ZÁPADOČESKÁ UNIVERZITA V PLZNI

FAKULTA STROJNÍ

Studijní program: B 2301 Strojní inženýrství

Studijní zaměření: Průmyslové inženýrství a management

BAKALÁŘSKÁ PRÁCE

Tvorba studijních opor pro modernizaci výuky Technické informatiky v C#

Autor: **Jakub Müller**

Vedoucí práce: **Ing. Pavel Raška, Ph.D.**

Akademický rok 2018/2019

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta strojní
Akademický rok: **2018/2019**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Jakub MÜLLER**
Osobní číslo:	**S16B0100P**
Studijní program:	**B2301 Strojní inženýrství**
Studijní obor:	**Průmyslové inženýrství a management**
Název tématu:	**Tvorba studijních opor pro modernizaci výuky Technické informatiky v C#**
Zadávající katedra:	**Katedra průmyslového inženýrství a managementu**

Zásady pro vypracování:

1. Úvod do řešené problematiky
2. Analýza současného stavu
3. Návrh řešení
4. Zpracování podpůrných materiálu k výuce Technické informatiky
5. Závěr

Rozsah grafických prací: **0 výkresů**

Rozsah kvalifikační práce: **30 - 40 stran**

Forma zpracování bakalářské práce: **tištěná**

Seznam odborné literatury:

1. **KLAUSEN, P.** *C# 1 Introduction to programming and the C# language.* **B.m.: Poul Klausen & bookboon.com, 2012. ISBN 978-954-400-773-7**
2. **MILES, R.** *C# Programming Yellow Book.* **B.m.: Rob Miles, 2016. ISBN 150-930-115-7**
3. **SVETLIN, N.** *Fundamentals of Computer Programming with C#.* **Sofia: Nakov Svetlin & Co., 2013. ISBN 978-954-400-773-7**

Vedoucí bakalářské práce: **Ing. Pavel Raška, Ph.D.**
Katedra průmyslového inženýrství a managementu

Konzultant bakalářské práce: **Ing. Petr Hořejší, Ph.D.**
Katedra průmyslového inženýrství a managementu

Datum zadání bakalářské práce: **24. září 2018**

Termín odevzdání bakalářské práce: **24. května 2019**

Západočeská univerzita v Plzni
Fakulta strojní

L.S.

Doc. Ing. Milan Edl, Ph.D.
děkan

Doc. Ing. Michal Šimon, Ph.D.
vedoucí katedry

V Plzni dne 24. září 2018

Poděkování

Chtěl bych poděkovat Ing. Pavlovi Raškovi, Ph.D., vedoucímu mé bakalářské práce za cenné rady a trpělivost. Rovněž bych chtěl poděkovat Ing. Petru Hořejšímu, Ph.D., doc. Ing. Zdeňku Ulrychovi, Ph.D. a všem, kteří mi radou a cennými zkušenostmi umožnili dokončit práci.

Prohlášení o autorství

Předkládám tímto k posouzení a obhajobě bakalářskou práci, zpracovanou na závěr studia na Fakultě strojní Západočeské univerzity v Plzni.

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně, s použitím odborné literatury a pramenů, uvedených v seznamu, který je součástí této bakalářské práce.

V Plzni dne: …….. . . . . .

podpis autora

ANOTAČNÍ LIST BAKALÁŘSKÉ PRÁCE

AUTOR	**Příjmení** Müller	**Jméno** Jakub
STUDIJNÍ OBOR	B2301 „Průmyslové inženýrství a management“	
VEDOUCÍ PRÁCE	**Příjmení (včetně titulů)** Ing. Raška Ph.D.	**Jméno** Pavel
PRACOVIŠTĚ	ZČU - FST - KPV	

DRUH PRÁCE	~~**DIPLOMOVÁ**~~	**BAKALÁŘSKÁ**	**Nehodící se škrtněte**
NÁZEV PRÁCE	Tvorba studijních opor pro modernizaci výuky Technické Informatiky v C#		

FAKULTA	strojní		**KATEDRA**	KPV		**ROK ODEVZD.**	2019

POČET STRAN (A4 a ekvivalentů A4)

CELKEM	47		**TEXTOVÁ ČÁST**	36		**GRAFICKÁ ČÁST**	0

STRUČNÝ POPIS (MAX 10 ŘÁDEK) **ZAMĚŘENÍ, TÉMA, CÍL POZNATKY A PŘÍNOSY**	Bakalářská práce je zaměřena na tvorbu podkladů k modernizaci výuky Technické informatiky. V práci je provedena analýza současného stavu předmětu a následně navržena řešení. Návrhy inovace a modernizace byly provedeny v posloupnosti cvičení, programech a tvorbě aplikací. Cílem práce je přiblížení předmětu studentům.
KLÍČOVÁ SLOVA **ZPRAVIDLA JEDNOSLOVNÉ POJMY, KTERÉ VYSTIHUJÍ PODSTATU PRÁCE**	Programování, jazyk C#, informatika, studijní podklady, modernizace výuky

SUMMARY OF BACHELOR SHEET

AUTHOR	**Surname** Müller	**Name** Jakub	
FIELD OF STUDY	B2301 "Industrial Engineering and Management"		
SUPERVISOR	**Surname (Inclusive of Degrees)** Ing. Raška Ph.D.	**Name** Pavel	
INSTITUTION	ZČU - FST - KPV		
TYPE OF WORK	~~**DIPLOMA**~~	**BACHELOR**	**Delete when not applicable**
TITLE OF THE WORK	Development of Study Support for the Modernization of the Course of Informatics with C# for Technical Fields		

FACULTY	Mechanical Engineering		**DEPARTMENT**	Industrial engineering and management		**SUBMITTED IN**	2019

NUMBER OF PAGES (A4 and eq. A4)

TOTALLY	47		**TEXT PART**	36		**GRAPHICAL PART**	0

BRIEF DESCRIPTION **TOPIC, GOAL, RESULTS AND CONTRIBUTIONS**	The Bachelor thesis is focused on the development of a study support used for the modernization of the Technical informatics course. An analysis of the current state of the subject is performed and the solution to this problem is proposed. Innovation and modernization of the study material, programs and software applications were performed. The aim of this thesis is to make the study support of this course more understandable to students.
KEY WORDS	Programming, C# language, education, informatics, study support, modernization

Obsah

Seznam příloh

Seznam obrázků

Seznam tabulek

1 Úvod

Problematika týkající se výuky předmětu Technická informatika (dále jen TI), který je vyučován během letního semestru v prvním roce bakalářského studia na Fakultě strojní, by měl studenta vybavit základy programování a ukázat, jak by měl student technického oboru přemýšlet.

Já, jakožto student, který tento předmět absolvoval před dvěma roky, jsem si ohledně tohoto předmětu vytvořil vlastní představu. Předmět TI spousty lidí nechápe a ani nechce pochopit, protože si nedokáží představit, k čemu strojaři využijí programování. Je vhodné, aby bylo ukázáno možné využití programů ve strojírenství a zároveň byly vysvětleny základy programování, jako jsou např. cykly a podmínky. V současné době při rychlém vývoji Průmyslu 4.0 se vše snažíme digitalizovat a to znamená, že poptávka po lidech v oblasti strojírenství, kteří umí programovat, enormně roste.

Předmět TI je vyučován v jazyce Delphi (Pascal), který je v současnosti brán svým návrhem jako zastaralý jazyk. Pokud není přímý důvod programovat v tomto jazyku, přicházejí na řadu moderní jazyky jako je C#, který je inovativnější, je po něm vysoká pracovní poptávka, má širokou podporu, a spoustu dalších výhod.

V této práci bude především popisován programovací jazyk C#. Jazyk je moderní, objektově orientovaný a typově dobře zabezpečený. V dnešní době je jazyk mezinárodně využívaný hlavně díky společnosti Microsoft. Jako náhrada stávajícího jazyka Delphi (Pascal) připadaly dva programovací jazyky, a to C# a Java. Jazyk C# je, podle mého mínění, oproti jazyku Java pro začátečníky lehčí na pochopení, a proto byl zvolen učiteli jako náhrada doposud využívaného programovacího jazyka Delphi.

Základní snahou bakalářské práce je zhotovení nových studijních materiálů pro cvičení z předmětu TI a zároveň ukázat kvalitu jazyka C#. Je těžké učit studenty něco, čemu nerozumí a nemají o to příliš velký zájem.

2 Analýza současného stavu

Analýza současného stavu pojednává o čtyřech tématech. První téma je zastaralost jazyku Delphi a jeho uživatelského prostředí Delphi. Obsahem druhého tématu je předmět TI a studenti. Třetím a čtvrtým tématem je shrnutí přednášek a cvičení.

2.1 Jazyk Delphi (Pascal)

Systémové prostředí Delphi je standardně složeno ze čtyř nebo pěti oken. Programování v Delphi je z velké části založeno na použití komponent. Komponenta je malý objekt, který vykonává určitou činnost. Delphi je založeno na programovacím jazyce Pascal, používá VCL (Visual Component Library) a CLX (Component Library for Croos Platform), možnost propojení s databázemi, deklarace metod a členských proměnných objektových tříd, používá vlastních zpráv k vyvolání událostí jednotlivých tříd, objektový model je nezávislý na počtu implementací jednotlivých tříd a zahrnuje možnost převedení do .NET kódu.

Velkou výhodou Delphi je spousta předvytvořených komponent, což zefektivňuje práci. Nevýhod je více a to obsáhlost, syntaxe Delphi nevyužitelnost jazyka. Jazyk Delphi je zastaralý a v dnešní době se příliš nevyužívá.

2.2 TI a studenti

Při rekonstrukci předmětu nejde o jeho změnu ohledně požadavků na zápočet a zkoušku, ale o snahu inovovat příklady, které budou zaměřeny blíže ke strojírenství a budou využívat modernější programovací jazyk. Tímto způsobem můžeme studentům ukázat, čeho mohou sami dosáhnout v programování. V Delphi se samozřejmě naučíme také velmi dobře programovat, ale vizuální výstup aplikace Delphi zaostává za Visual Studiem od společnost Microsoft.

Zároveň bychom chtěli do příkladů zahrnout komentáře, aby studenti lépe pochopili logiku daných příkazů. Dosti často se stává, že studenti se dané postupy příkladů ke zkoušce učí nazpaměť. Předmět TI by neměl být stěžejním předmětem prvního ročníku a ke zkoušce stačí pochopit základy programování. Základy programování se rozumí například, co operace provádí, nebo co se vytváří v daném programu.

2.3 Přednášky

Následující tabulka (viz Tabulka 2-1) obsahuje popis stávajících přednášek v rámci předmětu TI.

Přednáška	Obsah
1.	Základní pojmy a zobrazení údajů v počítači, převody čísel do dvojkové a šestnáctkové soustavy a aritmetika
2.	Základní datové typy, intuitivní příklad a programovací jazyky obecně
3. a 4.	Příkazy vstup a výstup, příkazy cyklů, podmíněné příkazy a složené příkazy
5. a 6.	Vývojové diagramy, jejich použití a jak pomáhají pochopit danou problematiku příkladu
7.	Procedury, funkce, rekurze, globální a lokální proměnné a uživatelské hodnoty
8.	Uživatelem definované typy – interval, množina, pole a práce s maticemi
9.	Soubory, druhy souborů, procedury, funkce pro práci se soubory a práce s formáty výstupu
10. a 11.	Řešení vzorových příkladů
12.	Nové prostředky a aplikace

Tabulka 2-1 – Stávající struktura přednášek

2.4 Cvičení

Předmět TI tvoří 12 cvičení. Následující tabulka (viz Tabulka 2-2) obsahuje popis stávajících cvičení v rámci předmětu TI.

