Microsoft

Ján Hanák

Základy paralelného programovania v jazyku C# 3.0

Microsoft® Visual Studio® 2008

Ján Hanák

Základy paralelného programovania v jazyku C# 3.0

Artax
2009

Autor: Ing. Ján Hanák, MVP

Základy paralelného programovania v jazyku C# 3.0

Recenzenti:	doc. RNDr. Jozef Fecenko, CSc. Ing. Ľudovít Markus, CSc.
Vydanie:	prvé
Rok prvého vydania:	2009
Náklad:	150 ks
Jazyková korektúra:	Ing. Peter Kubica
Vydal:	Artax a.s., Žabovřeská 16, 616 00 Brno pre Microsoft s.r.o., Vyskočilova 1461/2a, 140 00 Praha 4
Tlač:	Artax a.s., Žabovřeská 16, 616 00 Brno
ISBN:	978-80-87017-03-6

Učebný text *Základy paralelného programovania v jazyku C# 3.0* bol schválený v Edičnom pláne Ekonomickej univerzity v Bratislave na rok 2009 ako vysokoškolská učebnica pre študentov povinného predmetu *Automatizácia programovania* v študijnom programe *Hospodárska informatika*, 2. stupeň (inžinierske štúdium).

Obsah

Úvod

S príchodom viacjadrových procesorov sa pred vývojármi, programátormi a softvérovými expertmi otvoril nový svet možností vývoja moderného počítačového softvéru. Ako tvorcovia počítačových systémov sa už nemôžeme spoliehať na neustále zvyšovanie sa taktovacej frekvencie procesorov, ktoré počas predchádzajúcich rokov zabezpečovalo implicitný nárast výkonnosti našich softvérových produktov. Perspektíva je zreteľná, pretože počet exekučných jadier na procesoroch sa bude aj v budúcnosti rýchlo množiť. Ak chceme využiť synergický efekt, ktorý generuje súčasná „viacjadrová hardvérová revolúcia“, musíme sa naučiť, ako zhotovovať paralelné programy. Teda programy, ktoré sú schopné súbežne vykonávať viacero úloh, pričom dokážu flexibilne využívať všetku disponibilnú výpočtovú kapacitu hardvérovej platformy.

Cieľom predkladanej vysokoškolskej učebnice je poskytnúť základný teoreticko-praktický výučbový kurz paralelného programovania v jazyku C# 3.0 na vývojovo-exekučnej platforme Microsoft .NET Framework 3.5. Paralelné programovanie je jednou z paradigiem, ktorých zvládnutie je nutné na to, aby sme mohli úspešne čeliť výzvam budúcich technologických progresov. Hoci bola táto vysokoškolská učebnica primárne projektovaná pre potreby študentov 2. stupňa vysokoškolského (inžinierskeho) štúdia odboru *Hospodárska informatika* na *Fakulte hospodárskej informatiky Ekonomickej univerzity v Bratislave*, sme presvedčení o tom, že rovnako dobre poslúži aj študentom na iných vysokých školách informatického zamerania. Prirodzene, poznatky uvádzané v tejto publikácii s výhodou využijú i komerční vývojári, programátori a softvéroví experti, ktorých poslaním je analyzovať, navrhovať a implementovať paralelné počítačové aplikácie.

Napriek tomu, že počítačové vedy poznajú mnoho praktických modelových prístupov k paralelnému programovaniu, rozhodli sme sa koncentrovať na paralelné programovanie v jednom z najmodernejších programovacích jazykov, C# 3.0 od spoločnosti Microsoft. Naša voľba bola podmienená viacerými faktormi.

Po prvé, poslucháči odboru *Hospodárska informatika* na *Fakulte hospodárskej informatiky Ekonomickej univerzity v Bratislave* absolvujú počas 1. stupňa svojho vysokoškolského (bakalárskeho) štúdia výučbové kurzy programovacích jazykov C a C++, v rámci ktorých sa zoznamujú so štruktúrovanou a objektovo orientovanou filozofiou vývoja počítačového softvéru. Keďže chceme, aby študenti rozvíjali svoje nadanie v triáde C → C++ → C#, je príklon k jazyku C# logickým vyústením našich snáh o priame prepojenie pedagogického procesu s potrebami praxe.

