE V ČR DĚLÍ NA:

- **PRŮMYSLOVÉ VLASTNICTVÍ:** PATENTY, OCHRANNÉ ZNÁMKY, PRŮMYSLOVÉ VZORY
- **AUTORSKÉ PRÍVO A SOUVISEJÍCÍ PRÁVA**

AUTORSKÉ PRÁVO: PRÁVA AUTORŮ K JEJICH DÍLŮM. MÁ ABSOLUTNÍ POVAHU, TEDY PŮSOBÍ VŮČI VŠEM.
HISTORIE AUTORSKÉHO PRÁVA: POČÁTEK 15. STOLETÍ – KNIHTISK, OCHRANA NAKLADATELŮ, TISKAŘŮ; 1. AUTORSKÝ ZÁKON – ANGLIE 1609 – KRÁLOVNA ANNA, NA ÚZEMÍ ČECH, MORAVY A SLEZSKA – CÍSAŘSKÝ PATENT Č. 992 R 1846.

AUTORSKÉ PRÁVO MÁ 2 CÍLE:

- **CHRÁNIT INVESTICE TVŮRCŮ A PODPOROVAT JEJICH TVŮRČÍ ČINNOST**
- **PŘISPÍVAT K TOMU, ABY JEJICH TVORBA MOHLA BÝT PROSPĚŠNÁ SPOLEČNOSTI, ZAJISTIT ROZVOJ**

AUTORSKÉ PRÁVO URČUJE:

- **POSKYTOVÁNÍ VÝLUČNÝCH PRÁV AUTORŮ K JEJICH DÍLŮM + ŘADU VÝJIMEK A OMEZENÍ TĚCHTO PRÁV**

AUTOR: KTERÁKOLIV FYZICKÁ OSOBA, JEŽ DÍLO VYTVOŘILA, JEDNÁ SE O (AUTORA, SPOLUAUTORY, VÝKONNÉHO UMĚLCE, VÝROBCE ZÁZNAMU, VYSÍLATEL, NAKLADATEL, ZVEŘEJNITEL, POŘIZOVATEL DATABÁZE)
AUTORSKÉ DÍLO: JEDINEČNÝ VÝSLEDEK TVŮRČÍ ČINNOSTI AUTORA VYJÁDŘENÝ V JAKÉKOLI OBJEKTIVNĚ VNÍMATELNÉ PODOBĚ BEZ OHLEDU NA ROZSAH, ÚČEL NEBO VÝZNAM. DÍLO LITERÁRNÍ A JINÉ DÍLO UMĚLECKÉ A DÍLO VĚDECKÉ. **U SOFTWARU, DATABÁZÍ A FOTOGRAFIÍ SE VYŽADUJE PŮVODNOST. AUTORSKÉ DÍLO JE CHRÁNĚNO AŤ UŽ DOKONČENÉ, NEBO JEHO VÝVOJOVÉ FÁZE ČI ČÁSTI, VČETNĚ NÁZVU A JMEN POSTAV.**
AUTORSKÉ DÍLO NENÍ: NÁPAD, MYŠLENKA, METODA, PRINCIP, POSTUP, HOLÝ NÁMĚT, DENNÍ ZPRÁVA – ÚDAJ, OBJEV, VĚDECKÁ TEORIE, MATEMATICKÝ ČI OBDOBNÝ VZOREC, STATISTICKÝ GRAF A PODOBNÝ PŘEDMĚT SÁM O SOBĚ.
AUTORSKÉ DÍLO JE NAPŘ.: DÍLO SLOVESNÉ, HUDEBNÍ, DRAMATICKÉ, HUDEBNĚ DRAMATICKÉ, CHOREOGRAFICKÉ, PANTOMIMICKÉ, FOTOGRAFICKÉ, AUDIOVIZUÁLNÍ, VÝTVARNÉ, ARCHITEKTONICKÉ, KARTOGRAFICKÉ, SOUBORNÉ, UŽITNÉHO UMĚNÍ, VYSÍLÁNÍ, ZÁZNAM, **I NEJEDINĚČNÝ, ALE PŮVODNÍ: POČÍTAČOVÝ PROGRAM, DATABÁZE, FOTOGRAFIE**
VOLNÉ DÍLO: PO KONCI TRVÁNÍ AUTORSKÝCH PRÁV MAJETKOVÝCH. NAKLÁDAT BEZ SOUHLASU A PLACENÍ AUTORSKÉ ODMĚNY.
VZNIK PRÁV K AUTORSKÉMU DÍLU: PRÁVA VZNIKAJÍ OBJEKTIVNĚ SE VZNIKEM DÍLA, BEZ OHLEDU NA VŮLI AUTORA. PRO VZNIK NENÍ POTŘEBA ADMINISTRATIVNÍHO ÚKONU, ANI ZVEŘEJNĚNÍ. **AUTOR SE TĚCHTO PRÁV NEMŮŽE JEDNOSTRANNĚ VZDÁT**.
ZÁNIK PRÁV K AUTORSKÉMU DÍLU: VYPRŠENÍM PRÁV SE DÍLO STÁVÁ VOLNÝM. DOBA VYPRŠENÍ PRÁV JE RŮZNÁ.

