Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Programování řídicích aplikací

Studijní opora

Ing. Libor Havlíček, Ph.D.

Programování řídicích aplikací

Téma 01: Úvod do předmětu a organizace výuky

Studijní cíl

Seznámit studenty s organizačními záležitostmi předmětu, jsou uvedeny podmínky pro získání zápočtu z předmětu a je provedeno seznámení studentů s požadavky a průběhem vykonání zkoušky z předmětu.

Doba nutná k nastudování

2 hodiny

Klíčová slova

Semestrální práce, harmonogram, požadavky, přednáška, cvičení, softwarové prostředky, hardware, protokol, technická zpráva

1 Úvod do předmětu

1.1 Organizace výuky

Výuka předmětu je organizována do jednotlivých výukových týdnů příslušného semestru.

- Výuka je rozdělena na přednášky a cvičení, které se, z důvodu zpracované náplně, příslušně prolínají. Z uvedeného důvodu je vhodná účast studentů na přednáškách, kde se studenti dovědí podrobnější informace k řešení problémových okruhů, následně řešených na cvičeních.
- Počet přednášek a cvičení bude pokrývat výukové období celého semestru.
- Cvičení jsou povinná.
- Zadání jednotlivých cvičení bude dostupné v elektronické podobě v informačním systému STAG [případně MOODLE], kde budou postupně zveřejňovány.
- Z každého cvičení se odevzdává technická zpráva [TZ]. Technické zprávy [protokoly] s absolvovaným cvičením budou odevzdávány ihned následující týden, při následujícím cvičení z předmětu.
- V průběhu semestru každá pracovní skupina zpracuje semestrální práci [SP], jejíž zadání obdrží v první třetině semestru.
- Požadavkem k získání zápočtu je odevzdání řádně zpracovaných protokolů ze cvičení a kompletní řešení SP, zpracované dle níže uvedených požadavků na její zpracování.

1.2 Přednášky

Tematické přednášky budou úzce navázány na tematické okruhy, řešené na cvičeních. Přednášky mohou být kumulovány do tematických bloků, které budou zahrnovat více tematických okruhů, shrnutých do menšího počtu ucelených přednášek. Takto vzniklý časový prostor bude věnován cvičením z předmětu a zpracování SP.

1.3 Cvičení

Cvičení zpracují studenti v pracovních skupinách, ve dvojicích, které budou určeny na prvním cvičení z předmětu. Náplň jednotlivých cvičení je převážně zaměřena na tvorbu aplikací s grafickým uživatelským rozhraním v jazyce C#. Každá úloha je částečně zpracována, ve formě návodu ke cvičení. Pokud je součástí cvičení samostatná práce, vychází z návodu k příslušnému cvičení. Bližší informace dostanou studenti v průběhu předcházející přednášky.

1.4 Příprava na cvičení

Pro přípravu a zpracování jednotlivých cvičení si studenti tvoří poznámky do sešitu, optimálně čtverečkovaného, formátu A4. Jedná se o poznámky k tvorbě programového kódu, vývojové diagramy vytvářených programů, případně schémata zapojení hardware úlohy a zdrojového firmware mikropočítače. Návrhy vývojových diagramů, blokových schémat a schémat zapojení vytvářejí studenti do sešitu „ručně“. Kompletní podklady pro zpracování každého zadání cvičení a jejím úspěšném absolvování, si nechte vaše poznámky v sešitu podepsat od vyučujícího. Bez podepsaného řešení nebude odevzdání úlohy akceptováno. Udělení zápočtu bude podmíněno odevzdáním kompletního souboru protokolů ze všech cvičení, v požadovaných termínech. Protokoly budete odevzdávat podle stanoveného harmonogramu. Termín odevzdání úlohy je zpravidla, pokud není vyučujícím stanoveno jinak, jeden týden od jejího zadání. To znamená, v okamžiku zadání následující úlohy.

1.5 Formát protokolů

Protokoly se zpracováním jednotlivých úloh (včetně příslušných příloh), odevzdáte pouze v elektronické podobě (vlastní text protokolu v *.pdf, nebo *.doc formátu), v požadovaných termínech, v jednom, komprimovaném, „*.zip“, souboru. Jeho název uveďte v následujícím, jednotném formátu pro označení souborů, který je nutné bezpodmínečně dodržet:

„BPRA_sk_XX_ul_c_xx_DD_MM_RRRR“

[DD_MM_RRRR je datum odevzdání protokolu“]

Komprimovaný soubor s příslušnou úlohou odevzdáte vyučujícímu během probíhající výuky cvičení z předmětu.

1.6 Získání zápočtu z předmětu

Pro získání zápočtu je nutné mít odevzdané kompletní soubory protokolů, a to v požadovaných termínech, v rámci výukového období semestru (poslední možnost odevzdání závěrečného protokolu, je poslední termín konání výuky předmětu v semestru. U zápočtu budou tedy řešeny otázky, týkající se řešení zadání jednotlivých úloh cvičení a zpracování protokolů.

1.7 Obsah protokolů

- zadání úlohy (ne kopie originálního zadání ve formě obrázku pořízeného metodou „Print Screen“, ale srozumitelný výtah z tohoto zadání, protokol s typem tohoto zadání nebude přijat a úloha nebude považována za splněnou, bez možnosti opravy!)
- vlastnoručně překreslená schémata (např. v prostředí „wordu“, nebo návrhového CAD software pro elektroniku-„Eagle“ v7.x).
- vlastní postup řešení a zpracování úlohy,
- vývojové diagramy a dostatečně okomentované fragmenty zdrojového kódu mikropočítače a aplikace vytvořené ve Visual Studiu.
- případné vzorové výpočty použité k řešení zadaného úkolu,
- závěr (věcné zhodnocení výsledků měření úlohy).

1.8 Doplňující informace k formátu a přílohám odevzdávaných protokolů

Pro úvodní list jednotlivých protokolů použijte předloženou šablonu formátu první strany. Podle předlohy vytvořte první stranu v software „Word“, kam doplňte potřebné údaje, včetně čísla a názvu aktuálně zpracovávané úlohy. Protokoly zpracujte co nejpečlivěji. Neúplné protokoly a protokoly s nedbalým zpracováním nebudou vyučujícím přijaty, bez možnosti udělení zápočtu. Při zjištění drobných nedostatků v protokolech, budou protokoly vráceny autorům, kteří budou vyzváni k jejich odstranění (bude akceptována pouze jedna oprava každého protokolu). Přílohou vlastního textu protokolu budou zdrojové soubory programů aplikací z VS (kompletní projekty), vývojového prostředí pro mikropočítač a software „Eagle, verze 7.1 - 7.4 (soubory s příponou *.brd a *.sch, patřičně pojmenované tak aby název vystihoval název úlohy, nebo jiné vhodné identifikační údaje), případně další soubory, vzniklé při řešení cvičení.

1.9 Zadání semestrální práce – požadavky na řešení a požadované výstupy

Ve skupinách, případně samostatně, zpracujte zadané téma semestrálního projektu – návrhu zapojení elektronického obvodu (zařízení) s jednočipovým mikropočítačem. Celý projekt bude zpracován kompletně od návrhu schéma původního zapojení, přes návrh desky plošného spoje, až po realizaci zapojení a software (firmware) použitého jednočipového mikropočítače. Návrh schéma zapojení a desky plošného spoje proveďte v návrhovém CAD software „Eagle“, volně šiřitelné verze 7.X.X. V případě použití novější verze software „Eagle“ musíte zajistit

kompatibilitu s verzí 7.X.X. Příslušný firmware jednočipového mikropočítače proveďte v prostředí „Arduino IDE“. Program mikropočítače „odladťe“ a prezentujte jeho funkčnost vyučujícímu na vlastním vývojovém kitu Arduino UNO (nebo jeho klonu) s mikropočítačem ATmega328P. Vývojové prostředky budete mít k dispozici v laboratoři specializace.

1.10 Postup zpracování SP

- Nastudujte funkci přiděleného elektronického zařízení ze zadání;
- sestavte, v písemné formě, požadavky na funkce vaší aplikace. Identifikujte cíle řešení a požadavky na vstupní informace, potřebné pro korektní funkci vaší aplikace;
- nakreslete blokové schéma zapojení elektronického obvodu vaší aplikace a sepište podrobný popis její funkce;
- vytvořte soubor výrobních podkladů pro realizaci originálního zapojení vaší aplikace podle podkladů originální dokumentace (schémata, návrh plošného spoje podle originálu atd.);
- navrhněte potřebnou modifikaci zapojení laboratorního pracoviště pro testování funkčnosti popisované aplikace s využitím vývojového kitu „Arduino“. Pokud budou potřeba nějaké doplňující elektronické komponenty, sestavte jejich seznam a ten předejte vyučujícímu v dostatečném časovém předstihu;
- sestavte vývojový diagram firmware mikropočítače a sepište podrobný popis jeho funkce;
- realizujte příslušný firmware aplikace v jazyce C, v prostředí Arduino IDE, a provádějte průběžné testování jeho funkčnosti;
- proveďte testování kompletní realizace aplikace a výsledky uveďte do protokolu;
- funkční testovací aplikaci předveďte vyučujícímu. Po schválení řešení vyučujícím vytvořte konečnou podobu protokolu ve formě technické zprávy a tu odevzdejte vyučujícímu.

Poznámka: Začněte pracovat ihned po obdržení zadání. V průběhu semestru průběžně tvořte text vaší SP.

1.11 Výstupy zpracování SP

Schéma zapojení originálního zařízení s mikropočítačem (viz. individuální zadání);

- návrh desky plošného spoje originálního zapojení;
- návrh zapojení pro testování aplikace na kitu „Arduino“;
- zdrojový kód firmware pro mikropočítač vašeho zařízení vytvořený v jazyce C v „Arduino IDE“;
- Software (modifikovaný) pro prezentaci vašeho řešení na „Arduino UNO“ (funkčnost zařízení je nutné předvést vyučujícímu, případné požadavky na doplňkové součástky

vývojového kitu „Arduino UNO“ konzultujte s vyučujícím v dostatečném předstihu). Funkční firmware mikropočítače je nutnou podmínkou pro udělení zápočtu!

- *.Pdf., nebo *.Doc (*. Docx) soubor s textem vaší semestrální práce ve formě technické zprávy a uživatelských návodů.
- Kompletní datové soubory řešení vaší semestrální práce (firmware mikropočítače (včetně okomentovaných zdrojových kódů, návrh schéma zapojení, návrh desky plošného spoje, plné texty protokolu, v *.doc, nebo *.pdf formátu, případně další datové soubory, které budou zmíněny a použity při řešení semestrální práce), komprimované v jednom souboru (*.rar, nebo *.zip)
- Konečnou verzi řešení semestrální práce ve formě datových souborů odevzdejte nejpozději na předposledním cvičení z předmětu. Kompletní řešení uložte do adresáře: „SP_202X_BAMR_BPRA_Sk_XX“
- Připravte prezentaci s popisem řešení vaší SP (v délce cca 5 minut v „PowerPointu“), kterou budete prezentovat v zápočtovém týdnu vyučujícímu a ostatním studentům ročníku před získáním zápočtu z předmětu.
- Úspěšné zpracování semestrální práce a její prezentace je součástí a nutnou podmínkou pro získání zápočtu z předmětu. Termín odevzdání semestrální práce je shodný s datumem předposledního cvičení výuky předmětu v aktuálním zimním semestru. Pozdější odevzdání nebude akceptováno a zápočet z předmětu nebude udělen! Za studijní skupinu se odevzdává jeden protokol a jeden datový soubor!

2 Studijní literatura

2.1 Základní

VIRIUS, Miroslav, 2002. *C# pro zelenáče.* Praha: Neocortex. ISBN 80-72321-76-5.

SELLS, Chris, 2005. *C# a Winforms: programování formulářů ve Windows.* Brno: Zoner Press. ISBN: 80-86815-25-0.

NAGEL, Ch., EVJEN, B., GLYNN, J. a SKINNER, M. W., 2007. *C# 2005 – Programujeme profesionálně.* Brno: Computer Press. ISBN 80-251-1181-4.

2.2 Doporučená

LIBERTY, Jesse, 2002. *Programming C#*. 2nd ed. Sebastopol (USA): O´Reilly & Associates, Inc. ISBN 978-0-596-00309-9.

PETZOLD, Charles, 2003. *Programování Microsoft Windows v jazyce C#*. Překlad Jan Pokorný a Pavel Vaida. Praha: SoftPress. ISBN 80-86497-54-2.

Poznámka:

Pokud máte zadání SP, které souvisí s tématem vaší BP, postupujte podle vašeho individuálního zadání SP a její postupné řešení konzultujte s vyučujícím.

3 Software a Hardware

K programování aplikací v GUI C budeme používat IDE prostředí Visual Studio 2022. Pro aplikace mikropočítače použijeme Arduino IDE. Jako hardware využijeme vybraný typ vývojového kitu Arduino s jednočipovým mikropočítačem ATmega328P, například verzi UNO, MINI, MICRO, atp.

Jako podpůrný software bude použit software „com0com“, pro vytváření virtuálních komunikačních portů osobního počítače a terminálový program „Terminal“, umožňující obousměrnou komunikaci mezi sériovým portem osobního počítače (PC) a sériovým rozhraním mikropočítače.

Tab. 1 Harmonogram výuky BPRA

Týden	Harmonogram výuky BPRA	Protokol č.
1.	Úvod do předmětu, organizace výuky	SP
2.	Programovací jazyky, jazyk C#	
3.	IDE pro programování v jazyce C#	
4.	KA, datové typy, proměnné, číselné hodnoty a formátování textu v jazyce C#	1.
5.	KA Čísla, aritmetické operace v jazyce C#	
6.	KA větvení, cykly v jazyce C#	2.
7.	WFA ovládací prvky, jejich vlastnosti a události v jazyce C#	3.
8.	WFA aritmetické operace v jazyce C#	4.
9.	WFA binární operace v jazyce C#	5.
10.	WFA sériový port počítače v jazyce C#	6.
11.	WFA ovládání digitálních pinů mikropočítače v jazyce C#	7.
12.	WFA zobrazení stavu pinů mikropočítače v jazyce C#	8.
13.	WFA senzor teploty a vlhkosti v jazyce C#	9.

KA konzolová aplikace

WFA formulářová aplikace

SP semestrální práce

Celkem je zpracováno 13 cvičení, rozložených do 13 výukových týdnů. Každá pracovní skupina odevzdá celkem 9 protokolů (technických zpráv).

Univerzita Pardubice, Fakulta elektrotechniky a informatiky,
Pardubice, Nám. Čs. Legií 565

BPRA

Katedra automatizace a matematiky

Příjmení, Jméno:	Číslo úlohy:
Příjmení, Jméno:	Datum zadání:
Akademický rok:	Datum odevzdání:
Skupina:	Klasifikace:

Název úlohy

Počet stran:

Seznam zkratek

BP	Bakalářská práce
GUI	grafické uživatelské rozhraní
IDE	integrated development environment
KA	konzolová aplikace
PC	osobní počítač
SP	semestrální práce
TZ	technická zpráva
VS	Visual Studio
WFA	Windows Forms aplikace

Rejstřík

Programování řídicích aplikací

Téma 02: Programovací jazyky, jazyk C#

Studijní cíl

Seznámit studenty se základními pojmy z tvorby programů a programování, s vývojovým prostředím „MS Visual Studio“, jeho instalací, konfigurací a použitím pro vytváření aplikací v jazyce C#.

Doba nutná k nastudování

2 hodiny

Klíčová slova

Programovací jazyk, Počítačový program, programový kód, zdrojový kód, kód, C#, algoritmus, vykonávání kódu, kompilace, interpreter

1 Úvod

Programovat znamená dávat počítači příkazy, například "zobraz hodnoty ze snímače", "zapni světlo", "vytiskni údaj na obrazovku", "přečti vstup z konzole" nebo "vypočítej hodnoty konstant PID regulátoru". Pokud se provádí zpracování příkazů jeden po druhém, můžeme takový proces nazvat počítačovým programem. Text, představující počítačový program, se potom nazývá „programový kód“ (nebo též „zdrojový kód“, nebo jen prostě „kód“).

Příklad řádku programového kódu:

```
Console.WriteLine("Zapni LED") ;
```

Který je součástí kódu:

```
using System;

class ControllerCmd
{
   public static void Main()
   {
      Console.WriteLine("Zapni LED");
   }
}
```

Příkaz vypíše na konzoli text „Zapni LED“ (bez uvozovek)

2 Počítačové programy

Počítačové programy představují posloupnost příkazů, které jsou napsány v určitém programovacím jazyce, jako je C#, C++, Java, JavaScript, Python, PHP, C nebo jiný.

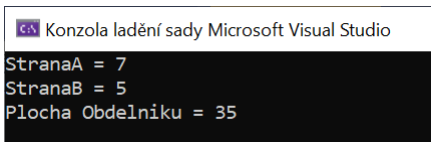
Příklad počítačového programu v C#:

```
using System;

class PlochaObdelniku
{
   public static void Main()
   {
      var StranaA = 7;
      var StranaB = 5;

      Console.WriteLine("StranaA = " + StranaA);
      Console.WriteLine("StranaB = " + StranaB);
      Console.WriteLine("Plocha Obdelniku = " + StranaA * StranaB);
   }
}
```

Výše uvedený program definuje **třídu** , která obsahuje **metodu** , která obsahuje sekvenci 5 **příkazů**: **PlochaObdelniku()**

1. Deklarace a přiřazení proměnné: var StranaA = 7;
2. Deklarace a přiřazení proměnné: var StranaB = 5;
3. Výpočet a tisk výrazu: Console.WriteLine("StranaA = " + StranaA);
4. Výpočet a tisk výrazu: Console.WriteLine("StranaB = " + StranaB);
5. Výpočet a tisk výrazu: Console.WriteLine("Plocha Obdélníku = " + StranaA * StranaB);

Výsledek (výstup) z výše uvedeného programu je následující:



Obr. 1 – Výstup programu spuštěného z příkazového řádku (printscreen) (Microsoft, 2022)

Podrobně bude rozebíráno, jak vytvářet (psát) programy v C#, proč je potřeba definovat třídu a metodu. Nyní předpokládejme, že jazyk C# vyžaduje veškerý výše uvedený kód, aby bylo možné provést sekvenci příkazů nacházejících se v Main().
Abychom mohli psát příkazy, musíme znát tzv. syntaxi a sémantiku jazyka, se kterým pracujeme. V našem případě to bude jazyk C#. Postupně se tedy seznámíme se syntaxí a sémantikou jazyka C#.

3 Algoritmy

Počítačové programy obvykle provádějí nějaký algoritmus. Algoritmy jsou sledem kroků, které jsou nezbytné pro dokončení určitého úkolu a pro získání nějakého očekávaného výsledku, něco jako "recept".

Pokud například smažíme vejce, postupujeme podle nějakého receptu (algoritmu): na pánvi zahřejeme olej, rozbijeme v něm vejce, počkáme, až se usmaží, a oddálíme je od sporáku.

Podobně při programování počítačové programy provádějí algoritmy: sekvenci příkazů, které jsou nezbytné pro dokončení určitého úkolu. Například k uspořádání posloupnosti čísel ve vzestupném pořadí je zapotřebí algoritmus, např. najít nejmenší číslo a vytisknout ho, poté najít nejmenší číslo mezi ostatními čísly a vytisknout jej, a to se opakuje, dokud nezbývají žádná další čísla.

Pro pohodlí při vytváření programů, pro psaní programovacího kódu, pro provádění programů a další operace související s programováním potřebujeme vývojové prostředí, například Visual Studio.

4 Interpreter

Některé programovací jazyky nepoužívají kompilátor a jsou **interpretovány přímo** specializovaným softwarem zvaným "interpret". **Interpret** je "**program pro provádění programů**", napsaný v nějakém programovacím jazyce. Příkazy v programu provádí jeden po druhém, protože rozumí nejen jednomu příkazu a sekvencím příkazů, ale i dalším jazykovým konstrukcím (vyhodnocení, iterace, funkce atd.). Jazyky jako Python, PHP a JavaScript pracují s interpretem a jsou spouštěny bez kompilace. Vzhledem k absenci předchozí kompilace jsou v interpretovaných jazycích chyby nalezeny během doby provádění, po spuštění programu, nikoli dříve.

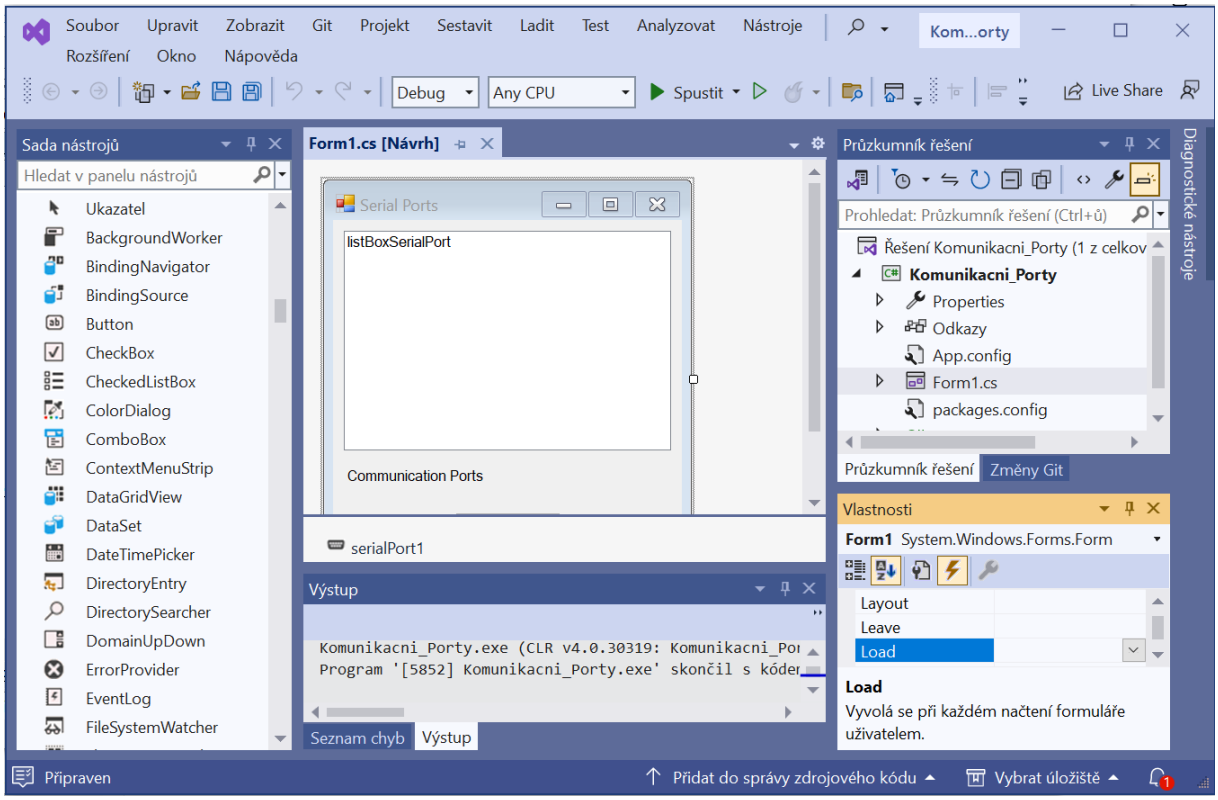
5 Kompilátor

Kompilátor překládá kód z programovacího jazyka do **strojového kódu**, protože pro každou konstrukci (příkazy) v kódu zvolí správný, předem připravený fragment strojového kódu a mezitím **kontroluje text programu, zda neobsahuje chyby**. Dohromady zkompilované fragmenty tvoří program do strojového kódu tak, jak to mikroprocesor počítače očekává. Poté, co byl program zkompilován, může být spuštěn přímo z mikroprocesoru ve spolupráci s operačním systémem. U programovacích jazyků založených na překladačích je **kompilace programu** povinná před jeho spuštěním a syntaktické chyby (nesprávné příkazy) jsou nalezeny během kompilace. Jazyky jako C++, C# a Java pracují s **kompilátorem**.

6 Vývojová prostředí

Vývojové prostředí (Integrated Development Environment – **IDE)** v sobě zahrnuje kombinaci standardních nástrojů, sloužících k vývoji softwarových aplikací. Ve vývojovém prostředí můžeme vytvářet kód naší aplikace, můžeme provádět jeho kompilaci a spouštět vytvořené programy. Vývojová prostředí v sobě spojují **editor textu**, který slouží pro tvorbu vlastního zdrojového kódu, prostředí **programovacího jazyka**, **kompilátor nebo interpret** a **runtime prostředí** pro spouštění vytvářených programů. Součástí vývojových prostředí jsou i nástroje pro „ladění“ zdrojového kódu, **debuggery, které umožňují analyzovat běh** programu a efektivní vyhledávání chyb. **Nástroje pro návrh uživatelského rozhraní, spolu s dalšími** nástroji a doplňky umožňují tvorbu aplikace s příslušným uživatelským rozhraním.

Kompletní vývojová prostředí jsou vhodná pro vývoj aplikací, protože zahrnují vše potřebné pro vývoj programu, aniž by bylo nutné, v procesu vývoje zdrojového kódu a uživatelského rozhraní, prostředí opouštět. Pokud bychom prováděli vývoj aplikace bez vývojového prostředí, což je také možné, tak bychom museli kód vytvořit v nějakém textovém editoru, provést kompilaci příkazem na konzoli, následně spustit dalším příkazem na konzoli a vytvořit další potřebné příkazy, což by bylo nejen časově náročné, ale i poněkud nepohodlné. Proto převážná většina programátorů používá ke své práci IDE.



Obr. 2 – Vzhled vývojového prostředí IDE Visual Studio (printscreen) (Microsoft, 2022)

Dá se předpokládat, že pro programování aplikací **v jazyce C#** je nejpoužívanějším IDE **Visual Studio**, které je vyvíjeno a volně distribuováno softwarovou společností Microsoft. Prostředí lze stáhnout z www adresy: https://visualstudio.microsoft.com/cs/vs/community/.

I my budeme , pro tvorbu aplikací, používat vývojové prostředí Visual Studio.

Abychom si mohli udělat představu o konkurenčních vývojových prostředích jazyka C#, lze uvést, že společně s vývojovým prostředím sady „Visual Studio“, jsou dostupné další alternativní vývojové prostředky, například:

- **Visual Studio Code** - https://code.visualstudio.com
- **Eclipse aCute** - https://projects.eclipse.org/projects/tools.acute
- **MonoDevelop / Xamarin Studio** - https://www.monodevelop.com
- **Rider** - https://www.jetbrains.com/rider

(Nakov, 2024)

7 Použitá literatura

NAKOV, C. a kol., 2024. *The Book Uses C# and Visual Studio* [online]. [cit. 20.5.2024]. Dostupné na WWW: https://csharp-book.softuni.org/Content/Chapter-1-first-steps-in-programming/exercises-graphical-and-web-apps/exercises-graphical-and-web-apps.html

MICROSOFT Corp. a. s., 2022. MS Visual Studio 2022, verze 17.8.9. 14. 5. 2024 [cit. 25. 5. 2024]. Dostupné na WWW: https://visualstudio.microsoft.com/cs/

8 Studijní literatura

VIRIUS, Miroslav, 2002. *C# pro zelenáče.* Praha: Neocortex. ISBN 80-72321-76-5.

SELLS, Chris, 2005. *C# a Winforms: programování formulářů ve Windows.* Brno: Zoner Press. ISBN: 80-86815-25-0.

NAGEL, Ch., EVJEN, B., GLYNN, J. a SKINNER, M. W., 2007. *C# 2005 – Programujeme profesionálně.* Brno: Computer Press. ISBN 80-251-1181-4.

9 Otázky k procvičení

1 Jaká je funkce interpreteru?
2 Jaká je funkce kompilátoru?
3 Co znamená zkratka GUI?
4 Co znamená zkratka IDE?
5 Jaký je význam algoritmizace?

Seznam zkratek

IDE	Integrated Development Environment
GUI	grafické uživatelské rozhraní
VS	Visual Studio

Rejstřík

Programování řídicích aplikací

Téma 03: Vývojová prostředí IDE pro programování v jazyce C#

Studijní cíl

Seznámit studenty s vývojovým prostředím „MS Visual Studio“, jeho instalací, konfigurací a použitím pro vytváření aplikací v jazyce C#.

Doba nutná k nastudování

2 hodiny

Klíčová slova

Programovací jazyk, počítačový program, programový kód, zdrojový kód, kód, C#, algoritmus, vykonávání kódu, kompilace, interpreter

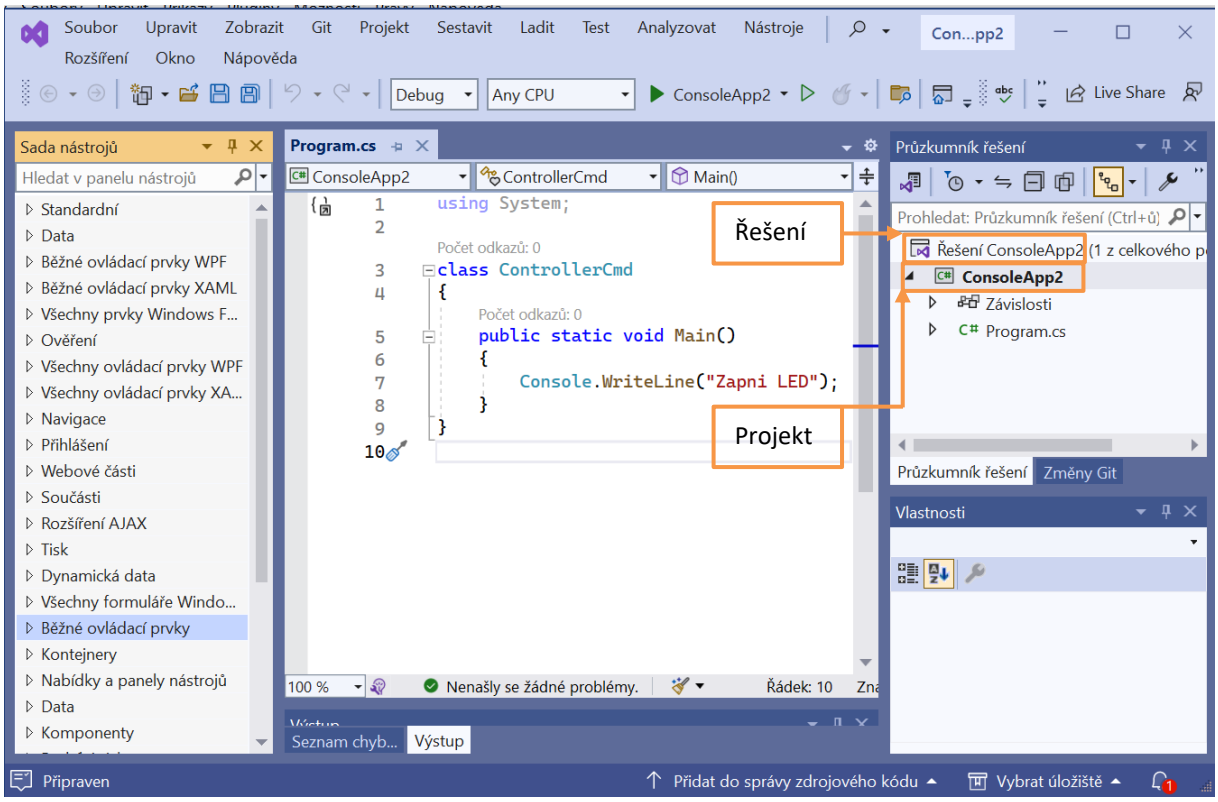
1 Projektová řešení a projekty v aplikaci Visual Studio

Než začneme pracovat s Visual Studiem, je dobré se seznámit s konceptem **Visual Studio Solution** a **Visual Studio Project**, které jsou jeho nezbytnou součástí.

