CodEx – Kterak sobě pořídit úložku

Obsah

1. Úvod 1
2. Příprava úlohy 1
2.1 Zadání 2
2.2 Testovací data 2
2.2.1 Vstup 2
2.2.2 Výstup 3
2.3 Vzorové řešení 3
3. Vložení do CodExu 3
3.1 Vytvoření nové úlohy 4
3.2 Text zadání 4
3.3 Připojené soubory 4
3.4 Nastavení testů 4
3.4.1 Filtry a judge 5
3.4.2 Paměťové a časové limity 6
3.4.3 Konfigurační soubory 6
3.5 Soubory testů 7
3.6 Testování úlohy a vzorová řešení 7
4. Správa existujících úloh 7
4.1 Delegování práv 8
5. Závěrem 8
Příloha A: Značky povolené v zadání 9

1. Úvod

Tato příručka je určena především operátorům, kteří mají možnost vytvářet vlastní úlohy. Následující dokument berte jako doporučený postup, kterak úlohu vytvořit a vložit do CodExu.

V následujícím textu se předpokládají obecné znalosti z *Příručky běžného uživatele*.

2. Příprava úlohy

Prvním krokem při zadávání nové úlohy je její příprava. Zpravidla je třeba připravit celkem tři věci:

- text zadání
- sada testovacích dat (vstupy a vzorové výstupy)
- vzorové řešení

U složitějších úloh může být potřeba připravit také judge (aplikaci, která zkontroluje uživatelský výstup).

2.1 Zadání

Zadání má podobu čistého textu, ve kterém můžete používat některé značky jazyka XHTML (včetně obrázků, tabulek a odkazů). Přesný popis povolených značek naleznete na konci tohoto dokumentu.

Zadání musí být naprosto přesné (zejména pokud jde o popis formátu vstupu a výstupu), aby nemohlo dojít ke špatnému pochopení ze strany uživatelů, kteří úlohu řeší. Je vhodné úlohu opatřit také nějakým neformálním příběhem, aby byla pro čtenáře lépe stravitelná (v tomto ohledu doporučuji nechat se inspirovat zadáními úloh praktické části celostátního kola MO-P, úloh KSP nebo jiné programátorské soutěže).

2.2 Testovací data

Testovací data představují sadu vstupů a výstupů, na které je odevzdané řešení spuštěno. Jedním *testem* budeme dále rozumět jednu sadu vstupních dat a k nim odpovídající vzorový výstup. Testů může být libovolně mnoho. Je nezbytné mít alespoň jeden test a není vhodné mít jich více než řádově desítky. Rozumný počet je okolo deseti testů na jednu úlohu, což zpravidla stačí na otestování všech speciálních případů i extrémně velkých vstupů. Pro každý test je potřeba připravit následující:

2.2.1 Vstup

Testovaný program může vstupní data dostat třemi způsoby:

- ze standardního vstupu (i v takovém případě musí být data uložena v souboru)
- z jednoho souboru předem daného názvu
- z více souborů, které musí být uloženy v jednom adresáři

V prvním případě jsou vstupní data poslána na standardní vstup testovaného programu při jeho spuštění, ve zbylých dvou případech je k příslušným souborům poskytnut programu přístup.

Testy jsou číslovány od jedničky a vstupní soubory testů jsou pojmenovávány **`<číslo_testu>.in`**, tedy např **`2.in`** pro druhý test (při testování je tento soubor nabídnut programu pod určeným názvem). V případě, že používáte více vstupních souborů, musí být všechny vstupní soubory umístěny v adresáři s názvem **`<číslo_testu>.in`** a názvy jednotlivých souborů musí být takové, pod jakými k nim má přistupovat testovaný program.

Maximální velikost testovacích dat není omezena, avšak u velkých souborů může nastat potíž s natahováním na CodEx přes webové rozhraní (viz kapitola 3.5). Určitě není dobré zadávat extrémně velká data (desítky megabyte), protože s tak velkými daty se již špatně manipuluje. Na druhou stranu je ve většině případů rozumné zadat několik testů s daty velikosti jednotek megabyte, aby bylo možné otestovat časovou složitost.

