Autópálya forgalma

Feladat:

Egy objektummodell készítése az autópálya forgalmának modellezésére. Egy L cellára osztott autópályán N autó van. Egy cellában csak egy autó tartózkodhat egyszerre, így L-N cella üres. Minden autónak van egy egész értékű sebessége. A szimulációt ciklusonként végezzük. Minden ciklusban minden autóra elvégezzük a szükséges műveleteket.

Az osztályok és az öröklődések nagy vonalakban:

6 fő osztály lenne:

- Jármű osztály, ami a járműk közös tulajdonságit tárolná
- Autó osztály, ami az autó viselkedését tartalmazná
- Illetve egy kamion osztály, ami a kamion plusz tulajdonságát és viselkedését tárolná
- Autópálya osztály, ami tartalmazná a jármű objektumok tömbjét egy heterogén kollekcióban és további tulajdonságait az autópályának
- Továbbá egy autópályát generáló osztály, ami közvetlen leszármazottja lenne az autópálya osztálynak és ő lenne felelős a véletlenszerű autók elhelyezéséért és inicializásáért
- Végül pedig egy szimulátor osztály lenne, ami a megadott paraméterek alapján lefuttatná a szimulációt majd a ciklusok száma szerint elvégezné a megfelelő műveleteket az autópálya objektumon.

Bemenetek:

A felhasználó feladata a szimulátor objektum létrehozásánál megadni a következő paramétereket:

- Mekkora legyen az autópálya mérete
- Mennyi autó és kamion helyezkedjen el az autópályán
- Az autó sebességének maximális értékét 1 és 10 közötti szám (alapértelmezetten 5)
- A vezetők figyelmetlenségének az állandóját, ami egy 0 és 1 közötti szám (alapértelmezetten 0.15)

Különbség a kamion és az autó között:

A kamion feleannyi sebességel tud menni maximum mint az autó (ha tört akkor az alsó egész rész). Továbbá nem tud előzni az autókkal ellentétben. Ráadásul 2 mezőnyi helyet foglal az úton.

Minden ciklusban elvégzendő műveletek:

1. Ha egy jármű sebessége még nem érte el a maximumot (x), akkor a sebességét eggyel megnöveljük.
2. Kamion vagy autó függvényében: Ha egy autó előtt levő üres cellák száma (az előtte levő autóig) kisebb, mint a sebessége és a sebességével megeggyező cellával előrébb levő helyen nem tartózkodik másik autó, akkor megelőzheti az autót/autókat, azonban ha mégis tartózkodik akkor lecsökkentjük a sebességét annyival, hogy beálljon egy másik autó mögé, úgy hogy a sebessége a lehető legnagyobb maradjon. Kamion esetén ha a sebessége nagyobb, mint az előtte lévő járműig levő mezők, akkor a sebességét a köztük lévő mezők számára csökkentjük.
3. Egy adott p valószínűséggel csökkentjük a mozgó járműk sebességet eggyel. (Vezetők figyelmetlensége).
4. Minden járműt előremozgatunk annyi cellával, amennyi a sebessége.

Kimenet:

A program gyszerű karakteres kimenetet feltételezve "rajzolja ki" az autópálya állapotát egy-egy szimulációs ciklus után. A felhasználó választhat, hogy fájlba vagy pedig konzolra írassa ki az autópálya állapotát minden ciklus után.

Megjegyzés:

A program tesztelni fogja, hogy az járműk (kamionok*2 + autók) száma kisebb legyen, mint az autópálya mérete, a figyelmetlenség állandóját valamint az járműk sebességét, azaz a felhasználók által megadott adatok helyességét. Végül pedig demonstrálni fogja a működést egy külön modulként fordított tesztprogrammal! A megoldáshoz nem fog használni STL tárolót! A program képes lesz visszatölteni az autópálya helyzetét egy fájlból. Az autópálya objektumon végre lehet majd hajtani az értékadást és lesz másoló konstruktra.