Visual Studio Project představuje projekt s názvem, který jsme si zvolili v průvodci založení nového projektu. Nejprve to budou naše konzolové aplikace, které se budeme vytvářet v rámci předmětu. Později tyto projekty zaměníme za projekty vytvářené pro desktopové aplikace. Projekt ve VS logicky seskupuje více souborů, které vytvářejí danou aplikaci nebo třeba komponentu. Projekt jazyka C# obsahuje jeden nebo více zdrojových souborů jazyka C#, konfiguračních souborů a dalších prostředků. V každém zdrojovém souboru jazyka C# existuje jedna nebo více definic typů (tříd nebo jiných definic). Ve třídách jsou metody (akce) a obsahují posloupnost příkazů. Zní to možná komplikovaně, ale u rozsáhlejších projektů je struktura, jako je tato, velmi výhodná a umožňuje dobrou organizaci pracovních souborů.

Řešení sady Visual Studio představuje kontejner (pracovní řešení), ve kterém je logicky svázáno několik projektů. Účelem vazby těchto projektů VS je vytvořit možnost, pro kód z libovolného projektu, spolupracovat s kódem z ostatních projektů VS, aby se zajistilo, že aplikace nebo web (v případě web aplikací) budou správně fungovat. V případě, že vyvíjíme rozsáhlý softwarový produkt, nebo službu, je sestavena jako řešení VS a toto řešení je diverzifikováno do samostatných projektů (VS Projects). Uvnitř každého projektu jsou složky se zdrojovými soubory. Tato hierarchická organizace je mnohem výhodnější u komerčních projektů (které čítají mnohdy desetitisíce řádků kódu).

U menších projektů VS, kombinace Řešení a VS Projektů, práci spíše komplikují, než by pomáhaly, ale s tím se jistě snadno vyrovnáme. (Nakov, 2024)



Obr. 1 – Uspořádání oken v prostředí IDE Visual Studio (printscreen) (Microsoft, 2022)

2 Běhová prostředí

Některé jazyky potřebují **běhová prostředí** (Runtime environments) ke spouštění kompilovaných programů. Například zkompilované programy **jazyka C#** jsou spouštěny běhovým prostředím **.NET Core** a zkompilované programy **Java** jsou spouštěny běhovým prostředím **Java JRE**. Jiné jazyky nepotřebují kompilaci, ale přesto vyžadují běhové prostředí. Například programy **v Pythonu** jsou spouštěny **interpretem a** běhovým prostředím Pythonu a programy **JavaScriptu** jsou spouštěny běhovým prostředím **Node.js**, nebo v prostředí **webového prohlížeče** (který poskytuje jiné běhové prostředí JS). (Nakov, 2024)

3 Programovací jazyky: nízkoúrovňové a vysokoúrovňové

Existují různé druhy **programovacích jazyků**. Prostřednictvím jazyků nejnižší úrovně můžete psát **instrukce**, které **řídí procesor**, například pomocí jazyka "**assembler**". S jazyky vyšší úrovně, jako **je C** a **C++**, můžete vytvořit operační systém, ovladače pro správu hardwaru (například ovladač grafické karty), webové prohlížeče, kompilátory, enginy pro grafiku a hry (game engine) a další systémové komponenty a programy. S jazyky ještě vyšší úrovně, jako

je **C#**, Python a JavaScript, můžete vytvářet aplikační programy, například program pro čtení e-mailů nebo chatovací program.

Nízkoúrovňové jazyky spravují hardware přímo a vyžadují velké úsilí a velký počet příkazů k provedení jednoho úkolu. **Jazyky vyšší úrovně** vyžadují méně kódu pro jednu úlohu, ale nemají přímý přístup k hardwaru. Aplikační software je vyvíjen pomocí těchto jazyků, například webové aplikace a mobilní aplikace.

Většina softwaru, který denně používáme, jako jsou přehrávače hudby, videa, GPS trackery atd., je napsána **jazyky pro programování aplikací**, které jsou na vysoké úrovni, jako je C#, Java, Python, C++, JavaScript, PHP a další.

C# je zkompilovaný jazyk, což znamená, že píšeme příkazy, které jsou kompilovány před jejich spuštěním. Přesně tyto příkazy jsou pomocí nápovědního programu (překladače) transformovány do souboru, který lze spustit (spustitelný). K napsání jazyka, jako je **C#**, potřebujeme textový editor nebo vývojové prostředí a **prostředí .NET Runtime Environment** (například .NET Core). (Nakov, 2024)

4 Běhové prostředí .NET

Prostředí .NET Runtime Environment představuje virtuální počítač, něco jako počítač v počítači, který může spustit zkompilovaný kód jazyka C#. S rizikem, že zajdeme příliš hluboko do podrobností, musíme vysvětlit, že jazyk C# je zkompilován do zprostředkujícího kódu .NET a je spouštěn z prostředí .NET, které tento zprostředkující kód dodatečně zkompiluje do strojových instrukcí (strojový kód), aby mohl být proveden mikroprocesorem. Prostředí .NET obsahuje knihovny s třídami, kompilátor **CSC**, **CLR** (Common Language Runtime – CLR) a další komponenty, které jsou nutné pro práci s jazykem C# a spouštění programů v jazyce C#.

Prostředí .NET je k dispozici jako volně šiřitelný software s otevřeným zdrojovým kódem pro každý moderní operační systém (jako jsou Windows, Linux a Mac OS X). Má dvě varianty, .NET Framework (starší) a .NET Core (novější), ale nic z toho není nezbytné, pokud jde o programování . Zaměřme se na psaní programů v jazyce C#. (Nakov, 2024)

5 Kompilace a spouštění programů v jazyce C#

Jak jsme již zmínili, program je **posloupnost příkazů**, jinak řečeno, popisuje posloupnost výpočtů, vyhodnocení, iterací a všech druhů podobných operací, jejichž cílem je získat určitý výsledek.

Program v jazyce C# je napsán v textovém formátu a text programu se nazývá **zdrojový kód**. Zkompiluje se do spustitelného souboru, nebo se spustí přímo z prostředí .NET.

Proces kompilace kódu před jeho spuštěním se používá pouze v kompilovaných jazycích jako C#, Java a C++. U skriptů a interpretovaných jazyků, jako je JavaScript, Python a PHP, je zdrojový kód spouštěn interpretem krok za krokem. (Nakov, 2024)

6 Použitá literatura

NAKOV, C. a kol., 2024. *The Book Uses C# and Visual Studio* [online]. [cit. 20.5.2024]. Dostupné na WWW: https://csharp-book.softuni.org/Content/Chapter-1-first-steps-in-programming/exercises-graphical-and-web-apps/exercises-graphical-and-web-apps.html

MICROSOFT Corp. a. s., 2022. MS Visual Studio 2022, verze 17.8.9. 14. 5. 2024 [cit. 25. 5. 2024]. Dostupné na WWW: https://visualstudio.microsoft.com/cs/

7 Studijní literatura

VIRIUS, Miroslav, 2002. *C# pro zelenáče.* Praha: Neocortex. ISBN 80-72321-76-5.

SELLS, Chris, 2005. *C# a Winforms: programování formulářů ve Windows.* Brno: Zoner Press. ISBN: 80-86815-25-0.

NAGEL, Ch., EVJEN, B., GLYNN, J. a SKINNER, M. W., 2007. *C# 2005 – Programujeme profesionálně.* Brno: Computer Press. ISBN 80-251-1181-4.

8 Otázky k procvičení

1 Co je běhové prostředí?
2 Co patří mezi vývojové prostředky?
3 Jaké úrovně je jazyk C#?
4 Co znamená zkratka CLR?
5 Co znamená zkratka CSC?

Seznam zkratek

CLR	Common Language Runtime
CSC	Kompilátor jazyka C# (CompileCS)
GUI	grafické uživatelské rozhraní
IDE	Integrated Development Environment
VS	Visual Studio

Rejstřík

Programování řídicích aplikací

Téma 04: Konzolové aplikace, datové typy, proměnné, čísla, formátování textu v C#

Studijní cíl

Seznámit studenty s tvorbou konzolových aplikací c prostředí IDE Visual Studia 2022. Dále s datovými typy, proměnnými, číselnými operacemi a formátováním textu v jazyce C#.

Doba nutná k nastudování

2 hodiny

Klíčová slova

Programový kód, zdrojový kód, kód, C#, algoritmus, vykonávání kódu, výraz, datový typ, formát textu, číselné operace, proměnná, konzolový program

1 Úvod

1.1 Systémová konzole

Nazývaná krátce jen "konzole", nebo "systémová konzole", "systémový terminál", "standardní vstup / výstup", také "počítačová konzole příkazového řádku", představuje nástroj, pomocí kterého zadáváme příkazy počítači v textovém formátu a získáváme výsledky z jejich provádění opět jako text.

Obecně platí, že systémová konzole představuje textový terminál, což znamená, že přijímá a vizualizuje pouze text bez jakýchkoli grafických prvků, jako jsou tlačítka, nabídky atd. Obvykle se zobrazuje jako černě zbarvené okno.

Ve většině operačních systémů je konzole dostupná jako samostatná aplikace, na kterou píšeme konzolové příkazy. Ve Windows se nazývá příkazový řádek a v Linuxu a Macu terminál. V konzole jsou spuštěny konzolové aplikace. Čtou text z příkazového řádku a tisknou text na konzoli. V této knize se naučíme programovat především prostřednictvím tvorby konzolových aplikací.

V dalších příkladech budeme číst data (jako jsou celá čísla, čísla s plovoucí desetinnou čárkou a řetězce) z konzoly a vypíšeme data na konzoli (text a čísla).

1.2 Čtení celých čísel z konzole

Abychom mohli z konzole přečíst celočíselné (nikoli plovoucí) číslo, musíme deklarovat proměnnou, deklarovat číselný typ a použít standardní příkaz pro čtení textového řádku ze systémové konzoly a poté převést textový řádek na celé číslo pomocí:

```
Console.ReadLine()int.Parse(text);

var num = int.Parse(Console.ReadLine());
```

Výše uvedený řádek kódu jazyka C# přečte celé číslo z prvního řádku v konzole.

1.3 Datové typy a proměnné

V programování každá proměnná ukládá určitou hodnotu určitého typu. Datové typy mohou být například: číslo, písmeno, text (řetězec), datum, barva, obrázek, seznam a další. Tady je několik příkladů datových typů:

- integer: 1, 2, 3, 45, 66, 202, …
- float: 0.53, 3.14, -10.15, -6.06, …
- character (symbol): 'a', 'b', 'c', 'd', '#', '@', 'X', …
- text (string): "Input", "Alarm", "Output?", …
- day of week: Monday, Tuesday, …, Sunday
- date and time: 24-June-1992 9:30:00, 30-Dec-2024 23:59:59

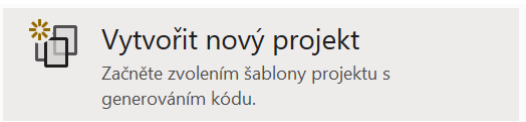
2 Konzolová aplikace

Vytvoříme nejprve konzolový program v prostředí Visual Studiu. Spustíme Visual Studio IDE, vytvoříme nový konzolový C# projekt, napíšeme několik řádků C# kódu a program zkompilujeme a spustíme.

2.1 Založení nové konzolové aplikace v prostředí Visual Studio 2022

Po spuštění Visual Studia můžeme pomocí průvodce vytvořit nový projekt a zvolit jeho typ, v tomto případě zvolíme konzolovou aplikaci (.NET Framework) (viz. obr. 1 a 2). Další možností, jak založit nový projekt je z prostředí spuštěné aplikace Visual studia. Pak bychom nový konzolový projekt vytvořili následujícím, postupným, výběrem ze sub menu aplikace:

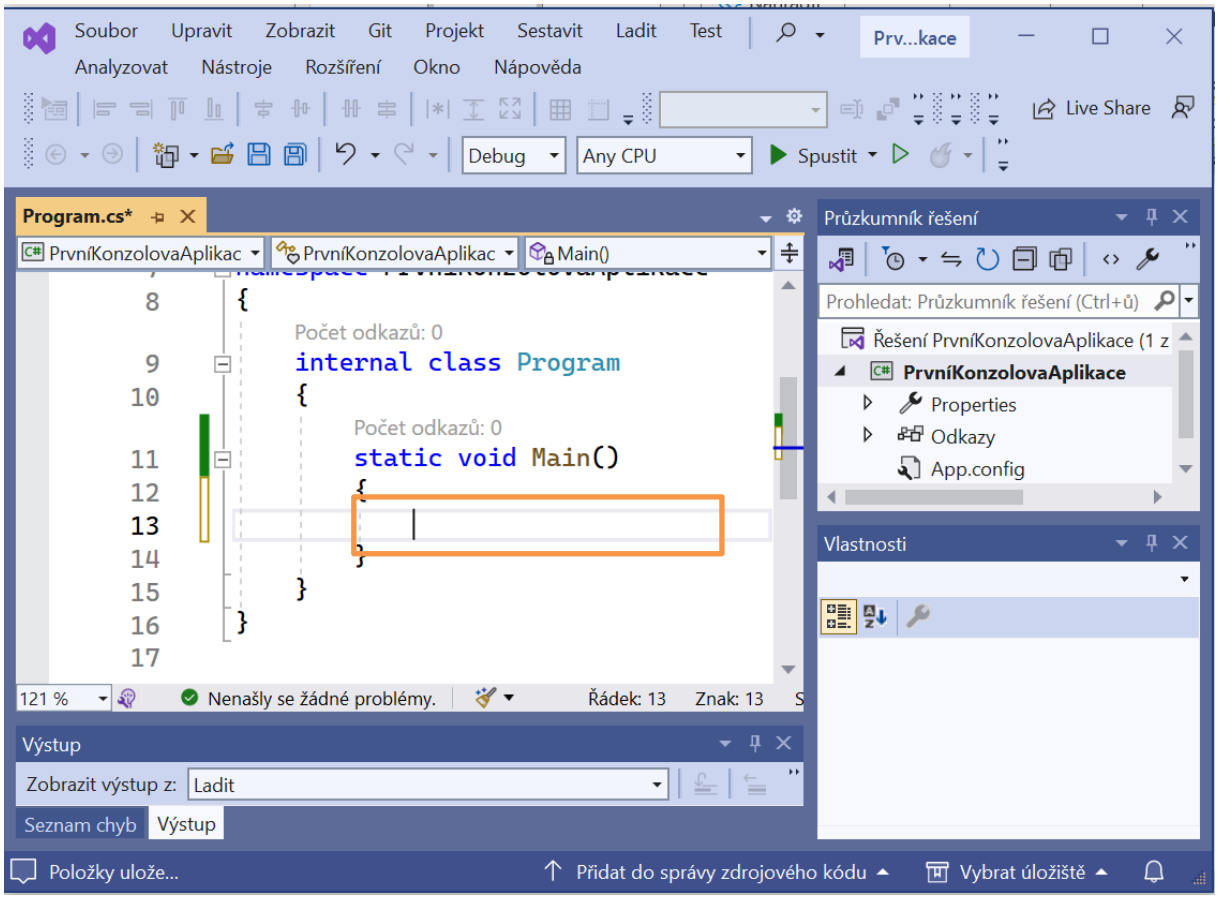
[Soubor] → [Nový] → [Project] → [Visual C#] → [Windows] → [Konzolová aplikace (.NET Framework)].



Obr. 1 – Spuštění průvodce vytvořením nového projektu (printscreen) (Microsoft, 2022)

Obr. 2 – Výběr typu projektu (printscreen) (Microsoft, 2022)

Pro nový projekt zvolíme vhodný název a jeho umístění. Visual Studio nám následně vytvoří prázdný program v C# (viz obr. 3), který musíme sami vytvořit.



Obr. 3 – Základní kostra programu (printscreen) (Microsoft, 2022)

2.2 Tvorba programového kódu

Zdrojový kód programu v jazyce C# je zapsán v sekci , mezi levou a pravou složenou závorkou { }. Jedná se o hlavní metodu (akci), která se provádí při spuštění programu v jazyce C#. Tuto metodu lze zapsat dvěma způsoby: Main(string[] args){ }Main()

static void Main(string[] args) – s parametry z příkazového řádku

static void Main() – bez parametrů z příkazové řádky.

Platí oba způsoby, protože se doporučuje ten druhý, protože je kratší a přehlednější. Ve výchozím nastavení ale Visual Studio při vytváření konzolové aplikace používá první způsob, který můžeme upravit ručně, pokud chceme, a odstranit část s parametry .string[] args

Posuneme se kurzorem, pomocí šipek klávesnice počítače, nebo kliknutím počítačové myši, za levou, složenou, závorku, pomocí klávesy [Enter] odřádkujeme a můžeme psát programový kód. Kód programu je napsán (umístěn), dovnitř mezi složené závorky, protože je to součástí formátování textu kódu, pro usnadnění při jeho následné kontrole, nebo ladění.

2.3 Konzolová aplikace

Založte nový projekt a zvolte konzolovou aplikaci s .NET Framework, kterou nazvěte „Prepona trojuhelniku“, nebo jinak, podle vašeho uvážení. Je dobré zvolit název programu tak, aby vystihoval jeho funkci. Jak už zvolený název sám napovídá, půjde o programový kód, který provádí výpočet délky přepony pravoúhlého trojúhelníku, pro zadané délky jeho odvěsen. Pro jednoduchost programu budou délky odvěsen zadány do programu ve tvaru deklarovaných proměnných s uvedením jejich datového typu.

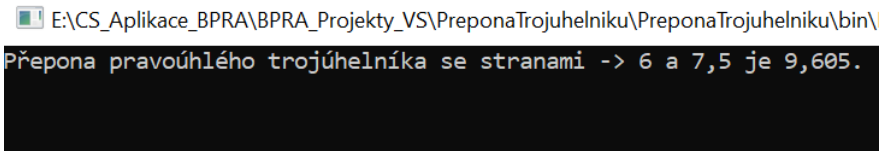
Do vytvořené, zatím prázdné aplikace, napište následující programový kód:

2.4 Program Přepona trojúhelníku

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace PreponaTrojuhelniku
{
  internal class Program
  {
    static void Main()
      {
        // Délky odvěsen trojúhelníka
        double s1 = 6.0;
        double s2 = 7.5;
        // výpočet přepony s využitím třídy Math. a jeho metody Sqrt
        double prepona = Math.Sqrt((s1 * s1) + (s2 * s2));
        Console.Write("Přepona pravoúhlého trojúhelníka se stranami -> " + s1 + " a " + s2 +
      " je ");
        Console.WriteLine("{0:#.###}.", prepona); //format to display using 3 decimal values
        Console.ReadKey();
      }
  }
}
```

Sestavený program můžeme spustit tlačítkem v ovládacím panelu VS, nebo stiskem kombinace kláves [**Ctrl + F5**]

Pokud nedojde k chybám při kompilaci programu, bude výstupní okno ve Visual Studiu po spuštění programu vypadat následovně:



Obr. 4 – Okno spuštěného programu „PreponaTrojuhelniku“ (printscreen) (Microsoft, 2022)

Série příkazových řádků, vytvořených v jazyce C#, postupně vykonaly deklaraci proměnných pro délky odvěsen trojúhelníka, jejich inicializaci (naplnění číselnou hodnotou), výpočet délky přepony trojúhelníka a formátovaný výpis řetězce na konzoli.

Jako datový typ proměnných je v aplikaci použit typ double, který umožňuje pracovat s desetinnými čísly, v rozsahu $\pm 5.0 \cdot 10^{-324}$ až $\pm 1.7 \cdot 10^{308}$.

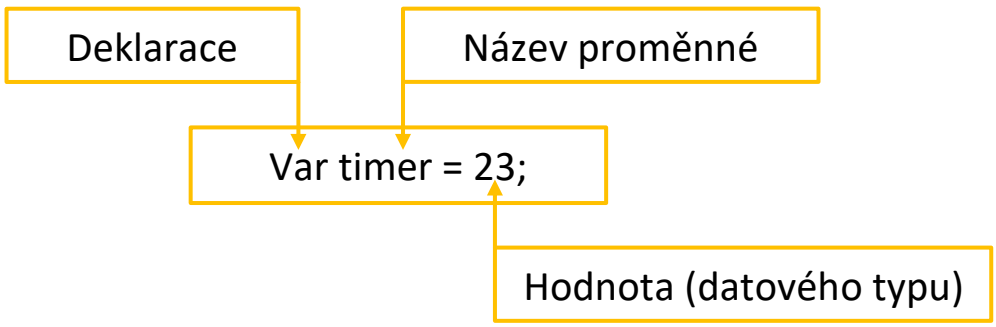
Pro oddělení desetinné části čísel používáme, ve zdrojovém kódu jazyka C#, tečku! (Nakov, 2024)

3 Deklarace a použití proměnných

Víme, že počítače zpracovávají data. Všechna data jsou uložena v paměti počítače (RAM) v proměnných. Proměnné jsou pojmenované oblasti v paměti, které uchovávají určitý datový typ, například číslo nebo text. Každá z proměnných v jazyce C# má název, typ a hodnotu. Zde je návod, jak bychom deklarovali proměnnou a zároveň jí přiřadili hodnotu:

```
var timer = 23;
```

3.1 Deklarace proměnné



Obr. 5 – Deklarace datové proměnné typu int (Nakov, 2024)

Po zpracování jsou data opět uložena v proměnných (na nějakém místě v paměti uložené pro náš program):

```
timer = timer - 1;
```

Po zpracování výše uvedeného kódu proměnná timer změní svou hodnotu, a sníží se o 1.

3.2 Čtení čísel s plovoucí desetinnou čárkou z konzoly

Chcete-li přečíst číslo s plovoucí desetinnou čárkou (zlomkové číslo, jiné než celé číslo) z konzoly, použijte následující příkaz:

```
var num = double.Parse(Console.ReadLine());
```

Výše uvedený kód jazyka C# nejprve přečte textový řádek z konzoly a pak ho převede („parsuje“) na číslo s plovoucí desetinnou čárkou. (Nakov, 2024)

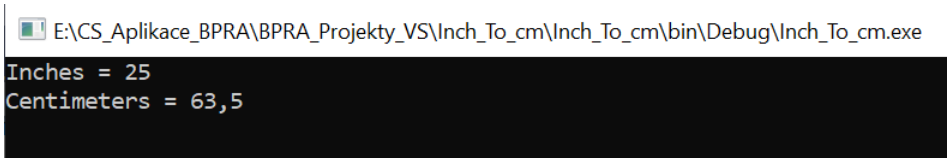
3.3 Příklad: Převod palců na centimetry

Napíšeme program, který přečte z konzole číslo s plovoucí desetinnou čárkou v palcích a převede ho na centimetry.

Založte novou konzolovou aplikaci, pojmenujte ji „Inch_To_cm“ a pro program použijte následující programový kód:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Inch_To_cm
{
  internal class Program
  {
    static void Main(string[] args)
    {
      Console.Write("Inches = ");
      var inches = double.Parse(Console.ReadLine());
      var centimeters = inches * 2.54;
      Console.Write("Centimeters = ");
      Console.WriteLine(centimeters);
      Console.Read();
    }
  }
}
```

Pokud nedojde k chybám při kompilaci programu, bude výstupní okno ve Visual Studiu po spuštění programu vypadat následovně:



Obr. 6 – Okno spuštěného programu „Inch_To_cm“ (printscreen) (Microsoft, 2022)

Poznámka:

Pokud zadáme nekorektní vstup (neodpovídají požadovanému datovému typu), např. "xyz", program spadne s chybovým hlášením (výjimkou). Jak tuto situaci vyřešit se dozvíme později.

4 Čtení textu z konzole

Pro čtení textu (stringu) z konzole musíme opět deklarovat novou proměnnou a použít standardní příkaz pro čtení textu z konzole:

```
var str = Console.ReadLine();
```

Ve výchozím nastavení metoda vrací textový výsledek – textový řádek, který se čte z konzole kódem **Console.ReadLine()**

- Po přečtení textu z konzole můžeme navíc text konvertovat na celé číslo, nebo číslo s plovoucí desetinnou čárkou pomocí kódu:

  ```
  var num1 = int.Parse(Console.ReadLine());

  var num2 = double.Parse(Console.ReadLine());
  ```

- Pokud se převod textu (datového typu string) na číslo neprovede, každé číslo bude typu string a nebude možno s ním provádět aritmetické operace.

5 Tisk a formátování textu a čísel

V C# je při tisku textu, čísel a dalších dat na konzoli je můžeme spojit pomocí šablon {1} {2} {3} {4} atd. V programování se tyto šablony nazývají zástupné symboly.

Pokud použijeme tři zástupné symboly: {0} {1} {2}

```
Console.WriteLine("{0} + {1} = {2}", 12, 16, 12+16);
```

Zástupné symboly {0}, {1} and {2} budou nahrazeny výrazy uvedenými za textem.

Výsledek z výše uvedeného kódu je:
12+16=28

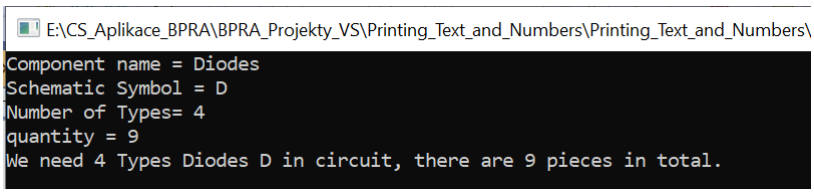
5.1 Příklad: Tisk testu a čísel

Napíšeme program, který přečte z konzole celkem čtyři proměnné. Dvě proměnné budou typu strig a dvě budou typu int. Aplikace provede zřetězení zprávy s těmito proměnnými a provede jejich zřetězení a tisk pomocí konzole. Tisk řetězce na konzoli provede s využitím zástupných symbolů. V kódu programu můžete vidět, že pořadí šablon nemusí být číselně seřazen. Šablony mohou být použity v kódu vícenásobně.

Založte novou konzolovou aplikaci, pojmenujte ji „Printing_Text_and_numbers“ a pro program použijte následující programový kód:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Printing_Text_and_Numbers
{
  internal class Program
  {
    static void Main(string[] args)
    {
      Console.Write("Component name = ");
      var ComponentName = Console.ReadLine();
      Console.Write("Schematic Symbol = ");
      var SchematicSymbol = Console.ReadLine();
      Console.Write("Number of Types= ");
      var number_of_types = int.Parse(Console.ReadLine());
      Console.Write("quantity = ");
      var quantity = int.Parse(Console.ReadLine());
      Console.WriteLine("We need {2} Types {0} {1} in circuit, there are {3} pieces in total.",
      ComponentName, SchematicSymbol, number_of_types, quantity);
      Console.Read();
    }
  }
}
```

Pokud nedojde k chybám při kompilaci programu, bude výstupní okno ve Visual Studiu po spuštění programu vypadat následovně:



Obr. 7 – Okno spuštěného programu „Print_Text_and_Numbers“ (printscreen) (Microsoft, 2022)

Všimněte si, jak by měla být každá proměnná předána v pořadí, ve kterém ji chceme vypsat. Šablona (zástupný symbol) prakticky přijímá proměnné libovolného typu.

5.2 Zřetězení textu a čísel

Kromě sčítání čísel se operátor používá také pro spojování částí textu (zřetězení dvou řetězců za sebou). V tvorbě kódu se spojení několika částí textu nazývá "zřetězení". Níže je uvedena ukázka, jak můžeme zřetězit text s číslem operátorem +

```
var ComponentName = "Tranzistor";
var ComponentType = "Tr";
var Quantity = 19;
var str = ComponentName + " " + ComponentType + " @ " + Quantity;
Console.WriteLine(str);  // str = " Tranzistor Tr 19 "
```

5.2.1 Příklady: Zřetězení textu a čísel

Zde je další příklad zřetězení textu a čísel:

```
var a = 1.5;
var b = 2.5;
var sum = "Součet je: " + a + b;
Console.WriteLine(sum);  // Součet je: 1.52.5
```

Ve výše uvedeném příkladu bychom očekávali, že čísla ve výrazech se budou sčítat. Ve skutečnosti zřetězení funguje zprava doleva a výše uvedený výsledek je naprosto správný. Pokud chceme čísla sečíst, musíme použít závorky, abychom změnili pořadí provádění operací: + a + b

```
var a = 1.5;
var b = 2.5;
var sum = "Součet je: " + (a + b);
Console.WriteLine(sum);  // Součet je: 4
```

6 Použitá literatura

NAKOV, C. a kol., 2024. *The Book Uses C# and Visual Studio* [online]. [cit. 20.5.2024]. Dostupné na WWW: https://csharp-book.softuni.org/Content/Chapter-1-first-steps-in-programming/exercises-graphical-and-web-apps/exercises-graphical-and-web-apps.html

MICROSOFT Corp. a. s., 2022. MS Visual Studio 2022, verze 17.8.9. 14. 5. 2024 [cit. 25. 5. 2024]. Dostupné na WWW: https://visualstudio.microsoft.com/cs/

7 Studijní literatura

VIRIUS, Miroslav, 2002. *C# pro zelenáče.* Praha: Neocortex. ISBN 80-72321-76-5.

SELLS, Chris, 2005. *C# a Winforms: programování formulářů ve Windows.* Brno: Zoner Press. ISBN: 80-86815-25-0.

NAGEL, Ch., EVJEN, B., GLYNN, J. a SKINNER, M. W., 2007. *C# 2005 – Programujeme profesionálně.* Brno: Computer Press. ISBN 80-251-1181-4.

8 Otázky k procvičení

1 Jak se deklaruje konstanta?
2 Jak se deklaruje proměnná?
3 Jaké datové typy má jazyk C#?
4 Jakých hodnot může nabývat datový typ integer?
5 Jakých hodnot může nabývat datový typ double a float?

Seznam zkratek

IDE Integrated Development Environment

MVS Microsoft Visual Studio

VS Visual Studio

Rejstřík

Programování řídicích aplikací

Téma 05: Konzolové aplikace, čísla, aritmetické operace, třída Math v C#

Studijní cíl

Seznámit studenty s tvorbou konzolových aplikací c prostředí IDE Visual Studia 2022. Dále s aritmetickými operacemi a použitím třídy matematických operací „Math“ v jazyce C#.

Doba nutná k nastudování

2 hodiny

Klíčová slova

Programový kód, zdrojový kód, kód, C#, algoritmus, vykonávání kódu, výraz, datový typ, formát textu, číselné operace, proměnná, konzolový program

1 Aritmetické operace

Základní aritmetické operace v programování. Čísla můžeme sčítat, odčítat, násobit a dělit pomocí operátorů + - * a /

1.1 Sčítání čísel: Operátor +

Čísla můžeme sčítat pomocí operátoru: +

```
var a = 6;

var b = 8;

var sum = a + b; // výsledkem je 14
```

1.2 Odečítání čísel: Operátor -

Odečítání čísel se provádí pomocí operátoru: -

```
var a = int.Parse(Console.ReadLine());

var b = int.Parse(Console.ReadLine());

var result = a - b;

Console.WriteLine(result);
```

1.3 Násobení čísel: operátor *

Pro násobení čísel používáme operátor: *