Rovněž je třeba brát zřetel na fakt, že čtení z disku prodlužuje čas běhu programu (a především zpomaluje celý server), takže není vhodné zadávat úlohy, které hodně používají soubory a málo počítají (jako je například Polyfázové třídění).

Data by měla být za normálních okolností uložena v textových souborech. Binární soubory používejte pouze tehdy, pokud by použití textového souboru výrazně zkomplikovalo zadání úlohy.

2.2.2 Výstup

Program může své výsledky zapsat:

- na standardní výstup
- do jednoho souboru (s předem daným názvem)

Výstup programu se následně porovná se vzorovým výstupním souborem pomocí vybraného *judge*. Podrobnosti o tom, jak judge funguje, naleznete v kapitole 3.4.1. Vzorový výstup musí být uložen v souboru **`<číslo_testu>.out`**, tedy například pro druhý test to bude **`2.out`**, bez ohledu na to, zda program píše výsledky na standardní výstup, nebo do souboru.

U složitějších úloh se může stát, že možných správných výstupů je větší množství a standardní porovnávání se vzorovými řešeními nestačí. V takovém případě je možné napsat vlastní skript (judge), který zkontroluje, zda je výstup v pořádku. Použití vlastního judge je mírně komplikovanější a určitě se neobejde bez konzultace a asistence správce.

2.3 Vzorové řešení

Vzorové řešení by mělo být nejlepším možným algoritmickým řešením úlohy (tzn. s nejlepší časovou a paměťovou složitostí). V ideálním případě by mělo být napsané alespoň ve dvou používaných jazycích. Pokud chcete používat také jazyk C#, je nezbytné napsat vzorové řešení také v něm, protože programy běžící v prostředí Mono zaberou více paměti a běží déle než řešení v ostatních jazycích.

Řešení by mělo být napsané čitelně a okomentované, aby ho mohli snadno prostudovat další operátoři. Další komentáře k němu můžete napsat přímo do CodExu.

Vzorových řešení může být i více, pokud chcete ukázat různé přístupy k řešení problému, případně představit různě efektivní řešení a následně podle nich vyladit časové a paměťové limity pro testy (např. tak, že za neefektivní řešení bude nějaký malý počet bodů, za efektivnější bude větší počet bodů a za nejlepší bude maximální počet bodů).

3. Vložení do CodExu

Na vkládání úloh do CodExu je třeba mít potřebná práva. Pokud práva nemáte, kontaktujte svého správce. Při vkládání úloh vřele doporučujeme dodržovat následující postup:

- Vytvořte novou (zamčenou) úlohu a vyplňte její parametry.
- Vyplňte text zadání.
- Pokud je potřeba, natáhněte přiložené soubory k textu zadání.
- Nastavte testy.
- Natáhněte soubory testů.
- Úlohu otestujte a případně dolaďte časové a paměťové limity podle vzorových testů.

- Odemkněte úlohu.

Výše uvedené kroky jsou podrobně rozebrány v následujících kapitolách

3.1 Vytvoření nové úlohy

Novou úlohu můžete vytvořit pomocí formuláře na stránce „**Úlohy – Vytvořit**“. Při vytváření je potřeba zadat základní informace o úloze, přičemž význam jednotlivých položek je vysvětlen přímo ve formuláři. Úlohu vytvářejte vždy zamčenou a neveřejnou a neodemykejte ji dřív, než bude kompletní a otestovaná. Pokud plánujete úlohu zveřejnit, učiňte tak co nejdříve, ale rozhodně ne dřív, než vyplníte text zadání.

Zvláštní pozornost věnujte také povoleným příponám. Cvičící nebudou moci zadat úlohu k řešení pro jiné jazyky, než které zde povolíte. To může být užitečné, pokud např. vymýšlíte úlohu pouze pro Pascal nebo máte problémy s některým z překladačů. Zejména pak nepovolujte C#, pokud jste pro něj nenapsali vzorové řešení (a nebudete jej tedy testovat).

3.2 Text zadání

Po vytvoření úlohy jste automaticky přesměrováni na stránku „**Zadání – Upravit**“, kam můžete vložit text vašeho zadání. Text se vkládá do formulářového okna, avšak je vhodnější zkopírovat do něho již předpřipravený text metodou copy-and-paste.

Pokud vkládáte první verzi zadání nebo provádíte rozsáhlejší změny v sémantice, použijte tlačítko *uložit novou verzi*. Tím zajistíte, že staré zadání nebude přepsáno, ale odsune se do historie. Tak umožníte ostatním operátorům, aby si prostudovali provedené změny a případně na to upozornili členy svých skupin.

V okamžiku, kdy je text zadání uložen a nepředpokládáte, že se bude ještě zásadně měnit, můžete úlohu zveřejnit.

3.3 Připojené soubory

V textu zadání se můžete odkazovat na externí soubory (obrázky, dokumenty apod.) pomocí XHTML značek **<img>** a **<a>**. Tyto soubory jsou uloženy v samostatném adresáři pro každé zadání a přístup k nim máte přes stránku *připojené soubory*. V textu pak můžete nastavit cestu k těmto souborům jako `$DIR/nazev_souboru`, kde `$DIR/` je automaticky nahrazen cestou k reálnému souboru (resp. potřebným URL fragmentem).

Např. pokud připojíte k zadání soubor `picture.png` do textu pak tento obrázek vložíte značkou:

```
<img src="$DIR/picture.png" alt="popis obrázku" />
```

Názvy připojených souborů můžou obsahovat pouze písmena anglické abecedy, čísla a znaky „-“ (pomlčka), „_“ (podtržítko) a „.“ (tečka). Navíc tečka se nesmí vyskytovat na začátku.

3.4 Nastavení testů

Dalším krokem je nastavení testů. Nastavení je rozděleno do dvou stránek. Základní nastavení se nachází na „**Nastavení testů**“, nastavení bodů a paměťové a časové limity se

nachází na stránce „**Limity testů**“. Podrobnosti o jednotlivých parametrech jsou uvedeny přímo ve formulářích.

Nejdůležitější položkou je seznam testů, který ovlivňuje, jak se mají jmenovat soubory testů a kolik jich má být.

3.4.1 Fitry a judge

Nejprve je třeba pochopit, jak funguje vyhodnocování výsledků testovaného programu. Výstupní data jsou nejprve prohnána zvoleným *filtrem*, který je může případně modifikovat. Následně je spuštěn *judge*, který dostane výstup programu a porovná jej se vzorovým výstupem. Judge pak oznámí verdikt, zda je výstup v pořádku, a pokud ano, uživatel dostane příslušný počet bodů (promile).

V současné době je v CodExu pouze jeden filtr – *filtr komentářů*. Pokud je nastaven, odstraní se z výstupu programu uživatelské komentáře. Za komentář je považován text začínající „`//`“ (dvěma lomítky) a vše, co za nimi následuje až do konce řádku. Tím je umožněno slabším uživatelům, aby si do výstupu přidali své řetězce, které jim usnadní ladění a zpřehlední výstup. Pokud není nastaven žádný filtr, výstup se nijak nemění.

Běžné judge již jsou v CodExu připraveny:

- *Textový judge* – Judge který porovnává soubory tak, že ignoruje počty a typy běžných bílých znaků (mezery, tabelátory) a kontroluje pouze textové tokeny a jejich pořadí. Jeho varianta, která navíc *ignoruje zalomení řádků*, pak počítá znak zalomení řádku také jako běžný bílý znak (třeba mezeru).
- *Striktní (binární) judge* – Judge který porovnává soubory binárně, jeden byte po druhém, a pokud najde sebemenší odchylku, prohlásí řešení za chybné. Tento judge je vhodný zejména pro porovnávání binárních souborů.
- *Floatový judge* – Porovnává soubory po tokenech, ale v případě, že je tokenem desetinné číslo, bude jej tak kontrolovat a při porovnávání bere ohled i na možnou chybu v zaokrouhlování (tj. počítá s tím, že výsledky se mohou nepatrně lišit). Jeho varianta, která navíc *ignoruje zalomení řádků*, pak počítá znak zalomení řádku také jako běžný bílý znak (třeba mezeru).
- *Textový judge ignorující pořadí tokenů v řádku* – Funguje jako běžný textový judge, ale nehlídá pořadí tokenů v řádku (rozdělení tokenů na řádky a pořadí řádků ale stále hlídá).
- *Textový judge ignorující pořadí řádků* – Funguje jako textový judge, ale nezáleží mu na pořadí řádků. Na pořadí tokenů v jednotlivých řádcích ovšem záleží.
- *Textový judge ignorující pořadí řádků a tokenů v řádku* – Kombinace předchozích dvou judge, která ignoruje jakékoli pořadí. Nicméně stále hlídá, zda jsou tokeny správně rozděleny po řádcích.
- Textový judge ignorující zalomení řádků a pořadí tokenů v souboru – Speciální varianta, která na celý soubor nahlíží, jako na množinu tokenů. Zalomení řádků ani jakékoli pořadí zde nehraje roli.

