OPGAVE – DUNGEON ESCAPE

GRUNDLÆGGENDE PROGRAMMERING - AFSLUTTENDE

Indholdsfortegnelse

Beskrivelse

Spillet "Dungeon Escape" er et tekstbaseret eventyrspil, hvor spilleren skal navigere gennem en labyrint af rum for at finde en udgang. Spilleren står overfor forskellige udfordringer, såsom at finde nøgler, undgå fælder og vælge den rigtige vej i labyrinten.

Spillet består af en række "rum" (repræsenteret i koden som 2D-arrays og lister), som spilleren kan bevæge sig mellem. I nogle rum er der nøgler, i andre rum kan der være fælder, og nogle gange vil en sti ende blindt. Spilleren vinder ved at finde nøglen og udgangen.

Målgrupper

Spillet er designet til begyndere i C#, der lærer konsolprogrammering og ønsker at arbejde med kontrolstrukturer, variabler, arrays, metoder og debugging.

Krav og funktioner

1. **Navigation i labyrinten**:
    - Spilleren kan bevæge sig op, ned, venstre eller højre ved at skrive en kommando som "op", "ned", "venstre" eller "højre".
    - Et simpelt 2D-array kan repræsentere labyrinten, hvor spilleren starter i ét felt og skal navigere til andre felter.
2. **Datatyper og kontrolstrukturer**:
    - Arrays bruges til at oprette labyrinten.
    - `if` og `switch` statements bruges til at håndtere spillerens bevægelser og de forskellige handlinger, der udføres baseret på spillerens position.
3. **Metoder**:
    - Spillet benytter flere simple metoder, fx:
        - En metode til at opdatere spillerens position baseret på input.
        - En metode til at vise spilleren den aktuelle status (f.eks. "Du har fundet en nøgle!" eller "Pas på, en fælde!").
        - En metode til at genstarte spillet, hvis spilleren går i en fælde.
        - En metode til at tjekke, om spilleren har nået udgangen og vundet spillet.

Teknisk krav

- **Kontrolstrukturer**: `if`, `else`, `switch`, `while` og `for` loops til at håndtere spillets flow.
- **Datatyper**: Brug af `int`, `bool`, `string` samt `char` (til at repræsentere labyrintens felter).
- **Arrays og lister**: Et 2D-array til labyrintens struktur.
- **Metoder**: Enkle metoder til at håndtere spilfunktioner og strukturere koden.
- **Variable og konstanter**: Konstanter til symboler for de forskellige rum (nøgle, fælde, udgang), samt variabler til spillerens position og tilstand.

Eksempel på spildiagram

- Et simpelt flowdiagram viser spillets flow, f.eks.:
  - Start → Bevæg spiller → Tjek for hændelser (nøgle, fælde, udgang) → Opdater status eller genstart.

Kodekommentarer

- Kommentarer beskriver, hvad metoderne gør, og hvad variablerne repræsenterer.

Versionsstyring og debugging

1. **Versionsstyring**: Versionsstyring bruges til at gemme fremskridt og tillader, at tidligere versioner af koden kan tilgås.
2. **Debugging**: Brug debugging-værktøjer til at identificere fejl, f.eks. fejl i bevægelseslogikken.

Eksempel på spilflow

1. **Spilstart**: Spilleren præsenteres for spillets regler og starter i en bestemt position i labyrinten.
2. **Bevægelse**: Spilleren taster "op", "ned", "venstre" eller "højre" for at bevæge sig mellem rum.
3. **Hændelser**:
   - Spilleren finder en nøgle: Spilleren får besked om, at de har fundet nøglen.
   - Spilleren går i en fælde: Spillet slutter, og spilleren kan starte forfra.
   - Spilleren når udgangen med nøglen: Spilleren vinder, og spillet slutter.
4. **Genstart**: Hvis spilleren går i en fælde, gives der en besked, og spillet kan genstartes.

Ekstra opgave

Spilfunktioner:

- Spilleren starter uden nøglen og skal finde den for at kunne åbne udgangen.
- Der kan være felter med fælder, hvor spilleren "taber" spillet og må starte forfra.
- Ved at bevæge sig ind i et felt, får spilleren besked om, hvad feltet indeholder (f.eks. en nøgle, en fælde eller ingenting).