```
var a = 6;
var b = 7;
var result = a * b; // 42
```

1.4 Dělení čísel: operátor /

Dělení čísel se provádí pomocí operátoru. Funguje jinak s celými čísly a čísly s plovoucí desetinnou čárkou.

- Když dělíme dvě celá čísla, použije se celočíselné dělení a získaný výstup je bez zlomkové části. Příklad: 16 / 3 = 5.
- Když vydělíme dvě čísla a alespoň jedno z nich je číslo s plovoucí desetinnou čárkou, použije se plovoucí dělení a získaným výsledkem je číslo s plovoucí desetinnou čárkou, stejně jako v matematice. Příklad 15 / 5,0 = 3,20. Pokud to nelze provést s přesnou přesností, výsledek se zaokrouhluje, například 16,0 / 3 = 5,33333333333333.
- Celočíselné dělení 0 způsobí výjimku za běhu.
- Čísla s plovoucí desetinnou čárkou dělená 0 nezpůsobují výjimku a výsledek je +/- nekonečno nebo speciální hodnota NaN. Příklad 5 / 0,0 = ∞.

(Nakov, 2024)

1.4.1 Příklady s operátorem dělení

```
var a = 25;
    var i = a / 4;      // Aplikujeme celočíselné dělení
                        // výsledek této operace bude 6 – zlomková část bude oříznuta,
                        // protože dělíme celá čísla
    var f = a / 4.0; // 6.25 – plovoucí dělení. Deklarovali jsme číslo 4 jako
                        // float přidáním oddělovače desetinných míst následovaného nulou
    var error = a / 0;        // Chyba: Celé číslo děleno nulou
```

1.5 Dělení celých čísel

Podívejme se na několik příkladů celočíselného dělení (nezapomeňte, že když dělíme celá čísla v C#, výsledkem je celé číslo):

```
var a = 25;
Console.WriteLine(a / 4);  // Celočíselný výsledek: 6
Console.WriteLine(a / 0);  // Error: Dělení 0
```

1.6 Dělení čísel s plovoucí desetinnou čárkou

Podívejme se na několik příkladů dělení čísel s plovoucí řádovou čárkou. Když dělíme čísla s plovoucí desetinnou čárkou, výsledkem je vždy číslo s plovoucí desetinnou čárkou a dělení nikdy nezklame a funguje správně se speciálními hodnotami + ∞ a - ∞:

```
var a = 23;
Console.WriteLine(a / 2.0);     // Výsledek Float: 11.5
Console.WriteLine(a / 0.0);     // Výsledek: Nekonečno
Console.WriteLine(-a / 0.0);    // Výsledek: -Nekonečno
Console.WriteLine(0.0 / 0.0); // Výsledek: NaN (Not a Number), např. výsledek
                                // z operace kde není platná číselná hodnota
```

Při tisku hodnot ∞ a -∞ může být výstup konzoli , protože konzola ve Windows nefunguje správně s Unicode a poruší většinu nestandardních symbolů, písmen a speciálních znaků. Výše uvedený příklad by s největší pravděpodobností poskytl následující výsledek: ?

(Nakov, 2024)

```
11.5
?
-?
NaN
```

1.7 Zřetězení textu a čísel

Kromě sčítání čísel se operátor používá také pro spojování částí textu (zřetězení dvou řetězců za sebou). V tvorbě kódu se spojení několika částí textu nazývá "zřetězení". Níže je uvedena ukázka, jak můžeme zřetězit text s číslem operátorem: +

```
var ComponentName = "Tranzistor";
var ComponentType = "Tr";
var Quantity = 19;
var str = ComponentName + " " + ComponentType + " @ " + Quantity;
Console.WriteLine(str);  // str = " Tranzistor Tr 19 "
```

1.7.1 Příklady: Zřetězení textu a čísel

Zde je další příklad zřetězení textu a čísel:

```
var a = 1.5;
var b = 2.5;
var sum = "Součet je: " + a + b;
Console.WriteLine(sum);  // Součet je: 1.52.5
```

Ve výše uvedeném příkladu bychom očekávali, že čísla ve výrazech se budou sčítat. Ve skutečnosti zřetězení funguje zprava doleva a výše uvedený výsledek je naprosto správný. Pokud chceme čísla sečíst, musíme použít závorky, abychom změnili pořadí provádění operací: + a + b

```
var a = 1.5;
var b = 2.5;
var sum = "Součet je: " + (a + b);
Console.WriteLine(sum);  // Součet je: 4
```

2 Číselné výrazy

V programování můžeme počítat číselné výrazy, například:

```
var expr = (4 + 9) * (7 – 2);
```

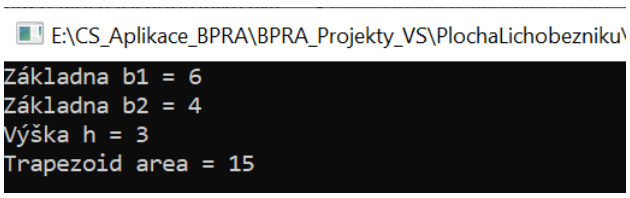
Platí standardní pravidlo pro priority aritmetických operací: násobení a dělení se provádí vždy před sčítáním a odčítáním. V případě výrazu v závorce se nejprve vypočítá, ale to vše už známe ze školní matematiky.

2.1 Příklad: Výpočet lichoběžníkové plochy

Napišme program, který zadá délky dvou základen lichoběžníku a jeho výšku (jedno číslo s plovoucí desetinnou čárkou na řádek) a vypočítá plochu lichoběžníku podle standardního matematického vzorce:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace PlochaLichobezniku
{
   internal class Program
   {
      static void Main(string[] args)
      {
         Console.Write("Základna b1 = ");
         var b1 = double.Parse(Console.ReadLine());
         Console.Write("Základna b2 = ");
         var b2 = double.Parse(Console.ReadLine());
         Console.Write("Výška h = ");
         var h = double.Parse(Console.ReadLine());
         var area = (b1 + b2) * h / 2.0;
         Console.WriteLine("Trapezoid area = " + area);
         Console.Read();
      }
   }
}
```

Pokud spustíme program a zadáme hodnoty pro strany: 6, 4 a 3, dostaneme následující výsledek:



Obr. 1 – Okno spuštěného programu „PlochaLichobezniku“ (printscreen) (Microsoft, 2022)

2.2 Příklad: Obvod a obsah kružnice

Vytvořme program, který vypočítá plochu kružnice a obvod načtením jejího poloměru r.

```
using System;
using System.Collections.Generic;
```

```
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ObvodObsahKruznice
{
   internal class Program
   {
      static void Main(string[] args)
      {
         Console.Write("Poloměr kružnice r = ");
         var r = double.Parse(Console.ReadLine());
         Console.WriteLine("Plocha kružnice = " + Math.PI * r * r);
         // Math.PI – build in constant for π in C#
         Console.WriteLine("Obvod kružnice = " + 2 * Math.PI * r);
         Console.Read();
      }
   }
}
```

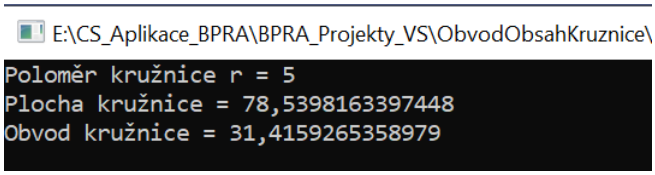
Vzorce pro výpočet:

Plocha = π * r * r

Obvod = 2 * π * r

π ≈ 3.14159265358979323846...

Pokud spustíme program a zadáme hodnotu r = 5, dostaneme následující výsledek:



Obr. 2 – Okno spuštěného programu „ObvodObsahKruznice“ (printscreen) (Microsoft, 2022)

3 Samostatné cvičení

Níže uvedená zadání zpracujte samostatně a řešení zpracujte do technické zprávy. Postupujte podle požadavků na její zpracování.

3.1 Úkol: Převodník ze stupňů °C na stupně °F a °K

Vytvořte program, který čte stupně na stupnici Celsia (°C) a převádí je na stupně na stupnici Fahrenheita (°F) a Kelvina (K). Výsledek zaokrouhlete na 2 číslice za desetinnou čárkou. Zde je několik hodnot:

Tab. 1 – Ukázkový vstup a výstup aplikace

Vstup [°C]	Výstup [°F]	Výstup [K]
22	71	295.15
0	32	273.15
-7.5	18.5	265.65
36.6	97.88	309.75

Vzorce pro přepočet jednotek:

°C = (°F – 32) * 5/9

°F = (°C * 9/5) + 32

K = °C + 273.15

3.2 Úkol: Převodník z radiánů na stupně

Vytvořte program, který přečte z příkazového řádku hodnotu úhlu v radiánech a převede je na hodnotu ve stupních. Hodnota π v programech jazyka C# je k dispozici prostřednictvím třídy Math. Pomocí její metody Math.Round zaokrouhlete výsledek na nejbližší celé číslo.

Tab. 2 – Ukázkový vstup a výstup aplikace

Vstup [rad]	Výstup [°]
6.2832	360
3.1416	180
0.7854	45
0.5236	30

Bližší informace o třídě Math jsou dostupné například na:

https://learn.microsoft.com/cs-cz/dotnet/api/system.math.round?view=net-8.0#system-math-round(system-decimal-system-int32)

Příklad kódu metody přepočtu radiánu na stupně:

```
using System;
public static double ConvertRadiansToDegrees(double radians)
{
        double degrees = (180 / Math.PI) * radians;
        return (degrees);
}
```

4 Použitá literatura

NAKOV, C. a kol., 2024. *The Book Uses C# and Visual Studio* [online]. [cit. 20.5.2024]. Dostupné na WWW: https://csharp-book.softuni.org/Content/Chapter-1-first-steps-in-programming/exercises-graphical-and-web-apps/exercises-graphical-and-web-apps.html

MICROSOFT Corp. a. s., 2022. MS Visual Studio 2022, verze 17.8.9. 14. 5. 2024 [cit. 25. 5. 2024]. Dostupné na WWW: https://visualstudio.microsoft.com/cs/

5 Studijní literatura

VIRIUS, Miroslav, 2002. *C# pro zelenáče.* Praha: Neocortex. ISBN 80-72321-76-5.

SELLS, Chris, 2005. *C# a Winforms: programování formulářů ve Windows.* Brno: Zoner Press. ISBN: 80-86815-25-0.

NAGEL, Ch., EVJEN, B., GLYNN, J. a SKINNER, M. W., 2007. *C# 2005 – Programujeme profesionálně.* Brno: Computer Press. ISBN 80-251-1181-4.

6 Otázky k procvičení

1 Co patří mezi aritmetické operace?
2 Jaký operátor se používá pro celočíselné dělení?
3 Jaký operátor se používá pro operaci MOD?
4 % se používá pro jakou aritmetickou operaci?
5 Pro jaký typ proměnné se používá klíčové slovo double?
6 Pro jaký typ proměnné se používá klíčové slovo integer?

Seznam zkratek

IDE	Integrated Development Environment
MVS	Microsoft Visual Studio
VS	Visual Studio

Rejstřík

Programování řídicích aplikací

Téma 06: Konzolové aplikace, větveni programu, cykly v C#

Studijní cíl,

Seznámit studenty s tvorbou konzolových aplikací v prostředí IDE Visual Studia 2022. Dále s programovou konstrukcí větvení programu (funkce IF), programovými cykly a deklaracemi datových polí.

Doba nutná k nastudování

2 hodiny

Klíčová slova

Programový kód, zdrojový kód, kód, C#, vykonávání kódu, výraz, datový typ, větvení programu, cyklus

1 Úvod

Nyní se budeme zabývat podmíněnými příkazy v jazyce C#, jejichž prostřednictvím může mít program různou činnost, v závislosti na vyhodnocení podmínky (podmínek). Na vhodných příkladech si ověříme a otestujeme syntaxi podmíněných operátorů pro řízení běhu programu a budeme testovat, v jakém rozsahu se proměnná nachází. Nakonec se seznámíme s možnostmi ladění programu, abychom mohli sledovat cestu zpracování programu, kterým pracuje náš program v průběhu implementace konstrukce if , if-else , Switch - Case.

V průběhu zpracování programového kódu můžeme průběžně testovat příslušné podmínky a v závislosti na jejich vyhodnocení můžeme vykonávat různé bloky kódu.

2 Větvení programu: if-else_if-else

Kód programu přečte z konzole číslo a v průběhu zpracování kódu několikrát porovnává jeho velikost a v závislosti na aktuálním vyhodnocení testovacích podmínek vytiskne na konzoli příslušnou zprávu. Pokud bude číslo ve stanoveném rozmezí vypočítá se plocha čtverce a vytiskne se příslušná zpráva.

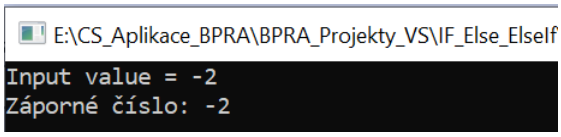
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace IF_Else_ElseIf
{
   internal class Program
   {
      static void Main(string[] args)
      {
        int max = 10;
            Console.Write(" Input value = ");
        var size = decimal.Parse(Console.ReadLine());
        if (size < 0)
            Console.WriteLine($"Číslo je záporné : {size}");
        else if (size > max)
            Console.WriteLine($"%Číslo je větší než max: {size}");
        else
         {
            Console.WriteLine($"Číslo je v pořádku: {size}");
            Console.WriteLine($"Plocha je: {size * size}");
         }
         Console.Read();
      }
   }
}
```

Funkce programového kódu:

Proměnná se pro vstupní číslo jmenuje vstup a její hodnota je 10.

Pokud bude číslo záporné, to odpovídá podmínce vstup < 0, vytiskne se zpráva:

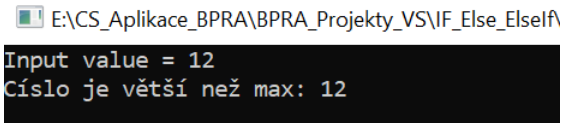
„Záporné číslo: vstup“



Obr. 1 – Okno spuštěného programu „IF_Else_ElseIF“ (printscreen) (Microsoft, 2022)

Pokud bude číslo větší, než stanovený rozsah, vstup > max, vytiskne se zpráva:

„Číslo je větší než max:“



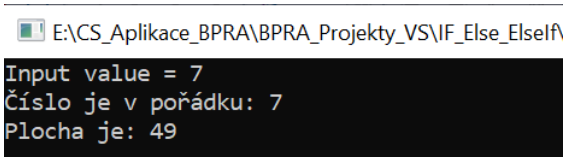
Obr. 2 – Okno spuštěného programu „IF_Else_ElseIF“ (printscreen) (Microsoft, 2022)

Pokud bude číslo ve stanovené rozsahu, to znamená, 0 < vstup a zároveň vstup < max, vytiskne se zpráva:

„Input value = “

„Číslo je v pořádku:“

Plocha je:



Obr. 3 – Okno spuštěného programu „IF_Else_ElseIF“ (printscreen) (Microsoft, 2022)

3 Podmíněný příkaz Switch-Case

Pracuje jako přepínač, jho funkce je podobná, jako posloupnost bloků if-else. Kdykoli funkce vytvářeného programu závisí na hodnotě jedné proměnné, lze místo tvorby po sobě jdoucích podmínek s bloky if-else použít příkaz podmíněného přepínače - switche. Používá se tedy pro výběr mezi seznamem možností. Příkaz porovná danou hodnotu proměnné s předem definovanou sadou konstant a v závislosti na výsledku tohoto porovnání provede příslušnou akci. Proměnnou, kterou chceme porovnávat, umístíme do závorek za příkaz Switch. Tuto proměnnou můžeme nazvat "selektor". U proměnné selektoru a porovnávaných prvků musí být použit shodný datový typ (například čísla, řetězce). Program začne postupně porovnávat vstupní proměnnou selektoru s každou hodnotou, která se nachází za klíčovými slovy case. Při nalezené shodě začíná vykonávání kódu z tohoto místa, kde ke shodě došlo, a pokračuje, dokud nedosáhne operátoru break. V jazyce C# je použití klíčového příkazu break povinná v každé větvi, která obsahuje logiku větvení programu. Pokud není při porovnávání nalezena žádná shoda, spustí se kód, který následuje za klíčovým slovem default, pokud tedy takováto část kódu, existuje.

3.1 Program Switch – Case 1

```
switch (selector)
{
        case value1:
```

```
        construction;
        break;
        case value2:
        construction;
        break;
        case value3:
        construction;
        break;
        …
        default:
        construction;
        break;
}
```

V jazyce C# máme možnost použít více popisků case pro vykonání jednoho, stejného, oddílu kódu. Tímto způsobem, když náš program najde shodu, provede další kód, protože za příslušným popiskem případu není žádný kód pro vykonání a operátor break.

3.2 Program Switch – Case 2

```
switch (selector)
{
        case value1:
        case value2:
        case value3:
        construction;
        break;
        case value4:
        case value5:
        construction;
        break;
        …
        default:
        construction;
        break;
}
```

4 Smyčky (opakování)

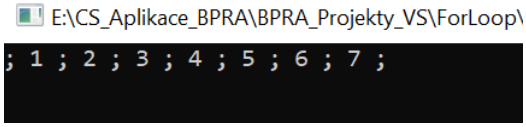
Nyní se seznámíme s tím, jak provést opakování bloků příkazů, známé také pod pojmenováním "smyčky". Vytvoříme několik jednoduchých smyček pomocí operátoru v jeho nejjednodušší podobě (for i = 1 … n). Nakonec vyřešíme řadu praktických problémů, které vyžadují opakování řady akcí pomocí smyček.

Při tvorbě programového kódu můžeme spustit blok kódu vícekrát pomocí jednoduché smyčky, for - loop, jako je tato:

4.1 Program Cyklus For 1

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ForLoop
{
   internal class Program
   {
      static void Main(string[] args)
      {
         Console.Write("; ");
         for (int i = 1; i <= 7; i++)
         {
            Console.Write(i + " ; ");
         }
         Console.Read();
      }
   }
}
```

Po spuštění kódu obdržíme takovýto výpis v konzoli:



Obr. 4 – Okno spuštěného programu „For_Loop“ (printscreen) (Microsoft, 2022)

Můžeme zadat více čísel z konzoli a zpracovat je pomocí smyček, jako je tato:

Zadejte počet sčítanců - n , celých čísel (typ integer).

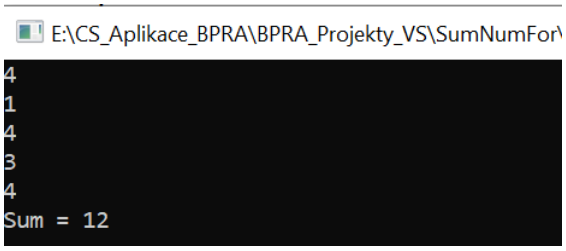
Ve smyčce for zadáváme n krát vždy jedno číslo.

Výše uvedené zadání lze vyřešit například takto:

4.2 Program Cyklus For 2

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace SumNumFor
{
   internal class Program
   {
      static void Main(string[] args)
      {
         int n = int.Parse(Console.ReadLine());
         long sum = 0;
         for (int i = 0; i < n; i++)
         {
            int num = int.Parse(Console.ReadLine());
            sum += num;
         }
         Console.WriteLine("Sum = {0}", sum);
         Console.Read();
      }
   }
}
```

Výstup z výše uvedeného příkladu může vypadat takto (když zadáme 4 čísla: 1, 4, 3 a 4):



Obr. 5 – Okno spuštěného programu „SumNumFor“ (printscreen) (Microsoft, 2022)

4.3 While Loop

Další typem smyček, se kterými se seznámíme, se nazývá smyčky while. Specifické na nich je, že opakují blok příkazů, dokud je podmínka pravdivá. Jako struktura se liší od for smyček a dokonce mají jednoduchou syntaxi.

Při programování se smyčka while používá, když chceme opakovat provádění určité logiky, zatímco je podmínka v platnosti. Pod pojmem "podmínka" rozumíme každý výraz, který vrací hodnotu pravda nebo nepravda. Pokud je podmínka nesprávná, smyčka while se přeruší, program pokračuje v provádění zbývajícího kódu po cyklu.

4.4 Příklad: Smyčka While, výpočet číselné řady

Napište program, který vytiskne všechna čísla ≤ n řady: 1, 3, 7, 15, 31, ..., za předpokladu, že každé další číslo = předchozí číslo * 2 + 1.

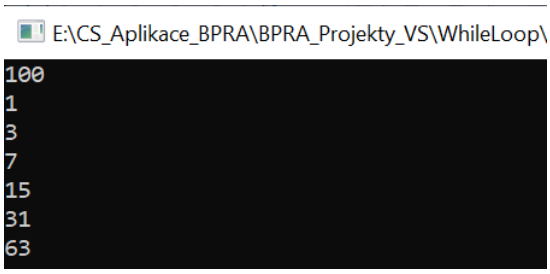
Zde je možný postup, jak můžeme zadání vyřešit:

- Pro aktuální číslo vytvoříme proměnnou **num**, které přiřadíme počáteční hodnotu 1.
- Pro podmínku cyklu vložíme **aktuální číslo** <= n.
- V těle smyčky: vytiskneme hodnotu **aktuálního čísla** a aktuální číslo zvýšíme pomocí vzorce z popisu problému.

Ukázková implementace tohoto řešení:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace WhileLoop
{
  internal class Program
  {
    static void Main(string[] args)
    {
      int n = int.Parse(Console.ReadLine());
      int num = 1;
      while(num <= n)
      {
        Console.WriteLine(num);
        num = 2 * num + 1;
      }
       Console.ReadKey();
    }
  }
}
```

Výstup z výše uvedeného příkladu může vypadat takto (když zadáme číslo 100 obdržíme čísla: 1, 3, 7, 15, 31 a 63):



Obr. 6 – Okno spuštěného programu „SumNumFor“ (printscreen) (Microsoft, 2022)

4.5 Do - While Loop

Dalším typem smyček, se kterými se seznámíme, jsou smyčky do-while. Strukturou se tento typ smyčky podobá smyčce while, ale je mezi nimi významný rozdíl. Jde o to, že smyčka do - while provede kód ve svém těle alespoň jednou. Proč se to děje? V konstrukci do-while cyklu je podmínka vždy kontrolována až za tělem, což zajišťuje, že první iterace smyčky provede kód a kontrola konce smyčky se použije na každou následující iteraci do-while.

4.6 Příklad: Smyčka Do - While, výpočet faktoriálu

Pro přirozené **číslo n** vypočítejte n! = 1 * 2 * 3 * ... * n. Pokud například n = 5, výsledek bude: **7!** = 1 * 2 * 3 * 4 * 5 * 6 * 7 = 5040.

Zde je návod, jak konkrétně vypočítat faktoriál:

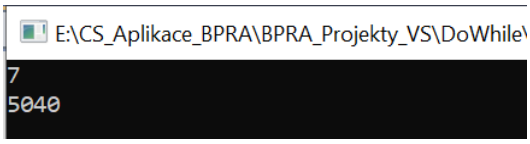
- Vytvoříme proměnnou n, které přiřadíme celočíselnou hodnotu převzatou ze vstupu konzoly.
- Vytvoříme další proměnnou s názvem `fakt`, jehož počáteční hodnota je 1. Použijeme jej pro výpočet a uložení faktoriálu.
- Pro podmínku cyklu použijeme `n > 1`, protože pokaždé, když provedeme výpočty v těle smyčky, snížíme hodnotu **n** o 1.
- V těle smyčky:
  - Proměnné fakt přiřadíme novou hodnotu, která je výsledkem vynásobení aktuální hodnoty fakt aktuální hodnotou n.
  - Snížíme hodnotu n o -1.
- Mimo tělo smyčky vytiskneme konečnou hodnotu faktoriálu.

Ukázkový kód, který implementuje výše popsané kroky:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace DoWhile
{
  internal class Program
  {
    static void Main(string[] args)
    {
      int n = int.Parse(Console.ReadLine());
      int fact = 1;
      do
      {
```

```
            fact = fact * n;
            n--;
        }
        while (n > 1) ;
        Console.WriteLine(fact);
        Console.Read();
    }

