Druhá skupina zadání projektů do předmětu Algoritmy II, letní semestr 2013/2014

doc. Mgr. Jiří Dvorský, Ph.D.

6. dubna 2014

Verze zadání

1. dubna 2014	První verze
6. dubna 2014	Oprava formátování, překlepů atd., po věcné stránce beze změny

1 Automobilový závod

Problém

Auta se zúčastní automobilového závodu. Každé auto má stav, který definuje, že buď jede předem zadanou průměrnou rychlostí, nebo stojí mimo trať. Každé auto si také pamatuje, jakou vzdálenost už ujelo. Po běžné zastávce auto pokračuje nezměněnou průměrnou rychlostí, po větší opravě se jeho průměrná rychlost trvale sníží.

Implementace

- Implementuje třídu Auto, která bude mít pět metod – Cislo, Pokracovat, Zastavit, Opravit a Vzdalenost. V konstruktoru zadejte číslo auta, průměrnou rychlost a procento snížení rychlosti po opravě. Procento bude náhodně generováno a bude to přirozené číslo menší než P (např. P = 10). Průměrná rychlost bude náhodně generována a bude větší než přirozené číslo R1 a menší než přirozené číslo R2 (např. R1 = 150 a R2 = 180). Číslo auta bude přirozené číslo.
- Metoda Pokracovat zvýší autu ujetou vzdálenost tak, aby to odpovídalo nastavené průměrné rychlosti. Metoda Zastavit neprovede nic (nezvětší se ujetá vzdálenost). Metoda Opravit nezvětší ujetou vzdálenost, ale zmenší o nastavené procento průměrnou rychlost.
- Pro simulaci závodu implementujte třídu Zavod, která bude udržovat jednotlivá auta v dynamicky alokovaném poli délky A, kde A je počet aut (např. A = 20). Třída bude mít metodu VlozitAuto, která vloží auto na první volné místo v poli, metodu Pozice, která vrátí číslo auta, které je na zadané pozici (pokud se zadá pozice 1, vrátí se auto, které vede v závodu) a metodu DalsiMinuta, která pro každé auto uložené v poli vygeneruje volání právě jedné z metod Pokracovat, Zastavit nebo Opravit. Poté provede přetřídění pole tak, aby auta byla v poli setříděna podle ujeté vzdálenosti od nejdelší po nejkratší.
- Nejdříve v programu vygenerujte A aut (např. A = 20), každé s číslem od jedničky a postupně se zvyšujícím o jedničku). Všechna auta zařaďte závodu (vložte do instance třídy Zavod).
- Pomocí cyklu simulujte jednotlivé úseky závodu s M úseky v minutách (např. M = 120). Jeden krok cyklu bude reprezentovat jeden úsek, tedy jednu minutu jízdy (volání metody DalsiMinuta).
- S čísly A, M, P, R1, R2 pracujte jako s globálními konstantami.

Na výstupu zobrazte odděleny mezerou tyto operace:

- Vygenerování nového auta ve formě `G#1[#2][#3]`, kde `#1` je číslo auta, `#2` je průměrná rychlost, `#3` je procento snížení rychlosti. Třetí auto s průměrnou rychlostí 160 a procentem snížení rychlosti 5 vytisknuto ve tvaru `G3[160][5]`.
- Zastavení auta ve formě `Z#1`, kde `#1` je číslo auta. Zastavení třetího auta bude vytisknuto ve tvaru `Z3`.
- Oprava auta ve formě `O#1`, kde `#1` je číslo auta. Oprava třetího auta bude vytisknuto ve tvaru `O3`.
- Pořadí v závodu po ukončení volání metody DalsiMinuta ve formě `P#0[#1][#2][#3]...[#A]`, kde `#1`, `#2`, `#3`, ..., `#A` jsou čísla aut. Pořadí po třetím kole v závodě s pěti auty bude vytisknuto např. ve tvaru `P3[3][1][2][4][5]`.

2 Prospěch studentů

Problém

Předpokládejme, že studenti si zapsali stejné předměty a chodí na stejné termíny zkoušek. Dále předpokládejme, že zkoušku vykonají vždy úspěšně, ale s různým hodnocením. Po ukončení zkouškového období je jim spočítán průměrný prospěch.

