MÄSTARPROV 8

Tillämpning av modul 1-7

Game of Life

Introduktion

Game of Life uppfanns 1970 av John Conway. Det är en enkel simulering av en slags artificiell livsform. Spelplanen är en tvådimensionell matris, där varje cell antingen kan vara tom eller bebos av exakt en individ. I varje generation kan ett antal individer födas eller dö. En individs öde avgörs av hur många av dess grannceller som är befolkade:

- En individ som har 0 eller 1 levande granne dör i nästa generation.
- En individ som har 2 eller 3 levande grannar överlever in i nästa generation.
- En individ som har 4 eller fler levande grannar dör i nästa generation.
- I en tom cell som har exakt 3 levande grannar föds en ny individ i nästa generation.

De flesta celler har åtta grannceller, men celler på kanten har färre. Till exempel har en hörncell bara tre grannceller. I denna obligatoriska uppgift ska en enkel implementation av Conway's Game of Life tas fram. I figur 1 nedan ser du en simulering av en 10x10-värld före och efter ett generationsskifte enligt reglerna ovan (X = levande, . = död). Observera att uppdateringen av cellerna i världen sker synkront, vilket innebär att man först tar reda på vilket tillstånd alla celler ska ha i nästa generation och därefter uppdaterar man tillståndet för alla celler.

Uppgiften ska lösas enskilt.

```
. . . . . . . . . .     . . . . . . . . . .
. . . . . . . . . .     . . . . . . . . . .
. . . . . . . . . .     . . . . . . . . . .
. . . X . X . . . .     . . . . X . . . . .
. . . . X . . . . .     . . . X X X . . . .
. . . . X . . . . .     . . . X X X . . . .
. . . X . X . . . .     . . . . X . . . . .
. . . . . . . . . .     . . . . . . . . . .
. . . . . . . . . .     . . . . . . . . . .
. . . . . . . . . .     . . . . . . . . . .
```

Figur 1. Visar ett steg när reglerna för Conway's Game of Life appliceras på en 10x10-värld.

Syfte
Syftet med uppgiften är att testa att du kan applicera den kunskap du bemästrat i Modul 1-7 och sätta samman ett komplett program i enlighet med givna instruktioner.

Uppgift
Din uppgift är att implementera en enkel version av Conway's Game of Life. Till din hjälp har du en källkodsfil med viss kod given. Källkodsfilen kompilerar utan varning. Den givna koden hanterar delar av initialiseringen av världen. Funktionerna `main` och `load_random` saknar funktionalitet som ska implementeras. Övriga funktioner (`init_field`, `load_glider`, `load_semaphore` och `load_custom`) är fullständiga och ska inte ändras.

För att se hur ditt program ska fungera finns det ett körbart komplett program som du får undersöka tillsammans med den givna koden (att förstå given kod och det körbara programmet är en del av uppgiften). Den givna källkodsfilen och det körbara programmet nås via länkar i beskrivningen till detta mästarprov.

Algoritm
Följande algoritm ska användas i programmet:

1. Initialisering av världen
2. Så länge användaren inte valt att avsluta
   2.1 Skriv ut världen
   2.2 Skriv ut menyn
   2.3 Läs in användarens val
   2.4 Om nytt steg
      2.4.1 Uppdatera världen en generation

Krav på implementationen
Här nedan finns det ett antal krav som programmet ska uppfylla:

- Programmet ska använda sig av funktioner (dvs fler än de ovan nämnda) och den givna egendefinierade datastrukturen `cell`.
- All utskrift på skärm ska se exakt ut som i det tillhandahållna körbara programmet.
- Funktionerna `init_field`, `load_glider`, `load_semaphore` och `load_custom` är fullständiga och ska inte ändras.
- I din inlämnade lösning ska världen ha 20 x 20 celler, men det ska vara lätt att ändra.
- Källkodsfilen ska heta `mp8.c`.
- I den beskrivande texten i filen ska du skriva ditt namn och din cs-användare.
- Programmet ska kunna kompileras med kompilatorn `gcc` med flaggorna `-Wall` och `-std=c99`.

Programmet behöver bara hantera korrekt indata. Ingen validering av indata krävs, förutom att det ska gå att använda programmet på det sätt som illustreras i det tillhandahållna körbara programmet. Om du väljer att validera data ska valideringen vara korrekt.