  }
}
```

Výstup z výše uvedeného příkladu může vypadat takto:



Obr. 7 – Okno spuštěného programu „SumNumFor“ (printscreen) (Microsoft, 2022)

5 Klávesové zkratky pro psaní znaků stiskem kombinací kláves:

< >	[Pravý ALT + < >]	$	[Pravý ALT + ů]	^	[Pravý ALT + 3]
|	[Pravý ALT + W]	÷	[Pravý ALT + ú]	\	[Pravý ALT + Q]
&	[Pravý ALT + C]	×	[Pravý ALT + ×]	*˙	[Pravý ALT + -]
[	[Pravý ALT + F]	€	[Pravý ALT + E]	{	[Pravý ALT + B]
]	[Pravý ALT + G]	~	[Pravý ALT + 1]	}	[Pravý ALT + N]
@	[Pravý ALT + V]	°	[SHIFT + ;]		
#	[Pravý ALT + V]				

Tip pro VS:

Pro formátování textu zdrojových kódů ve Visual Studiu můžeme použít funkci **-auto format** můžeme použít kombinaci kláves:

- pro celý soubor [CTRL + K + D]
- Pro výběr [CTRL + K + D]

Nebo z hlavní nabídky Upravit -> Upřesnit -> [Formátovat dokument], nebo [Formátovat výběr]

6 Použitá literatura

MICROSOFT Corp. a. s., 2022. MS Visual Studio 2022, verze 17.8.9. 14. 5. 2024 [cit. 25. 5. 2024]. Dostupné na WWW: https://visualstudio.microsoft.com/cs/

7 Studijní literatura

VIRIUS, Miroslav, 2002. *C# pro zelenáče.* Praha: Neocortex. ISBN 80-72321-76-5.

SELLS, Chris, 2005. *C# a Winforms: programování formulářů ve Windows.* Brno: Zoner Press. ISBN: 80-86815-25-0.

NAGEL, Ch., EVJEN, B., GLYNN, J. a SKINNER, M. W., 2007. *C# 2005 – Programujeme profesionálně.* Brno: Computer Press. ISBN 80-251-1181-4.

8 Otázky k procvičení

1 Vysvětlete funkci větvení programu IF-ElseIF-Else?
2 Vysvětlete funkci větvení programu Switch - Case?
3 Vysvětlete funkci smyčky For?
4 Vysvětlete funkci smyčky While?
5 Vysvětlete funkci smyčky Do-While?

Seznam zkratek

MVS Microsoft Visual Studio

VS Visual Studio

Rejstřík

Programování řídicích aplikací

Téma 07: Windows Forms aplikace v jazyce C# v prostředí Visual Studia

Studijní cíl

Seznámit studenty s tvorbou Windows Forms aplikace, založením projektu formulářové aplikace ve Visual studiu, použitím komponent, nastavením jejich vlastností a obsluha jejich událostí. Součástí zpracování tematického okruhu je tvorba několika kompletních, demonstračních, GUI aplikaci.

Doba nutná k nastudování

2 hodiny

Klíčová slova

Programovací jazyk, počítačový program, programový kód, zdrojový kód, kód, C#, algoritmus, vykonávání kódu, kompilace, událost, obsluha události, komponenty, ovládací prvky, Button, Label, Textové pole

1 Konzolové, webové a grafické aplikace

U konzolových aplikací, jak můžete sami zjistit, jsou všechny operace pro čtení vstupu a tisk výstupu prováděny prostřednictvím konzole. Vstupní data se zadávají do konzole, která pak aplikace také čte, a výstupní data se vypisují na konzoli po nebo za běhu programu.

Zatímco konzolová aplikace používá textovou konzoli, webové aplikace používají webové uživatelské rozhraní. K jejich provedení jsou potřeba dvě věci – webový server a webový prohlížeč, protože prohlížeč hraje hlavní roli ve vizualizaci dat a interakci s uživatelem. Webové aplikace jsou pro uživatele mnohem příjemnější, vizuálně vypadají lépe, lze použít myš a dotykový displej (pro tablety a chytré telefony).

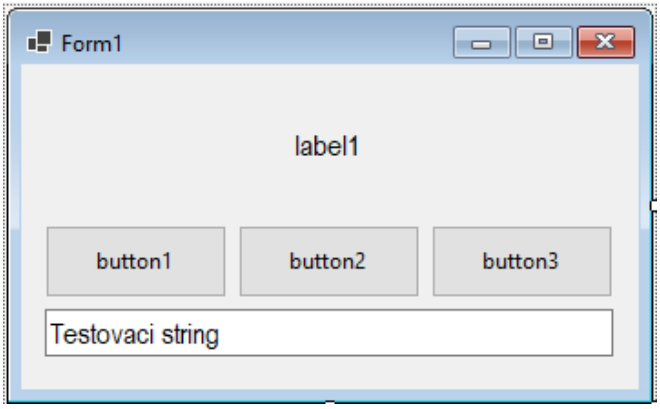
Grafické (GUI) aplikace mají vizuální uživatelské rozhraní přímo do vašeho počítače nebo mobilního zařízení bez webového prohlížeče. Grafické aplikace (známé také jako desktopové aplikace) obsahují jedno nebo více grafických oken, ve kterých jsou umístěny určité ovladače (textová pole, tlačítka, obrázky, tabulky a další), které slouží k intuitivnějšímu dialogu s uživatelem. Podobné jsou mobilní aplikace ve vašem telefonu nebo tabletu. Základním prvkem je formulář, spolu s dalšími ovládacími prvky (komponenty). Jsou to například, textová pole, tlačítka, rozbalovací seznamy, obrázkové kontejnery, a další ovládací prvky, které ovládáme programovacím kódem.

2 Grafické aplikace

Nyní vytvoříme grafickou, formulářovou .NET, aplikaci, do které umístíme několik ovládacích prvků, pro které vytvoříme obslužný programový kód. Vyzkoušíme si tak umístění vybraných ovládacích prvků, nastavení jejich vlastností a obsluhu jejich událostí. Získáme tak základní dovednosti z tvorby formulářových, GUI, aplikací.

3 Grafická aplikace, testování programové obsluhy vlastností a událostí objektů

Cílem je vytvořit **grafickou aplikaci** (GUI aplikaci), která bude mít jeden formulář, tři tlačítka, textové pole a prvek Label. Obslužné události tlačítek budou pracovat s vytvořeným obslužným kódem, který každému tlačítku přiřadí jiný typ práce s daty. První tlačítko provede akci počítání svých stisků. Druhé tlačítko provede výpočet vybrané matematické operace. Třetí tlačítko bude po svém stisku kopírovat text z textového pole do ovládacího prvku Label.

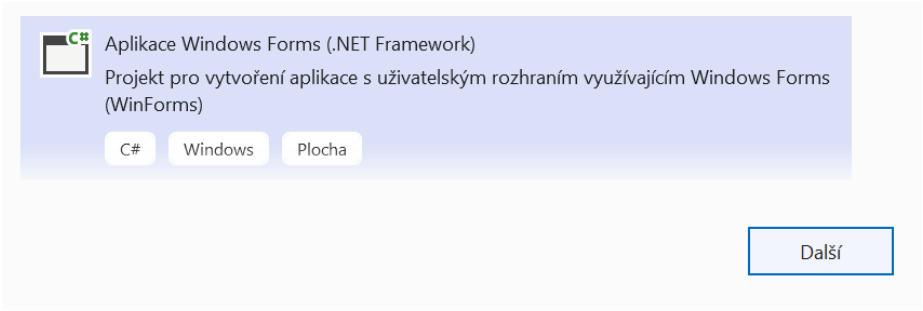


Obr. 1 – Vzhled aplikace (vytvořeno v Microsoft Visual Studio 2022)

V tomto cvičení si procvičíme práci s komponentami použitými k tvorbě konkrétní aplikace s grafickým uživatelským rozhraním.

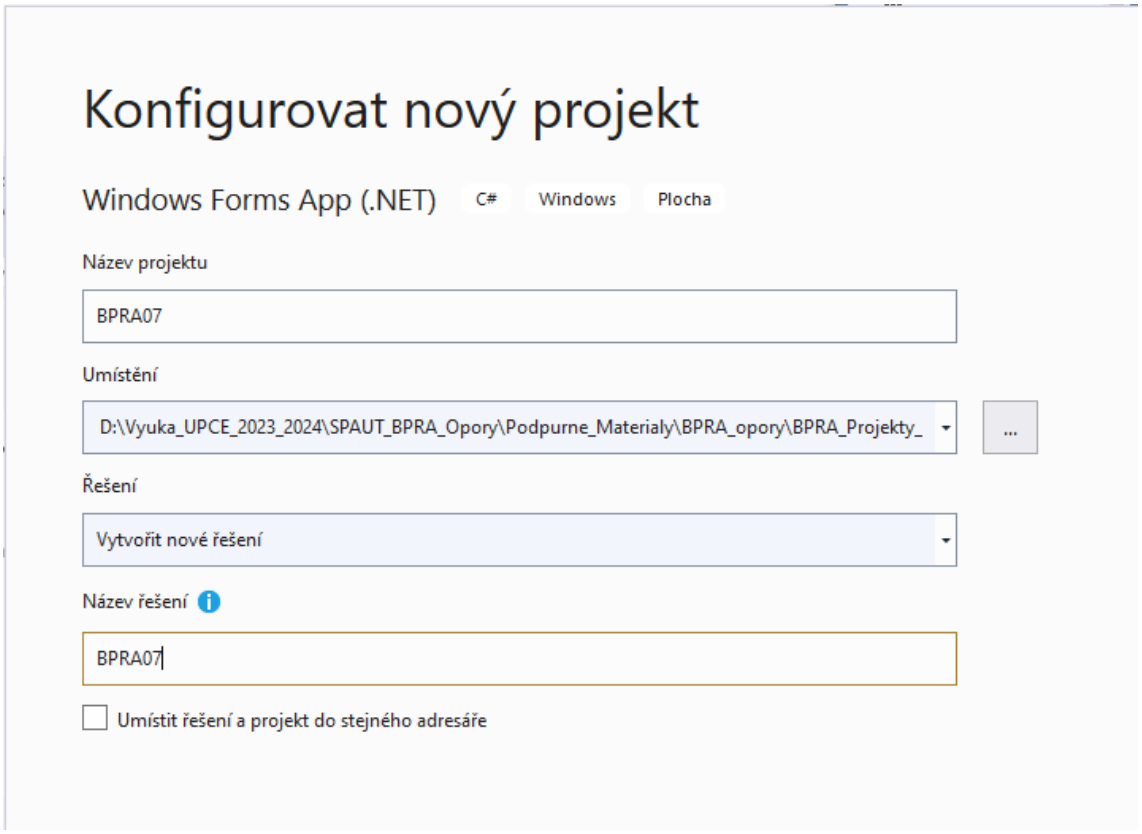
3.1 Vytvoření nového projektu v jazyce C#

Vytvoříme projekt Windows Forms aplikaci .NET v C# s názvem "BPRA07". Po spuštění průvodce ve Visual Studiu zvolíme Windows Forms aplikaci pro .NET



Obr. 2 – Založení nového projektu v Windows Forms (printscreen) (Microsoft, 2022)

Do dialogového okna vložíme název projektu a jeho umístění na disku.



Obr. 3 – Založení nového projektu v Windows Forms (printscreen) (Microsoft, 2022)

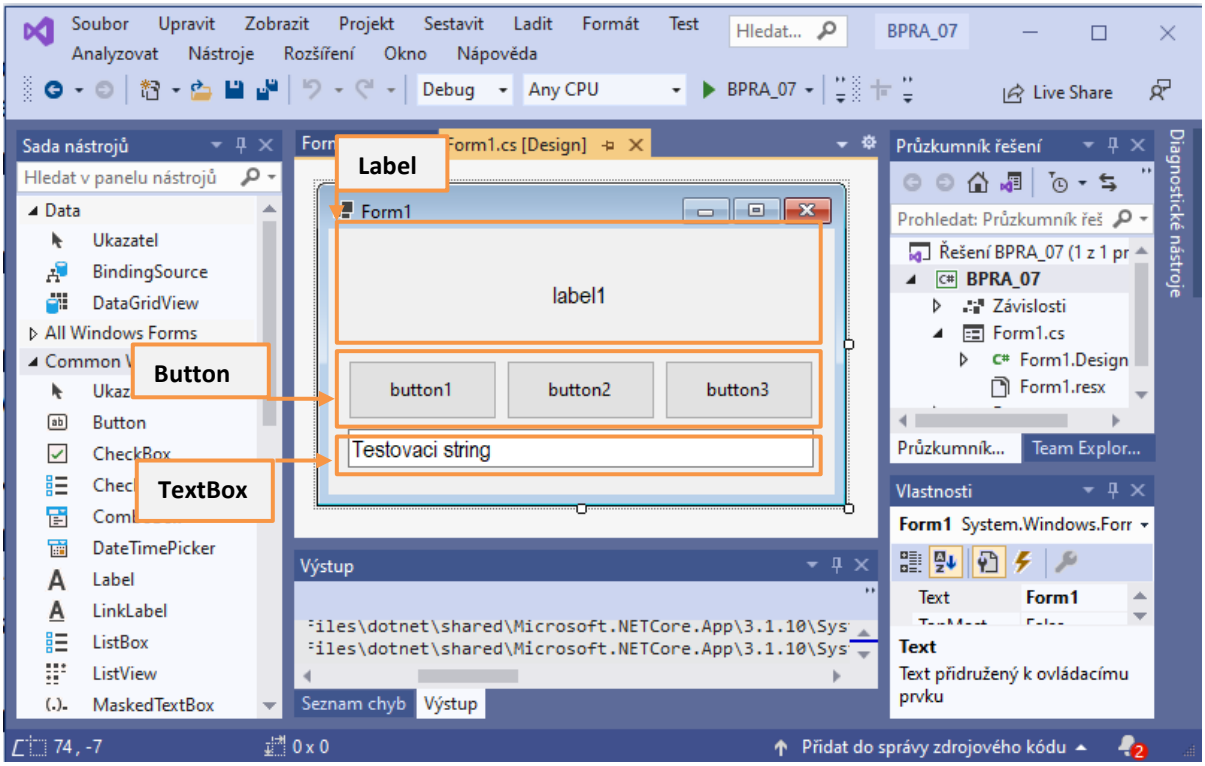
3.2 Přidání ovládacích prvků uživatelského rozhraní

Použijeme následující komponenty UI (User Interface) a rozmístíme je na formuláři:

- **Label** s názvem label1 – pro zobrazení výsledků jednotlivých obslužných kódů
- **TextBox** s názvem textBox1 – pro zadávání textové zprávy pro label1.text
- **Button** s názvem button1 - pro počítání stisků tlačítka button1
- **Button** s názvem button2 – pro zobrazení zprávy v label1
- **Button** s názvem button3 – pro výpočet aritmetické operace

Po vložení ovládacích prvků je rozmístěte na formuláři tak, jak je vyobrazeno na obr. č. 4,

Vzhled grafického editoru uživatelského rozhraní může vypadat podobně jako tento:



Obr. 4 – Rozložení komponent v aplikaci (printscreen) (Microsoft, 2022)

3.3 Konfigurace ovládacích prvků uživatelského rozhraní

Nyní je potřeba nastavit vlastnosti vložených ovládacích prvků. Pro ovládací prvky uživatelského rozhraní použijeme následující nastavení:

- Pro **hlavní formulář** (), který obsahuje všechny ovládací prvky: Form
    - (name) = Form1
    - Size = 500 ; 350
    - Text = "Form1"
    - Font.Size = 8
    - MaximizeBox = False
    - MinimizeBox = False
    - StartPosition = CenterScreen
    - FormBorderStyle = FixedSingle

- Pro **text pro zobrazení výsledku** () : Label
    - (name) = label1
    - AutoSize = False
    - Size = 440 ; 120
    - TextAlign = MiddleCenter
    - Font.Name = „Arial Narrow“
    - Font.Size = 12
    - Font.Bold = True

- Tlačítko pro počítání počtu jeho stisknutí () : Button

    - (name) = Button1
    - Size = 140 ; 65
    - Text = "button1"

- Tlačítko pro kopírování textu z textBox1 do label1 () : Button

    - (name) = Button2
    - Size = 140 ; 65
    - Text = "button2"

- Tlačítko pro výpočet matematické operace () : Button

    - (name) = Button3
    - Size = 140 ; 65
    - Text = "button3"

- Zdrojový string pro zobrazení textové zprávy na label1 () : TextBox

    - (name) = textBox1
    - AutoSize = False
    - Size = 440 ; 35
    - TextAlign = Left
    - Font.Name = „Arial Narrow“
    - Font.Size = 12
    - textBox1.text = „Testovací string“

3.4 Události a obslužné rutiny událostí

K vytvoření **událostí** použijeme ikonu událostí v okně [**Vlastnosti**] v aplikaci Visual Studio, viz. obrázek č. 3.

Nyní vytvoříme následující události:

- **FormConverter.Load** (dvojklikem myší)
- **button1_Click** (dvojklikem myši na ovládací prvek button1)
- **button2_Click** (dvojklikem myši na ovládací prvek button2)
- **button3_Click** (dvojklikem myši na ovládací prvek button3)

Událost **Form.Load** se provede při spuštění programu, před zobrazením okna aplikace. Události **button1_Click (), button2_Click () a button3_Click ()** se vykonají při stisknutí příslušné klávesy. Při každé z těchto událostí budeme chtít zobrazit výsledek zpracování programového kódu v ovládacím prvku label1.

Programový kód doplníme o následující programové konstrukce:

1. Definujeme globální proměnnou pro počítání stisků tlačítka „button1“

```
int PocetStisknutiTlacitka = 0;
```

2. Dynamicky přiřadíme ovládacím prvkům vlastnost „#.text“

```
// text pro zobrazení v ovládacích prvcích
Text = "První cvičení s WinForm";
button1.Text = "Stiskni!";
button2.Text = "Text";
button3.Text = "Výpočet";
label1.Text = "";
```

3. Vytvoříme kód pro událost button1_Click

```
PocetStisknutiTlacitka = ++PocetStisknutiTlacitka;
label1.Text = "Tlačítko bylo stisknuto celkem "
                + PocetStisknutiTlacitka.ToString()
                + " x!";
```

4. Vytvoříme kód pro událost button2_Click

```
label1.Text = textBox1.Text;
```

5. Vytvoříme kód pro událost button3_Click

```
int a, b;
double c;

 a = 15;
b = 6;

c = a / b;    // 2

 c = a / Convert.ToDouble(b);      // 2.5

label1.Text = "15/6 = " + c.ToString();
```

3.5 Testování aplikace

Nyní projekt spustíme pomocí [Ctrl+F5] a otestujeme, zda funguje správně.

(Nakov, 2024)

4 Klávesové zkratky pro psaní znaků stiskem kombinací kláves

>	[Pravý ALT + >]	#	[Pravý ALT + V]	^	[Pravý ALT + 3]
<	[Pravý ALT + <]	$	[Pravý ALT + ů]	\	[Pravý ALT + Q]
|	[Pravý ALT + W]	÷	[Pravý ALT + ú]	*˙	[Pravý ALT + -]
&	[Pravý ALT + C]	×	[Pravý ALT + ×]	{	[Pravý ALT + B]
[	[Pravý ALT + F]	€	[Pravý ALT + E]	}	[Pravý ALT + N]
]	[Pravý ALT + G]	~	[Pravý ALT + 1]	‘	[Levý Shift + ň]
@	[Pravý ALT + V]	°	[SHIFT + ;]		

Tip pro VS:

Pro formátování textu zdrojových kódů ve Visual Studiu můžeme použít funkci **-auto format** můžeme použít kombinaci kláves:

- pro celý soubor [CTRL + K + D]
- Pro výběr [CTRL + K + D]

Nebo z hlavní nabídky Upravit -> Upřesnit -> [Formátovat dokument], nebo [Formátovat výběr]

5 Použitá literatura

NAKOV, C. a kol., 2024. *The Book Uses C# and Visual Studio* [online]. [cit. 20.5.2024]. Dostupné na WWW: https://csharp-book.softuni.org/Content/Chapter-1-first-steps-in-programming/exercises-graphical-and-web-apps/exercises-graphical-and-web-apps.html

MICROSOFT Corp. a. s., 2022. MS Visual Studio 2022, verze 17.8.9. 14. 5. 2024 [cit. 25. 5. 2024]. Dostupné na WWW: https://visualstudio.microsoft.com/cs/

6 Studijní literatura:

VIRIUS, Miroslav, 2002. *C# pro zelenáče.* Praha: Neocortex. ISBN 80-72321-76-5.

SELLS, Chris, 2005. *C# a Winforms: programování formulářů ve Windows.* Brno: Zoner Press. ISBN: 80-86815-25-0.

NAGEL, Ch., EVJEN, B., GLYNN, J. a SKINNER, M. W., 2007. *C# 2005 – Programujeme profesionálně.* Brno: Computer Press. ISBN 80-251-1181-4.

7 Otázky k procvičení

1 Co je obsluha události?
2 Jak se vytvářejí obslužné rutiny událostí komponent?
3 Jaký formát textu umožňuje zobrazit TextBox?
4 Kde, v jakém místě programu, se definuje globální proměnná?
5 Kdy se provede metoda Form.Load?

Seznam zkratek

GUI grafické uživatelské rozhraní

VS Visual Studio

Rejstřík

Programování řídicích aplikací

Téma 08: Windows Forms aplikace pro testování aritmetických operací v jazyce C#

Studijní cíl

Seznámit studenty s tvorbou Windows Forms aplikace, založením projektu formulářové aplikace ve Visual studiu, použitím komponent, nastavením jejich vlastností a obsluha jejich událostí. Cílem je vytvořit GUI aplikaci pro testování aritmetických operací.

Doba nutná k nastudování

2 hodiny

Klíčová slova

Programovací jazyk, počítačový program, programový kód, zdrojový kód, kód, C#, algoritmus, vykonávání kódu, kompilace, událost, obsluha události, komponenty, ovládací prvky, Button, Label, Textové pole, GroupBox, Math

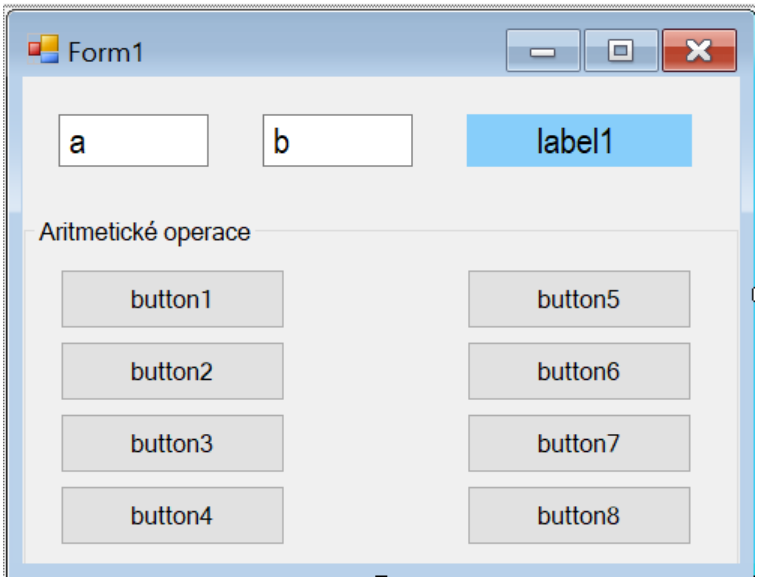
1 Grafická aplikace pro testování aritmetických operací

Cílem je vytvořit **grafickou aplikaci** (GUI aplikaci), která bude mít jeden formulář, osm tlačítek, dvě textové pole pro zadávání operandů aritmetických operací a prvek Label pro zobrazení výsledků těchto operací. Obslužné události tlačítek budou pracovat s vytvořeným obslužným kódem, který každému tlačítku přiřadí jiný typ matematické operace.

Bude provedeno vytvoření aplikace pro realizaci následujících aritmetických operací:

- Součet
- Celočíselné dělení
- Druhou mocninu čísla
- Post inkrementace
- Pre inkrementace
- Výpočet délky přepony obdélníka
- Výpočet úhlu mezi stranami odvěsen trojúhelníka
- Operace pro výpočet zbytku po celočíselném dělení MOD

Na tlačítkách aplikace budou zobrazeny příslušné znaky výše uvedených operací. Tlačítka budou pojmenována dynamicky, v inicializační části kódu aplikace.



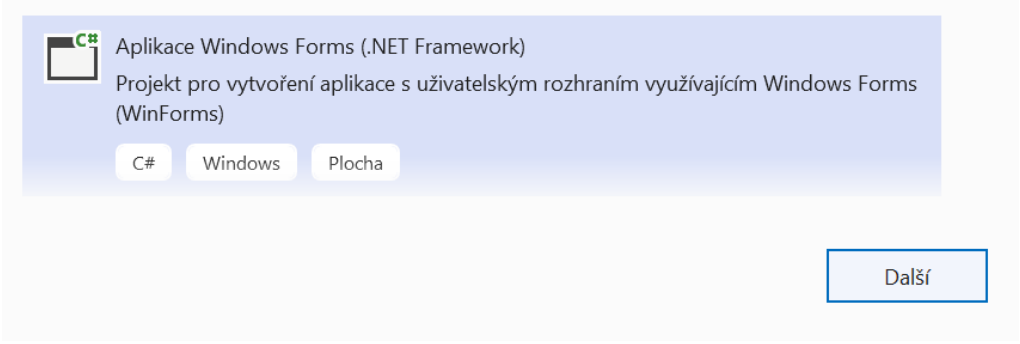
Obr. 1 – Vzhled aplikace (vytvořeno v Microsoft Visual Studio 2022)

V tomto cvičení si procvičíme práci s komponentami aplikace a tvorbou programového kódu pro realizaci požadovaných operací a práci s typem dat.

1.1 Vytvoření nového projektu v jazyce C#

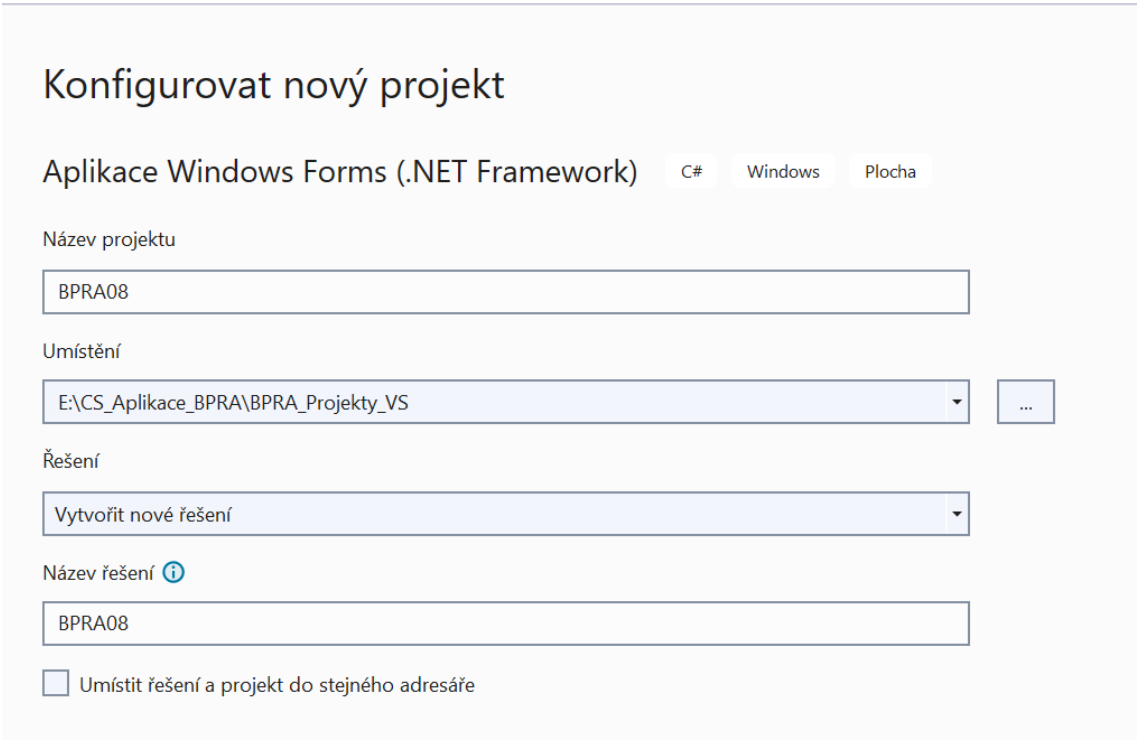
Vytvoříme projekt Windows Forms aplikaci .NET v C# s názvem "BPRA08":

Po spuštění průvodce ve Visual Studiu zvolíme Windows Forms aplikaci pro .NET



Obr. 2 – Založení nového projektu v Windows Forms (printscreen) (Microsoft, 2022)

Do dialogového okna vložíme název projektu a jeho umístění na disku.



Obr. 3 – Založení nového projektu v Windows Forms (printscreen) (Microsoft, 2022)

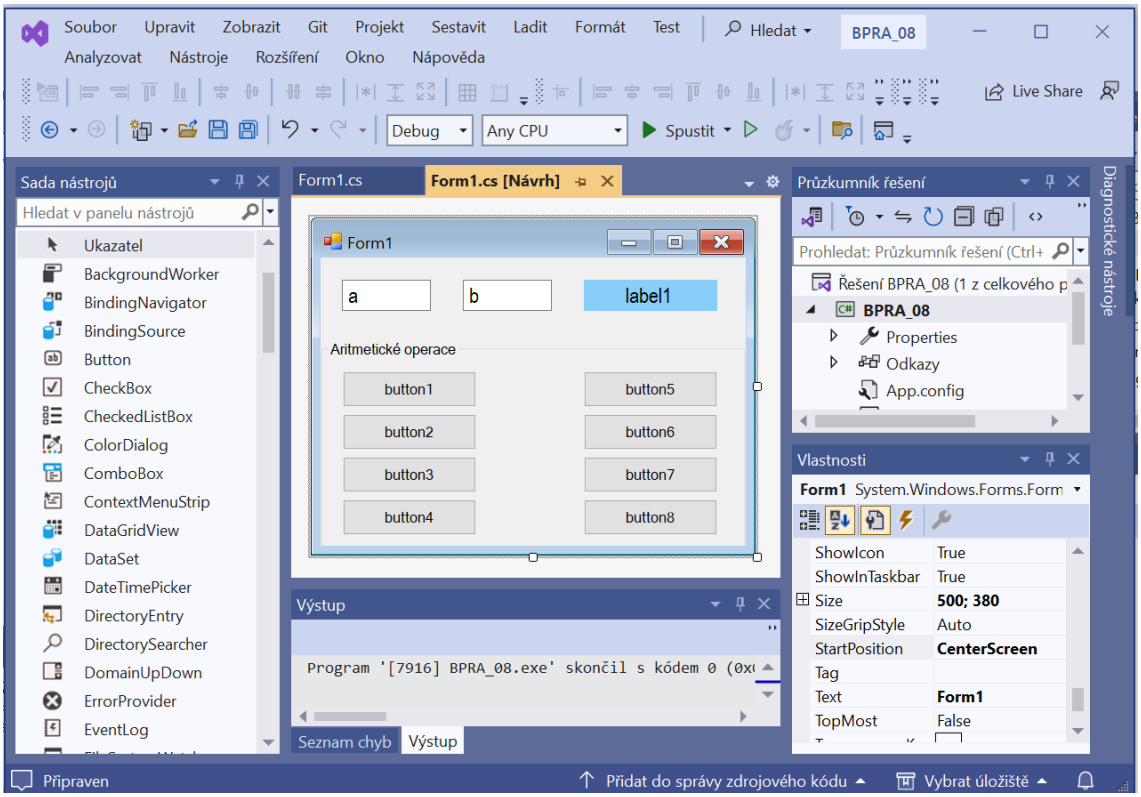
1.2 Přidání ovládacích prvků uživatelského rozhraní

Použijeme následující komponenty UI (User Interface) a rozmístíme je na formuláři:

- **Label** s názvem label1 – pro zobrazení výsledků ar. operací
- **TextBox** s názvem textBox1 – pro zadávání operandu „a“
- **TextBox** s názvem textBox2 – pro zadávání operandu „b“
- **Button** s názvem button1 – pro výpočet aritmetické operace
- **Button** s názvem button2 – pro výpočet aritmetické operace
- **Button** s názvem button3 – pro výpočet aritmetické operace
- **Button** s názvem button4 – pro výpočet aritmetické operace
- **Button** s názvem button5 – pro výpočet aritmetické operace
- **Button** s názvem button6 – pro výpočet aritmetické operace
- **Button** s názvem button7 – pro výpočet aritmetické operace
- **Button** s názvem button8 – pro výpočet aritmetické operace
- **GroupBox** s názvem groupBox1 – pro seskupení tlačítek Button

Po vložení ovládacích prvků je rozmístěte na formuláři tak, jak je vyobrazeno na obr. 4.

Vzhled grafického editoru uživatelského rozhraní může vypadat podobně jako tento:



Obr. 4 – Rozložení komponent v aplikaci (printscreen) (Microsoft, 2022)

1.3 Konfigurace ovládacích prvků uživatelského rozhraní

Nyní je potřeba nastavit vlastnosti vložených ovládacích prvků. Pro ovládací prvky uživatelského rozhraní použijeme následující nastavení:

- Pro **hlavní formulář** (), který obsahuje všechny ovládací prvky: Form
    - (name) = Form1
    - Size = 500 ; 380
    - Text = "Form1"
    - Font.Size = 8
    - MaximizeBox = False
    - MinimizeBox = False
    - StartPosition = CenterScreen
    - FormBorderStyle = FixedSingle

- Pro **text pro zobrazení výsledku** () : Label
    - (name) = label1
    - AutoSize = False
    - Size = 150 ; 35
    - TextAlign = MiddleCenter
    - Font.Name = „Arial Narrow“
    - Font.Size = 12
    - BackColor = „LightSkyBlue“

- Tlačítko pro počítání počtu jeho stisknutí () : Button
    - (name) = button1
    - Size = 150 ; 40
    - Text = "button1"
- Tlačítko pro počítání počtu jeho stisknutí () : Button
    - (name) = button2
    - Size = 150 ; 40
    - Text = "button2"
- Tlačítko pro počítání počtu jeho stisknutí () : Button
    - (name) = button3
    - Size = 150 ; 40
    - Text = "button3"
- Tlačítko pro počítání počtu jeho stisknutí () : Button
    - (name) = button4
    - Size = 150 ; 40
    - Text = "button4"
- Tlačítko pro počítání počtu jeho stisknutí () : Button
    - (name) = button5
    - Size = 150 ; 40
    - Text = "button5"
- Tlačítko pro počítání počtu jeho stisknutí () : Button
    - (name) = button6
    - Size = 150 ; 40
    - Text = "button6"
- Tlačítko pro počítání počtu jeho stisknutí () : Button
    - (name) = button7
    - Size = 150 ; 40
    - Text = "button7"
- Tlačítko pro počítání počtu jeho stisknutí () : Button
    - (name) = button8
    - Size = 150 ; 40
    - Text = "button8"

- Pro **text pro zadání operandu „a“** () : TextBox

    - (name) = textBox1
    - AutoSize = False
    - Size = 100 ; 35
    - TextAlign = Left
    - Font.Name = „Arial Narrow“
    - Font.Size = 12
    - textBox1.text = 7

- Pro **text pro zadání operandu „b“** () : TextBox

    - (name) = textBox2
    - AutoSize = False
    - Size = 100 ; 35
    - TextAlign = Left
    - Font.Name = „Arial Narrow“
    - Font.Size = 12
    - textBox1.text = 6

1.4 Události a obslužné rutiny událostí

K vytvoření **událostí** použijeme ikonu událostí v okně [**Vlastnosti**] v aplikaci Visual Studio, viz. obrázek č. 4.

Nyní vytvoříme následující události:

- **FormConverter.Load** (dvojklikem myší na formulář)
- **button1_Click** (dvojklikem myši na ovládací prvek button1)
- **button2_Click** (dvojklikem myši na ovládací prvek button2)
- **button3_Click** (dvojklikem myši na ovládací prvek button3)
- **button4_Click** (dvojklikem myši na ovládací prvek button3)
- **button5_Click** (dvojklikem myši na ovládací prvek button3)
- **button6_Click** (dvojklikem myši na ovládací prvek button3)
- **button7_Click** (dvojklikem myši na ovládací prvek button3)
- **button8_Click** (dvojklikem myši na ovládací prvek button3)

Událost **Form.Load** se provede při spuštění programu, před zobrazením okna aplikace. Události **button1_Click ()**, **button2_Click ()**, **button3_Click ()**, **button4_Click ()**, **button5_Click ()**, **button6_Click ()**, **button7_Click () a button8_Click ()** se vykonají při stisknutí příslušné klávesy. Při každé z těchto událostí budeme chtít zobrazit výsledek zpracování programového kódu v ovládacím prvku label1.

Programový kód doplníme o následující programové konstrukce:

1. Dynamicky přiřadíme ovládacím prvkům vlastnost „#.text“

   Text = "BPRA_08";

```
button1.Text = "+";
button2.Text = "/";
button3.Text = "MOD";
button4.Text = "mocnina";
button5.Text = "přepona";
button6.Text = "úhel";
button7.Text = "a++";
button8.Text = "++a";
label1.Text = "";
//počáteční hodnotu textových polí, aby nebyla prázdná
textBox1.Text = "7";
textBox2.Text = "6";
```

2. Vytvoříme kód pro událost button1_Click

```
int a = Convert.ToInt32(textBox1.Text);
int b = Convert.ToInt32(textBox2.Text);
int c = a + b;
label1.Text = Convert.ToString(c);
```

3. Vytvoříme kód pro událost button2_Click

```
double a = Convert.ToDouble(textBox1.Text);
double b = Convert.ToDouble(textBox2.Text);
double c = Convert.ToDouble(a) / b;  //konverze typu je vyžadována pro celá
                                     //data
label1.Text = Convert.ToString(c);
```

4. Vytvoříme kód pro událost button3_Click

```
double a = Convert.ToDouble(textBox1.Text);
double b = Convert.ToDouble(textBox2.Text);
double c = Convert.ToDouble(a) % b;
label1.Text = Convert.ToString(c);
```

5. Vytvoříme kód pro událost button4_Click

```
double a = Convert.ToDouble(textBox1.Text);
double b = Convert.ToDouble(textBox2.Text);
double c = Math.Pow(a, b); //Math.Pow umocňování
label1.Text = c.ToString("F2");
```

6. Vytvoříme kód pro událost button5_Click

```
double a = Convert.ToDouble(textBox1.Text);
```

```
double b = Convert.ToDouble(textBox2.Text);
double c = Math.Pow(a, b); //Math.Pow umocňování
label1.Text = c.ToString("F2");
```

7. Vytvoříme kód pro událost button6_Click

```
double a = Convert.ToDouble(textBox1.Text);
double b = Convert.ToDouble(textBox2.Text);
double uhel = Math.Atan(a / b);
uhel *= 180 / Math.PI;
label1.Text = uhel.ToString("F2") + " °";
```

8. Vytvoříme kód pro událost button7_Click