Po druhé, pri paralelnom programovaní v jazyku C# 3.0 sú študenti schopní maximalizovať svoju pracovnú produktivitu, pretože vývojovo-exekučná platforma Microsoft .NET Framework 3.5 obsahuje nielen virtuálny exekučný systém, ktorý riadi životné cykly vytvorených aplikácií, ale aj robustnú bázovú knižnicu tried, implementujúcu niekoľkotisícovú množinu dátových typov, ktoré sú charakteristické vysokou mierou abstrakcie.

Po tretie, domnievame sa, že znalosti objektovo orientovaného programovania (OOP) a paralelného objektovo orientovaného programovania (POOP) prispejú k maximalizácii pracovného uplatnenia študentov po skončení 2. stupňa vysokoškolského štúdia. Samozrejme, veľmi radi uvítame, ak sa študenti rozhodnú ďalej prehlbovať svoje informatické vedomosti aj v rámci 3. stupňa vysokoškolského (doktorandského) štúdia. Sme presvedčení o tom, že ak schopní mladí ľudia ovládnu sofistikované informačné technológie, ich potenciál je nekonečný.

Vysokoškolská učebnica *Základy paralelného programovania v jazyku C# 3.0* je podľa našich zistení jedinou slovenskou publikáciou, ktorá poskytuje elementárny výučbový kurz paralelného programovania pomocou najmodernejších technológií spoločnosti Microsoft.

Autor knihy by si rád splnil svoju milú povinnosť a srdečne poďakoval recenzentom, doc. RNDr. Jozefovi Fecenkovi, CSc. a Ing. Ľudovítovi Markusovi, CSc. za dôkladné preštudovanie diela a námety na jeho ďalšie skvalitnenie. Rovnako úprimne ďakuje autor Ing. Jiřímu Burianovi a Mgr. Miroslavovi Kubovčíkovi zo spoločnosti Microsoft za ich mnohoročnú aktívnu spoluprácu, ktorej výsledkom sú hodnotné produkty

permanentne zlepšujúce stav akademického a vývojárskeho ekosystému. V neposlednom rade vyjadruje autor hlbokú úctu a poďakovanie všetkým nadaným a aktívnym študentkám a študentom, ktorých usilovná práca a ľudský prístup pomáhajú zvyšovať kvalitu pedagogického procesu.

Ján Hanák

Bratislava, február 2009

Obsahová štruktúry knihy

Vysokoškolskú učebnicu tvorí dovedna 18 kapitol s nasledujúcim tematickým zameraním:

1. kapitola: Kapitola podáva výklad hlavných komponentov, z ktorých sa skladá vývojovo-exekučná platforma Microsoft .NET Framework 3.5. Charakterizovaný je virtuálny exekučný systém, bázová knižnica tried, spoločný typový systém a spoločná jazyková špecifikácia.

2. kapitola: Kapitola charakterizuje zostavenie riadenej aplikácie (aplikácie .NET) a ozrejmuje jej internú štruktúru.

3. kapitola: Kapitola sa sústreďuje na technickú analýzu riadenej exekúcie jednovláknovej riadenej aplikácie.

4. kapitola: Kapitola sa koncentruje na technickú analýzu riadenej exekúcie viacvláknovej riadenej aplikácie.

5. kapitola: Kapitola predstavuje základnú typológiu vlákien, pričom charakterizuje programové (aplikačné) vlákna, vlákna jadra operačného systému a hardvérové vlákna.

6. kapitola: Kapitola prezentuje technické podrobnosti hlavných typov procesorových architektúr súčasných počítačových systémov. Výklad sa venuje nasledujúcim typom procesorov: jednojadrové procesory architektúry Intel IA-32 bez a s technológiou HT, viacjadrové procesory architektúry Intel IA-32/Intel 64 bez a s technológiou HT a viacprocesorové architektúry Intel IA-32/Intel 64.

