Univerzita Pardubice

Fakulta elektrotechniky a informatiky

Tomáš Uher

Počítačem podporovaná výuka předmětu základy algoritmizace

Bakalářská práce

2009

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Katedra informačních technologií
Akademický rok: **2008/2009**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Tomáš UHER**

Studijní program: **B2646 Informační technologie**

Studijní obor: **Informační technologie**

Název tématu: **Počítačem podporovaná výuka předmětu Základy Algoritmizace**

Zásady pro vypracování:

V úvodní části práce je nutné provést seznámení s pojmem blended e-learning Ô Stanovení obsahu e-learningového kurzu, Zhotovení hrubého konceptu podoby studijního materiálu, navržení grafického ztvárnění studijních materiálů Hlavní úkol spočívá ve výběru témat jednotlivých kapitol (modulů) a vytvoření studijního materiálu, ve kterém je vysvětlena učební látka k příslušnému tématu. Doplněním studijních materiálů jsou kontrolní otázky ve formě testu, které ověřují, zda si student osvojil probíranou látku, a praktické úkoly, ve kterých student uplatňuje poznatky získané ze studijních materiálů. Ke každému úkolu jsou k dispozici pokyny pro vypracování, potřebná data či odkazy na ně a vzorové řešení.

Rozsah grafických prací:

Rozsah pracovní zprávy:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **Www.moodle.cz [online]. 2005 [cit. 2008-09-03]. Dostupný z WWW: . 2. ŠVEJDAR , Vítězslav. Logika neúplnost, složitost a nutnost. [s.l.] : Academia, 2002. 464 s**

Vedoucí bakalářské práce: **Ing. Soňa Neradová**
Katedra softwarových technologií

Datum zadání bakalářské práce: **15. ledna 2009**

Termín odevzdání bakalářské práce: **15. května 2009**

Univerzita Pardubice Česká republika

L.S.

doc. Ing. Simeon Karamazov, Dr.
děkan

Ing. Lukáš Čegan
vedoucí katedry

V Pardubicích dne 31. března 2009

Prohlášení autora

Prohlašuji:

Tuto práci jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 13. 8. 2009

Tomáš Uher

Anotace

Práce se ve své části věnuje možnostem a využití e-learningu zaměřeným na blended learning. V druhé části se práce věnuje vytvoření webové aplikace a studijních materiálů pro základy algoritmizace.

Klíčová slova

e-learning, blended learning, algoritmizace, Java

Title

Computer supported education subject bases algorithm development

Annotation

Work in their part of the opportunities and the use of dedicated e-learning focused on blended learning. In the second part of the work is devoted to the creation of Web application and study materials for the foundations algorithmization.

Keywords

e-learning, blended learning, algorithmization, Java

Poděkování

Rád bych poděkoval vedoucí mé bakalářské práce Ing. Soně Neradové za její čas a návrhy při práci.

Obsah

Seznam zdrojových kódů

Seznam obrázků

Seznam tabulek

1. Seznam zkratek

ASP – Active Server Pages

CSS – Cascading Style Sheets – tabulka kaskádových stylů

SQL – Structured Query Language – strukturovaný dotazovací jazyk

2. Úvod

V dnešní době, kdy už téměř každý vlastní počítač, chytrý mobilní telefon a další chytrá zařízení, byla jen otázka času, kdy se začne rozvíjet elektronická výuka do všedního života. Elektronická výuka byla vytvořena už před desítkami let, ale až v dnešní době se začíná maximálně prosazovat a tedy i rozšiřovat v různých odvětvích.

Cílem této bakalářské práce je seznámit s pojmy elektronické výuky a vypracování webové aplikace s vytvořením studijních materiálů pro výuku předmětu Základy algoritmizace.

3. E-learning

Je efektivní využívání IT technologií pro vzdělání.

3.1 Vztah mezi klasickou výukou a e-learningem

E-learning jsou z tohoto pohledu nové možnosti, které lze využít pro výuku. Klasická výuka je za pomocí lektora, která ale má plno nedostatků, proto se e-learning snaží tyto nedostatky odstranit.

3.2 E-learning a jeho využití

E-learning se používá v mnoha odvětvích. Tato odvětví jsou různorodá a začínají školami a končí armádou.

4. Blended learning

je kombinace standardní výuky s výukou elektronickou.

4.1 Proč vlastně tento typ výuky vznikl

Hlavním důvodem vzniku blended learningu byly nedostatky ve výuce pomocí e-learningu pro různé uživatele. Z tohoto důvodu se začal využívat blended learning v kombinaci s jiným druhem výuky. Je to tedy více metod výuky využívaných ke vzdělání. I když se těchto více metod využije, neznamená to cestu k úspěchu, pro co nejlepší výsledek se musí jednotlivé metody správně zkombinovat. Tím se myslí, využít pro každou část výuky tu nejlépe možnou metodu výuky. A výukové části správně zrealizovat časově, aby i různé druhy výuky na sebe navazovali. Lektor má možnosti získat zpětně data, jak si student v kterých lekcích vedl, kolik získal bodů, kolik strávil času v různých částech výuky. Pomocí těchto hodnot může pak se studentem komunikovat více

a má více možností než by měl při klasické výuce. Namísto neustálého opakování pořád stejné látky v učebnách pomocí klasické výuky. Lektor se pak může intenzivněji věnovat přípravě látky, zdokonalování už vytvořených látek a jejich aktualizacích. Jsou to tedy pro lektory výkonné nástroje pro snadný a rychlý převod vlastních zkušenosti, dovedností a znalostí do podoby, která tyto aktiva udělá okamžitě dostupné všem, kteří je využijí.

4.2 Synchronní výuka

Tato výuka probíhá pomocí komunikačních kanálů jako je chat, sdílené aplikace, video konference, virtuální třídy. Je to tedy druh výuky, kdy výuka probíhá v reálném čase, kdy všichni studenti současně přijímají nové poznatky a mohou o nich konzultovat. Mezi tyto typy výuky patří například výuka v učebnách, kdy studenti i vyučující jsou ve stejném prostoru a čase a mají plnou možnost konzultovat názory k probírané látce. Dále do tohoto typu výuky patří i virtuální učebny, kdy studenti fyzicky nejsou na stejném místě, ale pomocí počítače a multimediálních komunikačních prostředků mohou reagovat ve stejný čas, i když fyzicky jsou od sebe několik kilometrů.

Hlavní výhody synchronní výuky:

- vyučující může reagovat různě na podněty studentů,
- dovoluje studentům a vyučujícím vzájemně komunikovat a usměrňovat výuku,
- jednodušší na přípravu.

Nevýhody:

- vyžaduje plánování prostorů pro výuku,
- potřeba placení cestovních nákladů pro studenty,
- tempo výuky není nastaveno pro jednotlivce,
- způsob učení také není nastaven pro podmínky jednotlivce.

4.3 Asynchronní výuka

Tato výuka probíhá pomocí komunikačních kanálů jako je e-mail, diskusní skupiny, online nástěnky. Asynchronní výuka je aplikovaná v různých časech, může se jí účastnit jeden i více studentů. Do této kategorie výuky patří manuály, odborné knihy, video nahrávky, výukové programy.

Mezi výhody patří:

- není závislá na časovém harmonogramu studentů,
- jednoduše se distribuuje,
- studenti si volí vlastní tempo studia,
- studenti si mohou zvolit hloubku a věnování jednotlivým částem výuky,
- dobrá standardizace,
- dobré pro fakta a koncepty.

Nevýhody jsou:

- drahá a časově náročná výroba,
- některým studentům dělá problém samostudium,
- špatná komunikace s lektory a ostatními studenty.

4.4 Hlavní body pro určení druhu výuky:

- **Stabilita obsahu** – pojednává o tom, jak se bude obsah měnit. V případech kdy víme, že se obsah bude měnit jen párkrát za pár let, je lepší využít asynchronního způsobu výuky. Ale v okamžiku dynamicky se měnícího obsahu je výhodnější využít synchronní výuky. Kde ušetříme náklady na výrobu a přepracování celého obsahu do formy pro asynchronní výuku. V případech kdy se jen některá část obsahu výuky bude měnit, je možné využít toho, že u obsahu neměnitelného využijeme asynchronní obsah, a pro měnitelná data využijeme části, které půjde jednoduše měnit.
- **Časová flexibilita** – při krátkém čase na přípravu výuky je jednodušší využít synchronní výuky, která nepotřebuje na vytvoření tolik času a

zdrojů. Protože synchronní výuka potřebuje čas hlavně na zorganizování, ale na připravení materiálů už ne tolik, protože vyučující nemusí upřesňovat všechny detaily, ale může reagovat na vědomosti studentů.