Implementace

- Implementuje třídu Student, která bude mít tři metody – Cislo, VlozitZnamku, PrumernyProspech. V konstruktoru zadejte číslo studenta a počet známek Z (např. Z = 6), který mu bude zadán. Pro uchování známek použijte dynamicky alokované pole délky Z. Číslo studenta bude přirozené číslo. Všichni studenti budou mít stejný počet známek.
- Pro uložení studentů implementuje třídu Seznam. Implementaci proveďte formou jednosměrného spojového seznamu pomocí ukazatelů.
- Nejdříve v programu vygenerujte N studentů (např. N = 100), každý s číslem (čísla studentů budou začínat jedničkou a postupně se budou o jedničku zvyšovat). Všechny studenty vložte do objektu třídy Seznam.
- Pomocí cyklu se Z opakováními simulujte postupné zadání známek, které budete generovat náhodně tak, aby to byla jedno z čísel 1, 2, 3 (jeden krok cyklu je jeden zkouškový termín). V každém kroku cyklu projděte celý seznam a pro každého studenta proveďte vložení jedné vygenerované známky (student si je bude postupně ukládat do svého pole).
- Po ukončení cyklu projděte celý seznam a zjistěte průměrný prospěch všech studentů.
- S čísly N, Z pracujte jako s globálními konstantami.

Na výstupu zobrazte odděleny mezerou tyto operace:

- Vygenerování nového studenta ve formě `G#1[#2]`, kde `#1` je číslo studenta, `#2` je počet známek. Třetí student s počtem zkoušek 7 bude vytisknut ve tvaru `G3[7]`.
- Vložení známky ve formě `Z#1[#2]`, kde `#1` je číslo studenta, `#2` je známka. Třetí student se známkou 2 bude vytisknut ve tvaru `Z3[2]`.
- Průměrná známka ve formě `P#1[#2]`, kde `#1` je číslo studenta, `#2` je průměrná známka. Třetí student s průměrnou známkou 1,5 bude vytisknut ve tvaru `P3[1.5]`.

3 Sledování elektroměrů

Problém

V panelovém domě je u každého bytu elektroměr, který měří spotřebu energie. Spotřeba na každém elektroměru střídavě klesá a stoupá. Jednou za čas se provádí zápis celkové spotřebované energie.

Implementace

- Implementuje třídu Elektromer, která bude mít pět metod – Cislo, ZvysitSpotrebu, SnizitSpotrebu, PonechatSpotrebu a CelkovaSpotreba. V konstruktoru zadejte výchozí naposled zapsanou hodnotu celkové spotřeby, hodnotu aktuálně spotřebovávané energie a číslo elektroměru. Výchozí celková hodnota bude náhodně generována a bude to přirozené číslo menší než H (např. H = 100). Hodnota aktuálně spotřebovávané energie bude náhodně generována a bude to přirozené číslo menší než A (např. A = 10). Číslo elektroměru bude přirozené číslo.
- Pro správu elektroměrů implementuje třídu Seznam. Implementaci proveďte formou jednosměrného spojového seznamu pomocí ukazatelů.
- Nejdříve v programu vygenerujte N elektroměrů (např. N = 20), každý s vygenerovanou výchozí hodnotou a číslem (čísla elektroměrů budou začínat jedničkou a postupně se budou o jedničku zvyšovat). Všechny elektroměry vložte do objektu třídy Seznam.
- Pomocí cyklu simulujte změny ve spotřebě elektroměrů v M časových jednotkách (např. M = 1440 – jeden den). Jeden krok cyklu bude reprezentovat jednu minutu, kdy se spotřebovává energie v nastavené hodnotě. V každém kroku cyklu projděte celý seznam a pro každý účet proveďte právě jednu z operací ZvysitSpotrebu, SnizitSpotrebu, PonechatSpotrebu, přičemž první resp. druhá z nich zvýší resp. sníží aktuální spotřebu o jedničku, třetí ponechá aktuální spotřebu. Jednotlivé elektroměry si započítají do celkové spotřeby energii za poslední minutu v kilowatthodinách (minutu vydělíme 60, dostaneme čas v hodinách a vynásobíme jej aktuální spotřebou).
- Po ukončení cyklu projděte celý seznam a zjistěte nový stav na elektroměru.
- S čísly A, H, M a N pracujte jako s globálními konstantami.