Redovisning

Uppgiften redovisas genom att lämna in källkodsfilen via webbgränssnittet i Labres, se länk i beskrivningen till detta mästarprov. Din inlämnade fil kompileras och testkörs. Du kan lämna in flera gånger.

När du vill få din inlämning bedömd, gör mästarprovet för denna modul (MP8 - Mästarprov modul 8: Önskan om bedömning). I det mästarprovet ska du bara skriva din cs-användare och lämna in provet.

Tips

Här följer några tips som kan hjälpa till:

- Börja med att förstå uppgiften: Vad ska göras? Vilka krav finns det? Vad ska lämnas in?
- Skapa en fullständig förståelse för den givna algoritmen. Testkör den mot det tillhandahållna körbara programmet.
- Skapa källkodsfilen och skriv in algoritmer och beskrivande text.
- Utveckla programmet stegvis! Ett förslag är att skriva huvudfunktionen först och sedan implementera funktionerna i en ordning som är till hjälp vid utvecklandet av koden.
- Försök att testa varje steg utförligt innan nästa steg tas och kom ihåg att testa att tidigare steg fortfarande fungerar när nästa steg testas.
- När du vill ha hjälp av handledare, var beredd på att först förklara vad du vill uppnå/göra och hur du tänkt göra det i koden.

Kvalitetskriterier

Den inlämnade lösningen kommer att bedömas enligt följande kriterier:

Kriterium	**Godkänd**	**Godkänd med anmärkning**	**Ofullständig**
Kompilering	Utan varning	Mindre allvarlig	Allvarlig
Testkörning/ Korrekthet	Utan fel	Mindre fel	Felaktig output Räknar fel
Kommentarer	Informativa Lagom omfattning Konsekvent språk	För lite eller för mycket Saknar viss information Otydliga Upprepar koden Ej konsekvent språk	Missvisande Saknar nödvändig information Saknar beskrivande text för program och/eller funktioner
Indentering	Korrekt	Något fel	Många fel Visar tecken på att ej förstå varför och hur man indenterar
Variabel-deklaration	Konsekvent Olika datatyper på olika rader Väl valda datatyper används	Blandning av deklaration i början och vid behov Mindre brister i val av datatyper `const`-deklaration saknas på konstant värde	Felaktig Olika datatyper på samma rad Visar flera tecken på att ej förstå vilka datatyper som är lämpliga att använda
Namngivning (variabler, funktioner, arrayer, parametrar)	Bra namn Konsekvent namngivning	Mindre bra namn Ej konsekvent namngivning	Missvisande namn Olämpliga namn
Valstrukturer	Bra struktur Bra val av villkor	Villkorsoperatorn används Mindre bra val av villkor	Allt på en rad Felaktiga villkor
Loop-strukturer	Bra val av loop-konstruktion Bra val av villkor	Mindre bra val av loop-konstruktion Mindre bra val av villkor	Felaktiga villkor `break` och `continue` används i onödan
Funktioner	Programmet är uppdelat i lämpliga funktioner med bra parametrar och returvärden	Mindre problem med val av parametrar/returvärden Någon funktion utför för många orelaterade uppgifter	Funktioner saknas Många funktioner utför för många orelaterade uppgifter Felaktiga val av parametrar/returvärden
Arrayer	Arrayer används på ett korrekt sätt	Mindre bra användning av array	Indexeringsfel Indexerar utanför array Felaktig användning av array
Egen-definierade datastrukturer	Används korrekt Används på ett konsekvent sätt	Någon miss i användandet	Saknas Används felaktigt Används inkonsekvent
Program-struktur	Måsvingar placeras konsekvent Måsvingar används till alla val- och loop-konstruktioner Funktionsdeklarationer och funktionsdefinitioner är konsekvent placerade	Fall av inkonsekvent placering av måsvingar Inkonsekvent placering av funktionsdeklarationer och funktionsdefinitioner Anropar `srand()` på fel ställe	Måsvingar saknas eller placeras hur som helst Onödigt många return-satser i en funktion exit() används Annat än 0 returneras i main Globala variabler Kommandona `goto` och/eller `longjmp` används
Algoritm	Följer given algoritm	Mindre avvikelse från algoritm	Följer ej algoritm Ändrar given kod som ej ska ändras