7. kapitola: Kapitola poukazuje na zmenu paradigmy z objektovo orientovaného programovania (OOP) na paralelné objektovo orientované programovanie (POOP).

8. kapitola: Kapitola sa zaoberá kvantifikáciou nárastu výkonnosti počítačových aplikácií pri použití paralelného objektovo orientovaného programovania.

9. kapitola: Kapitola charakterizuje Amdahlov zákon, pričom poukazuje na dôsledky plynúce z jeho praktického použitia.

10. kapitola: Kapitola zoznamuje čitateľov s Gustafsonovým zákonom a umožňuje zaujať iný pohľad na kvantifikáciu nárastu výkonnosti počítačových aplikácií pri paralelnom objektovo orientovanom programovaní.

11. kapitola: Kapitola definuje a vizuálne analyzuje lineárny, sublineárny a superlineárny nárast výkonnosti pri paralelnom objektovo orientovanom programovaní.

12. kapitola: Kapitola sa tematicky orientuje na praktické paralelné programovanie v jazyku C# 3.0. Čitatelia sa dozvedia, ako sa v tomto programovacom jazyku vytvárajú pracovné vlákna a ako môžeme s týmito vláknami manipulovať.

13. kapitola: Kapitola vyzdvihuje použitie fondu pracovných programových vlákien s algoritmizáciou praktického problému.

14. kapitola: Kapitola sa venuje problematike synchronizácie programových vlákien a vysvetleniu vybraných synchronizačných primitív.

15. kapitola: Kapitola ponúka výklad variantov paralelizmu (implicitný a explicitný paralelizmus, dátový paralelizmus, úlohový paralelizmus, paralelizmus dátových tokov, deklaratívny a imperatívny paralelizmus).

16. kapitola: Kapitolu tvorí praktické cvičenie, v ktorom je demonštrovaný proces paralelizácie pôvodne sekvenčnej riadenej aplikácie.

17. kapitola: Kapitola zoznamuje čitateľov s paralelnou platformou Microsoft Parallel Extensions.

18. kapitola: Kapitola ponúka teoreticko-praktický pohľad na natívne rozhranie OpenMP, ktoré slúži na podporu paralelizácie výpočtových procesov.

Typografické konvencie

Aby sme vám čítanie tejto knihy spríjemnili v čo možno najväčšej miere, bol prijatý kódex typografických konvencií, pomocou ktorých došlo k štandardizácii a unifikácii použitých textových štýlov a grafických symbolov. Veríme, že prijaté konvencie zvýšia prehľadnosť a používateľskú prívetivosť výkladu. Prehľad použitých typografických konvencií uvádzame v tab. A.

Typografická konvencia	Ukážka použitia typografickej konvencie
Štandardný text výkladu, ktorý neoznačuje zdrojový kód, identifikátory, modifikátory a kľúčové slová jazyka C# 3.0, ani názvy iných syntaktických elementov a entít, je formátovaný týmto typom písma.	Vývojovo-exekučná platforma Microsoft .NET Framework 3.5 kreuje spoločne s jazykom C# 3.0 jednotnú technologickú bázu na vytváranie moderných riadených aplikácií pre Windows, web a inteligentné mobilné zariadenia.

Tab. A: Prehľad použitých typografických konvencií

Typografická konvencia	Ukážka použitia typografickej konvencie
Názvy ponúk, položiek ponúk, ovládacích prvkov, komponentov, dialógových okien, podporných softvérových nástrojov, typov projektov ako aj názvy ďalších súčastí grafického používateľského rozhrania sú formátované **tučným** písmom. **Tučným** písmom sú rovnako formátované aj identifikátory programových entít, ktoré sú uvádzané priamo vo výkladovom texte.	Nový projekt štandardnej aplikácie pre systém Windows (**Windows Forms Application**) v prostredí produktu Visual C# 2008 založíme takto: 1. Otvoríme ponuku **File**, ukážeme na položku **New** a klikneme na príkaz **Project**. 2. V dialógovom okne **New Project** klikneme v stromovej štruktúre **Project Types** na položku **Visual C#**. 3. Zo súpravy projektových šablón (**Templates**) vyberieme ikonu šablóny **Windows Forms Application**. 4. Do textového poľa **Name** zapíšeme názov pre novú aplikáciu a stlačíme tlačidlo **OK**.