```
int a = Convert.ToInt32(textBox1.Text);
int b = a++;
label1.Text = "a:" + Convert.ToString(a)
+ " b:" + Convert.ToString(b);
```

9. Vytvoříme kód pro událost button8_Click

```
int a = Convert.ToInt32(textBox1.Text);
int b = a++;
label1.Text = "a:" + Convert.ToString(a)
 + " b:" + Convert.ToString(b);
```

1.5 Testování aplikace

Nyní projekt spustíme pomocí [Ctrl+F5] a otestujeme, zda funguje správně.

(Szakály, 2024)

2 Klávesové zkratky pro psaní znaků stiskem kombinací kláves

>	[Pravý ALT + >]	]	[Pravý ALT + G]	×	[Pravý ALT + ×]
<	[Pravý ALT + <]	@	[Pravý ALT + V]	€	[Pravý ALT + E]
|	[Pravý ALT + W]	#	[Pravý ALT + V]	~	[Pravý ALT + 1]
&	[Pravý ALT + C]	$	[Pravý ALT + ů]	°	[SHIFT + ;]
[	[Pravý ALT + F]	÷	[Pravý ALT + ú]	^	[Pravý ALT + 3]

\	[Pravý ALT + Q]	{	[Pravý ALT + B]	‘	[Levý SHIFT + ň]
*˙	[Pravý ALT + -]	}	[Pravý ALT + N]		

Tip pro VS:

Pro formátování textu zdrojových kódů ve Visual Studiu můžeme použít funkci **-auto format** můžeme použít kombinaci kláves:

- pro celý soubor [CTRL + K + D]
- Pro výběr [CTRL + K + D]

Nebo z hlavní nabídky Upravit -> Upřesnit -> [Formátovat dokument], nebo [Formátovat výběr]

3 Použité zdroje

SZAKÁLY, Norbert, 2024. *GÉPÉSZETI INFORMATIKA GYAKORLAT* [online]. [cit. 20.5.2024]. Dostupný na WWW: http://www.inflab.bme.hu/~szakaly/gepinfo/

MICROSOFT Corp. a. s., 2022. MS Visual Studio 2022, verze 17.8.9. 14. 5. 2024 [cit. 25. 5. 2024]. Dostupné na WWW: https://visualstudio.microsoft.com/cs/

4 Studijní literatura

VIRIUS, Miroslav, 2002. *C# pro zelenáče.* Praha: Neocortex. ISBN 80-72321-76-5.

SELLS, Chris, 2005. *C# a Winforms: programování formulářů ve Windows.* Brno: Zoner Press. ISBN: 80-86815-25-0.

NAGEL, Ch., EVJEN, B., GLYNN, J. a SKINNER, M. W., 2007. *C# 2005 – Programujeme profesionálně.* Brno: Computer Press. ISBN 80-251-1181-4.

5 Otázky k procvičení

1 Jak vytvoříme událost „button1_Click“?
2 Jak se vytvářejí obslužné rutiny událostí komponent?
3 Jaký příkaz použijeme pro výpočet mocniny?
4 Jaký příkaz použijeme pro výpočet odmocniny?

Seznam zkratek

GUI grafické uživatelské rozhraní

VS Visual Studio

Rejstřík

Programování řídicích aplikací

Téma 09: Windows Forms aplikace, binární operace, větvení programu v jazyce C#

Studijní cíl

Seznámit studenty s tvorbou Windows Forms aplikace, založením projektu formulářové aplikace ve Visual studiu, použitím komponent, nastavením jejich vlastností a obsluha jejich událostí. Cílem je vytvořit GUI aplikaci pro testování větvení v programu a binárních operací.

Doba nutná k nastudování

2 hodiny

Klíčová slova

Programovací jazyk, počítačový program, programový kód, zdrojový kód, kód, C#, algoritmus, vykonávání kódu, kompilace, událost, obsluha události, komponenty, ovládací prvky, Button, Label, Textové pole, GroupBox, Math

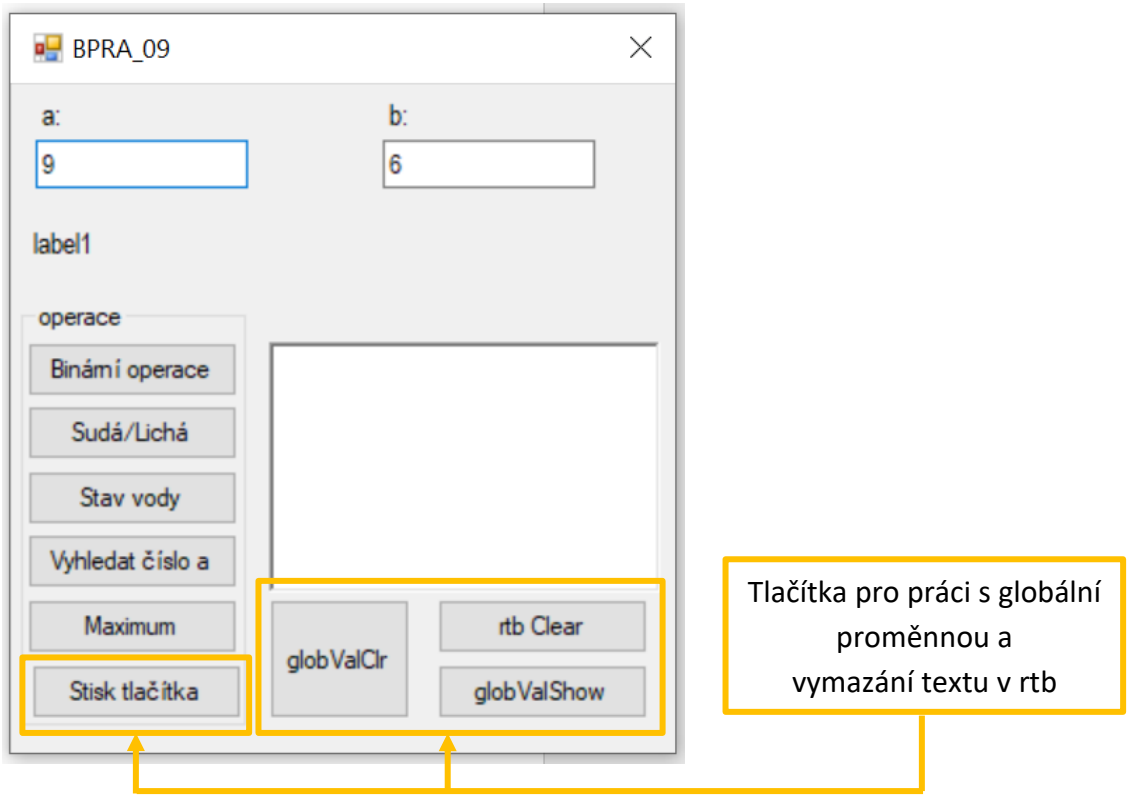
1 Grafická aplikace pro testování binárních operací

Cílem je vytvořit **grafickou aplikaci** (GUI aplikaci), která bude mít jeden formulář, „group Box“ pro seskupení tlačítek v levé části formuláře, celkem devět tlačítek, dvě textová pole pro zadávání operandů binárních operací, několik prvků typu Label pro zobrazení upřesňujícího komentáře aktuálně vykonávané operace a prvek „rich Text box“ pro zobrazení výsledků těchto operací. Obslužné události tlačítek budou pracovat s vytvořeným obslužným kódem, který každému tlačítku přiřadí jiný typ binární operace.

Bude provedeno vytvoření aplikace pro realizaci následujících binárních operací:

- Zobrazí výsledky vybraných binárních operací v komponentě „rich Text box“
- Určení, zda číslo v „a:“ je sudé, nebo liché
- Podle zadané teploty v proměnné „a:“ určí skupenství vody
- Zobrazí výsledky vyhledání čísla zadaného v „a:“ v metodě „switch“
- Určení, které číslo je větší, zda „a:“, nebo „b“.
- Uložení počtu stisků tlačítka do globální proměnné.

Na tlačítkách aplikace budou zobrazeny příslušné znaky výše uvedených operací. Tlačítka budou pojmenována dynamicky, v inicializační části kódu aplikace.



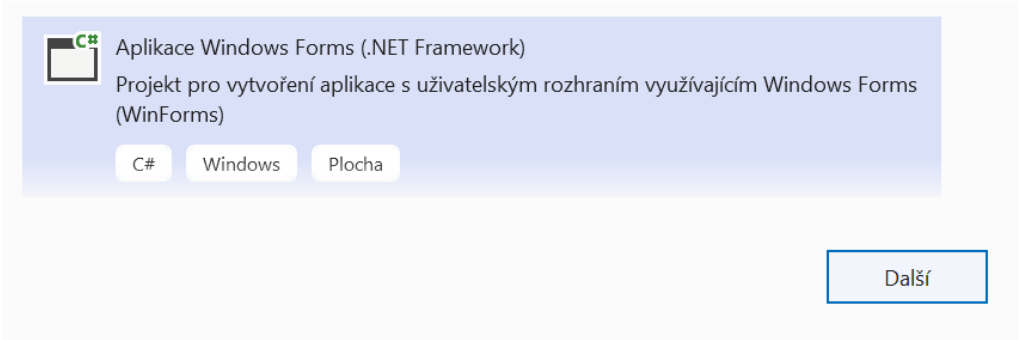
Obr. 1 – Vzhled aplikace (vytvořeno v Microsoft Visual Studio 2022)

V tomto cvičení si procvičíme práci s komponentami aplikace a tvorbou programového kódu pro realizaci požadovaných operací a práci s různým typem dat.

1.1 Vytvoření nového projektu v jazyce C#

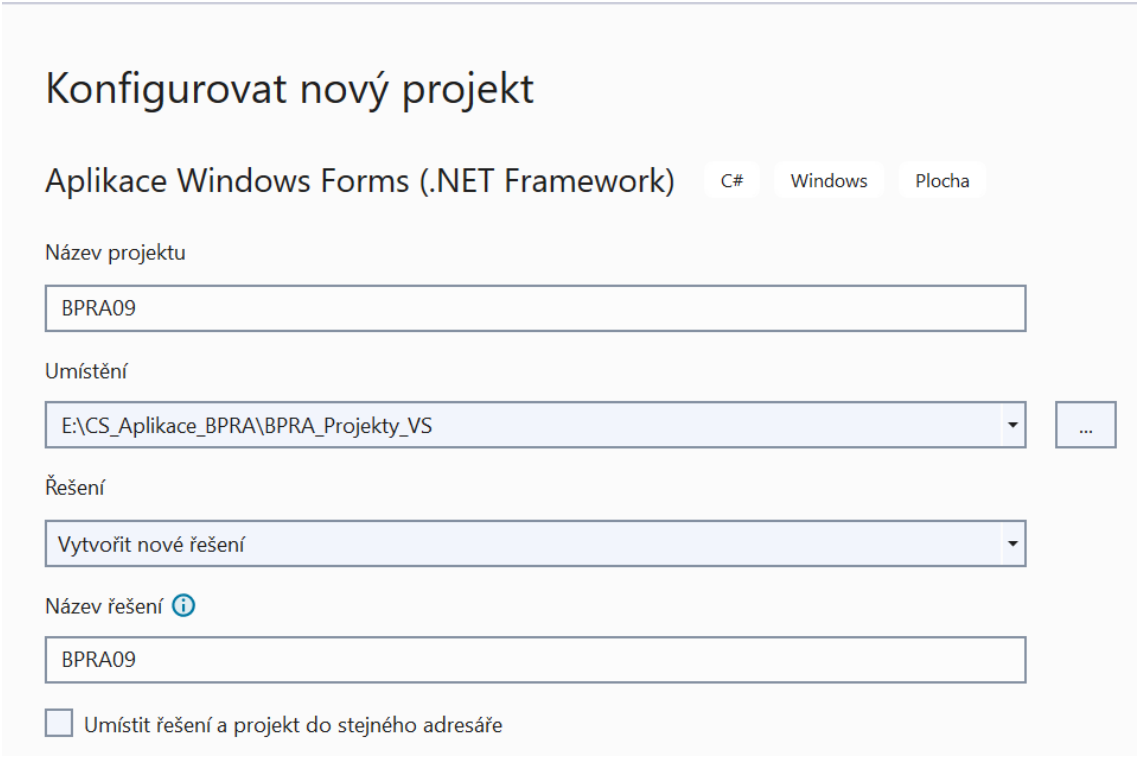
Vytvoříme projekt Windows Forms aplikaci .NET v C# s názvem "BPRA09":

Po spuštění průvodce ve Visual Studiu zvolíme Windows Forms aplikaci pro .NET



Obr. 2 – Založení nového projektu v Windows Forms (printscreen) (Microsoft, 2022)

Do dialogového okna vložíme název projektu a jeho umístění na disku.



Obr. 3 – Založení nového projektu v Windows Forms (printscreen) (Microsoft, 2022)

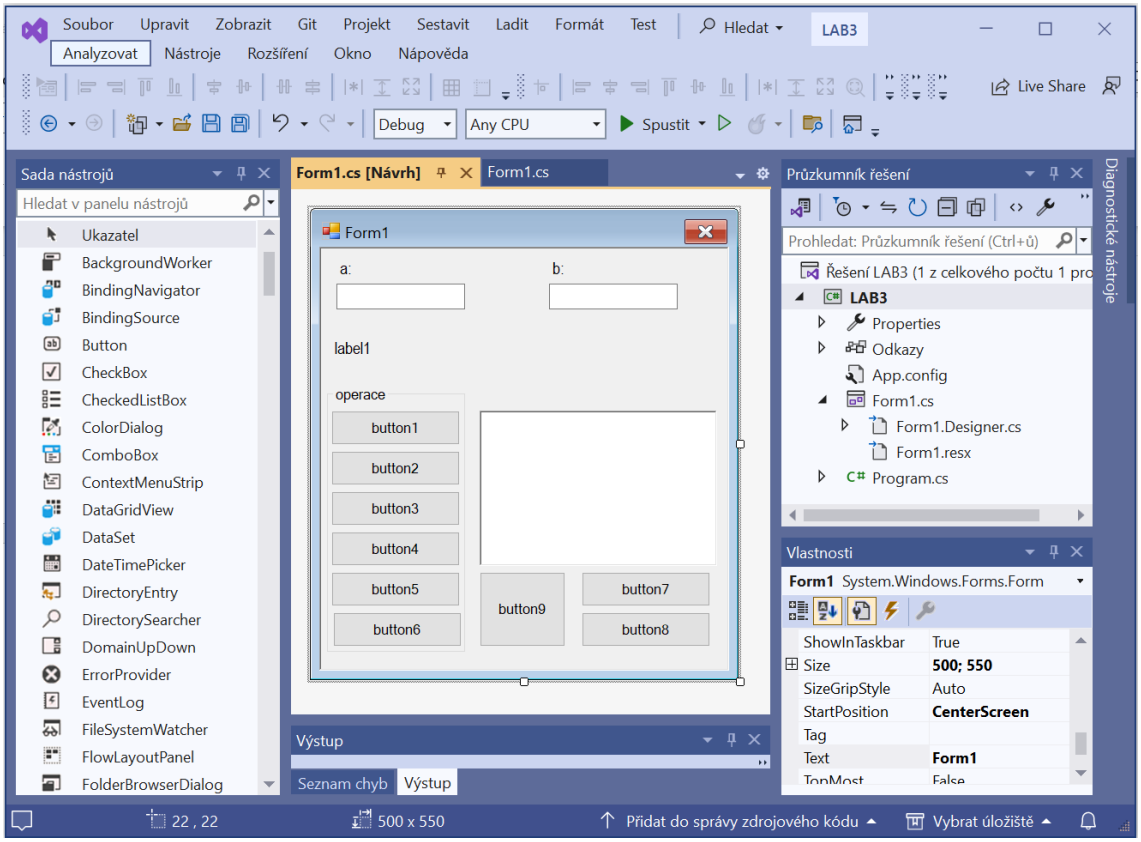
1.2 Přidání ovládacích prvků uživatelského rozhraní

Použijeme následující komponenty UI (User Interface) a rozmístíme je na formuláři:

- **Label** s názvem label1 - pro zobrazení textu popisu provedené operace
- **Label** s názvem label2 - pro zobrazení textu popisu názvu proměnné „a“
- **Label** s názvem label3 - pro zobrazení textu popisu názvu proměnné „b“
- **TextBox** s názvem textBox1 - pro zadávání operandu „a“
- **TextBox** s názvem textBox2 - pro zadávání operandu „b“
- **richTextBox s názvem richTextBox1** - pro zobrazení výsledků operací
- **Button** s názvem button1 - pro provedení příslušné operace
- **Button** s názvem button2 - pro provedení příslušné operace
- **Button** s názvem button3 - pro provedení příslušné operace
- **Button** s názvem button4 - pro provedení příslušné operace
- **Button** s názvem button5 - pro provedení příslušné operace
- **Button** s názvem button6 - pro provedení příslušné operace
- **Button** s názvem button7 - pro provedení příslušné operace
- **Button** s názvem button8 - pro provedení příslušné operace
- **Button** s názvem button9 - pro provedení příslušné operace
- **GroupBox** s názvem groupBox1 - pro seskupení tlačítek Button 1 – 6

Po vložení ovládacích prvků je rozmístěte na formuláři tak, jak je vyobrazeno na obr. 4.

Vzhled grafického editoru uživatelského rozhraní může vypadat podobně jako tento:



Obr. 4 – Rozložení komponent v aplikaci (printscreen) (Microsoft, 2022)

1.3 Konfigurace ovládacích prvků uživatelského rozhraní

Nyní je potřeba nastavit vlastnosti vložených ovládacích prvků. Pro ovládací prvky uživatelského rozhraní použijeme následující nastavení:

- Pro **hlavní formulář** (), který obsahuje všechny ovládací prvky: Form
    - (name) = Form1
    - Size = 500 ; 550
    - Text = "Form1"
    - Font.Size = 8
    - MaximizeBox = False
    - MinimizeBox = False
    - StartPosition = CenterScreen
    - FormBorderStyle = Fixed3D

- Pro **text pro zadání operandu „a“** () : TextBox
    - (name) = textBox1

- AutoSize = False
- Size = 150 ; 30
- TextAlign = Left
- Font.Name = „Arial Narrow“
- Font.Size = 10
- textBox1.text = „“

- Pro **text pro zadání operandu „b“** () : TextBox

  - (name) = textBox2
  - AutoSize = False
  - Size = 150 ; 30
  - TextAlign = Left
  - Font.Name = „Arial Narrow“
  - Font.Size = 10
  - textBox1.text = 6

- Pro **text pro zobrazení výsledku** () : RichTextBox

  - (name) = richTextBox1
  - Size = 277 ; 181
  - TextAlign = MiddleCenter
  - Font.Name = „Arial Narrow“
  - Font.Size = 12

- Pro **text pro zobrazení popisku aktuální operace** () : Label

  - (name) = label1
  - AutoSize = False
  - Size = 450 ; 40
  - TextAlign = MiddleCenter
  - Font.Name = „Arial Narrow“
  - Font.Size = 12
  - BorderStyle = FixedSingle

- Pro seskupení tlačítek 1 až 6 () : GroupBox

  - (name) = groupBox1
  - Size = 161 ; 311
  - Text = "operace"

- Tlačítko „Binární operace“ () : Button

  - (name) = button1
  - Size = 150 ; 40
  - Text = "button1"

- Tlačítko „Sudá / Lichá“ () : Button

    - (name) = button2
    - Size = 150 ; 40
    - Text = "button2"

- Tlačítko „Skupenství vody“ () : Button

    - (name) = button3
    - Size = 150 ; 40
    - Text = "button3"

- Tlačítko „Vyhledat číslo a“ () : Button

    - (name) = button4
    - Size = 150 ; 40
    - Text = "button4"

- Tlačítko „Maximum“ () : Button

    - (name) = button5
    - Size = 150 ; 40
    - Text = "button5"

- Tlačítko „Stisk tlačítka“ () : Button

    - (name) = button6
    - Size = 150 ; 40
    - Text = "button6"

- Tlačítko „rtb Clear“ () : Button

    - (name) = button7
    - Size = 150 ; 40
    - Text = "button7"

- Tlačítko „globValShow“ () : Button

    - (name) = button8
    - Size = 150 ; 40
    - Text = "button8"

- Tlačítko“globValClr“ () : Button

    - (name) = button8
    - Size = 100 ; 87
    - Text = "button9"

1.4 Události a obslužné rutiny událostí

K vytvoření **událostí** použijeme ikonu událostí v okně [**Vlastnosti**] v aplikaci Visual Studio, viz. obr. 3.

Nyní vytvoříme následující události:

- **FormConverter.Load** (dvojklikem myší na formulář)
- **button1_Click** (dvojklikem myši na ovládací prvek button1)
- **button2_Click** (dvojklikem myši na ovládací prvek button2)
- **button3_Click** (dvojklikem myši na ovládací prvek button3)
- **button4_Click** (dvojklikem myši na ovládací prvek button4)
- **button5_Click** (dvojklikem myši na ovládací prvek button5)
- **button6_Click** (dvojklikem myši na ovládací prvek button6)
- **button7_Click** (dvojklikem myši na ovládací prvek button7)
- **button8_Click** (dvojklikem myši na ovládací prvek button8)
- **button9_Click** (dvojklikem myši na ovládací prvek button9)

Událost **Form1_Load** se provede při spuštění programu, před zobrazením okna aplikace. Události **button1_Click ()**, **button2_Click ()**, **button3_Click ()**, **button4_Click ()**, **button5_Click ()**, **button6_Click ()**, **button7_Click ()**, **button8_Click ()** a **button9_Click ()** se vykonají při stisknutí příslušné klávesy. Při každé z těchto událostí budeme chtít zobrazit výsledek zpracování programového kódu v ovládacím prvku richTextBox1.

Programový kód doplníme o následující programové konstrukce:

1. Dynamicky přiřadíme ovládacím prvkům vlastnost „#.text“

```
Text = "BPRA_09";
textBox1.Text = "9";
textBox2.Text = "6";
button1.Text = "Binární operace";
button2.Text = "Sudá/Lichá";
button3.Text = "Skupenství vody";
button4.Text = "Vyhledat číslo a";
button5.Text = "Maximum";
button6.Text = "Stisk tlačítka";
button7.Text = "rtb Clear";
button8.Text = "globValShow";
button9.Text = "globValClr";
richTextBox1.Text = "";
label2.Text = "a:";
label3.Text = "b:";
```

2. Vytvoříme kód pro událost button1_Click

```
// Binární operace
// | , & , ^ , ~ , << , >>
```

```
int a = Convert.ToInt32(textBox1.Text);
int b = Convert.ToInt32(textBox2.Text);
richTextBox1.Text = "";
richTextBox1.Text += $"a = {a}\n";
// invertování
richTextBox1.Text += $"~a = {~a}\n";
// binární OR
richTextBox1.Text += $"a|b = {a | b}\n";
// binární AND
richTextBox1.Text += $"a&b = {a & b}\n";
// binární XOR
richTextBox1.Text += $"a^b = {a ^ b}\n";
richTextBox1.Text += $"(a^b)^b = {(a ^ b) ^ b}\n";
// rotace doleva
richTextBox1.Text += $"a<<1 = {a << 1}\n";
// rotace doprava
richTextBox1.Text += $"a>>1 = {a >> 1}\n";
```

3. Vytvoříme kód pro událost button2_Click

```
// Větvení
// if (podmínka) příkazy;
// else příkazy;

int  a = Convert.ToInt32(textBox1.Text);
if (a % 2 == 0) richTextBox1.Text = "Sudé číslo";
//if (a % 2 != 0) label1.Text = "Sudé číslo";
else richTextBox1.Text = "Liché číslo";
```

4. Vytvoříme kód pro událost button3_Click

```
// vyhodnocení podmínek pro if ; else ; if else
// if (podmínka1) příkaz1;
// else if (podmínka2) příkaz2;
// else if (podmínka3) příkaz3;
// podmínka nenalezena výpis4;

double temp = Convert.ToDouble(textBox1.Text);

// vždy kontroluje každý řádek zvlášť
if (temp >= 100) richTextBox1.Text = "Pára";
if (temp <= 0) richTextBox1.Text = "Led";
if (temp < 100 && temp > 0) richTextBox1.Text = "Kapalina";

// vnořené do sebe, pokud je true, ostatní části již nejsou kontrolovány
if (temp < 0) richTextBox1.Text = "Led";
```

```
else if (temp > 100) richTextBox1.Text = "Pára";
else richTextBox1.Text = "Kapalina";

// logické AND(&&), logické NEBO(||)
if (temp > 0 && temp < 100) richTextBox1.Text = "Kapalina";
if (temp < 0 || temp > 100) richTextBox1.Text = "Není kapalina";

// s instrukčním blokem
if (temp < 0)
{
   richTextBox1.Text = "Led";
}
else if (temp > 100)
{
   richTextBox1.Text = "Pára";
}
else
{
   richTextBox1.Text = "Kapalina";
}
```

5. Vytvoříme kód pro událost button4_Click

```
int Cislo = Convert.ToInt32(textBox1.Text);

switch (Cislo)
{
   case 5: // případů v libovolném pořadí
      richTextBox1.Text = "Zadané číslo je číslo 5";
      break; //break je povinný
   case 0:
      richTextBox1.Text = "Zadané číslo je číslo 0";
      break;
   case 1:
      richTextBox1.Text = "Zadané číslo je číslo 1";
      break;
   case 2:
      richTextBox1.Text = "Zadané číslo je číslo 2";
      break;
   case 3:
      richTextBox1.Text = "Zadané číslo je číslo 3";
      break;
   case 4:
      richTextBox1.Text = "Zadané číslo je číslo 4";
      break;
   default: // pokud neexistuje případ, který jsme testovali
```

```
        richTextBox1.Text = "Neznámé číslo!";
        break;
    }
```

6. Vytvoříme kód pro událost button5_Click

```
// Instrukce se 3 operandy

double a = Convert.ToDouble(textBox1.Text);
double b = Convert.ToDouble(textBox2.Text);

// s větvením if else

if (a > b) richTextBox1.Text = a.ToString();
else richTextBox1.Text = b.ToString();

// s pomocnou proměnnou max

double max = a;
if (a < b) max = b;

// se zkráceným zápisem; (podmínka)? true_value : false_value;

max = (a > b) ? a : b;
richTextBox1.Text = max.ToString();

globVal = 0;
```

7. Vytvoříme kód pro událost button6_Click

```
// lokální proměnná, lze ji použít pouze v daném bloku

int a = 5;
// další instrukční blok, stejná proměnná jinou hodnotu

{
// lokální proměnná ve vnitřním bloku příkazů

int b = 6;
}
// b zde neexistuje, takže jej nelze použít

//a = b;

// globální proměnnou lze předefinovat lokálně,

// ale pak se použije místní

// int proměnná = 0;

globVal++;
```

```
richTextBox1.Text = $"Tlačítko bylo stisknuto celkem {globVal} x";
```

8. Vytvoříme kód pro událost button7_Click

```
// smažeme obsah rich Text boxu
richTextBox1.Text = "";
```

9. Vytvoříme kód pro událost button8_Click

```
// smažeme obsah rich Text boxu
richTextBox1.Text = $"Hodnota globální proměnné je {globVal}";
```

10. Vytvoříme kód pro událost button8_Click

