Bakalářské zkoušky (příklady otázek)

podzim 2022

1 Programování (3 body)

Pro tuto otázku si vyberte libovolný mainstreamový objektově orientovaný staticky typovaný jazyk (C++, C# nebo Java) a svůj výběr napište na začátek řešení.

1. Naimplementujte/nadeklarujte vhodné objektové rozhraní (pomocí interfaces nebo abstraktních tříd) pro neorientované grafy $G = (V, E)$ ve své nejběžnější podobě (hrany spojují právě dva vrcholy, mezi dvěma vrcholy vede nejvýš jedna hrana). Motivací je, aby toto rozhraní mohly implementovat různé reprezentace grafů (seznam sousedů, matice sousednosti, ...), zatímco programátor grafových algoritmů (BFS, hledání min. kostry, ...) může napsat svůj kód pouze jednou a bude fungovat se všemi reprezentacemi grafů.

   Vámi definované rozhraní by mělo pokrývat 3 entity (*vrchol*, *hranu* a *graf*), přičemž *vrchol* nabídne přístup ke všem incidentním hranám, *hrana* nabídne přístup ke svým dvěma vrcholům a *graf* umožní přístup k seznamu všech vrcholů a seznamu všech hran. Modifikace grafu v tomto rozhraní neuvažujeme (struktura grafu je pouze pro čtení).

   Každý vrchol a každá hrana má příznak (pro jednoduchost mu říkejme *tag*). Tag u vrcholu může například indikovat, zda byl vrchol již navštíven, tag u hrany může například značit její váhu. Tagy tedy mohou mít obecně různé datové typy, avšak pro danou instanci grafu mají všechny vrcholy stejný typ tagu a všechny hrany taktéž. Váš interface by měl být obecný, aby si programátor mohl při jeho implementaci zvolit typy tagů pro vrcholy a pro hrany. Interface také nabídne pro každý vrchol a hranu vhodné prostředky, jak tag přečíst a nastavit.

2. Představme si implementaci vašeho rozhraní, kde tagy vrcholů značí index komponenty souvislosti (typu `int`, indexované od 0) a tagy hran jsou typu `float` (případně podobného typu ve vašem zvoleném jazyce) a značí jejich délky. Tagy hran jsou na počátku již inicializované, tagy vrcholů musíte nastavit vy.

   Napište kód těla funkce využívající vaše rozhraní, která na vstupu dostane graf a reálné číslo r. Váš kód provede rozklad grafu na komponenty souvislosti tak, že každý vrchol bude mít svůj index komponenty uložený jako tag (konkrétní hodnoty indexů nejsou podstatné, důležité je, že vrcholy ve stejné komponentě mají stejný tag). Hrany, které mají větší délku (tag) než stanovený parametr r, nebereme při určování komponent v potaz (tedy jako by vůbec neexistovaly).

2 Virtuální paměť (3 body)

Uvažujte architekturu procesoru s podporou stránkování a délkou virtuální a fyzické adresy 32 bitů. Pro jednoduchost předpokládejte, že k překladu adres procesor používá jednoúrovňovou stránkovací tabulku. Velikost stránek je 4 KiB. Vedle absolutně nezbytných atributů je možné pro jednotlivé stránky nastavit, zda je smí proces modifikovat a zda je možné v nich spouštět kód. Procesor také pro jednotlivé stránky poskytuje informaci o tom, zda se k dané stránce přistupovalo a zda do ní bylo zapsáno.

Část A

1. Navrhněte a schematicky znázorněte formát položky stránkovací tabulky a vysvětlete význam jednotlivých polí. Položka stránkovací tabulky musí obsahovat všechny nutné informace a musí být pro procesor efektivně přístupná (maximálně 1 čtení z paměti).
2. Kolik položek bude mít stránkovací tabulka a kolik paměti bude tato tabulka zabírat pro každý spuštěný proces?

Část B

1. Napište (v pseudokódu) funkci `create_pte`, která pro zadanou fyzickou adresu a atributy stránky vrátí položku stránkovací tabulky. Funkce by měla mít zhruba následující signaturu:

```
pte_t create_pte(address_t phys_addr, bool present, bool writable, bool executable)
```

Typ `address_t` je celočíselný typ, který postačuje k uložení fyzické adresy předané v parametru `phys_addr`. Typ `pte_t` je celočíselný typ, který reprezentuje Vámi navrženou položku stránkovací tabulky (a splňuje kritérium pro efektivní přístup).