Cvičení	Obsah
1.	Program "Ahoj Svete"
2.	Převod sekund na hodiny, minuty a sekundy
3.	Výčetka platidel, kde se opět využijí „if“, „else if“ a „else“, ale nyní s trochu komplikovanějšími podmínkami
4.	Výpočet objemu a povrchu koule, za i bez pomoci procedur a funkcí
5.	Výběr většího ze dvou, tří nebo čtyř čísel
6.	Výpočet kořenů lineární a kvadratické rovnice, kdy vytváříme procedury pro lineární rovnici, rovnici s reálnými kořeny a rovnici s komplexními kořeny, které pak voláme pod tlačítkem „Vypočti“
7., 8. a 9.	Cykly, které se procvičují na příkladech výpočet faktoriálu, výpis prvních dvaceti prvočísel a převod čísla do binárního tvaru. V Object Delphi máme celkem tři cykly „for“, „while“ a „repeat“
10.	Odstranění diakritiky z textu
11. a 12.	Jednorozměrné pole

Tabulka 2-2 - Stávající struktura cvičení

3 Rešerše

Rešerše byla provedena za účelem nastudování dokumentů zaměřených na výuku C#. Tři rešerše byly vybrány na základě doporučení vedoucího bakalářské práce. Čtvrtá rešerše byla vybrána pro dostatečné porozumění látky. Dvě učebnice využívané k výuce na zahraničních univerzitách, jeden webový odkaz s kurzem a všemi základy ohledně programování v C#. Dále byla vybrána kniha pro pochopení logiky programování. Každý z těchto zdrojů má své klady i zápory. Z této rozmanitosti zdrojů lze jednoduše pochopit jazyk C#.

Součástí této rešerše je i analýza diskuzí na internetu kvůli porovnávání jazyků Delphi a C# mezi širší komunitou.

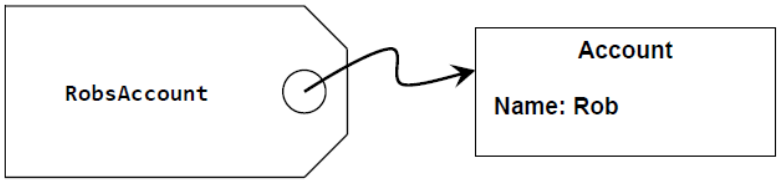
3.1 Pramen č.1 – C# Programming Yellow Book, Rob Miles

Pro účely této práce byly vyhledány učebnice pro začátečníky o programování v jazyce C#. Uváděný pramen č. 1 se vyskytoval na 5. místě v oblíbenosti pro rok 2017. Pro zajímavost na 1. místě podle zdroje byla učebnice „C# in Depth“. Autorem učebnice je kantor z University of Hull v Anglii. [1]

```
class Account
{
      public string Name ;
} ;

class StructsAndObjectsDemo
{
      public static void Main ()
      {
            Account RobsAccount ;
            RobsAccount = new Account();
            RobsAccount.Name = "Rob";
            Console.WriteLine (RobsAccount.Name );
      }
}
```

Code Sample 28 Compiling Account Class

The line I have added creates a new `Account` object and sets the reference `RobsAccount` to refer to it.



We have seen this keyword `new` before. We use it to create arrays. This is because an array is actually implemented as an object, and so we use `new` to create it. The thing that new creates is an *object*. An object is an instance of a class. I'll repeat that in a posh font:

"An object is an instance of a class"

Obrázek 3.1 - Ukázka stylu výkladu prvního pramene [2]

Učebnice je převážně určena pro začátečníky. Autor na začátku učebnice uvádí obecnou definici počítače, a jakou funkci má počítač vzhledem k programování. Na základním příkladu autor vysvětluje základy programování, jako jsou datové typy, důvod použití „void“, nebo co dělá program funkční. Všechny typy jsou vždy uvedeny a vysvětleny pod příslušným kódem.

Další důležitou částí učebnice pro seznámení se s programovacím jazykem C# je práce s textem, kde autor popisuje datové typy „char, string, bool“ a přiřazovaní hodnot k proměnným. V učebnici se nachází odstavce „Programmer's Point“, kterými se autor snaží zjednodušit danou problematiku. Následuje ukázka, jak správně v posloupnosti psát program. V souvislosti s psaním programu autor vysvětluje ovládání v průběhu vývoje. Zde jsou poprvé zmíněny podmínky a operátory k nim vztažené. Podmínka je i součástí cyklů, a proto autor navazuje cykly „do-while“, „while“, „for“ a „foreach“. V učebnici se uvádí i mnohé propojení a vnoření cyklů.

Další téma je velice důležité, jelikož se zde zmiňuje popis různých prvků jako je například přiřazování hodnot.

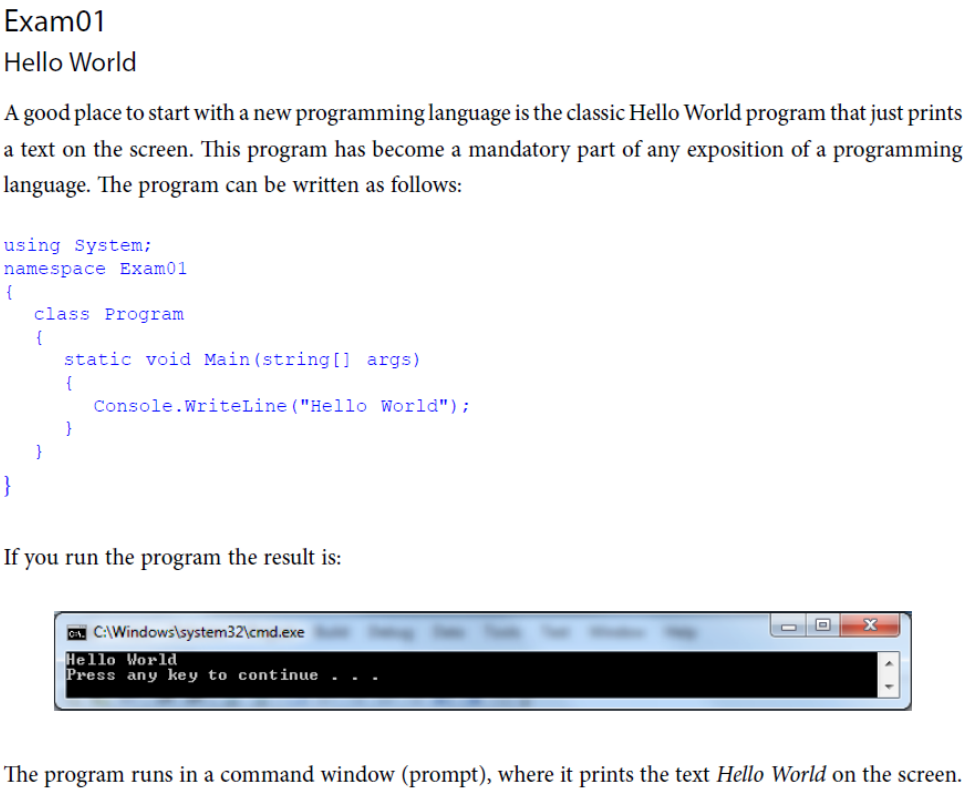
Následuje téma metody. Autor na začátku kapitoly téma popíše a vysvětluje užívání metod hlavně díky přehlednosti a zjednodušení. Závěrem kapitoly je popsáno ovládání skrze parametry a vracení hodnot.

Po metodách přichází na řadu pole. Autor zde vypisuje využití polí a rozdělení na jednorozměrné či vícerozměrné. Text je prolínán jednoduchými příklady pro správné pochopení. Prvek využívající se při výběru z více možností je „switch“. Pro prvek „switch“ je uvedeno, jak by mohla vypadat konstrukce takové funkce. Struktury, objekty a definice autor vysvětluje pomocí obrázků a textu. Příklad pro logické pochopení této problematiky je „bankovní účet“.

Dalším tématem je práce s textem. Popis zahrnuje příkazy, editaci textu a manipulaci. Dokument obsahuje ještě další užitečné informace. Tyto informace jsou však nad rámec předmětu TI, a tedy pro účel této práce nadbytečné. Na závěr autor přiložil přehled důležitých pojmů. [2]

3.2 Pramen č.2 − C# 1 – Introduction to programming and the C# language

Učebnice byla mezi volně dostupných materiálů pro výuku jazyku C#. Učebnice byla napsána kantorem z Aarhus University v Dánsku. Příklady jsou lehké a učebnice není těžká na pochopení. [3]



Exam01

Hello World

A good place to start with a new programming language is the classic Hello World program that just prints a text on the screen. This program has become a mandatory part of any exposition of a programming language. The program can be written as follows:

```
using System;
namespace Exam01
{
   class Program
   {
      static void Main(string[] args)
      {
         Console.WriteLine("Hello World");
      }
   }
}
```

If you run the program the result is:

The program runs in a command window (prompt), where it prints the text *Hello World* on the screen.

Obrázek 3.2 - Ukázka stylu výkladu druhého pramene [4]

Kniha začíná jako většina kurzů a učebnic na jednoduchém příkladu „Hello World“. K příkladu je podrobně sepsán postup vytvoření nového projektu v aplikaci Visual Studio. Zde bych vytkl autorovu snahu o vysvětlení založení nového projektu v aplikaci, jelikož se na založení samostatného projektu velmi často zapomíná, a přitom se jedná o jednu z nejzákladnějších informací. Stejně jako u předešlé rešerše (viz kapitola 3.1) jsou dalším tématem v osnově proměnné. Zde autor popisuje správné deklarování proměnných a jejich rozsah. Součástí tématu jsou i příklady v kterých můžeme vidět různé využití operátorů.

V další části autor představuje konzolovou aplikaci (čistě kód s výstupem příkazového řádku) na zajímavém příkladu pro vypočtení obvodu a plochy kruhu. Pod kódem autor popisuje nutnost použití třídy „Math“ a přidává také vysvětlení pojmu „placeholder“.

Kontrola programů je další téma po základním uvedení do podstaty. Autor začíná popisem smyček „while“, „for“ a „do while“. Jsou také vysvětleny podmínky „if“ a „switch“. Ke každému z cyklů je uveden příklad s přidaným komentářem k lepšímu pochopení. Ukázkou na vnořené rozhodování „if-elseif – else“ je program na výpočet diskriminantu, nebo pro příkaz „switch“, který je využit v programu na výběr dne z týdne.

Důležité téma je práce s textem. Autor se opět snaží vysvětlit problematiku pomocí příkladu doplněného stručným popisem, formátováním textu, a jak deklarovat text v podobě objektu.

V dalším tématu učebnice jsou popsána pole. Autor popisuje pole stejně jako v předešlé učebnici (viz kapitola 3.1). Autor v následujícím tématu otevírá kapitolu objektově orientovaného programování. Autor zde začíná pojmem třída, co je to třída, kde se používá a

jak se používá. Vysvětleni funkčnosti třídy je ukázáno na příkladu „hod mincí“. V příkladu se objevují metody a funkce pro vrácení náhodné hodnoty.

Následující téma začíná příkladem „bod“ na kterém se autor snaží vysvětlit dědičnost. Stručný popis přikladu „bod“ je: vytvoření rodiče, dědění na potomka. Právě dědění na potomka je ukázkou dědičnosti.

Ve střední části učebnice autor doplňuje využití vícerozměrných polí, popis statických členů a strukturu „struct“.

Až na závěr je zbytek učebnice pro pokročilejší programování, které je nad rámec této práce. V závěru autor popisuje matematické funkce ve spojení s programováním a práci se soubory, kdy aplikace dokáže načíst text ze souboru nebo vepsat text do souboru. [4]

3.3 Pramen č.3 – webová stránka www.ITnetwork.cz

Vzhledem k programování patří stránka k jednomu z největších serverů zabývajících se programováním na Českém internetu. Na stránce jsou placené online kurzy, ale i neplacené kurzy určené k výuce základů programování ve více než osmi programovacích jazycích. Server nabízí kurzy: Základní konstrukce jazyka C#, Objektově orientované programování v C#, formulářové aplikace v C#.NET, Visual Studio a plno dalších kurzů vztažených k jazyku C#.

Kurz Základní konstrukce jazyka C# umožňuje vypracování kódů s dodatečnými informacemi k pochopení. Na konci každé lekce je napsáno, co bylo probráno a jaký typ příkladu jsme schopni naprogramovat. V kurzu se učí programovat jednoduché konzolové aplikace, jakými jsou například řazení s třídou, tvorba kalkulačky a tvorba dvourozměrného pole. Kurz je rozdělen do dvanácti lekcí.

Druhým kurzem, který je potřeba projít k znalosti tvorby WPF, je kurz tzv. okenní, tedy formulářové aplikace v C#.NET. Kurz je rozdělen do deseti lekcí. Vychází z absolvování kurzu o konzolových aplikací. Tento kurz slouží k přiřazení kódu k ovládacímu prvku. Zde se tedy probírá práce s formulářem. Pracovat s formulářem je možné pomocí dvou způsobů. První možností je XAML kód, což zjednodušeně znamená, že píšeme kód s určitými příkazy, který nám mění vlastnosti – vzhled formuláře. Druhou možností je tvorba pomocí panelu nástrojů a vlastností. Pro začínajícího studenta je druhá možnost o něco rychlejší a mnohem snazší, ale zároveň mnohem méně přesná vůči polohování nástrojů. V předmětu TI budeme tvořit okno pomocí nástrojů, takže není zapotřebí učit se detailně jazyk XAML. [5]

3.4 Pramen č.4 – Fundumentals of Computer Programming with C#

Autorem této knihy je Svetlin Nakov & Co. Kniha je určena, jak pro úplné začátečníky s programováním, tak i pro lehce pokročilé. Kniha se snaží naučit čtenáře přemýšlet v algoritmech. Tato kniha je určena pro pokročilé programování, ne jako kniha orientovaná přímo na C#.

Autor se na začátku knihy snaží vysvětlit, co je programovací jazyk C# a Microsoft .NET Framework. Dále porovnává jazyky mezi sebou, vypisuje výhody moderních programovacích jazyků vzhledem k dnešní poptávce po programátorech a snaží se popsat, čím vším by měl programátor disponovat. V knize se vyskytuje 350 problémů, které se sám čtenář snaží opravit. Kniha by se dala přirovnat k webové stránce www.ITnetwork.cz, vysvětlivky k příkladům jsou psány podobným stylem. Kniha je výborně vizuálně provedena, jelikož texty jsou doplněny o snímky obrazovky přímo z Visual Studia. [6]

How Does Our First C# Program Work?

Our first program consists of three logical parts:

- Definition of a class **HelloCSharp**;
- Definition of a method **Main()**;
- Contents of the method **Main()**.

Defining a Class

On the first line of our program we define a class called **HelloCSharp**. The simplest definition of a class consists of the keyword **class**, followed by its name. In our case the name of the class is **HelloCSharp**. The content of the class is located in a block of program lines, surrounded by curly brackets: **{}**.

Defining the Main() Method

On the third line we define a method with the name **Main()**, which is the starting point for our program. Every program written in C# starts from a **Main()** method with the following title (signature):

```
static void Main(string[] args)
```

The method must be declared as shown above, it must be **static** and **void**, it must have a name **Main** and as a list of parameters it must have only one parameter of type **array of string**. In our example the parameter is called **args** but that is not mandatory. This parameter is not used in most cases so it can be omitted (it is optional). In that case the entry point of the program can be **simplified** and will look like this:

```
static void Main()
```

If any of the aforementioned requirements is not met, the program will compile but it will not start because the starting point is not defined correctly.

Contents of the Main() Method

The content of every method is found after its signature, surrounded by opening and closing curly brackets. On the next line of our sample program we use the system object **System.Console** and its method **WriteLine()** to print a message on the default output (the console), in this case "Hello, C#!".

In the **Main()** method we can write a random sequence of expressions and they will be executed in the order we assigned to them.

More information about expressions can be found in chapter "Operators and Expressions", working with the console is described in chapter "Console Input and Output", classes and methods can be found in chapter "Defining Classes".

Obrázek 3.3 - Ukázka stylu výkladu čtvrtého pramene [6]

4 Návrh řešení

Návrh řešení by měl podpořit inovaci předmětu TI a uvést nedostatky v současné době. Dva návrhy „A“ a „B“ se od sebe velmi liší, v každém je zvolen jiný druh tvorby aplikací. V návrhu „A“ tvorba aplikací WPF a v návrhu „B“ tvorba konzolových aplikací.

4.1 Návrh řešení A

Návrh nové osnovy přednášek a cvičení. Návrh A pracuje s tvorbou aplikací WPF a vychází z analýzy současného stavu (viz kapitola 2). Přednášky a cvičení budou transformovány do nové podoby (viz Tabulka 2-1 a Tabulka 2-2).

4.1.1 Přednášky

Od současného stavu by se první přednáška lišila tím, že základní pojmy by byly zkráceny a přidaly by se datové typy a příkaz na výstup „Console.WriteLine“.

Druhá přednáška by obsahovala základní prvky „Button“, „Label“ a „TextBox“, jejich použití a základní příkazy na vstup a výstup. Základní výstup pomocí příkazu „Console.WriteLine“ by byl popsán v první přednášce. V druhé přednášce by se objevily výstupy „Message.Show“ a „Txt.Text=“.

Ve třetí a čtvrté přednášce by byl zásadní rozdíl v přidání základů XAML kódu, který sice na cvičeních využíván není, ale bylo by dobré se o něm zmínit. Jelikož plno programů tvořených na cvičení obsahuje matematické operátory a funkce je vhodné zavést informace o používání i do přednášek. Dále by bylo přiřazeno téma ohledně předávání parametrů.

U páté přednášky je pouze rozdíl v přidání pokračování podmínek a rekurze.

V šesté přednášce by mělo být plně obsaženo téma probírané v současné době na sedmé přednášce doplněné o cykly.

Sedmá přednáška by odpovídala současné osmé přednášce s větším rozsahem o typu pole, délka pole, ukládání do pole a vysvětlení zbylých cyklů z předchozí přednášky.

Novým tématem, které v současné podobě přednášek není detailně probíráno, jsou tří a vícerozměrná pole. Příklad na vícerozměrné pole by byl programován na jedenáctém cvičení. Na současných cvičeních se tří a vícerozměrná pole nevyskytují, ale pro případné další programování je dobré studenta s tímto tématem seznámit.

Dvanáctá přednáška by byla úplně nová. Na přednášce by se probíralo spojení programování a strojírenství. Tyto dva obory jsou v dnešní době propojené a pro studenta může být zajímavým tématem stejně jako poptávka po práci.

Přednáška	Obsah
1.	Základní pojmy do úvodu v programování, datové typy a příkaz na výstup "Console.WriteLine"
2.	Základní prvky pro vytváření formuláře WPF, XAML kód, příkazy vstupu a výstupu WPF aplikace
3. a 4.	Práce s datovými typy, matematické funkce, předání parametrů a podmínky
5.	Podmínky pokračování a vývojové diagramy, rekurze a jak se vytvářejí a k čemu se používají
6.	Cykly, procedury, funkce, globální a lokální proměnné a uživatelské hodnoty
7.	Pokračování cykly, jednorozměrné pole a uživatelem definované hodnoty – interval, množina a práce s maticemi
8.	Práce se soubory, příkazy pro načítání nebo ukládání textu
9.	Jednorozměrná pole a shrnutí – podmínky, cykly a vícerozměrná pole
10. a 11.	Práce s třídami a řešení vzorových příkladu
12.	Spojení programování a strojírenství, ukázka poptávky a modernizace

Tabulka 4-1 - Nově navržená struktura přednášek

4.1.2 Cvičení

Pro cvičení budou navrženy dvě varianty (WPF a konzolové aplikace). Konzolové aplikace jsou rozebrány v návrhu „B“ viz (kapitola 4.2).

Modifikace třetího cvičení je z důvodu podobnosti příkladu vytvořeném na druhém cvičení. Současný příklad na třetím cvičení opět využívá podmínky typu „if“, „else if“ a „else“, je celkově velmi zdlouhavý pro začátečníka a lze v něm udělat lehce chybu. V novém programu by mohlo být použito vícenásobné větvení „switch“ a jednalo by se o celkem krátký program.

Příklad ze šestého cvičení na lineární a kvadratické rovnice není špatný, ale spíše problematický při přenášení vzorců do kódu a uvědomění si, jak podmínky specifikovat. Tento příklad je pro mě velkým otazníkem, každopádně bych se spíše naklonil k řešení programu na výpočet faktoriálu a procvičení cyklů. Cykly máme celkem čtyři v C#, a to „for“, „foreach“, „while“ a „do while“. Vypočet faktoriálu je názorným příkladem toho, jak funguje cyklus „for“.

Sedmé a osmé cvičení by bylo zaměřeno na cykly, kde by byl naprogramován jeden příklad na cyklus „while“ a druhým příkladem by byly dva cykly v sobě na velmi jednoduchém a logickém příkladu. V současné době se cykly probírají na sedmém, osmém a devátém cvičení, což je dle mého názoru až moc velké soustředění se na cykly.

Na devátém cvičení by byl vytvořen program na tvorbu pole o deseti prvcích, které by bylo naplněno náhodnými čísly pomocí funkce „Random“.

Desáté cvičení by navazovalo na deváté a pokračovalo by se v psaní programu. Do programu by se přidal cyklus, který by seřadil všech deset náhodných čísel, jelikož současný program na odstranění diakritiky mi přijde nevhodně vložený do pořadí cvičení.

Jedenácté a dvanácté cvičení by bylo celkově modifikováno. Na jedenáctém cvičení by se psal kód pro vytvoření vícerozměrné pole. Dvanácté cvičení by bylo zaměřeno na práci s třídami.

Cvičení	**Obsah**
1.	Program "Ahoj Svete"
2.	Převod sekund na hodiny, minuty a sekundy
3.	Nový příklad na procvičení podmínky "switch"
4	Program na výpočet objemu a povrchu koule pomocí procedur a funkcí
5.	Výběr většího ze dvou, tří nebo čtyř čísel
6.	Program na výpočet faktoriálu --> využití cyklů
7.	Program na procvičení cyklu "while"
8.	Zacyklení dvou cyklů do sebe na jednoduchém a logickém programu
9.	Vytvoření jednorozměrného pole o deseti náhodných číslech přes funkci "Random"
10.	Navázání na předchozí cvičení, přidání cyklu na seřazení deseti náhodných čísel
11.	Program na vícerozměrná pole
12.	Práce s třídami

Tabulka 4-2 - Nově navržená struktura cvičení

4.1.3 Rozdíly mezi programovacími jazyky C# a Delphi

V této kapitole jsou na jednoduchém příkladu ukázány rozdíly mezi oběma programovacími jazyky – C# a Delphi.

C# kód:

```
namespace Prevod
{
   public partial class MainWindow : Window
   {
      public MainWindow()
      {
         InitializeComponent();
      }
      private void BtnPrevod_Click(object sender, RoutedEventArgs e)
      {
         int celkem, hodin, minut, sekund;
         if (int.TryParse(TxtBoxCelkem.Text, out celkem))
         {
            hodin = celkem / 3600;
            celkem = celkem % 3600;
            minut = celkem / 60;
            sekund = celkem % 60;
            TxtBoxHodin.Text = hodin.ToString();
            TxtBoxMinut.Text = minut.ToString();
            TxtBoxSekund.Text = sekund.ToString();
         }
         else
         {
            TxtBoxCelkem.Focus();
         }
      }
      private void BtnKonec_Click(object sender, RoutedEventArgs e)
      {
         this.Close();
      }
   }
}
```

Obrázek 4.1 - Příklad na převod celkového počtu sekund – kód v C#

Delphi kód:

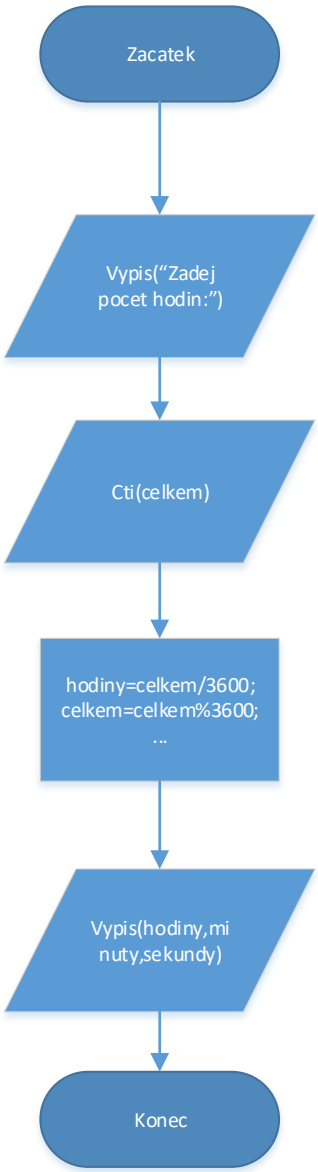
```
var
  frmPrevod: TfrmPrevod;
implementation
{$R *.dfm}
procedure TfrmPrevod.btnPrevodClick(Sender: TObject);
var
  celkem, hodin, minut, sekund: Integer;
begin
  If TryStrToInt(edtcelkem.Text, celkem) then
  begin
    hodin:= celkem div 3600;
    celkem:= celkem mod 3600;
    minut:= celkem div 60;
    sekund:= celkem mod 60;

    self.edtHodin.Text:= IntToStr(hodin);
    self.edtMinut.Text:= IntToStr(minut);
    self.edtSekund.Text:= IntToStr(sekund);
  end
  else
    self.edtCelkem.SetFocus
end;
  procedure TfrmPrevod.btnKonecClick(Sender: TObject);

  begin

    close;

  end;
```

Obrázek 4.2 - Příklad na převod celkového počtu sekund – kód v Delphi

Tato práce slouží jako podklad pro tvorbu studijních materiálů pro C#, proto zde bude popsán detailněji kód jazyka C# a čím se liší od jazyku Delphi. Deklarace proměnných je v obou případech skoro stejná. První větší rozdíl je, že C# používá pro začátek a konec „{}“ namísto „begin“ a „end“, což je velmi příjemné obzvlášť při práci s podmínkami.

Další rozdíl lze vidět při použití příkazu „int.TryParse(txtCelkem.Text, out celkem)“. Do pole TextBox pojmenovaném „TxtBoxCelkem“ se napíše počet sekund, tento údaj načte program jako datový typ „string“ a příkaz „int.TryParse(txtCelkem.Text, out celkem)“ ho převede na celočíselný datový typ „integer“. V Delphi převedení datového typu docílíme příkazem „TryStrToInt(edtcelkem.Text, celkem)“. Posledním rozdílem v uváděném příklady, který může být celkem otravný při psaní těžšího programu zabývajícím se matematikou, je nevýhoda u Delphi v používání matematických operátorů, kdy například děleno v C# píšeme do kódu klasicky „/“, ale v Delphi musíme použít příkaz „div“.

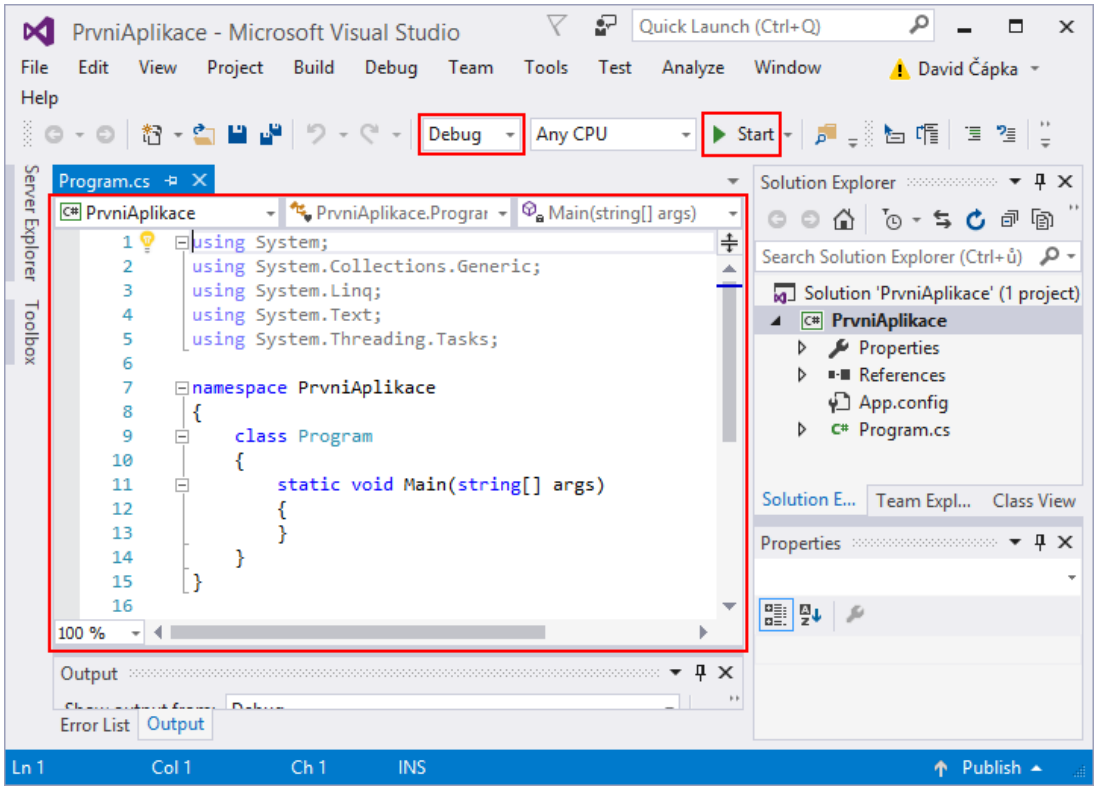
Ke každému příkladu bude vytvořen příslušný vývojový diagram, který by měl sloužit k pochopení dané problematiky programu a pomoci při jeho řešení – viz Obrázek 4.3.



Obrázek 4.3 - Vývojový diagram k příkladu na druhém cvičení

4.2 Návrh řešení B

Tento návrh je založen na základu absolvování začátečnických kurzů a předmětů týkajících se programování vyučovaných na Západočeské univerzitě. Návrh spočívá v tom, že by se celý předmět TI vedl pouze formou konzolových aplikacích. Konzolová aplikace je kód s výstupem příkazového řádku.



Obrázek 4.4 - Začátek konzolové aplikace

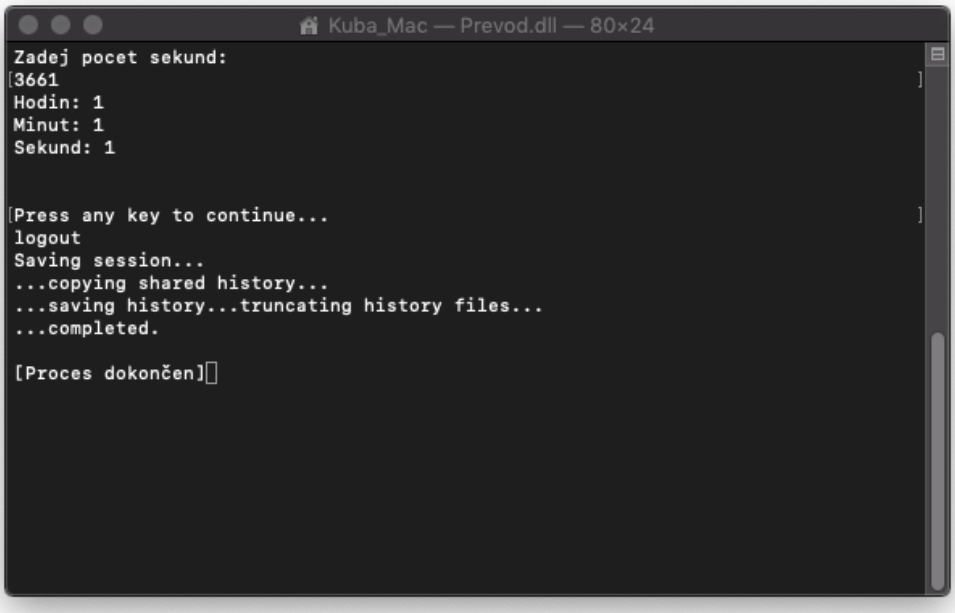
U WPF propojujeme kód s formulářem a pracujeme zde i se samostatným formulářem. Kód je pak o něco složitější a pro úplné začátečníky méně pochopitelný. Grafický výstup u WPF je přehlednější a po vizuální stránce hezčí, ale jde o to naučit studenty základům programování. WPF bych zcela jistě zmínil na přednáškách, ukázal zde i nějaký jednoduchý program, aby studenti viděli, čeho se může dosáhnout. Příkazový řádek nám ulehčuje práci, jelikož například nemusíme vždy konvertovat text na číslo a obráceně. U zadávaných hodnot musíme dodržovat posloupnosti, jaké bylo zvoleno v kódu kvůli výpisu do příkazového řádku.

Posloupnost přednášek a cvičení bych zanechal stejnou jako v případě návrhu řešení A (viz kapitola 4). Rozdíl by byl v daném obsahu. Na cvičeních by se vše vedlo formou konzolových aplikacích. Na přednáškách by se WPF ukázalo, ale záměr přednášek by byl především o seznámení s kódem samotným.

```
using System;

namespace Prevod
{
   class Program
   {
      static void Main(string[] args)
      {
         Console.WriteLine("Zadej pocet sekund: ");
         int celkem = int.Parse(Console.ReadLine());
         int hodin=celkem / 3600;
         celkem = celkem % 3600;
         int minut = celkem / 60;
         int sekund = celkem % 60;
         Console.WriteLine("Hodin: " +hodin);
         Console.WriteLine("Minut: " + minut);
         Console.WriteLine("Sekund: " + sekund);
         Console.ReadKey();
      }
   }
}
```

Obrázek 4.5 - Porovnání – Konzolová aplikace

Zde máme možnost vidět kód konzolové aplikace „Porovnání“. WPF tohoto programu je v návrhu řešení A. Kód je o poznání kratší a přehlednější. Konvertovat musíme pouze zadávanou hodnotu. Jedinou nevýhodou je, že pro další přepočet je nutné aplikaci spustit od začátku. Opakované přepočítávání by šlo do kódu samozřejmě napsat ovšem tím by se stal kód poměrně komplikovaný při porovnání s okolní úrovní kódu. Není zde nutnost přidávat podmínku kvůli celým číslům. Pokud uživatel zadá něco jiného než celé číslo, program sám detekuje chybu a zvýrazní řádek „int celkem = int.Parse(Console.ReadLine());“. Tuto chybu o špatně zadané vstupní hodnotě nám dokáže zpracovat i aplikace WPF, ale u WPF je mnohem důležitější si chybu příkazově ohlídat a tím přidat výstupní okna „MessageBox.Show“, která zobrazí uživateli zprávu a špatně zadané vstupní hodnotě.



Obrázek 4.6 - Výstup programu "Porovnání"

Výstup konzolové aplikace je jednoduchý. Zadává se do řádků a potvrzuje pomocí tlačítka „enter“. Odsazení či výpis do sloupců je zde také možný.

5 Zpracování podpůrných materiálů k výuce Technické informatiky

Praktická část bakalářské práce pojednává hlavně o transformaci nynějších programů do jazyku C#. Všechny konvertované programy jsou součástí přílohy. Mezi návrhem a zpracováním podpůrných materiálů je několik rozdílů. V návrhu řešení je navržena struktura posloupnosti výuky. Zde při zpracování podpůrných materiálů musí být splněny podmínky na konverzi základních programů. Při zpracovávání podpůrných programů je v práci odkazováno i na vlastní návrhy na úpravu výuky či změnu stávajících programů. K zpracování podpůrných materiálů patří i navržení nových programů, vytvoření vývojových diagramů a materiálů pro podporu konzolových aplikací.

5.1 Konverze základních programů

Programy určené ke konverzi do jazyku C# byly vybrány na základě dohody s vedoucím a konzultantem bakalářské práce. Celkem bylo nutné přepracovat 13 programů, které budou základním výstupem této bakalářské práce. Jednotlivé programy jsou doplněny o komentáře ve formě: „kód; // komentář“. Komentáře byly přidány na základě usnadnění pochopení problematiky pro studenty. Komentáře vysvětlují, co příkazy dělají a proč byly použity. Dále jsou k programům vytvořeny vývojové diagramy, které slouží k lepšímu pochopení dané problematiky celého programu. Vývojové diagramy jsou podrobněji popsány níže.

Program s příslušným komentářem a vývojovým diagramem by měl být současně promítán na cvičení. Většina programů byla zachována ze současného stavu i kvůli zamezení zvýšení náročnosti.

Cvičení	Obsah
1.	Ahoj světe
2.	Převod sekund na hodiny, minuty a sekundy
3.	Výčetka platidel
4	Program na výpočet objemu a povrchu koule pomocí procedur a funkcí
5.	Výběr většího ze dvou, tří nebo čtyř čísel
6.	Výpočet kvadratické rovnice
7.	Program na cykly a výpočet faktoriálu
8.	Výpis prvních 20 prvočísel
9.	Převod na binární a hexadecimální soustavu
10.	Odstranění diakritiky z textu
11.	Vícerozměrná pole
12.	Generické listy

Tabulka 5-1 - Upravený seznam programů podle výuky v C#

U jedenáctého cvičení jsme nahradili jednorozměrná pole za vícerozměrná a do dvanáctého cvičení by měly být přidány generické listy. Tento výstup cvičení bude základním odrazem pro přestavbu předmětu TI. V Delphi jsou některé texty vypisovány do „Label“ na rozdíl od C#, kde jsme zvolili vypisování textu pouze do „TextBox“ a s „Label“ pracujeme pouze jako s nadpisem nebo popisem určitého prvku. Až do devátého cvičení jsou kódy velmi podobné současnému stavu. U devátého cvičení je rozdíl v práci s ASCII kódem.

```
for (i = 1; i <= p; i++)
        {
           c = n % d;
           if (c < 10)
           {
              s = (char)(c +(char)'0')+s;
           }
           else s = (char)(c - 10+(char)'A') + s;
           if (i % 4 == 0) s = " " + s;
           n = n / d;
        }
        if (n > 0) s = s + " preteceni";
        return s;
     }
```

Tabulka 5-2 - Převod čísel v C#

Práci s ASCII kódem v C# je nutné definovat v závorkách, a to i pro samostatné znaky. ASCII je v podstatě kódová tabulka, která definuje znaky anglické abecedy a jiné znaky používané v informatice. Jde o historicky nejúspěšnější znakovou sadu, ze které vychází většina současných standardů pro kódování textu. [7]

```
for i := 1 to p do begin
  c:= n mod d;
  if c<10 then
    s:= chr(c + ord('0'))+s
  else
    s:= chr(c -10 + ord('A'))+s;
  if i mod 4 = 0 then
        s:= ' ' + s;
        n:= n div d;
end;
if n>0 then
  s:= s + ' preteceni';
Prevod:= s;
```

Tabulka 5-3 - Převod čísel v jazyce Delphi

Při přepracování programu týkající se odstranění diakritiky už je rozdíl v kódech mnohem větší, jelikož C# pracuje s textem jinak než Delphi. Zadávaný text bylo nutné převést na pole znaků, a to za pomoci příkazu „char [] r = s.ToCharArray();“, kde si vytvoříme novou proměnou „r“ o délce „s“ uživatelem zadaného textu. Proměná „r“ je ve formě pole znaků. Následuje odstranění znaků přes dva cykly „for“. Pro zpětné vypsání je nutné přepsané pole znaků převést na text. Vytvoříme si tedy další proměnou „t“ typu „string“ a pomocí cyklu načteme pole znaků. Příkaz pro naplnění je „t += r[y];“.

```
        static string BezDia(string s)
            {
                string sdia = "ÁáČčĎďÉéĚěÍíŇňÓóŘřŠšŤťúůÚÝýŽž";
                string bdia = "AaCcDdEeEeIiNnooRrSsTtuuUYyZz";
                char[] r = s.ToCharArray(); // ulozeni stringu do nove promenne do pole
    znaku
                for (int i = 0; i < s.Length; i++)
                {
                   for (int j = 0; j < sdia.Length; j++)
                   {
                      if (s[i] == sdia[j]) r[i] = bdia[j];
                   }
                }
                string t = "";
                for (int y = 0; y < r.Length; y++)
                {
                   t += r[y]; // vytvoreni stringu z pole znaku
                }
                return t; // vracime upraveny text t
            }
```

Tabulka 5-4 - Diakritika C#

Rozdíl mezi kódy je vidět na první pohled. Jazyk C# ukazoval chybu při psaní kódu „s[i]=BDia[j]“. Chyba spočívá v tom, že oproti jazyku Delphi nedokáže C# pracovat s typem „string“ jako s polem a postupně ho procházet a přepisovat. Pro vyřešení toho problému bylo více řešení. Vytvoření nové proměné „r“ jako pole znaků bylo nejjednodušší a pro studenty nijak komplikované.

```
        procedure BezDia(var s:string);
        const
           SDia = 'ÁáČčĎďÉéĚěÍíŇňÓóŘřŠšŤťúůÚÝýŽž';
           BDia = 'AaCcDdEeEeIiNnooRrSsTtuuUYyZz';

        var
           i,j:integer;
        begin
          for i := 1 to Length(s) do
            for j := 1 to Length(SDia) do
              if s[i] = SDia[j] then s[i]:= BDia[j]
        end;
```

Tabulka 5-5 - Diakritika Delphi

5.2 Návrh nových programů

Zde se vychází z návrhu cvičení v návrhu A (viz kapitola 4). Předmět TI je vyučován na Fakultě strojní, tedy by bylo vhodné ukázat studentům programy, které jsou bližší strojírenství. Jedním z navrhovaných je program na úpravu materiálu podle drsnosti. Program je lehký a snadný na procvičení podmínek „if“, „else if“ a „else“ nebo „switch“. Tento program by mohl nahradit současný program výčetka platidel, který se probírá na třetím cvičení. Z hlediska programování je tento program mnohem lepší, jelikož v programu výčetka platidel se opakují věci z programu vypracovaného na předchozím cvičení.

```
class Program
{
   static void Main(string[] args)
   {
      Console.WriteLine("Zadej drsnost povrchu Ra v rozmezi (0,012-400):");
      double x=double.Parse(Console.ReadLine());

      if ((x >= 0.012) && (x <= 0.8))
         Console.WriteLine("Dokoncovaci metody (brouseni, lapovani,...)");

      else if ((x > 0.8) && (x <= 12.5))
         Console.WriteLine("Bezne obrabeni (soustruzeni, frezovani, vrtani,...)");

      else if ((x > 12.5) && (x <= 400))
         Console.WriteLine("Povrch polotovaru (vykovky, odlitky,...)");

      else
      {
         Console.WriteLine("Spatne zadani drsnosti");
      }

      Console.WriteLine("Diky za pouziti");
      Console.ReadKey();
   }
}
}
```

Obrázek 5.1 – Drsnosti – C# - nový program

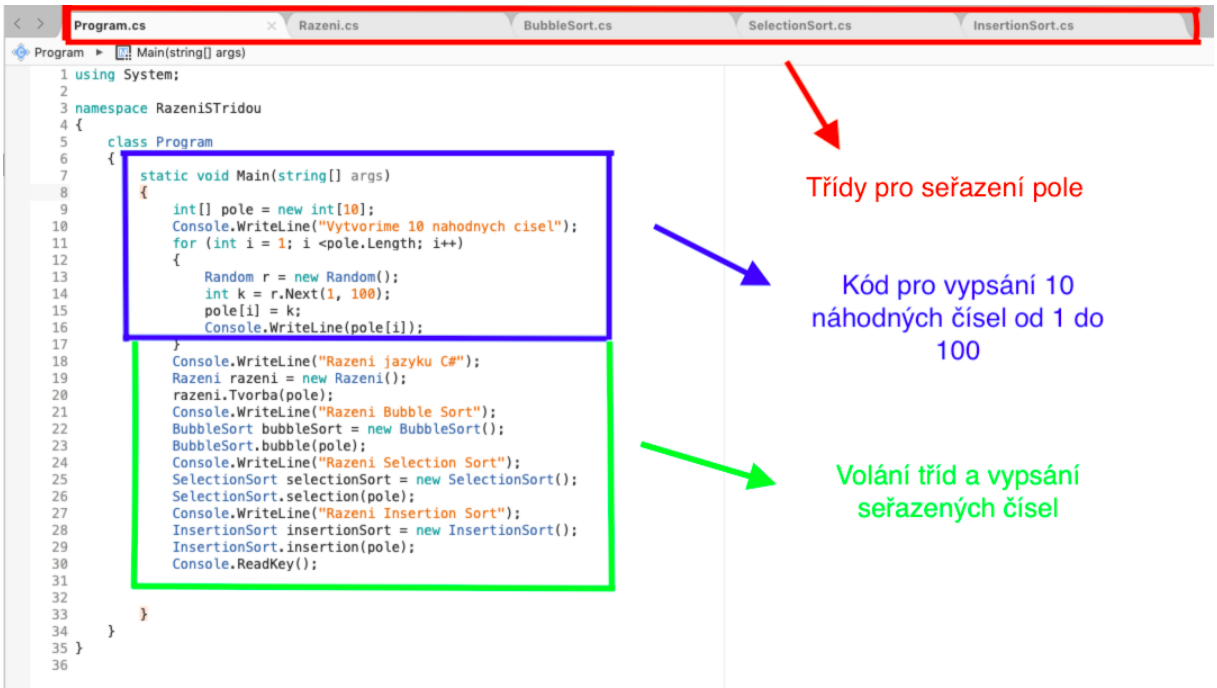
Uživatel zadá hodnotu. Pokud hodnota bude v daném rozmezí, program vypíše, jaká metoda obrábění se hodí, a to za pomoci větvení podmínek. Jestli uživatel zadá jinou hodnotu, než je v rozmezí, program vypíše „Spatne zadani drsnosti“. Drsnosti bych zařadil do třetího cvičení. Na programu se výborně procvičí podmínky a zápis podmínek do závorek. Dále je program blízký strojírenství a je možné, že studentům ukáže podstatu programování, a tedy i ulehčení práce. Nahrazení stávajícího programu výčetka platidel by přinesla mnohá zlepšení (důkladné procvičení podmínek, krátký kód, velmi dobře pochopitelný). V případě WPF by program měl dvě textová pole pro načtení zadaného čísla a vypsání výsledné metody, dvě tlačítka pro spuštění a ukončení programu.

Na první pohled by se mohlo zdát, že by nám v programu Drsnosti usnadnil práci „switch“ ovšem zde bychom také museli pracovat s množinou pomocí matematických operátorů. Program na procvičení „switch“ je zařazen na třetí cvičení (viz Tabulka 4-2). Jazyk Delphi pracovat s podmínkou „switch“ přes množiny neumí, takže je možné ukázat tento program jako výhodu C#.

```
switch (x)
{
   case 0 when (x>=0.012 && x <= 0.8 ):
      Console.WriteLine("Dokoncovaci metody (brouseni, lapovani,...)");
      break;

   case 1 when ((x > 0.8) && (x <= 12.5)):
      Console.WriteLine("Bezne obrabeni (soustruzeni, frezovani, vrtani,...)");
      break;
   case 2 when ((x > 12.5) && (x <= 400)):
      Console.WriteLine("Povrch polotovaru (vykovky, odlitky,...)");
      break;
   case 3 when (x>400):
      Console.WriteLine("Spatne zadani rozsahu");
      break;
}
```

Tabulka 5-6 - Nový program Drsnosti - "switch"

Podle návrhu A byl vytvořen program, který by mohl zahrnout deváté, desáté a dvanácté cvičení. Pokud je program v celku, tak je nepřehledný, ale díky volání tříd by šel rozdělit do více cvičení. Program vygeneruje pole o deseti náhodných číslech pomocí funkci „Random“. Pomocí cyklu „for“ vypíše všechny hodnoty pole. Dále do programu pomocí „tříd“ byly vloženy čtyři druhy řazení na kterých se výborně vysvětluje programování, a co kód v daném místě přesně dělá se seřazením čísel.



Obrázek 5.2 - Řazení – C# - nový program

První řazení je „Array.Sort();“. Řazení je vytvořeno pomocí jazyka C#. Seřazení pole o deseti hodnotách je vhodné díky jednoduchosti zapsání do programu. Při komplikovanějším programování je řazení nevyhovující kvůli jeho rychlosti. Řazení zabírá paměť a program se tedy načítá a běží pomaleji.

```
using System;
namespace RazeniSTridou
{
   public class Razeni
   {
      public void Tvorba(int[] pole)
      {
         Console.WriteLine("Serazeny pole");
         Array.Sort(pole);
         for (int j = 1; j < pole.Length; j++)
         {
            Console.WriteLine(pole[j]);

         }
      }
   }
}
```

Obrázek 5.3 - Řazení - C#

Druhé řazení se nazývá „Bubble Sort“. Algoritmus nemá příliš dobré vlastnosti, ale má velice zajímavý průběh, který jde dobře vysvětlit. Probíhá ve vlnách, kdy pokaždé propadne „nejtěžší“ prvek na konec. Funguje na porovnání vedlejších prvků a v případě, že je levý prvek větší než pravý, prvky prohodí. [8]

```
using System;
namespace RazeniSTridou
{
   public class BubbleSort
   {
      public static void bubble(int[] pole)
      {
         Console.WriteLine("Serazeny pole");
         int j = pole.Length - 2, temp;
         // kontrola prohozeni
         bool prohozeni = true;
         while (prohozeni)
         {
            prohozeni = false;
            for (int i = 0; i <= j; i++)
            {
               // prohozeni
               if (pole[i] > pole[i + 1])
               {
                  temp = pole[i];
                  pole[i] = pole[i + 1];
                  pole[i + 1] = temp;
                  prohozeni = true;
               }
            }
            j--;
         }
         for (int k = 1; k < pole.Length; k++)
         {
            Console.WriteLine(pole[k]);

         }
      }
   }
}
```

Obrázek 5.4 - Řazení BubbleSort – C#

Další řazení je „Selection Sort“. Řadí se mezi nejjednodušší řadící algoritmus. Funguje na základě nalezení minima (či maxima), které přesune na začátek pole. Následuje opětovné projetí pole, avšak už neprochází dříve nalezené minimum. Po dostatečném počtu cyklů je pole seřazené. Nevýhodou algoritmu je jeho časová složitost a stabilita. [9]

```
using System;
namespace RazeniSTridou
{
   public class SelectionSort
   {
      public static void selection(int[] pole)
      {
         Console.WriteLine("Serazeny pole");
         int temp, min;
         for (int i = 0; i < (pole.Length - 1); i++)
         {
            min = pole.Length - 1;
            // hledani minima
            for (int j = i; j < (pole.Length - 1); j++)
               if (pole[min] > pole[j])
                  min = j;
            // prohozeni prvku
            temp = pole[min];
            pole[min] = pole[i];
            pole[i] = temp;
         }
         for (int k = 1; k < pole.Length; k++)
         {
            Console.WriteLine(pole[k]);

         }
      }
   }
}
```

Obrázek 5.5 - Řazení SelectionSort – C#

Posledním z řazení je „Insertion Sort“. Algoritmus je jednoduchý, chová se inteligentně na již předtříděných polích a kódově je oproti „Bubble Sort“ a „Selection Sort“ kratší. Řazení si pole rozdělí na dvě části, a to na setříděnou a nesetříděnou. Vybírá prvky z nesetříděného pole a řadí je do pole setříděného, tak aby pole zůstalo setříděné. Algoritmus má velice dobrou časovou složitost díky tomu, že prvky vkládá přesně tam, kam patří a nedělá zbytečné kroky navíc. [10]

```
using System;
namespace RazeniSTridou
{
   public class InsertionSort
   {
      public static void insertion(int[] pole)
      {
         Console.WriteLine("Serazeny pole");
         int item, j;
         for (int i = 1; i <= (pole.Length - 1); i++)
         {
            // ulozeni prvku
            item = pole[i];
            j = i - 1;
            while ((j >= 0) && (pole[j] > item))
            {
               pole[j + 1] = pole[j];
               j--;
            }
            pole[j + 1] = item;
         }
         for (int k = 1; k < pole.Length; k++)
         {
            Console.WriteLine(pole[k]);

         }
      }
   }
}
```

Obrázek 5.6 - Řazení InsertionSort – C#

Řazení nebyly přidány z důvodu porovnání časové složitosti, ale z důvodu pochopení problematiky. K řazením jsou na internetu vytvořeny i videa, které znázorňují pohyb prvků při řazení. Videa by mohla pomoci k lepšímu pochopení.

Poslední a velmi krátký navržený program je zacyklení dvou cyklů do sebe. Tento program je opravdu krátký, ale studentovi velmi pomůže pochopit logiku cyklů a jak cyklus funguje a probíhá. Jednoduše řečeno, první cyklus se spustí s hodnotou „j=1“ a v tomto cyklu se přejde do druhého vnořeného cyklu, kde je přiřazena hodnota „i=1“. Druhý cyklus vypíše „1*1“, odřádkuje a startuje znovu, s tím rozdílem, že hodnota „i=2“. Vypíše „2*1“ a odřádkuje. Takto probíhá druhý cyklus až do doby, kdy je hodnota „i=10“, poté se vracíme zpět do prvního cyklu, kde se nastaví hodnota „j=2“ a projíždí cyklus znovu. Oba cykly končí, když hodnoty „i“ a „j“ jsou rovny deseti.

```
using System;

namespace Nasobilka_Dva_Cykly
{
   class Program
   {
      static void Main(string[] args)
      {
         Console.WriteLine("Mala nasobilka pomoci dvou cyklu:");
         for (int j = 1; j <= 10; j++)
         {
            for (int i = 1; i <= 10; i++)
               Console.Write("{0} ", i * j);
            Console.WriteLine();
         }
         Console.ReadKey();
      }
   }
}
```

Obrázek 5.7 - Nový program "Násobilka"

Většina těchto navržených programů by měla souhlasit s návrhem řešení A. Všechny nově navržené programy jsou pouze ve formě konzolové aplikace a bude záležet na kantorech, zda využijí tyto případné navržené změny programů. Zakomponování programů do navrhované výuky podle tabulky (viz Tabulka 2-2) by nemělo být těžké a nahrazení novým příkladem by mělo zapadat do posloupnosti výuky. Násobilka není program, který by měl být rozvržen na celé cvičení.

5.3 Konzolové aplikace

K zpracování podpůrných materiálů patří i návrh na vedení cvičení ve formě konzolových aplikací, které by mohlo vést ke zlepšení výuky. WPF je v dnešní době určitě velmi používané, ale je otázka, proč učit studenty, kteří programování nikdy neviděli, právě ve WPF. Pro studenta může být hezké vidět výstup, ale zase WPF je pouze pro Windows a tím pádem i když se studentovi tvorba GUI spolu s kódem zalíbí, nemusí vést k WPF, jelikož je v dnešní době na trhu obrovská možnost výběru z jiných jazyků a jiných programovacích aplikací. Programátor webových stránek, systémů, 3D her, databází, atd. Možností je opravdu mnoho a výuka WPF za účelem pouze výstupu GUI dává smysl v případě, kdy student zná základy programování. Po prostudování různých webových stránek na toto téma je možná vhodnější studentům poskytnout GUI kostru hotovou. Student zde uvidí výstup aplikace bez toho, aniž by modifikoval XAML kód a může se zaměřit na čistě logické programování. Spojení kódu s GUI lze udělat pomocí „Data Binding“. Dvě proměnné pod různým názvem lze propojit. Student by tedy psal kód, kde by měl zadáno pomocí komentářů, jak pojmenovat dané proměnné. Výsledkem bude, že student lépe pochopí základy programování a logiku, která je pro strojaře důležitější než grafický výstup. Zároveň studenti uvidí, jak se GUI mění v závislosti s tím, co napíší do kódu.

Další otázkou je, zda vůbec potřebují vidět na cvičeních GUI. Na přednáškách je možno ukázat většinu různých výstupů. Proč by měl student umět programovat XAML kód. XAML kód se na cvičeních nestíhá probrat a studenti vytvářejí GUI pomocí panelu prvků, který se při pokročilejším programování nepoužívá. Znamená to tedy, že studenti vlastně ví a neví, jak

pracovat s GUI. Přidání XAML kódu do cvičení je časově nerealizovatelné. Tím se dostáváme k otázce, proč nevyužít při programování pouze konzolové aplikace. Student díky tomu pochopí lépe logiku. Při cvičení se pak nestane, že by se někdo zasekl už při tvoření GUI. Vyučující by musel řešit případné opravy a tím by se brzdila výuka. Zároveň bude více času na probrání základů programování. Kód je přehlednější, jelikož se nemusíme odvolávat na jednotlivá tlačítka.

V popisu předmětu TI jsou základních pojmů výpočetní techniky a algoritmizace. Dle mého zde není potřeba vyučovat WPF. WPF by dávalo smysl na předmětu, který není povinný v rámci studia bakaláře na Fakultě strojní. Na tento předmět by se přihlásili studenti, které programování opravdu zajímá a mají snahu se zlepšovat v tomto ohledu. Tím se, ale dostáváme k tomu, že takoví studenti s největší pravděpodobností vůbec nenastoupí na strojní fakultu, ale raději zvolí Fakultu aplikovaných věd. Z toho vyplývá, že z celého ročníku začne programování bavit velmi malou část ze všech studentů. Navíc programování je záležitost, kdy si můžete zapsat kurz, předmět nebo se přímo přihlásit na školu programování, ale základ je si sám doma programovat a věnovat se programování naplno. Tímto textem nechci jít proti názoru katedry na WPF. Rád bych pomocí této práce na tuto problematiku poukázal. Přestavba předmětu bude tvořena více učiteli a třeba tento názor někoho zaujme a začne ho prosazovat také. Tento názor jsem si ověřil na základě studia jiných programovacích předmětů a konfrontoval jej se znalostmi jiných programátorů.

5.4 Vývojové diagramy

Vývojový diagram je druh diagramu, který slouží ke grafickému znázornění jednotlivých kroků algoritmu, pracovního postupu nebo nějakého procesu. Vývojový diagram obsahuje obrazce různého tvaru, navzájem propojené pomocí šipek. Obrazce reprezentují jednotlivé kroky, šipky tok řízení. Vývojové diagramy se rozdělují do čtyř skupin. Ukazují řízení toků dokumentů v systému, toků dat v systému, toků fyzické vrstvy a toků v programu v rámci systému. Symboly vývojového diagramu: šipka – určuje směr zpracování, obdélník – definuje dílčí krok zpracování, kosočtverec – větvení postupu v závislosti na splnění podmínek, obdélník se zaoblenými rohy – počátek nebo ukončení zpracování, kruh – spojka několika šipek. Začátkem dvacátého století začal průmyslový inženýr Allan H. Mogensen trénovat obchodníky k využití některých z nástrojů průmyslového inženýrství. [11]

5.5 Porovnání jazyků na programu Prvočísla

V této sekci je poukazováno na rozdíl jazyků C# a Delphi. Rozdíl je předložen v kódové a vizuální formě. Program Prvočísla byl vybrán na základu jeho obsáhlosti. Porovnání jazyků také patří k zpracování podpůrných materiálů, jelikož je snaha o výuku v modernějším jazyce. Porovnání by mělo ukázat výhody modernějšího jazyka. Program Prvočísla je vůči TI obsáhlejší, ale pořád je to program pro začátečníky, proto zde výhody nebudou příliš znatelné.

Kód C#:

```
namespace Prvocisla_WPF
{
   public partial class MainWindow : Window
   {
      public MainWindow()
      {
         InitializeComponent();
      }

      static Boolean PrvoCislo (int n)
      {
         bool b = true;
         int i = 2;
         while (b==true && i < n)
         {
            if (n % i == 0) b = false;
            i++;
         }
         return b;
      }

      private void BtnVypis_Click(object sender, RoutedEventArgs e)
      {
         int i = 2;
         int n = 0;
         while (n <= 20)
         {
            if (PrvoCislo(i))
            {
               TxtPrvo.Text = TxtPrvo.Text + i.ToString();
               if (n != 20) TxtPrvo.Text = TxtPrvo.Text + ", ".ToString();
               n++;
            }
            i++;
         }
      }

      private void BtnKonec_Click(object sender, RoutedEventArgs e)
      {
         this.Close();
      }
   }
}
```

Obrázek 5.8 – Prvočísla - C# - WPF

Delphi kód:

```
const
  dvacet = 20;

function Prvocislo(n: Integer):Boolean;
var
  b: boolean;
  i: Integer;
begin
  b:= True;
  i:= 2;
  while (b = True) and (i<n) do begin
    if n mod i = 0 then b:= false;
    inc(i);
  end;
  Prvocislo:= b;
end;

procedure TForm1.btnTestClick(Sender: TObject);
var
  i,n: Integer;
begin
  i:= 2;
  n:=0;
  while n<= dvacet do begin
    if Prvocislo(i) then begin
    self.lblVysledek.Caption:= self.lblVysledek.Caption + IntToStr(i);
    if not(n = dvacet) then self.lblVysledek.Caption:= self.lblVysledek.Caption + ',';
      inc(n);
    end;
    inc(i);
  end;
end;

procedure TForm1.btnKonecClick(Sender: TObject);
begin
  close;
end;

end.
```

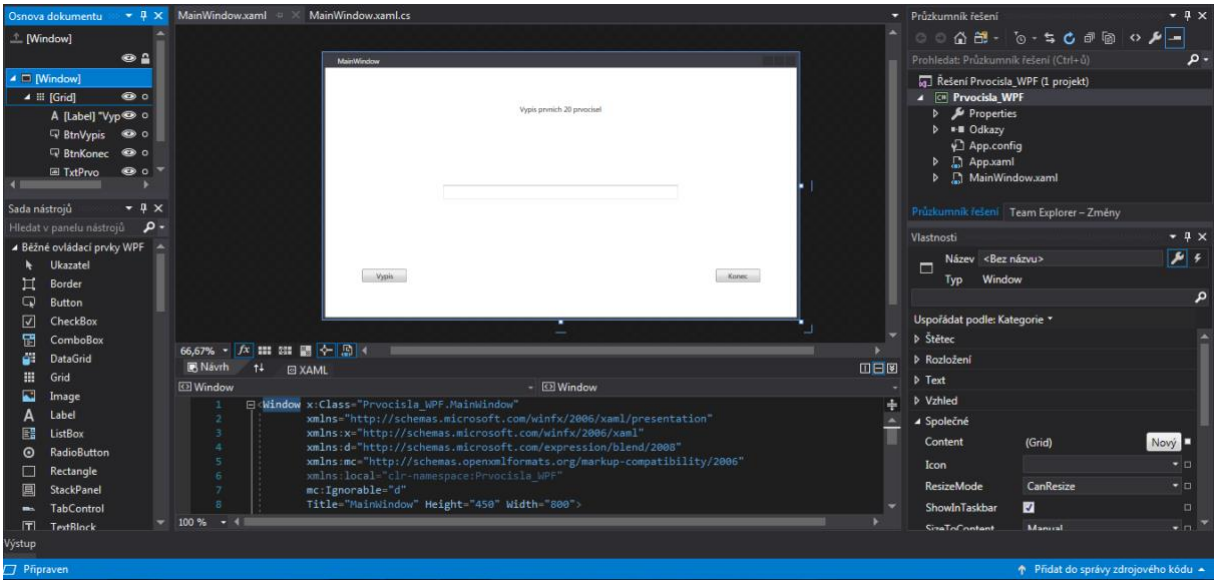
Obrázek 5.9 – Prvočísla - Object Delphi

Mezi kódy je vidět velký rozdíl v zapisování jednotlivých příkazů. Pohledově vypadá kód v jazyku C# přehledněji. V jazyku C# zcela jistě mohou za větší přehlednost příkazy k začátku a ukončení „{}“. Další kratší a přehlednější je příkaz pro výpis textu „TxtPrvo.Text = TxtPrvo.Text + i.ToString();“. Rozdíly jsou i v matematických symbolech.

V programu se využívá funkce „Boolean“, která vrací proměnnou s hodnotou „True“ nebo „False“. Proměnná „b“ se mění v závislosti na cyklu „while“ a podmínkou, kdy zbytek při dělení je roven nule. V hlavní části programu je opět využíván cyklus „while“. Pokud funkce

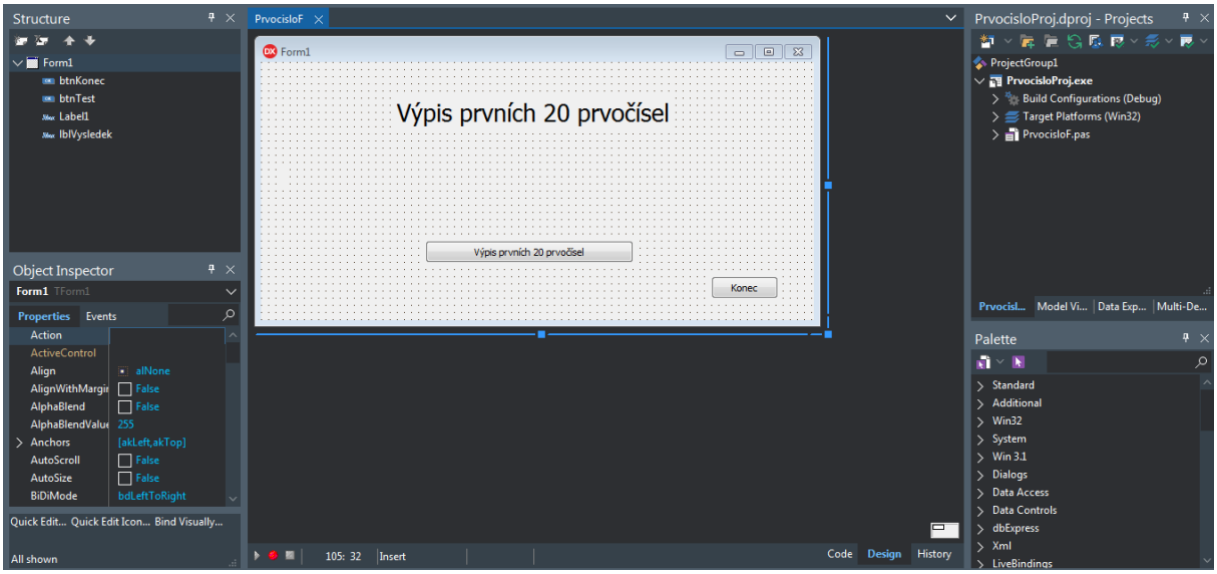
vrací proměnnou „b“ jako „True“, číslo se vypíše. Pokud číslo není rovno dvaceti, vypíše se i čárka oddělující čísla mezi sebou.

Uživatelské prostředí pro tvorbu WPF:



Obrázek 5.10 - Visual Studio – Tvorba GUI

Uživatelské prostředí Delphi:



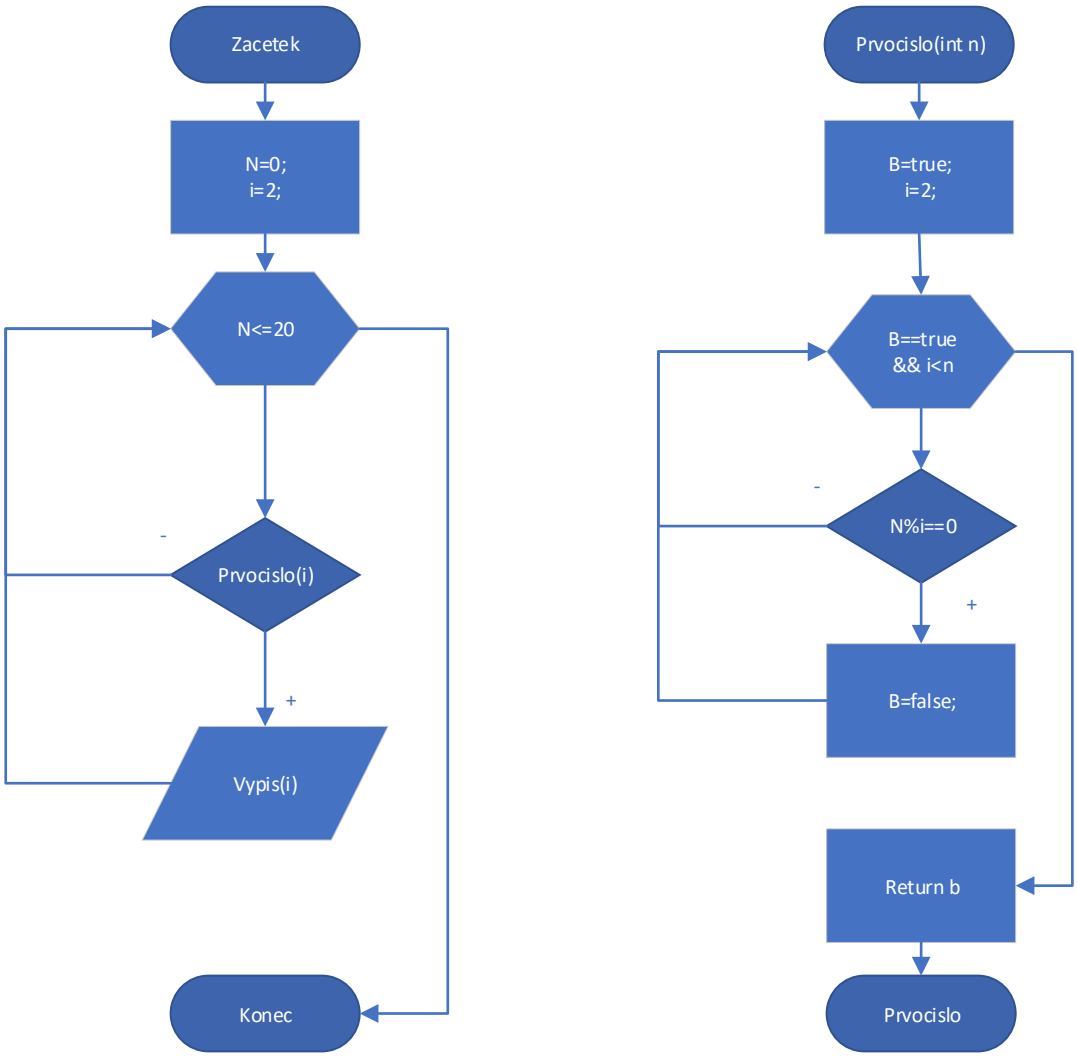
Obrázek 5.11 - Delphi – Tvorba GUI

Zde máme i porovnání prostředí ve kterých se tvoří vizuální vzhled aplikace. Pro práci začátečníka má Visual Studio lépe udělaný panel nástrojů. Velká výhoda Visual Studia je XAML kód. V Delphi tvoříme rozvržení formuláře pomocí panelu nástrojů nebo přímo v hlavním kódu. XAML kód se na cvičeních nestíhá probrat, ale na přednáškách čas je, a tedy i možnost ukázat tuto výhodu.

```
<Window x:Class="Prvocisla_WPF.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:Prvocisla_WPF"
        mc:Ignorable="d"
        Title="MainWindow" Height="450" Width="800">
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="1*"/>
            <RowDefinition Height="1*"/>
            <RowDefinition Height="1*"/>
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="1*"/>
            <ColumnDefinition Width="2*"/>
            <ColumnDefinition Width="1*"/>
        </Grid.ColumnDefinitions>
        <Label Content="Vypis prvnich 20 prvocisel" Margin="5,40"
Grid.Column="1" HorizontalContentAlignment="Center"
VerticalContentAlignment="Center" HorizontalAlignment="Center"
VerticalAlignment="Center"/>

        <Button x:Name="BtnVypis" Content="Vypis" Margin="5,40" Width="75"
Click="BtnVypis_Click" Grid.Row="2" HorizontalAlignment="Center"
VerticalAlignment="Center"/>

        <Button x:Name="BtnKonec" Content="Konec" Margin="5,40"
Width="75" Grid.Column="2" Click="BtnKonec_Click" Grid.Row="2"
HorizontalAlignment="Center" VerticalAlignment="Center"/>

        <TextBox x:Name="TxtPrvo" Height="24" Margin="1,58"
TextWrapping="Wrap" Grid.Column="1" Grid.Row="1"
VerticalAlignment="Center" HorizontalContentAlignment="Center"
VerticalContentAlignment="Center"/>

    </Grid>
</Window>
```

Tabulka 5-7 – Prvočísla - XAML kód

XAML kód slouží ke kódování prezentační vrstvy aplikace. Vychází z XML, což je značkovací jazyk, do kterého si každý může přidat své vlastní značky. Struktura dokumentu je stromová. Stromová struktura znamená, že elementy v sobě mohou obsahovat další elementy a tak dále. Každý dokument obsahuje právě jeden kořenový element. V ukázce je kořenový element „<Window>“ v něm je dále vložený „<Grid>“ a v něm dále „<TextBox>“. Elementy je nutné v XAML ukončit. V C# také existují jmenné prostory, které v sobě obsahují jednotlivé třídy. Příklad jmenného prostoru z ukázky je „xmlns“.



Obrázek 5.12 – Prvočísla - vývojový diagram

Pro ilustraci je zobrazen vývojový diagram k programu Prvočísla – viz Obrázek 5.12. V pravé části diagramu je popis funkce „Prvocislo(int n)“, která je volána v cyklu. Levá část je tedy hlavní stavba programu. Plusy a mínusy u podmínky značí, kdy podmínka je splněna a naopak.

6 Statistika

Následná statistika u vypracované bakalářské práce poskytuje přehled o velkém množství příloh, které byly zpracovány v rámci této bakalářské práce. Účelem této práce je poskytnout základní přehled o systematice tvorby nového kurzu TI a ne kompletní ukázka všech vytvořených kódů. Všechny kódy a vývojové diagramy vypracované k bakalářské práci jsou přiloženy na CD disku.

- Počet programů: 13
- Počet vývojových diagramů: 6

Programy	Počet řádek	
	XAML	Kód
Ahoj světe	29	47
Výčetka platidel	45	68
Program na výpočet objemu a povrchu koule pomocí procedur a funkcí	41	90
Výběr většího ze dvou, tří nebo čtyř čísel	34	60
Výpočet kvadratické rovnice	35	87
Program na cykly a výpočet faktoriálu	74	244
Výpis prvních 20 prvočísel	27	65
Převod na binární a hexadecimální soustavu	45	75
Odstranění diakritiky z textu	29	72
Řazení		148
Drsnosti		48
Násobilka		20
Celkem	**359**	**1024**

Tabulka 6-1 - Statistika programů

7 Závěr

V této bakalářské práci byla provedena analýza současného stavu, kde byly nastíněny mezery v posloupnosti programů na cvičení, užívání zastaralého programovacího jazyku Delphi a špatný vztah studenta k předmětu TI.

Dalším úkolem bylo najít vhodný programovací jazyk a následně nastudovat práci se SW aplikací a tvorbě kódu. Vhodným jazykem byl zvolen C# a s ním související aplikace Visual Studio. Nejvíce využívaným zdrojem byl pramen č.3 IT Network. Na základě absolvování kurzu na této webové stránce bylo z části vyhodnoceno připojení tématu ohledně konzolové aplikace. Pramen vedl i k použití některých kódů pro zlepšení výuky.

Jako další krok byl zvolen návrh řešení, který byl rozdělen na variantu „A“ a „B“. Návrh A byl zaměřen na nalezení chyb v jednotlivých přednáškách a cvičeních, kde byla následně formou tabulek ukázána vylepšená posloupnost a možnost provedení změn. Návrh B už se vydal cestou transformace celého předmětu. V druhém návrhu se poukazuje na vedení cvičení a programů ve formě konzolových aplikací. Řešení ve formě konzolových aplikací je v návrhu B pouze nastíněna. Hlavní poukázání na konzolovou aplikaci se nachází v tvorbě podpůrných materiálů.

Zpracování podpůrných materiálů se rozděluje do více částí. První část je zaměřena na konverzi programů. Jsou demonstrovány změny v kódu C# od Delphi (Pascal) a pomocí tabulky je ukázána výsledná posloupnost programů na cvičeních. Dalším záměrem bylo zlepšení či úplná náhrada některých stávajících programů podle návrhu A. V této sekci byly navrženy tři programy a určení zařazení programů do výuky v průběhu semestru. Všechny nové programy jsou psány ve formě konzolové aplikace. Další částí je zpracování základů pro přesvědčení využívat k výuce konzolové aplikace. Zde se popisují výhody konzolových aplikací, které jsou především v jednoduchosti a také popis proč vlastně programovat aplikace WPF. Poslední částí je stručný popis vývojových diagramů a rozdílů v kódu.

Tato bakalářská práce slouží jako podpora při tvorbě nových studijních materiálů na základě zadání katedry. Osobně jsem přesvědčen o vhodnosti využití konzolové aplikace při výuce. Práce rozhodně není finálním výsledkem předmětu TI. Ohledně přestavby předmětu je to ještě dlouhá cesta. Doufám, že tato práce pomůže při restrukturalizace předmětu a nastíní učitelům možný nový obsah předmětu TI.

8 Bibliografie

[1] Hodnocení učebnic C# [online]. [cit. 2018-01-12]. Dostupné z: https://www.quora.com/What-are-the-best-C-books-for-beginners.

[2] MILES, Rob. C# Programming Yellow Book. 2014. ISBN B00HNSGM9A.

[3] Učebnice C# [online]. [cit. 2018-11-15]. Dostupné z: http://freecomputerbooks.com/langCSharpBooks.ht.

[4] KLAUSEN, Poul. Introduction to programming and the C# language. Bookboon. 2013. ISBN 9788740302509.

[5] ČÁPKA, David. ITnetwork [online]. Praha: Čápka, 2017 [cit. 2018-08-24]. Dostupné z: www.ITnetwork.cz.

[6] NAKOV, Svetlin. Fundamentals of Computer Programming with C#. Sofia, 2013. ISBN 9544007733.

[7] ASCII [online]. [cit. 2019-03-04]. Dostupné z: https://cs.wikipedia.org/wiki/ASCII.

[8] ČÁPKA, David. BubbleSort [online]. [cit. 2019-04-08]. Dostupné z: https://www.itnetwork.cz/algoritmy/razeni/algoritmus-bubblesort-probublavani-trideni-cisel.

[9] ČÁPKA, David. SelectSort [online]. [cit. 2019-04-08]. Dostupné z: https://www.itnetwork.cz/algoritmy/razeni/algoritmus-selection-sort-razeni-cisel-podle-velikosti.

[10] ČÁPKA, David. InsertSort [online]. [cit. 2019-04-08]. Dostupné z: https://www.itnetwork.cz/algoritmy/razeni/algoritmus-insertion-sort-trideni-cisel-podle-velikosti.

[11] Vývojový diagram [online]. [cit. 2019-03-10]. Dostupné z: https://cs.wikipedia.org/wiki/V%C3%BDvojov%C3%BD_diagram.

[12] TIOBE [online]. [cit. 2018-09-20]. Dostupné z: https://www.tiobe.com/tiobe-index/.