- **Počty studujících** – zde při velkých počtech se vyplatí využít asynchronní výuky, z důvodu kompenzace nákladů vytvořených prací na asynchronní výuku vrátí v ušetřených poplatcích za dopravu studentů. Pro malé skupinky zase bude výhodnější využít synchronní výuku z opačných důvodů.
- **Komplikovanost výuky** – je důležitý faktor pro výběr metody výuky. Pro výuku kde není složité vysvětlování, se dobře využije asynchronní výuka. Ale musí se využít zajímavého podání výuky s možnostmi asynchronní výuky. V případech kdy je potřeba rozsáhlých možností vysvětlení a výuka potřebuje spolupráci a předvádění vědomostí v reálném čase. Pro tuto výuku se dobře hodí synchronní výuka.
- **Struktura obsahu -** pro obsah využívající komplexních situací a studenti musejí aplikovat v určitých situacích. Odpovědi u tohoto tématu mají mnoho možností jak odpovědět a je tu potřeba zpětná vazba. Pro tento typ výuky poslouží synchronní vypracování výuky. V případech kde jsou definovaná fakta a otázky mají jasný výsledek a obsah je tedy strukturovaný je dobré využít asynchronní zpracování.
- **Podobnost s realitou** - pro potřeby co nejvíc přizpůsobit výuku skutečnému životu tedy praxi. I zde tedy je rozhodující pro výběr správného typu výuky, požadavky na výuku. Pro příklady kdy jsou virtuální simulátory téměř identické pro práci (jako v praxi) se hodí asynchronní výuka, kdy se student může vzdělávat sám. Pro jiné typy, hlavně jak zvládnout nějaké situace, je potřeba komunikace s vyučujícím a jinými studenty bude tedy potřeba synchronní výuky.

Pro jednu výuku nemusí být využito jen jednoho typu výuky. Nejlepší výsledek bude zkombinovat různé metody pro jednotlivé části, tak aby využily svého maximálního potenciálu a eliminovaly nevýhody druhé metody výuky.

4.5 Efektivita e-learningu, proč ho zavést

Podle mnohých studií nám e-learning sníží náklady na výuku, povede se vyučovat více studentů, udělat studium obsáhlejší, lépe řídit vzdělávací procesy.

Již v roce 1959 na Wisconsinské univerzitě vyvinul Donald L. Kirkpatrick metody pro měření efektivity školících programů. Jeho model zahrnuje čtyři stupně vyhodnocení.

Tyto čtyři stupně se ptají:

1) Reakce – Jak studenti reagují na školení?
2) Výuka – Kolik se toho naučili?
3) Chování – Jak se změnilo jejich chování?
4) Výsledky – Jaký efekt mělo školení pro organizaci?

Další stupeň přidal Jack Philips a to:

5) Návratnost investic – Převážily výsledky školení jeho cenu?

I když byly tyto body navrhnuty před několika desítkami let, i dnes jsou aktuální a používají se jako metody měření efektivity e-learningu.

4.6 Vysvětlení stupňů:

1) Reakce – Měří se názor studenta na výuku. Jeho přínosnost, zajímavost.
2) Výuka – Tento stupeň se snaží rozeznat rozdíl vzdělání před výukou a po výuce, například testem probíhajícím právě před testem a po testu.
3) Chování – Jak se změnil u studentů postoj k problematice, jestli změnili své návyky pro práci a jestli byly tyto změny k lepšímu nebo horšímu.
4) Výsledky – Snaží se zjistit, jaký měla výuky smysl. A to se snaží řešit pomocí dvou ukazatelů, takzvaného ukazatele tvrdého což značí

přesně identifikované výsledky a měkkého ukazatele určující spokojenost.

5) Návratnost investic – určuje výsledek mezi investicemi do výuky a zvýšením celkových příjmů.

4.7 Specifikace přínosů e-learningu v českých organizacích

Zákazník	Snížení nákladů LMS + obsah	Zvýšení příjmů + další výhody
Česká pojišťovna	- eliminace administrace spojené s organizací a provozem kurzů - značné finanční a časové úspory a to i se započítáním počátečních nákladů spojených se zavedením e-learningu	- podstatné zkrácení doby na vyškolení pracovníků v nových produktech - konkurenční výhoda na trhu - image progresivního inovátora
České dráhy	- minimalizace potřeby dojíždění na školení	- zvýšení počítačové gramotnosti zaměstnanců - zvýšení sebevědomí a schopnosti přijímat změny
Český telecom	- úspora finančních prostředků (až 60%) - možnost transferu financí na projekty s vysokou přidanou hodnotou	- kladný vliv na výkon společnosti díky zrychleným reakcím na vzdělávací potřeby - změna firemní kultury směrem k učící se organizaci - posílení image jako významného hráče na poli zaměstnavatelů
McDonald's	- vyřešení problému každodenního stárnutí vzdělávacích materiálů	- atraktivita výuky - možnost sledování studijních výsledků - rychlejší pochopení probírané látky

1. Tabulka - Specifikace přínosu e-learningu v českých organizacích (3)

4.8 Budoucnost e-learningu

Dále se díky rozvíjejícím digitálním technologiím bude elektronická výuka rozšiřovat a její možnosti porostou. Elektronická výuka se bude snažit maximálně uzpůsobit životnímu programu lidí. Výuka nebude mít vliv na prostředí, ani na časový harmonogram. Bude se moci vzdělávat více lidí a překážky jako cestování budou z převážné části odstraněny. Výuka se tedy více přesune do virtuálních tříd.

Program výuky se bude z velké části snažit zprostředkovat pomocí elektronické výuky. Výuka dále může probíhat ve stylu, kdy studenti dostanou do své e-mailové schránky zprávu, kterou látku by si měli připravit a naučit. Dále zpráva už bude jen obsahovat datum a čas kdy pomocí videokonference bude možnost se dotázat na nejasné věci a popřípadě řešit podrobnější vysvětlení. V nějakých termínech se budou studenti dostavovat do fyzických tříd, kde budou studovat látku, u které není vhodné využití elektronické výuky. Ale i ve fyzických třídách se bude maximálně využívat výpočetní techniky.

Aby se nezdálo, že studenti budou jen zavření ve virtuálních třídách ve své výuce. V jedné části se snaží i zpracovávat výuku v místech pro jednotlivé předměty maximálně užitečné, pro výuku přírodovědy navštívení zoologické zahrady a pomocí digitálních fotoaparátů zaznamenávání jednotlivých druhů zvířat a v dalších krocích už ve společné třídě jejich promítání a výkladu látky.

Nové způsoby výuky se budou také snažit více poměřovat růst vzdělání studentů a jejich možné porovnávání v čase.

Elektronická výuka bude se snažit učitelům dát více času na učení než prací okolo, kdy vykládali pořád dokola stejnou látku, opravovali písemky.

Velký podíl na vývoji elektronické výuky vidím, že obrovské softwarové společnosti jako Microsoft a Google mají ve svém portfoliu oddělení zabývající se elektronickou výukou.

5. Vývoj aplikace

5.1 Výběr programovacího jazyka

Pro tvorbu webové aplikace e-learningu jsem si vybral programovací jazyk ASP .NET s podporou C#. Tento programovací jazyk jsem si vybral z důvodu jeho kvality a také se mi tento programovací jazyk zalíbil nejvíce. Hlavním důvodem bylo oddělení vzhledu stránek od samotného programovacího jazyka, v mém případě C#. Jeden z dalších důvodů bylo využití C#, který je využíván jako základ pro další nástavby. Pomocí jazyka C# lze programovat webové aplikace (ASP .NET, ASP .NET MVC), animované webové aplikace (Silverlight), dále mobilní aplikace pro chytré telefony s Windows Mobile, desktopové aplikace (WinForm, WPF), cloud aplikace, serverové služby (WCF), XNA Frameworku pro tvorbu her (PC, XBOX360 a Zune) a další.

Zajímavá vlastnost ASP .NET a jeho velká výhoda proti jiným jazykům pro webové aplikace je možnost psát aplikační kód v různých jazycích podporujících .NET Framework mezi tyto jazyky patří (C#, Visual basic net, upravené jazyky jako PHP, Delphi, Ruby, C++). Je pak možnost rozsáhlou aplikaci psát v různých jazycích kdy jeden programátor píše ve Visual basicu a druhý v C#.

Mezi důležité prvky bych také zmínil širokou podporu ze strany výrobce tohoto programovacího jazyka, který se o jazyk stará a pravidelně se snaží jazyk vylepšit a rozšířit.

Důležitá je také velká komunita okolo tohoto jazyka, která pořádá mnohá školení a vytváří tutoriály. Mezi poslední body patří i kvalita webového aplikačního serveru IIS 7, pod kterým ASP. NET aplikace běží.

Pro ukládání dat jsem si vybral relační databázi Microsoft SQL Server 2008. Databáze je na vysoké úrovni a dle mého hlediska pro mé využití lepší než databáze od Oracle. Pro výběr rozhodnuli určitě stabilita, rychlost, bezpečnost databáze.

5.2 Výběr vývojových nástrojů

Pro vývoj aplikace jsem si vybral Microsoft Visual studio 2010 v beta verzi. Visual Studio je produktem firmy Microsoft a je to hlavní vývojový nástroj pro C#, ASP .NET, Visual basic, F# a další jazyky.

Pro práci s databází jsem využíval SQL Server management studio dodávané zdarma k Microsoft SQL Serveru 2008.

Jak už jsem psal, webovou aplikaci jsem naprogramoval v jazyku ASP. NET ve verzi 3 s programovacím jazykem C# pro NET Framework 3.5 připojenou na relační databázi běžící na Microsoft SQL server 2008 SP1.

5.3 Struktura aplikace

Aplikace je tvořená hlavní stránkou tzv. MasterPage, kterou využívají všechny stránky mimo login.aspx. MasterPage obsahuje informace o přihlášeném uživateli a možnost se odhlásit.