- MAJETKOVÁ PRÁVA ZANIKAJÍ 70 LET PO SMRTI AUTORA, NEBO POSLEDNÍHO SPOLUAUTORA, KDYŽ JE JICH VÍC.

U AUDIOVIZUÁLNÍCH DĚL ZA 70 LET PO SMRTI POSLEDNÍHO Z REŽISÉRA, AUTORA SCÉNÁŘE, AUTORA DIALOGŮ A SKLADATELE HUDBY. (NAKLADATEL OD VYDÁNÍ, VÝROBCE ZVUKOVÉHO A ZVUKOVĚ OBRAZOVÉHO ZÁZNAMU OD POŘÍZENÍ ZÁZNAMU, ROZHLASOVÝ A TELEVIZNÍ VYSÍLATEL PO PRVNÍM VYSÍLÁNÍ 50 LET), PRÁVO POŘIZOVATELE DATABÁZE 15 LET, PRVNÍ ZVEŘEJNĚNÍ VOLNÉHO NEZVEŘEJNĚNÉHO DÍLA: 25 LET.

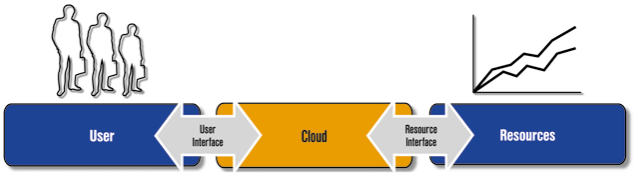
OBSAH AUTORSKÝCH PRÁV:

- **PRÁVA MAJETKOVÁ –** PRO KAŽDÉ UŽITÍ DÍLA JE POTŘEBA ZÍSKAT SOUHLAS AUTORA. JSOU NEPŘEVODITELNÁ = AUTOR SE JICH NEMŮŽE VZDÁT A JSOU SOUČÁSTÍ DĚDICTVÍ. AUTOR MŮŽE JEN NĚKOMU UDĚLIT OPRÁVNĚNÍ DÍLO UŽÍT.
- **PRÁVA OSOBNOSTNÍ –** VÁŽÍ SE K OSOBĚ AUTORA, ZANIKAJÍ JEHO SMRTÍ, ALE I PO NÍ NUTNO RESPEKTOVAT. **PRÁVO BÝT UVEDEN JAKO AUTOR** (POKUD ZAMĚSTNANCEM MUSÍ STRPĚT ZÁSAH DO TOHOTO PRÁVA) **A URČIT JAK BÝT JAKO AUTOR UVEDEN, PRÁVO NA TECHNICKOU A MORÁLNÍ INTEGRITU DÍLA** (ZDA LZE DOPLNIT, ZMĚNIT, VYUŽÍVÁT TAK, ABY NEBYLA SNÍŽENA JEHO VÁŽNOST), **PRÁVO DÍLO ZVEŘEJNIT**

