	Java	Slovenský jazyk (Java)	Pascal (Free Pascal)	Slovenský jazyk (Free Pascal)	Poznámky

Vstupný bod programu	`public class «názov triedy»`
	`{`
	`  public static void`
	`  main(String[] args)`
	`  {`
	`  «príkazy» ;`
	`  }`
	`}`
		`verejná trieda «názov triedy»`
		`{`
		`  verejná statická bez hodnoty`
		`  hlavná(Reťazec[] argum)`
		`  {`
		`  «príkazy» ;`
		`  }`
		`}`
			`program «názov programu» ;`
			`begin`
			`  «príkazy» ;`
			`  «príkaz»`
			`end.`
				`program «názov programu» ;`
				`začiatok`
				`  «príkazy» ;`
				`  «príkaz»`
				`koniec.`
					V Jave musí byť hlavná metóda vložená do definície triedy.
					
					V Pascale je dokonca prvý riadok – uvedenie názvu programu – nepovinný. V Pascale sa za end dáva obvykle bodkočiarka, ale hlavný blok musí byť ukončený bodkou (ako veta).

Blok	`{`
	`  «príkazy» ;`
	`}`
		`{`
		`  «príkazy» ;`
		`}`
			`begin`
			`  «príkazy» ;`
			`  «príkaz»`
			`end;`
				`začiatok`
				`  «príkazy» ;`
				`  «príkaz»`
				`koniec;`
					Blok sa používa na zoskupenie viacerých príkazov. (V Pascale je možné bodkočiarku za posledným príkazom, t. j. napríklad pred `end`, vynechať, resp. ak tam je, v podstate ide o vloženie prázdneho príkazu navyše. Niekedy dokonca bodkočiarka navyše prekáža.)

Základné riadiace štruktúry					
Podmienené spracovanie	`if («podmienka» )` `  «príkaz» ;`	`ak («podmienka» )` `  «príkaz» ;`	`if «podmienka» then` `  «príkaz» ;`	`ak «podmienka» tak` `  «príkaz» ;`	Ak chceme podmienene vykonať viac než jeden príkaz, musíme viacero príkazov uzavrieť do bloku. To platí pre všetky riadiace štruktúry.

Jednoduché vetvenie	`if («podmienka» )`
	`  «príkaz» ;`
	`else`
	`  «príkaz» ;`
		`ak («podmienka» )`
		`  «príkaz» ;`
		`inak`
		`  «príkaz» ;`
			`if «podmienka» then`
			`  «príkaz»`
			`else`
			`  «príkaz» ;`
				`ak «podmienka» tak`
				`  «príkaz»`
				`inak`
				`  «príkaz» ;`
					Aj keď sa v oboch jazykoch chápe táto štruktúra ako „jednota“ – niečo ako „komplexnejší príkaz“ utvorený z menších častí (tak ako všetky štruktúry), iba v Pascale je toto chápanie podčiarknuté zákazom vloženia bodkočiarky za príkazom v hlavnej vetve. To znamená, že v tejto štruktúre v Pascale nesmie byť pred `else` uvedená bodkočiarka! (Súvisí to s rozdielnym chápaním bodkočiarok v Jave a Pascale.)

Viacnásobné vetvenie	`switch («riadiaca premenná» )`
	`{`
	`  case «hodnota 1» :`
	`  «príkazy» ;`
	`  break;`
	
	`  case «hodnota 2» :`
	`  «príkazy» ;`
	`  break;`
	
	`  case «hodnota 3» :`
	`  «príkazy» ;`
	`  break;`
	
	`  …`
	
	`  case «hodnota n» :`
	`  «príkazy» ;`
	`  break;`
	
	`  default:`
	`  «príkazy» ;`
	`}`
		`prepínač («riadiaca premenná» )`
		`{`
		`  prípad «hodnota 1» :`
		`  «príkazy» ;`
		`  preruš;`
		