AutoPalyaGeneral
+ AutoPalyaGeneral(hossz: int, maxseb: int, all: float, kamiondb: int, autodb: int)
+ GetAutoPalya(): AutoPalya&
+ KezdoAllapotGeneralas(): void
Szimulacio
- ciklusszam: int
+ Szimulacio(ciklusszam: int = 0)
+ SetCiklus(filename: const char*): void
+ Szimulal(ap: AutoPalya&, os: std::ostream&): void
AutoPalya
- jarmuk: Jarmu**
- hossz: int
- maxsebesseg: int
- allando: float
- kamiondb: int
- autodb: int
- osszjarmu: int
+ AutoPalya(hossz: int = 0, maxseb: int = 0, all: float = 0, kamiondb: int = 0, autodb: int = 0)
+ AutoPalya(other: const AutoPalya&)
+ OsszJarmu(): int
+ SetOsszJarmu(oj: int): void
+ MaxSeb(): const int
+ GetHossz(): int
+ GetJarmu(i: int): Jarmu&
+ SetJarmu(j: Jarmu**): void
+ FajlbaIras(os: std::ostream&): void
+ FajlbolToltes(filename: const char*): AutoPalya
+ DeleteJarmu(J: Jarmu&): void
+ RandomPozi(): int
+ Foglalte(pozi: int): bool
+ keres(a: int): Jarmu&
+ operator=(uj: const AutoPalya&): AutoPalya&
Kamion
- pozi2: int
+ Kamion(sebesseg: int, poz1: int, poz2: int, ap: AutoPalya&)
+ SetPozicio(poz: int): void
+ GetPozicio(poz1: int&, poz2: int&): void
+ kiir(a: int&, os: std::ostream&): void
+ megy(): void
+ clone(): Jarmu*
Auto
+ Auto(sebesseg: int, pozi: int, ap: AutoPalya&)
+ megy(): void
+ kiir(a: int&, os: std::ostream&): void
+ clone(): Jarmu*
Jarmu
- sebesseg: int
- pozi: int
- meret: int
- autop: AutoPalya&
+ Jarmu(sebesseg: int, pozi: int, meret: int, ap: AutoPalya&)
+ GetSebesseg(): int
+ SetSebesseg(a: int): void
+ GetAP(): AutoPalya&
+ SetPozicio(poz: int): void
+ GetPozicio(poz1: int&, poz2: int&): void
+ kiir(a: int&, os: std::ostream&): void
+ megy(): void
+ clone(): Jarmu*

A Kamion osztály egy adott típusú járművet reprezentál. Az osztály a Jarmu osztályból származik és további specifikus funkcionalitást és attribútumokat ad a Kamion objektumokhoz.

Attribútumok:

pozi2 (egész szám): A Kamion második pozícióját jelenti az AutoPályán.

Konstruktor:

Kamion(int sebesseg, int poz1, int poz2, AutoPalya& ap): Létrehoz egy Kamion objektumot a megadott sebesség, első és második pozíció, valamint egy AutoPalya referencia alapján. Az osztály a Jarmu osztály konstruktorát is meghívja az örökölt attribútumok inicializálásához.

Metódusok:

void SetPozicio(int poz): Felülírja az ősosztály metódusát a Kamion pozíciójának beállítására. Emellett frissíti a pozi2 attribútumot a megadott pozícióból poz kivonásával.

void GetPozicio(int& poz1, int& poz2): Felülírja az ősosztály metódusát a Kamion pozíciójának lekérdezésére. Visszaadja a Kamion első pozícióját poz1-ben és a második pozícióját poz2-ben.

void kiir(int& a, std::ostream& os): Felülírja az ősosztály metódusát a Kamion kiírására. Az a értékét növeli, majd kiírja "KK" stringet az os ostream objektumba.

void megy(): Felülírja az ősosztály metódusát a Kamion mozgatására. A Kamion sebességét véletlenszerűen csökkenti a figyelmetlenségi tényező figyelembe vételével, ha a sebesség nagyobb, mint 1. Ha a Kamion sebessége kisebb, mint az AutoPalya maximális sebességének a fele, akkor növeli a sebességet. A Kamion új pozícióját a jelenlegi pozícióhoz képest kiszámítja, amíg a sebesség és az új pozíció alapján a Kamion nem lép túl az AutoPalya hosszán vagy nem talál foglalt pozíciót. Ha a Kamion túllép az AutoPalya hosszán, akkor törli magát az AutoPalya-ról.

Jarmu* clone(): Felülírja az ősosztály metódusát a Kamion klónozására. Létrehoz és visszaad egy új Kamion objektumot, amely az aktuális Kamion objektumot másolja.

~Kamion(): Virtuális destruktor, amely felülírja az ősosztály destruktorát. A Kamion osztály destruktora alapértelmezetten működik.

Az Auto osztály egy adott típusú járművet reprezentál. Az osztály a Jarmu osztályból származik és további specifikus funkcionalitást és attribútumokat ad az Auto objektumokhoz.

Metódusok:

Auto(int sebesseg, int pozi, AutoPalya& ap): Konstruktor, amely létrehoz egy Auto objektumot a megadott sebesség, pozíció , valamint egy AutoPalya referencia alapján. Az osztály a Jarmu osztály konstruktorát is meghívja az örökölt attribútumok inicializálásához.

void megy(): Felülírja az ősosztály metódusát az Auto mozgatására. Az Auto sebességét véletlenszerűen csökkenti az AutoPalya allando attribútumának figyelembe vételével, ha a sebesség nagyobb, mint 1. Ha az Auto sebessége kisebb, mint az AutoPalya maximális sebessége, akkor növeli a sebességet. Az új pozíciót az aktuális pozícióhoz és a sebességhez hozzáadva számítja ki. Ha az új pozíció meghaladja az AutoPalya hosszát, akkor törli az Auto-t az AutoPalya-ról. Ha az új pozíció foglalt, akkor visszafelé haladva keres egy szabad pozíciót, amíg el nem éri az aktuális pozíciót, vagy nem talál szabad pozíciót. Ha talál szabad pozíciót, akkor beállítja az Auto új pozícióját.

void kiir(int& a, std::ostream& os): Felülírja az ősosztály metódusát az Auto kiírására. Kiírja az "A" karaktert az os ostream objektumba.

Jarmu* clone(): Felülírja az ősosztály metódusát az Auto klónozására. Létrehoz és visszaad egy új Auto objektumot, amely az aktuális Auto objektumot másolja.

~Auto(): Virtuális destruktor, amely felülírja az ősosztály destruktorát. Az Auto osztály destruktora alapértelmezetten működik.

Az Jarmu osztály egy absztrakt alaposztály, amely reprezentálja a járművek általános tulajdonságait és funkcionalitását. Az osztályból leszármaztatott konkrét járműosztályoknak implementálniuk kell az absztrakt metódusokat.

Attribútumok:

int sebesseg: A jármű sebességét tároló változó.

int pozi: A jármű aktuális pozícióját tároló változó.

int meret: A jármű méretét tároló változó.

AutoPalya& autop: Az AutoPalya objektumra mutató referencia, amelyen a jármű található.

Metódusok:

Jarmu(int sebesseg, int pozi, int meret, AutoPalya& ap): Konstruktor, amely létrehoz egy Jarmu objektumot a megadott sebesség, pozíció, méret, egy AutoPalya referencia alapján.

int GetSebesseg(): Visszaadja a jármű sebességét.

void SetSebesseg(int a): Beállítja a jármű sebességét a megadott értékre.

AutoPalya& GetAP(): Visszaadja a járműhöz tartozó AutoPalya objektumra mutató referenciát.

virtual void SetPozicio(int poz): Virtuális metódus, amely beállítja a jármű pozícióját a megadott értékre. Az alapértelmezett implementáció beállítja az pozi attribútumot.

virtual void GetPozicio(int& poz1, int& poz2): Virtuális metódus, amely visszaadja a jármű pozícióját. Az aktuális pozíciót az poz1 változóba helyezi, és a második pozíciót (-1) az poz2 változóba helyezi.

virtual void kiir(int& a, std::ostream& os) = 0: Tiszta virtuális metódus, amelyet a leszármaztatott osztályoknak implementálniuk kell. A metódus kiírja a járműt az os ostream objektumba.

virtual void megy() = 0: Tiszta virtuális metódus, amelyet a leszármaztatott osztályoknak implementálniuk kell. A metódus felelős a jármű mozgatásáért.

virtual Jarmu* clone() = 0: Tiszta virtuális metódus, amelyet a leszármaztatott osztályoknak implementálniuk kell. A metódus klónozza a járműt, és visszaadja az új járműre mutató pointert.

virtual ~Jarmu(): Virtuális destruktor, amely meghatározza az ősosztály destruktorát. Az alapértelmezett implementáció működik.

Az AutoPalya osztály reprezentálja az autópályát, amelyen a járművek közlekednek. Az osztály felelős a járművek tárolásáért, kezeléséért és az autópálya műveleteinek végrehajtásáért.

Attribútumok:

Jarmu** jarmuk: Járművek tömbjére mutató pointer.

int hossz: Az autópálya hossza.

const int maxsebesseg: Az autópálya maximális sebessége.

const float allando: Az autópálya állandó értéke.

int kamiondb: A kamionok száma az autópályán.

int autodb: Az autók száma az autópályán.

int osszjarmu: Az autópályán található járművek összessége.