- Predmety.aspx – Stránka vypisující seznam předmětů, ke kterým má student přístup.
- Kapitoly.aspx – Vypíše seznam kapitol vybraného předmětu.
- Lekce.aspx – Stránka zobrazující látku vybrané lekce.
- Login.aspx – Stránka sloužící k přihlašování do aplikace.
- Ukoly.aspx – Stránka generující zadání úkolů k vybrané kapitole.
- Admin/prava.aspx – Zde se nastavují práva pro přístup studentů do jednotlivých předmětů.
- Admin/novyuzivatel.aspx – Možnost vytvořit nového uživatele
- Admin/prehledukolu.aspx – Stránka zobrazující zadané úkoly studentům.
- Temp/ - Adresář pro možnosti dočasného uložení souborů.
- Upload/ - Adresář pro ukládání obrázků.
- Web.config – Soubor s konfigurací aplikace.
- Admin/web.config – Soubor k změně konfigurace aplikace uložené v části Admin.

5.4 Zabezpečení aplikace

Pro přístup k aplikaci je potřeba se přihlásit. Pro přihlašování jsem nepoužil defaultního asp .net providera, ale využil Altairis web providera. Bez přihlášení se uživatelé nedostanou na žádnou stránku mimo stránky login.aspx, na kterou budou ze všech ostatních přesměrovány. Do části Admin je možnost přístupu jen pro uživatele, kteří mají nastavenou roli „Admin“.

5.5 Ukázky kódu aplikace

Jako ukázky kódu webové aplikace jsem vybral následující funkci. Funkce slouží k nastavování práv uživatelům. Funkce se spouští při vzniku události RowCommand (změně řádku). Funkce se nachází v souboru Admin/prava.aspx.cs. Tělo funkce se provádí jen při nastavování práv. Funkce dále se připojí k databázi a provede SQL příkaz na vložení řádku do databáze, kde do tabulky UsersInPredmet vloží hodnotu vybraného předmětu a jméno uživatele, kterému se nastavují práva. Na konci se ukončí připojení do databáze.

```
protected void GridView1_RowCommand(object sender, GridViewCommandEventArgs e)
   {
      if (e.CommandName == "NastavPrava")
      {
          const string sqlConnectionString = "server=ipadresa;
uid=login;pwd=heslo;database=nazevdatabaze;";
          SqlConnection conn = new SqlConnection(sqlConnectionString);
          conn.Open();
          SqlCommand cmd = new SqlCommand("insert into Usersinpredmet
(predmet,username) values (@predmet,@username)", conn);
          cmd.Parameters.AddWithValue("@predmet", DropDownList1.SelectedValue);
```

```
            cmd.Parameters.AddWithValue("@username",
e.CommandArgument.ToString());

            SqlDataReader rdr = cmd.ExecuteReader();

            rdr.Close();

            conn.Close();

        }

    }
```

1. Zdrojový kód - ukázka funkce GridView1_RowCommand

5.6 Grafický vzhled aplikace - témata

Pro grafický vzhled bylo využito témat programovacího jazyka ASP .NET, která zjednodušují případné změny a grafické návrhy aplikace. Téma je vlastně adresář obsahující nastavení vzhledu aplikace. Změna vzhledu aplikace je pak velmi jednoduchá změnou jednoho parametru, názvu používaného tématu. Každé téma může obsahovat více souborů pro konfiguraci vzhledu, ať to jsou css styly, skinfile nebo jiné soubory využívané pro nastavení vzhledu (obrázky, animace) a při změně tématu se nastaví všechny odkazy na nové skinfile a css soubory. Pro moji aplikaci jsem vytvořil jedno grafické téma.

5.6.1 Ukázka kódu v skin souboru

Skinfile soubor slouží k nastavování vzhledu asp komponent. Vzhled lze nastavit přímo nebo odkazem na css styly. Na následující ukázce kódu je nastavení css stylů pro GridView, protože jsem nenastavil SkinId bude nastavení platit pro všechny GridView v mé aplikaci. V případě že bych chtěl nastavit vzhled jen pro jeden nebo určitý počet GridView musel bych nastavit SkinId, což stačí připsat jako další parametr ve tvaru SkinId=“nazev“. Po nastavení SkinId se u GridView, které mají využívat určitý vzhled, napíše stejný

parametr SkinId=“nazev“. Ukázka dále ukazuje nastavení střídání různých Css stylů GridViewu pro lichý a sudý řádek tabulky.

```
<asp:GridView runat="server" GridLines="None" CssClass="grid">
    <RowStyle CssClass="r0" />
    <AlternatingRowStyle CssClass="r1" />
</asp:GridView>
```

2. Zdrojový kód - ukázka Skinfile souboru

5.6.2 Ukázka css souboru

Jako druhý soubor obsahuje téma Css soubor s konfigurací vzhledu aplikace.

```
.grid
{
    border-collapse: collapse;
    border: solid 1px #808080;
}
.grid th
{
    background-color: #990000;
    color: White;
    font-weight: bold;
}
.grid th, .grid td
{
    padding: 3px 10px;
}
```

```
.grid .r0
{
    background-color: #FFFF99;
}
.grid .r1
{
    background-color: white;
}
```

3. Zdrojový kód – část CSS souboru

6. Relační databáze

6.1 Návrh tabulek

Moje databáze obsahuje 9 tabulek. E-R diagram tabulek naleznete v příloze A (obrázek 12.). U tabulek PrehledUkoly, Ukoly, UsersInRoles, Users, UsersInPredmet, Lekce jsou nastaveny primární klíče typu int a mají nastaveny sekvence, které jim při každém novém vložení řádku zvětší hodnotu o 1 proti předešlému.

6.2 SQL dotazy z aplikace

Na následující ukázce zdrojového kódu je SQL příkaz na mazání řádků v tabulce UsersInPredmet a SQL dotaz na vypsání uživatelů a informace jestli mají práva pro přístup do vybraného předmětu. Dále v ukázce jsou DeleteParameters, které využívám z důvodu SQL injection a jeho ošetření.

```
DeleteCommand="DELETE FROM [UsersInPredmet] WHERE username=@username"
```

```
 SelectCommand="select username, CASE WHEN (select count(username) from UsersInPredmet where UserName=Users.UserName) and (predmet='" + DropDownList1.SelectedValue + "'))>=1 THEN 'Aktivní' ELSE 'Deaktivní' END AS 'prava' from users">
        <DeleteParameters>
            <asp:Parameter Name="username" Type="String" />
        </DeleteParameters>
```

4. Zdrojový kód – ukázka sql dotazů z aplikace

SQL příkaz na vložení záznamu do databáze o zadání úkolu.

```
insert into PrehledUkoly (datum,id_ukolu,userName) values (@datum,@id,@user)
```

5. Zdrojový kód – SQL příkaz vložení řádku

SQL příkaz na aktualizaci záznamu v tabulce Ukoly

```
update Ukoly set pocet_zadani = @hodnota where id_ukolu= @id
```

6. Zdrojový kód – SQL příkaz na aktualizaci záznamu

7. Instalace webové aplikace

Aplikace na svůj chod není náročná a potřebuje minimum prostředků. Aplikace se instaluje na aplikační server. Pro svůj chod aplikace potřebuje IIS 6 nebo IIS 7 s nainstalovaným NET Framework 3.5, toho lze docílit na serverových řešeních Microsoft Windows 2003, Windows 2008 nebo Windows 2008 R2, popřípadě na klientských operačních systémech Windows XP, Windows Vista a Windows 7, kde lze taktéž nainstalovat IIS (pozor, ne všechny edice mají možnost instalace IIS).

Po nainstalování IIS je potřeba základní konfigurace od správce sítě. IIS nabízí mnoho parametrů pro konfiguraci, jako nastavení IP adresy (pod kterou bude webový server přístupný), nastavit cestu ke kořeni www adresáře, na-

stavení aliasů, chybových hlášek, přístupy pro jednotlivé uživatele, povolování zápisu do adresářů, přesměrovávání a mnoho dalšího.

Po konfiguraci IIS bude potřeba nahrát obsah adresáře www do wwwrootu webového serveru nebo popřípadě jinam podle konfigurace IIS. Dále bude potřeba povolit aplikaci zápis do složek Temp a Upload.

Další krok bude nainstalovat Microsoft SQL server 2008, kde bude stačit express verze, která je zdarma. Instalaci provází jednoduchý průvodce. Po nainstalování bude potřeba vytvořit databázi a nastavit aplikaci přístup do této databáze pomocí nově vytvořeného uživatele.

Po vytvoření databáze ještě bude potřeba vytvořit pomocí SQL jazyka tabulky a naplnit je potřebnými daty. SQL skript je v digitální podobě na přiloženém CD, v souboru script.sql, tento skript bude stačit spustit jako SQL dotazy a vše potřebné se vytvoří.

Po nainstalování všeho potřebného bude ještě nutné nastavit connection string k databázi. Tento řetězec nastavíme v souboru web.config v kořenovým adresáři aplikace. Název tohoto řetězce musí být „VyukaConnectionString“.