Fragmenty zdrojového kódu jazyka C# 3.0, prípadne akýchkoľvek iných programovacích jazykov sú formátované neproporcionálnym písmom `Courier New`.	`// Definícia premennej typu`
	`// string.`
	`string správa;`
	`// Inicializácia premennej.`
	`správa = "Vitajte v jazyku " +`
	`"C# 3.0!";`
	`// Zobrazenie okna so správou`
	`// pomocou metódy Show triedy`
	`// MessageBox.`
	`MessageBox.Show(správa);`

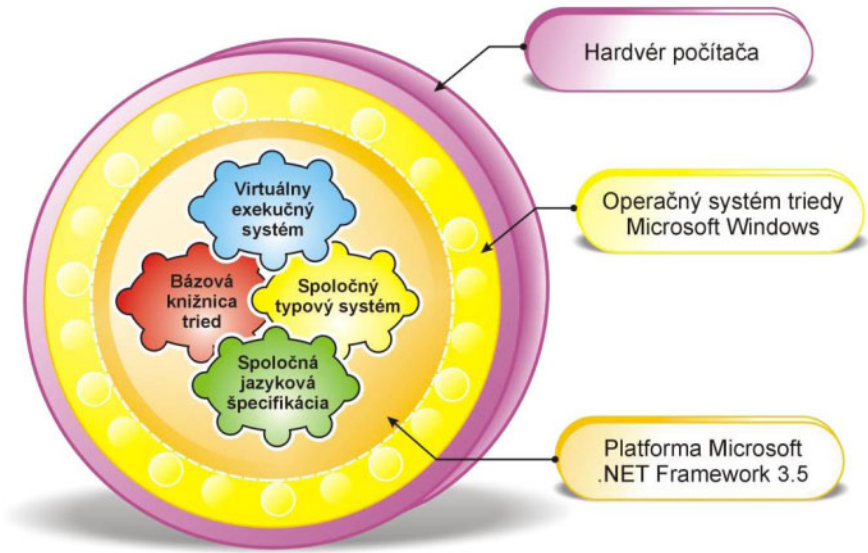
Tab. A: Prehľad použitých typografických konvencií (pokračovanie)

1 Architektonická štruktúra vývojovo-exekučnej platformy Microsoft .NET Framework 3.5

Vývojovo-exekučná platforma Microsoft .NET Framework 3.5 obsahuje kolekciu nasledujúcich základných (jadrových) komponentov:

1. Virtuálny exekučný systém (Virtual Execution System, VES).
2. Bázová knižnica tried (Base Class Library, BCL[1]).
3. Spoločný typový systém (Common Type System, CTS).
4. Spoločná jazyková špecifikácia (Common Language Specification, CLS).

Komponenty vývojovo-exekučnej platformy Microsoft .NET Framework 3.5 sú znázornené na obr. 1.1.



Obr. 1.1: Základné komponenty vývojovo-exekučnej platformy Microsoft .NET Framework 3.5

[1] Pre označenie bázovej knižnice tried sa niekedy používa aj označenie FCL, teda „Framework Class Library“.

V nasledujúcom texte ponúkneme stručnú charakteristiku spomenutých základných komponentov vývojovo-exekučnej platformy Microsoft .NET Framework 3.5.

1.1 Virtuálny exekučný systém (VES)

Virtuálny exekučný systém je softvérový stroj, ktorý realizuje riadenú exekúciu aplikácií .NET. Prijmime dohovor, že termíny „aplikácia .NET“ a „riadená aplikácia“ budú reprezentovať akýkoľvek typový počítačový program, ktorý vyhovuje týmto požiadavkám:

- Program bol vytvorený v niektorom z .NET-kompatibilných programovacích jazykov.
- Proces spustenia a vykonania programu na počítači je riadený virtuálnym exekučným systémom.