Metódusok:

AutoPalya(int hossz=0, int maxseb=0, float all=0, int kamiondb=0, int autodb=0): Konstruktor, amely létrehoz egy AutoPalya objektumot a megadott paraméterek alapján. Inicializálja az attribútumokat és létrehoz egy üres járműtömböt.

AutoPalya(const AutoPalya& other): Másoló konstruktor, amely lemásolja az AutoPalya objektumot.

int OsszJarmu(): Visszaadja az autópályán található járművek összes számát.

void SetOsszJarmu(int oj): Beállítja az autópályán található járművek összes számát.

const int MaxSeb(): Visszaadja az autópálya maximális sebességét.

const int GetAllando(): Visszaadja az autópálya figyelmetlenségi tényezőjének értékét.

int GetHossz(): Visszaadja az autópálya hosszát.

Jarmu& GetJarmu(int i): Visszaadja a megadott indexű járműre mutató referenciát az autópályán.

void SetJarmu(Jarmu** j): Beállítja az autópályán található járművek tömbjét a megadott tömbre (j).

void FajlbaIras(std::ostream& os): Kiírja az autópálya állapotát az os ostream objektumba.

AutoPalya FajlbolToltes(const char* filename): Betölti az autópálya állapotát egy fájlból (filename) és létrehoz egy AutoPalya objektumot az adatok alapján.

void DeleteJarmu(Jarmu& J): Törli a megadott járművet az autópályáról.

int RandomPozi(): Véletlenszerűen választ egy szabad pozíciót az autópályán.

bool Foglalte(int poz): Ellenőrzi, hogy a megadott pozíció foglalt-e az autópályán.

Jarmu& keres(int a): Megkeresi és visszaadja a megadott pozícióhoz tartozó járműre mutató referenciát.

AutoPalya& operator=(const AutoPalya& uj): Értékadás operátor túlterhelése, amely másolja az egyik AutoPalya objektumot a másikba.

~AutoPalya(): Destruktor, amely felszabadítja az autópályán található járművek memóriáját.

Az AutoPalyaGeneral osztály az AutoPalya osztály leszármazottja, és kibővíti azt a kezdeti állapot generálásával. Az osztály felelős az autópálya kezdeti állapotának előállításáért.

Metódusok:

AutoPalyaGeneral(int hossz, int maxseb, float all, int kamiondb, int autodb): Konstruktor, amely létrehoz egy AutoPalyaGeneral objektumot a megadott paraméterek alapján. A konstruktor inicializálja az AutoPalya ősosztály attribútumait és meghívja a KezdoAllapotGeneralas() függvényt a kezdeti állapot generálásához.

AutoPalya& GetAutoPalya(): Visszaadja az AutoPalyaGeneral objektumot, amely az AutoPalya osztályra vonatkozó műveletek végrehajtására szolgál.

void KezdoAllapotGeneralas(): Generálja az autópálya kezdeti állapotát. A függvény először kamionokat generál az autópályán, majd autókat. A kamionokhoz véletlenszerű sebességet és pozíciót rendel, és a kamion helyzetétől függően létrehozza a megfelelő járművet. Az autókhoz is véletlenszerű sebességet és pozíciót rendel. A generált járművek hozzá lesznek adva az autópályához. A függvény figyelembe veszi hogy nehogy két jármű akár a kamion egy része és egy másik autó vagy kamion és kamion sehogy ne érintkezzen, sőtt azt is figyelembe veszi hogy ne lógjon le egy kamion se az út valamelyik végéről.

Az Szimulacio osztály felelős az autópálya szimulációjának végrehajtásáért. A szimuláció során a járművek a megfelelő sebességgel haladnak előre a pályán, és a pálya állapota minden ciklusban frissül.

Metódusok:

Szimulacio(int ciklusszam = 0): Konstruktor, amely létrehoz egy Szimulacio objektumot a megadott ciklusszámmal. Alapértelmezetten a ciklusszám 0.

void SetCiklus(const char* filename): Beállítja a ciklusszámot a megadott fájlból. A fájlban a ciklusszám számnak kell lennie. Ha a fájl nem található, vagy a fájl nem tartalmaz számot, kivételt dob.

void Szimulal(AutoPalya& ap, std::ostream& os): Végrehajtja az autópálya szimulációját. A függvény végigmegy a ciklusszámnak megfelelő számú cikluson. Minden ciklusban végigmegy az autópályán a járművek aktuális pozícióinak csökkenő sorrendjében. Ha egy pozíción van jármű, a jármű haladását végrehajtja. Az autópálya állapotát minden ciklus után kiírja a megadott kimeneti adatfolyamra.