		`  prípad «hodnota 2» :`
		`  «príkazy» ;`
		`  preruš;`
		
		`  prípad «hodnota 3» :`
		`  «príkazy» ;`
		`  preruš;`
		
		`  …`
		
		`  prípad «hodnota n» :`
		`  «príkazy» ;`
		`  preruš;`
		
		`  predvolené:`
		`  «príkazy» ;`
		`}`
			`case «riadiaca premenná» of`
			`  «hodnota 1» : «príkaz» ;`
			`  «hodnota 2» : «príkaz» ;`
			`  «hodnota 3» : «príkaz» ;`
			`  …`
			`  «hodnota n» : «príkaz»`
			`  else`
			`  «príkazy» ;`
			`  «príkaz»`
			`end;`
				`v prípade, že «riadiaca premenná»`
				`  «hodnota 1» : «príkaz» ;`
				`  «hodnota 2» : «príkaz» ;`
				`  «hodnota 3» : «príkaz» ;`
				`  …`
				`  «hodnota n» : «príkaz»`
				`  inak`
				`  «príkazy» ;`
				`  «príkaz»`
				`koniec;`
					Táto štruktúra vyžaduje v Jave ukončenie každej vetvy príkazom `break` (`preruš`). Je to preto, že umožňuje, aby sa po sebe idúce vetvy prekrývali. Môže to byť výhoda, no pri bežnom používaní je to skôr nevýhoda, pretože v prípade, že programátor zabudne uviesť príkaz `break` na konci niektorej vetvy, dôjde k neúmyselnému prekrytiu vetiev a vzniku logickej chyby, ktorá sa ťažšie hľadá (oproti chybám syntaktickým).
					
					Pascal prekrývanie vetiev neumožňuje. Keď chceme do niektorej vetvy vložiť viacero príkazov, musíme použiť blok. (V tejto štruktúre sa v Pacale pred `else` bodkočiarka vyskytnúť môže (a nemusí), je to podobné ako pred `end`.)

	Java	Slovenský jazyk (Java)	Pascal (Free Pascal)	Slovenský jazyk (Free Pascal)	Poznámky
Cyklus s podmienkou na začiatku	`while («podmienka») «príkaz»;`	`pokiaľ («podmienka») «príkaz»;`	`while «podmienka» do «príkaz»;`	`pokiaľ «podmienka» rob «príkaz»;`	Táto štruktúra je v oboch jazykoch takmer identická. Keď platí podmienka, iterácia sa zopakuje.
Cyklus s podmienkou na konci	`do { «príkazy»; } while («podmienka»);`	`rob { «príkazy»; } pokiaľ («podmienka»);`	`repeat «príkazy»; «príkaz» until «podmienka»;`	`opakuj «príkazy»; «príkaz» dokedy «podmienka»;`	Java a Pascal sa zásadne odlišujú v tom, že v Jave podmienka musí platiť na to, aby sa cyklus opakoval a v Pascale platiť nesmie. (Rezervované slová v tomto prípade zastupujú blok, čiže `repeat` je ako `begin` a `until` je ako `end`.)

Cyklus s explicitným počtom iterácií	`for («deklaračný príkaz»; «podmienka»; «iteračný príkaz») «príkaz»;`	`pre («deklaračný príkaz»; «podmienka»; «iteračný príkaz») «príkaz»;`	`for «riadiaca premenná» := «spodná hranica» to «horná hranica» do «príkaz»;`
			
			`for «riadiaca premenná» := «horná hranica» downto «spodná hranica» do «príkaz»;`
				`pre «riadiaca premenná» := «spodná hranica» až «horná hranica» rob «príkaz»;`
				
				`pre «riadiaca premenná» := «horná hranica» dolepo «spodná hranica» rob «príkaz»;`
					Táto štruktúra má v Jave trochu komplikovanejší zápis, pretože ide vlastne o „prepis“ cyklu s podmienkou na začiatku… (podrobnosti sme vysvetlili inde).
					
					V Pascale má štruktúra dva varianty, podľa toho, či má hodnota riadiacej premennej počas iterácií cyklu stúpať alebo klesať.

Spracovanie výnimiek	`try { «príkazy»; } catch («výnimka» «premenná») { «príkazy»; }`	`skús { «príkazy»; } zachyť («výnimka» «premenná») { «príkazy»; }`	`try «príkazy»; except on «premenná»: «výnimka» do «príkazy»; end;`	`skús «príkazy»; okrem «premenná»: «výnimka» rob «príkazy»; koniec;`	Výnimky sú novším javom v programovacích jazykoch. Do Javy prišli ešte v čase jej vzniku, no do Pascalu boli pridané neskôr. Používajú sa na hlásenie rôznych „výnimočných“ stavov v programe, čím sa myslí na chybové stavy. Programátor má zabezpečiť korektné zachytenie a spracovanie týchto stavov.
Základné aritmetické operácie (s operáciou spájania reťazcov), relačné operácie a operácie (ne)zhody					
Priradenie	`«premenná» = «výraz»;`	`«premenná» = «výraz»;`	`«premenná» := «výraz»;`	`«premenná» := «výraz»;`	Tu zdanlivo nie je rozdiel. Na programátorov, ktorí menia programovací jazyk (v smere Pascal → Java) však čaká nepríjemné prekvapenie. To, čo je v Pascale operátorom zhody (=), je v Jave operátorom priradenia…

Aritmetické operácie a operácia spájania reťazcov
 	
(spájanie reťazcov nie je aritmetickou operáciou, ale používa sa na ňu jeden z aritmetických symbolov)
	`«výraz/reťazec» + «výraz/reťazec»`
	`«výraz» - «výraz»`
	`«výraz» * «výraz»`
	`«výraz» / «výraz»`
	`«výraz» % «výraz»`
		`«výraz/reťazec» + «výraz/reťazec»`
		`«výraz» - «výraz»`
		`«výraz» * «výraz»`
		`«výraz» / «výraz»`
		`«výraz» % «výraz»`
			`«výraz/reťazec» + «výraz/reťazec»`
			`«výraz» - «výraz»`
			`«výraz» * «výraz»`
			`«výraz» / «výraz»`
			`«výraz» div «výraz»`
			`«výraz» mod «výraz»`
				`«výraz/reťazec» + «výraz/reťazec»`
				`«výraz» - «výraz»`
				`«výraz» * «výraz»`
				`«výraz» / «výraz»`
				`«výraz» del «výraz»`
				`«výraz» zvš «výraz»`
					Operátory +, -, * a / majú pre nás takmer prirodzený význam a to: sčítanie alebo spájanie reťazcov (podľa kontextu), odčítanie, násobenie a podiel. Operátor % v Jave znamená modulus – zvyšok po delení. Rovnaký význam má operátor `mod` v Pascale. Operátor `div` v Pascale znamená celočíselné delenie, pričom Java v tomto nepozná rozdiel a pri celočíselnom delení prípadnú desatinnú časť jednoducho „odstrihne“… (Pozor, nezaokrúhli! Čiže ak by výsledok reálnočíselného podielu mal byť napríklad 5,8, tak výsledok celočíselného by bol 5.)

Relačné operácie	`«výraz» < «výraz»`
	`«výraz» > «výraz»`
	`«výraz» <= «výraz»`
	`«výraz» >= «výraz»`
		`«výraz» < «výraz»`
		`«výraz» > «výraz»`
		`«výraz» <= «výraz»`
		`«výraz» >= «výraz»`
			`«výraz» < «výraz»`
			`«výraz» > «výraz»`
			`«výraz» <= «výraz»`
			`«výraz» >= «výraz»`
				`«výraz» < «výraz»`
				`«výraz» > «výraz»`
				`«výraz» <= «výraz»`
				`«výraz» >= «výraz»`
					Operátory porovnania ≤ a ≥ sú v obidvoch jazykoch zložené z dvoch po sebe idúcich znakov: <= a >=. Je to preto, že znaky ≤ a ≥ nie je možné na štandardnej klávesnici PC vložiť triviálnym spôsobom a tiež z historických dôvodov.

	Java	Slovenský jazyk (Java)	Pascal (Free Pascal)	Slovenský jazyk (Free Pascal)	Poznámky
Operácie (ne)zhody	*«výraz»* == *«výraz»* *«výraz»* != *«výraz»*	*«výraz»* == *«výraz»* *«výraz»* != *«výraz»*	*«výraz»* = *«výraz»* *«výraz»* <> *«výraz»*	*«výraz»* = *«výraz»* *«výraz»* <> *«výraz»*	Tu je jeden „zákerný“ rozdiel, na ktorý sme upozornili už pri operácii priradenia. Operátor zhody z Pascalu (=) znamená v Jave priradenie (pozri vyššie), preto má Java na vyjadrenie operácie zhody rezervovaný operátor dvojitého rovná sa (==). Tento rozdiel býva zdrojom častých chýb (nielen) pre programátorov, ktorí menia jazyk v smere Pascal → Java. Jazyky sa odlišujú aj v operátoroch nezhody (ale menej „zákerne“). Pascal vyjadruje nezhodu znakmi <> a Java znakmi != (pričom samostatný znak ! je logickou operáciou negovania).
Logické operácie – „skladanie podmienok“					
Logický súčet (operátor „or“ – alebo)	*«výraz»* || *«výraz»*	*«výraz»* || *«výraz»*	*«výraz»* **or** *«výraz»*	*«výraz»* **alebo** *«výraz»*	Pascal je jazyk slovných operátorov. Všetko, čo sa dá, je v ňom vyjadrené prednostne slovne. Logický súčet je logická operácia, ktorej výsledkom je pravda v prípade, že je pravdivá hodnota aspoň jedného z operandov (v našom prípade máme na mieste operandu *«výraz»* , čím máme na mysli logický výraz, aby bolo jasné, že operandom môže byť čokoľvek, čoho pravdivostná hodnota je `true` – `pravda` alebo `false` – `lož`).
Logický súčin (operátor „and“ – a súčasne)	*«výraz»* && *«výraz»*	*«výraz»* && *«výraz»*	*«výraz»* **and** *«výraz»*	*«výraz»* **a** *«výraz»*	Logický súčin je logická operácia, ktorej výsledkom je pravda len v takom prípade, keď sú pravdivé hodnoty obidvoch operandov naraz.
Logická negácia (operátor „not“ – nie)	!*«výraz»*	!*«výraz»*	**not** *«výraz»*	**nie** *«výraz»*	Logická negácia jednoducho prevracia pravdivosť logickej hodnoty (výrazu).
Údajové typy (primitívne – základné, resp. s jednou výnimkou…)					
Celé čísla	byte, short, int, long	bajt, krátke, cel, dlhé	**byte, word, integer, int64…**	**bajt, slovo, celéčíslo, cel64…**	Tieto údajové typy slúžia na uchovanie celých čísiel. Odlišujú sa v rozsahu hodnôt, ktoré je do nich možné uložiť.
Reálne čísla	float, double	plávajúce, dvojité	**real, single, double…**	**reálne, jednoduché, dvojité…**	Tieto údajové typy slúžia na uchovanie reálnych čísiel. Podobne ako celočíselné typy aj tieto odlišujú v rozsahu hodnôt, ktoré je do nich možné uložiť.
Logické hodnoty	boolean	booleovské	**boolean**	**booleovské**	Tento údajový typ slúži na uchovanie pravdivostných hodnôt `true` a `false`.
Znaky	char	znk	**char**	**znk**	Tento údajový typ slúži na uchovanie jedného znaku. Vo Free Pascale má viacero variantov.

	Java	Slovenský jazyk (Java)	Pascal (Free Pascal)	Slovenský jazyk (Free Pascal)	Poznámky
Reťazce	`String`	`Reťazec`	`string`	`reťazec`	Tento údajový typ slúži na ukladanie reťazcov zložených zo znakov. Môžu to byť ľubovoľné texty, kódy, prepisy čísiel, čokoľvek, čo sa dá vyjadriť znakmi. V Jave je tento údajový typ triedou, v Pascale ide tiež o špeciálny typ. Oba jazyky sa však usilujú programátorov týmto faktom príliš nezaťažovať.
Deklarácie a definície premenných					

Deklarácie premenných	*«údajový typ» «názov premennej»*`;`
	
	Príklady:
	`int i;`
	`float x;`
	`String fullName;`
		«údajový typ» «názov premennej»`;`
		
		Príklady:
		`cel i;`
		`plávajúce x;`
		`Reťazec celéMeno;`
			«názov premennej»`:` *«údajový typ»*`;`
			
			Príklady:
			`var i: integer;`
			`x: real;`
			`fullname: string;`
				«názov premennej»`:` *«údajový typ»*`;`
				
				Príklady:
				`prem i: celéčíslo;`
				`x: reálne;`
				`celémeno: reťazec;`
					V Pascale smie byť deklarácia umiestnená výhradne v deklaračnej časti, v Jave sa smie deklarácia vyskytnúť aj v príkazovej časti (v skutočnosti v Jave v deklaračnej časti deklarujeme atribút triedy, nie „klasickú“ premennú).