```
// smažeme obsah rich Text boxu
globVal = 0;
richTextBox1.Text = $"Hodnota globální proměnné je {globVal}";
```

1.5 Testování aplikace

Nyní projekt spustíme pomocí [Ctrl+F5] a otestujeme, zda funguje správně.

(Szakály, 2024)

2 Klávesové zkratky pro psaní znaků stiskem kombinací kláves:

>	[Pravý ALT + >]	#	[Pravý ALT + V]	ˆ	[Pravý ALT + 3]
<	[Pravý ALT + <]	$	[Pravý ALT + ů]	\	[Pravý ALT + Q]
|	[Pravý ALT + W]	÷	[Pravý ALT + ú]	*˙	[Pravý ALT + -]
&	Pravý ALT + C]	×	[Pravý ALT + ×]	{	[Pravý ALT + B]
[	Pravý ALT + F]	€	[Pravý ALT + E]	}	[Pravý ALT + N]
]	Pravý ALT + G]	~	Pravý ALT + 1]	‘	[Levý SHIFT + ň]
@	[Pravý ALT + V]	°	[SHIFT + ;]		

Tip pro VS:

Pro formátování textu zdrojových kódů ve Visual Studiu můžeme použít funkci **-auto format** můžeme použít kombinaci kláves:

- pro celý soubor [CTRL + K + D]
- Pro výběr [CTRL + K + D]

Nebo z hlavní nabídky Upravit -> Upřesnit -> [Formátovat dokument], nebo [Formátovat výběr]

3 Použité zdroje

SZAKÁLY, Norbert. *GÉPÉSZETI INFORMATIKA GYAKORLAT* [online]. [cit. 20.5.2024]. Dostupný na WWW: http://www.inflab.bme.hu/~szakaly/gepinfo/

MICROSOFT Corp. a. s., 2022. MS Visual Studio 2022, verze 17.8.9. 14. 5. 2024 [cit. 25. 5. 2024]. Dostupné na WWW: https://visualstudio.microsoft.com/cs/

4 Studijní literatura

VIRIUS, Miroslav, 2002. *C# pro zelenáče.* Praha: Neocortex. ISBN 80-72321-76-5.

SELLS, Chris, 2005. *C# a Winforms: programování formulářů ve Windows.* Brno: Zoner Press. ISBN: 80-86815-25-0.

NAGEL, Ch., EVJEN, B., GLYNN, J. a SKINNER, M. W., 2007. *C# 2005 – Programujeme profesionálně.* Brno: Computer Press. ISBN 80-251-1181-4.

5 Otázky k procvičení

1 Jak se liší binární operace od aritmetických?
2 Kdy se vykoná metoda Form_Load?
3 Jak se definuje globální proměnná?
4 Jak se definuje lokální proměnná?

Seznam zkratek

GUI grafické uživatelské rozhraní

VS Visual Studio

Rejstřík

Programování řídicích aplikací

Téma 10: Sériové porty počítače v jazyce C#

Studijní cíl

Seznámit studenty s tvorbou Windows Forms aplikace pro práci se sériovým portem osobního počítače. Cílem je vytvořit GUI aplikaci pro identifikaci, výběr a zobrazení aktuálně dostupných portů PC. Součástí je i práce s virtuálním sériovým portem pro vytvoření virtuálního spojení dvou zařízení, komunikujícího prostřednictvím sériového portu.

Doba nutná k nastudování

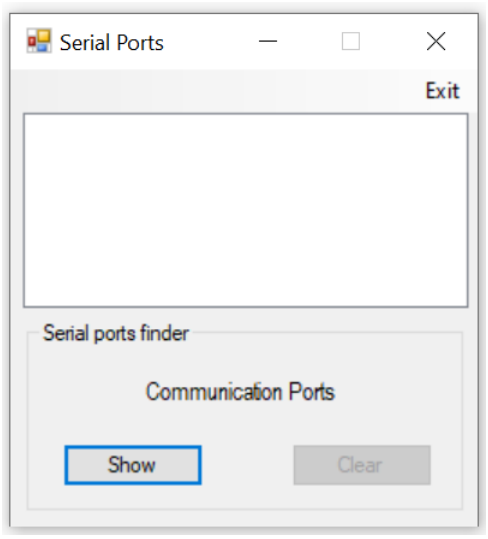
2 hodiny

Klíčová slova

Sériový port, zdrojový kód, kód, C#, algoritmus, událost, obsluha události, komponenty, ovládací prvky, Button, Label, Textové pole, GroupBox

1 Sériové porty počítače

Cílem je vytvořit **grafickou aplikaci** , která bude mít jeden formulář, dvě tlačítka, ovládací prvek ListBox, GroupBox, Label a menuStrip (pro ukončení aplikace Exit).

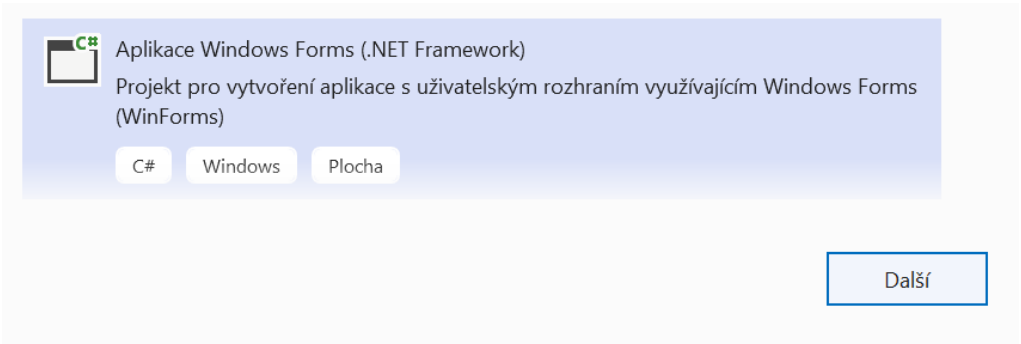


Obr. 1 – Vzhled aplikace (vytvořeno v Microsoft Visual Studio 2022)

1.1 Vytvoření nového projektu v jazyce C#

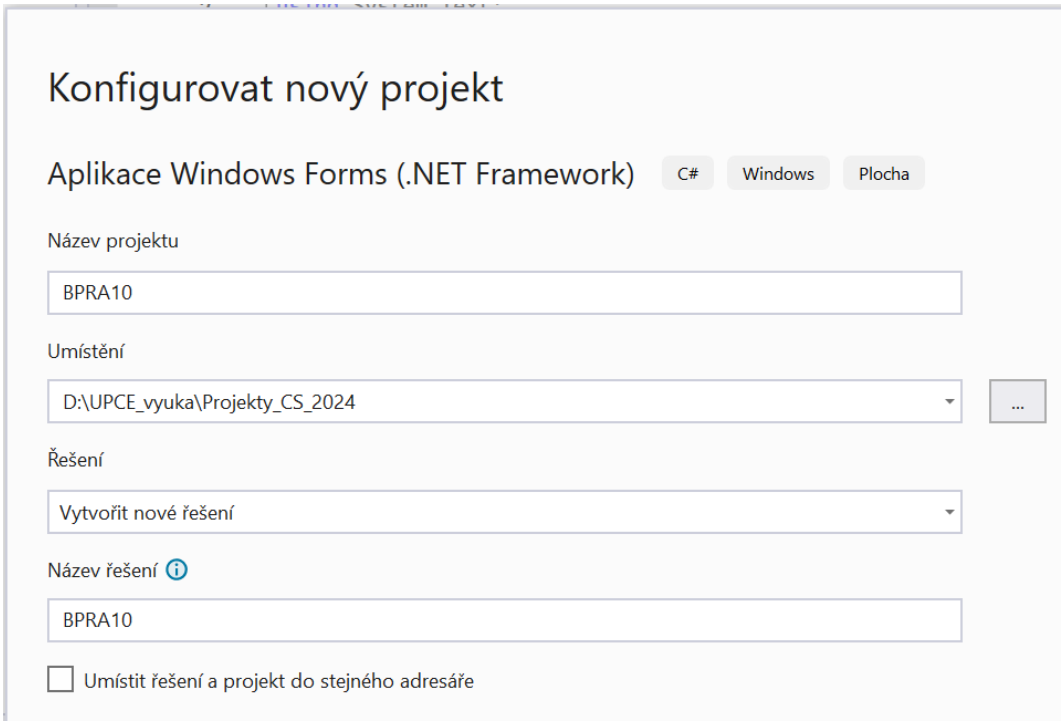
Vytvoříme projekt Windows Forms aplikaci .NET v C# s názvem "BPRA10":

Po spuštění průvodce ve Visual Studiu zvolíme Windows Forms aplikaci pro .NET



Obr. 2 – Založení nového projektu v Windows Forms (printscreen) (Microsoft, 2022a)

Do dialogového okna vložíme název projektu a jeho umístění na disku.



Obr. 3 – Založení nového projektu v Windows Forms (printscreen) (Microsoft, 2022a)

1.2 Přidání ovládacích prvků uživatelského rozhraní

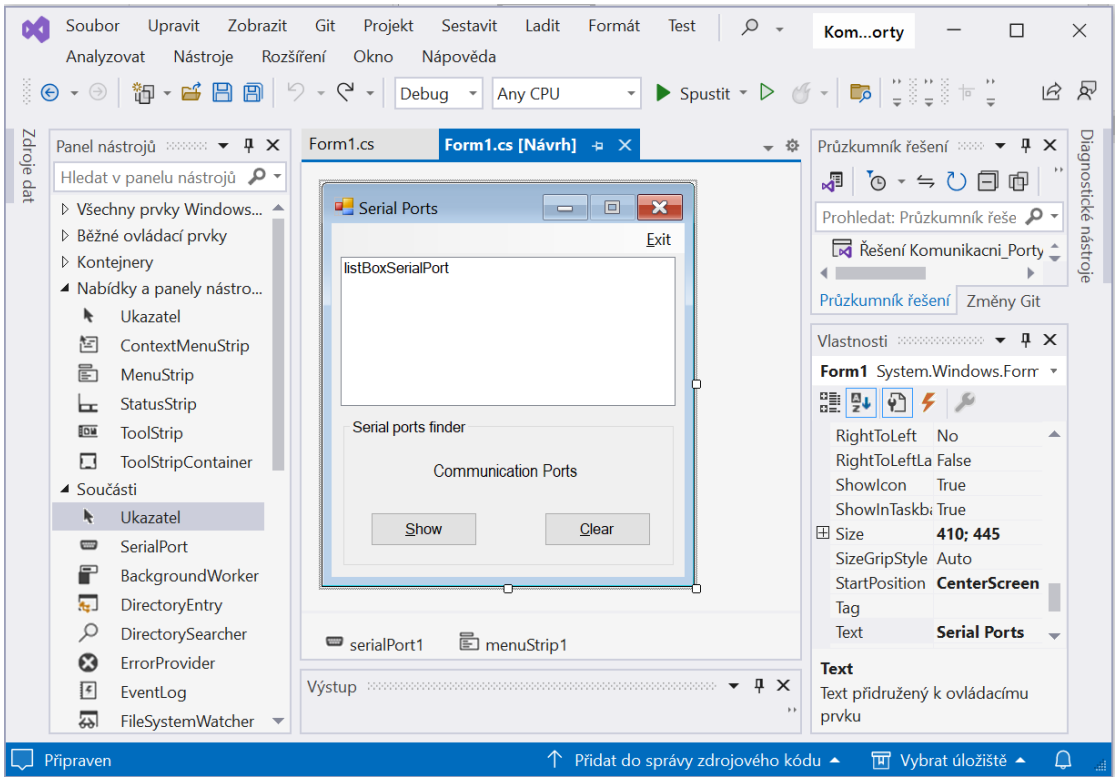
Použijeme následující komponenty UI (User Interface) a rozmístíme je na formuláři:

- **Label** s názvem labelSelected – pro zobrazení vybraného portu

- **ListBox** s názvem listBoxSerialPort – pro zobrazení vyhledaných portů
- **Button** s názvem ShowSerialPorts – pro vyhledání a zobrazení sériových portů
- **Button** s názvem button2 – pro výpočet aritmetické operace
- **GroupBox** s názvem groupBox1 – pro seskupení tlačítek Button
- **MenuStrip** s názvem menuStrip1 – pro seskupení tlačítek Button

Po vložení ovládacích prvků je rozmístěte na formuláři tak, jak je vyobrazeno na obr. 4.

Vzhled grafického editoru uživatelského rozhraní může vypadat podobně jako tento:



Obr. 4 – Rozložení komponent v aplikaci (printscreen) (Microsoft, 2022a)

1.3 Konfigurace ovládacích prvků uživatelského rozhraní

Nyní je potřeba nastavit vlastnosti vložených ovládacích prvků. Pro ovládací prvky uživatelského rozhraní použijeme následující nastavení:

- Pro **hlavní formulář** (), který obsahuje všechny ovládací prvky: Form
    - (name) = Form1
    - Size = 410 ; 445
    - Text = "Form1"
    - Font.Size = 8
    - MaximizeBox = False

    - MinimizeBox = True
    - StartPosition = CenterScreen
    - FormBorderStyle = FixedSingle

- **Pro zobrazení dostupných portů** () : Label

    - (name) = labelSelected
    - AutoSize = False
    - Size = 307 ; 31
    - TextAlign = MiddleCenter
    - Font.Name = „Arial Narrow“
    - Font.Size = 10
    - Text = „Communication Ports“

- Tlačítko pro vyhledání a zobrazení portů () : Button

    - (name) = ShowSerialPorts
    - Size = 117 ; 37
    - Text = "&Show"

- Tlačítko pro vymazání výsledků vyhledávání () : Button

    - (name) = buttonClr
    - Size = 117 ; 37
    - Text = "&Clear"

- MenuStrip (z ovládacích prvků viz. Obr. 4) pro Exit () : menuStrip
    - (name) = menuStrip1
    - Položka "&Exit"

- SerialPort (z ovládacích prvků viz. Obr. 4) () : serialPort1
    - (name) = serialPort1

Po vložení MenuStrip a SerialPort na formulář, tyto komponenty na formuláři nezůstanou zobrazeny, ale budou umístěny se do lišty pod návrhem formuláře.

1.4 Události a obslužné rutiny událostí

K vytvoření **událostí** použijeme ikonu událostí v okně [**Vlastnosti**] v aplikaci Visual Studio, viz. obr. 4.

Nyní vytvoříme následující události:

- Form1_Load (dvojklikem myší na formulář)
- ShowSerialPorts_Click (dvojklikem myši na ovládací prvek Show)

- buttonClr_Click (dvojklikem myši na ovládací prvek Clear)

Událost **Form.Load** se provede při spuštění programu, před zobrazením okna aplikace. Události Form1_Load (), ShowSerialPorts_Click (), buttonClr_Click (), ShowSelected () a exitToolStripMenuItem_Click () se vykonají při stisknutí příslušné klávesy. Při každé z těchto událostí budeme chtít zobrazit výsledek zpracování programového kódu v ovládacím prvku label1.

Programový kód doplníme o následující programové konstrukce:

1. Vytvoříme událost Form1_Load

```
listBoxSerialPort.Enabled = false;
ShowSerialPorts.Enabled = true;
buttonClr.Enabled = false;
```

2. Vytvoříme kód pro událost ShowSerialPorts_Click

```
// smazat položky v listboxu
listBoxSerialPort.Enabled = true;
listBoxSerialPort.Items.Clear();
// projít všechny dostupné sériové porty
foreach (string currentPort in System.IO.Ports.SerialPort.GetPortNames())
{
// přidání jednotlivých sériových portů do listboxu
        listBoxSerialPort.Items.Add(currentPort);
}
ShowSerialPorts.Enabled = false;
buttonClr.Enabled = true;
```

3. Vytvoříme kód pro událost buttonClr_Click

```
listBoxSerialPort.Items.Clear();
listBoxSerialPort.Enabled = false;
labelSelected.Text = "";
ShowSerialPorts.Enabled = true;
buttonClr.Enabled = false;
```

4. Vytvoříme událost ShowSelected

```
labelSelected.Text = "Selected port – " + listBoxSerialPort.SelectedItem.ToString();
```

5. Vytvoříme událost exitToolStripMenuItem_Click

```
Application.Exit();
```

1.5 Testování aplikace

Nyní projekt spustíme pomocí [Ctrl+F5] a otestujeme, zda funguje správně.

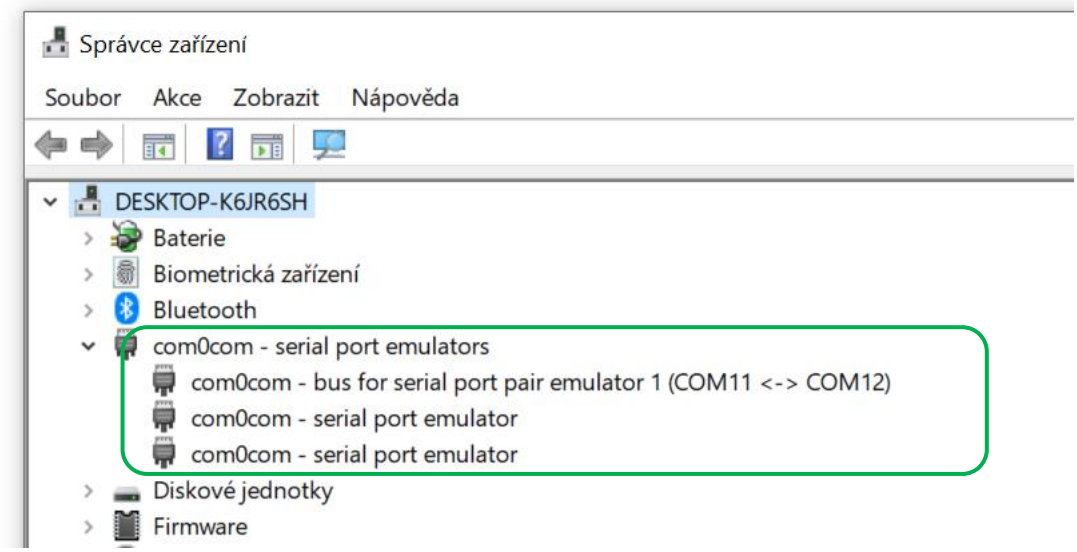
2 Virtuální sériové porty

Pokud provádíme vývoj aplikací, které komunikují s hardwarovými prostředky pomocí sériové linky, je vhodné při vývoji aplikace využít nástroje pro vytvoření tzv. virtuálního spojení sériových potů. Nespornou výhodou použití tohoto virtuálního spojení zejména v tom, že při tvorbě aplikace osobního počítače lze eliminovat úvahy o funkčnosti aplikace a lze tento vývoj provádět dříve, než je realizovaný potřebný hardware s mikropočítačem.

3 Vytvoření virtuálního spojení

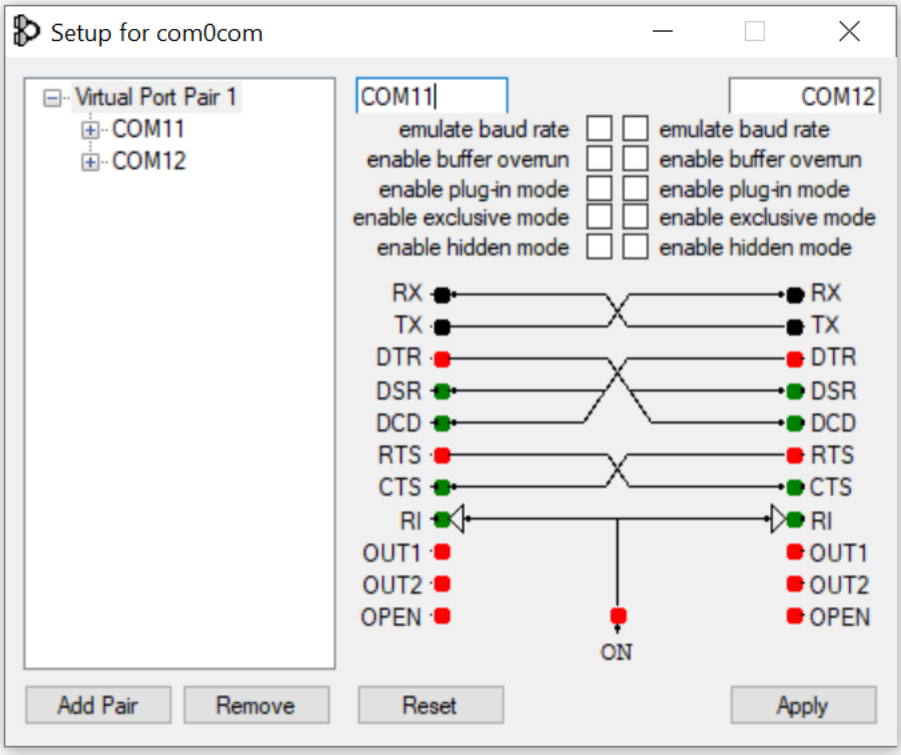
K vytvoření spojení dvou virtuálních sériových portů použijeme aplikaci "Null-modem emulator" com0com. Pokud používáme Windows 10 64bit je potřeba mít podepsaný instalační soubor pro 64bitovou verzi, který lze stáhnout na tomto odkazu.

Po instalaci aplikace automaticky vznikne virtuální pár "CNCA0" a "CNCB0" a také jeden spojený sériový pár v tomto případě "COM11" a "COM12", který lze poté použít ke komunikaci pomocí terminálu. Pokud instalace proběhla v pořádku, tak lze ve "Správci zařízení" vidět vytvořený virtuální sériový pár, viz obr. 5.



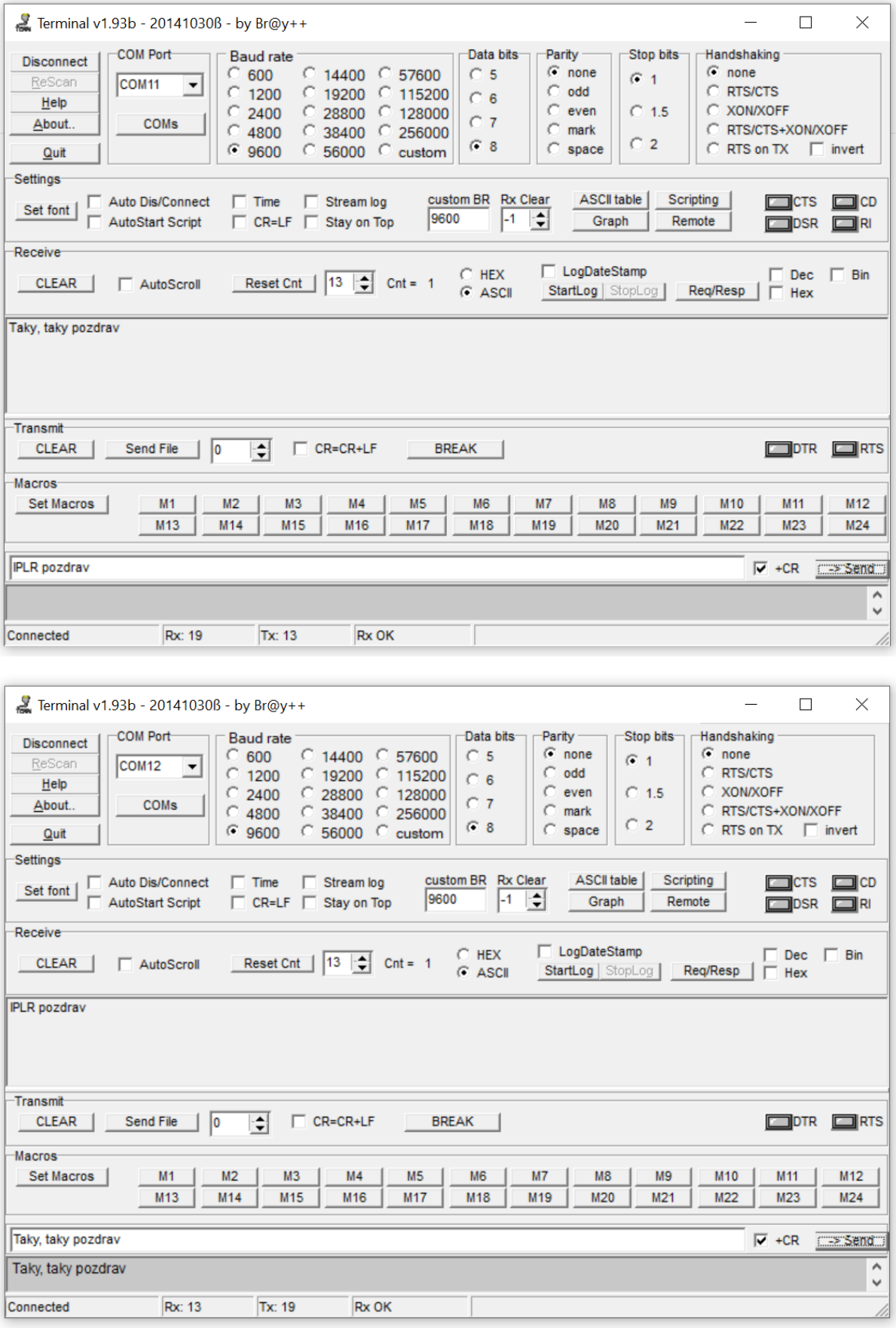
Obr. 5 – Zobrazení virtuální sériových párů ve Správci zařízení (printscreen) (Microsoft, 2022b)

Prostředí nastavení aplikace „com0com" lze vidět na obr. 6. V aplikaci lze vytvořit libovolný počet virtuálních sériových párů. Přidat nový pár lze pomocí tlačítka "Add Pair" a odebrat pár lze pomocí tlačítka "Remove".



Obr. 6 – Prostředí aplikace com0com (vytvořeno v com0com)

Lze i přejmenovat již vytvořené porty a potvrdit pomocí tlačítka „Apply“, Otestovat vytvořené spojení lze pomocí dvou terminálových programů (např. Terminal v1.93b – by Br@y++), Na jednom terminálu otevřeme sériový port “COM11” a na druhém port “COM12”. Následně můžeme poslat zprávu z jednoho terminálu do druhého, viz obr.7.



Obr. 7 – Test virtuálního propojení pomocí dvou terminálů (vytvořeno v Terminal v1.93b)

4 Klávesové zkratky pro psaní znaků stiskem kombinací kláves:

>	[Pravý ALT + >]	×	[Pravý ALT + ×]
<	[Pravý ALT + <]	€	[Pravý ALT + E]
|	[Pravý ALT + W]	~	[Pravý ALT + 1]
&	[Pravý ALT + C]	°	[SHIFT + ;]
[	[Pravý ALT + F]	ˆ	[Pravý ALT + 3]
]	[Pravý ALT + G]	\	[Pravý ALT + Q]
@	[Pravý ALT + V]	*˙	[Pravý ALT + -]
#	[Pravý ALT + V]	{	[Pravý ALT + B]
$	[Pravý ALT + ů]	}	[Pravý ALT + N]
÷	[Pravý ALT + ú]	‘	[SHIFT + ň]

Tip pro VS:

Pro formátování textu zdrojových kódů ve Visual Studiu můžeme použít funkci **-auto format** můžeme použít kombinaci kláves:

- pro celý soubor [CTRL + K + D]
- Pro výběr [CTRL + K + F]

Nebo z hlavní nabídky Upravit -> Upřesnit -> [Formátovat dokument], nebo [Formátovat výběr]

5 Použitá literatura

MICROSOFT Corp. a. s., 2022a. MS Visual Studio 2022, verze 17.8.9. 14. 5. 2024 [cit. 25. 5. 2024]. Dostupné na WWW: https://visualstudio.microsoft.com/cs/

MICROSOFT Corp. a. s., 2022b. MS Windows 10, verze 22H2 18. 10. 2022 [cit. 25. 5. 2024]. Dostupné na WWW: https://www.microsoft.com/cs-cz/software-download/windows10

6 Studijní literatura

VIRIUS, Miroslav, 2002. *C# pro zelenáče.* Praha: Neocortex. ISBN 80-72321-76-5.

SELLS, Chris, 2005. *C# a Winforms: programování formulářů ve Windows.* Brno: Zoner Press. ISBN: 80-86815-25-0.

NAGEL, Ch., EVJEN, B., GLYNN, J. a SKINNER, M. W., 2007. *C# 2005 – Programujeme profesionálně.* Brno: Computer Press. ISBN 80-251-1181-4.

7 Otázky k procvičení

1 Jak vytvoříme událost „click“?
2 Jak se vytvářejí obslužné rutiny událostí komponent?
3 Jak pracuje C# se sériovým portem?
4 Lze se připojit již s otevřeným sériovým portem?
5 Jak jsou označeny sériové porty?

Seznam zkratek

GUI grafické uživatelské rozhraní

VS Visual Studio

Rejstřík

Programování řídicích aplikací

Téma 11: Ovládání digitálních pinů mikropočítače v jazyce C#

Studijní cíl

Seznámit studenty s tvorbou Windows Forms aplikace pro ovládání digitálních pinů mikropočítače. Cílem je vytvořit GUI aplikaci pro ovládání digitálních pinů mikropočítače. Na pinech mikropočítače bude ovládána svítivá, LED, dioda způsobem zap/vyp.

Doba nutná k nastudování

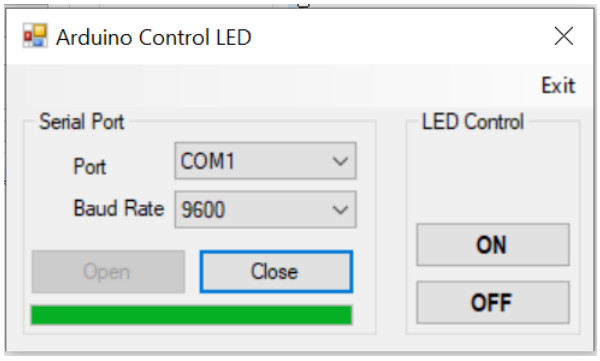
2 hodiny

Klíčová slova

Sériový port, zdrojový kód, kód, C#, algoritmus, událost, obsluha události, komponenty, ovládací prvky, Button, Label, Textové pole, GroupBox

1 Sériové porty počítače

Cílem je vytvořit **grafickou aplikaci**, která bude mít jeden formulář, dvě tlačítka, ovládací prvek ListBox, GroupBox, Label a menuStrip (pro ukončení aplikace Exit).

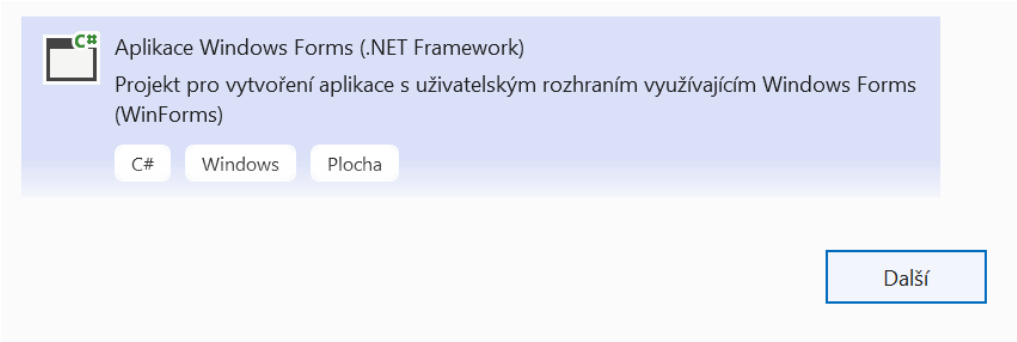


Obr. 1 – Vzhled aplikace (vytvořeno v Microsoft Visual Studio 2022)

1.1 Vytvoření nového projektu v jazyce C#

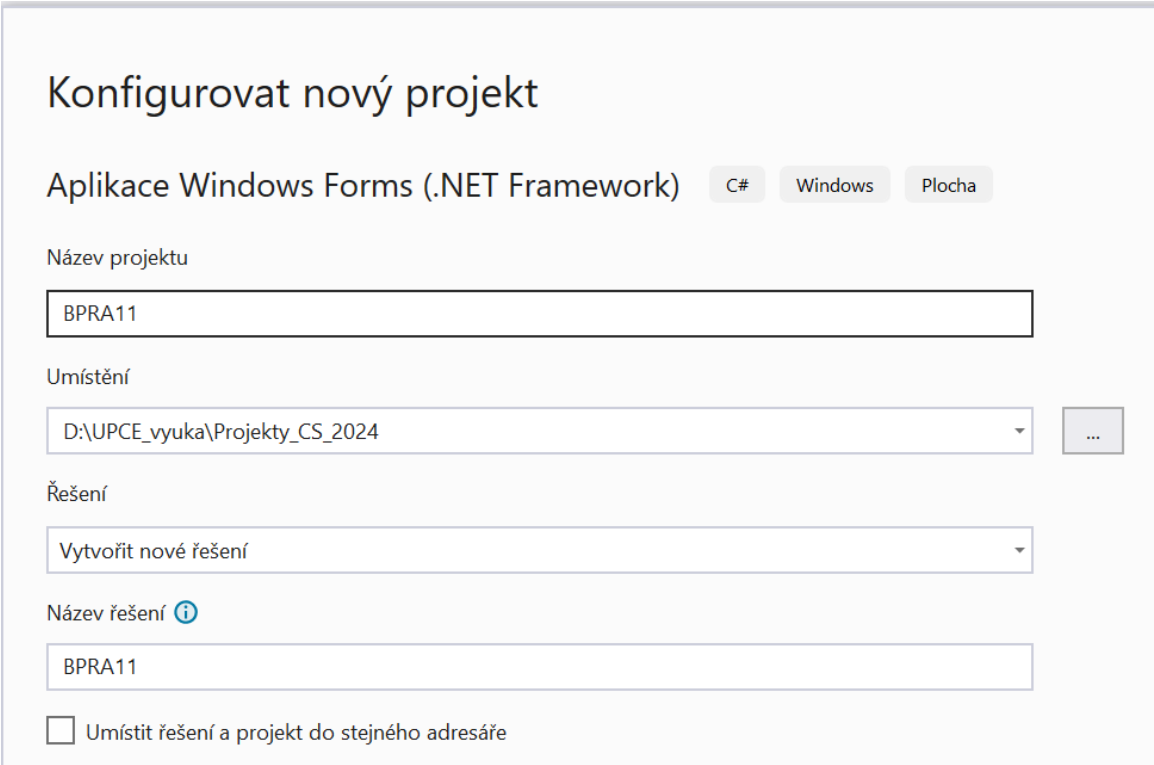
Vytvoříme projekt Windows Forms aplikaci .NET v C# s názvem "BPRA11":

Po spuštění průvodce ve Visual Studiu zvolíme Windows Forms aplikaci pro .NET



Obr. 2 – Založení nového projektu v Windows Forms (printscreen) (Microsoft, 2022)

Do dialogového okna vložíme název projektu a jeho umístění na disku.



Obr. 3 – Založení nového projektu v Windows Forms (printscreen) (Microsoft, 2022)

1.2 Přidání ovládacích prvků uživatelského rozhraní

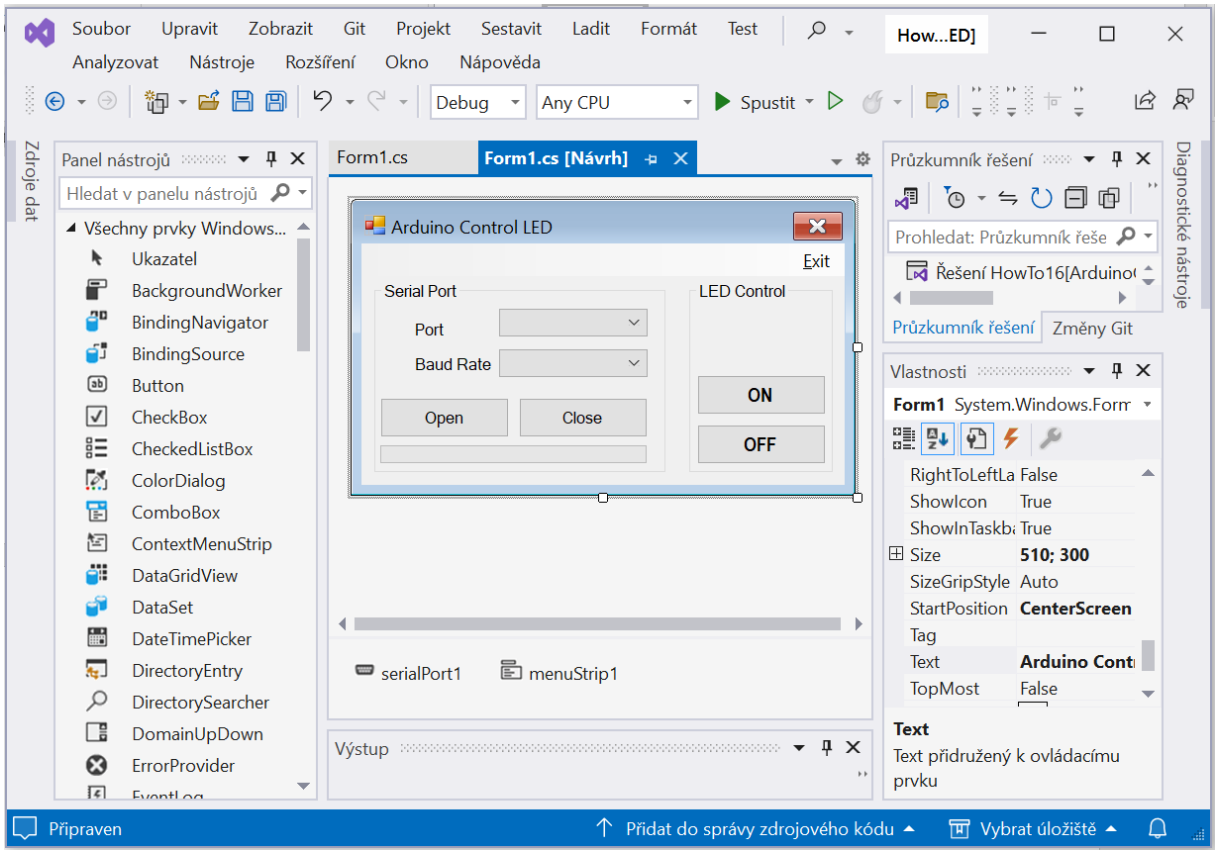
Použijeme následující komponenty UI (User Interface) a rozmístíme je na formuláři:

- **Label** s názvem label1 – pro text Port
- **Label** s názvem label2 – pro text Baud Rate
- **ListBox** s názvem ports_comBox – pro zobrazení dostupných portů
- **ListBox** s názvem bautRate_comBox – pro výběr přenosové rychlosti
- **Button** s názvem OpenPort_Btn – pro otevření vybraného sériového portu

- **Button** s názvem ClosePort_Btn – pro zavření otevřeného sériového portu
- **Button** s názvem LED_ON_Btn – pro rozsvícení ovládané LED
- **Button** s názvem LED_OFF_Btn – pro zhasnutí ovládané LED
- **GroupBox** s názvem groupBox1 – pro seskupení ovládacích prvků sériového portu
- **GroupBox** s názvem groupBox1 – pro seskupení ovládacích tlačítek LED
- **ProgressBar** s názvem progressBar1 – pro indikaci připojení sériového portu
- **MenuStrip** s názvem menuStrip1 – pro ovládací menu Exit
- **SerialPort** s názvem serialPort1 – ovládací prvek sériového portu

Po vložení ovládacích prvků je rozmístěte na formuláři tak, jak je vyobrazeno na obr. 4.

Vzhled grafického editoru uživatelského rozhraní může vypadat podobně jako tento:



Obr. 4 – Rozložení komponent v aplikaci (printscreen) (Microsoft, 2022)

Vytvořeno s využitím (Kruprapas, 2024).

1.3 Konfigurace ovládacích prvků uživatelského rozhraní

Nyní je potřeba nastavit vlastnosti vložených ovládacích prvků. Pro ovládací prvky uživatelského rozhraní použijeme následující nastavení:

- Pro **hlavní formulář** (), který obsahuje všechny ovládací prvky: Form

- (name) = Form1
- Size = 510 ; 300
- Text = „Arduino Control LED"
- Font.Size = 8
- MaximizeBox = False
- MinimizeBox = True
- StartPosition = CenterScreen
- FormBorderStyle = Fixed3D

- **Pro zobrazení dostupných portů () : Label**

  - (name) = label1
  - AutoSize = False
  - Size = 38 ; 20
  - TextAlign = MiddleLeft
  - Font.Name = „Arial Narrow“
  - Font.Size = 10
  - Text = „Port“

- **Pro zobrazení volitelných přenosových rychlostí () : Label**

  - (name) = label2
  - AutoSize = False
  - Size = 38 ; 74
  - TextAlign = MiddleLeft
  - Font.Name = „Arial Narrow“
  - Font.Size = 10
  - Text = „Baud Rate“

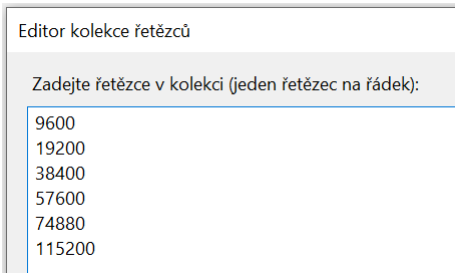
- **Pro zobrazení výběru COM portu () : ComboBox**

  - (name) = ports_comBox
  - DropDownStyle = DropDownList
  - Size = 150 ; 28
  - TextAlign = MiddleLeft
  - Font.Name = „Arial Narrow“
  - Font.Size = 10
  - Text = „Baud Rate“

- **Pro zobrazení výběru přenosových rychlostí () : ComboBox**

  - (name) = baudRate_comBox
  - DropDownStyle = DropDownList
  - Size = 150 ; 28
  - TextAlign = MiddleLeft
  - Font.Name = „Arial Narrow“
  - Font.Size = 10
  - Text = „Open“

- Items (Kolekce) vložte následující řadu přenosových rychlostí (viz obr. 5)



Obr. 5 – Přenosové rychlosti sériového portu (printscreen) (Microsoft, 2022)

- **Tlačítko pro otevření COM portu () : Button**
  - (name) = OpenPort_Btn
  - Size = 130 ; 40
  - Text = "&Open"
- **Tlačítko pro zavření COM portu () : Button**
  - (name) = ClosePort_Btn
  - Size = 130 ; 40
  - Text = "&Close"
- **Tlačítko pro rozsvícení LED () : Button**
  - (name) = LED_ON_Btn
  - Size = 130 ; 40
  - Text = "&ON"
- **Tlačítko pro zhasnutí LED () : Button**
  - (name) = LED_OFF_Btn
  - Size = 130 ; 40
  - Text = "O&FF"
- **MenuStrip (z ovládacích prvků viz. Obr. 4) pro Exit () : menuStrip**
  - (name) = menuStrip1
  - Položka "&Exit"
  - Alignment = Right
- **SerialPort (z ovládacích prvků viz. Obr. 4) () : serialPort1**
  - (name) = serialPort1
- **GroupBox pro seskupení komponent pro výběr portu () : GroupBox**

- (name) = groupBox1
- Text = "Serial Port"

- **GroupBox pro seskupení komponent pro ovládání LED () : GroupBox**

  - (name) = groupBox2
  - Text = "LED Control"

- **ProgressBar pro indikaci připojení portu () : ProgressBar**

  - (name) = progressBar1
  - Size = 269 ; 18

Vytvořeno s využitím (Kruprapas, 2024).

Poznámka:
Po vložení MenuStrip a SerialPort na formulář, tyto komponenty na formuláři nezůstanou zobrazeny, ale budou umístěny se do lišty pod návrhem formuláře.

1.4 Události a obslužné rutiny událostí

K vytvoření **událostí** použijeme ikonu událostí v okně [**Vlastnosti**] v aplikaci Visual Studio, viz. obrázek č. 4.

Nyní vytvoříme následující události:

- Form1_Load (dvojklikem myší na formulář)
- Form1_FormClosing (klikem myší na události pro Form1)
- ports_comBox_MouseClick (klikem myší na události pro ports_comBox)
- OpenPort_Btn_Click (dvojklikem myši na ovládací prvek Button)
- ClosePort_Btn_Click (dvojklikem myši na ovládací prvek Button)
- LED_ON_Btn_Click (dvojklikem myši na ovládací prvek Button)
- LED_OFF_Btn_Click (dvojklikem myši na ovládací prvek Button)
- exitToolStripMenuItem_Click (dvojklikem myši na položku Exit na StripMenu)

Událost **Form.Load** a **FormClosing** se provede při spuštění (zavření) programu, před zobrazením okna aplikace. Události ports_comBox_MouseClick(), OpenPort_Btn_Click(), ClosePort_Btn_Click(), LED_ON_Btn_Click(), LED_OFF_Btn_Click() a exitToolStripMenuItem_Click (), se vykonají při stisknutí příslušné klávesy, respektive na položku „Exit“ v menuStrip.

Nyní je nutné vytvořit programový kód pro obsluhu jednotlivých událostí:

1. Vytvoříme kód pro událost Form1_Load