Keďže virtuálny exekučný systém vytvára prostredie pre beh ľubovoľnej riadenej aplikácie, je zrejmé, že medzi ním a riadenou aplikáciou existujú veľmi úzke väzby. Spojenie medzi uvedenými entitami je dokonca tak tesné, že bez virtuálneho exekučného systému nie je možné vykonanie aplikácie .NET na počítačovej stanici.

Zatiaľ čo termín „virtuálny exekučný systém“ je v oblasti počítačových vied všeobecným pomenovaním softvérového stroja, ktorý uskutočňuje riadenú exekúciu počítačového programu, spoločnosť Microsoft ako tvorca vývojovo-exekučnej platformy .NET Framework 3.5 označuje svoju vlastnú, teda konkrétnu implementáciu virtuálneho exekučného systému, ako „spoločné behové prostredie CLR[2]“. Hoci v bežných praktických podmienkach sa termíny „virtuálny exekučný systém“ a „spoločné behové prostredie“ vzájomne zamieňajú, považujeme za dôležité poukázať na skutočnosť, že „spoločné behové prostredie“ je len jednou z potenciálnych konkrétnych implementácií „virtuálneho exekučného systému“.

[2] Akronym CLR je skratkou viacslovného pomenovania „Common Language Runtime“.

V záujme zachovania čistoty výkladového textu budeme ďalej v tejto publikácii používať termín „virtuálny exekučný systém“.

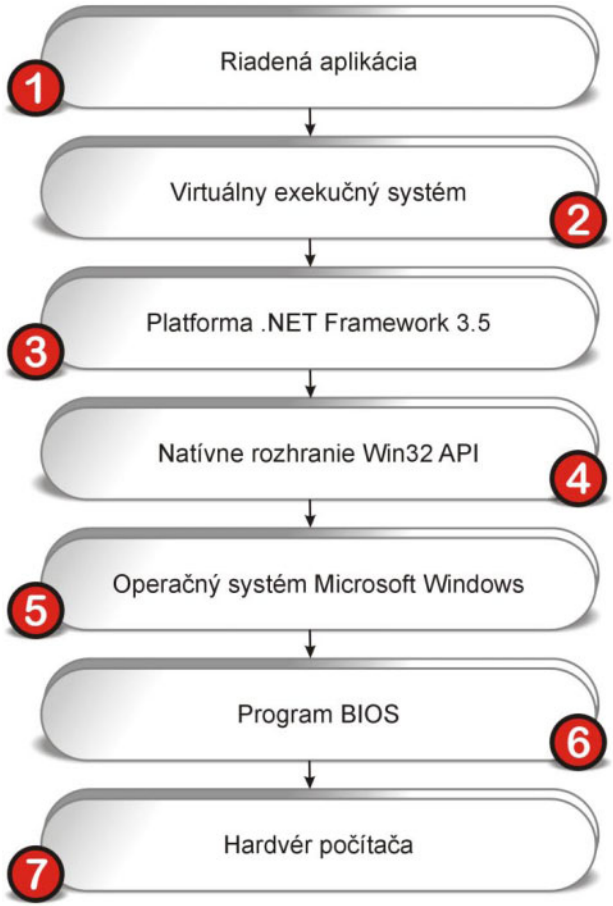
Virtuálny exekučný systém poskytuje riadeným aplikáciám súpravu nízkoúrovňových softvérových služieb, do ktorej patrí:

- Just-In-Time (JIT) kompilácia pseudostrojového kódu MSIL (Microsoft Intermediate Language[3]) do podoby natívnych (strojových) inštrukcií triedy x86.
- Alokácia a dealokácia pamäťových segmentov, ktoré sú asociované s objektmi.
- Garancia typovej bezpečnosti.
- Správa aplikačných domén a programových vlákien.
- Automatický manažment životných cyklov objektov, ktorý zabezpečuje automatický správca pamäte (Garbage Collector, GC).
- Interoperabilita medzi vrstvami natívneho a riadeného programového kódu.