AUTORSKÝ ZÁKON V ČR: V ÚČINNOSTI OD 1. 12. 2000. ZÁKON Č. 121/2000.
TŘÍSTUPŇOVÝ TEST: POUŽITÍ PŘI OMEZENÍ AUTORSKÉHO PRÁVA. KDYŽ PROJDE TENTO TEST, LZE AUTORSKÉ PRÁVO OMEZIT.
POČÍTAČOVÝ PROGRAM JE CHRÁNĚN JAKO DÍLO LITERÁRNÍ! UŽIVATEL SI MŮŽE ZHOTOVIT JEN ZÁLOŽNÍ KOPII.
DRUHY PRÁV DÍLO UŽÍT: ROZMNOŽOVÁNÍ, ROZŠIŘOVÁNÍ, PRONÁJEM, PŮJČOVÁNÍ, VYSTAVOVÁNÍ, SDĚLOVÁNÍ VEŘEJNOSTI.
TŘI FORMY UŽITÍ DÍLA: VOLNÉ DÍLO – (BEZ SVOLENÍ A BEZÚPLATNĚ), **ZÁKONNÁ LICENCE** – (NESMÍ JÍT O UŽITÍ ZA ÚČELEM HOSPODÁŘSKÉHO PROSPĚCHU, PRO ÚČEL VZDĚLÁNÍ, VÝZKUM, KRITIKU, KNIHOVNÍ LICENCE, ZPRAVODAJSKÁ, ÚŘEDNÍ LICENCE) A **LICENČNÍ SMLOUVA: AUTOR POSKYTUJE PRÁVO DÍLO UŽÍT. PRÁVO UŽITÍ MŮŽE BÝT VÝHRADNÍ (NEMŮŽE DÁL UŽÍT 3. OSOBA, NUTNÁ PÍSEMNÁ FORMA) NEBO NEVÝHRADNÍ (AUTOR MŮŽE POSKYTNOUT DALŠÍM OSOBÁM). NABYVATEL SE ZAVAZUJE POSKYTNOUT AUTOROVI ODMĚNU (POKUD NEUVÁDÍ JINAK). LICENCE MŮŽE BÝT OMEZENA NA JEDNOTLIVÉ ZPŮSOBY UŽITÍ.**
SOFTWAROVÁ LICENCE: FREEWARE – LZE ZDARMA POUŽÍVAT I ŠÍŘIT. NENÍ K DISPOZICI ZDROJOVÝ KÓD. NELZE UPRAVOVAT A VYTVÁŘET VLASTNÍ VERZE (CCLEANER, OPERA, 7-ZIP). **OPEN SOURCE** - LZE ZDARMA POUŽÍVAT I ŠÍŘIT. JE PŘÍSTUP KE ZDROJOVÝM KÓDŮM, LZE UPRAVOVAT A VYTVÁŘET NOVÉ VERZE (OPENOFFICE, FIREFOX, LINUX). **SHAREWARE** – VE ZKUŠEBNÍ VERZI S OMEZENÝMI FUNKCEMI, S REKLAMOU ČI ODSOUHLASENÍM (TOTAL COMMANDER). **TRIALWARE** – UŽÍVÁNÍ ČASOVĚ OMEZENÉ – PAK REGISTRACE ČI ZAKOUPIT (AVAST! HOME, WINRAR). **DEMO VERSION** – UKÁZKOVÁ VERZE, OMEZENÍ NA FUNKCI ČI V ČASE (VĚTŠINA HER). **ADWARE** – ZOBRAZUJE SE REKLAMA. **PUBLIC DOMAIN** – VEŘEJNÉ VLASTNICTVÍ – NEJSOU CHRÁNĚNÉ AUTORSKÝM PRÁVEM (VOLNÉ DÍLO, VEŘEJNĚ FINANCOVANÉ DÍLO, AUTOR TAK ROZHODL). **CLOSED SOURCE** – S NEJMENŠÍ SVOBODOU, NEMŮŽE PROHLÍŽET ZDROJOVÝ KÓD, UPRAVOVAT ANI ROZŠIŘOVAT (MS WINDOWS, MS OFFICE). **PLNÁ VERZE ZDRAMA** – NELZE VOLNĚ ŠÍŘIT ANI UPRAVOVAT. **OEM – ORIGINAL EQUIPMENT MANUFACTURER** – SOFTWARE PRODÁVANÝM VÝROBCŮM HARDWARU A TĚM CO SESTAVUJÍ POČÍTAČE. JSOU SPOJENY S HARDWAREM. (SW K FOŤÁKU, TISKÁRNĚ, I WINDOWS)
EULA = END-USER-LICENSE-AGREEMENT – UŽIVATEL MUSÍ PŘED STAŽENÍM NEBO POUŽITÍM PROGRAMU SOUHLASIT S PODMÍNKAMI POUŽITÍ AUTORA PROGRAMU. **KOMERČNÍ DISTRIBUCE OBECNĚ** – PLACENÁ, VÁZÁNA UŽIVATELEM, OMEZENÝ POČET POČÍTAČŮ, DOBY UŽÍVÁNÍ OFFICE 365, LICENČNÍ - KÓD, HARDWAROVÝ KLÍČ – USB FLASH | **PRAKTICKÁ DOVEDNOST:** URČIT CO JE A CO NENÍ SW PIRÁTSTVÍ
INTERNETOVÉ PIRÁTSTVÍ: NAPŘ.: ZPŘÍSTUPNĚNÍ DÍLA VEŘEJNOSTI BEZ SOUHLASU AUTORA | TERMÍNY: PROJÍT TEXT!!!
DOPORUČENÉ ZDROJE: http://zakony.centrum.cz/autorsky-zakon, http://www.stoppiratstvi.cz/cs/o-piratstvi/co-je-dusevni-vlastnictvi.shtml, http://old.zssromotovo.cz/inf/web/S_24.htm, http://knihovna.osu.cz/dokumenty/iv-autorskyzakon.pdf, http://www.mgplzen.cz/download/ivt/ivt_autorsky_zakon.pdf, http://www.upv.cz/dms/pdf_dokumenty/ippv/web_dusevni_vlastnictvi.pdf, http://files.zskaznejov.webnode.cz/200024172-cbb56ccafc/VY_32_INOVACE_2_16.pdf,