Definície premenných	*«deklarácia»* `=` *«hodnota»*`;`
	
	Príklady:
	`int i = 10;`
	`float x = 5.2;`
	`String fullName = "Joseph Black";`
		«deklarácia» `=` *«hodnota»*`;`
		
		Príklady:
		`cel i = 10;`
		`plávajúce x = 5.2;`
		`Reťazec celéMeno = "Jozef Čierny";`
			«deklarácia» `=` *«hodnota»*`;`
			
			Príklady:
			`var i: integer = 10;`
			`x: real = 5.2;`
			`fullname: string =`
			`'Joseph Black';`
				«deklarácia» `=` *«hodnota»*`;`
				
				Príklady:
				`prem i: celéčíslo = 10;`
				`x: reálne = 5.2;`
				`celémeno: reťazec =`
				`'Jozef Čierny';`
					Všimnite si, že v Pascale je „klasický“ Pascalovský operátor priradenia (`:=`) nahradený jednoduchým rovná sa (`=`)…
					
					O výskyte definícií v programe platí to isté, čo pri deklaráciách.

Štandardný vstup a výstup					
Výstup – výpis údajov na konzolu	`System.out.print(`*«reťazec»*`);` `System.out.println(`*«reťazec»*`);`	`Systém.výstup.tlač(`*«reťazec»*`);` `Systém.výstup.tlačrdk(`*«reťazec»*`);`	`write(`*«zoznam argumentov»*`);` `writeln(`*«zoznam argumentov»*`);`	`píš(`*«zoznam argumentov»*`);` `píšrdk(`*«zoznam argumentov»*`);`	Štandardným výstupom sa za normálnych okolností myslí výpis textu na konzolu. Táto časť je v obidvoch jazykoch podobná. Oba jazyky poskytujú dva varianty príkazov. Jeden vypíše zadaný obsah tak, aby mohol najbližší výpis pokračovať na tom istom riadku, druhý po výpise automaticky prejde na nový riadok. V Jave príkazy prijímajú jediný reťazec, ktorý môžeme získať aj zlúčením viacerých údajov konvertovateľných na text. Na zlúčenie reťazcov slúži operátor `+`. V Pascale príkazy prijímajú ľubovoľne dlhý zoznam argumentov oddelených čiarkami.
Vstup – načítanie údajov z konzoly	*«Nie je triviálne…»*	*«Nie je triviálne…»*	`read(`*«zoznam argumentov»*`);` `readln(`*«zoznam argumentov»*`);`	`čítaj(`*«zoznam argumentov»*`);` `čítajrdk(`*«zoznam argumentov»*`);`	Štandardným vstupom sa za normálnych okolností myslí načítanie údajov z konzoly.

Vstup z konzoly na príklade načítania celého čísla	*Pred definíciou triedy:*
	`import java.util.Scanner;`
	
	V príkazovej časti metódy:
	`Scanner scan =`
	`new Scanner(System.in);`
	`int number = scan.nextInt();`
		Pred definíciou triedy:
		`importuj java.pomôc.Skener;`
		
		V príkazovej časti metódy:
		`Skener sken =`
		`nový Skener(Systém.vstup);`
		`cel číslo = sken.ďalšieCel();`
			V deklaračnej časti programu:
			`var number: integer;`
			
			V príkazovej časti programu:
			`readln(number);`
				V deklaračnej časti programu:
				`prem číslo: celéčíslo;`
				
				V príkazovej časti programu:
				`čítajrdk(číslo);`
					Vysvetliť technicky presne to, čo znamená vykonať štandardný vstup v Jave nie je triviálne a teda ani vhodné pre začiatočníkov. Prístup jazyka Pascal je v tomto smere vhodnejší. My si tento nedostatok jazyka Java nahrádzame pomocou triedy `Vstup`.