Virtuálny exekučný systém vykonáva svoju činnosť „nad“ vrstvou, ktorú tvorí operačný systém triedy Microsoft Windows[4]. Vizualizáciu komunikačného modelu medzi riadenou aplikáciou, virtuálnym exekučným systémom, operačným systémom Microsoft Windows a hardvérom počítača zobrazuje obr. 1.2.

[3] Jazyk MSIL sa niekedy označuje len ako IL (Intermediate Language), resp. CIL (Common Intermediate Language).

[4] Napriek tomu, že virtuálny exekučný systém, podobne ako aj ďalšie základné komponenty vývojovo-exekučnej platformy Microsoft .NET Framework 3.5, boli úspešne prenesené na iné počítačové platformy a operačné systémy (napríklad Mac OS či Linux), budeme v tejto publikácii uvažovať iba o operačných systémoch triedy Microsoft Windows.



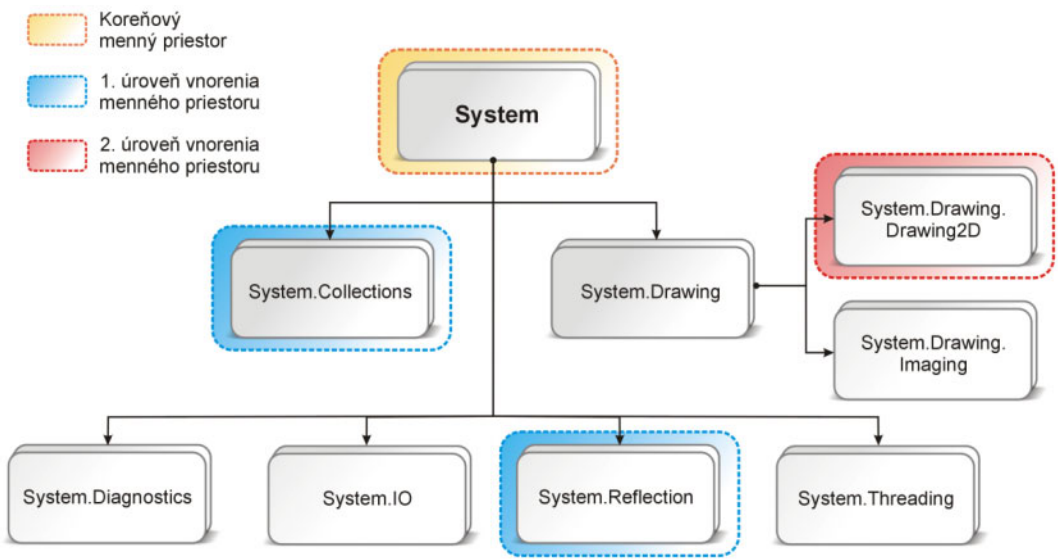
Obr. 1.2: Komunikačný model medzi riadenou aplikáciou, virtuálnym exekučným systémom, operačným systémom a hardvérom počítača

1.2 Bázová knižnica tried (BCL)

Bázová knižnica tried je abstraktnou knižnicou, ktorá bola vytvorená v súlade so smernicami pre hladkú implementáciu objektovo orientovanej a komponentovej paradigmy programovania. Bázová knižnica tried je množinou tisícov hodnotových a odkazových dátových typov, ku ktorým patria štruktúry, vymenované typy, triedy,

rozhrania a delegáti. Dátové typy zapuzdrené do bázovej knižnice tried disponujú vopred naprogramovanou funkcionalitou. Možnosť ich okamžitého použitia efektívne minimalizuje čas, ktorý by bolo nutné vynaložiť na vytvorenie adekvátneho dátového typu v réžii finálneho vývojára. Okrem rýchlej technologickej adopcie je konkurenčnou výhodou bázovej knižnice tried rovnako aj zvýšenie pracovnej produktivity vývojára a v neposlednom rade tiež poskytnutie vysokej miery abstrakcie pri práci s objektmi a spriaznenými dátovými štruktúrami.