Pokud by vám nevyhovoval žádný předpřipravený judge, je možné napsat si judge vlastní (typickým příkladem jsou úlohy typu: „Pokud existuje řešení víc, najděte libovolné

z nich.“). Judge může být napsán v libovolném běžném překládaném nebo skriptovacím jazyce (např. v C nebo C++). Názvy souborů, které má porovnávat, a případně další soubory (konfigurační data apod.) by měl dostat jako argumenty z příkazové řádky. Výsledek judge vrací prostřednictvím návratové hodnoty programu. Vrátí-li judge 0, je výsledek považován za správný. V ostatních případech se nahlásí chyba a uživateli nejsou přiděleny body. Obvykle se ještě rozlišují hodnoty: 1 = řešení není v pořádku, 2 = interní chyba judge.

Instalace a nastavení vlastního judge na CodEx musí následně provést správce, který judge před přidáním zkontroluje.

3.4.2 Paměťové a časové limity

Nastavení paměťových a časových limitů a získaných bodů naleznete na stránce „**Limity testů**“. Nastavit je můžete obecně, nebo pro každý test zvlášť. Navíc se rozlišují obecné hodnoty platné pro všechny jazyky a specializované pro vybraný jazyk. Jednotlivé hodnoty se skládají ve níže uvedeném pořadí. Pokud některá hodnota není vyplněna, vezme se obecnější – tj. další v pořadí:

- hodnoty nastavené přímo u jednotlivých testů pro konkrétní jazyk
- hodnoty nastavené u jednotlivých testů pro všechny jazyky
- výchozí hodnoty pro konkrétní jazyk
- výchozí hodnoty pro všechny jazyky

U limitů nejprve nastavte vyšší hodnoty, aby vaše řešení určitě sběhlo. Až úlohu řádně otestujete (nejlépe na několika vzorových řešeních), můžete limity upravit. Limity by neměly být příliš těsné.

U časových limitů je vhodné přidat nejméně 15% oproti času, za který sběhlo vzorové řešení (minimálně však alespoň 0,1 s). Rovněž nemá smysl pracovat s jemnější granularitou, než 0,1 s, protože měření času není úplně přesné a závisí na aktuálním vytížení serveru (tj. kolik přerušení, výpadků stránek, případně dalších událostí nastane při běhu programu).

Paměťový limit by neměl být příliš přísný. Cílem je omezit řešení s nevhodnou asymptotickou složitostí, ovšem už ne tolik multiplikativní konstantu složitosti. Výsledné číslo je dobré zaokrouhlit nahoru na celé megabyte. Minimální doporučená hodnota je 8 MB pro běžné jazyky a 16 MB pro jazyk C#. U jazyka C# je navíc třeba si uvědomit, že nízký limit může mít za následek časté spouštění garbage collectoru a tím ovlivnit také dobu běhu programu.

3.4.3 Konfigurační soubory

Všechna nastavení se ukládají do konfiguračního souboru, který má vždy název `config` a je uložen ve stejném adresáři jako testovací data (viz kapitola 3.5). Pokaždé, když znovu uložíte formulář s nastavením nebo časovými limity, konfigurační soubor se přegeneruje.

Konfigurační soubory lze upravovat i ručně, avšak tento postup je doporučen pouze velmi pokročilým autorům úloh, kteří jsou obeznámeni s jádrem vyhodnocovacího systému. Nicméně i začínající autor může využít rozhraní pro manipulaci se soubory k zálohování a obnově konfigurace úlohy stažením, resp. natažením souboru `config`.

3.5 Soubory testů

Na stejnojmenné stránce máte přístup ke všem souborům testů (a také ke konfiguračnímu souboru). Pomocí formuláře můžete na server natáhnout soubory s testovacími daty a vzorovými výsledky (viz kapitola 2.2). Každý test očekává jeden vstupní soubor nebo adresář (v závislosti na typu vstupu) s názvem **`<číslo_testu>.in`** a jeden výstupní soubor s názvem **`<číslo_testu>.out`**. Pokud je například v konfiguraci úlohy uveden seznam testů „1 2 3 4“, je potřeba natáhnout soubory **`1.in`**, **`1.out`**, **`2.in`** … až 4**`.in`** a 4**`.out`**.

CodEx běží interpretovaně pod PHP, které má zpravidla nastavený limit na velikost natahovaných dat a souborů (typicky 8 MB). Pokud byste potřebovali uložit na CodEx větší soubor, kontaktujte správce aplikace. Zkušenější uživatelé mohou také použít přístup přes SSH.

Používáte-li vstup a výstup z/do jednoho souboru, vyhodnocovací systém automaticky přejmenuje soubory na zvolená jména. Název, pod kterým uvidí soubor testovaný program, je možné zvolit libovolně (není třeba omezovat se konceptem číslo.in a číslo.out).

Používáte-li vstup z více souborů, umístěte tyto soubory do příslušného podadresáře s názvy, pod kterými je dostane k dispozici testovaný program.

3.6 Testování úlohy a vzorová řešení

Ke každé úloze existuje databáze vzorových a testovacích řešení. Každý uživatel s potřebnými právy (např. operátoři) sem může odevzdat zdrojový kód. Kód je opraven a vyhodnocen stejným způsobem jako u řešení odevzdaných k zadané úloze v nějaké skupině.

Tento mechanismus lze jednak použít k otestování úlohy a jednak je vhodné, aby sem autor umístil alespoň jedno vzorové řešení, které bude veřejné, a tedy viditelné pro všechny uživatele, kteří mají právo je vidět. Vzorové řešení může ostatním operátorům pomoci při odhalování potíží nebo při rozhodování, zda úlohu zadat, či nikoli.

Testování úlohy má dva významy. Jednak tím ověříte, že zadaná úloha je funkční a že vaše vzorové řešení dostane plný počet bodů, jednak vám to umožní lépe nastavit časové a paměťové limity (viz kapitola 3.4.2).

Poté, co je úloha řádně otestována a limity jednotlivých testů jsou doladěny, můžete úlohu odemknout a tím umožníte ostatním operátorům, aby ji mohli zadávat ve svých skupinách.

4. Správa existujících úloh

Není-li stanoveno správcem jinak, potom každý, kdo zveřejní úlohu, je povinen se o ni starat (odstraňovat případné chyby apod.). Pokud je potřeba upravit úlohu, která je zadaná v nějaké skupině a uživatelé ji řeší, je vhodné dodržovat určitá pravidla.

Před jakoukoli změnou je nezbytné úlohu zamknout. Pokud plánujete větší úpravy, je dobré chvíli počkat (alespoň několik desítek vteřin), aby zamknutí zaregistrovali ostatní uživatelé, kteří jsou zrovna přihlášeni a pracují vaší s úlohou (např. ji řeší). Po dokončení úprav úlohu opět odemkněte. V případě, že se zásadně změnilo zadání, upozorněte na to vhodnou cestou alespoň ostatní operátory.

Při úpravách je také třeba mít na paměti, že někteří uživatelé mohou mít nastavené zasílání oznámení při změně úlohy. U odemčených úloh se toto oznámení se posílá vždy, když je libovolná část úlohy změněna (parametry, konfigurace, soubory testů apod.). Pokud úlohu zamknete, oznámení se neposílají s každou změnou, ale až v okamžiku, kdy je úloha opět odemčena.

V okamžiku, kdy se již nehodláte dále o úlohu starat, můžete ji předat jinému uživateli, který má také právo vytvářet úlohy. Předání se provádí pomocí formuláře na stránce „**Úlohy – Vlastnosti**“. Po předání bude nový majitel veden jako *autor* úlohy.

4.1 Delegování práv

Pokud nastane situace, kdy potřebujete nějakému uživateli dočasně umožnit provádět s úlohou operace, které by vzhledem k výši svých oprávnění nemohl provádět, můžete mu delegovat k úloze práva. Na stránce „**Uživatelé**“ se u každého uživatele nachází odkaz na stránku „**Práva**“. Delegovat smíte pouze práva k úlohám, které vlastníte nebo k nim máte sami právo *Administrace*.

Při delegování musíte vybrat úroveň oprávnění, kterou chcete uživateli svěřit. Na této úrovni závisí, jaké všechny operace bude uživatel schopen s vaší úlohou provádět. Práva jsou uvedena vzestupně a nižší práva jsou podmnožinou vyšších (tj. např. právo *Čtení* je podmnožinou práva *Odstranění*).

- *Čtení* – Uživatel úlohu vidí a může přečíst libovolné parametry, testovací soubory i veřejná vzorová řešení. Toto právo je nezbytné, aby mohl uživatel úlohu zadat do nějaké skupiny. Nicméně běžní operátoři toto právo vlastní implicitně, takže není třeba jej delegovat.
- *Omezené úpravy* – Uživatel smí odevzdávat testovací řešení, ale nemůže je zveřejnit.
- *Úpravy* – Uživatel smí upravovat veškerá nastavení úlohy a konfigurace testů a manipulovat se soubory testů. Navíc také může zveřejňovat vzorová řešení a upravovat (případně i mazat) všechna vzorová řešení ostatních uživatelů.
- *Odstranění* – Uživatel může úlohu smazat. Mazat lze ovšem jen takové úlohy, které nebyly zadány do žádné skupiny.
- *Administrace* – Uživatel může s úlohou dělat cokoli, dokonce delegovat práva jiným uživatelům.

5. Závěrem

Při vysokém počtu úloh v CodExu začne být jejich administrace poměrně náročná. Dodržujte proto interní pravidla a pokyny správce CodExu pro vytváření úloh. Velkým problémem jsou zejména duplicitní úlohy, kdy více operátorů vytvoří téměř stejnou, avšak jinak pojmenovanou úlohu. Tato redundance představuje jednak zbytečné plýtvání lidskými zdroji a jednak znepřehledňuje databázi úloh. Než vytvoříte novou úlohu, přesvědčte se, zda obdobná úloha již neexistuje. V případě, že vytváříte alternativní variantu téže úlohy, snažte se, aby měla obdobný název (např: „*Suma*“ a „*Suma (vstup ze souboru)*“).

V případě jakýchkoli potíží a dotazů kontaktujte vašeho správce CodExu.

Příloha A: Značky povolené v textu zadání

Při psaní zadání můžete používat vybrané značky jazyka XHTML. XHTML se od HTML liší v několika drobnostech, které musíte dodržovat:

- všechny značky (tagy) i názvy atributů je třeba psát malým písmem
- všechny značky jsou párové (tzn. i **<img>** nebo **
** musí mít zavírací značku), avšak je povolen zkrácený zápis **
** (tzv. samouzavírací tag)
- elementy musí být uzavřeny ve správném pořadí (zavírací značky se nesmí křížit)
- každý atribut musí mít hodnotu uzavřenou v uvozovkách

Značky a atributy, které můžete použít, jsou navíc omezeny na následující množinu:

- **
** – zalomení řádku
- **<b>** – **tučné písmo**
- **<i>** – *kurzíva*
- **<u>** – <u>podtržení</u>
- **<sub>** – dolní index
- **<sup>** – horní index
- **<code>** – ukázka `zdrojového kódu`, nebo jiný text psaný `neproporcionálním písmem`
- **<pre>** – blok textu, ve kterém se zachovávají bílé znaky
- **<a>** – hypertextový odkaz - musí mít vyplněn atribut `href` (URL odkazu) a volitelně `title` (popisek bublinkové nápovědy)
- **<img />** – Vkládá do textu obrázek. Obrázek bude umístěn vždy na střed a text ho nebude obtékat. Správná syntaxe je:

  ****.
- **<p>** – ohraničuje odstavec textu

Kromě výše uvedených tagů je možné používat i tabulky:

- **<table>** – ohraničuje tabulku, uvnitř se mohou vyskytovat pouze tagy **<tr>** a povoleny jsou tyto atributy:
  - `align` - zarovnání tabulky (povolené hodnoty `left`, `center` a `right`)
  - `border` - šířka okrajů v pixelech (většinou zadávejte 0 nebo 1)
  - `cellpadding` - vnitřní odsazení buněk (v pixelech)
  - `cellspacing` - vnější odsazení buněk (v pixelech)
- **<tr>** – ohraničuje řádek tabulky (uvnitř se mohou vyskytovat pouze tagy **<td>**)
- **<td>** – ohraničuje jednu buňku tabulky (uvnitř může být cokoliv), povolené atributy:
  - `align` – zarovnání obsahu buňky (povolené hodnoty `left`, `center` a `right`)

- `valign` – vertikální zarovnání obsahu (hodnoty top, middle, bottom)
- `colspan` – přes kolik sloupců se buňka roztahuje
- `rowspan` – přes kolik řádků se buňka roztahuje

A také seznamy:

- **`<ul>`** – seznam s odrážkami, má jeden nepovinný atribut `type`, který nastavuje vzhled odrážek: `circle` – plná kolečka, `disk` – prázdná kolečka, `square` – čtverečky
- **`<ol>`** – číslovaný seznam, má jeden nepovinný atribut `type`, který nastavuje typ číslování: 1 – arabské číslice, a – malá písmena, A – velká písmená, i – malá řecká čísla, I – velká řecká čísla
- **`<li>`** – položka číslovaného, nebo nečíslovaného seznamu

Potřebujete-li do textu vložit speciální znaky, použijte entity:

- **` `** – pevná mezera
- **`&amp;`** – ampersand "`&`"
- **`&lt;`** – menšítko "<"
- **`&gt;`** – většítko ">"
- **`&quot;`** – uvozovka "

Pokud vás zajímají detaily, zde je příslušné DTD:

```
<!ELEMENT xhtml (#PCDATA | b | i | u | sub | sup | code | pre | br | a | img | p |
table | ul | ol)*>
<!ELEMENT b (#PCDATA | i | u | sub | sup | code | br | a)*>
<!ELEMENT i (#PCDATA | b | u | sub | sup | code | br | a)*>
<!ELEMENT u (#PCDATA | b | i | sub | sup | code | br | a)*>
<!ELEMENT sub (#PCDATA | b | i | u | code | a)*>
<!ELEMENT sup (#PCDATA | b | i | u | code | a)*>
<!ELEMENT code (#PCDATA | b | i | u | sub | sup | br | a)*>
<!ELEMENT pre (#PCDATA | b | i | u | code | sub | sup | br)*>
<!ELEMENT br EMPTY>
<!ELEMENT a (#PCDATA | b | i | u | sub | sup | code | br)*>
<!ELEMENT img EMPTY>
<!ELEMENT p (#PCDATA | b | i | u | sub | sup | code | br | a)*>

<!ELEMENT table (tr+)>
<!ELEMENT tr (td+)>
<!ELEMENT td (#PCDATA | b | i | u | sub | sup | code | br | a | img | p | table)*>

<!ELEMENT ul (li+)>
<!ELEMENT ol (li+)>

<!ELEMENT li (#PCDATA | b | i | u | sub | sup | code | br | a)*>

<!ATTLIST img
      src CDATA #REQUIRED
      alt CDATA #REQUIRED>
```



```
<!ATTLIST a
      href CDATA #REQUIRED
      title CDATA #IMPLIED
      target (_blank) #IMPLIED>

<!ATTLIST table
      align (left|center|right) #IMPLIED
      border CDATA #IMPLIED
      cellpadding CDATA #IMPLIED
      cellspacing CDATA #IMPLIED>

<!ATTLIST td
      align (left|center|right) #IMPLIED
      valign (top|middle|bottom) #IMPLIED
      colspan CDATA #IMPLIED
      rowspan CDATA #IMPLIED>

<!ATTLIST ul
      type (circle|disk|square) #IMPLIED>

<!ATTLIST ol
      type (1|a|A|i|I) #IMPLIED>
```