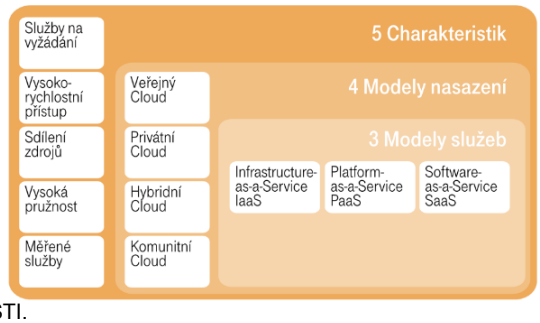
25. Cloud computing, Iaas, Paas, SaaS

CLOUD COMPUTING JE SDÍLENÍ HARDWAROVÝCH A SOFTWAROVÝCH PROSTŘEDKŮ POMOCÍ SLUŽEB NA INTERNETOVÉ SÍTI. HARDWARE, SOFTWARE, DATA A JEJICH STRUKTURY JSOU SDÍLENY PŘES INTERNET A LZE K NIM PŘISTUPOVAT POMOCÍ WEBOVÉHO PROHLÍŽEČE. CLOUD PŘEDSTAVUJE V PODSTATĚ SKUPINU SERVERŮ, KTERÉ POSKYTUJÍ SLUŽBY CLOUD COMPUTINGU.



VÝHODY CLOUD COMPUTINGU:

- VELKÁ ŠKÁLOVATELNOST VÝPOČETNÍCH ZDROJŮ
- VÝPOČETNÍ ZDROJE MŮŽEME NASTAVIT AKTUÁLNĚ V DOBĚ, KDY TO POTŘEBUJEME SAMOOBSLUŽNĚ
- PLATÍ SE JEN ZA TO, CO SE SKUTEČNĚ POUŽIJE. VÝPOČETNÍ ZDROJE SE MĚŘÍ. PLATÍ SE JEN ZA VYUŽITÉ ZDROJE A ZATÍŽENÍ
- UŽIVATELÉ SE NEMUSÍ STARAT O UDRŽOVÁNÍ HARDWARU ANI O AKTUALIZACI OS A POUŽITÉHO SW.
- PŘÍSTUP KE SLUŽBÁM JE V PODSTATĚ ODKUDKOLIV.
- PŘÍSTUP KE CLOUDU S WEB PROHLÍŽEČEM, NEZÁVISLÝ NA PLATFORMÁCH
- DATA JSOU PRAVIDELNĚ AUTOMATICKY ZÁLOHOVANÁ



PĚT CHARAKTERISTIK CLOUDU PODLE NIST (NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY):

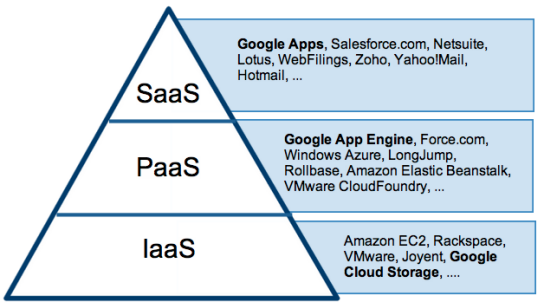
- **ON-DEMAND SELF SERVICE – SLUŽBY NA VYŽÁDÁNÍ.** MOŽNOST NEZÁVISLÉHO USPOŘÁDÁNÍ ZDROJŮ DLE POTŘEBY,
- **BROAD NETWORK ACCESS – VYSOKORYCHLOSTNÍ PŘÍSTUP.** SÍŤOVÝ PŘÍSTUP K POTŘEBNÝM VÝPOČETNÍM ZDROJŮM O DOSTATEČNÉ RYCHLOSTI.
- **RESOURCE POOLING - SDÍLENÍ ZDROJŮ.** SDRUŽOVÁNÍ ZDROJŮ – SERVEROVÝCH A ÚLOŽNÝCH JEDNOTEK DO VELKÝCH CELKŮ, KTERÉ JSOU ZPŘÍSTUPNĚNY PRO VÍCE UŽIVATELŮ.
- **RAPID ELASTICITY - VYSOKÁ PRUŽNOST.** PŘÍSTUP K VÝPOČETNÍM ZDROJŮM MŮŽE BÝT ODSTUPŇOVÁN V REÁLNÉM ČASE. ZÁKAZNÍKOVI MOHOU PŘIPADAT ZDROJE NEOMEZENÉ. JAKO PŘI DODÁVCE ELEKTŘINY
- **MEASURED SERVICE – MĚŘENÁ SLUŽBA.** AUTOMATICKÁ KONTROLA A OPTIMALIZACE ZDROJŮ. VŠE JE MĚŘENO, ABY MĚLI POSKYTOVATELÉ I UŽIVATELÉ PŘEHLED O ODBĚRU. DÍKY TOMU SE PAK DÁ PLATIT PŘESNĚ PODLE VYUŽITÍ – PAY-PER-USE.

SLUŽBA SPLŇUJÍCÍ 5 VÝŠE UVEDENÝCH VLASTNOSTÍ SE MŮŽE OZNAČIT ZA CLOUDOVOU SLUŽBU.

ČTYŘI MODELY NASAZENÍ CLOUDŮ:

- **VEŘEJNÝ CLOUD** – CLOUD JE ZPŘÍSTUPNĚN ŠIROKÉ VEŘEJNOSTI PROSTŘEDNICTVÍM INTERNETU. REALIZACE VELMI VYSOKÝCH ÚSPOR. VLASTNÍKEM INFRASTRUKTURY JE FIRMA, KTERÁ POSKYTUJE CLOUDOVÉ SLUŽBY. VEŘEJNÉ CLOUDOVÉ SLUŽBY ZPOPLATNĚNY PODLE VYUŽÍVÁNÍ, TYPYCKY NA HODINU NEBO MINUTU. ZÁKAZNÍK MŮŽE PLATIT ZA MNOŽSTVÍ CYKLŮ PROCESORU, ÚLOŽNÝ PROSTOR, RYCHLOST PŘIPOJENÍ, … NAPŘ.: AMAZON WEB SERVICES - AWS, MICROSOFT AZURE, IBM/SOFTLAYER, GOOGLE COMPUTE ENGINE
- **PRIVÁTNÍ CLOUD** – JSOU PROVOZOVÁNY JEN PRO ORGANIZACI NEBO FIRMU – PRO INTERNÍ UŽIVATELE V RÁMCI ORGANIZACE. UNIVERZÁLNOST A BEZPROBLÉMOVOST PŘI ZACHOVÁNÍ PLNÉ KONTROLY NAD ZDROJI A BEZPEČNOSTI.
- **HYBRIDNÍ CLOUD** – KOMBINOVANÉ POUŽITÍ RŮZNÝCH FOREM CLOUDU, KTERÉ JSOU NAVZÁJEM LOGICKY ODDĚLENY, ALE UMOŽŇUJÍ VZÁJEMNÉ SDÍLENÍ PROSTŘEDKŮ NAPŘÍKLAD V PŘÍPADĚ KDY VÝPOČETNÍ ZDROJE JEDNÉ ČÁSTI CLOUDU NESTAČÍ NA POKRYTÍ POŽADAVKŮ. VĚTŠINOU SE JEDNÁ O KOMBINACI VEŘEJNÝCH CLOUDOVÝCH SLUŽEB A PRIVÁTNÍCH CLOUDŮ S MOŽNOSTÍ PŘEPÍNÁNÍ MEZI TĚMITO DVĚMA. SPOLEČNOSTI MOHOU SPOUŠTĚT JAK APLIKACE EXTRÉMĚ NÁROČNÉ NA VÝKON, TAK I CITLIVÉ APLIKACE NA PRIVÁTNÍM CLOUDU. CÍLEM HYBRIDNÍCH CLOUDŮ JE VYTVOŘIT UNIFIKOVANÉ, AUTOMATIZOVANÉ, ŠKÁLOVATELNÉ PROSTŘEDÍ MAJÍCÍ VÝHODU NABÍZENOU VEŘEJNÝMI CLOUDY, A ZACHOVÁVAJÍCÍ SI KONTROLU NAD CITLIVÝMI DATY.
- **KOMUNITNÍ CLOUD** – CLOUD SDÍLÍ NĚKOLIK ORGANIZACÍ SE SPOLEČNÝMI ZÁJMY

TŘI MODELY SLUŽEB CLOUDŮ:



- **IaaS – Infrastructure-as-a-Service** – POSKYTUJE UŽIVATELI VÝPOČETNÍ VÝKON, ÚLOŽNÝ PROSTOR, KONEKTIVITU A OSTATNÍ. ZÁKAZNÍK MŮŽE NAINSTALOVAT OPERAČNÍ SYSTÉM A OSTATNÍ APLIKACE. ZÁKAZNÍK NESPRAVUJE ZÁKLADNÍ INFRASTRUKTURU CLOUDU, ALE MŮŽE OVLIVNIT OS, ÚLOŽNÝ PROSTOR, INSTALOVANÉ APLIKACE – SÁM SI JE NAINSTALUJE. TENTO TYP SLUŽEB JE FAKTUROVÁN ZA ODBĚR VÝŠKA PLATBY SE URČUJE PODLE POŽADOVANÉHO VÝKONU NEBO ČASU, PO KTERÝ SE BUDE VÝKON POUŽÍVAT. **VÝHODA:** NÍZKÉ POČÁTEČNÍ NÁKLADY, VLASTNÍ VÝBĚR PROSTŘEDÍ PRO BĚH.
- **PaaS – Platform-as-a-Service** – PRONAJÍMÁME SI PLATFORMU, KTERÁ HOSTUJE NAŠI APLIKACI. VĚTŠINOU JSOU URČENY VÝVOJÁŘŮM – HOSTUJÍ VÝVOJOVÉ NÁSTROJE. NAPÍŠETE APLIKACI A NAHRAJETE NA SERVER. PODOBNÝ PRINCIP JAKO U HOSTINGU. FAKTURUJÍ SE SPOTŘEBOVANÉ MEGACYKLY PROCESORU ZA HODINU. PAAS SAMO ŠKÁLUJE PODLE POTŘEBY VÝKONU (PŘIDÁ VÍCE PROCESORŮ). UŽIVATEL MÁ KONTROLU NAD INSTALOVANÝMI APLIKACEMI. **VÝHODA:** VÝKON ALOKOVANÝ NA ZÁKLADĚ AKTUÁLNÍ POTŘEBY
- **SaaS – Software-as-a-Service** – VYUŽÍVÁ SE HOSTING APLIKACÍ. ZÁKAZNÍK NEKUPUJE SOFTWARE, ALE PRONAJÍMÁ SI HO. ZÁKAZNÍK PLATÍ PODLE TOHO, JAK MOC SOFTWARE POUŽÍVÁ. UMOŽŇUJE PŘÍSTUP K APLIKACÍM PŘES INTERNET. APLIKACE JSOU PŘÍSTUPNÉ Z NEJRŮZNĚJŠÍCH ZAŘÍZENÍ POMOCÍ KLIENTA, JAKO JE NAPŘ. WEB BROWSER. ZÁKAZNÍK JE SCHOPEN NASTAVIT JEN SAMOTNOU APLIKACI

NEJVĚTŠÍ NEVÝHODY CLOUDU: BEZPEČNOST DAT (RELATIVNÍ, EXISTUJÍ CLOUDOVÉ ANTIVIROVÉ SYSTÉMY), NEMÁME DATA POD PŘÍMOU KONTROLOU, POMALEJŠÍ REAKČNÍ DOBA, SERVERY I TISÍCE KM DALEKO, VOLBA HW A SW OMEZENO NABÍDKOU POSKYTOVATELŮ

TERMÍNY: CLOUD COMPUTING, ŠKÁLOVATELNOST, SLUŽBY NA VYŽÁDÁNÍ, SDÍLENÍ ZDROJŮ, VEŘEJNÝ, PRIVÁTNÍ, HYBRIDNÍ, KOMUNITNÍ CLOUD, IaaS, PaaS, SaaS, PAY-PER-USE, INFRASTRUKTURA, PLATFORMA,

PRAKTICKÁ DOVEDNOST: UVEĎTE PRAKTICKÉ PŘÍKLADY SLUŽEB TŘÍ MODELŮ SLUŽEB CLOUDŮ.

DOPORUČENÉ ZDROJE: http://www.cloud.cz/cloud/158-cloud-computingco-ty-pojmy-znamenaji.html, http://www.businessvize.cz/software/co-je-to-cloud-computing-a-proc-se-o-nem-mluvi, http://www.t-systems.cz/produkty-a-reseni/cloud-computing/604902_1/blobBinary/pdf3-ps.pdf, https://dcvizcayno.wordpress.com/2012/04/13/cloud-computing-tips-for-financial-industry/, http://www.nist.gov/itl/cloud/,