Všetky hodnotové a odkazové dátové typy deklarované v bázovej knižnici tried sú na základe svojho pracovného zaradenia roztriedené do fyzicko-logických celkov, tzv. menných priestorov. Koreňový menný priestor má názov **System**, pričom združuje všetky základné menné priestory. Z uvedeného je zrejmé, že menné priestory smú byť vnárané do seba. (Technika vnárania menných priestorov znamená umiestnenie jedného menného priestoru do tela iného menného priestoru.) Keďže hĺbka vnárania menných priestorov môže byť variabilná, smieme pre lepšiu názornosť usporiadanie menných priestorov graficky znázorniť pomocou stromových štruktúr (obr. 1.3).



Obr. 1.3: Vybrané menné priestory bázovej knižnice tried

Menné priestory, ktoré sú vnorené v koreňovom mennom priestore **System**, resp. menné priestory, ktoré sú vnorené vo vnorených menných priestoroch, sú dosiahnuteľné aplikáciou operátora priameho prístupu (tiež bodkového operátora) (.). S nárastom úrovní vnárania sa zvyšuje absolútna početnosť výskytu bodkových operátorov v identifikačných výrazoch, pomocou ktorých sú pre nás požadované menné priestory dosiahnuteľné. Napríklad:

- **System.Collections**, **System.Drawing**, **System.Diagnostics**, **System.Reflection** a **System.Threading** (jedna úroveň vnorenia),

- **System.Drawing.Drawing2D**, **System.Drawing.Imaging**, **System.Data.OleDb**, **System.Linq.Expressions** a **System.Security.Cryptography** (dve úrovne vnorenia).

Bázová knižnica tried je rozšíriteľná, čo znamená, že ak to povaha zabudovaných dátových typov umožňuje, môžeme na ich základe vytvárať nové, odvodené, používateľsky deklarované dátové typy. Najväčšou mierou rozšíriteľnosti disponujú triedy a rozhrania.

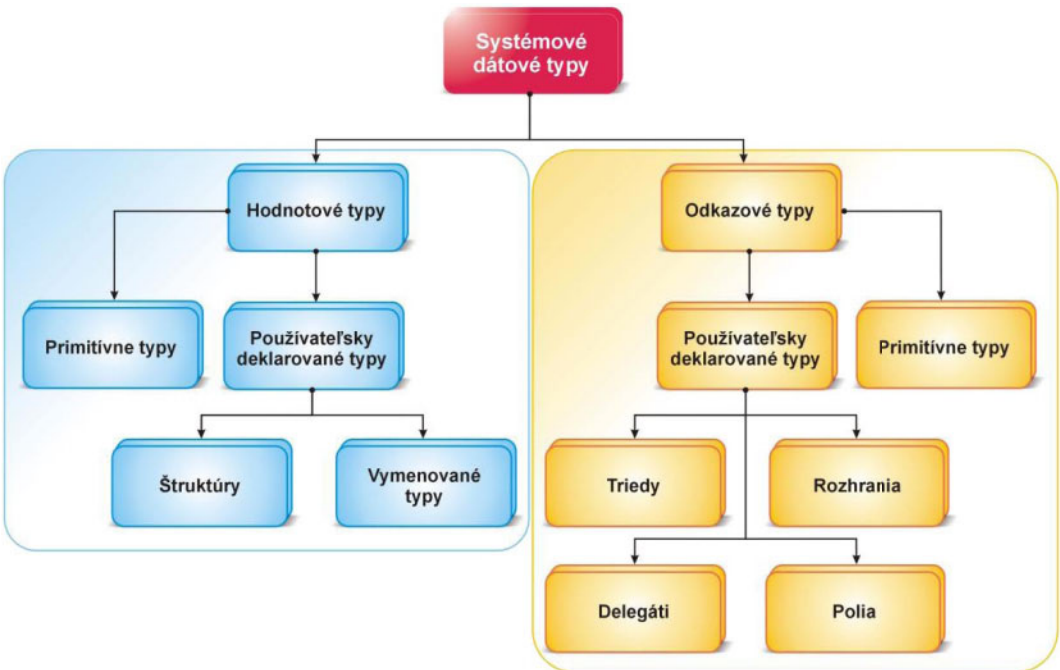
1.3 Spoločný typový systém (CTS)