```
LED_ON_Btn.Enabled = false;
LED_OFF_Btn.Enabled = false;
```

```
ClosePort_Btn.Enabled = false;
baudRate_comBox.SelectedIndex = 0;
string[] ports = SerialPort.GetPortNames();
ports_comBox.Items.AddRange(ports);
ports_comBox.SelectedIndex = 0;
```

2. Vytvoříme kód pro událost ports_comBox_MouseClick

```
ports_comBox.Items.Clear();
string[] ports = SerialPort.GetPortNames();
ports_comBox.Items.AddRange(ports);
ports_comBox.SelectedIndex = 0;
```

3. Vytvoříme kód pro událost OpenPort_Btn_Click

```
LED_ON_Btn.Enabled = true;
LED_OFF_Btn.Enabled = true;
ClosePort_Btn.Enabled = true;
OpenPort_Btn.Enabled = false;
if (serialPort1.IsOpen)
   serialPort1.Close();
serialPort1.PortName = ports_comBox.Text;
serialPort1.BaudRate = Convert.ToInt32(baudrate_comBox.Text);
progressBar1.Value = 100;
try
{
   serialPort1.Open();
}
catch (Exception ex)
{
   MessageBox.Show(ex.Message, "Message", MessageBoxButtons.OK,
MessageBoxIcon.Error);
}
```

4. Vytvoříme událost ClosePort_Btn_Click

```
LED_ON_Btn.Enabled = false;
LED_OFF_Btn.Enabled = false;
ClosePort_Btn.Enabled = false;
OpenPort_Btn.Enabled = true;
if (serialPort1.IsOpen)
   serialPort1.Close();
progressBar1.Value = 0;
```

5. Vytvoříme událost LED_ON_Btn_Click

```
try
{
   serialPort1.Write("a");
}
catch (Exception ex)
{
   MessageBox.Show(ex.Message);
}
```

6. Vytvoříme událost LED_OFF_Btn_Click

```
try
{
   serialPort1.Write("A");
}
catch (Exception ex)
{
   MessageBox.Show(ex.Message);
}
```

7. Vytvoříme událost Form1_FormClosing

```
if (serialPort1.IsOpen)
   serialPort1.Close();
Application.Exit();
```

8. Vytvoříme událost exitToolStripMenuItem_Click

```
Application.Exit();
```

2 Kód pro IDE mikropočítače „Arduino“

```
#define LED 13 // ovládání vestavěné diody na Arduinu
char Receive;

void setup() {
  pinMode(13, OUTPUT);
  Serial.begin(9600);
}
```

```
void loop() {
 Receive = Serial.read();
  switch (Receive) {
    case 'a':
      digitalWrite(LED, HIGH);
      break;
    case 'A':
      digitalWrite(LED, LOW);
      break;
  }
}
```

Vytvořeno s využitím (Kruprapas, 2024).

2.1 Testování aplikace

Nyní projekt spustíme pomocí [Ctrl+F5] a otestujeme, zda funguje správně.

3 Klávesové zkratky pro psaní znaků stiskem kombinací kláves:

>	[Pravý ALT + >]	×	[Pravý ALT + ×]
<	[Pravý ALT + <]	€	[Pravý ALT + E]
|	[Pravý ALT + W]	~	[Pravý ALT + 1]
&	[Pravý ALT + C]	°	[SHIFT + ;]
[	[Pravý ALT + F]	ˆ	[Pravý ALT + 3]
]	[Pravý ALT + G]	\	[Pravý ALT + Q]
@	[Pravý ALT + V]	*˙	[Pravý ALT + -]
#	[Pravý ALT + V]	{	[Pravý ALT + B]
$	[Pravý ALT + ů]	}	[Pravý ALT + N]
÷	[Pravý ALT + ú]	‘	[SHIFT + ň]

Tip pro VS:

Pro formátování textu zdrojových kódů ve Visual Studiu můžeme použít funkci **-auto format** můžeme použít kombinaci kláves:

- pro celý soubor [CTRL + K + D]
- Pro výběr [CTRL + K + F]

Nebo z hlavní nabídky Upravit -> Upřesnit -> [Formátovat dokument], nebo [Formátovat výběr]

4 Použité zdroje

KRUPRAPAS, S., 2024. *Lekce č. 16, ovládání LED* [online]. [cit. 26.5.2024]. Dostupný na WWW: https://www.praphas.com/forum/index.php?topic=375.0

MICROSOFT Corp. a. s., 2022. MS Visual Studio 2022, verze 17.8.9. 14. 5. 2024 [cit. 25. 5. 2024]. Dostupné na WWW: https://visualstudio.microsoft.com/cs/

5 Studijní literatura

VIRIUS, Miroslav, 2002. *C# pro zelenáče.* Praha: Neocortex. ISBN 80-72321-76-5.

SELLS, Chris, 2005. *C# a Winforms: programování formulářů ve Windows.* Brno: Zoner Press. ISBN: 80-86815-25-0.

NAGEL, Ch., EVJEN, B., GLYNN, J. a SKINNER, M. W., 2007. *C# 2005 – Programujeme profesionálně.* Brno: Computer Press. ISBN 80-251-1181-4.

6 Otázky k procvičení

1 Jak vytvoříme událost „click“?
2 Jak se zavírá přístup k sériovému portu?
3 Jak odesílá C# textový řetězec sériovým portem?
4 Lze se připojit již s otevřeným sériovým portem?
5 Jak jsou označeny sériové porty?

Seznam zkratek

GUI grafické uživatelské rozhraní

VS Visual Studio

Rejstřík

Programování řídicích aplikací

Téma 12: Monitorování stavu digitálních pinů mikropočítače v jazyce C#

Studijní cíl

Seznámit studenty s tvorbou Windows Forms aplikace pro zobrazení stavů digitálních pinů mikropočítače. Cílem je vytvořit GUI aplikaci pro zobrazení aktuálních stavů digitálních pinů mikropočítače. Na pinech mikropočítače budou ovládány stiskem tlačítek svítivé, LED, diody a zároveň bude stav pinů připojených tlačítek odesílán zpět, do formulářové aplikace.

Doba nutná k nastudování

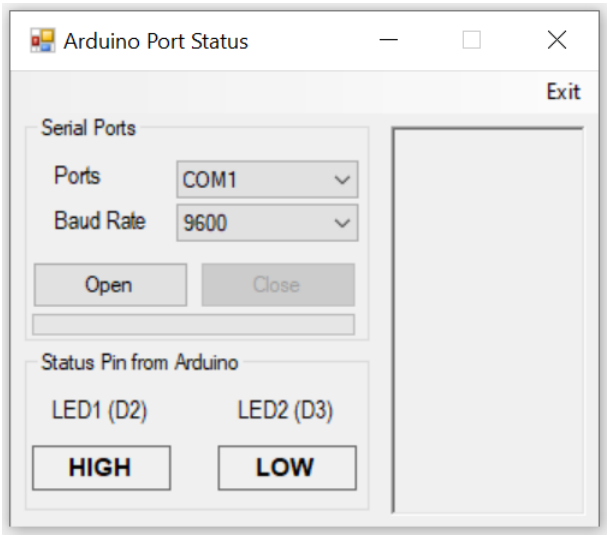
2 hodiny

Klíčová slova

Sériový port, zdrojový kód, kód, C#, algoritmus, událost, obsluha události, komponenty, ovládací prvky, Button, Label, Textové pole, GroupBox, ComboBox

1 Aplikace pro monitorování pinů mikropočítače

Cílem je vytvořit **grafickou aplikaci**, která bude mít jeden formulář, tlačítka, ovládací prvky ComboBox, GroupBox, Label a menuStrip (pro ukončení aplikace Exit).

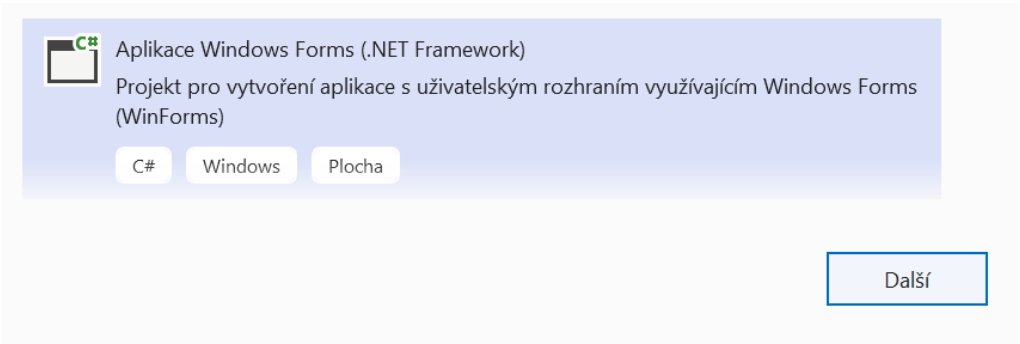


Obr. 1 – Vzhled aplikace (vytvořeno v Microsoft Visual Studio 2022)

1.1 Vytvoření nového projektu v jazyce C#

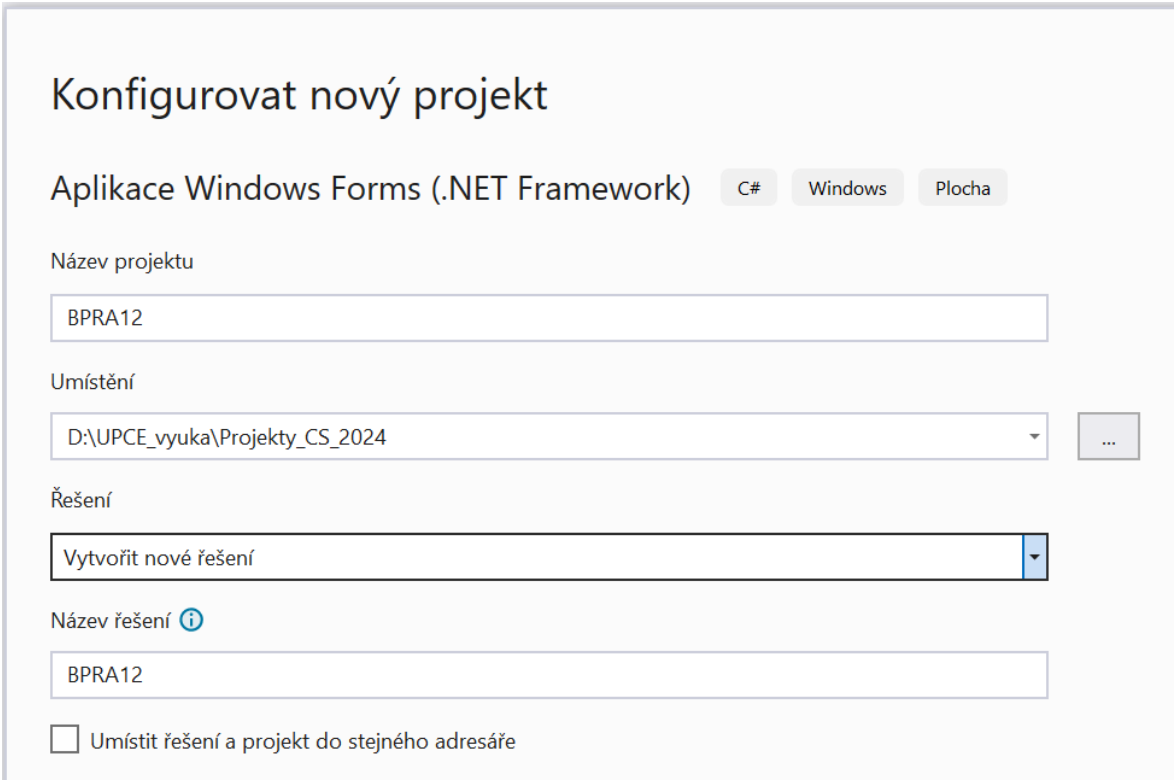
Vytvoříme projekt Windows Forms aplikaci .NET v C# s názvem "BPRA12":

Po spuštění průvodce ve Visual Studiu zvolíme Windows Forms aplikaci pro .NET



Obr. 2 – Založení nového projektu v Windows Forms (vytvořeno v Microsoft Visual Studio 2022)

Do dialogového okna vložíme název projektu a jeho umístění na disku.



Obr. 3 – Založení nového projektu v Windows Forms (vytvořeno v Microsoft Visual Studio 2022)

1.2 Přidání ovládacích prvků uživatelského rozhraní

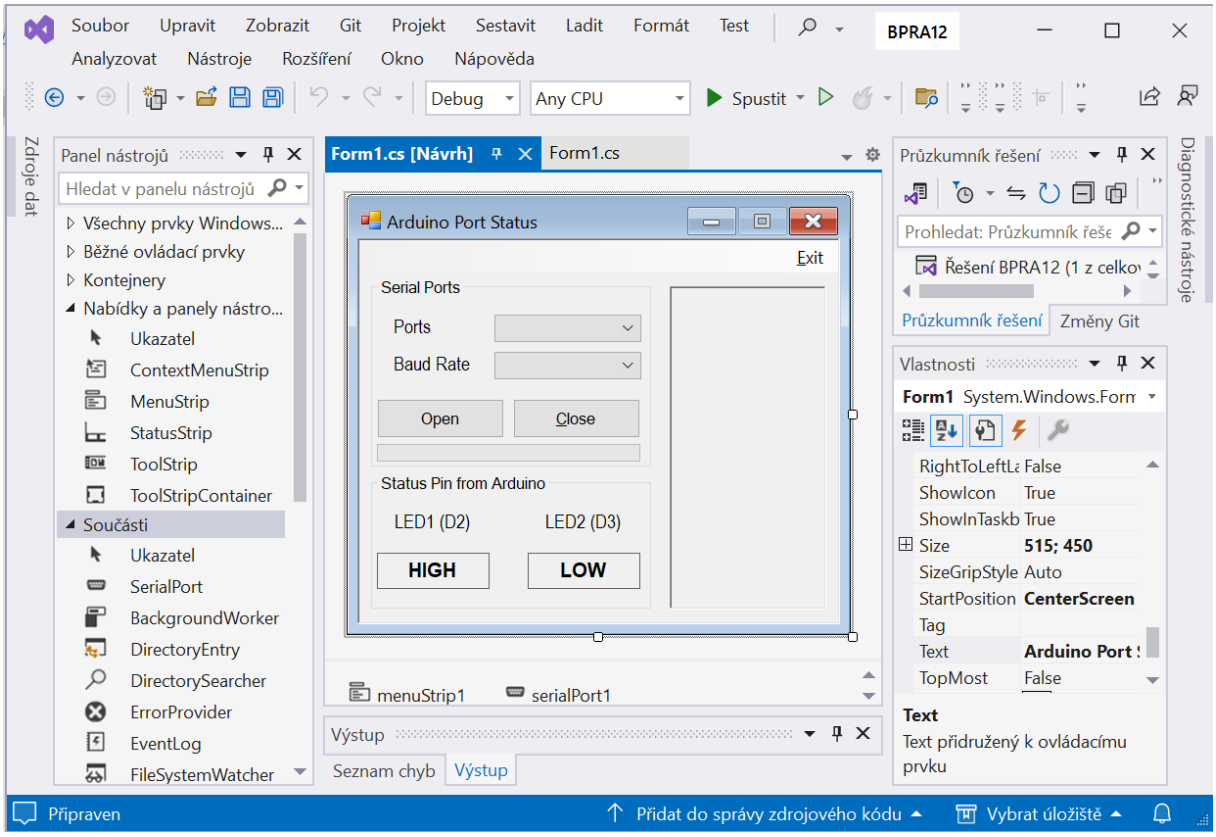
Použijeme následující komponenty UI (User Interface) a rozmístíme je na formuláři:

- **Label** s názvem label1 – pro text Port
- **Label** s názvem label2 – pro text Baud Rate

- **Label** s názvem label3 – zobrazení stavu pinu mikropočítače
- **Label** s názvem label4 – zobrazení stavu pinu mikropočítače
- **ComboBox** s názvem ports_comBox – pro zobrazení dostupných portů
- **ComboBox** s názvem bautRate_comBox – pro výběr přenosové rychlosti
- **Button** s názvem OpenPort_Btn – pro otevření vybraného sériového portu
- **Button** s názvem ClosePort_Btn – pro zavření otevřeného sériového portu
- **GroupBox** s názvem groupBox1 – pro seskupení ovládacích prvků sériového portu
- **GroupBox** s názvem groupBox1 – pro seskupení ovládacích komponent Label
- **ProgressBar** s názvem progressBar1 – pro indikaci připojení sériového portu
- **RichTextBox** s názvem richTextBox – pro zobrazení přijatých dat z mikropočítače
- **MenuStrip** s názvem menuStrip1 – pro ovládací menu Exit
- **SerialPort** s názvem serialPort1 – ovládací prvek sériového portu

Po vložení ovládacích prvků je rozmístěte na formuláři tak, jak je vyobrazeno na obr. 4.

Vzhled grafického editoru uživatelského rozhraní může vypadat podobně jako tento:



Obr. 4 – Rozložení komponent v aplikaci (vytvořeno v Microsoft Visual Studio 2022)

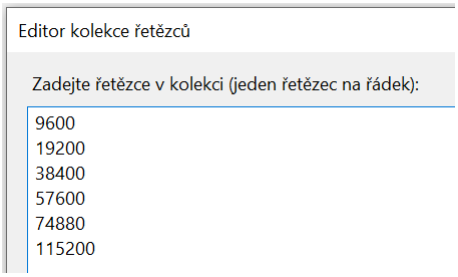
Vytvořeno s využitím (Kruprapas, 2024).

1.3 Konfigurace ovládacích prvků uživatelského rozhraní

Nyní je potřeba nastavit vlastnosti vložených ovládacích prvků. Pro ovládací prvky uživatelského rozhraní použijeme následující nastavení:

- Pro **hlavní formulář** (), který obsahuje všechny ovládací prvky: Form
    - (name) = Form1
    - Size = 510 ; 300
    - Text = „Arduino Control LED"
    - Font.Size = 8
    - MaximizeBox = False
    - MinimizeBox = True
    - StartPosition = CenterScreen
    - FormBorderStyle = Fixed3D
- **Pro zobrazení dostupných portů () : Label**
    - (name) = label1
    - AutoSize = False
    - Size = 38 ; 20
    - TextAlign = MiddleLeft
    - Font.Name = „Arial Narrow“
    - Font.Size = 10
    - Text = „Port“
- **Pro zobrazení volitelných přenosových rychlostí () : Label**
    - (name) = label2
    - AutoSize = False
    - Size = 38 ; 74
    - TextAlign = MiddleLeft
    - Font.Name = „Arial Narrow“
    - Font.Size = 10
    - Text = „Baud Rate“
- **Pro zobrazení výběru COM portu () : ComboBox**
    - (name) = ports_comBox
    - DropDownStyle = DropDownList
    - Size = 150 ; 28
    - TextAlign = MiddleLeft
    - Font.Name = „Arial Narrow“
    - Font.Size = 10
    - Text = „Baud Rate“
- **Pro zobrazení výběru přenosových rychlostí () : ComboBox**

- (name) = baudRate_comBox
- DropDownStyle = DropDownList
- Size = 150 ; 28
- TextAlign = MiddleLeft
- Font.Name = „Arial Narrow“
- Font.Size = 10
- Text = „Open“
- Items (Kolekce) vložte následující řadu přenosových rychlostí (viz. obr. č. 5)



Obr. 5 – Přenosové rychlosti sériového portu (vytvořeno v Microsoft Visual Studio 2022)

- **Tlačítko pro otevření COM portu () : Button**
  - (name) = OpenPort_Btn
  - Size = 130 ; 40
  - Text = "&Open"
- **Tlačítko pro zavření COM portu () : Button**
  - (name) = ClosePort_Btn
  - Size = 130 ; 40
  - Text = "&Close"
- **Pro zobrazení monitorovaného pinu mikropočítače () : Label**
  - (name) = label3
  - AutoSize = False
  - BorderStyle = None
  - Size = 115; 37
  - TextAlign = MiddleLeft
  - Font.Name = „Arial Narrow“
  - Font.Size = 10
  - Text = „LED2 (D2)“
- **Pro zobrazení monitorovaného pinu mikropočítače () : Label**
  - (name) = label5
  - AutoSize = False
  - BorderStyle = None

- Size = 115; 37
- TextAlign = MiddleLeft
- Font.Name = „Arial Narrow“
- Font.Size = 10
- Text = „LED2 (D3)“

- **Pro zobrazení stavu pinu mikropočítače () : Label**

  - (name) = label4
  - AutoSize = False
  - BorderStyle = FixedSingle
  - Size = 115; 37
  - TextAlign = MiddleLeft
  - Font.Name = „Arial Narrow“
  - Font.Size = 10
  - Text = „HIGH“

- **Pro zobrazení stavu pinu mikropočítače () : Label**

  - (name) = label6
  - AutoSize = False
  - BorderStyle = FixedSingle
  - Size = 115; 37
  - TextAlign = MiddleLeft
  - Font.Name = „Arial Narrow“
  - Font.Size = 10
  - Text = „LOW“

- **MenuStrip (z ovládacích prvků viz obr. 4) pro Exit (): menuStrip1**

  - (name) = menuStrip1
  - Položka "&Exit"
  - Alignment = Right

- **RiichTextBox pro zobrazení přijatých dat (): richTextBox1**

  - (name) = richTextBox1
  - BorderStyle = Fixed3D
  - Size = 160; 330
  - TextAlign = MiddleLeft
  - Font.Name = „Arial Narrow“
  - Font.Size = 10
  - ReadOnly = true

- **SerialPort (z ovládacích prvků viz obr. 4) (): serialPort1**

  - (name) = serialPort1

- **GroupBox pro seskupení komponent pro výběr portu () : GroupBox**
    - (name) = groupBox1
    - Text = "Serial Port"
- **GroupBox pro seskupení komponent pro ovládání LED () : GroupBox**
    - (name) = groupBox2
    - Text = "Status Pin from Arduino"
- **ProgressBar pro indikaci připojení portu () : ProgressBar**
    - (name) = progressBar1
    - Size = 269 ; 18

Poznámka:
Po vložení MenuStrip a SerialPort na formulář, tyto komponenty na formuláři nezůstanou zobrazeny, ale budou umístěny se do lišty pod návrhem formuláře.

1.4 Události a obslužné rutiny událostí

K vytvoření **událostí** použijeme ikonu událostí v okně [**Vlastnosti**] v aplikaci Visual Studio, viz obr. 4.

Nyní vytvoříme následující události:

- Form1_Load (dvojklikem myší na formulář)
- Form1_FormClosing (klikem myší na události pro Form1)
- ports_comBox_MouseClick (klikem myší na události pro ports_comBox)
- OpenPort_Btn_Click (dvojklikem myši na ovládací prvek Button)
- ClosePort_Btn_Click (dvojklikem myši na ovládací prvek Button)
- serialPort1_DataReceived (jako metodu v serialPort1)
- exitToolStripMenuItem_Click (dvojklikem myši na položku Exit na StripMenu)

Událost **Form.Load** a **FormClosing** se provede při spuštění (zavření) programu, před zobrazením okna aplikace. Události ports_comBox_MouseClick(), OpenPort_Btn_Click(), ClosePort_Btn_Click() a exitToolStripMenuItem_Click (), se vykonají při stisknutí příslušné klávesy, respektive na položku „Exit“ v menuStrip.

Nyní je nutné vytvořit programový kód pro obsluhu jednotlivých událostí:

1. Vytvoříme kód pro událost Form1_Load