Ukázka Connection stringu v konfiguračním souboru web.config

```
<connectionStrings>
        <clear/>
<add name="VyukaConnectionString" providerName="System.Data.SqlClient" connectionString="server=IPAdresa;uid=jmenoUzivatele;pwd=heslo;database=nazevDB"/>
        </connectionStrings>
```

7. Zdrojový kód – ConnectionStrings z web.config

Do uvedeného connection stringu je potřeba nastavit potřebná data. A tedy do proměnné server místo IPAdresa nastavit IP adresu serveru na kterém běží Microsoft SQL server. UID je jméno uživatele, pod kterým se aplikace bude přihlašovat do databáze, tak zde nastavíme tedy jméno uživatele, kte-

rého jsme pro potřebné účely vytvořili. Místo nazevDB se nastaví název databáze vytvořené pro účely aplikace a heslo nahradíme správným heslem pro přístup k aplikaci.

Příkaz <clear/> značí, aby se pro aplikaci nevyužívali connection stringy z nadřazených web.config souborů.

Pro přístup k aplikaci je potřeba webového prohlížeče a nainstalovaného NET Frameworku 3.5 (na linuxových distribucích lze využít Open Source implementace .NET Frameworku pod názvem Mono vyvíjeného firmou Novell). Nic víc na klientské stanici není potřeba.

7.1 Přístup do administrátorské části

Do administrátorské části lze přistupovat /Admin/default.aspx. Do této části mají přístup jen uživatelé s nastavenou rolí přístupu admin. V této části lze nastavovat práva pro přístup uživatelů do různých předmětů, dále možnost vytvářet nové uživatele, opravňovat uživatele rolí administrator a nakonec přehled zadaných úkolů studentům.

Defaultní přístup má uživatel „admin“ s heslem „Admin123“.

8. Vypracování studijních materiálů

Studijní materiály pro předmět Základy algoritmizace jsem rozdělil do základních kapitol. První tři kapitoly jsou spíše teoretické a snaží se popsat co to vlastně algoritmus je a jaké jsou jeho možnosti zápisu. Další kapitoly už se snaží dodržet svého uspořádání obsahující vzorové řešení daného algoritmu zobrazeného vývojovým diagramem, dále zápisem v jazyce Java a zadáním domácích úkolů.

8.1 Seznam vybraných kapitol

- Základní terminologie
- Možnosti zápisu algoritmu
- Značky vývojových diagramů
- Sekvence
- Větvení
- Cykly
- Maximum z posloupnosti čísel
- Práce s maticemi – načítání matice
- Práce s maticemi typu (m,n)
- Práce se čtvercovými maticemi
- Operace s maticemi

8.2 Základní terminologie

Algoritmus:

Algoritmus je přesný postup, který je potřeba k vykonání určité činnosti a splňuje tyto podmínky:

- Má začátek a konec,
- je jednoznačný,
- srozumitelný,
- opakovatelný,
- věcně správný,
- obecný.

Začátek a konec - To znamená, že se program musí vždy ukončit a nemůže například zůstat zacyklen s nemožností konce.

Jednoznačnost - Tady se jedná o zapsání všech potřebných podmínek a jejich možných řešení. Podmínkami ošetřit co se stane v určitých případech.

Srozumitelnost - Byl natolik srozumitelný, aby i jiný člověk byl schopen tento algoritmus pochopit.

Opakovatelnost - Možnost kdykoliv algoritmus zopakovat, a při stejných podmínkách se bude chovat stejně.

Věcná správnost - U zadání vzorců musíme být pozorní, abychom vzorec zadali správně a dodrželi správné zapsání pomocí závorek.

Obecnost - V zápisu algoritmu nejsou využity přímo specifikované parametry, algoritmus se dá využít pro více příkladů.

8.3 Možnosti zápisu algoritmů

1) Slovní vyjádření
2) Matematický zápis
3) Rozhodovací tabulky
4) Vývojové diagramy
5) Počítačové programy

Slovní vyjádření

Jsou určený pro lidi, kteří nemají programátorské vzdělání a jiné vyjádření by pro ně bylo nemyslitelné (různé návody). Dále se využívají pro komunikaci mezi programátory a laiky.

Je to tedy forma vyjádření algoritmu pro komunikaci s laikem, s kterou se dá vždy domluvit. Velkými nevýhodami je, že toto vyjádření je velmi nepřehledné a nepřesné.

Matematický zápis

Nelze využít vždy, ale jen u zápisů umožňujících zápis matematickými vztahy.

Hlavní výhodou tohoto zápisu je jeho jednoznačnost, ale není moc podrobný.

Příklad:

Pythagorova věta – a^2 +b^2 = c^2

Rozhodovací tabulky

Vhodná pro využití kdy se vyskytuje několik možností a řešení je pro každou možnost jednoduše popsatelné.

Například lze využít pro školní rozvrh ve škole.

Vývojové diagramy

Vývojový diagram je symbolický algoritmický jazyk, který názorně zobrazí algoritmus.

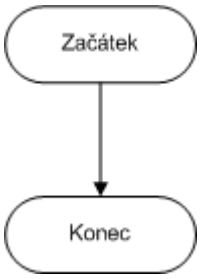
Slouží jako komunikační prostředek a k dokumentaci.

Diagramy se tvoří pomocí jednotlivých symbolů. Tyto symboly jsou mezi sebou spojeny orientovanými čárami.

8.4 Značky vývojových diagramů

Mezní značky

Používají se pro vstup do programu nebo výstup z programu. Tedy označují začátek a konec programu.



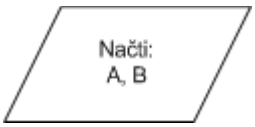
1. Obrázek – mezní značky

Sekvenční bloky

Vyskytují se uvnitř diagramu. Ukazují sekvenční postup algoritmem.

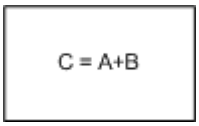
Používají se dva typy sekvenčních bloků:

Vstupně/výstupní sloužící pro komunikaci s uživatelem. Vstup načítá data a výstup data zobrazuje (obrazovka, tiskárna atd..)



2. Obrázek – značka vstupně/výstupního bloku

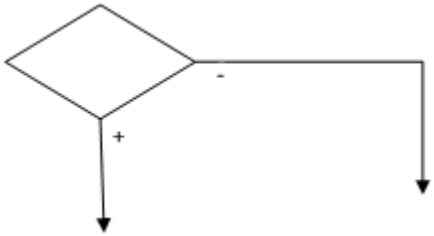
Zpracování zastupuje činnost programu. V níž jsou uvedeny příkazy, které se vykonají.



3. Obrázek – značka zpracování

Větvení

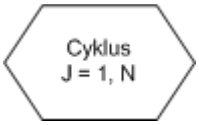
Používají se k rozvětvení algoritmu. K větvení dochází za nějaké podmínky. Dále program pokračuje podle toho, jestli je podmínka splněna nebo nesplněna.



4. Obrázek – značka větvení

Příprava

Používá se k značení přípravné fáze programu (tedy inicializace).



5. Obrázek – značka přípravy

Spojka

Využívá se k spojení dvou částí vývojové diagramu. Číslo udává začátek a konec stejné spojky.

6. Obrázek – značka spojky

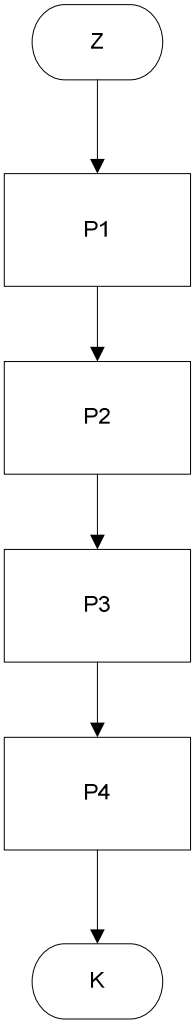
Podprogram

Část programu, která se využívá ve více programech nebo v jednom programu vícekrát, se vytvoří jako podprogram.

7. Obrázek – značka podprogramu

8.5 Sekvence

Je posloupnost příkazů, které se postupně provádí.



8. Obrázek - Ukázka sekvence

Popisek k ukázce sekvence

Na znázorněném diagramu je ukázka posloupnosti příkazu. Začínající značkou začátku algoritmu a pokračující 4mi příkazy. Konec algoritmu zakončený značkou konce.

Příklad v jazyce Java

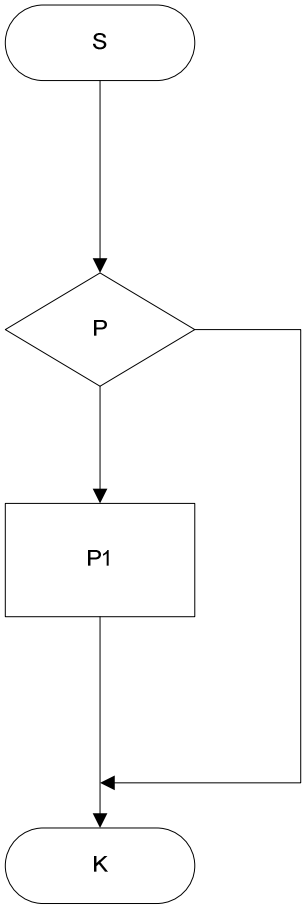
```
public class Vyuka {
    public static void main(String[] args) {
        P1();
        P2();
        P3();
        P4();
        P5();
      }
}
```

8. Zdrojový kód – ukázky sekvence v jazyce Java

8.6 Větvení

8.6.1 Větvení s prázdnou akcí

Větvení, které jen při splnění podmínky provede určenou sekvenci příkazů (popřípadě jen jednoho příkazu), naopak při nesplnění podmínky neprovede nic. Využitelné v mnoha případech, kdy potřebujeme provést nějaký blok příkazů jen při splnění určité podmínky.