Spoločný typový systém definuje požiadavky, ktoré musia spĺňať všetky dátové typy, s ktorými vývojári na platforme Microsoft .NET Framework 3.5 pracujú. Spoločný typový systém rozdeľuje dátové typy do dvoch základných skupín:

1. Hodnotové dátové typy.
2. Odkazové dátové typy.

Každá z uvedených množín dátových typov je tvorená primitívnymi typmi a používateľsky deklarovanými typmi (obr. 1.4). Primitívne typy sú dátové typy priamo vstavané do spoločného typového systému, pričom jazykové špecifikácie .NET-kompatibilných programovacích jazykov spravidla definujú špeciálne kľúčové

slová, prostredníctvom ktorých sú tieto primitívne typy explicitne dosiahnuteľné. Používateľsky deklarované dátové typy sú produktom abstraktného myslenia vývojárov, programátorov a softvérových expertov. Tieto dátové typy nie sú vstavané do spoločného typového systému, no spoločný typový systém garantuje podmienky, za akých smú byť uvedené typy vytvárané a používané.



Obr. 1.4: Klasifikácia dátových typov spoločného typového systému

Dátové typy, ktoré definuje spoločný typový systém, sú známe ako systémové dátové typy. Primitívne typy .NET-kompatibilných programovacích jazykov spolupracujú s ekvivalentnými systémovými dátovými typmi. Napríklad, v jazyku C# 3.0 existuje kľúčové slovo **int**, ktoré na syntaktickej úrovni reprezentuje primitívny integrálny znamienkový dátový typ jazyka C# 3.0, ktorý je schopný interpretovať 32-bitové celočíselné hodnoty. V skutočnosti je však kľúčové slovo **int** iba iným pomenovaním (tzv. aliasom) pre príslušný systémový dátový typ **System.Int32**. Presnejšie, primitívny typ **int** jazyka C# 3.0 odkazuje na hodnotovú štruktúru **Int32**,

ktorej deklarácia je umiestnená v koreňovom mennom priestore **System**. Analogické interakcie jestvujú aj medzi ostatnými primitívnymi hodnotovými dátovými typmi jazyka C# 3.0 (tab. 1.1).

Primitívny typ jazyka C# 3.0	Ekvivalentný systémový typ	Rozsah hodnôt	Charakteristika
bool	System.Boolean	logická pravda (true), logická nepravda (false)	8-bitová logická hodnota
char	System.Char	znaky súpravy Unicode (interne 16-bitové celé čísla bez znamienka)	16-bitový číselný kód textového znaku súpravy Unicode
short	System.Int16	<-32768, 32767>	16-bitová celočíselná hodnota so znamienkom
ushort	System.UInt16	<0, 65535>	16-bitová celočíselná hodnota bez znamienka
int	System.Int32	$<-2^{31}, 2^{31} - 1>$	32-bitová celočíselná hodnota so znamienkom
uint	System.UInt32	$<0, 2^{32} - 1>$	32-bitová celočíselná hodnota bez znamienka
long	System.Int64	$<-2^{63}, 2^{63} - 1>$	64-bitová celočíselná hodnota so znamienkom
ulong	System.UInt64	$<0, 2^{64} - 1>$	64-bitová celočíselná hodnota bez znamienka
float	System.Single	$<-3,4 \times 10^{38}, 3,4 \times 10^{38}>$	32-bitová reálna hodnota s jednoduchou presnosťou
double	System.Double	$<-1,79 \times 10^{308}, 1,79 \times 10^{308}>$	64-bitová reálna hodnota s dvojnásobnou presnosťou

Tab. 1.1: Vzťah primitívnych hodnotových typov jazyka C# 3.0 a ekvivalentných systémových dátových typov

Spoločný typový systém vytvára unifikovanú typovú abstrakciu, ktorá je zdieľaná všetkými .NET-kompatibilnými programovacími jazykmi. To je citeľná výhoda, pretože jednotný typový systém napomáha úspešne rozvinúť koncepciu skutočnej jazykovej interoperability. Na platforme Microsoft .NET Framework 3.5 je teda