2. Napište (v pseudokódu) funkci `set_mapping`, která do stránkovací tabulky pro zadanou virtuální adresu vloží požadovaný záznam. Funkce by měla mít zhruba následující signaturu:

```
void set_mapping(pte_t[] page_table, address_t virt_addr, pte_t entry)
```

Předpokládejte, že stránkovací tabulka je reprezentována jako pole položek typu `pte_t`, takže s parametrem `page_table` můžete zacházet jako s polem. Parametr `virt_addr` představuje virtuální adresu, pro kterou máte do tabulky vložit položku v parametru `entry`.

Část C

Předpokládejte, že **souvislý** blok 4 stránek virtuální paměti od adresy `0x0000B000` byl namapován na **souvislý** blok fyzické paměti. Blok reprezentuje část heapu procesu, obsah paměti je tedy možné číst a modifikovat, ale není možné v něm spouštět kód. Předpokládejte také, že proces **přečetl** obsah proměnné (o velikosti 4 B) z virtuální adresy `0x0000CE90`, což bylo přeloženo na přístup do fyzické paměti na adrese `0xA1018E90`. Pozměněnou hodnotu proměnné pak proces **zapsal** na virtuální adresu `0x0000D854`.

1. Uveďte, na kterých fyzických adresách budou uloženy položky stránkovací tabulky pro výše uvedený blok paměti, pokud víte, že stránkovací tabulka procesu je uložena v souvislém bloku fyzické paměti od adresy `0x13400000`.
2. Uveďte (hexadecimální) hodnoty všech položek stránkovací tabulky pro výše uvedený rozsah virtuální paměti, včetně všech atributů.

3 Posloupnosti (3 body)

Nechť $(a_n) = (a_1, a_2, \ldots)$ je posloupnost reálných čísel.

1. Definujte, co znamená, že (a_n) je konvergentní (má vlastní limitu).
2. Definujte, co znamená, že (a_n) je neomezená.
3. Dokažte, že když (a_n) je neomezená, pak není konvergentní.

4 Báze a ortogonální báze (3 body)

Uvažujme matici $A = BC$, kde

$$B = \begin{pmatrix} 1 & 3 \\ 2 & 2 \\ 1 & 2 \\ 1 & 0 \\ 1 & -1 \end{pmatrix}, \quad C = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 5 & 4 & 3 & 2 & 1 \end{pmatrix}.$$

1. Najděte bázi sloupcového prostoru matice A.
2. Definujte pojem ortogonální báze.
3. Najděte ortogonální bázi sloupcového prostoru matice A.

5 Ramseyova teorie (OI) (3 body)

1. Zformulujte Ramseyovu větu pro grafy, s libovolným počtem barev.
2. Nechť $R(a, b)$ je nejmenší přirozené číslo takové, že každý graf s $R(a, b)$ vrcholy obsahuje kliku velikosti a nebo nezávislou množinu velikosti b. Ukažte, že $R(a, b) \leq R(a-1, b) + R(a, b-1)$.
3. Ukažte, že pro každé přirozené číslo $b \geq 1$ existuje graf bez trojúhelníků barevnosti alespoň $(R(3, b+1) - 1)/b$.

6 Párování v grafech (OI) (3 body)

1. Zformulujte Tutteovu větu o existenci perfektního párování.
2. Která z následujících tvrzení platí a proč?
   (a) Každý hranově 2-souvislý 3-regulární graf má perfektní párování.
   (b) Každý hranově 4-souvislý 4-regulární graf má perfektní párování.
   (c) Má-li graf G s n vrcholy perfektní párování, pak každá nezávislá množina v G má velikost nejvýše $n/2$.
   (d) Má-li největší nezávislá množina v grafu G s n vrcholy velikost nejvýše $n/2$, pak G má perfektní párování.

   Odpovědi zdůvodněte.

7 Optimalizace (OI) (3 body)

1. Uvažte následující definici a tvrzení.

   Definice. Volný vrchol, vzhledem k párování M, je vrchol, který není incidentní s žádnou hranou z M.

   Střídavá cesta, vzhledem k párování M, je cesta, jejíž každý vnitřní vrchol je incidentní s právě jednou hranou z M, která leží na cestě.

   Volná střídavá cesta, vzhledem k párování M, je střídavá cesta, jejíž oba koncové vrcholy jsou volné.

   Kytička, vzhledem k párování M, je podgraf tvořený dvěma částmi, stonkem a květem, s následujícími vlastnostmi:

   - stonek je střídavá cesta sudé délky začínající ve volném vrcholu,
   - květ je kružnice liché délky, která se se stonkem protíná právě v jeho posledním vrcholu,
   - každý vrchol květu s výjimkou vrcholu patřícího i do stonku, je incidentní s právě jednou hranou z M, která patří do květu.

   Lemma 1. Párování M v grafu $G = (V, E)$ je největší právě když v grafu G neexistuje volná střídavá cesta vzhledem k M.

   Lemma 2. Nechť M je párování M v grafu G a C je květ nějaké kytičky v G, vzhledem k M. Pak graf G obsahuje volnou střídavou cestu právě tehdy když graf G' získaný z G kontrakcí květu do jediného vrholu obsahuje volnou střídavou cestu (vzhledem k restrikci párování M na G').

   Dokažte aspoň jedno z uvedených dvou lemmat.
2. Popiště Edmondsův algoritmus na nalezení největšího párování v grafu $G = (V, E)$ a s pomocí lemmat 1 a 2 dokažte jeho správnost.

8 Funkce více proměnných (OI) (3 body)

Nechť $f(x, y, z, u) = ax^3 + by^3 + cz^3 + du^3$ je funkce čtyř proměnných x, y, z, u definovaná na $\mathbb{R}^4$, přičemž a, b, c, d jsou reálné parametry.

1. Vypočítejte parciální derivace $\partial^2 f/\partial z^2$ a $\partial^3 f/\partial x^2 \partial y$.
2. Nalezněte všechny hodnoty parametrů, pro něž f má v počátku (tj. v bodě $(0, 0, 0, 0)$) ostré lokální maximum. Poznámka: tuto úlohu lze řešit pomocí parciálních derivací i bez nich.

Své řešení zdůvodněte.

9 Zarovnání (PVS) (otázka studijního zaměření – 3 body)

Mějme fragment kódu v jazyce C

```
struct data_t {
        int    ver;
        double x, y;
        int    spin;
        char   orientation;
```

```
};

data_t data[128];
```

Architektura instrukční sady daného CPU vyžaduje striktní přístup do paměti na zarovnané adresy. Velikosti základních datových typů jsou následující:

char	1 B
int	4 B
double	8 B

1. Napište, na jakém posunutí od začátku pole leží položka `data[23].orientation`
2. Pokud začátek pole leží na adrese `0xA8243B40`, na jaké adrese leží výše uvedená položka?
3. Co se stane v případě, že se program pokusí o nezarovnaný přístup (vyberte jednu z možností, příp. doplňte vysvětlení)?
   (a) Nestane se nic a program bude pokračovat dál ve vykonávání instrukcí
   (b) CPU vyvolá výjímku a program bude ukončen
   (c) Tato situace nemůže nastat, protože to je zajištěno již architekturou instrukční sady
   (d) Stane se něco jiného, v tom případě doplňte vysvětlení

10 Politiky (PVS) (otázka studijního zaměření – 3 body)

Mějme následující fragment šablony matice v jazyce C++:

```
template<typename T> class Matrix
{
private:
    std::vector<T> data;
    std::size_t M, N;
public:
    // ...
    T operator[](std::size_t i, std::size_t j) { return data[i*M + j]; }
};
```

Tato implementace matice používá řádkovou reprezentaci dat, kdy jsou řádky matice lineárně seřazeny za sebe v paměti. Navrhněte a implementujte způsob, kterým by uživatel mohl v době kompilace zvolit vnitřní reprezentaci takové matice, tj. zda chce mít matici uloženou v paměti po řádcích nebo po sloupcích. Navrhněte takové rozhraní, aby uživatel případně mohl naimplementovat vlastní vnitřní organizaci dat (např. trojúhelníkové uložení). Implementaci (výkonný kód) tohoto řešení ale nepište, pouze rozhraní.

11 Relační databáze (PVS) (otázka studijního zaměření – 3 body)

Pro potřebu správy rozvrhů jsou definovány následujících uživatelské požadavky:

- Předmět má název, právě jednoho garanta a může být vyučován v LS, ZS nebo obou semestrech.
- Každý předmět může mít 0-N přednášek.
- Každá přednáška může mít 0-N vyučujících.
- Student může být zapsán na přednášku nejvýše jednou.
- Každý předmět může mít 0-N cvičení.
- Každé cvičení má maximální kapacitu a může mít 0-N cvičících.
- Student může být zapsán na cvičení nejvýše jednou.
- Každá přednáška i cvičení má specifikované délku a čas začátku.
- Garanti předmětů, cvičící, přednášející a studenti jsou identifikování pomocí *externího identifikátoru*.

Externí identifikátor, je textový řetězec o maximální délce 45 znaků.

1. Navrhněte databázové relační schéma pro správu rozvrhů podle výše uvedených uživatelských požadavků. Schéma musí splňovat 3NF. Schéma popište pomocí ER diagramu, nebo jako seznam typovaných sloupců v tabulkách s vyznačeným primárním klíčem a cizími klíči.
2. Napište SQL dotaz, který pro vytvořené schéma vrátí pro každý předmět a vyučujícího (přednášející či cvičící), kolik studentů učí.
3. Na straně databáze je třeba logovat vytvoření a rušení zápisů (dotazy INSERT a DELETE) studentů na předměty a cvičení. Navrhněte a popište možné řešení na straně DBMS, bez zásahu do aplikační logiky či změně posílaných dotazů. Popište případné dopady řešení na databázové schéma.

Při řešení můžete použít SQL dialekt běžně známých databází (Oracle, MySQL, PostgreSQL, MSSQL, ...).

12 API (PVS) (otázka studijního zaměření – 3 body)

1. Navrhněte a popište REST API pro správu předmětů, cvičení a zápisu na cvičení z předchozího příkladu. API musí umožňovat
   (a) získat seznam předmětů
   (b) vytvořit a zrušit předmět, získat detail předmětu
   (c) získat seznam cvičení předmětu
   (d) vytvořit a zrušit cvičení k předmětu
   (e) získat zapsané studenty na jednotlivá cvičení
   (f) vytvořit a zrušit zápis studenta na cvičení

   API musí používat JSON pro sdílení dat.
2. Napište JavaScript kód, který z daného REST API získá seznam cvičení a následně smaže všechny s nulovou kapacitou.
3. Popište jaký/é návrhový/é vzor/y a proč by bylo vhodné použít pro tvorbu webové aplikace pro přihlašování studentů na přednášky a cvičení. Můžete zmínit i některý/é framework/y a jejich vztah ke zmiňovaným návrhovým vzorům.

Ve příkladech kódu není třeba extenzivně ošetřovat chybové stavy. Na drobné syntaktické chyby nebude při hodnocení brán zřetel.

13 Relační databáze (DW) (otázka studijního zaměření – 3 body)

Pro potřebu správy rozvrhů jsou definovány následujících uživatelské požadavky:

- Předmět má název, právě jednoho garanta a může být vyučován v LS, ZS nebo obou semestrech.
- Každý předmět může mít 0-N přednášek.
- Každá přednáška může mít 0-N vyučujících.
- Student může být zapsán na přednášku nejvýše jednou.
- Každý předmět může mít 0-N cvičení.
- Každé cvičení má maximální kapacitu a může mít 0-N cvičících.
- Student může být zapsán na cvičení nejvýše jednou.
- Každá přednáška i cvičení má specifikované délku a čas začátku.
- Garanti předmětů, cvičící, přednášející a studenti jsou identifikování pomocí *externího identifikátoru*.

Externí identifikátor, je textový řetězec o maximální délce 45 znaků.

1. Navrhněte databázové relační schéma pro správu rozvrhů podle výše uvedených uživatelských požadavků. Schéma musí splňovat 3NF. Schéma popište pomocí ER diagramu, nebo jako seznam typovaných sloupců v tabulkách s vyznačeným primárním klíčem a cizími klíči.
2. Napište SQL dotaz, který pro vytvořené schéma vrátí pro každý předmět a vyučujícího (přednášející či cvičící), kolik studentů učí.

3. Na straně databáze je třeba logovat vytvoření a rušení zápisů (dotazy INSERT a DELETE) studentů na předměty a cvičení. Navrhněte a popište možné řešení na straně DBMS, bez zásahu do aplikační logiky či změně posílaných dotazů. Popište případné dopady řešení na databázové schéma.

Při řešení můžete použít SQL dialekt běžně známých databází (Oracle, MySQL, PostgreSQL, MSSQL, ...).

14 HTML & CSS (DW) (otázka studijního zaměření – 3 body)

Uvažujme následující fragment HTML kódu:

```
<div class="header">
  Hlavička
  <ul class="navbar">
    <li class="active">Fórum</li>
    <li>Zápisy</li>
    <li>Přehled</li>
    <li>Nápověda</li>
  </ul>
</div>
<div class="forum">
  <div class="post">
    First
  </div>
  <div class="post">
    Lorem ipsum ..
  </div>
</div>
<div>
  <form action="./create-post.php" method="post">
    <label>
      Post content:
      <textarea name="content" maxlength="280"></textarea>
    </label><br>
    <input type="submit" value="Submit">
  </form>
</div>
<div class="footer">
  <ul>
    <li>Copyright 2022</li>
    <li>Design by ProfDesign</li>
  </ul>
</div>
```

K HTML stránce obsahující zmíněný fragment jsou přilinkované následující CSS definice:

```
li {color:blue}
ul li {color: red;}
.navbar li {color: black;}
.header .navbar li {color: green;}
#container li {color: violet;}
li .class {color:brown}
```

Relevantní úryvek z PHP skriptu "create-post.php"pro validaci dat z formuláře:

```
function isFormPostValid() {
  return isset($_POST['content']) && strlen($_POST['content']) <= 280;
}
```

1. Vysvětlete princip určení specificity CSS selektorů (na čem specificita záleží, jak konkrétně se počítá). Který/které z výše uvedených CSS selektorů má nejvyšší specificitu?
2. Jakou barvu bude mít text "Fórum"v hlavičce ?

3. Na straně serveru jsou data z formuláře validována pomocí uvedeného PHP skriptu. Bylo by možné validaci na straně serveru vynechat? Odpověď stručně zdůvodněte.

15 API & Skriptování (DW) (otázka studijního zaměření – 3 body)

1. Navrhněte a popište REST API pro správu předmětů, cvičení a zápisu na cvičení z předchozího příkladu. API musí umožňovat
   (a) získat seznam předmětů
   (b) vytvořit a zrušit předmět, získat detail předmětu
   (c) získat seznam cvičení předmětu
   (d) vytvořit a zrušit cvičení k předmětu
   (e) získat zapsané studenty na jednotlivá cvičení
   (f) vytvořit a zrušit zápis studenta na cvičení

   API musí používat JSON pro sdílení dat.
2. Napište JavaScript kód, který z daného REST API získá seznam cvičení a následně smaže všechny s nulovou kapacitou. Pokud v řešení použijete funkce poskytnuté prostředím (prohlížeč), krátce popište jejich význam (zejména pokud si nejste jisti přesným názvem). V kódu není třeba extenzivně ošetřovat chybové stavy. Na drobné syntaktické chyby nebude při hodnocení brán zřetel.
3. V rámci implementace vznikl následující fragment PHP kódu:

```
<?php
  $result = $mysqli->query('SELECT * FROM enrollment_view WHERE id="' . $_GET['enrollment'] + '"');
  while ($row = $result->fetch_assoc()) {
    echo('<li>' . $row['subject'] . ':' . $row['student'] . '</li>');
  }
?>
```

   Identifikujte a popište alespoň dva závažné bezpečnostní nedostatky v uvedeném PHP kódu.

16 Datový management (DW) (otázka studijního zaměření – 3 body)

Pro zlepšení interoperability je třeba navrhnout a zkonstruovat tezaurus (kontrolovaný slovník) IT dovedností pomocí ontologie SKOS. Ten musí obsahovat zejména dovednosti pro: zpracování dokumentu, editaci dokumentu, tisk dokumentu. Editace dokumentu a tisk dokumentu jsou specializací dovednosti zpracování dokumentu. Všechny dovednosti musí mít uvedený label a být částí schématu s labelem "Dovednosti".

1. Vytvořte výše popsaný tezaurus a zapište ho jako RDF.
2. Pomocí vytvořeného tezauru anotujte následující předměty:
   (a) "Tisk"- V tomto předmětu se studenti naučí jak tisknout dokumenty, ale nic více.
   (b) "Dokumenty"- V tomto předmětu se studenti naučí jak editovat a tisknout dokumenty.
3. Napište SPARQL dotaz, který získá URL všech předmětů, na kterých se studenti naučí tisknout dokumenty.

Při zápisu můžeme využít předdefinovaný prefix *skos (http://www.w3.org/2004/02/skos/core#). Při hodnocení nebude brán zřetel na drobné syntaktické chyby. V případě potřeby vhodně zvolte vlastní URL.*

17 Barvení grafu, DPLL (UI-SU) (otázka studijního zaměření – 3 body)

Nechť $G = (V, E)$ je neorientovaný graf s N vrcholy ($|V| = N$).

1. Definujte pojem konjunktivní normální forma (CNF) formule ve výrokové logice.
2. Navrhněte formuli ve výrokové logice, která je splnitelná, právě když je graf G obarvitelný k barvami. Formuli popište obecně v konjunktivní normální formě (CNF) pro libovolný graf G a libovolné $k > 0$.

3. Napište formuli pro úplný graf se třemi vrcholy a pro $k = 3$. Na této formuli demonstrujte použití algoritmu DPLL.

18 Přeučení a regularizace (UI-SU) (otázka studijního zaměření – 3 body)

Předpokládejme, že chceme trénovat lineární model pro regresi.

1. Definujte střední čtvercovou chybu (MSE). Definujte upravenou chybovou funkci používanou při L_2 regularizaci.
2. Mějme dva lineární modely M_1, M_2 pro data se čtyřmi atributy. Nechť

$$\begin{aligned} M_1(\vec{x}) &= x_1 - 2x_2 + 4x_3 - x_4 + 3 \\ M_2(\vec{x}) &= x_2 - 5x_3 + x_4 - 1. \end{aligned}$$

   Předpokládejme, že oba modely mají pro daná data stejnou MSE. Který z modelů má menší hodnotu upravené chybové funkce pro L_2 regularizaci?
3. Popište alespoň jeden další způsob pro zlepšení generalizačních schopností modelů strojového učení (obecně, nemusí být pro lineární regresi).

19 Rozhodovací stromy (UI-SU) (otázka studijního zaměření – 3 body)

1. Definujte rozhodovací strom. Popište algoritmus pro trénování rozhodovacího stromu.
2. Definujte kritéria pro výběr atributu pro větvení pro klasifikaci a pro regresi (alespoň jedno pro klasifikaci a alespoň jedno pro regresi).
3. Mějme data s kategorickými atributy A_1, A_2 a cílovou hodnotou C uvedená v tabulce níže. Podle kterého z atributů budeme data dělit v kořeni rozhodovacího stromu pro regresi a proč?

A_1	A_2	C
0	0	5
0	0	3
0	1	4
1	0	6
1	1	2
1	1	4

20 k nejbližších sousedů (UI-SU) (otázka studijního zaměření – 3 body)

1. Popište obecně fungování techniky k nejbližších sousedů (k-NN) pro klasifikaci i pro regresi: Jak funguje trénování modelu? Jak probíhá předpovídání?
2. Mějme data se dvěma příznaky A_1 a A_2 a dvěma třídami $C \in \{0, 1\}$ v tabulce níže. Zakreslete data a rozhodněte, jak bude pomocí algoritmu k-NN pro $k = 3$ ohodnocen nový vstup $(5, 5)$. Používejte Manhattanskou vzdálenost.
3. Změní se odpověď na předchozí otázku při změně k? Jak? Uveďte alespoň jeden příklad k, pro které je odpověď jiná (pokud existuje).

A_1	A_2	C
0	3	0
2	2	0
3	0	0
4	2	0
6	6	1
4	5	1
1	1	1

21 HDR grafika (PGVVH-VPH) (otázka studijního zaměření – 3 body)

Otázka se týká HDR (High Dynamic Range) grafiky, principů a aplikací.

1. Z jakých důvodů a v jakých oblastech grafiky se používá HDR grafika? Uveďte příklady, kdy nám může pomoci nejvíce.
2. Jak se HDR obraz reprezentuje v paměti počítače a v jakém formátu ho můžeme zapsat na disk? Naznačte postup pořízení HDR obrázku s pomocí fotoaparátu a stativu.
3. Jak se HDR obraz prezentuje pozorovateli na běžném (LDR) výstupním zařízení? (není potřeba uvádět nějaké komplikované postupy, jen popsat princip)

22 Scanline vyplňování (PGVVH-VPH) (otázka studijního zaměření – 3 body)

Budeme se zabývat vybarvením vnitřku rovinného mnohoúhelníka v rastrovém prostředí (čtvercová mřížka pixelů), obrys útvaru není potřeba obkreslovat.

1. Jak by měl být mnohoúhelník zadán a jaké možnosti definice jeho vnitřku mají smysl?
2. Popište základní princip algoritmu řádkového vyplňování 2D mnohoúhelníka (uveďte i případné pomocné datové struktury používané v průběhu vyplňování).
3. Jak by se algoritmus zjednodušil, kdyby měl vyplňovat pouze konvexní útvary?

23 3D scéna pro kreslení na GPU (PGVVH-VPH) (otázka studijního zaměření – 3 body)

1. Jak byste v paměti počítače reprezentovali 3D scénu, která se pak bude v reálném čase vykreslovat na grafické kartě (GPU)? Nemusíte psát přímo deklarace dat, ale popište dostatečně podrobně svůj návrh slovy.
2. Uvažujte systém hierarchické reprezentace tak, aby se komponenty (objekty, modely) daly jednoduše ve scéně opakovat bez toho, aby se musela data duplikovat.
3. Jak technicky implementovat animaci (pohyb) jednotlivých objektů ve scéně? Omezte se na nedeformovatelná tělesa ("rigid body animation"), která se jen ve scéně přemisťují otáčením a translací.

24 Kvaterniony a jejich využití v animaci (PGVVH-VPH) (otázka studijního zaměření – 3 body)

Jde nám o reprezentace orientace pevného tělesa v 3D prostoru.

Pevné těleso se obvykle v čase animuje tak, že se jako celek posunuje (translace) a otáčí (orientace, rotace). V této otázce se zaměříme pouze na tu druhou - rotační - složku pohybu.

1. Definujte kvaternion, popište, jaký datový typ se při jeho implementaci používá (kolik zabírá paměti)? Napište některé základní algebraické vlastnosti kvaternionů. Jak byste odvodili vzorec pro násobení kvaternionů (abyste si ho nemuseli pamatovat)?
2. Která speciální třída kvaternionů se používá k reprezentaci orientace ve 3D? Jak pomocí kvaternionu otočit bod/vektor kolem dané osy o daný úhel?
3. Jaký je postup při animaci (přechodu) mezi dvěma orientacemi? Počáteční orientaci nechť vyjadřuje kvaternion q a koncovou orientaci kvaternion r. Čas t budeme pro jednoduchost předpokládat v intervalu $[0, 1]$. Napište vzorec pro orientaci v libovolném čase t. Naznačte postup, který by se použil pro složitější animaci mezi více klíčovými orientacemi.

25 Výkon počítače a procesoru (SP) (otázka studijního zaměření – 3 body)

Toto je (naivní) program pro výpočet čísla π částečným součtem řady:

```
1  #include <climits>
2  #include <iostream>
3
4  int main (void) {
5      int i = 1;
6      int s = 1;
7      double x = 0;
8      while (i < INT_MAX) {
```

```
9         x += s * ((double) 1)/i;
10        i += 2;
11        s *= -1;
12    }
13    std::cout << x * 4 << std::endl;
14    return (0);
15 }
```

Cyklus výpočtu přeložil překladač pro procesor řady x86 takto:

```
1         pxor        %xmm0, %xmm0
2         movl        $1, %edx
3         movl        $1, %eax
4 .L2:
5         pxor        %xmm1, %xmm1
6         cvtsi2sdl   %edx, %xmm1
7         pxor        %xmm2, %xmm2
8         cvtsi2sdl   %eax, %xmm2
9         divsd       %xmm2, %xmm1
10        addsd       %xmm1, %xmm0
11        addl        $2, %eax
12        negl        %edx
13        cmpl        $2147483647, %eax
14        jne         .L2
```

Sémantiku instrukcí odhadněte z názvu a funkce programu, zkratky `add`, `div`, `neg` a `xor` označují příslušné aritmetické a logické operace, `cmp` je porovnání, `mov` je přesun hodnoty, `cvtsi2sdl` je přesun s převodem integer na double.

1. Měření ukazují, že procesor vykonal uvedený kód s zhruba IPC 2.5. Vyznačte ve výpisu skupiny instrukcí, v rámci kterých se mohou všechny instrukce vykonat paralelně, tak, abyste mohli paralelním vykonáváním instrukcí vysvětlit alespoň IPC 2.
2. Uvedený kód lze snadno paralelizovat například tak, že iterace cyklu rozdělíme rovnoměrně mezi n vláken. Měření ukazují, že při použití 8 vláken (na 8 nezávislých procesorech) bylo zrychlení programu 7. Odhadněte celkový počet instrukcí (přes všechna vlákna), které bylo potřeba vykonat pro zajištění koordinace vláken. Váš postup vysvětlete.

26 Paralelismus a synchronizace (SP) (otázka studijního zaměření – 3 body)

1. V některém z jazyků C, C#, C++ nebo Java implementujte spin lock. Ze synchronizačních operací zvoleného jazyka může vaše implementace používat pouze standardní atomické operace (včetně běžných operací typu read-modify-write). Ty v implementaci vyznačte.
2. Ve stejném jazyce implementujte ještě rekurzivní zámek s pasivním čekáním. Můžete použit spin lock z předchozího bodu a běžné kolekce. Rozhraní pro pasivní čekání si sami navrhněte a popište jeho sémantiku.

27 Obsluha zařízení (SP) (otázka studijního zaměření – 3 body)

Uvažujte disk jako periferii, která může číst a zapisovat sektory délky 4096 bajtů. Bloky jsou adresované pořadovým číslem (LBA), řadič disku generuje žádost o obsluhu přerušení pro přenos jednotlivých sektorů.

1. Navrhněte rozhraní ovladače popsaného disku, které dovolí číst a zapisovat jednotlivé sektory.
2. Načrtněte implementaci navrženého rozhraní, včetně obsluhy žádosti o přerušení.

Dodržení přesné syntaxe konkrétního programovacího jazyka není nutné, odpovědi mohou použít pseudokód.

28 Testování funkčnosti (SP) (otázka studijního zaměření – 3 body)

Uvažujte následující jednoduchou implementaci zásobníku:

```
1 public class Stack {
2
```

```
3       private int [] data;
4       private int top;
5
6       public Stack (int capacity) {
7           if (capacity < 0) {
8               throw new NegativeArraySizeException ("Creating␣negative␣size␣stack");
9           }
10          data = new int [capacity];
11          top = 0;
12      }
13
14      public void push (int item) {
15          if (top == data.length) {
16              throw new ArrayIndexOutOfBoundsException ("Push␣to␣a␣full␣stack");
17          }
18          data [top] = item;
19          top ++;
20      }
21
22      public int pop () {
23          if (top == 0) {
24              throw new ArrayIndexOutOfBoundsException ("Pop␣from␣an␣empty␣stack");
25          }
26          top --;
27          return (data [top]);
28      }
29  }
```

1. Napište definici pokrytí vhodnou pro jednotkové testování uvedené implementace. Vaše definice by měla zahrnovat vzorec pro výpočet procentuálního pokrytí a význam použitých symbolů.
2. Načrtněte, jaké jednotkové testy jsou potřeba pro úplné pokrytí uvedené implementace podle vámi zvolené definice. Jednotlivé testy popište pseudokódem nebo popisem toho, co konkrétně mají testovat.
3. Napište jednu další definici pokrytí a rozhodněte, která z Vámi uvedených definic pokrytí je pro uvedenou implementaci vhodnější. Své rozhodnutí zdůvodněte.