9. Obrázek – větvení s prázdnou akcí

Ukázka algoritmu větvení v programovacím jazyce Java.

```
public class Vyuka {
    public static void main(String[] args) { //main metoda projektu
        boolean podminka; // deklarovaní proměnné podmínka typu boolean
        podminka = true; //nastavení proměnné na hodnotu true
   if (podminka == true) //podmínka zjištující jestli se podmínka rovná true
{ // blok kódu, který se provede při splnění podmínky
    System.out.println("Podmínka byla splněna!");//výpis do příkazové řádky
   }
 }
}
```

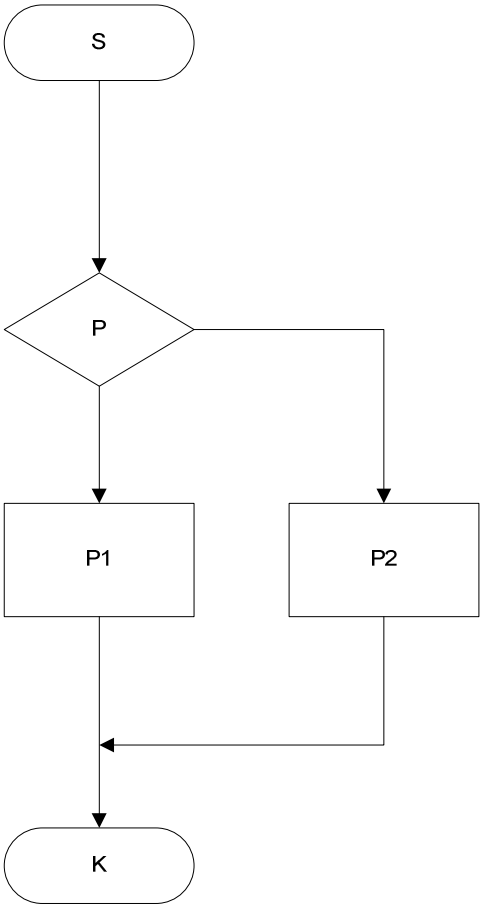
9. Zdrojový kód – ukázka algoritmu větvení s prázdnou akcí

8.6.2 Větvení s dvěma výsledky

Na rozdíl od větvení s prázdnou akcí se u tohoto větvení i při nesplnění podmínky provede blok příkazů. Využitelné v případech, kdy je potřeba provést určitou akci i při nesplnění podmínky.

Například:

Když bude hodnota menší jak 4, zapiš do výsledku „prospěl“, v případě nepravdy zapiš do výsledku hodnotu „neprospěl“.



10. Obrázek – větvení s dvěma výsledky

Obecný kód v Javě:

```
public class Vyuka {
    public static void main(String[] args) {
    if (5==5)
    {
```

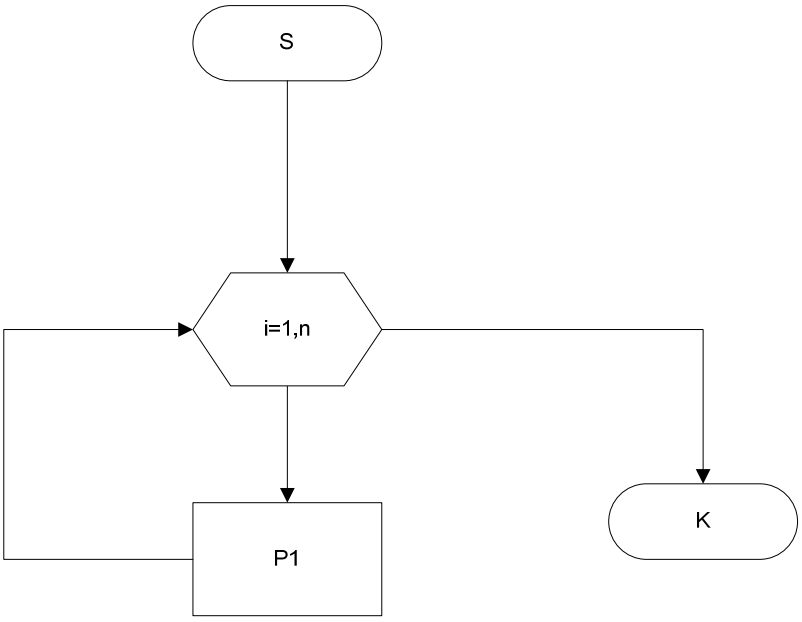
```
      P1();
   }
   else
   {
      P2();
    }
 }
}
```

10. Zdrojový kód – větvení s dvěma výsledky

8.7 Cykly

Se známým počtem cyklů – for

Využívá se v případě, kdy je potřeba nějaký určitý blok sekvencí opakovat v určitém počtu opakování.



11. Obrázek - cykly

```
public class Vyuka {
   public static void main(String[] args) {
```

```
    for (int j=1 ;j<n ;j++ )  {
        P1();
    }
  }
}
```

11. Zdrojový kód – ukázka cyklů v jazyce Java

8.8 Maximum z řady čísel

Algoritmus pro určení minima z řady čísel není složitý na tvorbu, užívá se zde cyklů pro načtení a třídění dat. Na začátku algoritmu si inicializujeme proměnnou Maximum a nastavíme ji na neutrální hodnotu nula. V dalším kroku načteme hodnotu počtu prvků posloupnosti. Nyní vytvoříme cyklus z důvodu pořád se opakujících kroků:

- Načti novou hodnotu.
- Porovnej číslo s maximem.
    - V případě, že číslo bude větší, než aktuální maximum stane se maximem.
    - V opačném případě bude ponecháno původní maximum.
- Tento cyklus opakujeme tolikrát, kolik má posloupnost hodnot.

Na konci algoritmu vypíšeme maximální hodnotu posloupnosti.

Vývojový diagram algoritmu nalezneme v příloze E (obrázek 16.).

Příklad v programovacím jazyce Java

```
public class Vyuka {
   public static void main(String[] args) {
double maximum = 0;
double Y;
Scanner sc= new Scanner(System.in);
```

```
Y = sc.nextDouble();
double x;
    for (int j=0 ;j<Y ;j++ )  {
     x=  sc.nextDouble();
              if (x > maximum)
              {
                    Maximum = x;
              };
    }
Systém.out.print(Maximum);
  }
}
```

12. Zdrojový kód – maximum z řady čísel

8.9 Práce s maticemi

Načtení prvků do matice je potřeba pro všechny algoritmy, kde se využívá práce s algoritmy.

Existují dvě možnosti načtení matice:

1) zleva doprava,
2) odshora dolů.

Budeme-li načítat první řádek, můžeme využít cyklus, který bude načítat postupně prvky zleva doprava, pomocí měnění druhého indexu od 1 do n. Po načtení celého řádku změníme první index, kterým začneme číst druhý řádek. Ve výsledku bude algoritmus tvořený dvěma cykly vnořenými do sebe.

Vývojový diagram je uložen v příloze F (obrázek 17.)

Příklad v jazyce Java:

```
public class Vyuka {
    public static void main(String[] args) {
```

```
int m=3;
int n=5;
double [][] matice;
Scanner sc= new Scanner(System.in);
    for (int i=1 ;i<=m ;i++ )  {
        for(int = j=1;j<=n;j++) {
          matice[i][j]=sc.nextDouble();
       }
    }
  }
}
```

13. Zdrojový kód – načtení matice

8.10 Práce s prvky matice typu (m,n)

Práce s prvky matice probíhají převážně pomocí cyklů, protože každá matice se musí procházet prvek po prvku, kde se mění vždy číslo řádku a číslo sloupce. Tím se dospěje k výsledku, že při práci budeme muset vytvořit dva cykly, jeden pro procházení řádků a druhý řádek pro průchod sloupci.

Pro příklad si vybereme zjištění největšího a nejmenšího prvku v matici.

V příkladě využijeme načítání prvků v průběhu. V tomto příkladu využijeme příklad načtení matice z minulé lekce a rozšíříme ji pro své účely. První změna se hned dotkne začátku, kde při načítání hodnot načteme navíc ještě první hodnotu matice a tuto hodnotu uložíme do nově vytvořených proměnných Min a Max. Což provedlo prvotní inicializaci maxima a minima matice. V dalším kroku budou nastaveny cykly na své hodnoty, kde se nic nemění proti předešlému příkladu. Další změna nás čeká po načtení prvku z matice. Kde nyní načtenou hodnotu budeme porovnávat s aktuálním minimem a maximem matice. V případě, že bude hodnota menší, než minimum bude nastavena jako minimum. V opačném případě když hodnota bude větší, jak

maximum bude nastavena jako maximum. Na konci algoritmu ještě přidáme zobrazení výsledků, tedy zobrazení hodnoty nejmenšího prvku matice a největšího prvku matice.

Vývojový diagram algoritmu je uveden v příloze B (obrázek 13.).

Příklad v jazyce Java:

```
public class Vyuka {
   public static void main(String[] args) {
Scanner sc= new Scanner(System.in);
int m=3;
int n=5;
double [][] matice;
double max, min;
matice[i][j]=sc.nextDouble();
min = matice[i][j];
max = matice[i][j];
    for (int i=1 ;i<=m ;i++ )  {
        for(int = j=1;j<=n;j++) {
          matice[i][j]=sc.nextDouble();
          if matice[i][j]<min     { min= matice[i][j];};
          if matice[i][j]>max    { max= matice[i][j];};
       }
    }
   System.Out.Printl(min);
   System.Out.Printl(max);
  }
}
```

14. Zdrojový kód – určení největší a nejmenšího prvku matice

8.11 Práce s maticemi čtvercového typu

V práci s maticemi typu (m, m) tedy čtvercového typu se moc neliší způsob práce proti maticím typu m, n.

Příklad čtvercové matici si předvedeme na algoritmu zjišťující, jestli je matice diagonální. Jako v předešlém případě budeme jednotlivé prvky matice načítat ze vstupu.

Pro vysvětlení diagonální matice je taková matice, která je čtvercová a u které se všechny prvky hlavní diagonály nesmí rovnat nule a všechny prvky pod a nad hlavní diagonálou musí rovnat nule.

Vysvětlení algoritmu:

Algoritmus na začátku si načte hodnoty rozměrů matice. V dalším kroku vyhodnotí pomocí podmínky, jestli je matice čtvercová, v případě že ano bude program pokračovat, jinak se program ukončí s textem, že matice není čtvercová. Dále vytvoříme proměnnou Diagonal, kterou nastavíme na hodnotu true (true značí, že matice je diagonálního typu). Nyní vytvoříme dva cykly na průchod maticí. Jeden cyklus na průchod řádky a druhý na sloupce. Nyní v cyklech už čteme prvky matice a budeme jednotlivé prvky testovat podmínkami. První podmínka určuje, zda prvek leží na hlavní diagonále, což ověříme tím, že porovnáme, jestli se I a J rovnají. V případě, že prvek leží na hlavní diagonále, prověříme ho dalším testem, jestli je různý od nuly, jestli ano pokračujeme dalším prvkem, jinak nastavíme hodnotu proměnné Diagonal na false (čímž matici označíme jako nediagonální). Jestli prvek neleží na hlavní diagonále, prověříme ho druhým testem a to jestli se rovná nule. V kladném výsledku pokračujeme v cyklu, v záporném nastavíme proměnou Diagonal na hodnotu false. Po průchodu maticí pomocí cyklů narazí algoritmus na poslední podmínku. Tato podmínka prověří hodnotu Diagonal a podle jejího výsledku zobrazí na výstup, jestli je matice diagonální nebo není.

Vývojový diagram uveden v příloze C (obrázek 14.).

Příklad v jazyce Java:

```
public class Vyuka {
public static void main(String[] args) {
Scanner sc= new Scanner(System.in);
int m, n;
double [][] matice;
boolean Diagonal = true;
m = sc.nextDouble();
n = sc.nextDouble();
if (m = n) {
    for (int i=1 ;i<=m ;i++ )  {
        for(int j=1;j<=n;j++) {
          matice[i][j]=sc.nextDouble();
          if (i = j)  {
           if (matice[i][j] = 0) { Diagonal = false;};
} else {
 if (matice[i][j] != 0) { Diagonal = false;};
 };
       }
    }
   If (Diagonal = true)
   {
    System.Out.Printl(„Je diagonální“);
   }
   Else
   {
     System.Out.Printl(„Není diagonální“);
```

```
    };
   } else { System.Out.Printl(„Matice není čtvercová“);};
  }
}
```

15. Zdrojový kód – určení diagonální matice

8.12 Operace s maticemi

Mezi základní operace s maticemi patří sčítání, rozdíl a násobení matic.

Jako ukázku práce s maticemi si ukážeme součet dvou matic. Jedna z podmínek sčítání matic je stejný počet sloupců a řádků první matice i u druhé matice.

Postup algoritmem

Pro začátek si načteme rozměry obou maticí. V dalším kroku zjistíme podmínkou, jestli mají matice stejné rozměry. V kladném případě načteme první matici ze vstupu. Další krok bude načítat druhou matici, ale pro zrychlení při každém načteném prvku. Sečte načtený prvek s prvkem odpovídající pozice matice A a zapíše výsledek do matice C. Během jednoho cyklu ještě navíc zobrazí výsledek matice C odpovídajícího prvku na výstup.

Vývojový diagram uveden v příloze D (obrázek 15.).

Příklad v jazyce Java

```
public class Vyuka {
public static void main(String[] args) {
Scanner sc= new Scanner(System.in);
int m, n, q, r;
double [][] maticeA, maticeB, maticeC;
m = sc.nextDouble();
n = sc.nextDouble();
```

```
q = sc.nextDouble();
r = sc.nextDouble();
if ((m = q) and (n = r)) {
    for (int i=1 ;i<=m ;i++ )  {
        for(int j=1;j<=n;j++) {
          maticeA[i][j]=sc.nextDouble();
       }
    }
    for(i=1 ;i<=m ;i++ )  {
        for(j=1;j<=n;j++) {
          maticeB[i][j]=sc.nextDouble();
          maticeC[i][j]= maticeA[i][j]+maticeB[i][j];
          System.Out.Printl(maticeC[i][j]);
       }
    }
   } else { System.Out.Printl(„Matice nemají stejný rozměr“);};
  }
}
```

16. Zdrojový kód – součet matic

9. Závěr

Ve své práci jsem se zaměřil na vysvětlení pojmů z oblasti e-learningu a k němu navazujícímu blended learningu. Snažil jsem se vysvětlit, co tento pojem vystihuje, jak se dělí a jak se blended learning měří.

V praktické části jsem měl za úkol vytvořit webovou aplikaci pro možnosti výuky a do této aplikace vypracovat studijní materiály předmětu Základy algoritmizace.

S tvorbou webové aplikace jsem spokojen, vím, že aplikace by v nějakých ohledech mohla být lepší, převážně z grafického hlediska.

Za nedostatky mé aplikace vidím neintegrování wysiwyg editoru z důvodu nemožnosti zprovoznění těchto editorů ve volně šiřitelných verzích. V případě placené verze možnosti editoru byly funkční, ale tyto verze byly jen ve zkušebních verzích.

Práce byla pro mě přínosná, když jsem si dále rozšířil své znalosti v jazyce ASP .NET a narazil na nové problémy při psaní webových aplikací.

10. Použitá literatura

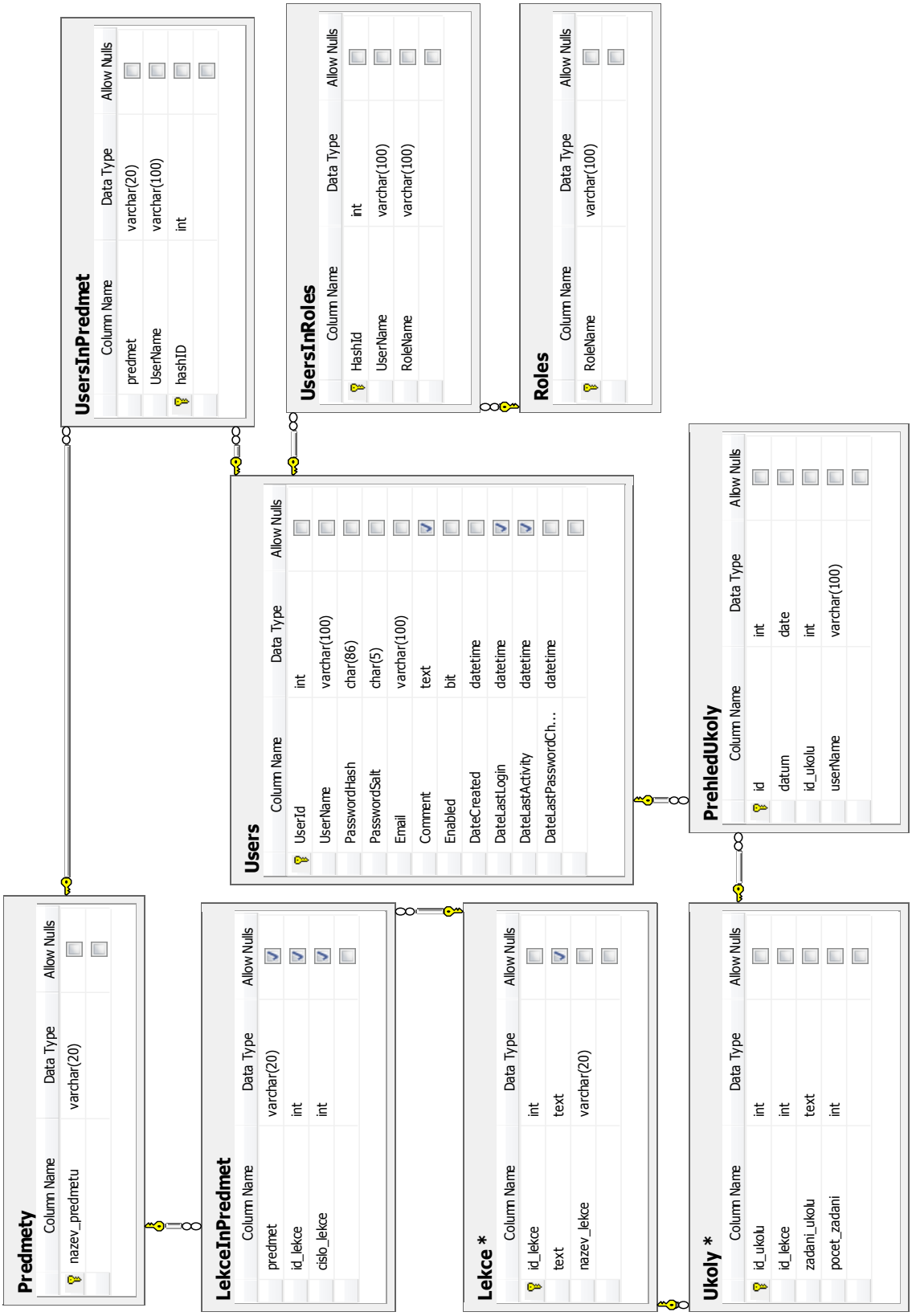
[1] PŠENČÍKOVÁ, Jana. *Algoritmizace*, Computer Media, 2007, 120 s.

[2] TOPFER, Pavel. Algoritmy a programovací techniky, Prometheus, 1995, 299 s.

[3] PEJŠA, Jan. *E-learning - trendy, měření efektivity, ROI, případové studie* [online]. [cit. 2009-08-06]. Dostupný z WWW: <http://www.e-learn.cz/soubory/e-learning_trends_ROI.pdf>.

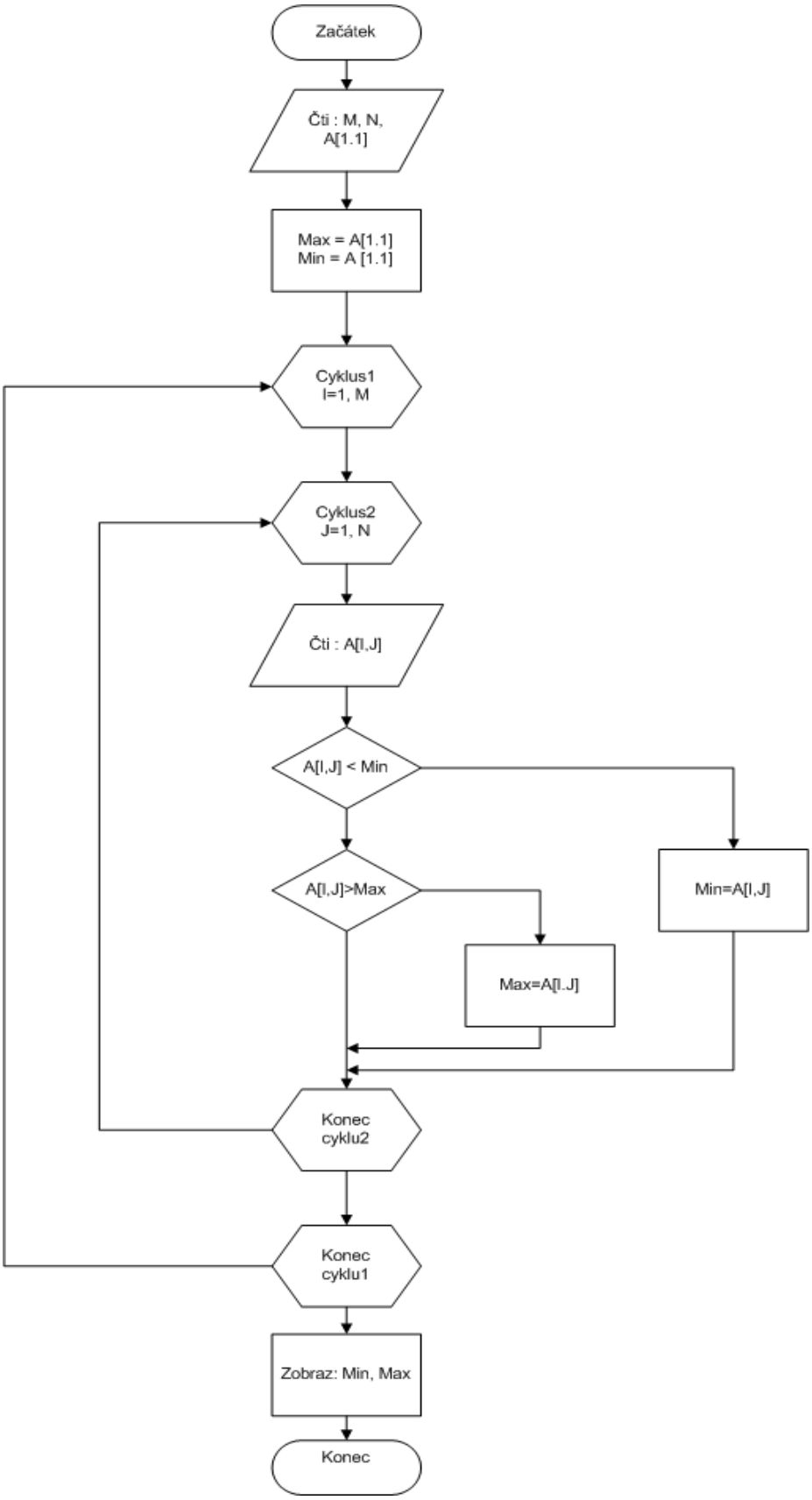
[4] KOPECKÝ, Kamil. *Blended learning jako skutečně efektivní přístup ke vzdělávání* [online]. 2007 [cit. 2009-08-03]. Dostupný z WWW: <http://158.194.48.95/cestina/kopecky//index.php?option=com_content&task=view&id=131&Itemid=7>.

[5] *Proč e-Learning?* [online]. 2005 [cit. 2009-08-08]. Dostupný z WWW: <http://www.e-learn.cz/uvod_proc.asp?menu=elearning&submenu=proc>.



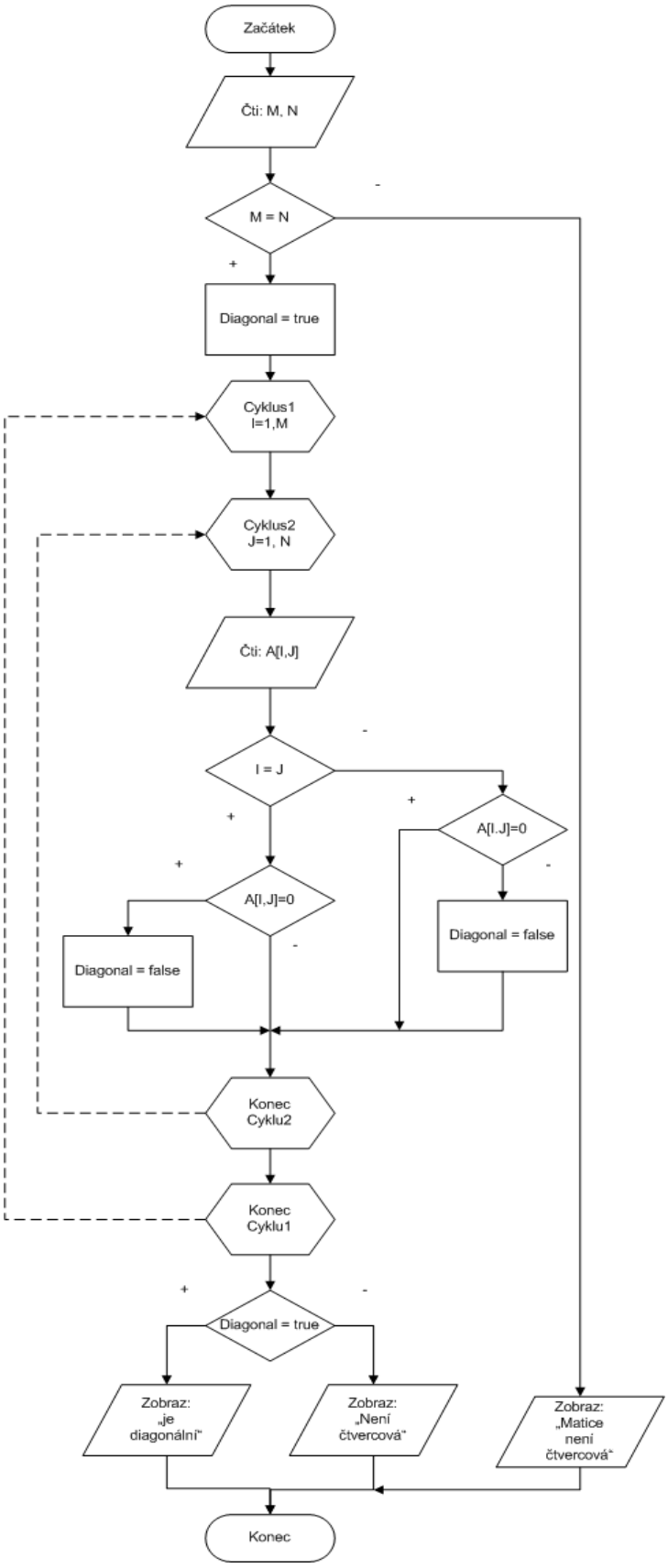
12. Obrázek – E-R diagram

Příloha B – diagram výběru největšího a nejmenšího prvku matice



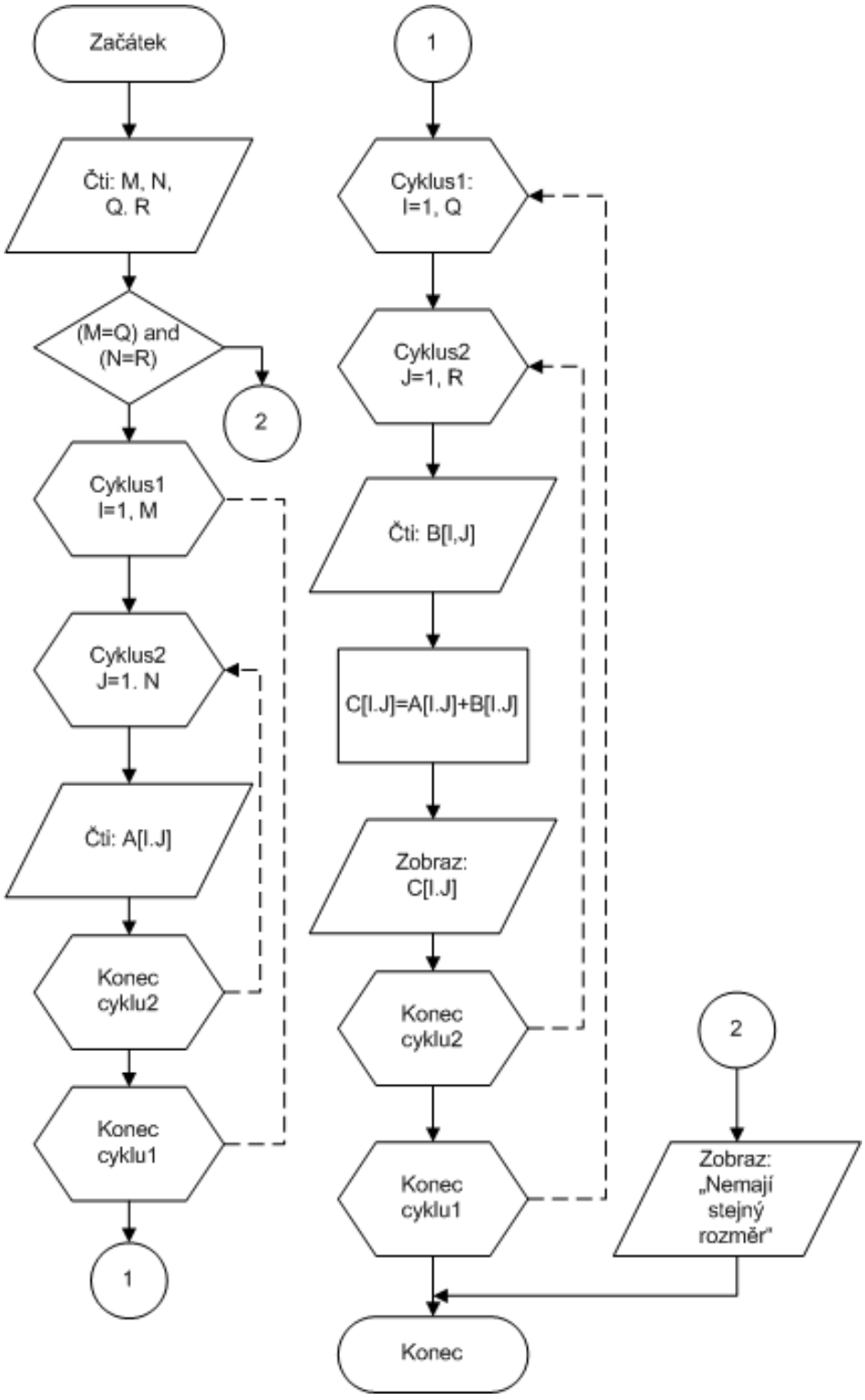
13. Obrázek – Diagram výběr největšího a nejmenšího prvku matice

Příloha C – diagram určení diagonální matice



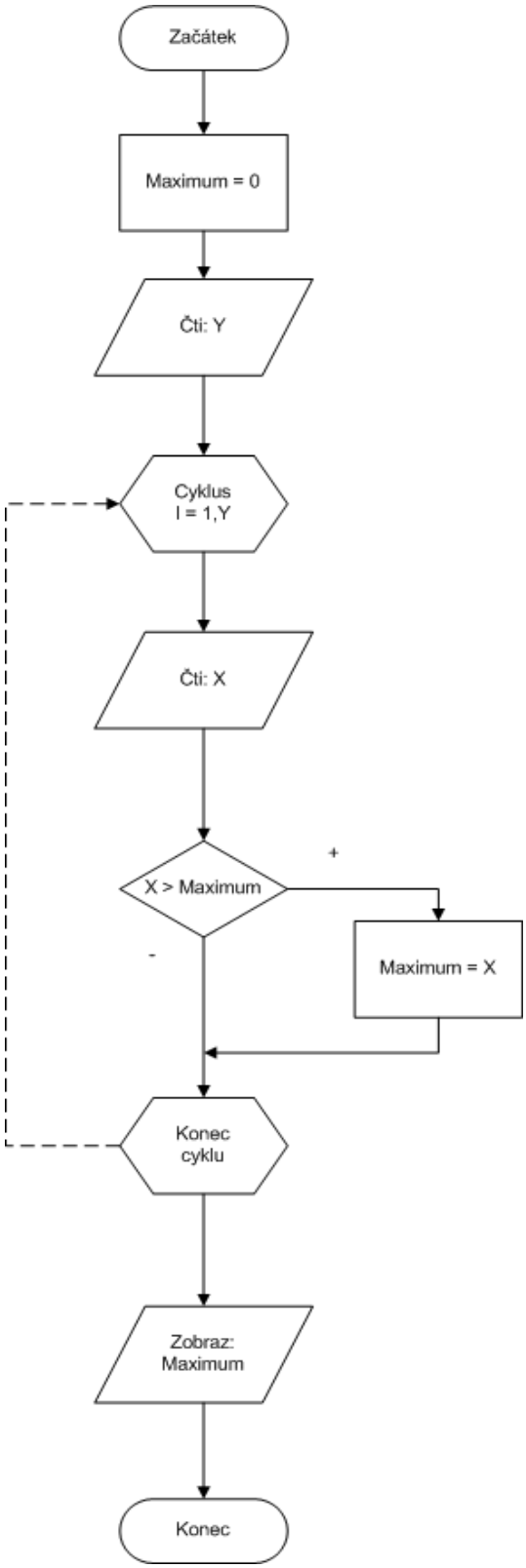
14. Obrázek – diagram určení diagonální matice

Příloha D – diagram součtu matic



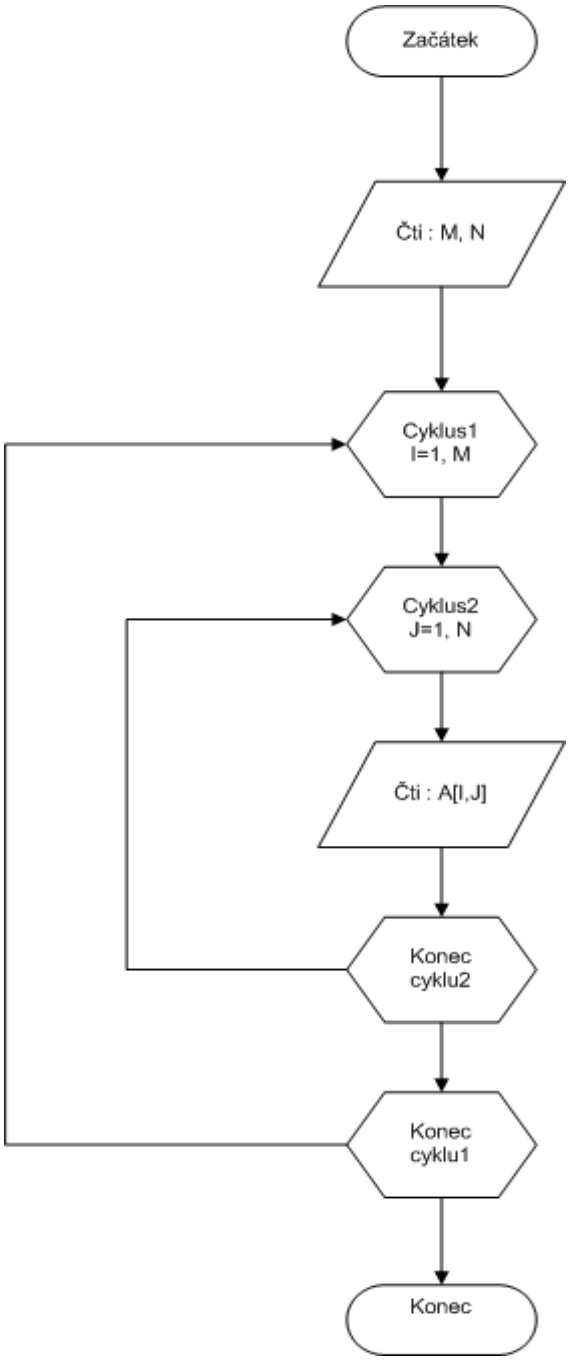
15. Obrázek – diagram součtu matic

Příloha E – diagram určení největšího prvku posloupnosti



16. Obrázek – diagram určení maxima z posloupnosti

Příloha F – diagram načítání matice



17. Obrázek – diagram načtení matice