```
ClosePort_Btn.Enabled = false;
baudRate_comBox.SelectedIndex = 0;
string[] ports = SerialPort.GetPortNames();
```

```
ports_comBox.Items.AddRange(ports);
ports_comBox.SelectedIndex = 0;
```

2. Do třídy class Form1 : Form vytvoříme proměnnou pro přijatá data

```
private string DispString;   //proměnná pro přijatá data
```

3. Vytvoříme kód pro událost ports_comBox_MouseClick

```
ports_comBox.Items.Clear();
string[] ports = SerialPort.GetPortNames();
ports_comBox.Items.AddRange(ports);
ports_comBox.SelectedIndex = 0;
```

4. Vytvoříme kód pro událost OpenPort_Btn_Click

```
ClosePort_Btn.Enabled = true;
OpenPort_Btn.Enabled = false;
if (serialPort1.IsOpen)
   serialPort1.Close();
serialPort1.PortName = ports_comBox.Text;
serialPort1.BaudRate = Convert.ToInt32(baudrate_comBox.Text);
progressBar1.Value = 100;
try
{
   serialPort1.Open();
}
catch (Exception ex)
{
   MessageBox.Show(ex.Message, "Message", MessageBoxButtons.OK,
MessageBoxIcon.Error);
}
```

5. Vytvoříme událost ClosePort_Btn_Click

```
ClosePort_Btn.Enabled = false;
OpenPort_Btn.Enabled = true;
if (serialPort1.IsOpen)
   serialPort1.Close();
progressBar1.Value = 0;
```

6. Vytvoříme událost serialPort1_DataReceived (jako metodu v serialPort1)

```
DispString = serialPort1.ReadExisting();
this.Invoke(new EventHandler(DisplayText));
```

7. Vytvoříme metodu DisplayText ()

```
private void DisplayText(object sender, EventArgs e)
{
   richTextBox1.AppendText(DispString);
   string[] value = DispString.Split(new string[] { ";" }, StringSplitOptions.None);

   if ((Convert.ToInt16(value[0])) == 1)
   {
      label4.Text = "HIGH";
   }
   else
   {
      label4.Text = "LOW";
   }

   if ((Convert.ToInt16(value[1])) == 1)
   {
      label6.Text = "HIGH";
   }
   else
   {
      label6.Text = "LOW";
   }
}
```

8. Vytvoříme událost Form1_FormClosing

```
if (serialPort1.IsOpen)
   serialPort1.Close();
Application.Exit();
```

9. Vytvoříme událost exitToolStripMenuItem_Click

```
Application.Exit();
```

Vytvořeno s využitím (Kruprapas, 2024).

2 Kód pro IDE mikropočítače „Arduino“

```
#define LED1  2

#define LED2  3

#define SW1   10

#define SW2   11
```

```
String newData="", oldData="";
void setup() {
  pinMode(LED1, OUTPUT);
  pinMode(LED2, OUTPUT);
  pinMode(SW1, INPUT_PULLUP);
  pinMode(SW2, INPUT_PULLUP);
  Serial.begin(9600);
}
void loop() {
  newData = "";
  if (!digitalRead(SW1)) {
    digitalWrite(LED1, HIGH);
    newData = "1,";
  }
  else {
    digitalWrite(LED1, LOW);
    newData = "0,";
  }
  if (!digitalRead(SW2)) {
    digitalWrite(LED2, HIGH);
    newData = newData + "1,";
  }
  else {
    digitalWrite(LED2, LOW);
    newData = newData + "0";
  }
  if (newData!=oldData){
  Serial.println(newData);
  }
  oldData=newData;
  delay(100);
```

}

(Kruprapas, 2024)

2.1 Testování aplikace

Nyní projekt spustíme pomocí [Ctrl+F5] a otestujeme, zda funguje správně.

3 Klávesové zkratky pro psaní znaků stiskem kombinací kláves:

>	[Pravý ALT + >]	×	[Pravý ALT + ×]
<	[Pravý ALT + <]	€	[Pravý ALT + E]
|	[Pravý ALT + W]	~	[Pravý ALT + 1]
&	[Pravý ALT + C]	°	[SHIFT + ;]
[	[Pravý ALT + F]	ˆ	[Pravý ALT + 3]
]	[Pravý ALT + G]	\	[Pravý ALT + Q]
@	[Pravý ALT + V]	*˙	[Pravý ALT + -]
#	[Pravý ALT + V]	{	[Pravý ALT + B]
$	[Pravý ALT + ů]	}	[Pravý ALT + N]
÷	[Pravý ALT + ú]	‘	[SHIFT + ň]

Tip pro VS:

Pro formátování textu zdrojových kódů ve Visual Studiu můžeme použít funkci **-auto format** můžeme použít kombinaci kláves:

- pro celý soubor [CTRL + K + D]
- Pro výběr [CTRL + K + F]

Nebo z hlavní nabídky Upravit -> Upřesnit -> [Formátovat dokument], nebo [Formátovat výběr]

4 Použité zdroje

KRUPRAPAS, S. *Lekce č. 21* [online]. [cit. 26.5.2024]. Dostupný na WWW: https://www.praphas.com/forum/index.php?topic=381.0

5 Studijní literatura:

VIRIUS, Miroslav. *C# pro zelenáče.* Neocortex. ISBN 80-72321-76-5.

SELLS, Chris. *C# a Winforms: programování formulářů ve Windows.* Brno: Zoner Press 2005, ISBN: 80-86815-25-0.

NAGEL, Ch., EVJEN, B., GLYNN, J., SKINNER, M. W. *C# 2005 – Programujeme profesionálně.* Brno: Computer Press, 2007. ISBN 80-251-1181-4.

6 Otázky k procvičení

1 Jak vytvoříme událost „serialPort1_DataReceived“?
2 Jak ošetříme výjimku pro pokus otevření neexistujícího sériového portu?
3 Jak odesílá C# textový řetězec sériovým portem?
4 Jak zapojíme PULLUP rezistor k pin portu ATmega328?
5 Jak korektně ukončíme formulářovou aplikaci v C#?

Seznam zkratek

GUI	grafické uživatelské rozhraní
UI	User Interface
VS	Visual Studio Code

Rejstřík

Programování řídicích aplikací

Téma 13: Senzor teploty a vlhkosti v jazyce C#

Studijní cíl

Seznámit studenty s tvorbou Windows Forms aplikace pro zobrazení údajů ze senzoru technologických veličin, teploty a vlhkosti DTH22. Hodnoty technologických veličin budou zobrazeny včetně příslušných znaků jednotek teploty ve „°C“ a vlhkosti v „%“.

Doba nutná k nastudování

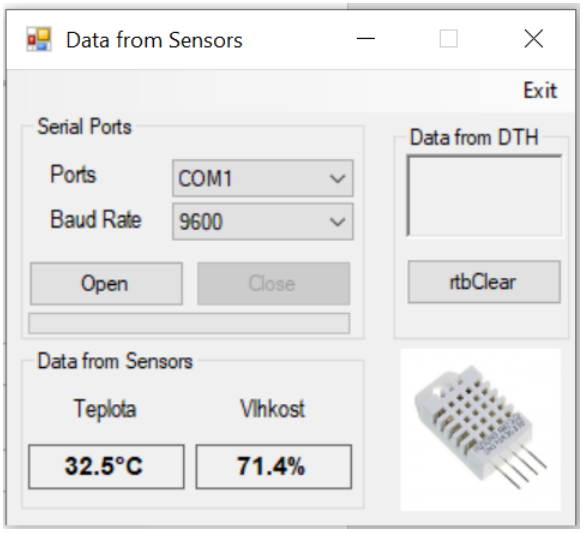
2 hodiny

Klíčová slova

Sériový port, zdrojový kód, kód, C#, algoritmus, událost, obsluha události, komponenty, ovládací prvky, Button, Label, richTextBox, GroupBox, ComboBox, vlhkost, teplota

1 Aplikace pro zpracování dat ze senzoru teploty a vlhkosti

Cílem je vytvořit **grafickou aplikaci**, která bude mít jeden formulář, tlačítka, ovládací prvky ComboBox, GroupBox, Label a menuStrip (pro ukončení aplikace Exit).

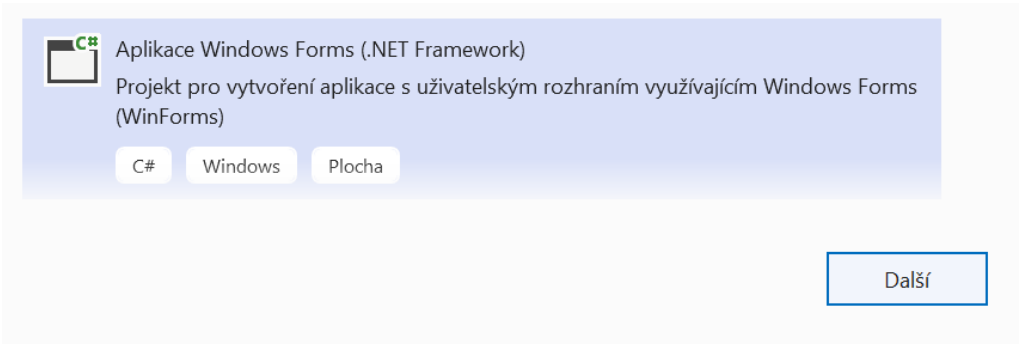


Obr. 1 – Vzhled aplikace (vytvořeno v Microsoft Visual Studio 2022)

1.1 Vytvoření nového projektu v jazyce C#

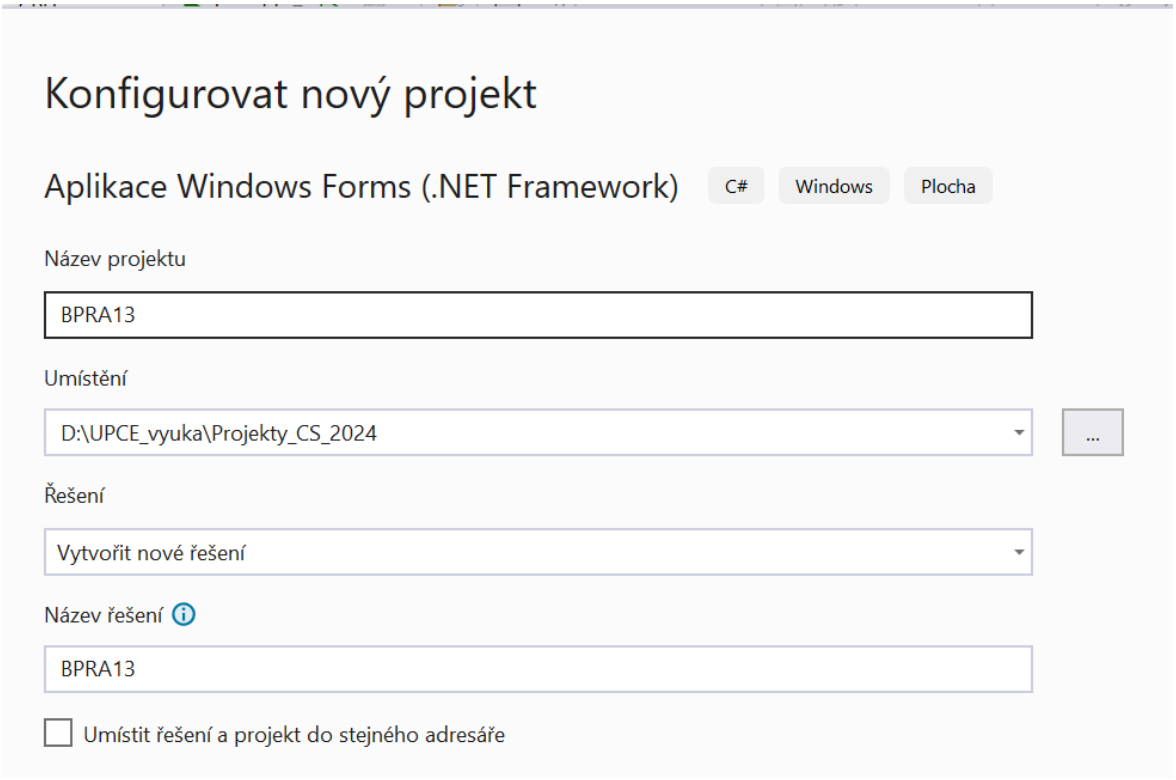
Vytvoříme projekt Windows Forms aplikaci .NET v C# s názvem "BPRA13":

Po spuštění průvodce ve Visual Studiu zvolíme Windows Forms aplikaci pro .NET



Obr. 2 – Založení nového projektu v Windows Forms (printscreen) (Microsoft, 2022)

Do dialogového okna vložíme název projektu a jeho umístění na disku.



Obr. 3 – Založení nového projektu v Windows Forms (printscreen) (Microsoft, 2022)

1.2 Přidání ovládacích prvků uživatelského rozhraní

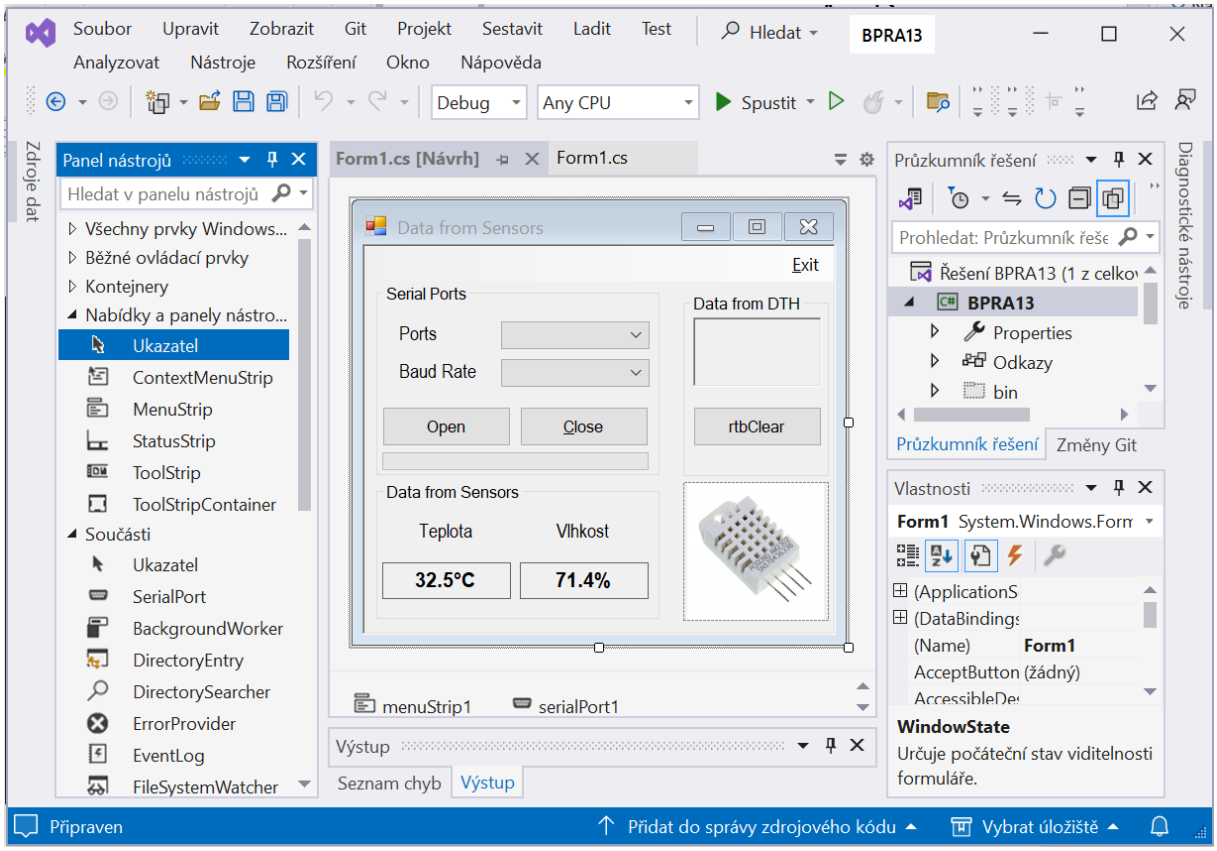
Použijeme následující komponenty UI (User Interface) a rozmístíme je na formuláři:

- **Label** s názvem label1 – pro text Port
- **Label** s názvem label2 – pro text Baud Rate

- **Label** s názvem label3 – zobrazení stavu pinu mikropočítače
- **Label** s názvem label4 – zobrazení stavu pinu mikropočítače
- **ComboBox** s názvem ports_comBox – pro zobrazení dostupných portů
- **ComboBox** s názvem bautRate_comBox – pro výběr přenosové rychlosti
- **Button** s názvem OpenPort_Btn – pro otevření vybraného sériového portu
- **Button** s názvem ClosePort_Btn – pro zavření otevřeného sériového portu
- **Button** s názvem rtbClear_Btn – pro vymazání obsahu RichTextBoxu
- **GroupBox** s názvem groupBox1 – pro seskupení ovládacích prvků sériového portu
- **GroupBox** s názvem groupBox1 – pro seskupení ovládacích komponent Label
- **ProgressBar** s názvem progressBar1 – pro indikaci připojení sériového portu
- **RichTextBox** s názvem richTextBox – pro zobrazení přijatých dat z mikropočítače
- **PictureBox** s názvem pictureBox1 – pro umístění obrázku senzoru na formulář
- **MenuStrip** s názvem menuStrip1 – pro ovládací menu Exit
- **SerialPort** s názvem serialPort1 – ovládací prvek sériového portu

Po vložení ovládacích prvků je rozmístěte na formuláři tak, jak je vyobrazeno na obr. 4.

Vzhled grafického editoru uživatelského rozhraní může vypadat podobně jako tento:



Obr. 4 – Rozložení komponent v aplikaci (printscreen) (Microsoft, 2022)

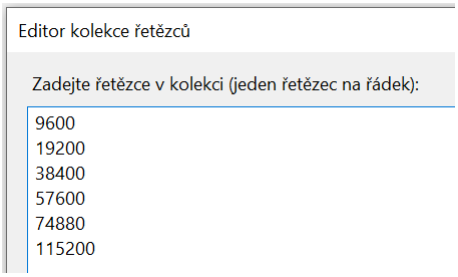
Inspirováno (Kruprapas, 2024).

1.3 Konfigurace ovládacích prvků uživatelského rozhraní

Nyní je potřeba nastavit vlastnosti vložených ovládacích prvků. Pro ovládací prvky uživatelského rozhraní použijeme následující nastavení:

- Pro **hlavní formulář** (), který obsahuje všechny ovládací prvky: Form
    - (name) = Form1
    - Size = 500 ; 450
    - Text = „Arduino Control LED"
    - Font.Size = 8
    - MaximizeBox = False
    - MinimizeBox = True
    - StartPosition = CenterScreen
    - FormBorderStyle = Fixed3D
- **Pro zobrazení dostupných portů () : Label**
    - (name) = label1
    - AutoSize = False
    - Size = 38 ; 20
    - TextAlign = MiddleLeft
    - Font.Name = „Arial Narrow“
    - Font.Size = 10
    - Text = „Port“
- **Pro zobrazení volitelných přenosových rychlostí () : Label**
    - (name) = label2
    - AutoSize = False
    - Size = 38 ; 74
    - TextAlign = MiddleLeft
    - Font.Name = „Arial Narrow“
    - Font.Size = 10
    - Text = „Baud Rate“
- **Pro zobrazení výběru COM portu () : ComboBox**
    - (name) = ports_comBox
    - DropDownStyle = DropDownList
    - Size = 150 ; 28
    - TextAlign = MiddleLeft
    - Font.Name = „Arial Narrow“
    - Font.Size = 10
    - Text = „Baud Rate“
- **Pro zobrazení výběru přenosových rychlostí () : ComboBox**

- (name) = baudRate_comBox
- DropDownStyle = DropDownList
- Size = 150 ; 28
- TextAlign = MiddleLeft
- Font.Name = „Arial Narrow“
- Font.Size = 10
- Text = „Open“
- Items (Kolekce) vložte následující řadu přenosových rychlostí (viz. obr. č. 5)



Obr. 5 – Přenosové rychlosti sériového portu (printscreen) (Microsoft, 2022)

- **Tlačítko pro otevření COM portu () : Button**
  - (name) = OpenPort_Btn
  - Size = 130 ; 40
  - Text = "&Open"
- **Tlačítko pro zavření COM portu () : Button**
  - (name) = ClosePort_Btn
  - Size = 130 ; 40
  - Text = "&Close"
- **Pro zobrazení monitorovaného pinu mikropočítače () : Label**
  - (name) = label3
  - AutoSize = False
  - BorderStyle = None
  - Size = 115; 37
  - TextAlign = MiddleLeft
  - Font.Name = „Arial Narrow“
  - Font.Size = 10
  - Text = „Teplota“
- **Pro zobrazení monitorovaného pinu mikropočítače () : Label**
  - (name) = label5
  - AutoSize = False
  - BorderStyle = None

- Size = 115; 37
- TextAlign = MiddleLeft
- Font.Name = „Arial Narrow“
- Font.Size = 10
- Text = „Vlhkost“

- **Pro zobrazení teploty () : Label**

  - (name) = label4
  - AutoSize = False
  - BorderStyle = FixedSingle
  - Size = 115; 37
  - TextAlign = MiddleLeft
  - Font.Name = „Arial Narrow“
  - Font.Size = 10
  - Text = „32.3**°C**“

- **Pro zobrazení vlhkosti () : Label**

  - (name) = label6
  - AutoSize = False
  - BorderStyle = FixedSingle
  - Size = 115; 37
  - TextAlign = MiddleLeft
  - Font.Name = „Arial Narrow“
  - Font.Size = 10
  - Text = „66.6%“

- **MenuStrip (z ovládacích prvků viz obr. 4) pro Exit (): menuStrip1**

  - (name) = menuStrip1
  - Položka "&Exit"
  - Alignment = Right

- **RichTextBox pro zobrazení přijatých dat (): richTextBox1**

  - (name) = richTextBox1
  - BorderStyle = Fixed3D
  - Size = 130; 70
  - TextAlign = MiddleLeft
  - Font.Name = „Arial Narrow“
  - Font.Size = 10
  - ReadOnly = true

- **SerialPort (z ovládacích prvků viz obr. 4) (): serialPort1**

  - (name) = serialPort1

- **GroupBox pro seskupení komponent pro výběr portu () : GroupBox**
    - (name) = groupBox1
    - Text = "Serial Port"
- **GroupBox pro seskupení komponent pro zobrazení dat () : GroupBox**
    - (name) = groupBox2
    - Text = "Data from Sensors"
- **GroupBox pro seskupení komponent pro ovládání rtbTextBoxu () : GroupBox**
    - (name) = groupBox3
    - Text = "Data from DTH"
- **pictureBox1 pro zobrazení obrázku senzoru () : pictureBox1**
    - (name) = (name) = groupBox2
    - Size = 147; 140
    - SizeMode = "Zoom"
- **ProgressBar pro indikaci připojení portu () : ProgressBar**
    - (name) = progressBar1
    - Size = 269 ; 18

Poznámka:
Po vložení MenuStrip a SerialPort na formulář, tyto komponenty na formuláři nezůstanou zobrazeny, ale budou umístěny se do lišty pod návrhem formuláře.

1.4 Události a obslužné rutiny událostí

K vytvoření **událostí** použijeme ikonu událostí v okně [**Vlastnosti**] v aplikaci Visual Studio, viz obr. 4.

Nyní vytvoříme následující události:

- Form1_Load (dvojklikem myší na formulář)
- Form1_FormClosing (klikem myší na události pro Form1)
- ports_comBox_MouseClick (klikem myší na události pro ports_comBox)
- OpenPort_Btn_Click (dvojklikem myši na ovládací prvek Button)
- ClosePort_Btn_Click (dvojklikem myši na ovládací prvek Button)
- serialPort1_DataReceived (jako metodu v serialPort1)
- exitToolStripMenuItem_Click (dvojklikem myši na položku Exit na StripMenu)

Událost **Form.Load** a **FormClosing** se provede při spuštění (zavření) programu, před zobrazením okna aplikace. Události ports_comBox_MouseClick(), OpenPort_Btn_Click(), ClosePort_Btn_Click() a exitToolStripMenuItem_Click (), se vykonají při stisknutí příslušné klávesy, respektive na položku „Exit“ v menuStrip.

Nyní je nutné vytvořit programový kód pro obsluhu jednotlivých událostí:

1. Vytvoříme kód pro událost Form1_Load

```
ClosePort_Btn.Enabled = false;
baudRate_comBox.SelectedIndex = 0;
string[] ports = SerialPort.GetPortNames();
ports_comBox.Items.AddRange(ports);
ports_comBox.SelectedIndex = 0;
```

2. Do třídy class Form1 : Form vytvoříme proměnnou pro přijatá data

```
private string DispString;   //proměnná pro přijatá data
```

3. Vytvoříme kód pro událost ports_comBox_MouseClick

```
ports_comBox.Items.Clear();
string[] ports = SerialPort.GetPortNames();
ports_comBox.Items.AddRange(ports);
ports_comBox.SelectedIndex = 0;
```

4. Vytvoříme kód pro událost OpenPort_Btn_Click

```
ClosePort_Btn.Enabled = true;
OpenPort_Btn.Enabled = false;
if (serialPort1.IsOpen)
   serialPort1.Close();
serialPort1.PortName = ports_comBox.Text;
serialPort1.BaudRate = Convert.ToInt32(baudrate_comBox.Text);
progressBar1.Value = 100;
try
{
   serialPort1.Open();
}
catch (Exception ex)
{
   MessageBox.Show(ex.Message, "Message", MessageBoxButtons.OK,
MessageBoxIcon.Error);
}
```

5. Vytvoříme událost ClosePort_Btn_Click

```
ClosePort_Btn.Enabled = false;
OpenPort_Btn.Enabled = true;
if (serialPort1.IsOpen)
   serialPort1.Close();
progressBar1.Value = 0;
```

6. Vytvoříme událost serialPort1_DataReceived (jako metodu v serialPort1)

```
DispString = serialPort1.ReadExisting();
this.Invoke(new EventHandler(DisplayText));
```

7. Vytvoříme metodu DisplayText ()

```
private void DisplayText(object sender, EventArgs e)
{
        richTextBox1.AppendText(DispString);
        string[] value = DispString.Split(new string[] { "," }, StringSplitOptions.None);
   try
   {
      label4.Text = value[0] + "°C";
      label6.Text = value[1] + "%";
   }
   catch (Exception)
   { }
}
```

8. Vytvoříme událost Form1_FormClosing

```
if (serialPort1.IsOpen)
   serialPort1.Close();
Application.Exit();
```

9. Vytvoříme událost exitToolStripMenuItem_Click

```
Application.Exit();
```

Inspirováno (Kruprapas, 2024).

2 Kód pro IDE mikropočítače „Arduino“

```
#include <DHT.h>

#define DHTPIN 8      // pin to connect DHT22
```

```
#define DHTTYPE DHT22  // Type of use DHT11,DHT21,DHT22

DHT dht(DHTPIN, DHTTYPE);
void setup() {
  Serial.begin(9600);
  Serial.println("DHT test!");
  dht.begin();
}
void loop() {
  delay(2000);
  float h = dht.readHumidity();
  float t = dht.readTemperature();
  if (isnan(h) || isnan(t))
  {
    Serial.println("Failed to read sensor!");
    return;
  }
  Serial.print(t,1);
  Serial.print(",");
  Serial.print(h,1);
  Serial.println(",");
}
```

(Kruprapas, 2024).

2.1 Testování aplikace

Nyní projekt spustíme pomocí [Ctrl+F5] a otestujeme, zda funguje správně.

3 Klávesové zkratky pro psaní znaků stiskem kombinací kláves

>	[Pravý ALT + >]	|	[Pravý ALT + W]
<	[Pravý ALT + <]	&	[Pravý ALT + C]

[	[Pravý ALT + F]	~	[Pravý ALT + 1]
]	[Pravý ALT + G]	°	[SHIFT + ;]
@	[Pravý ALT + V]	^	[Pravý ALT + 3]
#	[Pravý ALT + V]	\	[Pravý ALT + Q]
$	[Pravý ALT + ů]	*˙	[Pravý ALT + -]
÷	[Pravý ALT + ú]	{	[Pravý ALT + B]
×	[Pravý ALT + ×]	}	[Pravý ALT + N]
€	[Pravý ALT + E]	‘	[SHIFT + ň]

Tip pro VS:

Pro formátování textu zdrojových kódů ve Visual Studiu můžeme použít funkci **-auto format** můžeme použít kombinaci kláves:

- pro celý soubor [CTRL + K + D]
- Pro výběr [CTRL + K + F]

Nebo z hlavní nabídky Upravit -> Upřesnit -> [Formátovat dokument], nebo [Formátovat výběr]

4 Použité zdroje

KRUPRAPAS, S., 2024. *Lekce č. 19* [online]. [cit. 26.5.2024]. Dostupný na WWW: https://www.praphas.com/forum/index.php?topic=379.0

MICROSOFT Corp. a. s., 2022. MS Visual Studio 2022, verze 17.8.9. 14. 5. 2024 [cit. 25. 5. 2024]. Dostupné na WWW: https://visualstudio.microsoft.com/cs/

5 Studijní literatura

VIRIUS, Miroslav, 2002. *C# pro zelenáče.* Praha: Neocortex. ISBN 80-72321-76-5.

SELLS, Chris, 2005. *C# a Winforms: programování formulářů ve Windows.* Brno: Zoner Press. ISBN: 80-86815-25-0.

NAGEL, Ch., EVJEN, B., GLYNN, J. a SKINNER, M. W., 2007. *C# 2005 – Programujeme profesionálně.* Brno: Computer Press. ISBN 80-251-1181-4.

6 Otázky k procvičení

1 Jak vytvoříme událost „DisplayText“?
2 Jak vytvoříme nabídku Exit v komponentě MenuStrip?
3 Jakým způsobem přijímáme data ze sériového portu do formulářové aplikace?
4 Jak lze obsloužit výjimku při příjmu nesprávného formátu dat?
5 Jak lze zakázat minimalizaci a maximalizaci formuláře Form?

Seznam zkratek

GUI grafické uživatelské rozhraní

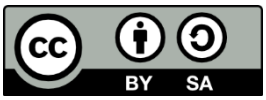
VS Visual Studio

UI User Interface

Rejstřík

UNIVERZITA PARDUBICE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Vytvořeno v rámci projektu **Studijní program Automatizace (SPAUT) na Univerzitě Pardubice**, reg. č. NPO_UPCE_MSMT-16591/2022.



Financováno Evropskou unií
NextGenerationEU

Národní plán obnovy

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY