Projekty

1. Simulace třídění balíků

Problém

Do balíkového třídícího stroje dopravuje N (např. N=5) vstupních pásů balíky. Každý z pásů je označen číslem 0 až N-1. Každý balík má své identifikační číslo, směrovací číslo adresáta a hmotnost. Na počátku činnosti stroje je na pásy náhodně vloženo celkem K (např. K=1000) balíků. Třídící stroj cyklicky odebírá z jednotlivých pásů balíky a ukládá je kontejnerů podle směrovacích čísel adresáta. Zpracování jednoho balíku trvá jednu časovou jednotku. Ke třídícímu stroji je na M přípojných míst připojeno M (např. M=10) kontejnerů. Přípojná místa jsou označena čísly 0 až M-1. Do kontejneru u přípojného místa s číslem m (m je z intervalu 0 až M-1) patří balíky, jejichž identifikační číslo dává po dělení číslem M zbytek m (opraveno 17.3.). Všechny kontejnery jsou označeny svým pořadovým číslem a mají shodnou maximální povolenou nosnost L. Pokud by při ukládání balíku do kontejneru došlo k překročení maximální povolené nosnosti, je nutné kontejner odeslat k expedici a začít skládat balíky do nového kontejneru. Na konci třídění je nutné odeslat k expedici všechny kontejnery, i když nejsou zcela zaplněné. Simulujte průběh třídění balíků.

Implementace

- Čísla N, M, K a L deklarujte jako globální konstanty.
- Pás s balíky reprezentujte jako frontu implementovanou pomocí ukazatelů.
- N pásů reprezentujte jako pole obsahující N front.
- Balík realizujte jako strukturu obsahující identifikační číslo (int), směrovací číslo (int) a hmotnost (int). Všechny tři údaje jsou kladná čísla. Hmotnost musí být větší než nula a zároveň menší než nosnost kontejneru L. Identifikační čísla balíků budou postupně 1, 2, 3, až K. Směrovací číslo a hmotnost generujte náhodně.
- Průběh třídění implementujte cyklem while, který bude ukončen v okamžiku, kdy budou všechny pásy prázdné. V každém kroku cyklu (jedné časové jednotce) je z pásu odebrán jeden balík a zařazen do správného kontejneru. Pokud by byl kontejner přeplněn, odešle se kontejner k expedici a balík se vloží až do nového kontejneru.
- Třídící stroj odebírá balíky z pásů střídavě, tj. nejprve odebere jeden balík z pásu číslo 0, pak jeden balík z pásu číslo 1 atd. Nakonec odebere jeden balík z pásu číslo N-1 a vrací se na začátek k pásu 0. Již vyprázdněné pásy jsou přeskakovány.
- Kontejner reprezentujte jako strukturu složenou z pořadového čísla kontejneru (int) a celkové hmotnosti balíků v něm uložených (int). Kontejnery číslujte průběžně od 1. Balíky uložené v kontejneru není nutné uchovávat, stačí jen součet jejich hmotností.

Na výstupu zobrazte odděleny mezerou tyto operace:

- Vložení nového balíku na pás ve formě B#1[#2][#3][#4], kde #1 je číslo pásu, #2 je identifikační číslo balíku, #3 je směrovací číslo adresáta a #4 je hmotnost balíku. Například balík vložený na pás 2 s identifikačním číslem 153, směrovacím číslem adresáta 730 a hmotností 3 bude vytisknut ve formě B2[153][730][3].

- Připojení nového kontejneru ve formě P#1[#2], kde #1 je číslo přípojného místa třídícího stroje a #2 je pořadové číslo kontejneru. Připojení celkově patnáctého kontejneru ke druhému přípojnému místu bude vytisknuto ve tvaru P2[15].
- Odebrání balíku z pásu ve formě R#1[#2], kde #1 je číslo pásu a #2 je identifikační číslo odebraného balíku. Odebrání balíku s identifikačním číslem 314 z prvního pásu bude vytisknuto ve tvaru R1[314].
- Uložení balíku do kontejneru ve formě U#1[#2][#3][#4][#5], kde #1 je číslo přípojného místa třídícího stroje, #2 je identifikační číslo balíku, #3 je směrovací číslo adresáta, #4 je hmotnost balíku a #5 je celková hmotnost balíků v daném kontejneru po vložení balíku #2. Uložení balíku s identifikačním číslem 153, směrovacím číslem adresáta 730 a hmotností 3 do kontejneru na přípojném místě 0 a s celkovou hmotností 415 bude vytisknut ve formě U0[153][730][3][415].
- Expedice kontejneru ve formě E#1[#2][#3], kde #1 číslo přípojného místa, #2 je pořadové číslo kontejneru a #3 je celková hmotnost balíků v kontejneru.

2. Simulace obchodu 1

Problém

V obchodě je N pokladen (např. N = 5), u každé z nich je fronta pro libovolné množství zákazníků. Zákazník, který si nakoupil, se staví do fronty, ve které před ním bude nejméně zákazníků. Doba strávená u pokladny je závislá na množství nakoupeného zboží, kterého bude nejméně jedna položka, nejvýše P položek (např. P = 10). Zpracování jedné položky nákupu u pokladny trvá jednu časovou jednotku. Pravidelně přicházejí noví zákazníci k pokladnám (v jedné časové jednotce právě jeden zákazník). Simulujte průběh zpracování front u pokladen pro M zákazníků obchodu (např. M = 1000).

Implementace

- Čísla N, M, P deklarujte jako globální konstanty.
- Každou pokladnu reprezentujte frontou implementovanou pomocí ukazatelů. Fronta si bude pamatovat, kolik prvků obsahuje a bude umět poskytnout první prvek. Nepoužívejte třídy (fronta bude reprezentována jako struktura).
- N pokladen reprezentujte polem, prvky pole budou fronty.
- Prvek fronty implementujte jako strukturu, která bude obsahovat identifikaci zákazníka (číslo typu int) a počet položek nákupu v košíku (číslo typu int).
- Průběh zpracování front u pokladen reprezentujte cyklem while, který bude ukončen, pokud budou všechny fronty prázdné. V každém kroku cyklu (jedné časové jednotce) bude všem zákazníkům, kteří jsou na řadě (první ve frontě), buď odebrána z košíku jedna položka nákupu, nebo pokud nebude v košíku žádná položka (nelze odebrat), bude zákazník odebrán z fronty. Pokud celkový počet zákazníků vygenerovaných během simulace nedosáhl M, bude vygenerován další zákazník s náhodným počtem položek nákupu v košíku (od 1 do P). Identifikátorem zákazníka je pořadí, ve kterém byl vygenerován (první zákazník bude mít identifikátor 1, poslední M). Zákazník se poté se zařadí do fronty s nejmenším počtem prvků (zákazníků). Pokud je takových front více, pak si vybere libovolnou z nich.

Na výstupu zobrazte odděleny mezerou tyto operace:

- Vygenerování nového zákazníka ve formě G#1[#2], kde #1 je identifikace zákazníka a #2 je počet položek nákupu v košíku. Třetí zákazník s šesti položkami bude vytisknut ve tvaru G3[6].
- Vložení zákazníka do fronty ve formě V#1[F#2][#3], kde #1 je identifikace zákazníka, #2 je index fronty v poli a #3 je počet zákazníku ve frontě včetně něj. Pokud bude vložen třetí zákazník do první fronty se dvěma předchozími zákazníky, bude na výstupu V3[F0][3].
- Odebrání jedné položky nákupu z košíku ve formě K#1[#2], kde #1 je identifikace zákazníka a #2 je počet zbývajících položek nákupu v košíku. Pokud odebereme třetímu zákazníkovi jednu položku nákupu z košíku s šesti položkami, bude na výstupu K3[5].
- Odebrání zákazníka z fronty ve tvaru O#1[F#2][#3], kde #1 je identifikace zákazníka, #2 je index fronty v poli a #3 je počet zákazníků ve frontě po odebrání. Pokud bude odebrán třetí zákazník z první fronty, ve které zbude jediný další, bude na výstupu O3[F0][1].

3. Simulace vlakové vlečky 1

Problém

Na malém nákladním nádraží je jedna příjezdová kolej, na které stojí vlak s M vagony (např. M = 50). Každý vagon je označen některým z čísel 0 až N-1, N < M (např. N = 4). Dále je na nádraží N slepých kolejí, které jsou také označeny čísly 0 až N-1. Lokomotiva je na opačné straně, než jsou nájezdy na slepé koleje. Slepé koleje slouží k dočasnému „odložení“ vagonů vlaku, který je na příjezdové koleji (a předpokládejme, že jsou dostatečně dlouhé). Úkolem je přeskládat vagony vlaku tak, aby byly uspořádány podle čísel. Tedy tak, aby za lokomotivou byly nejdříve vagony označené číslem 0, poté číslem 1, …, a nakonec s čísly N-1. Na dílčím uspořádání vagonů se stejnými čísly nezáleží. Principem je, že lokomotiva může opakovaně zatlačit celý vlak na jednu ze slepých kolejí, tam odpojit jeden vagon, a poté se vrátit na příjezdovou kolej.

Implementace

- Čísla N, M deklarujte jako globální konstanty.
- Vlak reprezentujte jako zásobník implementovaný pomocí ukazatelů, kde první vagon za lokomotivou bude na dně zásobníku a poslední vagon na vrcholu zásobníku.
- Každou slepou kolej reprezentujte zásobníkem implementovaným pomocí ukazatelů. Nepoužívejte třídy (zásobníky budou reprezentovány jako struktury).
- N slepých kolejí reprezentujte polem délky N, prvky pole budou zásobníky.
- Prvek zásobníku implementujte jako strukturu, která bude obsahovat identifikaci vagonu (číslo typu int) a číslo typu int reprezentující, na kterou slepou kolej bude dočasně „odložen“.
- Vytvoření vlaku na příjezdové koleji (s M vagony) reprezentujte cyklem for. V každém kroku cyklu bude vygenerován jeden vagon s náhodným číslem slepé koleje (od 0 do N-1). Identifikátorem vagonu je pořadí, ve kterém byl vygenerován (první vagon bude mít identifikátor 1, poslední M). Průběh „odložení“ všech vagonů na slepé koleje reprezentujte cyklem while. Zpětné připojení vagonů k lokomotivě reprezentujte cykly for a while.

Na výstupu zobrazte odděleny mezerou tyto operace:

- Vygenerování nového vagonu ve formě G#1[K#2], kde #1 je identifikace vagonu a #2 je číslo slepé koleje. Třetí vagon s označením druhé slepé koleje bude vytisknut ve tvaru G3[K1].
- „Odložení“ vagonu na slepou kolej ve formě O#1[K#2], kde #1 je identifikace vagonu a #2 je index slepé koleje (zásobníku) v poli. Pokud bude „odložen“ třetí vagon na druhou slepou kolej, bude na výstupu O3[K1]
- Připojení vagonu na slepé koleji zpět k vlaku ve formě P#1[K#2], kde #1 je identifikace vagonu a #2 je index slepé koleje (zásobníku) v poli. Pokud bude připojen třetí vagon ze druhé slepé koleje, bude na výstupu P3[K1].

4. Simulace pošty 1

Problém

Na poště je N přepážek (např. N = 5), každá z nich je určena pro jiný typ transakcí. U každé přepážky je fronta pro libovolné množství zákazníků. Zákazník, který přijde na poštu, si zvolí typ transakce a postaví se do fronty u odpovídající přepážky. Doba strávená u přepážky je závislá na tom, kolik transakcí daného typu potřebuje zákazník realizovat - nejméně jednu transakci, nejvýše P transakcí (např. P = 3). Zpracování jedné transakce bez ohledu na její typ trvá jednu časovou jednotku. Pravidelně přicházejí noví zákazníci k překážkám (v jedné časové jednotce přijde na poštu právě jeden zákazník). Simulujte průběh zpracování front u přepážek pro M zákazníků pošty (např. M = 100).

Implementace

- Čísla N, M, P deklarujte jako globální konstanty.
- Každou přepážku reprezentujte frontou implementovanou pomocí ukazatelů. Fronta bude umět poskytnout první prvek. Nepoužívejte třídy (fronta bude reprezentována jako struktura).
- N přepážek reprezentujte polem délky N, prvky pole budou fronty.
- Prvek fronty implementujte jako strukturu, která bude obsahovat identifikaci zákazníka (číslo typu int) a počet požadovaných transakcí (číslo typu int).
- Průběh zpracování front u přepážek reprezentujte cyklem while, který bude ukončen, pokud budou všechny fronty prázdné. V každém kroku cyklu (jedné časové jednotce) bude všem zákazníkům, kteří jsou na řadě (první ve frontě), buď zpracována jedna transakce, nebo pokud nebude žádná transakce požadována (není co zpracovat), bude zákazník odebrán z fronty. Pokud celkový počet zákazníků vygenerovaných během simulace nedosáhl M, bude vygenerován další zákazník s náhodně vygenerovaným typem transakce (0 až N-1 tak, aby to odpovídalo některému indexu fronty v poli) a s náhodným počtem požadovaných transakcí (od 1 do P). Identifikátorem zákazníka je pořadí, ve kterém byl vygenerován (první zákazník bude mít identifikátor 1, poslední M). Zákazník se poté se zařadí do fronty pro odpovídající typ transakce.

Na výstupu zobrazte odděleny mezerou tyto operace:

- Vygenerování nového zákazníka ve formě G#1[F#2][#3], kde #1 je identifikace zákazníka a #2 je typ transakce (index fronty v poli) a #3 je počet požadovaných transakcí. Třetí zákazník pro druhý typ transakce se třemi požadavky bude vytisknut ve tvaru G3[F1][3].
- Vložení zákazníka do fronty ve formě V#1[F#2], kde #1 je identifikace zákazníka, #2 je index fronty v poli. Pokud bude vložen třetí zákazník do druhé fronty, bude na výstupu V3[F1].
- Zpracování jedné transakce zákazníka ve formě T#1[#2], kde #1 je identifikace zákazníka a #2 je počet zbývajících transakcí. Pokud zpracujeme jednu transakci třetímu zákazníkovi, bude na výstupu K3[2].
- Odebrání zákazníka z fronty ve tvaru O#1[F#2], kde #1 je identifikace zákazníka, #2 je index fronty v poli. Pokud bude odebrán třetí zákazník z druhé fronty, bude na výstupu O3[F1].

5. Simulace obchodu 2

Problém

V obchodě jsou pokladny (předpokládejme, že jejich počet není omezen), u každé z nich může být fronta maximálně pro N zákazníků (např. N = 7). Všechny pokladny jsou na začátku uzavřeny. Zákazníci, kteří si nakoupí, jdou k pokladnám. Pokud není k dispozici pokladna, kam by se postavili do fronty (není žádná otevřená nebo jsou všechny už otevřené plné), je otevřena další pokladna. Doba strávená u pokladny je pro všechny zákazníky stejná a trvá jednu časovou jednotku. Pravidelně přicházejí noví zákazníci k pokladnám v počtu nejvýše Z v jedné časové jednotce (např. Z = 10) a staví se do front u pokladen tak, že nejdříve zaplní první frontu, pak další, … Pokud u některé pokladny není zákazník, pokladna je opět je uzavřena. Simulujte průběh zpracování front u pokladen pro M zákazníků obchodu (např. M = 1000).

Implementace

- Čísla N, M, Z deklarujte jako globální konstanty.
- Každou pokladnu reprezentujte frontou implementovanou v poli. Nepoužívejte třídy (fronta bude reprezentována jako struktura).
- Otevřené pokladny reprezentujte jako jednosměrný seznam implementovaný pomocí ukazatelů, jehož prvky jsou fronty. Otevření pokladny implementujte jako vložení nové fronty na konec seznamu. Uzavření pokladny implementujte jako odebrání prvku (fronty) ze seznamu.
- Prvek fronty implementujte jako strukturu, která bude obsahovat identifikaci zákazníka (číslo typu int).
- Průběh zpracování front u pokladen reprezentujte cyklem while, který bude ukončen, pokud budou všechny pokladny uzavřené (fronty budou prázdné a odebrané ze seznamu). V každém kroku cyklu (jedné časové jednotce) bude všem otevřeným pokladnám (frontám) buď odebrán jeden zákazník, nebo pokud bude fronta prázdná, bude odebrána ze seznamu. Pokud celkový počet zákazníků vygenerovaných během simulace nedosáhl M, bude náhodně vygenerována další skupina zákazníků (nejméně 1, nejvýše Z, celkem nejvýše M). Identifikátorem zákazníka je pořadí, ve kterém byl vygenerován (první zákazník bude mít identifikátor 1, poslední M). Zákazníci se poté se zařadí do fronty tak, že nejdříve zaplní první frontu, pak další, … Pokud není místo v žádné z front v seznamu, vytvoří se na konci seznamu další fronta.

Na výstupu zobrazte odděleny mezerou tyto operace:

- Vygenerování nového zákazníka ve formě G#1, kde #1 je identifikace. Třetí zákazník vytisknut ve tvaru G3.
- Otevření nové pokladny (vložení nové fronty na konec seznamu) ve formě F#1, kde #1 je pořadí fronty v seznamu počítáno od jedničky.
- Vložení zákazníka do fronty ve formě V#1[F#2], kde #1 je identifikace zákazníka, #2 je pořadí fronty v seznamu počítáno od jedničky. Pokud bude vložen třetí zákazník do první fronty v seznamu, bude na výstupu V3[F1].
- Odebrání zákazníka z fronty ve tvaru O#1[F#2], kde #1 je identifikace zákazníka, #2 je pořadí fronty v seznamu počítáno od jedničky. Pokud bude odebrán třetí zákazník z první fronty, bude na výstupu O3[F1].
- Uzavření pokladny (odebrání fronty ze seznamu) ve formě U#1, kde #1 je pořadí fronty v seznamu počítáno od jedničky

6. Simulace vlakové vlečky 2

Problém

Na malém nákladním nádraží je jedna příjezdová kolej, na které stojí vlak s M vagony (např. M = 50). Dále je na nádraží několik slepých kolejí (předpokládejme, že jejich počet není omezen. Lokomotiva je na opačné straně, než jsou nájezdy na slepé koleje. Slepé koleje slouží k dočasnému „odložení“ vagonů vlaku, který je na příjezdové koleji. Na každou slepou kolej lze „odložit“ nejvýše N vagonů (např. N = 7). Úkolem je „odložit“ dočasně všechny vagony vlaku na co nejmenší počet slepých kolejí. Poté je po nějakém čase potřeba vagony opět připojit, a to ve stejném pořadí, v jakém byly za lokomotivou původně. Principem je, že lokomotiva může opakovaně zatlačit celý vlak na jednu ze slepých kolejí, tam odpojit jeden vagon, a poté se vrátit na příjezdovou kolej.

Implementace

- Čísla N, M deklarujte jako globální konstanty.
- Vlak reprezentujte jako zásobník implementovaný pomocí pole pro M vagonů, kde první vagon za lokomotivou bude na dně zásobníku a poslední vagon na vrcholu zásobníku.
- Každou slepou kolej reprezentujte zásobníkem implementovaným pomocí pole pro N vagonů. Nepoužívejte třídy (zásobníky budou reprezentovány jako struktury).
- Zásobník si bude pamatovat pořadí, ve kterém vznikl. Zásobník reprezentující vlak bude mít pořadí 0, slepé koleje pak pořadí od 1.
- Použité slepé koleje reprezentujte jako jednosměrný seznam implementovaný pomocí ukazatelů, jehož prvky jsou zásobníky. Potřebu nové slepé koleje implementujte jako vložení nového zásobníku na začátek seznamu. Připojení posledního vagonu ze slepé koleje implementujte společně s odebráním slepé koleje (zásobníku) ze seznamu.
- Prvek zásobníku implementujte jako číslo, které bude reprezentovat identifikaci vagonu (číslo typu int).
- Vytvoření vlaku na příjezdové koleji (s M vagony) reprezentujte cyklem for. V každém kroku cyklu bude vygenerován jeden vagon. Identifikátorem vagonu je pořadí, ve kterém byl vygenerován (první vagon bude mít identifikátor 1, poslední M). Průběh „odložení“ všech vagonů na slepé koleje reprezentujte cyklem while. Zpětné připojení vagonů k lokomotivě reprezentujte cykly while.

Na výstupu zobrazte odděleny mezerou tyto operace:

- Vygenerování nového vagonu ve formě G#1, kde #1 je identifikace vagonu. Třetí vagon bude vytisknut ve tvaru G3.
- „Odložení“ vagonu na slepou kolej O#1[K#2], kde #1 je identifikace vagonu a #2 je pořadí slepé koleje (zásobníku). Pokud bude „odložen“ třetí vagon na druhou slepou kolej, bude na výstupu O3[K2]
- Připojení vagonu na slepé koleji zpět k vlaku ve formě P#1[K#2], kde #1 je identifikace vagonu a #2 je pořadí slepé koleje (zásobníku). Pokud bude připojen třetí vagon z druhé slepé koleje, bude na výstupu P3[K2].

7. Simulace autoservisu 1

Problém

V autoservisu plánují opravy dopředu na N pracovních dnů (např. N = 5). Každý den je možno naplánovat opravy maximálně pro Z zákazníků (např. Z = 7). Každý zákazník, který přišel do servisu, požaduje opravu na konkrétní den (nejdříve den 0, nejpozději den N - 1). Jednotlivé dny jsou plánovány jako fronta objednaných oprav. Pokud zákazník požaduje opravu na den, kdy fronta není plná, je oprava naplánována do odpovídající fronty. Pokud je fronta na požadovaný den plná, je zákazník odmítnut. Zpracování opravy trvá jednu časovou jednotku. Simulujte průběh plánování oprav, a poté jejich realizaci.

Implementace

- Čísla N, Z deklarujte jako globální konstanty.
- Každý den reprezentujte frontou implementovanou pomocí pole pro Z oprav. Nepoužívejte třídy (fronta bude reprezentována jako struktura).
- N dnů reprezentujte polem, prvky pole budou fronty.
- Prvek fronty (opravu požadovanou zákazníkem) implementujte jako strukturu, která bude obsahovat identifikaci požadavku na opravu (číslo typu int) a den, na který je oprava plánována (číslo typu int).
- Průběh plánování dnů (front) reprezentujte cyklem while, který bude ukončen, pokud budou všechny fronty naplněny. V každém kroku cyklu (jedné časové jednotce) bude vygenerován další požadavek na opravu s náhodně vybraným dnem (0 až N - 1). Identifikátorem požadavku je pořadí, ve kterém byl vygenerován (první požadavek bude mít identifikátor 1, poslední záleží na počtu odmítnutých zákazníků). Pokud je fronta pro požadovaný den opravy plná, je požadavek odmítnut, jinak je zařazen do fronty. Realizaci oprav reprezentujte cykly for a while, pomocí nichž postupně vyprázdníte po dnech všechny fronty s opravami.

Na výstupu zobrazte odděleny mezerou tyto operace:

- Vygenerování nového požadavku na opravu ve formě G#1[#2], kde #1 je identifikace požadavku a #2 je požadovaný den opravy. Desátý požadavek na druhý den bude vytisknut ve tvaru G10[1].
- Vložení požadavku do fronty ve formě V#1[F#2], kde #1 je identifikace požadavku, #2 je požadovaný den opravy. Pokud bude vložen desátý požadavek do fronty pro druhý den, bude na výstupu V10[F1].
- Odmítnutí požadavku X#1[F#2], kde #1 je identifikace požadavku, #2 je požadovaný den opravy. Pokud bude odmítnut desátý zákazník z fronty pro druhý den, bude na výstupu X10[F1].
- Realizace jednoho požadavku (odebrání z fronty) ve formě O#1[F#2], kde #1 je identifikace požadavku a #2 den opravy. Pokud realizujeme desátý požadavek ve frontě pro druhý den, bude na výstupu O10[1].

8. Simulace autoservisu 2

Problém

V autoservisu poskytují opravy v N různých provozech (např. N = 5). Každý požadavek na opravu v příslušném provozu je zařazen do seznamu na konec. Nicméně se může stát, že vznikne potřeba mimořádné opravy, v tom případě je požadavek na opravu zařazen na začátek seznamu a oprava je realizována neprodleně. Realizace opravy trvá jednu časovou jednotku. Simulujte průběh realizace oprav pro M požadavků (např. M = 500). Předpokládejte, že v jedné časové jednotce může vzniknout až P nových požadavků (např. P = 7).

Implementace

- Čísla N, P deklarujte jako globální konstanty.
- Každý provoz reprezentujte jednosměrným seznamem implementovaným pomocí ukazatelů. Nepoužívejte třídy (seznam bude reprezentován jako struktura).
- N provozů reprezentujte polem, prvky pole budou seznamy.
- Prvek seznamu (požadavek na opravu) implementujte jako strukturu, která bude obsahovat identifikaci požadavku na opravu (číslo typu int), provoz, pro který je oprava plánována (číslo typu int) a potřebu zařazení mimo pořadí (hodnota typu bool)
- Průběh realizace oprav reprezentujte cyklem while, který bude ukončen, pokud budou všechny seznamy prázdné. V každém kroku cyklu (jedné časové jednotce) bude všem provozům (seznamům), pokud nebudou prázdné, odebrán jeden požadavek na opravu. Pokud celkový počet požadavků vygenerovaných během simulace nedosáhl M, bude náhodně vygenerována další skupina požadavků (nejméně 1, nejvýše P, celkem nejvýše M). Identifikátorem požadavku na opravu je pořadí, ve kterém byl vygenerován (první požadavek bude mít identifikátor 1, poslední M). Dále se náhodně vygeneruje provoz, pro který je oprava určena (0 až N- 1), a náhodně se vygeneruje potřeba zařazení mimo pořadí. Poté se nově vygenerované požadavky na opravy zařadí správně do seznamů reprezentujících provozy (mimořádné na začátek a ostatní na konec).

Na výstupu zobrazte odděleny mezerou tyto operace:

- Vygenerování nového požadavku na opravu ve formě G#1[S#2][#3], kde #1 je identifikace požadavku, #2 je požadovaný provoz a #3 je potřeba zařazení mimo pořadí. Normální desátý požadavek na druhý provoz bude vytisknut ve tvaru G10[S1][0] (mimo pořadí G10[S1][1]).
- Vložení požadavku do seznamu ve formě V#1[S#2], kde #1 je identifikace požadavku, #2 je požadovaný provoz a #3 je potřeba mimo pořadí. Pokud bude vložen desátý požadavek do seznamu pro druhý provoz v normálním pořadí, bude na výstupu V10[S1][0].
- Realizace jednoho požadavku (odebrání ze seznamu) ve formě O#1[S#2], kde #1 je identifikace požadavku a #2 provoz. Pokud realizujeme desátý požadavek v seznamu pro druhý provoz, bude na výstupu O10[1].

9. Simulace lyžařských vleků 1

Problém

U svahu je stanice s N různými lyžařskými vleky (např. N = 4). Každý lyžař, který přijede do stanice, si vybere náhodně jeden z vleků a zařadí se do fronty na konec. Na svahu lyžují také instruktoři až se Z žáky. Ti se po příjezdu zařadí na začátek fronty (předběhnou ostatní). Za jednu časovou jednotku odveze vlek jednoho lyžaře. Simulujte průběh čekání na lyžařský vlek pro M lyžařů (např. M = 500).

Implementace

- Čísla N, M, Z deklarujte jako globální konstanty.
- Každý vlek reprezentujte jednosměrným seznamem implementovaným pomocí ukazatelů. Nepoužívejte třídy (seznam bude reprezentován jako struktura).
- N vleků reprezentujte polem, prvky pole budou seznamy.
- Prvek seznamu (lyžaře) implementujte jako strukturu, která bude obsahovat identifikaci lyžaře (číslo typu int), vlek, na který nasedne (číslo typu int) a potřebu zařazení mimo pořadí - instruktor nebo žák (hodnota typu bool)
- Průběh čekání na vlek reprezentujte cyklem while, který bude ukončen, pokud budou všechny seznamy prázdné. V každém kroku cyklu (jedné časové jednotce) bude všemi vlekům (seznamům), pokud nebudou prázdné, odebrán jeden lyžař. Pokud celkový počet požadavků vygenerovaných během simulace nedosáhl M, bude náhodně vygenerován jeden normální lyžař a jedna skupina s instruktorem (nejméně 2 lyžaři, nejvýše Z). Nesmí být celkově vygenerováno více než M lyžařů včetně instruktorů a žáků. Identifikátorem lyžaře je pořadí, ve kterém byl vygenerován (první lyžař bude mít identifikátor 1, poslední M). Dále se náhodně vygeneruje vlek, u kterého bude čekat (0 až N - 1), a pro instruktory a žáky se náhodně vygeneruje potřeba zařazení mimo pořadí. Poté se nově vygenerovaní lyžaři zařadí správně do seznamů reprezentujících vleky (instruktoři a žáci na začátek, jinak na konec).

Na výstupu zobrazte odděleny mezerou tyto operace:

- Vygenerování nového lyžaře ve formě G#1[S#2][#3], kde #1 je identifikace lyžaře, #2 je vlek a #3 je potřeba zařazení mimo pořadí. Normální desátý lyžař u druhého vleku bude vytisknut ve tvaru G10[S1][0] (mimo pořadí G10[S1][1]).
- Vložení lyžaře do seznamu ve formě V#1[S#2][#3], kde #1 je identifikace lyžaře, #2 je požadovaný vlek a #3 je potřeba mimo pořadí. Pokud vložíme desátého lyžaře do seznamu pro druhý vlek v normálním pořadí, bude na výstupu V10[S1][0].
- Nasednutí lyžaře na vlek (odebrání ze seznamu) ve formě O#1[S#2], kde #1 je identifikace lyžaře a #2 vlek. Pokud nasedne desátý lyžař na druhý vlek, bude na výstupu O10[S1].