Na výstupu zobrazte odděleny mezerou tyto operace:

- Vygenerování nového elektroměru ve formě `G#1[#2][#3]`, kde `#1` je číslo elektroměru, `#2` je výchozí celková hodnota, `#3` je počáteční aktuální spotřeba. Třetí elektroměr s výchozí hodnotou 5 a počáteční aktuální spotřebou 2 bude vytisknut ve tvaru `G3[5][2]`.
- Zvýšení spotřeby ve formě `Z#1[#2]`, kde `#1` je číslo elektroměru, `#2` aktuální spotřeba před zvýšením. Třetí elektroměr s aktuální spotřebou 5 bude vytisknut ve tvaru `Z3[5]`.
- Snížení spotřeby ve formě `N#1[#2]`, kde `#1` je číslo elektroměru, `#2` aktuální spotřeba před snížením. Třetí elektroměr s aktuální spotřebou 5 bude vytisknut ve tvaru `N3[5]`.
- Celkovou spotřebu ve formě `S#1[#2]`, kde `#1` je číslo elektroměru, `#2` je celková spotřebovaná energie v kilowatthodinách. Třetí elektroměr s celkovou spotřebovanou energií 5 bude vytisknut ve tvaru `S3[5]`.

4 Správa bankovních účtů

Problém

S každým bankovním účtem je možno provádět různé typy transakcí jako je vložení, výběr, zjištění aktuálního stavu na účtu a připsání ročního úroku. Vybrat lze nejvýše částku, která odpovídá stavu na účtu. Jinak je výběr jako celek zamítnut.

Implementace

- Implementuje třídu Ucet, která bude mít pět metod – Cislo, Vlozit, Vybrat, AktualniStav a PripsatUrok. V konstruktoru zadejte procento, kterým se bude úročit, výchozí vklad a číslo účtu. Procento bude náhodně generováno a bude to přirozené číslo menší než P (např. P = 10). Vklad bude přirozené číslo menší než M (např. M = 1000). Číslo účtu bude přirozené číslo.
- Pro uložení účtů implementuje třídu Seznam. Implementaci proveďte formou jednosměrného spojového seznamu pomocí ukazatelů.
- Nejdříve v programu vygenerujte N účtů (např. N = 20), každý účet s vygenerovanou úrokovou sazbou v procentech, počátečním vkladem a číslem (čísla účtů budou začínat jedničkou a postupně se budou o jedničku zvyšovat). Všechny účty vložte do seznamu (do instance třídy Seznam).
- Pomocí cyklu simulujte provádění transakcí v Y letech (např. Y = 5). Jeden krok cyklu bude reprezentovat jeden měsíc. V každém měsíci projděte celý seznam a pro každý účet proveďte jednu z operací vložení nebo výběr s náhodně vygenerovanou částkou větší než nula a menší než M. Pokud je počet kroků dělitelný 12, pro každý účet zároveň proveďte připsání úroku.
- S čísly M, N, P, Y pracujte jako s globálními konstantami.

Na výstupu zobrazte odděleny mezerou tyto operace:

- Vygenerování nového účtu ve formě `G#1[#2][#3]`, kde `#1` je číslo účtu, `#2` je úroková sazba, `#3` je počáteční vklad. Třetí účet s úrokovou sazbou 5% a počátečním vkladem 500 bude vytisknut ve tvaru `G3[5][500]`.
- Vklad na účet ve formě `V#1[#2][#3]`, kde `#1` je číslo účtu, `#2` je stav účtu před vložením, `#3` je vkládaná částka. Třetí účet se stavem 500 a vkladem 500 bude vytisknut ve tvaru `V3[500][500]`.
- Výběr z účtu ve formě `X#1[#2][#3]`, kde `#1` je číslo účtu, `#2` je stav účtu před výběrem, `#3` je vybíraná částka. Třetí účet se stavem 1000 a výběrem 500 bude vytisknut ve tvaru `X3[1000][500]`.
- Připsání úroku na účet ve formě `U#1[#2][#3]`, kde `#1` je číslo účtu, `#2` je stav účtu před transakcí, `#3` je stav po transakci. Třetí účet se stavem 1000 a úrokovou sazbou 5 bude vytisknut ve tvaru `U3[1000][1050]`.

5 Správa půjček

Problém

S každou půjčkou je možno provádět několik typů transakcí. Může to být průběžné splácení formou pravidelných splátek, pravidelné úročení předem dohodnutou sazbou (předpokládejme, že jednou za rok), zjištění, kolik ještě zbývá zaplatit a jednorázové doplacení zbytku půjčky.

Implementace

- Implementuje třídu Pujcka, která bude mít pět metod – Cislo, PravidelnaSplatka, JednorazoveDoplaceni, AktualniStav a PripsatUrok. V konstruktoru zadejte procento, kterým se bude úročit, výchozí hodnotu půjčky, hodnotu pravidelné splátky a číslo účtu. Procento bude náhodně generováno a bude to přirozené číslo menší než P (např. P = 10). Hodnota půjčky bude přirozené číslo od H1 do H2 (např. H1 = 10000, H2 = 100000). Splátka bude přirozené číslo menší než M (např. M = 1000). Číslo účtu bude přirozené číslo.
- Pro uložení půjček implementuje třídu Seznam. Implementaci proveďte formou jednosměrného spojového seznamu pomocí ukazatelů.
- Nejdříve v programu vygenerujte N půjček (např. N = 20), každý s vygenerovanou úrokovou sazbou v procentech, počáteční hodnotou a číslem (čísla půjček budou začínat jedničkou a postupně se budou o jedničku zvyšovat). Všechny půjčky vložte seznamu (do instance třídy Seznam).
- Pomocí cyklu simulujte provádění transakcí v Y letech (např. Y = 5). Jeden krok cyklu bude reprezentovat jeden měsíc. V každém měsíci projděte celý seznam a pro každou půjčku proveďte právě jednu z operací pravidelné splátky nebo jednorázového doplacení. Pokud je počet kroků dělitelný 12, pro každou půjčku zároveň proveďte připsání úroku. Pokud zbývá při splátce na doplacení půjčky menší nebo stejná částka, jako je pravidelná splátka (poslední splátka), pak se půjčka odstraní ze seznamu. Stejně tak se půjčka odstraní ze seznamu, pokud je jednorázově doplacena.
- S čísly H1, H2, M, N, P, Y pracujte jako s globálními konstantami.

Na výstupu zobrazte odděleny mezerou tyto operace:

- Vygenerování nové půjčky ve formě `G#1[#2][#3] [#4]`, kde `#1` je číslo půjčky, `#2` je úroková sazba, `#3` je počáteční vklad, `#4` je hodnota pravidelné splátky. Třetí půjčka s úrokovou sazbou 5% a počáteční hodnotou 10000 a pravidelnou splátkou 500 bude vytisknuta ve tvaru `G3[5][10000][500]`.
- Pravidelná splátka půjčky ve formě `S#1[#2] [#3]`, kde `#1` je číslo půjčky, `#2` je stav půjčky před doplacením, `#3` je splátka. Třetí půjčka se stavem 5000 a pravidelnou splátkou 1000 bude vytisknuta ve tvaru `D3[5000][1000]`.
- Poslední splátka (doplacení půjčky) půjčky ve formě `O#1[#2][#3]`, kde `#1` je číslo půjčky, `#2` je stav půjčky před splátkou, `#3` je splátka. Třetí půjčka se stavem 500 a pravidelnou splátkou 1000 bude vytisknuta ve tvaru `O3[500][1000]`.
- Jednorázové doplacení půjčky ve formě `D#1[#2]`, kde `#1` je číslo půjčky, `#2` je stav půjčky před doplacením. Třetí půjčka se stavem 5000 vytisknuta ve tvaru `D3[5000]`.

6 Vyložení vlaků

Problém

Nákladní vlak, který přijíždí na nádraží, veze vagony s různou cenou nákladu. Předpokládejme, že náklad vagonu se skládá z určitého počtu položek, z nichž každá má pevnou cenu. Vagony se rozdělí podle ceny nákladu na vedlejší koleje, ze kterých jsou poté postupně vyloženy. Doba vyložení vagonu je úměrná počtu položek nákladu.

Implementace

- Implementuje třídu Vagon, která bude mít čtyři metody – Cislo, PocetPolozek, CenaNakladu, VylozitPolozku. V konstruktoru zadejte číslo vagonu, počet položek a pevnou cenu jedné položky. Počet položek bude náhodně generován a bude to přirozené číslo mezi 1 a P (např. P = 10). Pevná cena jedné položky bude od C1 do C2 (např. C1 = 100, C2 = 1000). Maximální možná cena nákladu je tedy P∗C2. Číslo účtu bude přirozené číslo.
- Implementujte třídu Zasobnik pro objekty třídy Vagon pomocí ukazatelů. V programu budete mít celkem čtyři zásobníky. První bude reprezentovat přijíždějící vlak. Do druhého zásobníku se budou řadit vagony s cenou nákladu nejvýše jedna třetina maximální ceny nákladu, do třetího zásobníku vagonu s cenou nejvýše dvě třetiny a do čtvrtého zásobníku všechny ostatní vagony.
- Pomocí cyklu nejdříve vygenerujte vlak (zásobník), do kterého postupně generujte N vagonů (např. N = 10). Čísla vagonů budou začínat jedničkou a postupně se budou o jedničku zvyšovat.
- Pomocí cyklu simulujte rozdělení vagonů na vedlejší koleje. Jeden krok cyklu bude reprezentovat odebrání jednoho vagonu z vlaku a jeho zařazení do správného zásobníku.
- Pomocí cyklu simulujte vyložení všech vagonů. Jeden krok cyklu bude reprezentovat vyložení jedné položky z každého vagonu na vrcholu zásobníku. Pokud nebude vagon obsahovat žádnou položku, bude odebrán ze zásobníku a připojen zpět k vlaku. Cyklus ukončete, až budou všechny zásobníky prázdné.
- S čísly C1, C2, N, P pracujte jako s globálními konstantami.

Na výstupu zobrazte odděleny mezerou tyto operace:

- Vygenerování nového vagonu ve formě `G#1[#2][#3]`, kde `#1` je číslo vagonu, `#2` je počet položek, `#3` je pevná cena jedné položky. Třetí vagon s pěti pokožkami a cenou jedné položky 500 bude vytisknut ve tvaru `G3[5][500]`.
- Zařazení vagonu do zásobníku `Z#1[#2]`, kde `#1` je číslo vagonu, `#2` číslo zásobníku (1, 2, resp. 3). Třetí vagon ve třetí frontě bude vytisknut ve tvaru `Z3[3]`.
- Vyložení jedné položky z vagonu `V#1[#2]`, kde `#1` je číslo vagonu, `#2` počet zbývajících položek. Vyložení jedné položky z třetího vagonu, kterému zbývalo pět položek, bude vytisknuto ve tvaru `V3[4]`.
- Odstranění vagonu ze zásobníku `X#1[#2]`, kde `#1` je číslo vagonu, `#2` číslo zásobníku. Třetí vagon ve třetím zásobníku bude vytisknut ve tvaru `X3[3]`.

7 Výplaty zaměstnanců

Problém

Zaměstnanci mají různé platy s různým způsobem zdanění. Každý měsíc dostávají svou hrubou mzdu, která je zdaněna. Pro zjednodušení předpokládejme, že plat do 10000 Kč není daněn vůbec, rozdíl do platu do 20000 Kč je daněn 10%, rozdíl do 30000 je daněn 15% a zbytek nad 30000 Kč je daněn 20% (při mzdě 50000 je tedy 10000 daněno 10%, 10000 daněno 15% a 20000 daněno 20%). Každý zaměstnanec má předem dohodnutou a neměnící se mzdu a může jednou za čas dostat jednorázovou odměnu (ta se před zdaněním připočítá ke hrubé mzdě).

Implementace

- Implementuje třídu Zamestnanec, která bude mít pět metod – Cislo, VyplatitMzdu, VyplatitOdmenu, PrepocetNaCistouMzdu, PosledniCistaMzda. V konstruktoru zadejte číslo zaměstnance a jeho pevnou hrubou mzdu. Číslo zaměstnance bude přirozené číslo. Hrubá mzda bude přirozené číslo, a to nejméně 10000 a nejvýše H (např. H=50000).
- Pro uložení zaměstnanců implementuje třídu Seznam. Implementaci proveďte formou jednosměrného spojového seznamu pomocí ukazatelů.
- Nejdříve v programu vygenerujte N zaměstnanců (např. N = 20), každý s vygenerovanou pevnou mzdou a číslem (čísla zaměstnanců budou začínat jedničkou a postupně se budou o jedničku zvyšovat). Všechny zaměstnance vložte do seznamu (do instance třídy Seznam).
- Pomocí cyklu simulujte provádění výplat v M měsících (např. M = 20). Jeden krok cyklu bude reprezentovat jeden měsíc. V každém měsíci projděte celý seznam a pro každého zaměstnance proveďte výplatu mzdy, pro některé náhodně vygenerujte odměnu ve výši mezi O1 a O2 (např. O1 = 500, O2 = 10000), a poté pro všechny proveďte přepočet na čistou mzdu.
- Každý zaměstnanec si bude pamatovat poslední hrubou i čistou mzdu včetně odměny.
- S čísly H, M, N, O1, O2 pracujte jako s globálními konstantami.

Na výstupu zobrazte odděleny mezerou tyto operace:

- Vygenerování nového zaměstnance ve formě `G#1[#2]`, kde `#1` je číslo zaměstnance, `#2` je pevná hrubá mzda. Třetí zaměstnanec s pevnou hrubou mzdou 15000 bude vytisknut ve tvaru `G3[15000]`.
- Vyplacení odměny ve formě `O#1[#2]`, kde `#1` je číslo účtu, `#2` je výše odměny. Třetí zaměstnanec s odměnou 2000 bude vytisknut ve tvaru `O3[2000]`.
- Přepočet na čistou mzdu včetně odměny z účtu ve formě `M#1[#2][#3]`, kde `#1`je číslo účtu, `#2` je hrubá mzda včetně odměny, `#3` je čistá mzda včetně odměny. Třetí účet s pevnou hrubou mzdou 15000 a odměnou 2000 bude vytisknut ve tvaru `M3[17000][16300]`.

8 Vyskladnění aut

Problém

Nákladní auto, které přijíždí ke skladišti, se řadí k vyskladnění do fronty v závislosti na celkové ceně nákladu. Předpokládejme, že náklad se skládá z určitého počtu položek, z nichž každá má pevnou cenu. Doba vyskladnění auta je úměrná počtu položek nákladu.

Implementace

- Implementuje třídu NakladniAuto, která bude mít čtyři metody – Cislo, PocetPolozek, CenaNakladu, VyskladnitPolozku. V konstruktoru zadejte číslo auta, počet položek a pevnou cenu jedné položky. Počet položek bude náhodně generován a bude to přirozené číslo mezi 1 a P (např. P = 10). Pevná cena jedné položky bude od C1 do C2 (např. C1 = 100, C2 = 1000). Maximální možná cena nákladu je tedy P∗C2. Číslo auta bude přirozené číslo.
- Implementujte třídu Fronta pro objekty třídy NakladniAuto pomocí ukazatelů. V programu budete mít celkem tři fronty. Do první fronty se budou řadit auta s cenou nákladu nejvýše do jedné třetiny maximální ceny nákladu, do druhé fronty auta s cenou nejvýše do dvou třetin a do třetí fronty všechna ostatní auta.
- Pomocí cyklu simulujte vyskladnění N nákladních aut (např. N = 50). Čísla aut budou začínat jedničkou a postupně se budou o jedničku zvyšovat. Jeden krok cyklu bude reprezentovat příjezd (vygenerování) jednoho auta a jeho zařazení do správné fronty, a také vyskladnění jedné položky z každého auta na začátku fronty. Pokud nebude auto obsahovat žádnou položku, bude odebráno z fronty. Cyklus ukončete, až budou všechny fronty prázdné.
- S čísly C1, C2, N, P pracujte jako s globálními konstantami.

Na výstupu zobrazte odděleny mezerou tyto operace:

- Vygenerování nového auta ve formě `G#1[#2][#3]`, kde `#1` je číslo auta, `#2` je počet položek, `#3` je pevná cena jedné položky. Třetí auto s pěti pokožkami a cenou jedné položky 500 bude vytisknuto ve tvaru `G3[5][500]`.
- Zařazení auta do fronty `F#1[#2]`, kde `#1` je číslo auta, `#2` číslo fronty. Třetí auto ve třetí frontě bude vytisknuto ve tvaru `F3[3]`.
- Vyskladnění jedné položky z auta `V#1[#2]`, kde `#1` je číslo auta, `#2` počet zbývajících položek. Vyskladnění jedné položky z třetího auta, kterému zbývá pět položek, bude vytisknuto ve tvaru `V3[4]`.
- Odstranění auta z fronty `X#1[#2]`, kde `#1` je číslo auta, `#2` číslo fronty. Třetí auto ve třetí frontě bude vytisknuto ve tvaru `X3[3]`.