Симулатор на летище

изготвил: Георги Павлов, ФН: 80950, 7 група

Проектът използва само стандартната библиотека на C++ като единственият файл, който е възможно да прави проблем на по-стари компилатори, е <random>, който е включен в C++11 стандарта. Примерите за тестване на проекта са заредени в main.

Подходих към проекта като първо разделих логиката на летището на части. Реших с най-голям приоритет да е прибирането от пистите на самолетите, кацнали предния ход, защото това позволява на симулацията да обработи възможно най-много самолети на ход.

При намерен самолет за прибиране се блокират всички предни писти. Само в един случай може самолет да не се прибере в хода си за извозване – ако предишния ход всички самолети в опашката за кацане са се приземили (или е нямало самолети) и ако е нямало самолети за извозване от пистите. Тогава самолетите за излитане получават ‘зелена светлина’ и заемат най-подходящите писти, блокирайки прибирането на кацналите самолети в следващия ход.

През второто действие самолетите в опашката за кацане се обработват. Избрал съм те винаги да кацат на последната свободна писта, защото според мен е най-оптимално.

Третата стъпка е да се махнах самолетите, които са на пистата и чакат да излетят.

В последната стъпка се обработват самолетите в опашката за излитане. При условие, че няма повече самолети в опашката за кацане, самолет може да заходи към писта за излитане, като винаги избира последната свободна писта, до която може да се стигне.

Статистиката се събира през всяко действие и се пази до края на симулацията.

Структури

Реализирал съм проекта си чрез един основен клас Летище и няколко помощни структури. Използвах структури, за да реализирам самолетите, пистите и статистиката понеже нямат функционалност – основното им предназначение е да групират данни от няколко различни типа.

= class Airport - Реализира цялата логика на симулацията.
Член-данни:
- опашки със самолети за кацане и излитане
- вектор от писти
- статистика за дадената симулация
- време за симулацията, текущ ход и брой самолети на ход.

Публичните му функции са:
- **Airport(const int, const int , const int)** - конструктор, приемащ време, брой писти и брой самолети на ход.
- **simulation()** – изпълнява симулацията като върти цикъл с ходовете. След приключване на цикъла извежда статистика на стандартния изход.

Вътрешни функции:
- **spawnPlanes()** – пълни опашките за кацане и излитане с определения брой самолети като използва рандом генератор за избор на опашка.
- **organization()** – групира 4-те действия на ход.
- **firstAct(); secondAct(); thirdAct(); lastAct(); lastActHelper(int&) –** реализират описаната по горе логика на летището, помощната функция предодвратява повтаряне на ход.
- **runwayReset() –** променя булевата стойност на пистите, които са се освободили този ход.
- **finalizeStats() –** опразва опашките след края на симулацията и брой самолетите към статистиката.
- **printStats(); printRunways(); print(); -** извеждат на конзолата всички данни, акумулирани по време на симулацията.

= struct Airplane – създава обекти от тип самолет чрез конструктор с единствена член-данна - ход, през който са създанени.

= **struct Runway -** създава обекти от тип писта

- bool occupied – булева променлива, казваща дали пистата е заета или Свободна през даден ход.

- PlaneStatus plane – наименова стойност, оказваща какъв самолет заема дадената писта (NoPlane, TakingOff, Landed).

- int occupiedTime – време, през което дадената писта е била заета .

= struct Statistics – събира статистиката за дадената симулация и накрая на хода я изкарва на екрана.

Член-данни:

- брой самолети, които са в опашките за кацане, излитане, и такива, на които им е направен първоначален отказ.
- вектори с данни за брой кацащи/излитащи самолети на ход и брой ходове, които всеки самолет е трябвало да чака.

Функции:

- doMath(const vector<int>&, float&, float&) – извършва необходимите аритметични и акумулативни действия, за да подготвни данните от даден вектор за извеждане.

- print() - извежда цялата статистика за дадена симулация.

Бъдещи идеи

- Добавяне на дестинация на всеки самолет.
- Разпределяне на пистите в различни посоки като при реално летище (свързано с дестинациите).
- Реалистична логика за симулация: възможно разминаване на самолети, по-сложни приоритети на самолетите (спрямо дестинация, брой пътници и други..).
- Рандом генератор за метеорологично време.
- Статискитика на български.